Imperial College London

Imperial College London

Department of Electrical and Electronic Engineering

Adaptive Signal Processing and Machine Intelligence Coursework

Zhaolin Wang

An Assignment submitted for the ICL:

ELEC97003 - Adaptive Signal Processing and Machine Intelligence

April 6, 2021

Contents

1 Classical and Modern Spectrum Estimation

1.1 Properties of Power Spectral Density (PSD)

For a random sequence $x[n]$, its Power Spectral Density (PSD) is defined as

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

where $r(k)$ is the autocorrelation function (ACF) of the sequence $x[n]$, which is defined as

$$r(k) = \mathbb{E}\{x[n] x^*[n-k]\} \tag{2}$$

Apart from definition (1), the PSD can be defined as

$$P(\omega) = \lim_{N\to\infty} \mathbb{E}\left\{ \frac{1}{N} \left| \sum_{n=0}^{N-1} x(n) e^{-jn\omega} \right|^2 \right\} \tag{3}$$

if the ACF $r(k)$ decays rapidly, i.e.,

$$\lim_{N\to\infty} \frac{1}{N} \sum_{k=-(N-1)}^{N-1} |k| |r(k)| = 0 \tag{4}$$

Proof *Firstly, the following two vectors are defined*

$$\mathbf{x} = [x(0), x(1), \ldots, x(N-1)]^T \tag{5}$$

$$\mathbf{e} = \left[1, e^{-j\omega}, \ldots, e^{-j(N-1)\omega}\right]^T \tag{6}$$

The definition (3) *can be rewritten as*

$$\begin{aligned} P(\omega) &= \lim_{N\to\infty} \mathbb{E}\left\{ \frac{1}{N} \mathbf{e}^H \mathbf{x}\mathbf{x}^H \mathbf{e} \right\} \\ &= \lim_{N\to\infty} \frac{1}{N} \mathbf{e}^H \begin{bmatrix} r(0) & r(-1) & \cdots & r(-(N-1)) \\ \cdots & \cdots & \cdots & \cdots \\ r(N-1) & r(N-2) & \cdots & r(0) \end{bmatrix} \mathbf{e} \\ &= \lim_{N\to\infty} \sum_{k=-(N-1)}^{N-1} \frac{N-|k|}{N} r(k) e^{-j\omega k} \\ &= \sum_{k=-\infty}^{\infty} r(k) e^{-j\omega k} - \lim_{N\to\infty} \sum_{-(N-1)}^{N-1} |k| r(k) e^{-j\omega k} \end{aligned} \tag{7}$$

If the constrain (4) *is satisfied, the second term in* (7) *becomes zero, and definition* (3) *is equivalent to* (1). ■

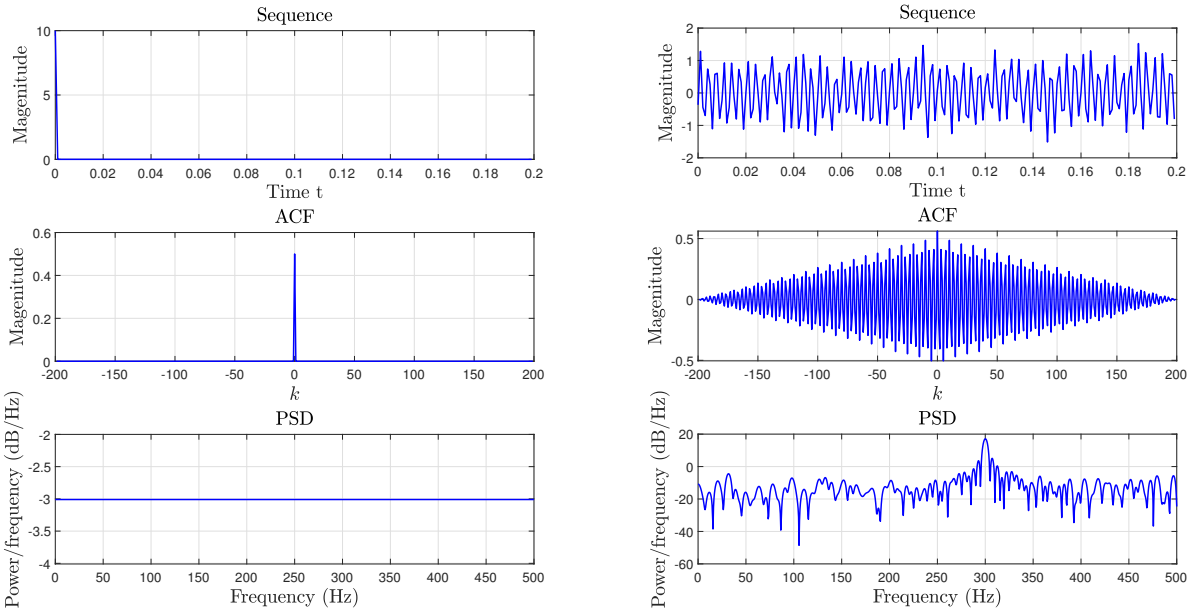
This equivalence can also be indicated by simulation. Consider the following impulse signal and sinusoidal signal:

$$x_1(t) = \delta(t) \tag{8a}$$

$$x_2(t) = \sin(2\pi \times 300t) + 0.2w(t) \tag{8b}$$

where $w(t)$ is the WGN signal.

Figure 1(a) shows the ACF and PSD of the impulse signal (8a) calculated by definition (3). As the ACF of the impulse signal decays very fast, the PSD defined by (3) is constant, which is the same as the theoretical one defined by (1). While for the sinusoidal signal (8b), which is shown in Figure 1(b), the PSD is not the same as the theoretical one as the ACF does not decay fast.



(a) impulse signal

(b) sinusoidal signal

Figure 1: ACF and PSD of the signal

1.2 Periodogram-based Methods Applied to Real–World Data

The Periodogram estimator is defined as

$$\hat{P}_{per}(\omega) = \frac{1}{N}\left|\sum_{k=0}^{N-1} x[k]\, e^{-j\omega k}\right|^2 \tag{9}$$

1.2.1 Task a: Sunspot Time Series

Figure 2(a) shows the comparison of the original and mean-removed sunspot data and their PSD. It is noticeable that the value of PSD at 0Hz is reduced by about 30dB. The mean value of the sequence contributes to the DC component, therefore the DC part (i.e., 0Hz) of the PSD is reduced.

Figure 2(b) shows the comparison of the original and trend-removed sunspot data and their PSD. In contrast to the mean-removed data, the DC component of the trend-removed data is almost totally removed (actually the magnitude at 0Hz of PSD is lower than -200dB).

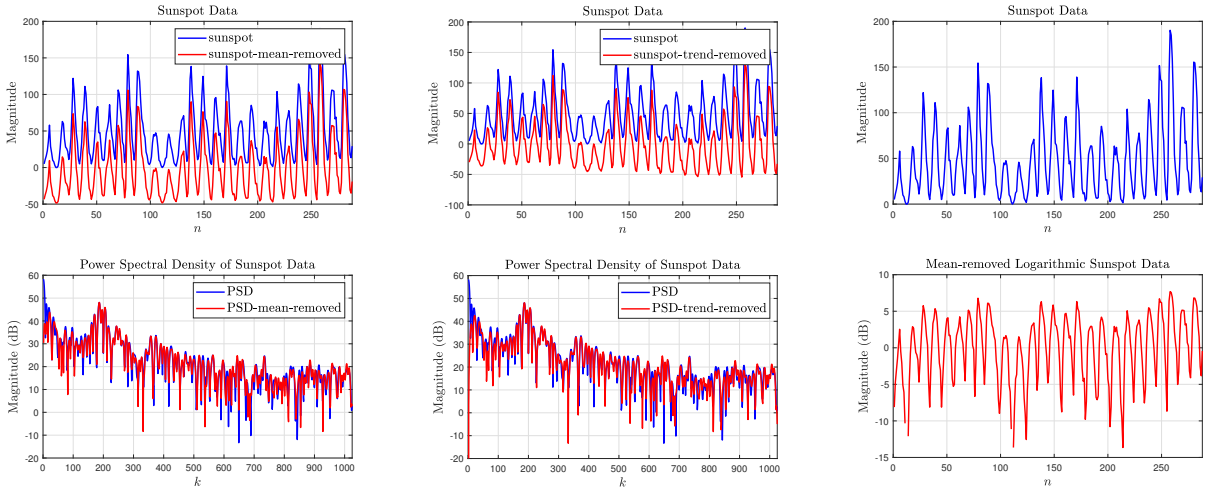
Figure 2(c) shows the comparison of the original and mean-removed logarithmic sunspot data. For the latter data, the periodicity is more evident, where the magnitude in dB oscillates roughly in the range between -10dB and 10dB and its period is around 10 samples.

