Imperial College London

COURSEWORK

IMPERIAL COLLEGE LONDON

DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING

Adaptive Signal Processing and Machine Intelligence

Junyu Yan (CID: 01895990)
junyu.yan20@imperial.ac.uk

Date: April 19, 2021

Contents

1 Spectrum Estimation

1.1 Properties of Power Spectral Density (PSD)

In order to verify the equivalence with two definitions of PSD, we start from the definition 2, that is,

$$\begin{aligned}
P(\omega) &= \lim_{N\to\infty} \mathcal{E}\left\{\frac{1}{N}\left|\sum_{n=0}^{N-1} x(n)e^{-jn\omega}\right|^2\right\} \\
&= \lim_{N\to\infty} \mathcal{E}\left\{\frac{1}{N}\sum_{n=0}^{N-1} x(n)e^{-jn\omega}\sum_{m=0}^{N-1} x^*(m)e^{jm\omega}\right\} \\
&= \lim_{N\to\infty} \frac{1}{N}\sum_{n=0}^{N-1}\sum_{m=0}^{N-1} \mathcal{E}\left\{x(n)x^*(m)\right\} e^{-j(n-m)\omega} \\
&= \lim_{N\to\infty} \frac{1}{N}\sum_{n=0}^{N-1}\sum_{m=0}^{N-1} r(n-m)e^{-j(n-m)\omega} \\
&= \lim_{N\to\infty} \frac{1}{N}\sum_{k=-(N-1)}^{N-1} (N-|k|)r(k)e^{-jk\omega} \\
&= \lim_{N\to\infty} \frac{1}{N}\sum_{k=-(N-1)}^{N-1} Nr(k)e^{-j\omega} - \lim_{N\to\infty}\frac{1}{N}\sum_{k=-(N-1)}^{N-1} |k|\,r(k)e^{-j\omega} \\
&= \sum_{k=-(N-1)}^{N-1} r(k)e^{-j\omega} - \lim_{N\to\infty}\frac{1}{N}\sum_{k=-(N-1)}^{N-1} |k|\,r(k)e^{-j\omega}
\end{aligned} \tag{1}$$

It is assumed that the covariance sequence $r(k)$ decays rapidly, that is,

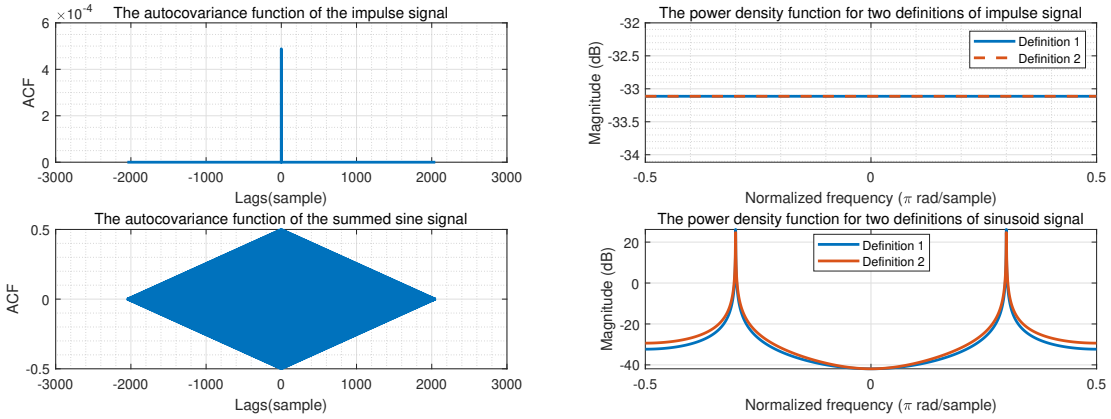
$$\lim_{N\to\infty}\frac{1}{N}\sum_{k=-(N-1)}^{N-1} |k||r(k)| = 0 \tag{2}$$

Therefore, the (1) can be reduced to (3), which is the definition 1 of PSD.

$$P(\omega) = \sum_{k=-\infty}^{\infty} r(k)e^{-j\omega k} \tag{3}$$

The definition 1 represents that the PSD of the signal is the Discrete Fourier Transform of the Autocovariance Function (ACF). While the definition 2 states that the PSD is the distribution of averaged signal power over frequency. Therefore, the equivalence proves that when the covariance sequence decays rapidly (i.e., $r(k)$ is absolutely integrable), the PSD of a discrete time deterministic sequence is equal to the DTFT of its ACF. Moreover, according to the definition 2, the PSD is absolutely positive and real. The definition 1 also satisfies this condition, since the Fourier Transform of the even function ACF is also real and positive.

To simulate the equivalence, we obtain the PSD of impulse signal and sinusoidal signal, respectively. As shown in Figure.1.(a), the covariance sequence of the impulse signal decays significantly rapidly, however; the ACF of the sinusoidal signal reduces slowly. Therefore, as shown in Fig.1.(b), the two definitions of PSD coincide with each other for the impulse signal, but they are different for the sinusoidal signal. Generally, the perfectly periodic signal does not satisfy the mild assumption as listed in (2), but most real-life signals is a combination of noise and periodic signal that satisfy the required condition. Therefore, in practice, the (2) will always hold, leading to the equivalence between two definitions.



(a) ACF (b) PSD

Figure 1: The autocovariance function and PSD for the impulse signal and the sinusoidal signal

1.2 Periodogram-based Methods Applied to Real-World Data

a) The time series data usually has a non-zero mean and a long-period linear trend that will highly influence the analysis of data, and must be removed before further processing. Removing mean eliminates the effect of DC component in the original data. Linearly detrending the data reduces the low-frequency components so that the high-frequency components can be more easily identified. Also, it can remove jitters in the data, which leads to less noise in the signal. Moreover, other preprocessing methods, such as the logarithm combined with the subtraction of mean, performs even better on removing jitters, and can be used to identify the change of periodicity in data. To avoid zero occurring in the logarithm, a small constant (MATLAB `eps` $= 2.2204\times10^{-16}$) is added into the sunspot time series. From the Fig.2, it is clear that removing mean as well as trend centralizes and smooths the original data. Removing mean and trend decreases the PSD spectrum below the normalized frequency 0.01, and it has little influence on the high-frequency components. The logarithm preprocessing method captures more details in the PSD spectrum. For example, the rise at normalized frequency equal to 0.01 is captured, and more harmonics can be identified. Moreover, it can also capture the change of perception of periodicity in data. As shown in Fig.2, the waveform after logarithm and subtraction of mean will experience a strong oscillation if there is less periodicity.

b) The flashing visual stimulus flashing at a fixed integral rate will induce the response known as steady state visual (SSVEP) in electroencephalogram (EEG) signal at the same frequency. As shown in Fig.3, both standard periodogram and averaged periodograms with different window length (i.e., 1s, 5s, 10s) are capable of identifying the peaks corresponding to SSVEP, which are the fundamental frequency at 13Hz with two harmonics at 26Hz and 39Hz, respectively. The peak between 8-10Hz, clearly captured by averaged periodogram, is the alpha-rhythm due to the tiredness of the subject during the recording. Moreover, the largest peak appearing at 50Hz is caused by the power-line interference, and it exceeds the third harmonic at 52Hz.

Compared with the standard periodogram, the averaged periodogram generates more recognizable peaks. It is clear that the averaged periodogram with window length equal to 10s has a smaller variance than the standard periodogram, but the dominant peaks can still be identified. It significantly reduces the effect of noise and highlight the main lobes, which is quite beneficial for the observation of meaningful information from PSD spectrum. For the window length equal to 1s, the spectrum is highly smoothed with clearer trend of PSD. However, it contains less details, such that the peak at 39Hz is hard to recognized. The reason is that the windowed periodogram is equal to the convolution of the PSD with the Fourier Transform of the window, which is a sinc-like function. The main lobe width of the sinc function is inversely proportional to the signal length. Therefore, truncating signal through decreasing the window length comes at an expense of a wider main lobe, which ulti-

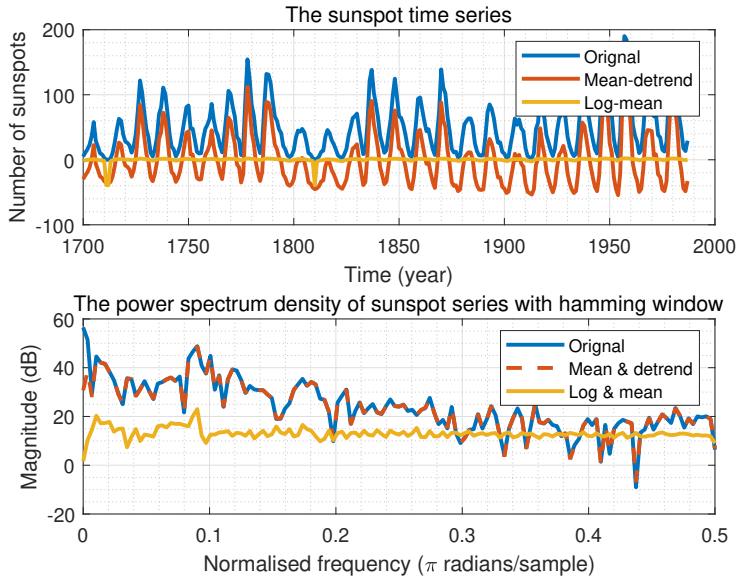


Figure 2: Top: the original and preprocessed sunspot time series. Bottom: the PSD of the sunspot time series with hamming window

mately fuses more nearby spectral peaks and reduces the frequency resolution. In conclusion, there is a trade-off between the variance reduction and the frequency resolution. Therefore, choosing a 5s window is more proper to simultaneously ensure the noise elimination and peaks distinguish-ability.

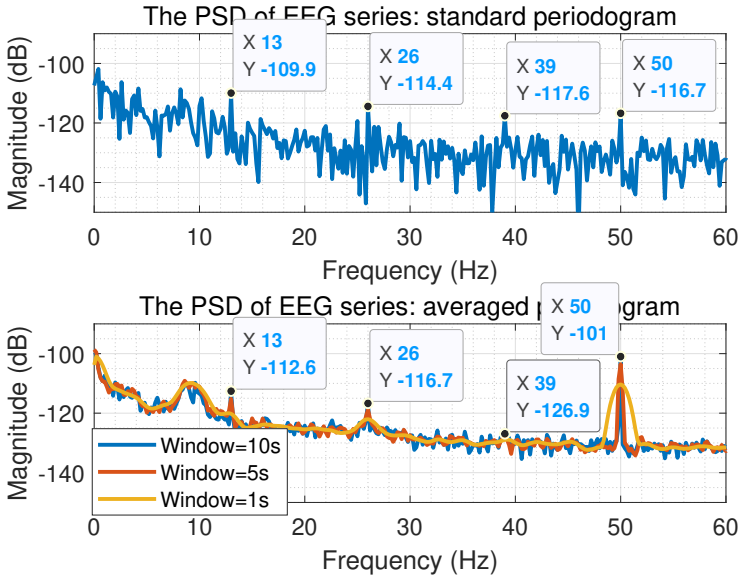


Figure 3: Top: the standard periodogram of EEG series. Bottom: the averaged periodogram of EEG series

1.3 Correlation Estimation

a) Fig.4 and Fig.**??** present the correlogoram and correlation-based PSD estimate of the WGN signal, noisy sinusoidal signal and low-pass filtered WGN signal. The estimate is based on the biased and

unbiased estimators given by,

$$\textbf{Biased} : \hat{\mathrm{r}}(k) = \frac{1}{N} \sum_{n=k+1}^{N} x(n)x^*(n-k) \tag{4}$$

$$\textbf{Unbiased} : \hat{\mathrm{r}}(k) = \frac{1}{N-k} \sum_{n=k+1}^{N} x(n)x^*(n-k) \qquad (0 \le k \le N-1) \tag{5}$$

By comparing the ACF estimates shown in Fig.4, we find that the biased and unbiased ACF are similar for small lags as $|k| \le 100$. However, as k increases, the biased ACF estimate decays to 0, while the unbiased one grows. This can be explained with (4) and (5) that when k is large, the similarity between samples will decreases, leading to smaller sum and smaller unbiased ACF estimate. But for unbiased ACF, the increase in k decreases the denominator in coefficients that will still produce a large value.

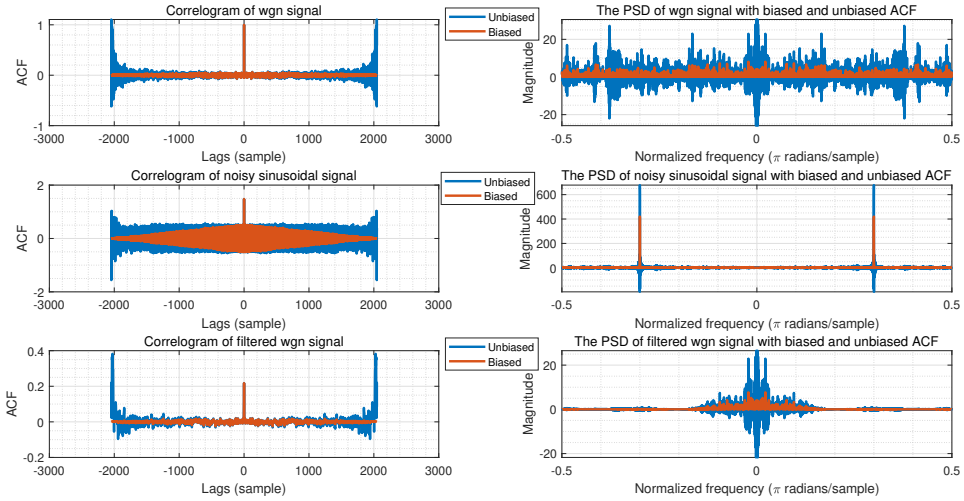


Figure 4: The correlogram and PSD estimate of wgn signal, noisy sinusoidal signal and filtered wgn signal. Left: correlogram. Right: PSD estimate.

Right plots in Fig.4 show that the PSD estimates with biased ACF for three kinds of signals agree with the expectation as flat for WGN signal, impulse for sinusoidal signal and rectangular for low-pass WGN. However, for unbiased PSD estimate, it is erratic and results in negative values for large k. This is due to the fact that for the unbiased estimator, the expect value of the periodogram of a finite length signal is equal to the convolution between the true PSD and the Fourier Transform of the rectangular window. Since $\mathcal{F}\{w_R\} = sin(\frac{\omega N}{2})/sin(\frac{\omega}{2})$, which is a *sinc* function, the PSD estimated by unbiased method can take negative values. But for the biased ACF estimator, the rectangular window is replaced by the Bartlett's window, where the Fourier transform is given by $sin^2(\frac{\omega N}{2})/sin^2(\frac{\omega}{2})$. Therefore, the biased estimator preserve the positiveness of PSD. Moreover, it can also be explained from the perspective of positive semi-definiteness. For unbiasd estimator, the ACF is not forced to decay to 0 for a large k, leading the PSD to be not positive semi-definite, but for the biased one, the convergence to 0 is guaranteed. For these reasons, the biased estimator is more widely used, since it has low variance, guarantees the positiveness of PSD, and is asymptotically unbiased for a large N.

b) We generate the PSD estimate based on the biased ACF estimator of 100 realisations for the signal composed of sinusoids and unit power WGN noise (AWGN singal) given by,

$$x(n) = 0.3\sin(2\pi 0.2n) + 0.6\sin(2\pi 0.25n) + w(n) \tag{6}$$

Fig.5 shows that the PSD estimate can successfully detect the frequencies of sinusoids contained in the signal (peaks at normalized frequency 0.2 and 0.25), although some realisations are larger than the mean at the frequency of interest. At other frequencies, the PSD estimate of 100 realisations fluctuates from 0 to 10, and the mean of them approach to 1, which is equal to the power of the WGN. In addition, the standard derivation is proportional to the PSD; therefore, the confidence interval is much greater at the peaks. Moreover, since the mean of PSD does not converge to 0 as the data length increases, and the variance approaches to the PSD that is not 0, therefore, the biased ACF estimator is not consistent.

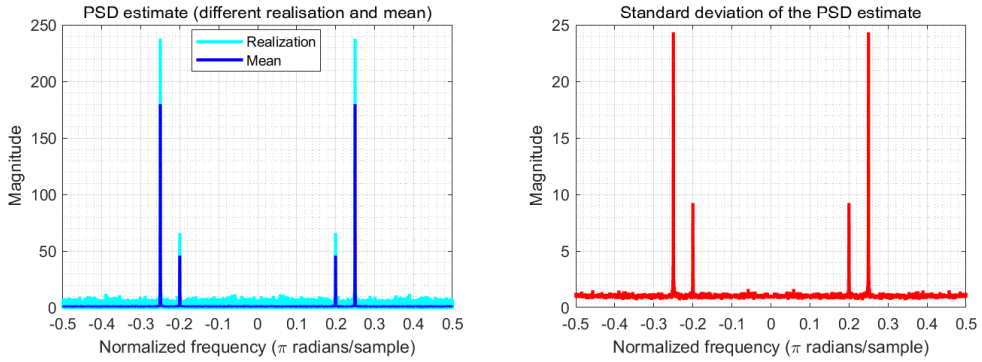


Figure 5: Top: the PSD estimate of 100 realisations and mean for the AWGN signal with biased ACF. Bottom: the standard derivation of AWGN signal with bised ACF

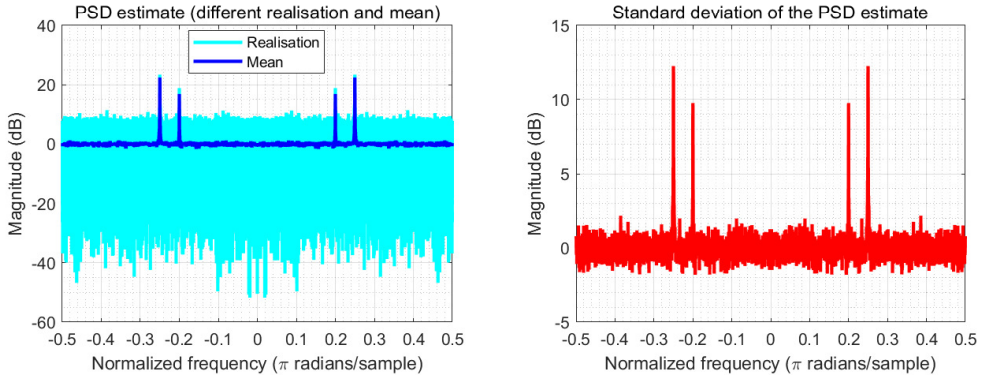


Figure 6: Top: the PSD estimate in dB of 100 realisations and mean for the AWGN signal with biased ACF. Bottom: the standard derivation in dB of AWGN signal with bised ACF

c) The conversion to dB can be seen as a smoothing operation on parts of the spectrum with large variations. This because that the gradient of the original spectrum is given by $\frac{d(P(f))}{df}$, while the logarithm changes the gradient into $\frac{1}{P(f)ln(10)}\frac{d(P(f))}{df}$. Thus, for large PSD in the original spectrum, it will be attenuated in dB version, and for small PSD, it will be relatively enlarged. As show in Fig.6, It maintains the trend, but attenuates the transition at peaks, while amplifies the fluctuation due to noise. As a result, the dB plot enables the clearer detection of peaks with smooth and more detailed trend information about low-amplitude signals. In conclusion, this presentation is effective since it provides more trend information while cares less about the meaningless transitions.

d) We generate three complex-valued signal of different frequencies and length as expressed in (7), (8) and (9),

$$x(n) = e^{2\pi(0.3)n} + w(n) \tag{7}$$

$$x(n) = e^{2\pi(0.3)n} + e^{2\pi(0.32)n} + w(n) \tag{8}$$

$$x(n) = e^{2\pi(0.3)n} + e^{2\pi(0.32)n} + e^{2\pi(0.35)n} + w(n) \tag{9}$$

where $w(n) \sim N(0, 0.02)$. The periodogram estimates with rectangular window for these three signals of different length N are shown in Fig.7, where the spectrum becomes sharper and closer to the ideal line spectral as N increases. The reason is that when the periodogram resolution, which is proportional to 1/N, is greater than the separation between signal frequencies, then the line spectra will not be able to be correctly detected. More specifically, the periodogram estimate is the convolution of the true PSD with the Fourier Transform of rectangular window, which is a sinc-like function; therefore, the periodogram resolution is determined by the main lobe width of the window. Take signal in (8) as an example, the frequency separation is $0.02Hz$, and the main lobe width of the rectangular window is $0.89/N$. Therefore, the minimum signal length to separate frequencies clearly is $N = 0.89/0.02 = 45.5$. As a result, for $N > 45$, the peaks can be divided by window's main lobe, therefore, other methods like subspace method should be taken into consideration.