1.2.2 Task b: EEG Signal

The Periodogram estimator in (9) is exploited to estimate the PSD of the EEG signal. The blue curve in Figure 3(a) shows the PSD between 11Hz and 20Hz estimated by standard Periodogram, which reveals an SSVEP peak at 13Hz.

Bartlett's method is a modified Periodogram approach, which is averaged Periodogram of non-overlapping segments. Three different window lengths (10 s, 5 s, and 1 s) are tested. To enable a fair comparison, the 5 DFT samples per Hz are kept for all the cases. Figure 3(a) shows the comparison of the standard Periodogram approach and Bartlett's method with a window length of 10s, where the SSVEP peak at 13Hz in the latter method is more distinguishable.

Nevertheless, Bartlett's method does not always benefit finding the SSVEP peaks, as the resolution of PSD is decreased with the decrease of window length. Figure 3(b) shows the effect of window length. In the case



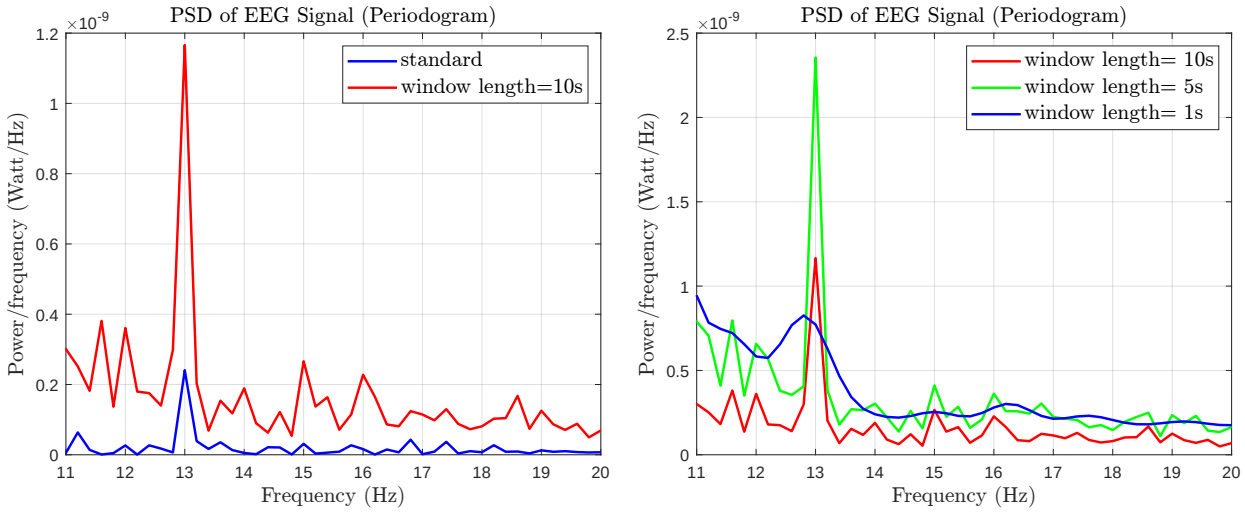
(a) Comparison of the original and mean-removed sunspot data (top) and their PSD (bottom)

(b) Comparison of the original and trend-removed sunspot data (top) and their PSD (bottom)

(c) Comparison of the original (top) and mean-removed logarithmic (bottom) sunspot data.

Figure 2: Analysis of the sunspot time series data

where the window length is 5s, the PSD result is better than the case where the window length is 10s. However, if the window length is very small, e.g., 1s in the figure, the peak at 13Hz disappears due to the low frequency resolution.



(a) Comparison of the standard periodogram approach and modified periodogram approach with windows length of 10s

(b) Comparison of Bartlett's method with various window lengths.

Figure 3: Standard Periodogram and Bartlett's Periodogram of the EEG signal

1.3 Correlation Estimation

Apart from the Periodogram-based method, the Correlogram method can also be applied to estimate the PSD, which is defined as

$$\hat{P}_{corr}(\omega) = \sum_{k=-(N-1)}^{N-1} \hat{r}(k) e^{j\omega k} \tag{10}$$

where $\hat{r}(k)$ is the biased or unbiased estimates of autocorrelation function (ACF).

1.3.1 Task a: Comparison of Biased and Unbiased Estimators

In order to compare the effect of biased and unbiased estimators of ACF in the Correlogram, the following two signals are considered:

$$x_3(t) = 0.7\sin(60\pi t) + \sin(120\pi t) + 0.3w(t) \tag{11a}$$

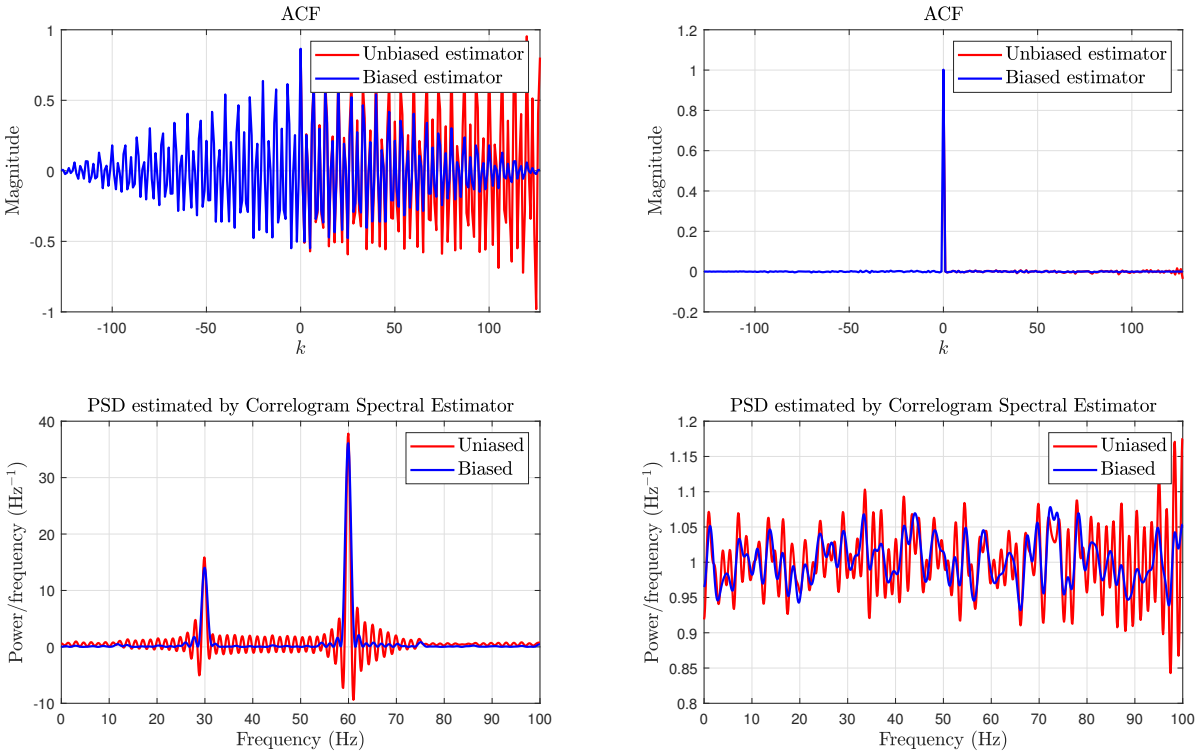
$$x_4(t) = w(t) \tag{11b}$$

where $w(t) \sim \mathcal{N}(0,1)$ is the WGN signal. These two signals are sampled with the sampling frequency $f_s = 200$Hz and $N = 128$ samples are obtained. For the second pure WGN signal, the averaged ACF and Correlogram PSD of 1000 realizations are calculated.

Figure 4(a) shows the ACF the PSD of biased and unbiased estimates of the noisy sinusoidal signal (11a). For the unbiased estimate of ACF, it is noticeable that the value is larger than that of the biased estimate for large lags. However, the unbiased estimate is subjected to high uncertainty for large lags, due to the fewer available samples.

On the other hand, the biased estimate applies low weights for the large lags, which reduces the uncertainty and makes the biased estimate more reliable. This can be indicated by the more oscillation in the PSD estimated by unbiased ACF, whose variance is higher. Furthermore, the unbiased ACF may result in the negative values of estimated PSD.

Figure 4(b) shows the average ACF of PSD of biased and unbiased estimates of 1000 realizations of the pure WGN signal (11b). Both PSD functions estimated by the biased and unbiased ACF oscillate around the theoretical PSD (i.e., $P(\omega) = 1$). However, the oscillation range of the unbiased estimate is higher than the biased one, which indicates the higher variance of the unbiased estimate again.



(a) ACF (top) and Correlogram PSD (bottom) of sampled signal (11a)

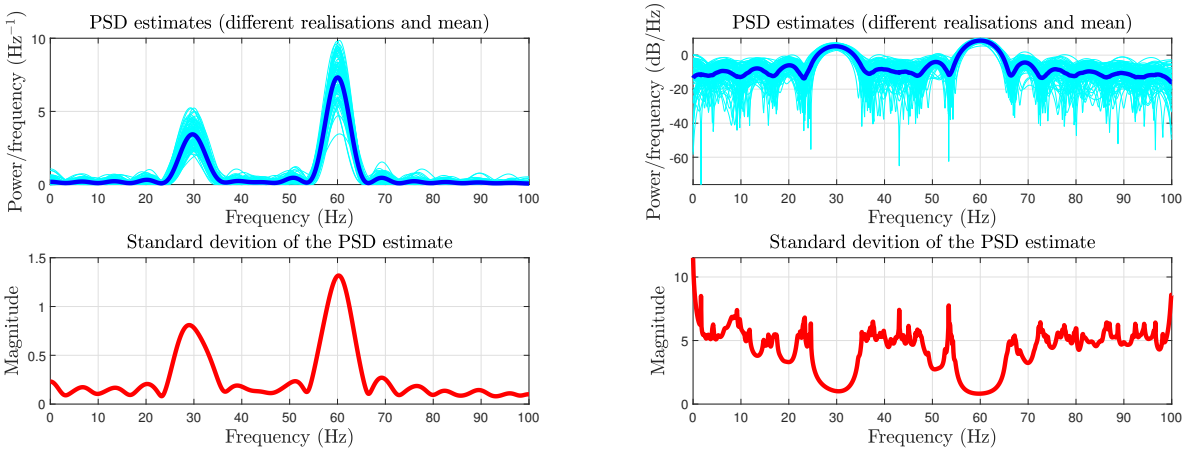
(b) ACF (top) and Correlogram PSD (bottom) of sampled signal (11b)

Figure 4: Comparsion of biased and unbiased estimator

1.3.2 Task b & c: Plotting in dB

The effect of plotting PSD in dB is investigated in this part. The biased ACF estimator of noisy sinusoidal signal (11a) is considered. In Figure 5(a), the PSD estimates of 100 realizations and their mean are not plotted

in dB. We can see that the different realizations of the spectral estimate are more disperse at the frequency near 30Hz and 60Hz, where the values of PSD are high. This phenomenon can also be indicated by the higher value of the standard deviation near 30Hz and 60Hz. However, the high-value part of PSD is usually what we are interested in.



(a) PSD estimates of signal (11a). Top: an overlay of 100 realizations and their mean. Bottom: standard deviation of the 100 realizations.

(b) PSD estimates of signal (11a) plotted in dB. Top: an overlay of 100 realizations and their mean. Bottom: standard deviation of the 100 realizations.

Figure 5: Comparison of standard plot and "dB" plot

Figure 5(b) plots the PSD estimates in dB and then the average PSD is calculated. In this case, the estimates of PSD are less spread out at the high value of PSD and the standard deviation is inversely proportional to the value of PSD. Therefore, there is less uncertainty at the positions that we are interested in the PSD.

1.3.3 Task d & e: Frequency Estimation by MUSIC

For the classic Periodogram approach, the frequency resolution of estimated PSD is limited by the length N of sequence, and the resolution is proportional to $1/N$.

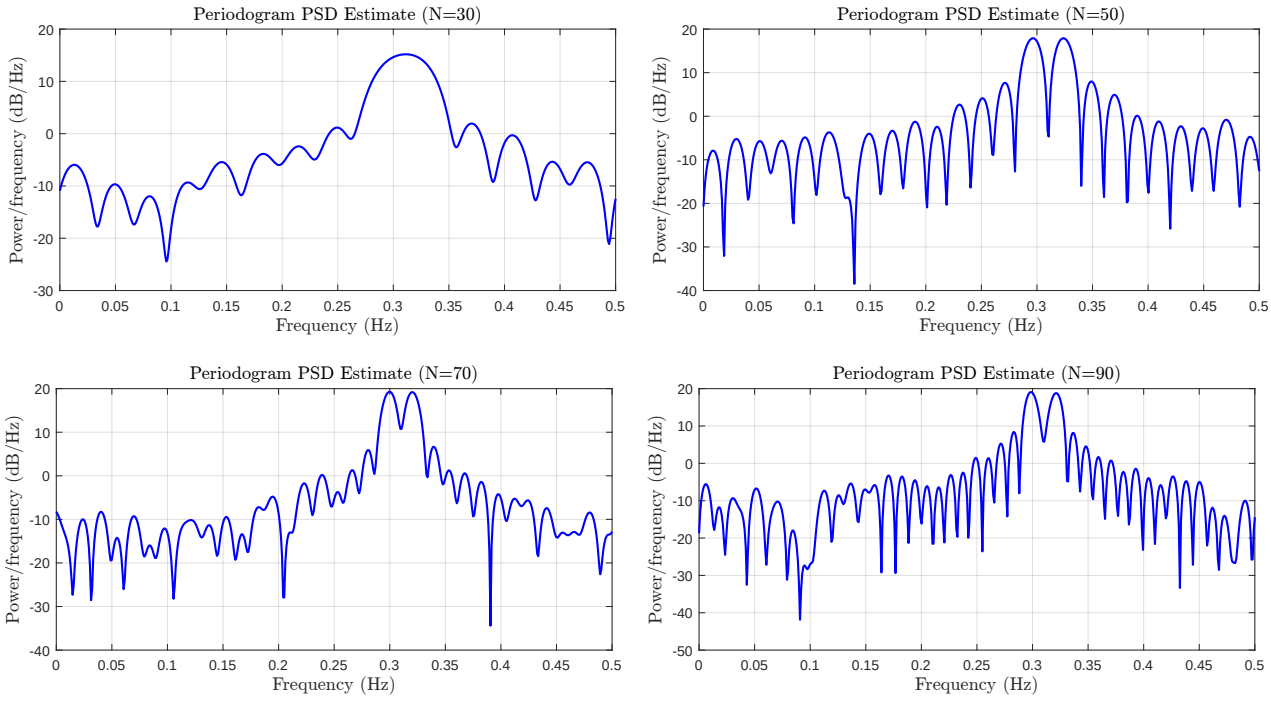


Figure 6: Periodogram of the complex exponential with different length

Figure 6 shows the Periodogram estimates of the following signal:

$$x_5(t) = e^{j2\pi\times 0.3t} + e^{j2\pi\times 0.32t} + 0.2n(t) \tag{12}$$

where $n(t) \sim \mathcal{CN}(0,1)$ is the complex Gaussian noise.