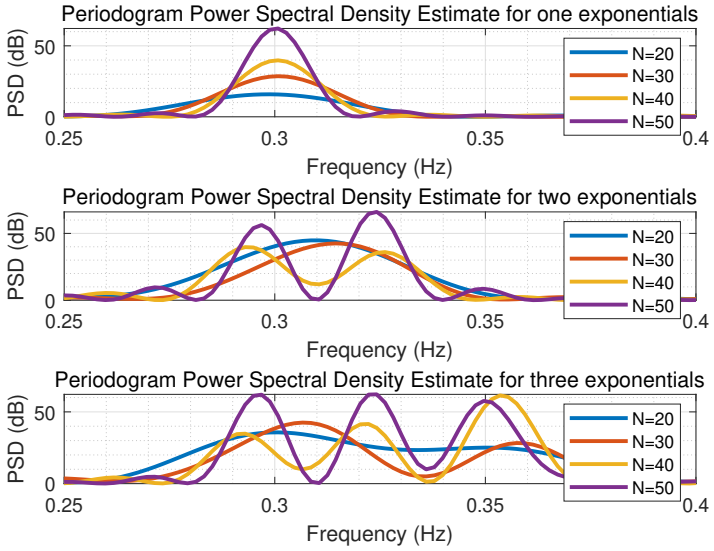


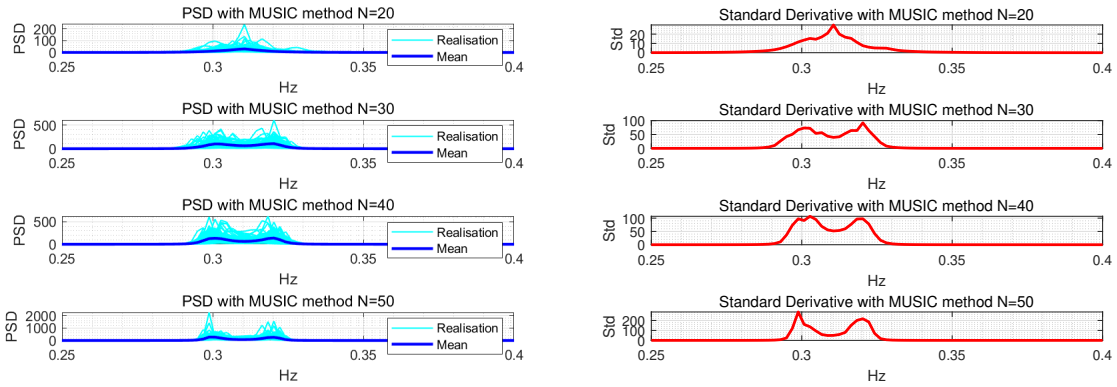
Figure 7: Periodogram estimate with rectangular for one exponential, two exponential and three exponential signals with different length

e) The code related is shown below. For the first line of code, the `corrmtx` is to obtain the autocorrelation matrix estimate for the signal `x`. The input signal `x` is a linear combination of complex exponential signals with the WGN noise. The `14` defines the dimension of the autocorrelation matrix that is $(14+1)\times(14+1)$. `mod` is set to take the modified covariance method so that the result can be more accurate. For the output of the `corrmtx`, `R` is the autocorrelation matrix, and `X` is a rectangular Toeplitz matrix, where `X'X` is a biased estimate of autocorrelation matrix.

For the second line of code, it estimates the periodogram based on the Multiple Signal Classification (MUSIC) algorithm. Input `R` is the autocorrelation matrix, `2` is the dimensionality of the signal subspace, [] represents to use the default FFT points, `1` the normalized sampling frequency, and `corr` indicates that `R` is the correlation matrix estimate. For the output, `S` is the pseudospectrum, while `F` is the normalized signal frequency.

For the third line, its aim is to plot the pseudospectrum versus the normalized signal frequency with the line-width equal to 2, while also set the frequency range in (0.25, 0.4).

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



(a) PSD estimation

(b) Standard derivation

Figure 8: The PSD estimate and standard derivations of two complex exponentials with different lengths. Left: PSD estimate. Right: Standard derivations

As shown in Fig.8, the two line spectra can be identified for $N = 30$, while the correlation-based estimator can not distinguish them well with the same signal length. Therefore, the MUSIC method is able to capture more details about the spectrum and outperforms the correlation-based method for fewer signal samples (form of superresolution). The MUSIC method generally has a higher resolution (lower bias) than the periodogram, which will decrease as the Signal-Noise-Ratio (SNR) increases. Moreover, MUSIC method also has higher variances than the periodogram method, and the variances are still proportional to the PSD. As a result, when SNR increases, the bias on peak decreases while the variance increase, which will benefit the detection of peaks locations.

The main disadvantage of the MUSIC method is that, as a subspace method, it requires the dimensions p to be known or estimated in advance. The estimation of p need be determined by Minimum Description Length (MDL) or Akaike Information Criterion (AIC), which significantly increases the computation complexity.

1.4 Spectrum of Autoregressive Processes

a) The AR parameters $\mathbf{a}$ is obtained from the Yule-Walker equation given by:

$$\mathbf{R_{xx}a} = \mathbf{r_{xx}} \Rightarrow \mathbf{a} = \mathbf{R_{xx}^{-1}r_{xx}} \tag{10}$$

where, $\mathbf{R_{xx}}$ is the autocorrelation matrix. From (10), it is clear that determination of $\mathbf{a}$ requires the $\mathbf{R_{xx}}$ to be invertible. For the biased estimator, $\mathbf{R_{xx}}$ is a positive definite Toeplitz matrix, which is definitely invertible. On the other hand, for the unbiased estimator, as shown in Fig.4, the ACF does not decay to 0 for large lags so that the positive definiteness may not be satisfied. Therefore, there may be non-positive eigenvalues of ACF matrix, and the inversion of autocorrelation matrix will not exist. As a result, the parameters $\mathbf{a}$ cannot be solve.

b) According to Fig.9, as the model order increases, the estimated PSD matches the truth better. For example, AR(2) only detects one peak, while AR(6) detects two peaks but with some deviations, and AR(10) successfully estimates the actual PSD with minor errors. With the model order increasing, the prediction error decreases, and the AR model will have more degrees of freedom to accurately capture the fully underlying structures of data to make more adequate estimation. However, increasing the model order will increase the computation complexity. Fig.9 suggests that the prediction error reaches a plateau at $p = 6$, therefore, the order $p = 7$ will be optimal for the trade-off between the prediction error and the complexity. Moreover, the prediction error at the actual order $p = 4$ is high, which illustrates that it performs not well on the estimation of PSD. This is due to the bias of the ACF and the small sample size. Also, different samples will produce different estimation result.

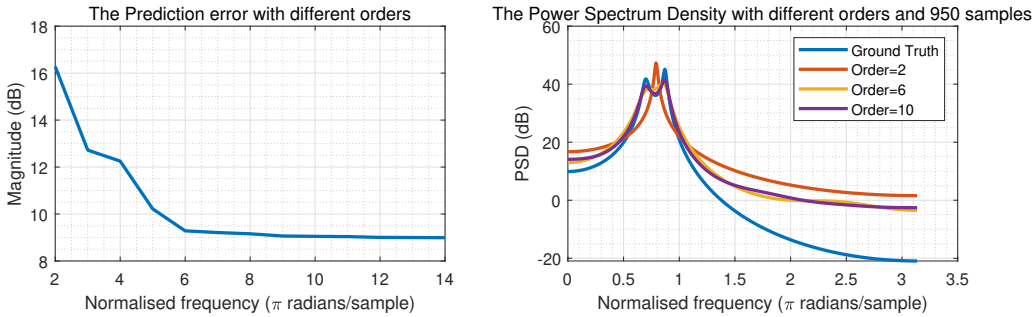


Figure 9: Left: the prediction error with different orders of the AR model with 1000 samples. Right: the estimated PSD with AR models of different order and the true PSD with 1000 samples.

As a result, the model accuracy can be improved by increasing the model order or using more samples.

c) As shown in Fig.10, compared with the aforementioned 1000-sample condition, the Mean Squared Errors (MSEs) of prediction are reduced for all orders after the sample size increased to 10000. The error is significantly low at the actual order $p = 4$, and the PSD estimation with it correctly detects the two peaks. In addition, the estimated PSD approaches to the true PSD better. For the chosen order is lower the correct order $p = 4$, then the AR model will not have enough freedoms to capture all variations of interest, which results in a high prediction error. On the other hand, for a high order, the model will be over-fitting, leading to great computation complexity.

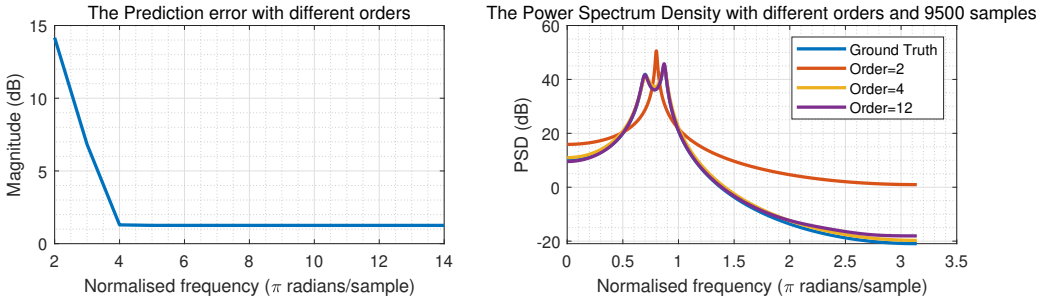


Figure 10: Top: the prediction error with different orders of the AR model with 10000 samples. Bottom: the estimated PSD with AR models of different order and the true PSD with 10000 samples.

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

a) Respiratory sinus arrhythmia (RSA) is the modulation of cardiac function with respiratory function, which can be observed through speeding up or slowing down the heart rate. The R-R-Interval(RRI) in a cardiac (ECG) data recording will be shortened during inspiration (breathing out) or extended during expiration (breathing in). Therefore, in order to study the features of RSA and ECG signal, we obtain the RRI data for three trials, as normal breathing, fast breathing at 25bpm and the slow breathing at 7.5bpm. The periodograms of these three-trial RRI data are obtained with the standard method as well as Bartlett's Method with different window lengths (i.e., 50s, 150s). The hamming window are applied for the wider main-lobe and more bounded side-lobes. The result is shown in Fig.11.

b) Fig.11 presents that there is a trade-off between the variance reduction and the frequency resolution. Comparing the standard periodogram (window length more than 200s) and averaged periodogram with the $150s$ window, it is clear that the averaged periodogram is more smooth than the standard one with smaller variance. Moreover, the $50s$-window periodogram only provides the trend for the PSD, while omit many meaningless variations. Although using a small window will reduce

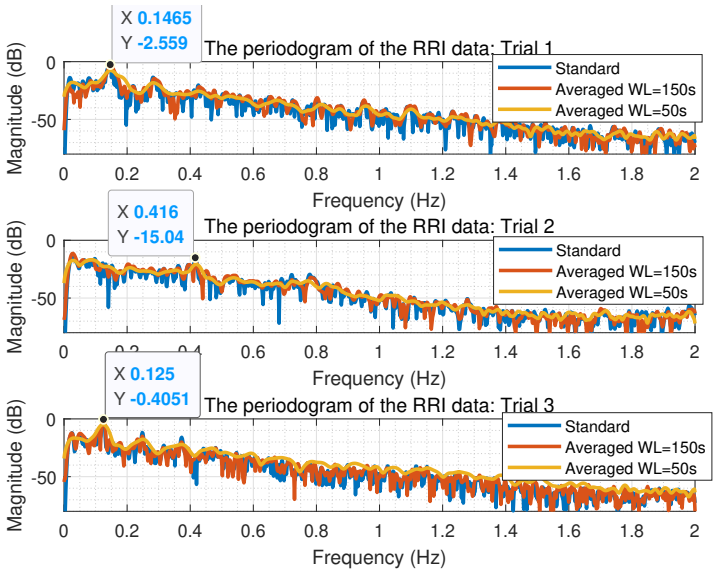


Figure 11: The standard and averaged periodogram with 50s and 150s windows for the RRI data of three trials

the amplitudes of oscillations, the peaks of interest can still be correctly located. In addition, the decrease in samples leads to the lower frequency resolution, allowing peaks become more wider and harmonics to be identified more easily.

The peaks of interest in Fig.11 occurs at $f_1 = 0.1465Hz$, $f_2 = 0.418Hz$ and $f_3 = 0.123Hz$. The respiratory rate corresponding to the RRI frequency is equal to $BreathsPerMinute(BPM) = 60 \times f$. Therefore, the breathing rates we obtained are $9bpm$, $25bpm$ and $7.5bpm$, respectively. Since the standard and average periodogram are approximately coincident at peaks, the rates acquired are valid and satisfy our initial assumptions for normal, fast and slow breaths.

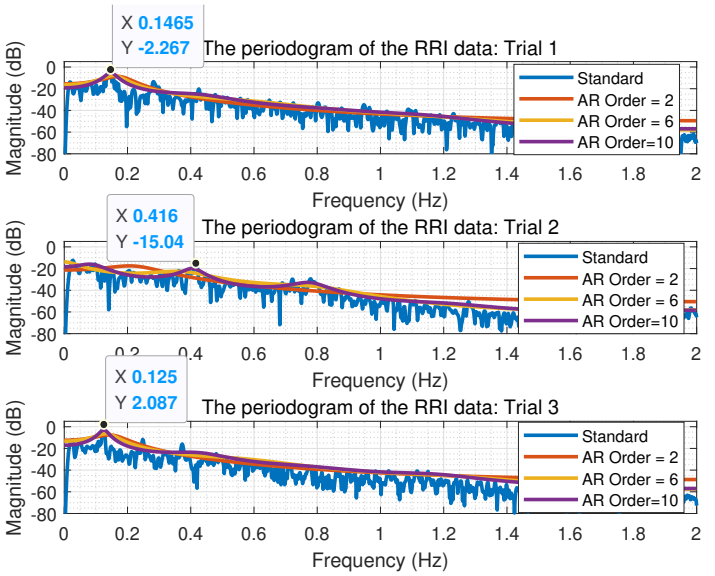


Figure 12: The standard and AR model periodogram with different orders for the RRI data of three trials

c) Fig.12 shows that the spectrum estimated by the AR model is much more smooth than the stan-

dard one with only the trend of PSD, and little information about noise and harmonics. The main characteristics of PSD, such as peaks, are much more easily to be identified. Additionally, the orders $p = 2$, $p = 6$ and $p = 10$ are the optimal orders for the normal, fast and slow breathing AR models, respectively. Although the AR model method does not suffer from the trade-off of bias and variance, it still needs a prudent choice for the order of model. A small model order will introduce high estimation error, and it will highly smooth the spectrum, leading fewer details about the spectrum can be identified. Also, a large order will lead to over-modelling and great complexity.

1.6 Robust Regression

a) As shown in Fig.13, non-zero singular values input signal $\mathbf{X}$ is 3, indicating the rank of $\mathbf{X}$ is equal to 3. The rank of the signal corrupted by noise $\mathbf{X}_{noise}$ is 10, with 10 non-zero singular values. This is because that the input signal will only have 3 subspace; however, the noise signal will occupy the whole subspace. Due to the noise, the singular values for signal subspace will be slightly increased, but for other remaining subspace, the singular values are increased significantly by noise. More specifically, there are quite small squared errors at $\mathbf{X}$ subspace, but much greater deviations at the others. Therefore, when the Signal-Noise-Ratio (SNR) is small, the errors in irrelevant subspace will make the singular values in all dimensions similar, then it will be much difficult to identify the rank of $\mathbf{X}_{\text{noise}}$.

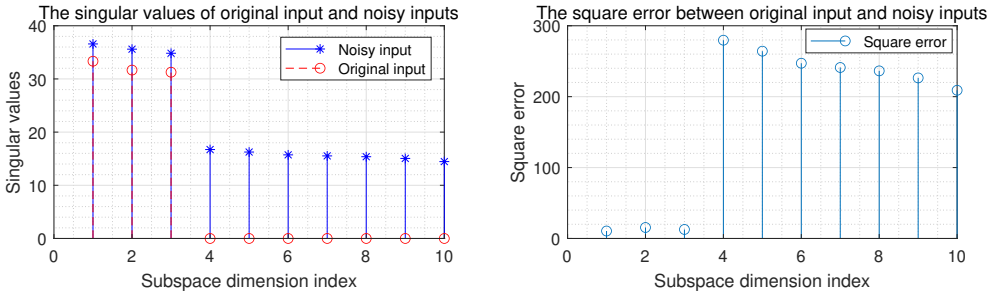


Figure 13: Left: the singular values of input signal and noisy input signal. Right: the square error between each singular value of original input and noisy input

b) Fig.14 suggests that the low-rank $\tilde{\mathbf{X}}_{\text{noise}}$, which uses only the 3 most significant principle components, is a good approximation of the original since the square error between $\mathbf{X}$ and $\tilde{\mathbf{X}}_{\text{noise}}$ are all low enough in all dimensions. This proves that the meaningful information of the original signal is only stored in a few principle components, while the others are the additive noise. The reason why squared errors are not reduced to 0 is that some information is lost due to the truncated Singular Value Decomposition (SVD). Also, it is quite interesting that not all errors reduced at a same scale. This is due to different SNRs, which means that a higher SNR will make it more difficult to remove noise as a result of overfitting of noise.

c) The output data are obtained as

$$\mathbf{Y} = \mathbf{XB} + \mathbf{N}_{\mathbf{Y}} \tag{11}$$

where the regression matrix $\mathbf{B}$ can be calculated based on two methods as Ordinary Least Squares (OLS) and Principle Component Regression (PCR). The $\hat{\mathbf{B}}_{\text{OLS}}$ and $\hat{\mathbf{B}}_{\text{PCR}}$ are given by (12) and (13). Fig.15 shows the square error between the original output and the reproduced output using OLS or PCR method with $r = 3$ largest principle components on both training and testing data. Table.1 shows the summation error. Comparing the results shown in Fig.15 and Table.1, we conclude that the OLS method fits training data better, while the PCR method performs better on testing data. This can be explained that the PLS considered the noise components, therefore, it over-fits the training set. Also, since it fits much on the random noise in training data, therefore, paying less attention on the relationships between signal samples, which results in a poor performance on testing data. As for PCR method, it ignores the noise components so that it does not over-fit the training set, therefore, is

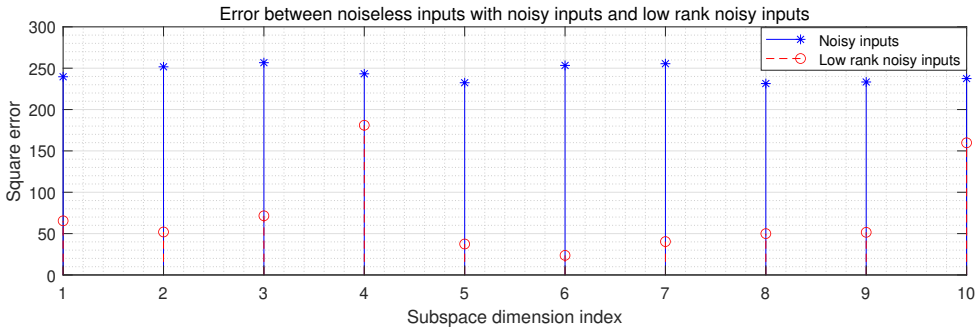


Figure 14: Square errors between input signal and noisy input signal or low-rank noisy input signal

Table 1: Summation of square error on training and testing data

	Training data	Testing data
OLS	3551.6569	2377.3501
PCR	3577.1496	2351.5383

able to predict test data better with more knowledge of signal samples.

$$\hat{\mathbf{B}}_{OLS} = (\mathbf{X}_{noise}^{\mathbf{T}}\mathbf{X}_{noise})^{-\mathbf{1}}\mathbf{X}_{noise}^{\mathbf{T}}\mathbf{Y} \tag{12}$$

$$\hat{\mathbf{B}}_{PCR} = \mathbf{V}_{1:r}(\Sigma_{1:r})^{-1}\mathbf{U}_{1:r}^{T}\mathbf{Y} \tag{13}$$

Figure 15: The estimation error by OLS and PCR

d) The best way to evaluate the OLS and PCR method is to test the estimated regression coefficients $\hat{\mathbf{B}}$, over an ensemble of test data. Then, we generate 100 realisations of output based on regression coefficients, and compute the Mean Square Error (MSE) between each realisation $\mathbf{Y}$ and its estimate $\hat{\mathbf{Y}}$ given by

$$MSE = \mathrm{E}\{\| \mathbf{Y} - \hat{\mathbf{Y}} \|_2^2\} \tag{14}$$

to compare the effectiveness of these two methods. The MSE errors are listed in Fig.16, and the summation errors for OLS and PCR are 2351.6783 and 2327.469, respectively. It is clear that PCR outperforms the OLS by a 0.5% smaller MSE, but both methods provides confident results.

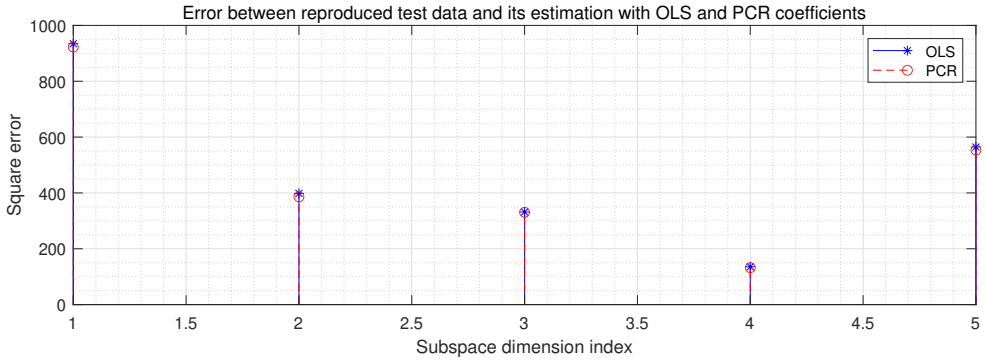


Figure 16: The MSE error by OLS and PCR with 100 realisations

2 Adaptive Signal Processing

2.1 The Least Mean Square (LMS) Algorithm

a) The LMS algorithm is implemented to estimate the coefficients of the auto-regressive process and the original input signal. In general, a second-order auto-regressive process (AR(2)) $x(n)$ can be expressed as:

$$x(n) = a_1 x(n-1) + a_2 x(n-2) + \eta(n) \tag{15}$$

where $\eta(n) \sim \mathcal{N}(0, \sigma_n^2)$. In this section, we uses $\mathbf{x}(n) = [x(n-1), x(n-2)]^T$ as the input signal to the LMS system. Therefore, the correlation matrix $\mathbf{R}_{xx}$ of the input vector is given by:

$$\mathbf{R}_{xx} = \mathbb{E}[\mathbf{x}(n)\mathbf{x}^T(n)] = \mathbb{E}\begin{bmatrix} x(n-1)x(n-1) & x(n-1)x(n-2) \\ x(n-2)x(n-1) & x(n-2)x(n-2) \end{bmatrix} = \begin{bmatrix} r_{xx}(0) & r_{xx}(1) \\ r_{xx}(-1) & r_{xx}(0) \end{bmatrix} \tag{16}$$

where $r_{xx}(k)$ is the autocorrelation function (ACF) of $x(n)$, and k indicates the time lag between two delayed signals. Then for the aforementioned AR(2) process, the ACF of it with time lag k is given by:

$$\begin{aligned} r_{xx}(k) = \mathbb{E}[x(n)x(n-k)] &= \mathbb{E}\Big[a_1 x(n-1)x(n-k) + a_2 x(n-2)(n-k) + \eta(n)x(n-k)\Big] \\ &= a_1 \mathbb{E}[x(n-1)x(n-k)] + a_2 \mathbb{E}[x(n-2)x(n-k)] + \mathbb{E}[\eta(n)x(n-k)] \end{aligned} \tag{17}$$

Since the $x(n)$ is the white noise, and noise signals with different delays are not correlated to each other, the term $\mathbb{E}[\eta(n)x(n-k)]$ in (17) equals σ_n^2 when $k = 0$, and vanishes for $k > 0$. As a result, (17) becomes:

$$r_{xx}(k) = \begin{cases} a_1 r_{xx}(1) + a_2 r_x x2 + \sigma_n^2 & , k = 0 \\ a_1 r_{xx}(k-1) + a_2 r_{xx}(k-2) & , k > 0 \end{cases} \tag{18}$$

Then as original parameters are $a_1 = 0.1$, $a_2 = 0.8$ and $\sigma_n^2 = 0.25$, and the ACF is an even function with $R_{xx}(k) = R_{xx}(-k)$, thus the $r_{xx}(0)$ and $r_{xx}(1)$ can be solved by three equations with three unknowns, which are listed as:

$$\begin{aligned} r_{xx}(0) &= a_1 r_{xx}(1) + a_2 r_x x2 + \sigma_n^2 \\ r_{xx}(1) &= a_1 r_{xx}(0) + a_2 r_{xx}(-1) = a_1 r_{xx}(0) + a_2 r_{xx}(1) \\ r_{xx}(2) &= a_1 r_{xx}(1) + a_2 r_{xx}(0) \end{aligned} \tag{19}$$

The solution gives $r_{xx}(0) = \frac{25}{27}$, $r_{xx}(1) = \frac{25}{54}$. Therefore, the entries of the correlation matrix of the input vector is given by:

$$\mathbf{R}_{xx} = \begin{bmatrix} r_{xx}(0) & r_{xx}(1) \\ r_{xx}(-1) & r_{xx}(0) \end{bmatrix} = \begin{bmatrix} r_{xx}(0) & r_{xx}(1) \\ r_{xx}(1) & r_{xx}(0) \end{bmatrix} = \frac{25}{54}\begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \tag{20}$$

The LMS algorithm converges in mean if

$$0 < \mu < \frac{2}{\lambda_{max}} \tag{21}$$

where λ_{max} is the maximum eigenvalue of $\mathbf{R}_{xx}$. Through eigenvalue decomposition, we get that $\lambda_{max} = \frac{25}{18}$. Consequently, the range for step size is:

$$0 < \mu < 1.44 \tag{22}$$

b) In this section, we implement the LMS adaptive predictor for 1 realization and 100 realizations of 1000 samples $x(n)$ with step size as $\mu = 0.05$ and $\mu = 0.01$, respectively. The squared prediction error $e^2(n)$ for 1 realization and the learning curve as $10log(e^2(n))$ for 100 realizations are shown in Fig.17.

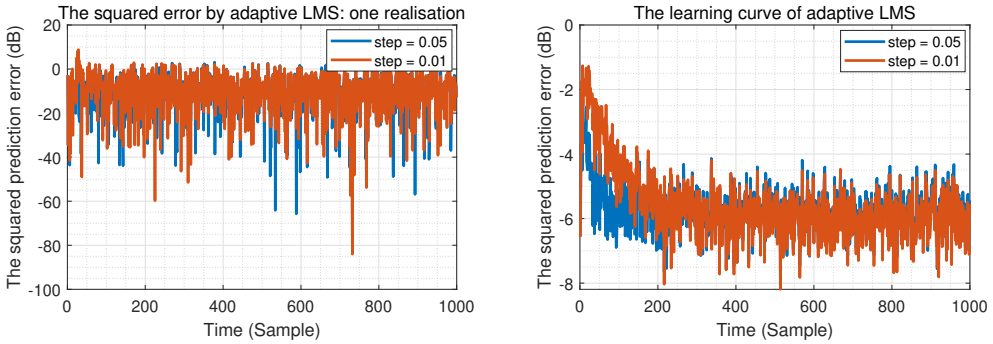


Figure 17: Performance of LMS adaptive predictor with $\mu = 0.05$ and $\mu = 0.01$. Left: squared prediction error for 1 realization $x(n)$. Right: learning curve for 100 realizations of 1000-sample $x(n)$

From the learning curve, it is clear that for $\mu = 0.05$, the averaged squared prediction error converges within 50 time steps, but for $\mu = 0.01$, it takes around 200 time steps, which satisfies the theoretical argument that larger step size will lead to steeper descent LMS predictor. However, there is a trade-off between the convergence speed and the stability of the final estimated result. As shown in Fig.17, the prediction errors fluctuated even after they have converged to their theoretical values. The fluctuations are resulted from the noisy gradient vector used to update weights. Although the large step size achieves faster convergence, the trajectory of the prediction error has much more oscillations, which requires the improvement of adaptive step size decaying with time to balances the speed and stability of convergence.

c) The theoretical LMS misadjustment is approximated by (23).

$$\mathcal{M}_{LMS} \approx \frac{\mu}{2}\mathrm{Tr}\{\mathbf{R}_{\mathrm{xx}}\} = \frac{\mu}{2}1.8519 \tag{23}$$

where $\mathbf{R}_{xx}$ is given by (20), and Tr is the trace operation. From the aforementioned analysis, the squared prediction error converges after the time index greater than 200. Therefore, in order to estimate the misadjustment over the steady state, we time average the squared prediction error for $t > 500$ with 100 independent experiments. The corresponding misadjustment is calculated by (24)

$$\mathcal{M}_{EST} = \frac{\mathrm{EMSE}}{\sigma_n^2} = \frac{\mathrm{MSE} - \sigma_{\mathrm{n}}^2}{\sigma_n^2} = \frac{\mathrm{MSE}}{\sigma_n^2} - 1 \tag{24}$$

μ	$\mathcal{M}_{LMS}$	$\mathcal{M}_{EST}$
0.05	0.0463	0.0488
0.01	0.0093	0.0074

Table 2: Theoretical and estimated misadjustments for LMS algorithm with $\mu = 0.05$ and $\mu = 0.01$

The theoretical and estimated misadjustments for LMS algorithm with $\mu = 0.05$ and $\mu = 0.01$ are summarized in Table 2. For both step sizes, the estimated misadjustments are closely agreed with

the theoretical ones. More specifically, the estimated error for $\mu = 0.05$ is 5.40%, and 20.43% for $\mu = 0.01$. Generally, the estimation error for smaller step size should be smaller. Thus, these errors and misalignment may be due to the approximation of theoretical $\mathcal{M}$, the finite sample size, the limited realizations, and the approximated transient period. The larger μ leads to a larger misadjustment, which agrees with the theory as the large step size will bring large steady-state error.

d) Fig.18 shows the trajectories of estimated LMS adaptive filter coefficients versus iteration times, with two different step sizes as $\mu = 0.05$ and $\mu = 0.01$. The results are obtained by averaging the estimated values over 100 trials of experiment to reduce the variance of estimation.

For both step sizes, $\hat{a}_1$ have converged to its actual value, but $\hat{a}_2$ have a negative offset with the optimal value (12.5% for $\mu = 0.05$ and 4.5% for $\mu = 0.01$). Moreover, for $\mu = 0.05$, the convergence is achieved within 140 iterations with great oscillations, however for $\mu = 0.01$, it takes about 600 times with a much smoother trajectory. Therefore, a larger step size has a faster convergence speed but with higher oscillations around the optimal solution, and the estimation may have a larger offset with the actual coefficients.

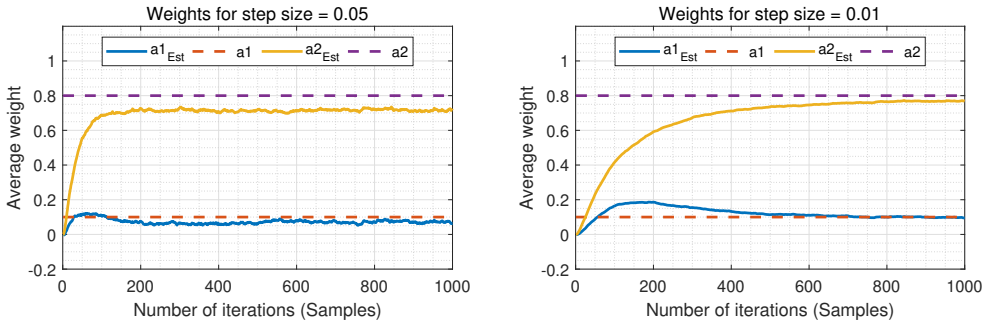


Figure 18: Estimation of LMS adaptive filter coefficients with $\mu = 0.05$ and $\mu = 0.01$ by averaging values of coefficients over 100 trials of experiment. Left: $\mu = 0.05$. Right: $\mu = 0.01$

e) The cost function is given as:

$$\mathcal{J}_2(n) = \frac{1}{2}\left(e^2(n) + \gamma\|\mathbf{w}(n)\|_2^2\right) \tag{25}$$

Replacing $e(n)$ in (25) in terms of the input vector $\mathbf{x}(n)$, target signal $y(n)$, and weights $\mathbf{w}(n)$:

$$\begin{aligned}\mathcal{J}_2(n) &= \frac{1}{2}\left(\|y(n) - \mathbf{w}(n)^T\mathbf{x}(n)\|_2^2 + \gamma\|\mathbf{w}(n)\|_2^2\right) \\ &= \frac{1}{2}\left((y(n) - \mathbf{w}(n)^T\mathbf{x}(n))^T(y(n) - \mathbf{w}(n)^T\mathbf{x}(n)) + \gamma\mathbf{w}(n)^T\mathbf{w}(n)\right)\end{aligned} \tag{26}$$

Then the gradient of cost function is given by:

$$\begin{aligned}\nabla_{\mathbf{w}}\mathcal{J}_2(n) &= -(y(n) - \mathbf{w}(n)^T\mathbf{x}(n))\mathbf{x}(n) + \gamma\mathbf{w}(n) \\ &= -e(n)\mathbf{x}(n) + \gamma\mathbf{w}(n)\end{aligned} \tag{27}$$

Finally, using gradient decent algorithm to update the weights:

$$\begin{aligned}\mathbf{w}(n+1) &= \mathbf{w}(n) - \mu\nabla_{\mathbf{w}}\mathcal{J}_2(n) \\ &= \mathbf{w}(n) - \mu\Big(-e(n)\mathbf{x}(n) + \gamma\mathbf{w}(n)\Big) \\ &= (1 - \mu\gamma)\mathbf{w}(n) + \mu e(n)\mathbf{x}(n)\end{aligned} \tag{28}$$

Therefore, the leaky LMS equation is shown as:

$$\text{Leaky LMS}: \qquad \mathbf{w}(\mathrm{n}+1) = (1 - \mu\gamma)\mathbf{w}(\mathrm{n}) + \mu\mathrm{e}(\mathrm{n})\mathbf{x}(\mathrm{n}) \tag{29}$$

f) Leakage coefficient is used to force the LMS filter coefficients to zero when either the error or the input vector becomes zero, and also force any undamped modes of the system to zero. From Fig.19, it is clear that the coefficients $\hat{a}_2$ have not converged to the actual value for all leakage γ and step sizes μ. The bias between the estimated and true coefficients increases as γ increases. Moreover, the leaky LMS also has the trade -off between the convergence speed and steady-state error variance as mentioned before.

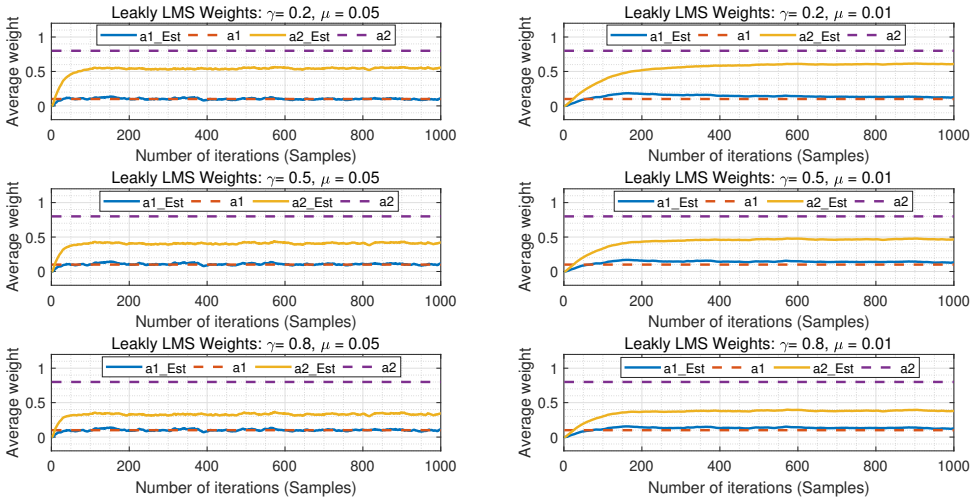


Figure 19: Estimation of coefficients by Leaky LMS adaptive filter with different leakage coefficients and step sizes. Left: $\mu = 0.05$, $\gamma = 0.2, 0.5, 0.8$. Right: $\mu = 0.01$, $\gamma = 0.2, 0.5, 0.8$

The optimal solution to the linear system is based on the Wiener Filter, which is given by:

$$\mathbf{w}_* = \mathbf{R}^{-1}\mathbf{p} \tag{30}$$

where $\mathbf{p}$ is the cross-correlation between the input vector and the target output. Therefore, when $R_{\mathbf{xx}}$ has zero eigenvalues, it becomes positive semi-definite, which may not guarantee the invertibility, leading to instability of LMS filter. With the leakage coefficient, the solution becomes:

$$\mathbf{w}_{Leak} = (\mathbf{R} + \gamma\mathbf{I})^{-1}\mathbf{p} \tag{31}$$

The autocorrelation matrix $R_{\mathbf{xx}}$ has been replaced by $\mathbf{R} + \gamma\mathbf{I}$, where $\gamma > 0$. In result, the eigenvalues of the autocorrelation will not be zero, and none undamped modes of LMS filter will occur. However, the leakage introduces a bias to the steady-state solution. The larger the leakage, the larger the $\mathbf{R}+\gamma\mathbf{I}$, resulting in a larger bias to the optimal solution.