This signal is sample with $f_s = 1$Hz. In Figure 6, the two frequency components 0.3Hz and 0.32Hz cannot be distinguished when the length of sampled sequence is 30, due to the low frequency resolution. With the increase of the sequence length, the two frequency components start to be distinguishable and the Periodogram spectrum starts showing the correct line spectra of the signal (12), which has high main-lobe level at frequency 0.3Hz and 0.32Hz and low side-lobe level at other frequencies.

The MUSIC is a subspace method for spectral estimation, which is defined as

$$\hat{P}_{\text{MUSIC}}(\omega) = \frac{1}{\sum_{i=p+1}^{M} |\mathbf{e}^H \mathbf{v}_i|^2} \tag{13}$$

where M is the length of the signal sequence, p is the number of frequency components in the signal, $\mathbf{e} = \left[1, e^{j\omega}, \ldots, e^{j\omega(M-1)}\right]$ is the basis vector, and $\mathbf{v}_i$ is the eigenvector corresponding to the i-th eigenvalue ranking from the largest to the smallest. If ω_1 is one of the frequency components in the signal and $\mathbf{e}_1$ is the corresponding basis vector, we have

$$\mathbf{e}_1^H \mathbf{v}_i = 0 \tag{14}$$

Therefore, $\hat{P}_{\text{MUISC}}(\omega)$ will have a large value at the frequency component ω_1.

In this part, the following code is applied to find the desired line spectra of signal (12), which is sample at $f_s = 1$Hz and has a length of $N = 30$, using the MUSIC method.

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

The first line in the code is to estimate the covariance matrix of the sequence. The first input argument for the function `corrmtx` is the signal sequence $x[n]$. Let the second argument is denoted by M, then the first output of `corrmtx` with the $'$`mod`$'$ argument is a $2(N-M) \times (M+1)$ matrix, which is defined as

$$\mathbf{H} = \frac{1}{\sqrt{2(N-M)}} \begin{bmatrix} x(M+1) & \cdots & x(1) \\ \vdots & \ddots & \vdots \\ x(N) & \cdots & x(N-M) \\ x^*(1) & \cdots & x^*(M+1) \\ \vdots & \ddots & \vdots \\ x^*(N-M) & \cdots & x^*(M) \end{bmatrix} \tag{15}$$

and the second output of `corrmtx` is the covariance matrix, which is defined as

$$\mathbf{R} = \mathbf{H}^H \mathbf{H} \tag{16}$$

According to (16) $\mathbf{R}$ is an $(M+1) \times (M+1)$ matrix, each entry of which is calculated by averaging $2(N-M)$ samples to get the expected value.

The second line of the code is to calculate the line spectra defined in (13). The first argument of `pmusic` is the covariance matrix calculated by (16), the second argument is the number of frequency components p, the forth argument is the sampling frequency, and the last one is to indicate that the first argument should be the covariance matrix. For the outputs of `pmusic`, the first one is the estimated line spectra and the second one is the corresponding frequency interval.

Figure 7 shows the spectrum of (12) estimated by MUSIC, where the two frequency components at 0.3Hz and 0.32Hz can be clearly identified although the length of sequence is only $N = 30$. Therefore, the MUSIC method can still have good performance to identify the frequency components with small sequence length, which outperforms the Periodogram method. Nevertheless, the number of frequency components need to be known for the MUSIC, as it is based on the orthogonality of the frequency basis vector and the noise space vector (equation (14)).

Furthermore, the MUSIC method can be evaluated for any frequency, while for the Periodogram can only evaluate the frequency of the DFT bins. In this case, the MUSIC method has higher accuracy of the estimated spectrum.

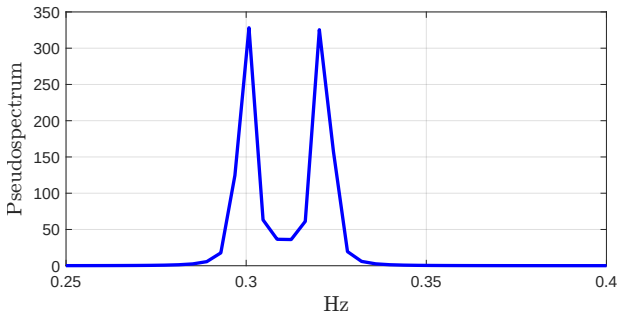


Figure 7: Spectrum estimated by MUSIC.

1.4 Spectrum of Autoregressive Processes

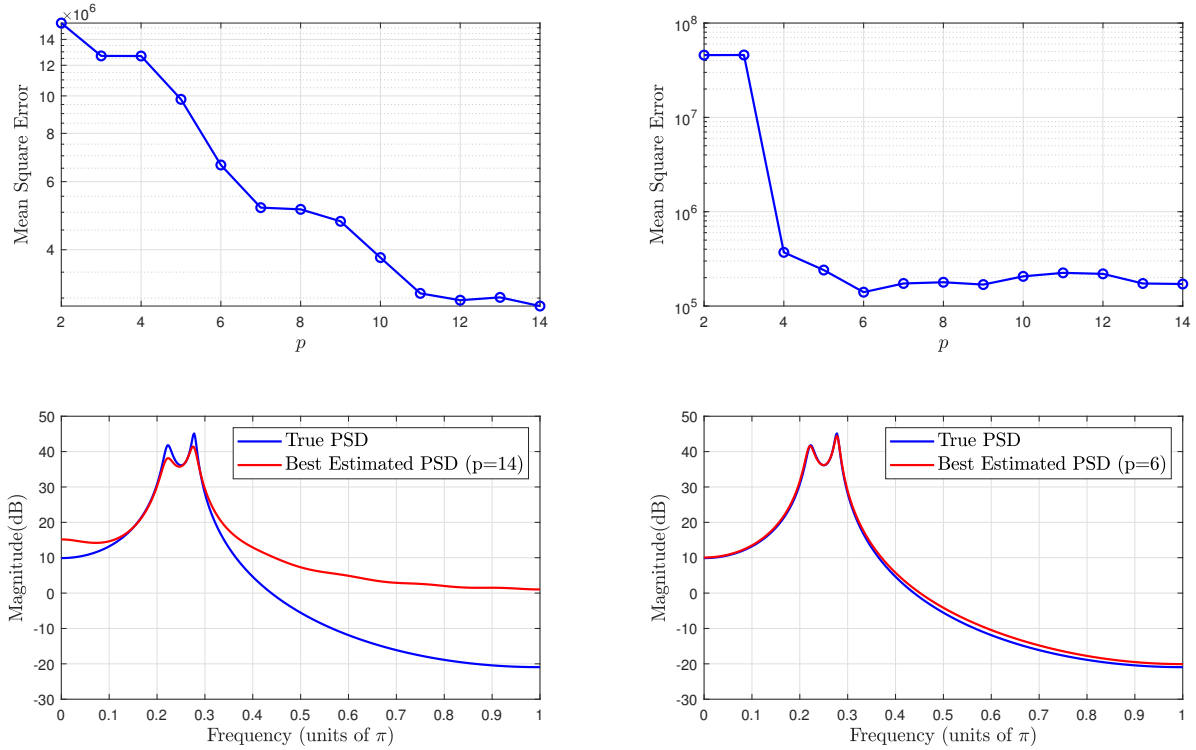
The model of an AR process is given as

$$x[n] = \sum_{k=1}^{p} a_k x[n-k] \tag{17}$$

For AR process, its power spectrum is given by

$$P_{\mathrm{AR}}(\omega) = \frac{\sigma_w^2}{\left|1 - \sum_{k=1}^{p} a_k e^{-jk\omega}\right|^2} \tag{18}$$

where σ_w^2 is the noise power, p is the order of AR process, and a_k are the AR coefficients. The σ_w^2 and a_k can be estimated using the Yule-Walker equations with biased ACF estimates or unbiased ACF estimates.