2.2 Adaptive Step Sizes

Since a constant step size leads to the trade-off between the rate of convergence and the oscillation errors at the steady-state values, it is important to introduce the variable step size (VSS) algorithms to achieve a small excess mean-square error, while maintain the convergence speed. In this section, we implement the gradient adaptive step-size (GASS) algorithms based on three different coefficient update equations (Benvenist, Ang & Farhan and Matthews Xi) to a real-valued MA(1) system, and compare their performance with the LMS filter with fixed step sizes ($\mu = 0.01$ and $\mu = 0, 1$). The results are shown in Fig.20 and Fig.21.

In Fig.20, we choose the initial step size as $\mu_0 = 0.1$, and in Fig.21, μ_0 is set as 0.2. From Fig.20, we can find that Benvenist algorithm outperforms the other algorithms with a faster coefficients update speed and convergence speed, but has a higher complexity as $\mathcal{O}(M^2)$ compared with other algorithms ($\mathcal{O}(M)$). This is due to the $\mathbf{x}(n-1)\mathbf{x}(n-1)^T$ term in the update equation. For Ang & Farhang algorithm, it reaches the steady-state value slower than Benvenist algorithm, but faster than others within

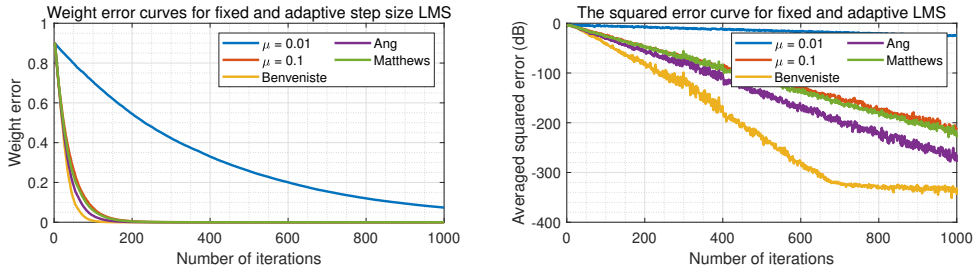


Figure 20: The performance comparison between three GASS algorithms with $\mu_0 = 0.1$ and the standard algorithms with $\mu = 0.1$ and $\mu = 0.01$. Left: weight error curve. Right: squared error curve.

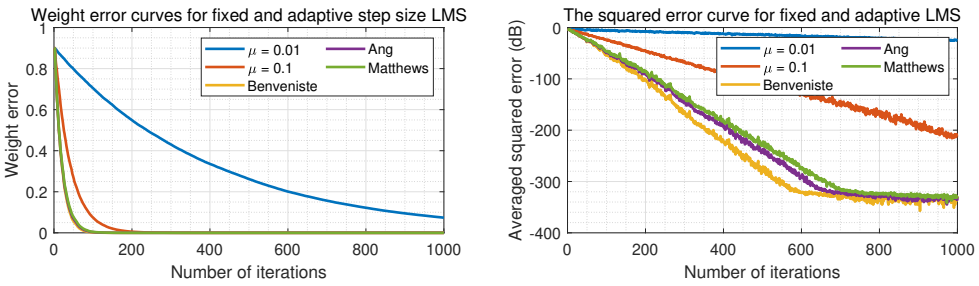


Figure 21: The performance comparison between three GASS algorithms with $\mu_0 = 0.2$ and the standard algorithms with $\mu = 0.1$ and $\mu = 0.01$. Left: weight error curve. Right: squared error curve.

80 iterations. However, when $\mu_0 = 0.1$, the performance of Matthews algorithm is quite similar to fixed $\mu = 0.1$. Moreover, when initial step size is set to 0.2, the performance difference between VSS algorithms and fixed step size LMS becomes larger, as the steady-state error for Benvenist, Ang and Matthews is all below -300 dB, but is -213.3 dB for $\mu = 0.1$, and -25.27 dB for $\mu = 0.01$. Therefore, by choosing a proper initial step size, the GASS algorithm will outperform the standard LMS filter with a faster rate of convergence and smaller steady-state error.

b) Starting from the posteriori error:

$$e_p(n) = d(n) - \mathbf{x}(n)^T \mathbf{w}(n+1) \tag{32}$$

Since the update equation is given by:

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu e_p(n)\mathbf{x}(n) \tag{33}$$

Therefore, replacing $\mathbf{w}(n+1)$ in (32) by (33) gives:

$$\begin{aligned} e_p(n) &= d(n) - \mathbf{x}(n)^T(\mathbf{w}(n) + \mu e_p(n)\mathbf{x}(n)) \\ &= d(n) - \mathbf{x}(n)^T\mathbf{w}(n) - \mu e_p(n)\|\mathbf{x}(n)\|^2 \\ &= e(n) - \mu e_p(n)\|\mathbf{x}(n)\|^2 \\ &= \frac{e(n)}{1+\mu\|\mathbf{x}(n)\|^2} \end{aligned} \tag{34}$$

Then $\triangle\mathbf{w}(n)$ can be expressed as:

$$\begin{aligned} \triangle\mathbf{w}(n) &= \mathbf{w}(n+1) - \mathbf{w}(n) \\ &= \mu e_p(n)\mathbf{x}(n) \\ &= \mu\frac{e(n)}{1+\mu\|\mathbf{x}(n)\|^2}\mathbf{x}(n) \end{aligned} \tag{35}$$

As a result, the update equation with posteriori error $e_p(n)$ can be written as:

$$\begin{aligned}\mathbf{w}(n+1) &= \mathbf{w}(n) + \frac{e(n)}{\frac{1}{\mu} + \|\mathbf{x}(n)\|^2}\mathbf{x}(n) \\ &= \mathbf{w}(n) + \frac{\beta}{\epsilon + \|\mathbf{x}(n)\|^2}e(n)\mathbf{x}(n)\end{aligned} \tag{36}$$

which is equivalent to the NLMS algorithm with $\beta = 1$ and $\epsilon = \frac{1}{\mu}$.

c) The generalized normalized gradient descent (GNGD) algorithm is another popular VSS algorithm, which is based on NLMS with adaptive $\epsilon(n)$ given by:

$$\epsilon(n+1) = \epsilon(n) - \rho\mu\frac{e(n)e(n-1)\mathbf{x}^T(n)\mathbf{x}(n-1)}{(\epsilon(n-1) + \|\mathbf{x}(n-1)\|^2)^2} \tag{37}$$

The comparison of weight error and squared prediction error between the Benveniste's GASS and GNGD is shown in Fig.22. It should be noticed that, in this section, we adopts the scale as $\rho = 0.05$, and initial step sizes as $\mu_{0(GASS)} = 0.1$, $\mu_{0(GNGD)} = 1$ to ensure two algorithms can converge to the optimal solution within 100 iterations.

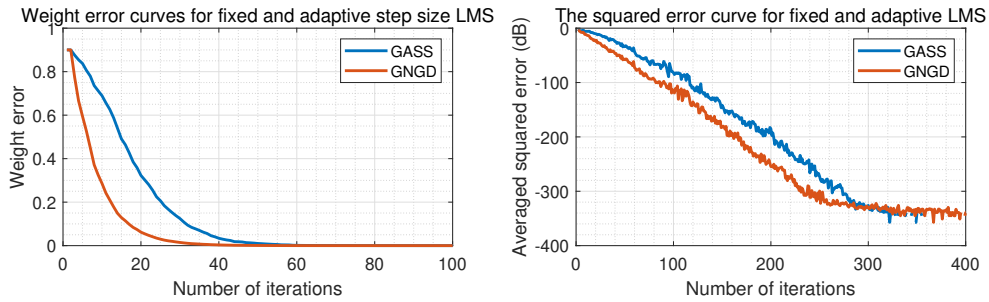


Figure 22: The performance comparison between the GNGD algorithm with $\mu_0 = 1$ and the Benveniste's GASS with $\mu_0 = 0.1$. Left: weight error curve. Right: squared error curve.

Step	Sub-step	Multiplications	Additions
$\psi(n)$	$\mu(n-1)\mathbf{x}(n-1)^T$	M	0
	$\mu(n-1)\mathbf{x}(n-1)\mathbf{x}(n-1)^T$	M^2	0
	$\left[I - \mu(n-1)\mathbf{x}(n-1)\mathbf{x}(n-1)^T\right]\psi(n-1)$	M^2	$2M^2 - M$
	$e(n-1)\mathbf{x}(n-1)$	M	0
	$\left[I - \mu(n-1)\mathbf{x}(n-1)\mathbf{x}(n-1)^T\right]\psi(n-1) + e(n-1)\mathbf{x}(n-1)$	0	M
$\mu(n+1)$	$\mathbf{x}^T(n)\psi(n)$	M	M-1
	$\rho e(n)\mathbf{x}^T(n)\psi(n)$	2	0
	$\mu(n) + \rho e(n)\mathbf{x}^T(n)\psi(n)$	0	1
$\mathbf{w}(n+1)$	$\mu(n)e(n)\mathbf{x}(n)$	$M+1$	0
	$\mathbf{w}(n) + \mu(n)e(n)\mathbf{x}(n)$	0	1
Total		$\mathcal{O}(M^2)$	$\mathcal{O}(M^2)$

Table 3: Multiplication and addition operations in Benveniste's GASS algorithm.

Fig.22 describes that the GNGD algorithm converges within 30 iterations, while Benveniste's GASS takes about 60 iterations for convergence. Moreover, the steady-state error of GNGD is always smaller than that of GASS with less oscillations. For example, at time index 60, the squared error is $-116.8dB$ for GASS and $-85.25dB$ for GNGD. For the complexity, the number of multiplications and additions of Benveniste's GASS algorithm and GNGD algorithm are listed in Table 3 and Table 4, respectively.

Step	Sub-step	Multiplications	Additions
$\epsilon(n+1)$	$\|\|\mathbf{x}(n-1)\|\|^2$	M	M-1
	$\mathbf{x}^T(n)\mathbf{x}(n-1)$	M	M-1
	$\rho\mu\frac{e(n)e(n-1)\mathbf{x}^T(n)\mathbf{x}(n-1)}{(\epsilon(n-1)+\|\|\mathbf{x}(n-1)\|\|^2)^2}$	6	1
	$\epsilon(n)-\rho\mu\frac{e(n)e(n-1)\mathbf{x}^T(n)\mathbf{x}(n-1)}{(\epsilon(n-1)+\|\|\mathbf{x}(n-1)\|\|^2)^2}$	0	1
$\mathbf{w}(n+1)$	$\|\|x(n)\|\|^2$	M	M-1
	$\frac{\beta e(n)\mathbf{x}(n)}{\epsilon(n)+\|\|x(n)\|\|^2}$	$M+2$	1
	$\mathbf{w}(n)+\frac{\beta e(n)\mathbf{x}(n)}{\epsilon(n)+\|\|x(n)\|\|^2}$	0	M
Total		$\mathcal{O}(M)$	$\mathcal{O}(M)$

Table 4: Multiplication and addition operations in GNGD algorithm.

According to Table 3, Table 4, the Benveniste's GASS algorithm has a complexity of $\mathcal{O}(M^2)$, and GNGD has a complexity of $\mathcal{O}(M)$. Therefore, GNGD outperforms Benveniste's GASS in terms of convergence speed, prediction error and computation complexity.

2.3 Adaptive Noise Cancellation

a) To set up an Adaptive Line Enhancer (ALE), the delay Δ must be carefully chosen to ensure the uncorrelation between the noisy signal $s(n)$ and the predictor input $\mathbf{u}(n)$. The noisy signal is composed of a clean signal $x(n)$ and a noise signal $\eta(n)$. To find the minimum delay for the ALE system, we start from the mean square error (MSE).

$$\begin{aligned}
\texttt{MSE} &= \mathbb{E}\Big[\big(s(n)-\hat{x}(n)\big)^2\Big] \\
&= \mathbb{E}\Big[\big(x(n)+\eta(n)-\hat{x}(n)\big)^2\Big] \\
&= \mathbb{E}\Big[\big(\eta(n)+(x(n)-\hat{x}(n))\big)^2\Big] \\
&= \mathbb{E}\Big[\eta^2(n)\Big]+\mathbb{E}\Big[\big(x(n)-\hat{x}(n)\big)^2\Big]+2\mathbb{E}\Big[\eta(n)\big(x(n)-\hat{x}(n)\big)\Big] \\
&= \mathbb{E}\Big[\eta^2(n)\Big]+\mathbb{E}\Big[\big(x(n)-\hat{x}(n)\big)^2\Big]+2\mathbb{E}\Big[\eta(n)x(n)\Big]-2\mathbb{E}\Big[\eta(n)\hat{x}(n)\Big]
\end{aligned} \tag{38}$$

In (38), the first term $\mathbb{E}\Big[\eta^2(n)\Big]$ is a constant independent of delay Δ. The second term $\mathbb{E}\Big[\big(x(n)-\hat{x}(n)\big)^2\Big]$ is irrelevant to noise $\eta(n)$. As the noise and clean signal is assumed to be uncorrelated, the third term $2\mathbb{E}\Big[\eta(n)x(n)\Big]$ is 0. Therefore, Δ should be chosen to minimize the last term $-2\mathbb{E}\Big[\eta(n)\hat{x}(n)\Big]$, while at the same time maintain the uncorrelation between $s(n)$ and $\mathbf{u}(n)$. Since $\eta(n)=v(n)+0.5v(n-2)$, the last term can be written as:

$$\begin{aligned}
\mathbb{E}\Big[\eta(n)\hat{x}(n;\Delta)\Big] &= \mathbb{E}\Big[\big(v(n)+0.5v(n-2)\big)\mathbf{w}^T\mathbf{v}(n;\Delta)\Big] \\
&= \mathbb{E}\Big[\big(v(n)+0.5v(n-2)\big)\sum_{k=0}^{M-1}w_k s(n-\Delta-k)\Big] \\
&= \mathbb{E}\Big[\big(v(n)+0.5v(n-2)\big)\sum_{k=0}^{M-1}w_k\big(x(n-\Delta-k)+\eta(n-\Delta-k)\big)\Big] \\
&= \mathbb{E}\Big[\big(v(n)+0.5v(n-2)\big)\sum_{k=0}^{M-1}w_k\big(\eta(n-\Delta-i)\big)\Big] \\
&= \mathbb{E}\Big[\big(v(n)+0.5v(n-2)\big)\sum_{i=0}^{M-1}w_i\big(v(n-\Delta-i)+0.5v(n-2-\Delta-i)\big)\Big]
\end{aligned} \tag{39}$$

Since $v(n)$ is the identical and independently distributed white noise, the $\mathbb{E}\Big[v(n-i)v(n-j)\Big] = 0$ if $i \neq j$. Therefore, when $\Delta > 2$, there is no overlapping between $v(n), v(n-2), v(n-\Delta-i)$ and $v(n-2-\Delta-i)$, the MSE can be minimized to 0. Consequently, the minimum delay Δ is 3.

Fig.23 and Fig.24 verify the conclusion derived before. When $\Delta < 3$ the MSPE is very large, leading to bad reconstruction of $x(n)$.

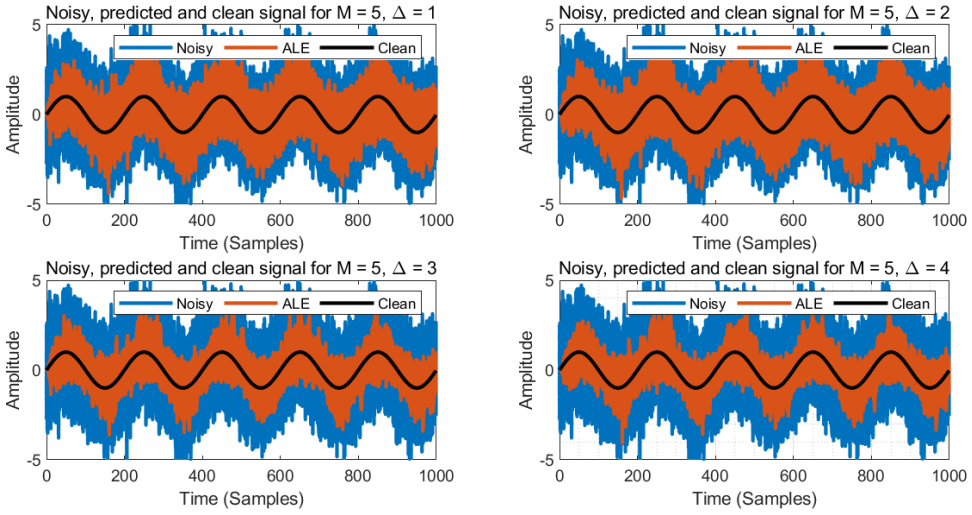


Figure 23: Noisy, predicted and clean signal predicted with model order $M = 5$, and different delays. (1) $\Delta = 1$. (2) $\Delta = 2$. (3) $\Delta = 3$. (4) $\Delta = 4$.

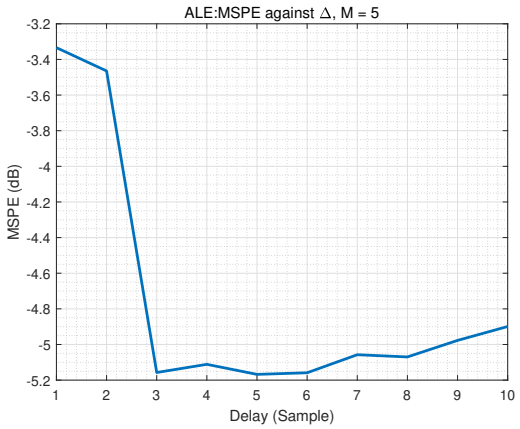


Figure 24: MSPE against delay from 1 to 10 with constant model order 5

b) Fig.25 illustrates the relationship between the mean square prediction error (MSPE), delay Δ, and filter orders M for ALE system. It is clear that the delay and filter order pair with minimum MSPE is $(\Delta = 3, M = 5)$.

Theoretically, the increase in model order can make the prediction more accurate. However, we find that for $M > 5$, the MSPE increases as model order increases. That is because a large model order will lead to over-modelling, resulting in high computation complexity and over-fitting noise. The result also illustrates that the model order should be at least 2 to successfully capture the signal variations. Moreover, when $M = 3, 4$, the MSPE is just slightly higher that that at $M = 5$. Therefore, $M = 3$ or 4 is recommended to balance the MSPE and complexity.

Initially, we consider the term $\mathbb{E}\Big[\big(x(n) - \hat{x}(n)\big)^2\Big]$ in (38) is 0. Nevertheless, as delay increases, the

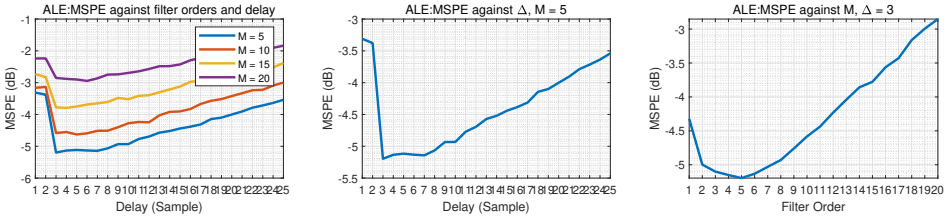


Figure 25: Relationship between MSPE, delay and filter order for ALE system. Left: MSPE against filter orders and delay. Middle: MSPE against delay with $M = 5$. Right: MSPE against filter order with $\Delta = 3$.

time-shift between $x(n)$ and $\hat{x}(n)$ will increases, causing the term $\mathbb{E}\left[\left(x(n) - \hat{x}(n)\right)^2\right]$ to increase. As a result, a large delay will also lead to a large MSPE. Also, $\Delta > 2$ should be satisfied as mentioned before for uncorrelation. Therefore, $\Delta = 3$ is an optimal choice to minimize the time shift between actual signal and estimated signal, and also guarantee $\hat{x}(n)$ and $\eta(n)$ to be uncorrelated.

c) In this section, we compare the performance of the ALE with Adaptive Noise Cancellation (ANC). For ANC, the secondary noise $\epsilon(n)$ is used as input, given by:

$$\epsilon(n) = 0.9\eta(n) + 0.05 \tag{40}$$

where $\eta(n)$ is the primary noise, and the two noise are uncorrelated in some unknown way.