(a) Estimated PSD of AR process using different orders with 10^3 samples.

(b) Estimated PSD of AR process using different orders with 10^4 samples.

Figure 8: Estimated PSD of AR process

1.4.1 Task a: Shortcomings of Unbiased ACF

In the Yule-Walker equations, if the biased ACF estimate is applied, there can be high erratic for the large lags, which may lead to the high uncertainty of the estimated AR coefficients. Therefore, the biased estimate of the ACF is a better choice.

1.4.2 Task b & c: AR Spectrum Estimation

In this part, the following AR coefficient is considered

$$\mathbf{a} = [2.76, -3.81, 2.65, -0.92]^T \tag{19}$$

and the noise power is assumed to be $\sigma_w^2 = 1$.

Figure 8(a) shows the true PSD of the AR process and the best-estimated PSD ($p = 12$) using 10^3 samples, as well as the error of the estimated at each order. This figure indicates that by increasing the order of the assumed model, the error is generally reduced.

Figure 8(b) shows the results of 10^4 samples. It is noticeable that when the model order is lower (under-modeling) than the true order, i.e., $p = 4$, the error is dramatically decrease as the model order increase, while for the higher order (over-modeling), although the error generally decreases, the reduction of error is not obvious with the increase of model order.

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

In order to better observe the frequencies of respiration in the spectrum, the trend of the RRI data in the three trails is firstly removed using `detrend` to remove the DC component.

1.5.1 Task a & b: Periodogram Estimates of PSD

Figure 9(a) shows the standard Periodogram spectrum estimates for the RRI data in the three trials. Figure 9(b) and Figure 9(c) shows the average Periodogram spectrum with window lengths of 50s and 150s, respectively.

The main difference between the PSD estimates for the trails is that there are no obvious peaks in the PSD for the first trail while both the others have the obvious peaks. In the PSD for the second trial, the peak appears at the frequency of 0.4Hz, which matches the 25 breaths per minute in the trail. In the third trial, the peak is at around 0.12Hz, which matches the 7.5 breaths per minute in this trial.

1.5.2 Task c: AR Spectrum Estimates

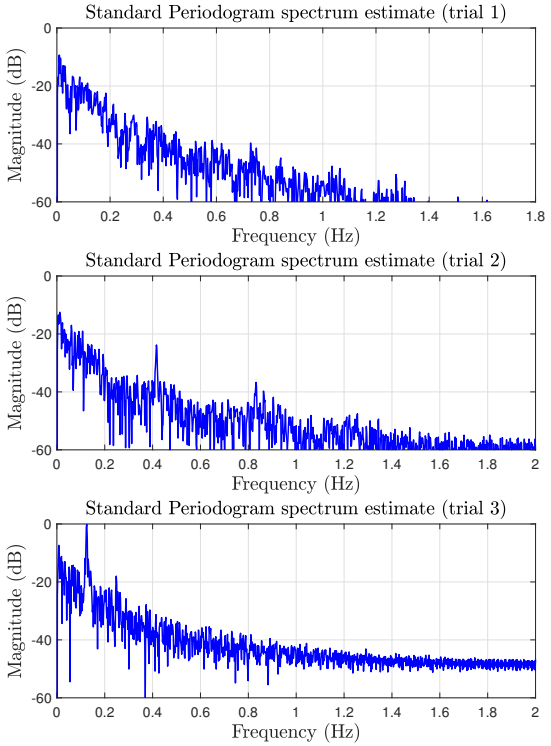
Figure 9(d) shows the AR spectrum estimates for the RRI in the three trails, where the order of the AR model is 40. We can see that the AR spectrum estimates is also capable of revealing the frequency peaks. Compared with the Periodogram estimates, there are two main difference:

- The spectrum power density at each frequency in the two kinds of spectrum estimates are different.
- The AR spectrum estimate is capable of showing the frequency components clearer than the Periodogram estimate.

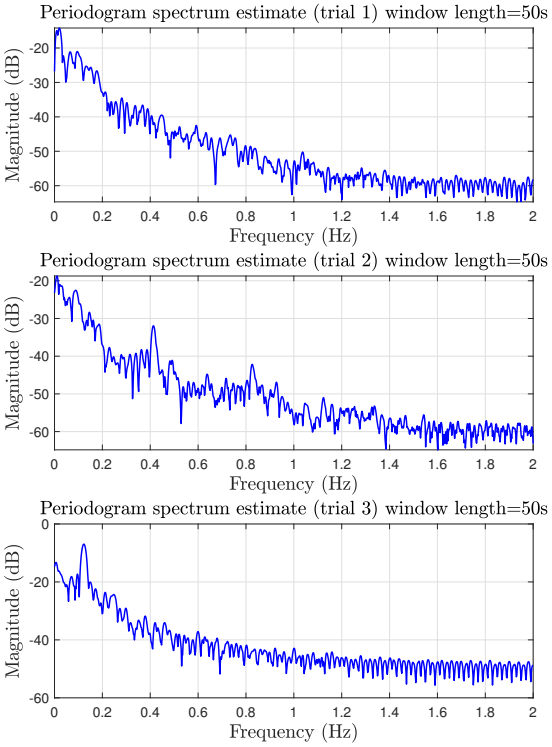
1.6 Robust Regression

1.6.1 Task a & b: Denoised Matrix

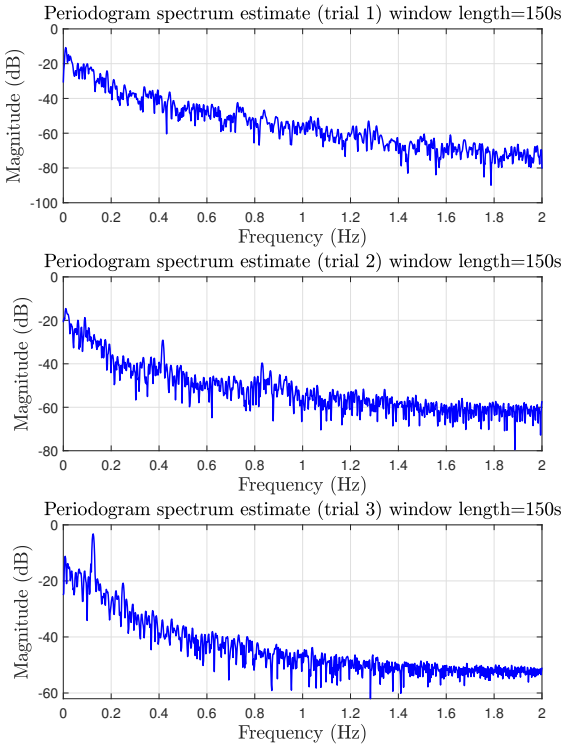
Figure 10(a) shows the singular values of the $\mathbf{X}$ and $\mathbf{X}_{noise}$ and the square errors between them. From the singular values of both $\mathbf{X}$ and $\mathbf{X}_{noise}$, it can be identified that the rank of the input data is 3, as the first 3 singular values are much larger than the rest. From the square error plotting, it is noticeable that the effect of the noise on the singular values is that it enlarges the zero singular values in the original input data. Therefore, if the noise makes all the singular values have the same value level, it becomes hard to identify the rank of the noisy matrix $\mathbf{X}_{noise}$.



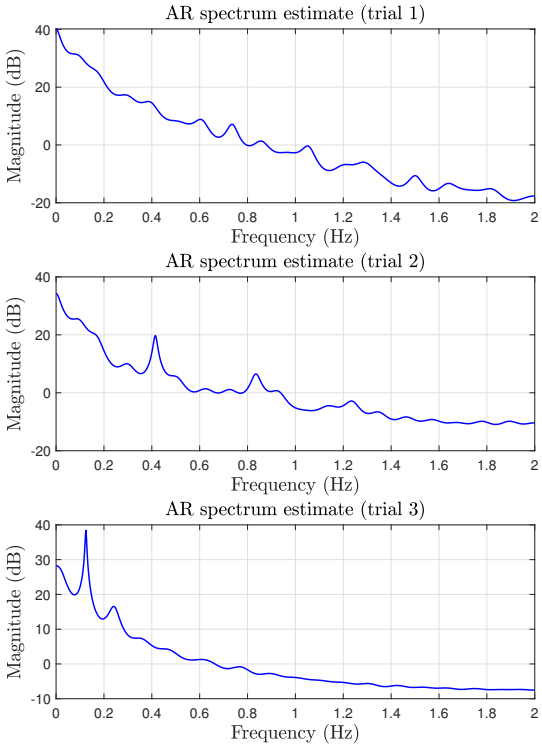
(a) Standard Periodogram spectrum estimates for the RRI data in the three trials.



(b) Averaged Periodogram spectrum estimates for the RRI data in the three trials.(windows length is 50s).

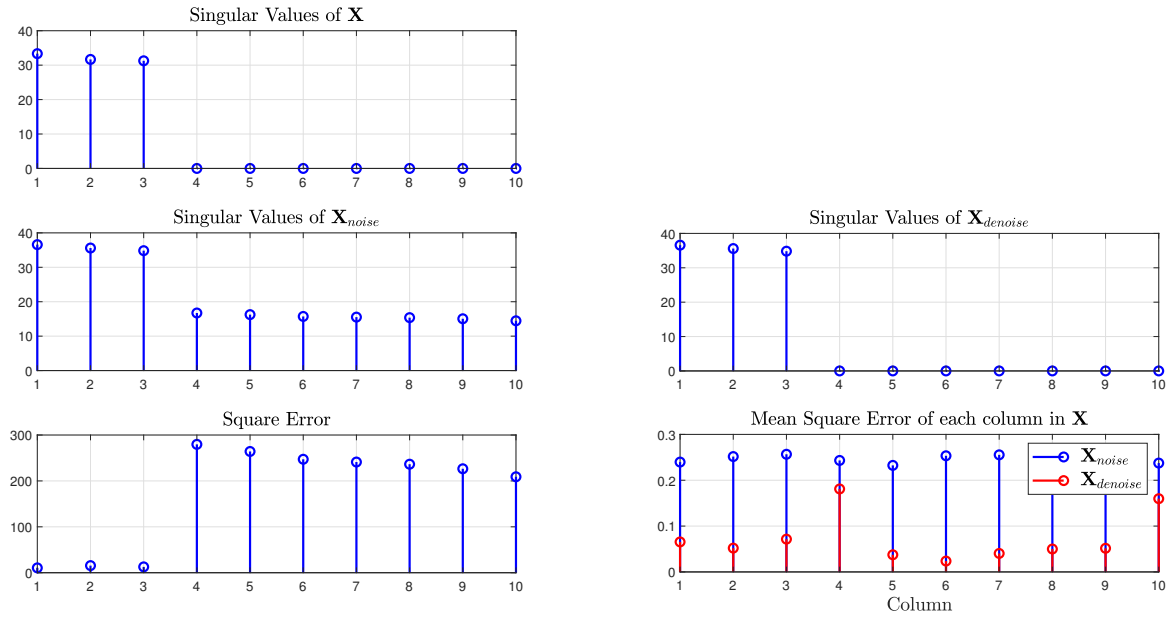


(c) Averaged Periodogram spectrum estimates for the RRI data in the three trials.(windows length is 150s).



(d) AR spectrum estimates for the RRI data in the three trials. (AR model order is 40).

Figure 9: Periodogram and AR spectrum estimates for the RRI data



(a) Singular values of the $\mathbf{X}$ and $\mathbf{X}_{noise}$ and the square errors between them.

(b) Singular values of the denoised matrix (top). The mean square error between the columns of the noiseless input matrix and the noisy and denoise matrices (bottom).

Figure 10: Analysis of the singular values of different data matrices

The $\mathbf{X}_{noise}$ can be denoised using only the r (rank of the matrix) most significant principal components. The denoised low-rank matrix $\tilde{\mathbf{X}}_{noise}$ is given by

$$\tilde{\mathbf{X}}_{noise} = \tilde{\mathbf{U}}\left(:, 1:r\right)\tilde{\mathbf{S}}\left(1:r, 1:r\right)\tilde{\mathbf{V}}^{H}\left(:, 1:r\right) \tag{20}$$

where $\tilde{\mathbf{U}}$, $\tilde{\mathbf{S}}$, and $\tilde{\mathbf{V}}$ are obtained by SVD of the $\mathbf{X}_{noise}$, i.e.,

$$\mathbf{X}_{noise} = \tilde{\mathbf{U}}\tilde{\mathbf{S}}\tilde{\mathbf{V}}^{H} \tag{21}$$

Figure 10(b) shows the singular values of the denoised matrix $\tilde{\mathbf{X}}_{noise}$, and indicates that this operation is capable of reducing the error with the noiseless data.

1.6.2 Task c & d: OLS and PCR

Table 1 shows the mean square error of the $\hat{\mathbf{Y}}_{\text{OLS}}$ and $\hat{\mathbf{Y}}_{\text{PCR}}$ with the $\mathbf{Y}$ on the provided training set and testing set. The last column of this table shows the averaged mean square error on the 1000 different testing sets.

In the training set, the OLS has a better performance than PCR on the training set, while has a worse performance than PCR on the testing set. The reason is that the OLS method also captures some noise characters, leading to overfit of the model on the training set. By testing the OLS and PCR model on 1000 different testing sets, we can conclude that the PCR method is more efficient on the out-of-sample data.

Table 1: Comparison of OLS and PCR methods

	OLS	PCR
Training set	0.7103	0.7154
Testing set	0.4755	0.4708
1000 testing set	0.4711	0.4674

2 Adaptive Signal Processing

2.1 The Least Mean Square (LMS) Algorithm

2.1.1 Task a: Convergence in Mean

The following 2-order AR process is considered

$$x(n) = a_1 x(n-1) + a_2 x(n-2) + \eta(n) \tag{22}$$

where $\eta(n) \sim \mathcal{N}(0, \sigma_\eta^2)$ is the Gaussian noise power. In order to estimate the coefficients a_1 and a_2 using LMS, the input vector is $\mathbf{x}(n) = [x(n-1), x(n-2)]^T$. Its correlation matrix is given as

$$\mathbf{R} = \mathbb{E}\left\{\mathbf{x}(n)\mathbf{x}^T(n)\right\} = \begin{bmatrix} \mathbb{E}\left\{x(n-1)x(n-1)\right\} & \mathbb{E}\left\{x(n-1)x(n-2)\right\} \\ \mathbb{E}\left\{x(n-2)x(n-1)\right\} & \mathbb{E}\left\{x(n-2)x(n-2)\right\} \end{bmatrix} = \begin{bmatrix} r_{xx}(0) & r_{xx}(1) \\ r_{xx}(1) & r_{xx}(0) \end{bmatrix} \tag{23}$$

The entries of the correlation matrix can be calculated by a set of 3 linear eqtions

$$\begin{bmatrix} r_{xx}(0) & r_{xx}(1) & r_{xx}(2) \\ r_{xx}(1) & r_{xx}(0) & r_{xx}(1) \\ r_{xx}(2) & r_{xx}(1) & r_{xx}(0) \end{bmatrix} \begin{bmatrix} 1 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} \sigma_\eta^2 \\ 0 \\ 0 \end{bmatrix} \tag{24}$$