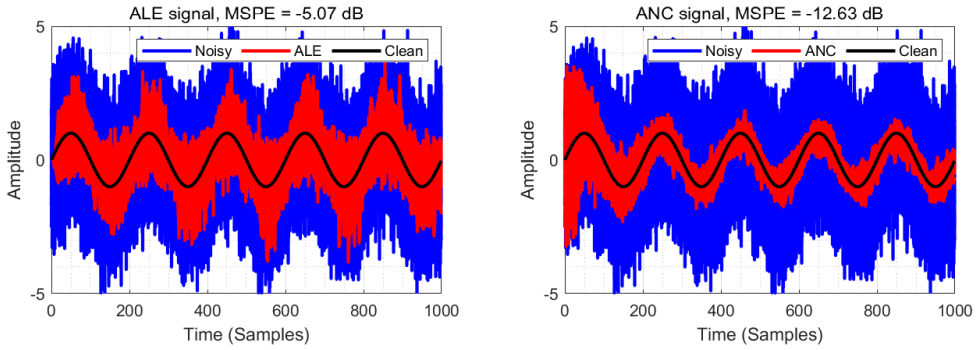


Figure 26: The noisy signal, clean signal and the estimated signal by ANC and ALE by 100 realizations. Left: ALE. Right: ANC.

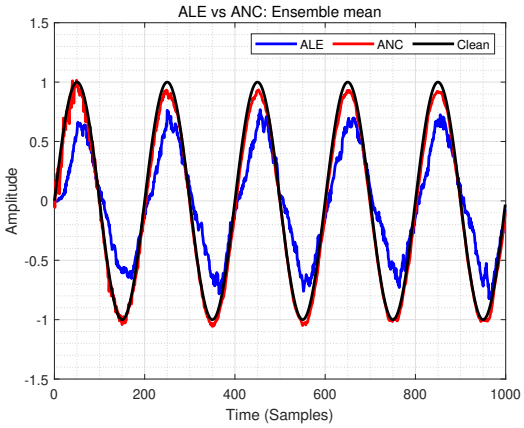


Figure 27: The ensemble mean of noisy signal, clean signal and the estimated signal by ANC and ALE

According to Fig.26, the ANC outperforms the ALE with a smaller MSPE (ALE: $-5.07dB$, ANC: $-12.26dB$). However, it can be noticed that the ANC has considerable errors at first time steps, but it

tracks the sinusoidal wave much better from 400 time index. In comparison, the performance of ALE over time does not be improved obviously. Therefore, if the sample size is large, ANC is recommended for noise cancellation, and ALE can be an alternative choice when the number of signal samples is small.

d) In order to remove the 50 Hz components in EEG signal by ANC, we generate a synthetic reference input $\epsilon(n)$, composed of a sinusoidal wave of 50 Hz and unit amplitude corrupted by white Gaussian noise of variance 0.01. Fig.28 shows the spectrum of original EEG data. The spectrum is obtained using a Hamming window of length $L = 4096$ and 80% overlap.

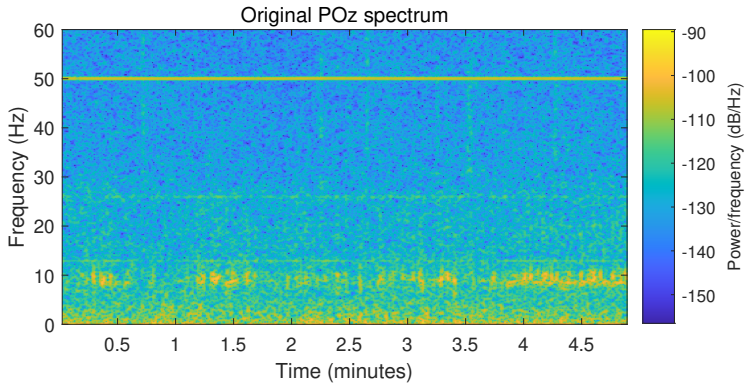


Figure 28: Original EEG POz spectrum

After applying ANC with different step sizes μ and filter length M, the obtained spectrum are shown in Fig.29.

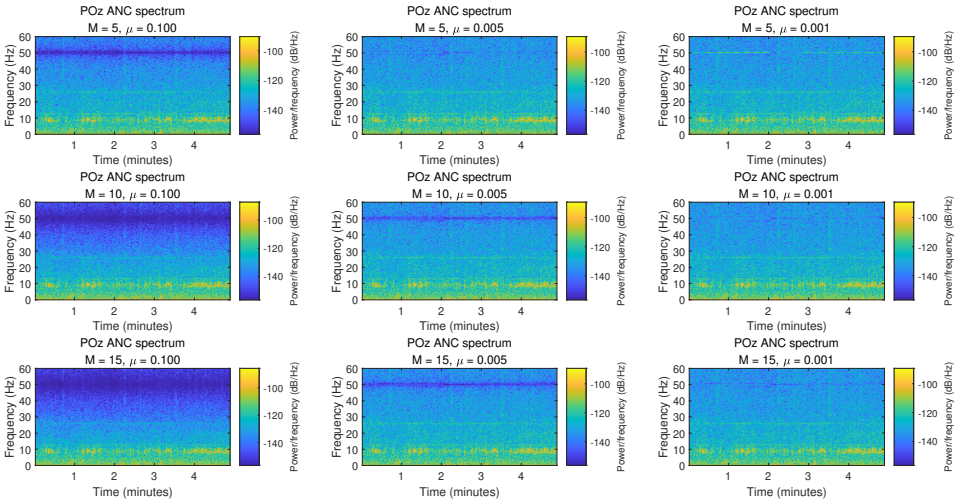


Figure 29: The ANC denoised signal spectrum with different filter order M and step size μ

From spectrogram, we can find that the model order defines the degree of denoising. When model order is small (e.g., $M = 5$), under-modeling will occur, leading to a poor performance as the power-line interface is not perfectly attenuated. However, very large model order, such as $M = 15$, will cause over-modelling, which will also weaken the neighbour frequency around 50 Hz, thus degrade the performance. Therefore, a middle order $M = 10$ is an appropriate order to remove the 50 Hz components, while not affecting other components.

Furthermore, the step size μ also affects the performance of ANC. It can be noticed that for large step size (e.g., $\mu = 0.1$), the performance of ANC is poor. That is because that a large step size covers

a large range of frequency so that the neighbour frequency around 50 Hz will also be degraded. Moreover, the small step size will take a long time to reach the steady-state, but will not disrupt the frequency nearby.

In conclusion, to accurately remove the 50 Hz component, while not affect the other frequencies, $(M = 10, \mu = 0.001)$ is an optimal choice. Fig.30 presents the periodogram before and after the ANC denosing. It is clear that except the 50 Hz, all other frequency components overlap. Therefore, ANC successfully remove the power-line interface.

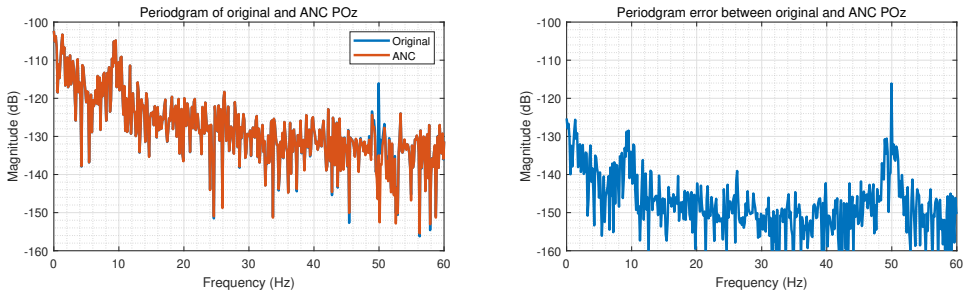


Figure 30: Left: Periodogram of original and ANC denoised signal. Right: Periodogram error between the original and ANC denoised signal

3 Adaptive Spectrum Estimation

3.1 Complex LMS and Widely Linear Modelling

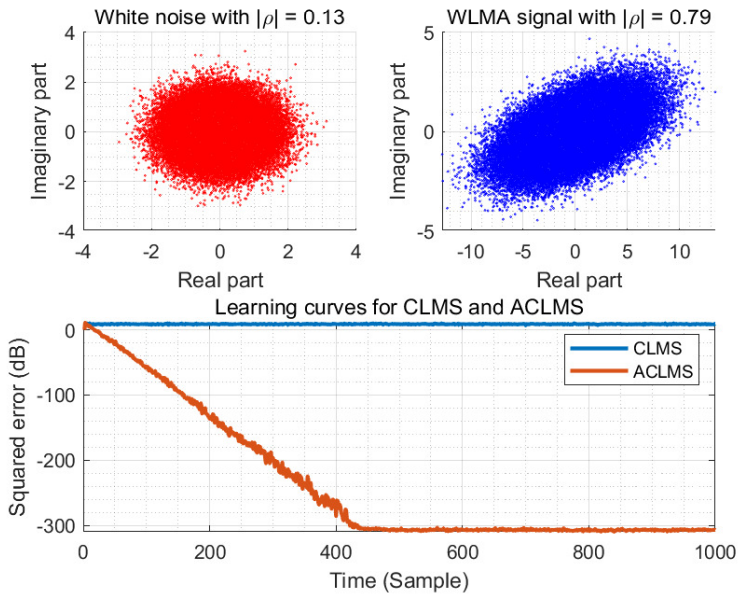


Figure 31: Top: circularity of white noise and WLMA(1) signal. Bottom: The learning curve of CLMS and ACLMS algorithms.

a) In this section, we implement the Complex LMS (CLMS) and Augmented CLMS (ACLMS) on a first-order widely-linear-moving-average process (WLMA(1)). The circularity of white noise and WLMA(1) signal is shown in Fig.31. Circularity coefficient measures the ratio between the magnitude of pseudo-covariance and covariance of a complex signal, and a smaller circularity indicates a more circular signal. As expected, the white noise is circularly distributed with $|\rho| = 0.13$. Though its

expectation is 0, it still has some circularity deviation, which is due to the small sample sizes 1000. The WLMA(1) is non-circular with $|\rho| = 0.79$.

It should be noted that CLMS can only analyze the circular data, since it only considers the covariance matrix of complex variables, while ignores the pseudo-covariance. But for ACLMS, since it introduces the augmented input vector with both input vector and its conjugate, and also contains an additional weights as $\mathbf{g}(n)$, thus it can fully capture the second order statistics of the complex data. Therefore, ACLMS is applicable to non-circular data. Fig.31 presents the learning curve of CLMS and ACLMS. The result agrees with our expectation, as the CLMS reaches the square error $8.35dB$, while the ACLMS achieves -300 dB. In conclusion, ACLMS outperforms CLMS for modeling non-circular data, but for circular data, CLMS can be chosen for a lower computation cost.

b) Fig.32, Fig.33 and Fig.34 show the circularity plot as well as learning curves of three wind regimes (low, medium, high). For learning curves, $\mu = 0.01, 0.005, 0.001$ for low, medium and high speed, respectively. The circularity coefficient $|\rho|$ measures the probability distribution of the random

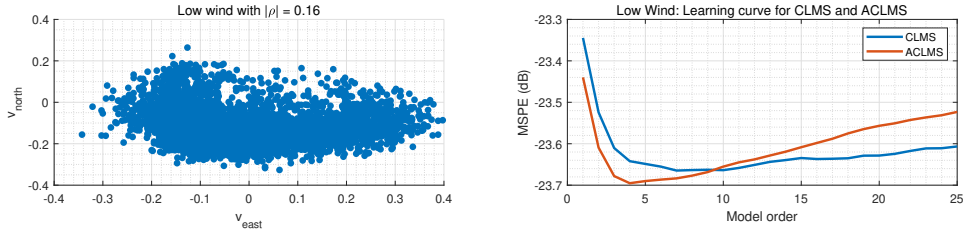


Figure 32: Circularity plot and the learning curve of low wind regime. Left: circularity plot. Right: learning curve.

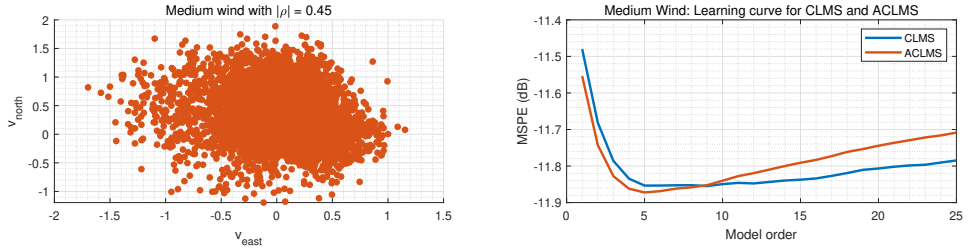


Figure 33: Circularity plot and the learning curve of medium wind regime. Left: circularity plot. Right: learning curve.

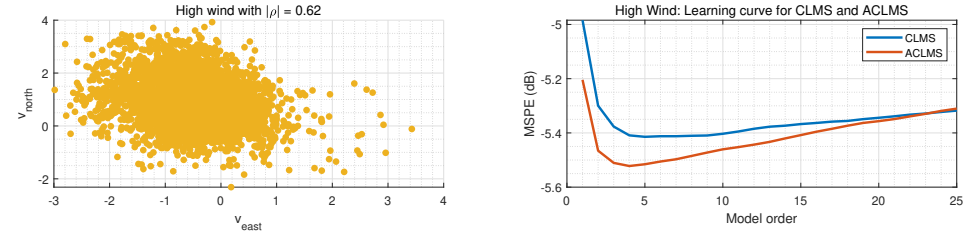


Figure 34: Circularity plot and the learning curve of high wind regime. Left: circularity plot. Right: learning curve.

variable, which means that the smaller the $|\rho|$, the more the probability distribution is independent of angle, thus the more circular the random variable. From results, high-speed data has the largest $|rho| = 0.62$, the medium-speed has $|rho| = 0.45$, and low-speed has the least $|rho| = 0.16$, which illustrates that the circularity of wind data increases as speed increases, and the data is less circular.

By comparing the learning curves of three wind regimes, we find that the ACLMS outperforms the CLMS below a relatively low model order (9 for low speed, 9 for medium, and 23 for high speed).

This agrees with the aforementioned analysis, as ACLMS is more appropriate for non-circular data with more degree of freedom. Especially, for least circular data (high wind), the gap between performance (MSPE) of ACLMS and CLMS is quite large as $0.1dB$. It should be noticed that at a low model order, both ALMS and CLMS have high MSPE. This is due to under-fitting, where both algorithms does not capture the information of data comprehensively, and the better performance of ACLMS is due to the excess degree of freedom. For model order is large, the over-fitting also causes MSPE to increase. Moreover, the ACLMS reaches the threshold of over-modelling order faster than CLMS because of its more parameters. Also, the over-modelling order occurs faster for a more circular patterns (e.g., low-speed 9 vs high-speed 23). Therefore, ACLMS is more applicable to data with large circularity, and requires a small model order. For a more circular data, the performance of CLMS approaches CLMS, but with less computation complexity and larger range for model order to avoid over-fitting.

c) In this part, we generate three-phase voltages with different amplitudes and phases to form the balanced and unbalanced system. The balanced system has three same peak voltages and total phase shift of three voltages are 120 degrees. As shown in Fig.35 and Fig.**??**, the balanced system has a circular shape with $|\rho| = 0$, while unbalanced systems don't. Moreover, the larger the $|\rho|$, the larger the amplitude and phases difference in the unbalanced system. Therefore, if the shape of the circularity diagram is not a perfect circle, the fault exists in the system (i.e. unbalance). The degree of distortion measures the severity of fault.

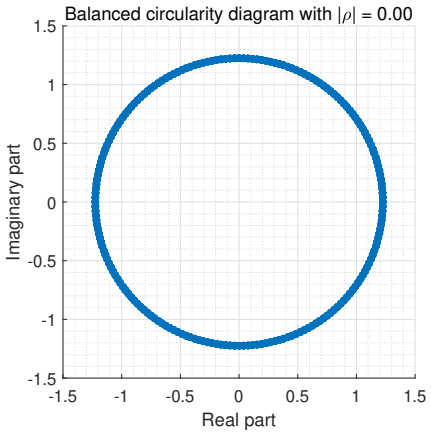


Figure 35: Circularity diagram of the balanced system

d) Starting from the balanced complex $\alpha - \beta$ voltage given by:

$$v(n) = \sqrt{\frac{3}{2}} V e^{j(2\pi \frac{f_o}{f_s} n + \phi)} \tag{41}$$

Then the voltage at time index $n+1$ is:

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

$$= \sqrt{\frac{3}{2}} V e^{j(2\pi \frac{f_o}{f_s} n + \phi)} e^{j2\pi \frac{f_o}{f_s}} \tag{43}$$

$$= v(n) e^{j2\pi \frac{f_o}{f_s}} \tag{44}$$

Then consider a strictly linear autoregressive model of order 1, and express the equation as a function

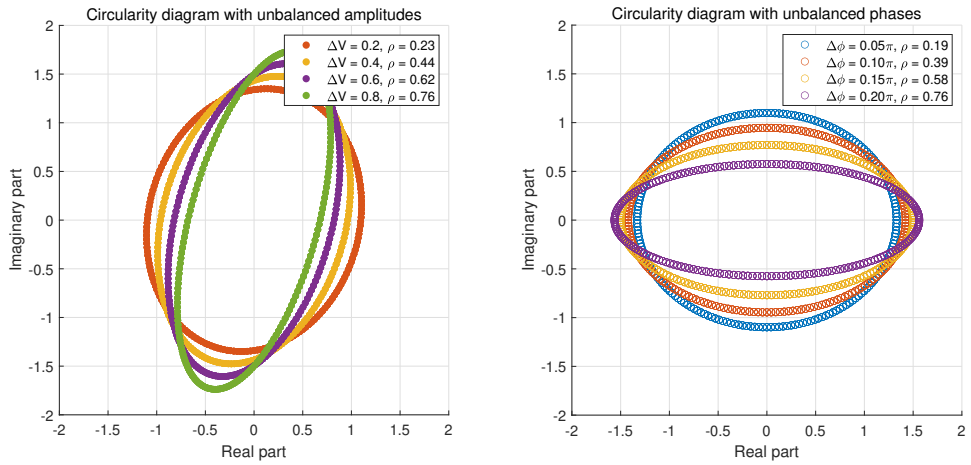


Figure 36: Circularity diagram of unbalanced system Left: Unbalanced systems with different amplitudes. Right: Unbalanced systems with different magnitude of phases

of coefficients $h(n)$:

$$\begin{aligned} v(n+1) &= v(n)h^*(n) \\ v(n)e^{j2\pi\frac{f_o}{f_s}} &= v(n)h^*(n) \\ e^{j2\pi\frac{f_o}{f_s}} &= h^*(n) \\ e^{-j2\pi\frac{f_o}{f_s}} &= h(n) \\ e^{-j2\pi\frac{f_o}{f_s}} &= |h(n)|e^{j\left(arctan\left(\frac{\mathrm{Im}\{h(n)\}}{\mathrm{Re}\{h(n)\}}\right)\right)} \end{aligned} \tag{45}$$