If the original parameters are $a_1 = 0.1$, $a_2 = 0.8$ and $\sigma_\eta^2 = 0.25$, the solution of (24) is $r_{xx}(0) = 0.6966$, $r_{xx}(1) = -0.0387$ and $r_{xx}(2) = -0.5534$. Therefore, the correlation matrix is

$$\mathbf{R} = \begin{bmatrix} 0.6966 & -0.0387 \\ -0.0387 & 0.6966 \end{bmatrix} \tag{25}$$

Denote the estimated AR coefficients at step n by $\mathbf{w}(n) = [\hat{a}_1(n), \hat{a}_2(n)]^T$. The LMS algorithm will minimize the cost function $J(n) = \frac{1}{2}e^2(n)$ by updating $\mathbf{w}(n)$ according to

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

where $e(n) = x(n) - \mathbf{w}^T(n)\mathbf{x}(n)$. To ensure the convergence of the LMS in mean, the step size μ needs to stisfy

$$0 < \mu < \frac{2}{\mathrm{Tr}\{\mathbf{R}\}} \tag{27}$$

where $\mathrm{Tr}\{\mathbf{R}\}$ is trace of the covariance matrix $\mathbf{R}$Therefore, the range of the step size μ is

$$0 < \mu < 1.44 \tag{28}$$

2.1.2 Task b: Step Size

Figure 11 shows the squared error and average squared error (over 100 realizations) with step size of 0.05 and 0.01, respectively. According to the learning curve (plots of the average square error), the LMS starts to converge at around 100 steps when $\mu = 0.05$ while it starts to converge at around 200 steps when $\mu = 0.01$, which indicates that the LMS can converge faster when the step size is larger.

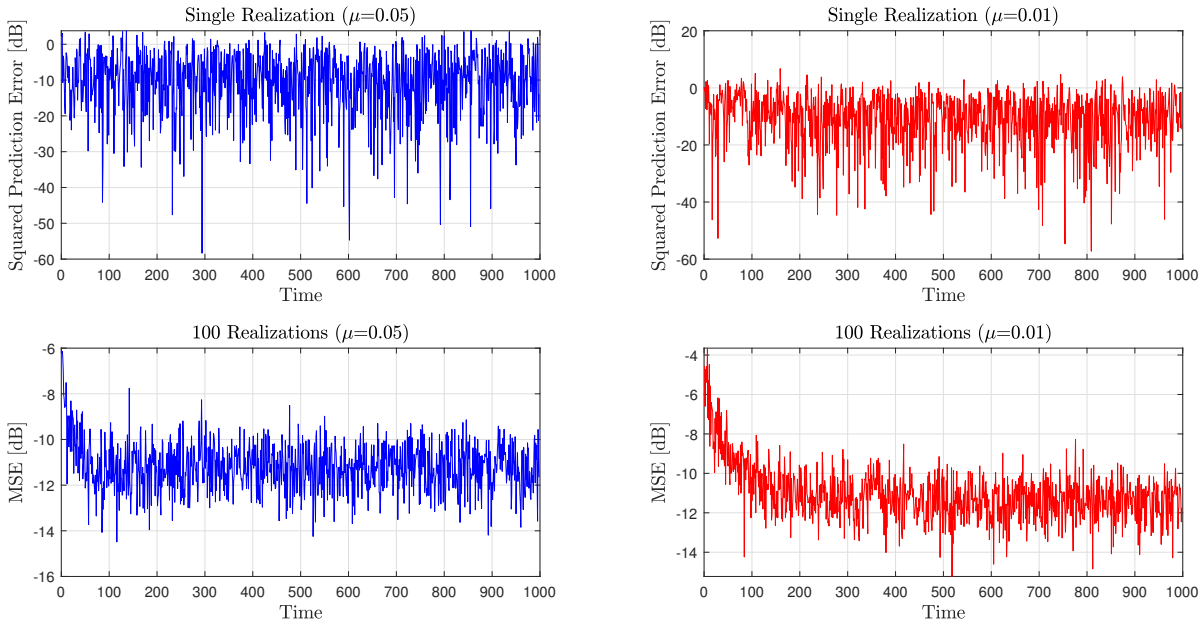
2.1.3 Task c & d: Excess Error

For small step-size, the misadjustment of the LMS is approximated as

$$\mathcal{M}_{\mathrm{LMS}} \approx \frac{\mu}{2}\mathrm{Tr}\{\mathbf{R}\} \tag{29}$$

According to the result in (25), the theoretical misadjustment for $\mu = 0.05$ and $\mu = 0.01$ should be $\mathcal{M}_{0.05} \approx 0.035$ and $\mathcal{M}_{0.01} \approx 0.007$, respectively.

The estimated misadjustment is obtained by time-averaging over the steady state of the ensemble-averaged learning curves using 100 independent trials of the experiment. For both $\mu = 0.05$ and $\mu = 0.01$, 10 estimated



(a) $\mu = 0.05$ (b) $\mu = 0.01$

Figure 11: The value of the $e^2(n)$ (dB) (top) and the average $e^2(n)$ (dB) of 100 different realizations (bottom)

values of misadjustment are calculated by repeating the aforementioned process 10 times, which are shown below

$$\hat{\mathcal{M}}_{0.05} = [0.040 \quad 0.045 \quad 0.054 \quad 0.039 \quad 0.050 \quad 0.068 \quad 0.055 \quad 0.051 \quad 0.059 \quad 0.061] \tag{30}$$

$$\hat{\mathcal{M}}_{0.01} = [0.014 \quad 0.018 \quad 0.016 \quad 0.006 \quad 0.022 \quad 0.004 \quad 0.013 \quad 0.021 \quad 0.007 \quad 0.014] \tag{31}$$

According to the estimated results, we can see that the estimated misadjustment is usually larger that the theoretical one. However, for the smaller step size (i.e., $\mu = 0.01$), the estimates are closer to the theoretical values.

The estimated AR coefficients $\hat{a}_1$ and $\hat{a}_2$ are also obtained by averaging the steady-state values of 100 trials of the experiment, which are shown in Table 2. It is obvious that for the smaller step size (i.e.,$\mu = 0.01$), the estimated coefficients have smaller error. The reason is that the smaller step size leads to the smaller excess error.

Table 2: Comparison of estimated AR coefficients using different step size

Value	$\mu = 0.05$	$\mu = 0.01$
$\hat{a}_1$	0.071	0.111
$\hat{a}_2$	0.716	0.748
$\lvert\hat{a}_1 - a_1\rvert$	0.029	0.011
$\lvert\hat{a}_2 - a_2\rvert$	0.084	0.052

2.1.4 Task e & f: The Leaky LMS Algorithm

In order to derive the Leaky LMS coefficient update for $\mathbf{w}(n)$, we can first consider the Steepest Descent methods, where the following cost function is minimized

$$\begin{aligned}
\tilde{J}_2(n) &= \mathbb{E}\{J_2(n)\} \\
&= \frac{1}{2}\mathbb{E}\left\{e^2(n) + \gamma\|\mathbf{w}(n)\|_2^2\right\} \\
&= \frac{1}{2}\mathbb{E}\left\{e^2(n)\right\} + \frac{1}{2}\gamma\|\mathbf{w}(n)\|_2^2 \\
&= \frac{1}{2}\mathbb{E}\left\{\left(x(n) - \mathbf{w}^T(n)\mathbf{x}(n)\right)\left(x(n) - \mathbf{w}^T(n)\mathbf{x}(n)\right)^T\right\} + \frac{1}{2}\gamma\mathbf{w}^T(n)\mathbf{w}(n) \\
&= \frac{1}{2}\mathbb{E}\left\{x^2(n) - 2x(n)\mathbf{x}^T(n)\mathbf{w}(n) + \mathbf{w}^T(n)\mathbf{x}(n)\mathbf{x}^T(n)\mathbf{w}(n)\right\} + \frac{1}{2}\gamma\mathbf{w}^T(n)\mathbf{w}(n) \\
&= \frac{1}{2}\left(\underbrace{\mathbb{E}\{x^2(n)\}}_{\sigma_x^2} - 2\mathbf{w}^T(n)\underbrace{\mathbb{E}\{\mathbf{x}(n)x(n)\}}_{\mathbf{p}} + \mathbf{w}^T(n)\underbrace{\mathbb{E}\{\mathbf{x}(n)\mathbf{x}^T(n)\}}_{\mathbf{R}}\mathbf{w}(n)\}\right) + \frac{1}{2}\gamma\mathbf{w}^T(n)\mathbf{w}(n) \\
&= \frac{1}{2}\sigma_x^2 - \mathbf{w}^T(n)\mathbf{p} + \frac{1}{2}\mathbf{w}^T(n)\mathbf{R}\mathbf{w}(n) + \frac{1}{2}\gamma\mathbf{w}^T(n)\mathbf{w}(n)
\end{aligned} \tag{32}$$