Since (45) holds if and only if the magnitude and phases of both sides are equal, then we get:

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

Therefore, the frequency of the balanced complex voltage $f_o(n)$ is expressed as:

$$f_o = \frac{f_s}{2\pi}arctan\left(\frac{\mathrm{Im}\{h(n)\}}{\mathrm{Re}\{h(n)\}}\right) \tag{47}$$

For unbalanced voltage, the voltage is expressed as:

$$v(n) = A(n)e^{j(2\pi\frac{f_o}{f_s}n+\phi)} + B(n)e^{-j(2\pi\frac{f_o}{f_s}n+\phi)} \tag{48}$$

The voltage at $n+1$ satisfies:

$$v(n+1) = A(n+1)e^{j(2\pi\frac{f_o}{f_s}(n+1)+\phi)} + B(n+1)e^{-j(2\pi\frac{f_o}{f_s}(n+1)+\phi)} \tag{49}$$

Also, it can be expressed in terms of $h(n)$ and $g(n)$:

$$\begin{aligned} v(n+1) &= h^*(n)v(n) + g^*(n)v^*(n) \\ &= h^*(n)\left[A(n)e^{j(2\pi\frac{f_o}{f_s}n+\phi)} + B(n)e^{-j(2\pi\frac{f_o}{f_s}n+\phi)}\right] \\ &+ g^*(n)\left[A^*(n)e^{-j(2\pi\frac{f_o}{f_s}n+\phi)} + B^*(n)e^{j(2\pi\frac{f_o}{f_s}n+\phi)}\right] \end{aligned} \tag{50}$$

Then, equating (49) and (50) gives:

$$A(n+1)e^{j(2\pi\frac{f_o}{f_s}(n+1)+\phi)} = \left[h^*(n)A(n) + g^*(n)B^*(n)\right]e^{j(2\pi\frac{f_o}{f_s}n+\phi)} \tag{51}$$

$$B(n+1)e^{-j(2\pi\frac{f_o}{f_s}(n+1)+\phi)} = \left[h^*(n)B(n) + g^*(n)A^*(n)\right]e^{-j(2\pi\frac{f_o}{f_s}n+\phi)} \tag{52}$$

Assuming the amplitude change between the adjacent time indexes is negligible, as $A(n+1) \approx A(n), B(n+1) \approx B(n)$, then we can get:

$$e^{j2\pi\frac{f_o}{f_s}} = \frac{h^*(n)A(n) + g^*(n)B^*(n)}{A(n+1)} \approx h^*(n) + g^*(n)\frac{B^*(n)}{A(n)} \tag{53}$$

$$e^{-j2\pi\frac{f_o}{f_s}} = \frac{h^*(n)B(n) + g^*(n)A^*(n)}{B(n+1)} \approx h^*(n) + g^*(n)\frac{A^*(n)}{B(n)} \tag{54}$$

Since $e^{j2\pi\frac{f_o}{f_s}} = (e^{-j2\pi\frac{f_o}{f_s}})^*$, therefore, we can get:

$$h^*(n) + g^*(n)\frac{B^*(n)}{A(n)} = (h^*(n) + g^*(n)\frac{A^*(n)}{B(n)})^* = h(n) + g(n)\frac{A(n)}{B^*(n)} \tag{55}$$

Then, divide each side in (55) by $\frac{A(n)}{B^*(n)}$, and collect same items:

$$g^*(n)\left(\frac{B^*(n)}{A(n)}\right)^2 + \left(h^*(n) - h(n)\right)\frac{B^*(n)}{A(n)} - g(n) = 0 \tag{56}$$

Notice that (56) is a quadratic function of $\frac{B^*(n)}{A(n)}$, hence we solve the equation and obtain:

$$\frac{B^*(n)}{A(n)} = \frac{-\left(h^*(n) - h(n)\right) \pm \sqrt{\left(h^*(n) - h(n)\right)^2 + 4g^*(n)g(n)}}{2g^*(n)} \tag{57}$$

$$= j\frac{\mathrm{Im}\{h(n)\} \pm \sqrt{\mathrm{Im}\{h(n)\}^2 - |g(n)|^2}}{g^*(n)} \tag{58}$$

Then, substitute the solution into (53):

$$e^{j2\pi\frac{f_o}{f_s}} = h^*(n) + j\left(\mathrm{Im}\{h(n)\} \pm \sqrt{\mathrm{Im}\{h(n)\}^2 - |g(n)|^2}\right) \tag{59}$$

$$= \mathrm{Re}\{h(n)\} \pm j\sqrt{\mathrm{Im}\{h(n)\}^2 - |g(n)|^2} \tag{60}$$

$$\tag{61}$$

Since $f_s \gg f_o > 0$, we give up negative sign solution, and rewrite the right side into the exponential form with magnitude $\rho > 0$:

$$e^{j2\pi\frac{f_o}{f_s}} = \mathrm{Re}\{h(n)\} + j\sqrt{\mathrm{Im}\{h(n)\}^2 - |g(n)|^2} \tag{62}$$

$$= \rho e^{j\left(\frac{\sqrt{\mathrm{Im}\{h(n)\}^2 - |g(n)|^2}}{\mathrm{Re}\{h(n)\}}\right)} \tag{63}$$

Then, equating the phases on two sides gives:

$$f_o = \frac{f_s}{2\pi} arctan\left(\frac{\sqrt{\mathrm{Im}\{h(n)\}^2 - |g(n)|^2}}{\mathrm{Re}\{h(n)\}}\right) \tag{64}$$

e) The CLMS and ALMS are tested on balanced system ($|\rho| = 0$), unbalanced amplitude system ($|\rho| = 0.40$, $,V_a = 0.4, V_b = 1, V_c = 1.6$), and unbalanced phase system ($|\rho| = 0.57, \Delta_b = 0.2\pi, \Delta_c = 0.4\pi$). The equation (47) and (64) are used for the 50 Hz frequency component estimation by CLMS and ACLMS. Fig.37, Fig.38 and Fig.39 present the learning curves and power-line frequency estimation by CLMS and ACLMS for balanced and unbalanced systems.

For the balanced system, both CLMS and ACLMS can estimate the power-line frequency accurately. However, the ACLMS oscillates for about 60 time steps, then reaches the steady value, but CLMS achieves the optimal value at the beginning. This is because that the balanced system is circular, but for ACLMS with two degree of freedom and small sample size, it is difficult to determine whether the system is circular or non-circular. Therefore, the ACLMS will over-fit the data, and lead to overshooting at the first time steps.

On the other hand, for the unbalanced system, the ACLMS performs much better than CLMS. The ACLMS converges after 100 time steps, and it successfully estimate the frequency of interest. However, though the CLMS converges faster within 30 iterations, it oscillates around the wrong frequencies at $46Hz$ for unbalanced amplitude system, and $40Hz$ for unbalanced phase system. This can also be verified in learning curves, as the MSPE of ACLMS finally goes to below $-200dB$, while the MSPE of CLMS for both unbalnced systems are above $-20dB$. The outstanding performance of ACLMS is due to its additional degree of freedom in the widely linear model. It should be noticed that, the $g(n)$ term does not converge to zero for the unbalanced system, which means that frequencies estimated by strictly linear model and widely linear model are different. The bias in frequency estimated by the strictly linear model is due to the ignorance of conjugate part in the complex-valued voltage. Moreover, the larger the circularity, the larger the bias, as illustrated in Fig.38 and Fig.39. For the speed of convergence, it is because that the $|g(n)|$ does not converge to zero, leading to lower speed of ACLMS compared to CLMS.

To conclude, for balanced system, the CLMS is preferable for low computation and fast convergence. But for unbalanced systems, ACLMS is more proper for low estimation error.

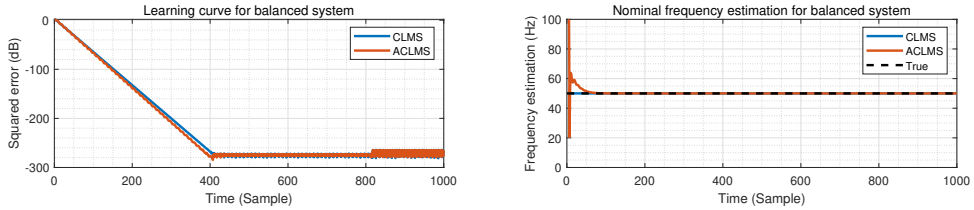


Figure 37: Learning curve and nominal frequency estimation for the balanced system Left: learning curve. Right: Nominal frequency estimation.

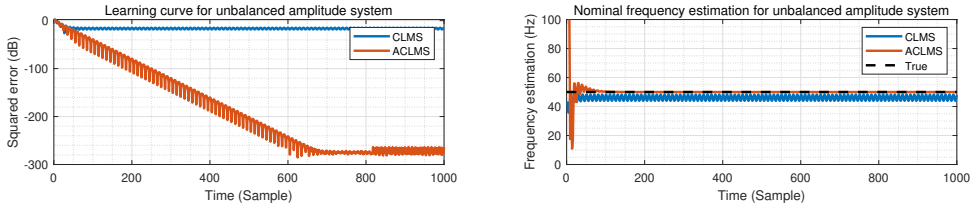


Figure 38: Learning curve and nominal frequency estimation for the unbalanced amplitude system. Left: learning curve. Right: Nominal frequency estimation.

3.2 Adaptive AR Model Based Time-Frequency Estimation

a) Fig.40 shows the frequency and phase of the frequency modulated (FM) signal. It it clear that the frequency and phase of FM signal are all time-variant, while the stationary signal requires the properties of the signal remains same over time. Therefore, FM signal is a non-stationary signal.

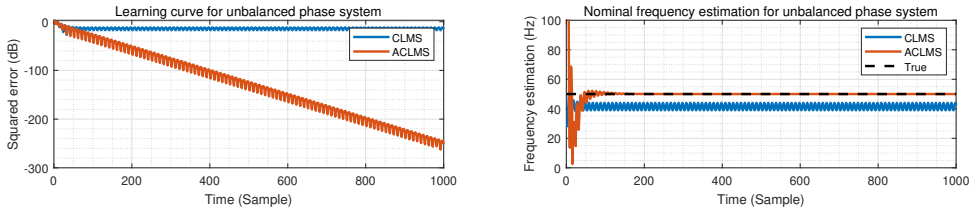


Figure 39: Learning curve and nominal frequency estimation for the unbalanced phase system. Left: learning curve. Right: Nominal frequency estimation.

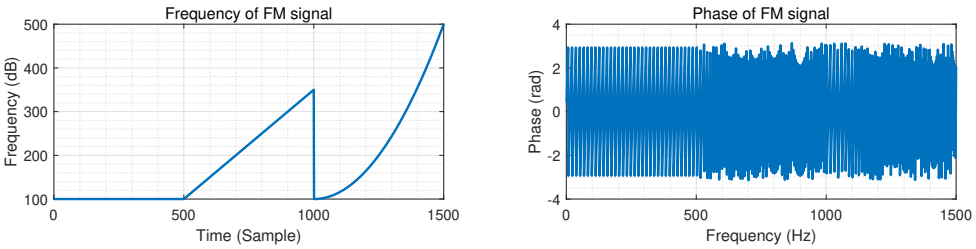


Figure 40: Frequency and phase of FM signal over time indexes Left: frequency. Right: phase.

Fig.41 shows the power spectrum of FM signal estimated by AR models with different orders. According to the left figure, AR(1) model does not adapt to the frequency-variant signal with only one peak (around $180Hz$) estimated. As model order increases, there are more peak, and for AR(25) the $100Hz$ components can be identified, but the variations in frequency are still not be correctly captured. Therefore, the estimation performance does not improve significantly with model order increasing. The reason is that the FM signal is non-stationary, so that the blocked-based estimate of the AR coefficients will be inaccurate.

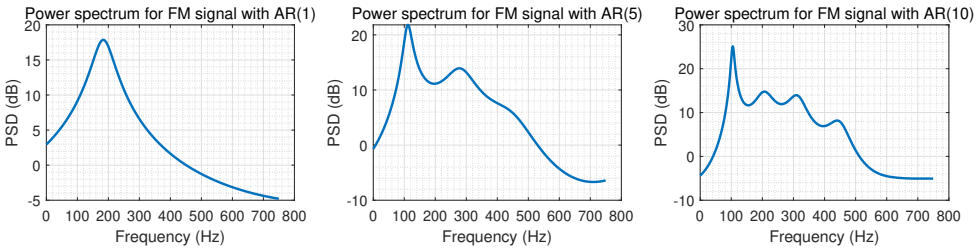


Figure 41: Estimate power spectrum of FM signal by AR model with different orders. Left: AR(1). Middle: AR(5). Right: AR(10)

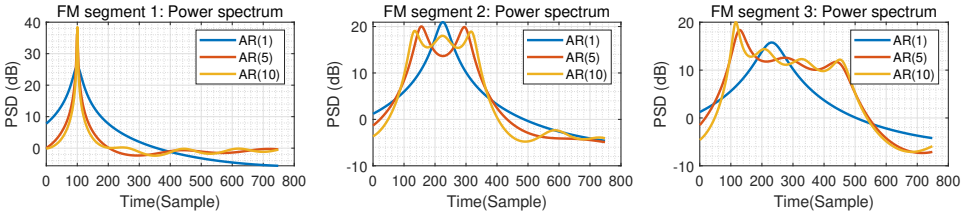


Figure 42: Estimate power spectrum of different segments in FM signal. Left: first segment. Middle: second segment. Right: third segment

Since the FM signal is composed of three segments, therefore, we can use AR model to estimate each part of the signal individually. We split the signal into segment with 500 elements, and test AR model estimation with three different model orders. The results are listed in Fig.42.

As Fig.42. shows, AR(1) can accurately capture the frequency element in the first range, since it is a constant signal that is stationary. However, for the linear second segment and quadratic third segment, AR(1) fails to adequately model the signals due to their non-stationary properties. Moreover, as order increases, there is still no significant improvement in capturing the changes in frequency of the signal in the second or third region, though more peaks occurs. Therefore, AR model can not be used to analyze the non-stationary signal.

b) Due to the limitations of AR model, we introduce the adaptive method for FM signal estimation. Because the signal is complex and highly circular ($|\rho| = 0.017$), we adopt the CLMS method for analysis. The time-frequency spectrum estimated by CLMS are shown in Fig.43.

The result suggests that the CLMS performs much better than AR model that successfully captures the changes in frequency of signal. Moreover, the step size in CLMS still needs to be carefully determined so that the trade-off between the convergence speed and steady-state error can be balanced. When the step size is small, such as $\mu = 0.001$, the model does not converge in time to the actual value. And the large step size ($\mu = 1$) leading to considerate oscillations around the optimal value. Also, the wider the spectra, the less the precision of estimation.

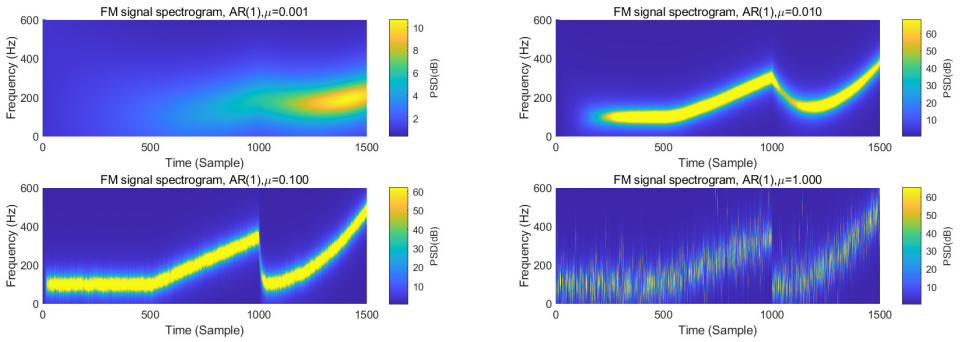


Figure 43: Time-frequency spectrum estimated by CLMS with different step sizes. (1):$\mu = 0.001$. (2): $\mu = 0.01$. (3): $\mu = 0.1$. (4): $\mu = 1$.

3.3 A Real Time Spectrum Analyser Using Least Mean Square

a) A signal $y(n)$ can be estimated with a combination of N harmonically related sinusoids, and it can be expressed as:

$$\hat{\mathbf{y}} = \mathbf{F}\mathbf{w} \tag{65}$$

where $\hat{\mathbf{y}}$ is the estimate of $y(n)$, $\mathbf{F}$ is the transformation matrix containing elements $F_N^{nm} = e^{j\frac{2\pi}{N}}(n)(m)$, $\mathbf{w}$ is the unknown weights vector, which can be estimated by the least squares (LS) method.

Expanding (65) gives:

$$\begin{bmatrix} \hat{y}(0) \\ \hat{y}(1) \\ \vdots \\ \hat{y}(N-1) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & F_N & F_N^2 & \cdots & F_N^{N-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & F_N^{N-1} & F_N^{2(N-1)} & \cdots & F_N^{(N-1)^2} \end{bmatrix} \begin{bmatrix} w(0) \\ w(1) \\ \vdots \\ w(N-1) \end{bmatrix} \tag{66}$$

The LS method aims to minimize the cost function $\mathcal{J}(\mathbf{w})$, which is defined as the sum of squared error between $y(n)$ and $\hat{y}(n)$. Therefore, $\mathcal{J}(\mathbf{w})$ is given by:

$$\begin{aligned} \mathcal{J}(\mathbf{w}) &= \|\mathbf{y} - \hat{\mathbf{y}}\|^2 = \|\mathbf{y} - \mathbf{F}\mathbf{w}\|^2 = (\mathbf{y} - \mathbf{F}\mathbf{w})^H(\mathbf{y} - \mathbf{F}\mathbf{w}) \\ &= \mathbf{y}^H\mathbf{y} - \mathbf{y}^H\mathbf{F}\mathbf{w} - \mathbf{w}^H\mathbf{F}^H\mathbf{y} + \mathbf{w}^H\mathbf{F}^H\mathbf{F}\mathbf{w} \end{aligned} \tag{67}$$

In order to minimize the cost function with respect to $\mathbf{w}$, we can take the first derivative of it, and find the optimal $\mathbf{w}_*$ that satisfies the equation:

$$\frac{\partial \mathcal{J}(\mathbf{w})}{\partial \mathbf{w}}\bigg|_{\mathbf{w}=\mathbf{w}_*} = 0 \tag{68}$$

Then, substituting (67) into (68) gives:

$$\frac{\partial}{\partial \mathbf{w}}\left(\mathbf{y}^H\mathbf{y} - \mathbf{y}^H\mathbf{F}\mathbf{w} - \mathbf{w}^H\mathbf{F}^H\mathbf{y} + \mathbf{w}^H\mathbf{F}^H\mathbf{F}\mathbf{w}\right)\bigg|_{\mathbf{w}=\mathbf{w}_*} = 0 \tag{69}$$

$$0 - \mathbf{F}^H\mathbf{y} - \mathbf{F}^H\mathbf{y} + 2\mathbf{F}^H\mathbf{F}\mathbf{w}_* = 0 \tag{70}$$

Since the column vectors of $\mathbf{F}$ are orthogonal, $\mathbf{F}$ and $\mathbf{F}^H\mathbf{F}$ are also full-rank. Moreover, $\mathbf{F}^H\mathbf{F} = N\mathbf{I}$, which illustrates that $\mathbf{F}^H\mathbf{F}$ is a symmetric diagonal matrix with positive values across the diagonal. Therefore, it is invertible, and the solution of $\mathbf{w}_*$ is:

$$\mathbf{w}_* = \left(\mathbf{F}^H\mathbf{F}\right)^{-1}\mathbf{F}^H\mathbf{y} \tag{71}$$

Also, a signal can be described using inversely discrete Fourier transform (IDFT):

$$\hat{x}(n) = \frac{1}{\sqrt{N}}\sum_{k=0}^{N-1} X(k)e^{j\frac{2\pi}{N}nk} = \frac{1}{\sqrt{N}}\sum_{k=0}^{N-1} X(k)F_N^{nk} \tag{72}$$

where $X(k)$ is the DFT coefficients, $F_N = e^{j\frac{2\pi}{N}}$. Then, we can rewrite the IDFT into vector form:

$$\begin{gathered}\hat{\mathbf{x}} = \mathbf{F}\mathbf{X} \\ \begin{bmatrix} \hat{x}(0) \\ \hat{x}(1) \\ \vdots \\ \hat{x}(N-1) \end{bmatrix} = \frac{1}{\sqrt{N}}\begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & F_N & F_N^2 & \cdots & F_N^{N-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & F_N^{N-1} & F_N^{2(N-1)} & \cdots & F_N^{(N-1)^2} \end{bmatrix}\mathbf{X}\end{gathered} \tag{73}$$

Therefore, the IDFT is also a linear transform, where the transformation matrix is composed of N harmonically related sinusoids. Since $\mathbf{F}$ is an unitary matrix, the DFT coefficients can be obtained as:

$$\mathbf{X} = \left(\mathbf{F}^H\mathbf{F}\right)^{-1}\mathbf{F}^H\hat{\mathbf{x}} = \mathbf{F}^H\hat{\mathbf{x}} \tag{74}$$

which has the same form as (71).

In conclusion, IDFT can also be expressed as a linear combination of N harmonically related sinusoids. The LS solution can also be seen as the Fourier coefficients of DFT. Since the LS method minimizes the squared error between the estimate signal and the true signal, IDFT can be regarded as an approximation of the original signal.

b) From aforementioned analysis, we conclude that the transformation matrix F of IDFT is orthogonal, as its column vectors are mutually orthonormal. If we define the vector of F is $\mathbf{f}_n = \frac{1}{\sqrt{N}}\left[1, \quad e^{j\frac{2\pi n}{N}}, \quad \ldots, \quad e^{j\frac{2\pi n(N-1)}{N}}\right]^T$, then the Hermitian inner product between two column vectors is given by:

$$\mathbf{f}_n^H\mathbf{f}_m = \begin{cases} 1 & \text{if } n = m \\ 0 & \text{otherwise} \end{cases} \tag{75}$$

Therefore, the DFT can be seen as the linear projection of a signal with N samples in the time domain into the N harmonically related sinusiods in the Fourier subspace, spanned by its orthogonal column vectors. The frequency of the sinusiods is a multiple of f_s/N, where f_s is the sampling

frequency. The operation changes the basis of $L_2 - norm$ into sinusoids. It should be noted that DFT just transforms the signal in one domain to another domain without increasing the information. The information contained may even be reduced if the sampling frequency is high, or the sample sizes in the time domain is small.

c) Fig.44 shows the time-frequency diagram of FM signal estimated by DFT-CLMS algorithm. When the leakage $\gamma = 0$, we find that the first segment of FM signal that is stationary with constant frequency has been successfully estimated. However, for the non-stationary parts, only the trends are roughly captured. It seems that there is a lasting effect, which means that once the frequency is picked up by the DFT-CLMS, it will remains forever and not update. As a result, the Fourier coefficients is superimposed over time. The reason for this effect is that DFT-CLMS is an adaptive algorithm, which calculates errors and updates weights corresponding to the current frequency components and ignores the previous components. Therefore, the error back-propagation mechanism is slow, the lasting effect will occur. Moreover, padding zero does not solves the problem, as shown in Fig.45 with $K = 2048$. In fact, the $K-point$ DFT-CLMS has the weight vector of dimension K, therefore, increasing K will lead to a slower error back-propagation and weights update, thus enhancing the everlasting effect.

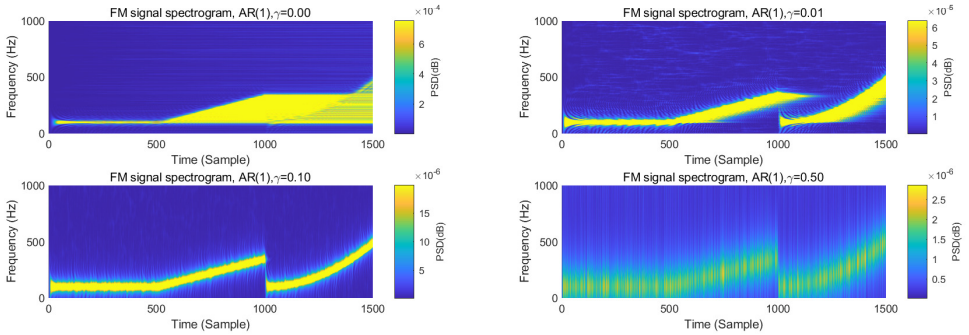


Figure 44: Time-frequency diagram estimated by DFT-CLMS with different leakage γ. (1):$\gamma = 0$. (2): $\gamma = 0.01$. (3): $\gamma = 0.1$. (4): $\gamma = 0.5$.

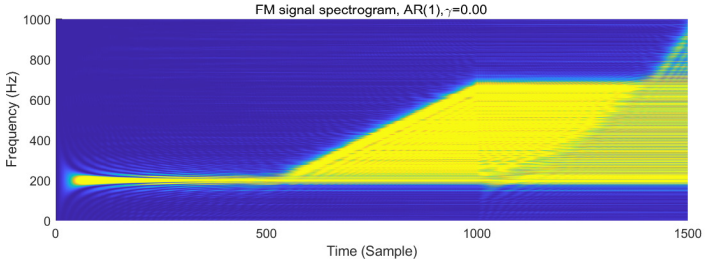


Figure 45: Time-frequency diagram estimated by DFT-CLMS with zero padding to 2048 points.

An alternative way to improve the accuracy of DFT-CLMS is to introduce the leakage coefficient. Notice that the leakage coefficients provides the forget mechanism that is useful for solving the everlasting problem. With leakage coefficient, the weights update equation becomes:

$$\mathbf{w}(n+1) = (1 - \gamma\mu)\mathbf{w}(n) + \mu e^*(n)\mathbf{x}(n) \tag{76}$$

As Fig.44 shows, DFT-CLMS with leakage coefficient performs much better than the standard method for estimating the non-stationary signal. However, there is a trade-off between the speed of error back-propagation and steady-state variance. When γ is small, the influence of previous weights cannot be properly eliminated, while the large γ results in large estimation variance. $\gamma = 0.1$ is a satisfying choice.

Compared the DFT-CLMS with AR-spectrum analyser shown in Fig.44 and Fig.43, we conclude that for stationary signal, the DFT-CLMS converges faster than AR model with less steady-state error. Moreover, it does not need to make a decision on the number of model order in advance. However, for non-stationary signal, the AR model estimates much more accurate than DFT-CLMS. After introducing the appropriate leakage coefficient into DFT-CLMS, the DFT-CLMS can outperform the AR model for non-stationary signal estimation with faster convergence and lower estimation variance.

d) Fig.46 presents the time-frequency plot of EEG signal estimated by DFT-CLMS. It is clear that the power-line interface of $50Hz$ as well as the fundamental and first harmonics of SSVEP ($13Hz$ and $26Hz$) have been clearly captured. The fundamental $13Hz$ component can be detected at about 400 iterations, and the accuracy of detection increases as iteration goes. The strong response within $8-10Hz$ is called alpha-rhythm caused by the tiredness of subject. The $39Hz$ component is undistinguished. However, the leaky DFT-CLMS does not perform well. This is due to the highly stationary property of POz signal, especially for a short segment. As shown before, the DFT-CLMS performs best for the stationary signal with fast convergence and little steady-state error. Nevertheless, the leaky DFT-CLMS is more proper for the non-stationary signal, since it adopts the forget mechanism. It will lose information contained in the stationary signal, thus increase the convergence time and introduce noise into spectrum. In this scenario, increasing the sample length can help to improve the performance.

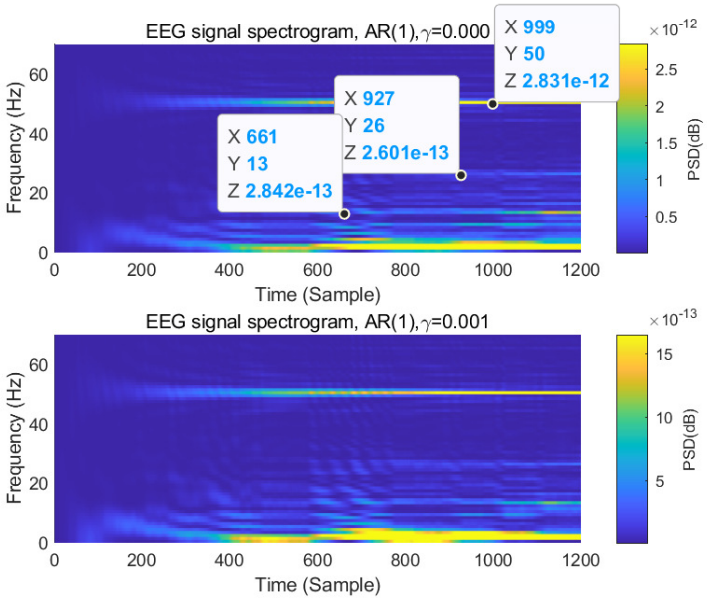


Figure 46: Time-frequency diagram of EEG signal estimated by DFT-CLMS with and without leakage. Top: $\gamma = 0$. Bottom: $\gamma = 0.001$

4 From LMS to Deep Learning

4.1 Linear LMS prediction for non-stationary signal

Fig.47 shows the zero-mean time-series and its one-step linear prediction by the standard LMS algorithm with AR(4) model. The LMS prediction converges to the true zero-mean signal after approximately 180 iterations with learning rate $\mu = 1x10^{-5}$. At first time steps, the LMS prediction is small, and it approaches closer and closer to the actual signal as time step increases. The MSE is larger at first and is decreasing with iteration increasing, which agrees with the characteristics of adaptive LMS. But due to the low convergence speed and the non-linearity of the signal, the absolute linear LMS prediction is always smaller than the truth even on the final iterations. The LMS provides a high

MSE of $16.03dB$ along with a low prediction gain R_p of $5.196dB$. Therefore, a more accurate model is needed for analyzing the non-stationary signal.

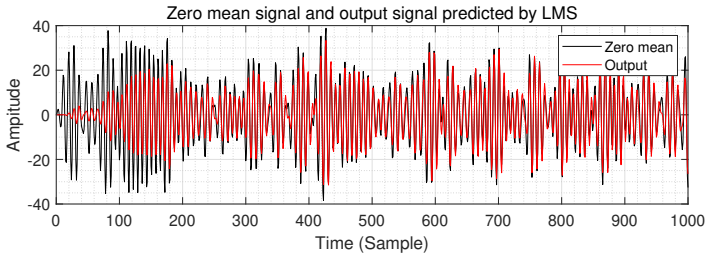


Figure 47: Zero-mean version and the one-step ahead prediction for time series

4.2 LMS with `tanh` activation function

Since generally the signal is non-linear, an non-linear function needs to be added to the output of LMS to make the model more expressive. In this section, we adopts the unit scale `tanh` activation function to perform on the zero-mean time series with learning rate $\mu = 1x10^{-5}$. Then the estimated signal is given by $\hat{\mathbf{y}} = tanh(\mathbf{w}^T\mathbf{x})$. Fig.48 shows the original zero-mean time-series and the unit-scale $tanh$ LMS estimated signal. Though the frequency variations has been successfully captured, the magnitude of the prediction is far from the truth. The reasons are that the unit scale `tanh` function maps any inputs from $(-\infty, +\infty)$ into $(-1, 1)$, therefore, it can not reflect the change in amplitude accurately due to the limited magnitude values. However, since `tanh` is a non-linear, smooth and continuous function, the non-linearity in frequency can be easily modelled. The unit-scale `tanh` has $MSE = 22.96dB$ and prediction error $R_p = -24.42dB$; therefore, its performance is worse than the standard LMS due to the magnitude difference. A more appropriate activation function is needed.

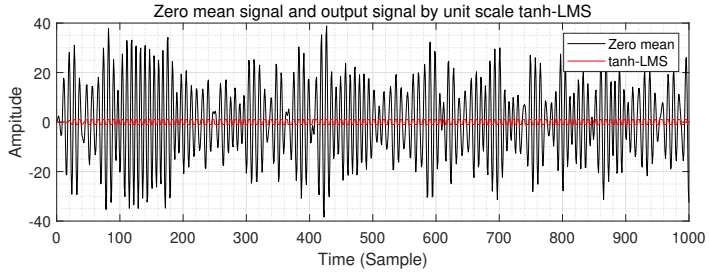


Figure 48: Zero-mean version and the one-step ahead prediction by unit-scale `tanh` LMS for time series

4.3 LMS with scaled `tanh` activation function

Adding a scale to the `tanh` function can significantly improve the performance of LMS, since it can maps the value into the range $(-a, a)$ with much more amplitudes. Fig.49 presents the predicted signal with different scaled `tanh` function along with the true signal. It can be seen that if a is small, the estimated result does not converge to the truth since the amplitude range is still not enough to cover the minimum and maximum value of the truth. Moreover, if a is too large, the over-fitting and instability will occur. For example, as shown in Fig.49 with $a = 100$, there is some oscillations at some instances due to the significant change in amplitude or frequency. Also, with scale, the weight update equation can be written as:

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu e(n) a[1 - tanh^2(\mathbf{w}^T(n)\mathbf{x}(n))]\mathbf{x}(n) \tag{77}$$

Therefore, the learning rate changes into μa, which means that increasing scale can increase the convergence speed, and if a is so large, the large step size will bring high steady-state error.

In order to choose the proper scale coefficient a, first, a should be larger than the maximum of time-series to enable the coverage of whole data range. That is $a > 38.81$. Moreover, since a increases the learning rate, then we have two conditions as 1) keeping μ constant to achieve various learning rate with a changing, or 2) changing μ corresponding to a to make the overall step size μa as a constant. For the first situation, we set $\mu = 1 \times 10^{-7}$, and the left plot in Fig.50 shows the MSE and the prediction gain R_p versus different scales. We can find that the optimal scale a is at 92, giving $MSE = 6.674dB$ and $R_p = 16.63dB$, which outperforms the standard LMS by 58.37% in error and 68.76 in gain. For the second condition, we keep the overall learning rate equal to 1×10^{-5}, then the comparison of MSE and R_p is shown in the right plot in Fig.50. The result suggests that the optimal a is at 87, which also achieves about 58% decease in error ($MSE = 6.663dB$) and 69% increase in gain ($R_p = 16.65dB$) compared with the standard LMS. In conclusion, the LMS with scaled tanh function highly improves the performance for non-linear signal estimation. The disadvantage for scaled tanh is that a cannot be determined without information about the signal.

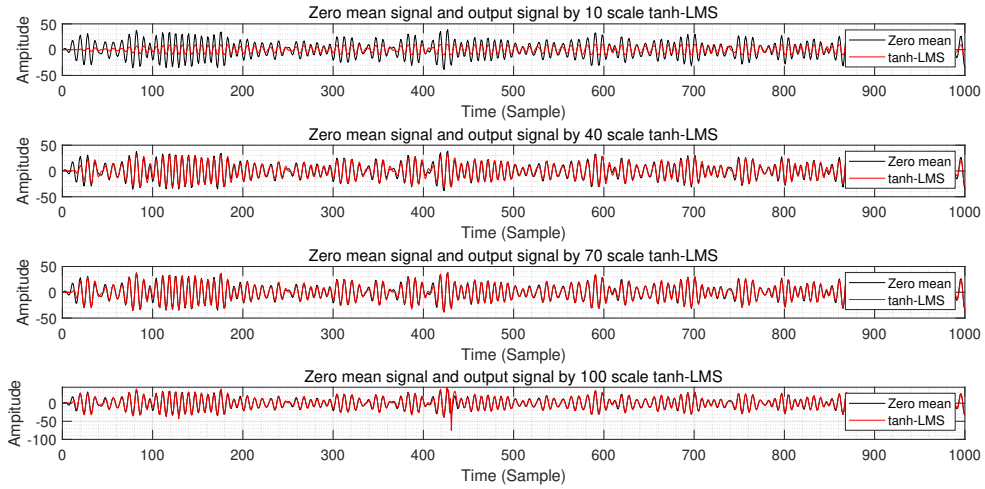


Figure 49: Zero-mean version and the one-step ahead prediction by different scaled tanh LMS for time series

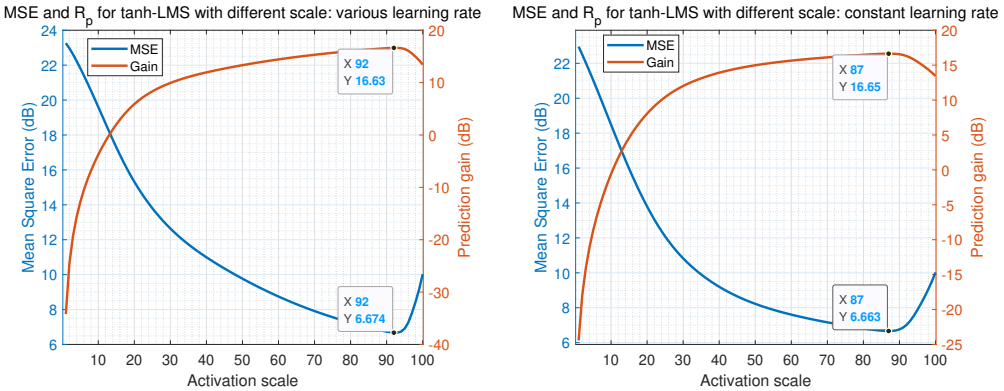


Figure 50: MSE and prediction gain for scaled tanh LMS. Left: with various learning rate. Right: with constant overall learning rate.

4.4 LMS with scaled `tanh` plus bias

To make prediction for the non-zero mean time-series, we add bias to the scaled `tanh` LMS. The output of the `tanh` LMS with scale and bias is given by: $\phi(\mathbf{w}^T\mathbf{x}+b)$. Generally, the bias can be implemented with augmented input to the LMS algorithm as $[1,\mathbf{x}]^T$ at each time step, where for a AR(4) model, $\mathbf{x}$ is the past 4 values of the input signal. Fig.51 shows the original non-zero mean signal and the predicted signal by scaled LMS with bias.

The result illustrates that with bias the model can successfully adapt the predicted data into the actual value with non-zero mean. But for a is small, the prediction fails with large error. The reason is that if a is small, the magnitude of the estimation is dominated by `tanh` function, which restricts the shift that moves the predicted mean to the accurate average value. Also, with large a, the instability and steady-steady error will be introduced. Moreover, the convergence speed of the algorithm with bias is faster than that without bias. It takes about 15 iterations for the biased model to converge, but 20 steps for the unbiased one. This is due to the extra task of estimating the DC offset. The MSE and R_p of the biased model are shown in Fig.52. The optimal scale is at $a = 76$ with $MSE = 10.17dB$ and $R_p = 13.41dB$, which is larger than the non-zero unbiased estimation. This is because of the additional estimation error of mean as the curse of an extra dimension.

Interestingly, even with the optimal scale, the amplitudes of the predicted result still does not reach the truth, which means that the predicted mean has a gap with the actual average level. This may due to the value 1 in the augmented value, which is extremely small compared with the amplitude of the time-series. Therefore, it cannot significantly improve the performance and capture the DC-offset correctly. The possible solutions for this problem are to normalised the data before prediction, or replace 1 in the augmented input by a larger value.

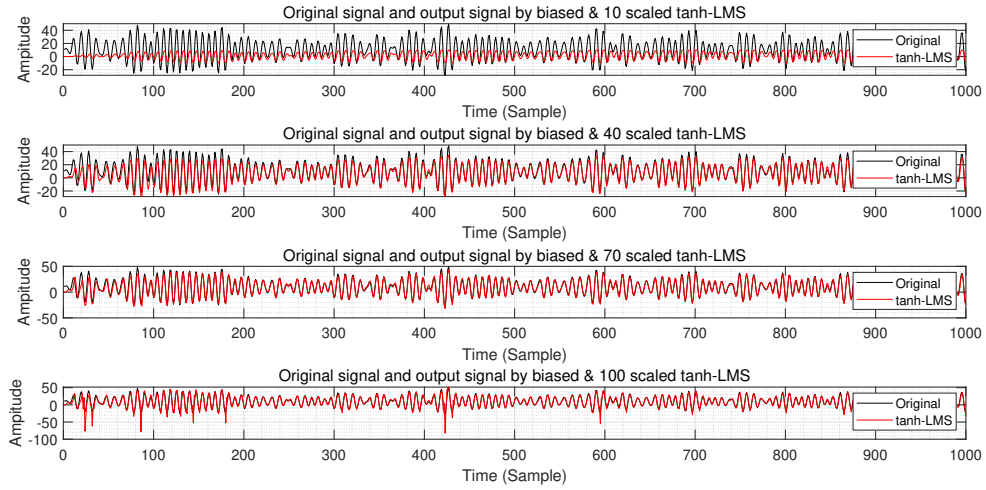


Figure 51: Non zero-mean version and the one-step ahead prediction by different scaled `tanh` LMS with bias for time series

4.5 Pre-train the weights

In this section, we over-fit the first 20 samples to get the pre-trained weight $\mathbf{w}_{init}$, and then use $\mathbf{w}_{init}$ as an initialisation for the latter prediction of non-zero mean signal. The original signal and the predicted result with bias and pre-trained weights is shown in Fig.53. It is clear that with pre-trained weights, the convergence is significantly faster within only 5 steps when $a = 40$. Moreover, the performance in terms of MSE and R_p has been highly improved with $8.07dB$ and $15.38dB$, respectively, as shown in Fig.54. The MSE is reduced by 20.65% compared with the model without pre-trained weights, and the gain R_p is 1.15 times greater than the previous model. Moreover, the mean of the predicted signal is much closer to the actual mean with only 4.4% bias. It should be

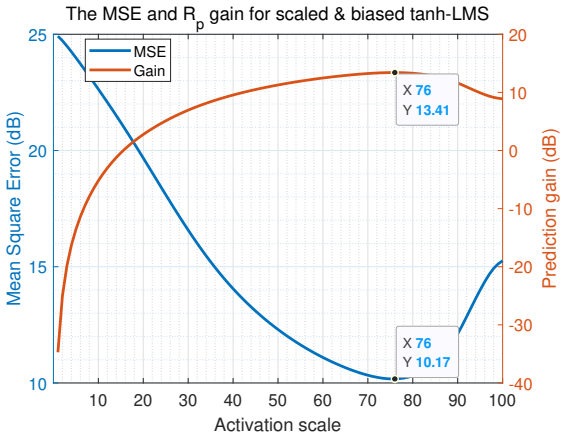


Figure 52: MSE and prediction gain for scaled `tanh` LMS with bias.

noticed that with pre-trained weights, a smaller scale is needed for a good prediction. If the scale a is large (i.e. 100), there is more overshoot in the result than the standard model. This is because that the pre-training phase have over-fit the first batch, leading the bias to converge to a larger value. In summary, pre-training weights can increase the convergence speed and prediction gain, while lower the estimation error; however, the scale a selected should be smaller than the standard method, and the batch size should be carefully considered.

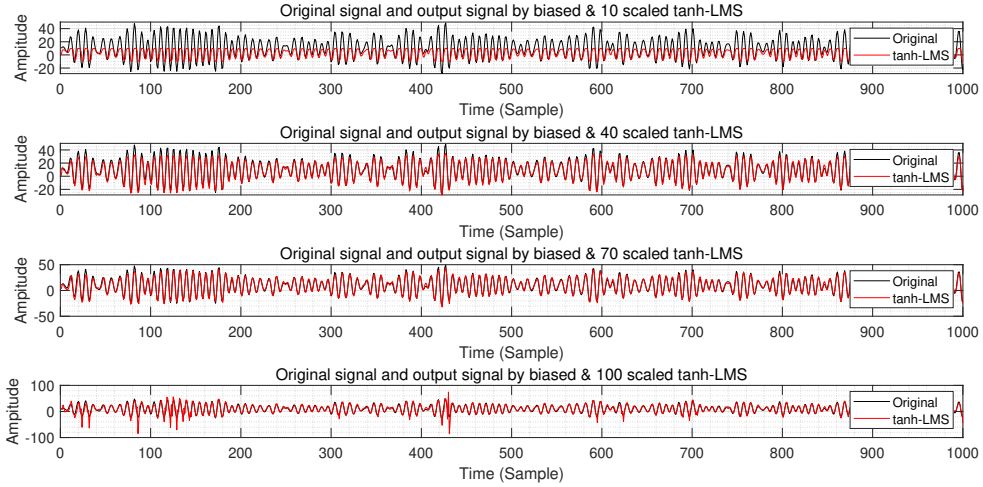


Figure 53: None zero-mean version and the one-step ahead prediction by different scaled `tanh` LMS with bias and pre-trained weights

4.6 Backpropagation algorithm

Since the `tanh` function is not expressive enough to model the high non-linear signal, the deep network constructed by many ”dynamical perception” models is a more popular way to process the real-world data. The backpropagation mechanism is used in the supervised learning of the neural network to define the rate that weights and bias need to change to minimize the error. That is, the backpropagation mechanism feeds the error back to the previous layer and update the weight and bias, so that the error in the next epoch can be reduced. Backpropagation adjusting the weight and bias based on the gradient-descent algorithm, which means to calculate the gradient of cost function

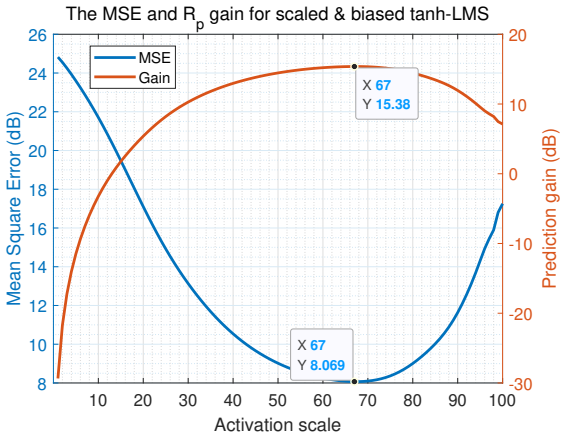


Figure 54: MSE and prediction gain for scaled `tanh` LMS with bias and pre-trained weights.

with respect to weights and bias.

Backpropagation can be used to train a deep network. The process contains six steps. First, the weights and bias are initialized to a small random value. Second, some past input samples (depends on the model order) will be received by the neural network, and is fed-forward through the network. Thirdly, the error and cost function between the predicted results and the desired outputs will be calculated. Fourth, the gradient of the cost function in the previous layer is calculated with respect to weights and biases. Fifth, the error and the gradient is back propagated to calculate the gradient of the cost function in all previous layers. Finally, the weights and biases are updated based on (78) and (78):

$$\mathbf{w}^k(n+1) = \mathbf{w}^k(n) + \mu\sigma^k\mathbf{g}^{k-1} \tag{78}$$

$$\mathbf{b}^k(n+1) = \mathbf{b}^k(n) + \mu\sigma^k \tag{79}$$

where σ^k is the activation function used in layer k, $\mathbf{g}^{k-1}$ is the output in the previous layer $k-1$. The step 2 to 7 will be repeated until the prediction converges to the satisfying result with the small error.

4.7 Performance of the deep network

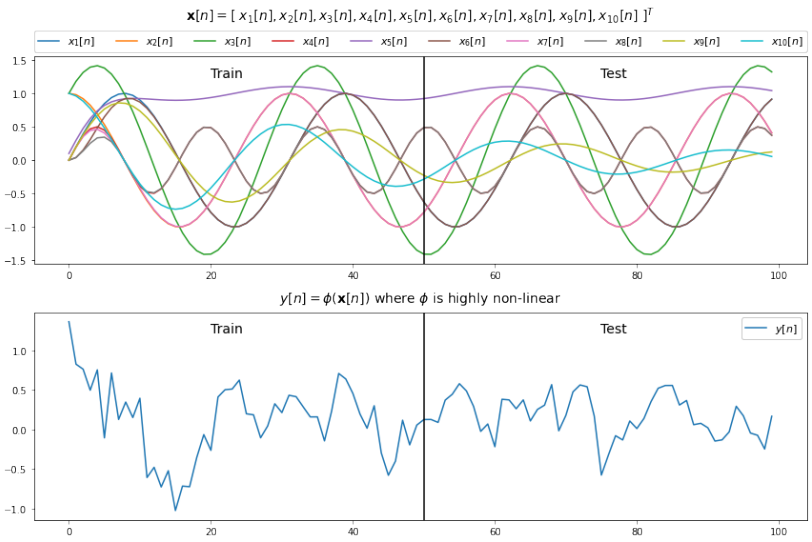


Figure 55: Top: input signal vector $\mathbf{x}(n)$. Bottom: desired output signal $\mathbf{y}(n)$ with noise corrupted.

Deep neural network is a quite useful tool to estimate the non-linear system. In this section, we train a 4-layer deep network with 20000 epochs and learning rate of 0.01 for the non-linear prediction problem. For comparison, the estimations by a linear single neuron, a tanh single neuron are also implemented. The activation function used in the Deep Network is the rectified linear unit (ReLU) rather than `tanh`. The reasons are that ReLU has faster convergence speed, lower computation complexity. Moreover, it can avoid the gradient vanishing due to derivatives of extreme large or small values, and is less likely for over-fitting. The desired signal is a combination of sinusoids with various amplitudes and frequencies constructed by a highly non-linear function $\phi(n)$ and corrupted with Gaussian white noise of power 0.05, as shown in Fig.55. Fig.56 presents the estimation results by all three methods along with the desired signal.

From the results, we can find that all three method have captured the trend of desire signal, but the single neuron method has much larger error than the deep network one. It is because that the deep network has more degree of freedom, and is more expressive to model the highly non-linear signals. But the estimation by neuron network forgets the small variations in the desired signal, and shows a smooth curve as the local average of the actual signal. The tanh single neuron outperforms the linear single neuron since tanh function allows the non-linear mapping from input to output. However, the performance improvement is not sufficient as there are a great deal of linear regions in the desired signal. Moreover, the weights of a single neuron is based on the Fourier series, therefore, the combination can not model the sudden change in the desired signal due to the Gibbs phenomenon. Also, since they don't have the back propagation mechanism, the estimation may be more dependent on the overall shape of inputs. For example, the first 20 samples are in an overall convex shape, then the estimation by a single neuron is a smooth convex that omits all information related to the concave points. Thus the predictions by a single neuron are all smooth sinusiods.

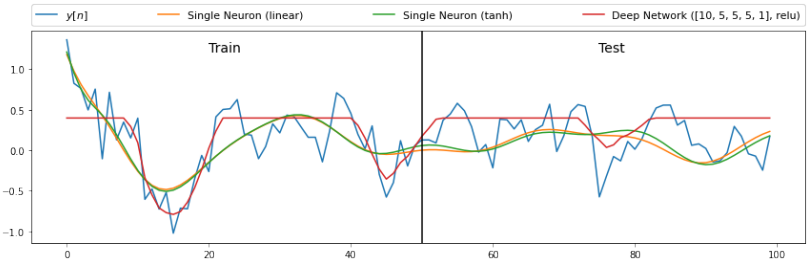


Figure 56: Predicted non-linear signal using single neuron (linear), single neuron (tanh) and the deep network (ReLU).

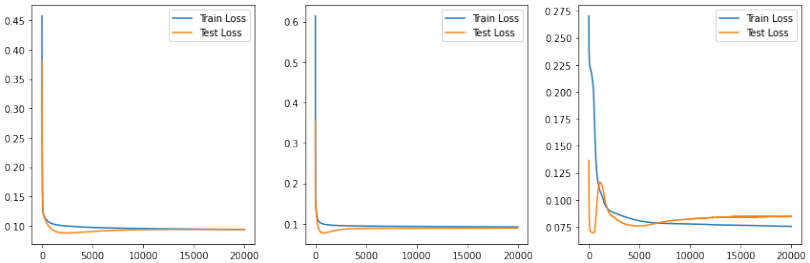


Figure 57: Learning curves using single neuron (linear), single neuron (tanh) and the deep network (ReLU).

Fig.57 shows the learning curves for three methods. The deep network has a smaller train and test loss as the minimum prediction error is 0.075. However, it takes around 10000 iterations to converge, while the linear and `tanh` single neuron take 7500 and 5000 respectively. This is because that in deep network, the weights is updated with error backpropagation that increases parameters caused by non-linearity and randomness due to stochastic gradient descent (SGD). Also, the non-linearity modelling capacity enables the `tanh` method converges faster than the linear one. Moreover, there are some

oscillations at the first time steps. The reason is that the model has over-fit the training data, which takes the random noise corrupted in the signal into consideration. Thus, for the test signal, the model parameters needs to fit the new noise leading to the oscillations and large errors. In comparison, the single neuron converges to 0.1 in a flat way, since they are less expressive to model the highly non-linear system, so they finally converge to a pseudo-optimum point in a dimension reduced signal subspace.

4.8 Deep network with different noise power

Figure 58 and 59 present the estimation and learning curves corresponding to the noise power $\sigma_n^2 = 0.01$. With a smaller noise the deep network achieves smaller prediction errors for both training and testing data. However the single neuron does not provide significant improvement on test data. The reason is that the training process of the deep network will over-fit the data by also considering the random noise. If the noise is small, the over-fit effect can be weakened, giving a better fit to the test data. But for the simple dynamical perception, it is based on the linear or weak linear system that can not be expressive enough to model the highly non-linear signal accurately. But if the nosie power is large ($\sigma_n^2 = 0.1$), as shown in Fig.60 and Fig.61, the deep network will over-fit the training data at a high level, leading to much larger loss at the test data, and is perform worse than the simple dynamical perception.

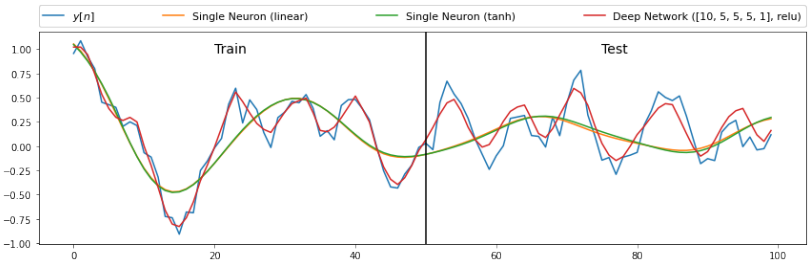


Figure 58: Predicted non-linear signal using single neuron (linear), single neuron (tanh) and the deep network (ReLU).

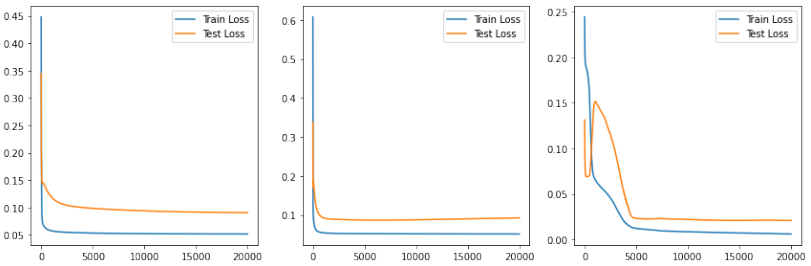


Figure 59: Predicted non-linear signal using single neuron (linear), single neuron (tanh) and the deep network (ReLU). Noise power is 0.01.

In summary, the deep network is more proper for the low noise non-linear signal modelling with less errors. But the convergence speed of it is much slower than the simple dynamical perception. Also, the feed-forward and error propagation mechanism leads to high computation complexity. The noise power corrupted in signal will also affects the performance of the deep network. A large noise power will lead to significant over-fitting that will cause deep network to predict highly inaccurate on the test data. There is a trade-off between the depth of the network and the response speed to the random noise. A deeper network is more likely to over-fit the training data, which will cause poor performance on the test data. However, a shallower network cannot capture all information in the highly complex non-linear signal, which limits its capability of modelling. Therefore, the power of noise and the depth of network needs careful considerations in reality.

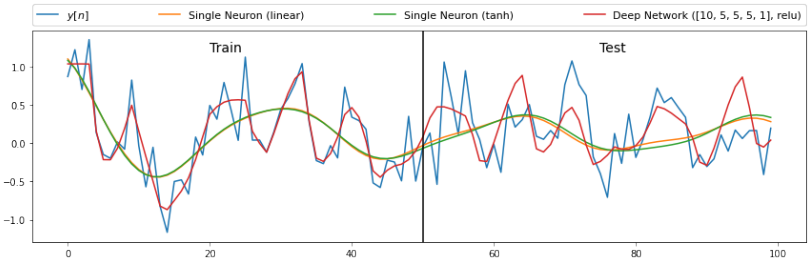


Figure 60: Learning curves using single neuron (linear), single neuron (tanh) and the deep network (ReLU). Noise power is 0.01

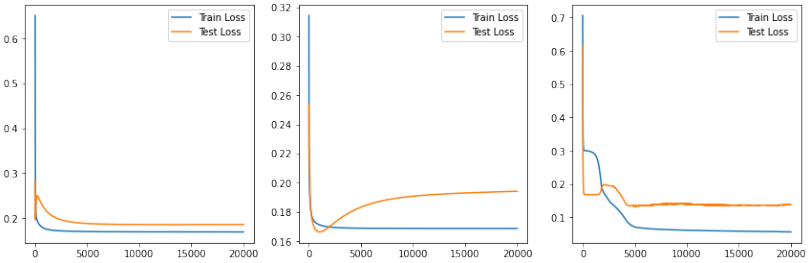


Figure 61: Learning curves using single neuron (linear), single neuron (tanh) and the deep network (ReLU). Noise power is 0.1.

References

[1] M. Hayes, Statistical digital signal processing and modelling. New York, N.Y.: Wiley, 1996. pages

[2] ”ReLu”, DeepAI, 2021. [Online]. Available: https://deepai.org/machine-learning-glossary-and-terms/relu. pages

[3] D. P. Mandic, ”Adaptive signal processing and machine intelligence,”, February 3, 2021. pages