In Steepest Descent, $\mathbf{w}(n)$ is updated in the direction of the negative gradient of cost function $\tilde{J}_2(n)$. The gradient is given as

$$\begin{aligned}
\nabla\tilde{J}_2(n)|_{\mathbf{w}(n)} &= -\mathbf{p} + \frac{1}{2}\left(\mathbf{R} + \mathbf{R}^T\right)\mathbf{w}(n) + \gamma\mathbf{w}(n) \\
&= -\mathbf{p} + \mathbf{R}\mathbf{w}(n) + \gamma\mathbf{w}(n)
\end{aligned} \tag{33}$$

Therefore, the coefficient update in Steepest Descent methods is

$$\begin{aligned}
\mathbf{w}(n+1) &= \mathbf{w}(n) + \mu\left(-\nabla\tilde{J}_2(n)|_{\mathbf{w}(n)}\right) \\
&= \mathbf{w}(n) + \mu\left(\mathbf{p} - \mathbf{R}\mathbf{w}(n) - \gamma\mathbf{w}(n)\right) \\
&= (1 - \mu\gamma)\mathbf{w}(n) + \mu(\mathbf{p} - \mathbf{R}\mathbf{w}(n))
\end{aligned} \tag{34}$$

In the LMS, we replace $\mathbf{p}$ and $\mathbf{R}$ by the instantaneous estimate, i.e., $\hat{\mathbf{p}} = \mathbf{x}(n)x(n)$ and $\hat{\mathbf{R}} = \mathbf{x}(n)\mathbf{x}^T(n)$. The coefficient update in (34) is converted to

$$\begin{aligned}
\mathbf{w}(n+1) &= (1 - \mu\gamma)\mathbf{w}(n) + \mu(\hat{\mathbf{p}} - \hat{\mathbf{R}}\mathbf{w}(n)) \\
&= (1 - \mu\gamma)\mathbf{w}(n) + \mu(\mathbf{x}(n)x(n) - \mathbf{x}(n)\mathbf{x}^T(n)\mathbf{w}(n)) \\
&= (1 - \mu\gamma)\mathbf{w}(n) + \mu\mathbf{x}(n)\underbrace{\left(x(n) - \mathbf{x}^T(n)\mathbf{w}(n)\right)}_{e(n)} \\
&= (1 - \mu\gamma)\mathbf{w}(n) + \mu e(n)\mathbf{x}(n)
\end{aligned} \tag{35}$$

We get the coefficient update (35) for leaky LMS.

The leaky LMS is implemented to estimated AR coefficients of (22), where $a_1 = 0.1$ and $a_2 = 0.8$. Table 3 and Table 4 show the estimates of a_1 and a_2, respectively.

Table 3: Estimates of a_1 using different μ and γ

	$\mu = 0.05$	$\mu = 0.01$
$\gamma = 0.01$	0.073	0.112
$\gamma = 0.1$	0.091	0.128
$\gamma = 5$	0.047	0.058
$\gamma = 10$	0.031	0.035

For the fixed γ, the smaller μ still results in the better estimates.For fixed μ, with the increase of the γ, the leaky LMS converge to incorrect values of the coefficients, which approach zero. This can be explained in two perspectives.

Table 4: Estimates of a_2 using different μ and γ

	$\mu = 0.05$	$\mu = 0.01$
$\gamma = 0.01$	0.705	0.736
$\gamma = 0.1$	0.619	0.663
$\gamma = 5$	0.105	0.114
$\gamma = 10$	0.060	0.063

- In the perspective of the cost function, there is an additional term $\gamma\|\mathbf{w}(n)\|^2$ in leaky LMS. If we reduce the cost function, we will also reduce the norm of the coefficients $\mathbf{w}$. Therefore, for the large γ, the term $\gamma\|\mathbf{w}(n)\|^2$ will become dominant in the cost function, which means we are mainly minimizing the norm of the coefficients. In this case, we will get the small coefficient, which is incorrect.
- In the perspective of the coefficient update in (35), we can consider the model of convergence. We define the distance vector $\mathbf{v}(n) = \mathbf{w}(n) - \mathbf{w}_{\text{opt}}$ and rewrite (22) as $x(n) = \mathbf{x}^T(n)\mathbf{w}_{\text{opt}} + \eta(n)$. In this case, we can rewrite (35) as

$$\begin{aligned}\mathbf{w}(n+1) &= (1-\mu\gamma)\mathbf{w}(n) + \mu\mathbf{x}(n)\left(\mathbf{x}^T(n)\mathbf{w}_{\text{opt}} + \eta(n) - \mathbf{x}^T(n)\mathbf{w}(n)\right) \\ &= (1-\mu\gamma)\mathbf{w}(n) + \mu\mathbf{x}(n)\mathbf{x}^T(n)\mathbf{w}_{\text{opt}} - \mu\mathbf{x}(n)\mathbf{x}^T(n)\mathbf{w}(n) + \mu\mathbf{x}(n)\eta(n) \end{aligned} \tag{36}$$

  By subtracting $\mathbf{w}_{\text{opt}}$ from both sides, we get

$$\mathbf{v}(n+1) = (1-\mu\gamma)\mathbf{v}(n) - \mu\mathbf{x}(n)\mathbf{x}^T(n)\mathbf{v}(n) + \mu\mathbf{x}(n)\eta(n) \tag{37}$$

$$\begin{aligned}\mathbb{E}\left\{\mathbf{v}(n+1)\right\} &= \left((1-\mu\gamma)\mathbf{I} - \mu\mathbb{E}\{\mathbf{x}(n)\mathbf{x}^T(n)\}\right)\mathbb{E}\{\mathbf{v}(n)\} + \mu\mathbb{E}\{\eta(N)\mathbf{x}(n)\} \\ &= \left((1-\mu\gamma)\mathbf{I} - \mu\mathbf{R}\right)\mathbb{E}\{\mathbf{v}(n)\} \end{aligned} \tag{38}$$

  We define the transformed vector $\tilde{\mathbf{v}}(n) = \mathbf{Q}\mathbf{v}(n)$, where $\mathbf{Q}$ is obtained from the eigenvalue decomposition of $\mathbf{R}$, i.e., $\mathbf{R} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T$. Then we can rewrite (38) as

$$\tilde{\mathbf{v}}(n+1) = ((1-\mu\gamma)\mathbf{I} - \mu\mathbf{\Lambda})\tilde{\mathbf{v}}(n) \tag{39}$$

$$\begin{bmatrix}\tilde{v}_1(n+1)\\ \vdots \\ \tilde{v}_1(n+1)\end{bmatrix} = \begin{bmatrix}1-\mu\gamma-\mu\lambda_{max} & 0 & \cdots & 0 \\ 0 & 1-\mu\gamma-\mu\lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1-\mu\gamma-\mu\lambda_{min}\end{bmatrix}\begin{bmatrix}\tilde{v}_1(n)\\ \vdots \\ \tilde{v}_1(n)\end{bmatrix} \tag{40}$$

  In order to ensure the convergence of the leaky LMS, we need to guarantee that $|1-\mu\gamma-\mu\lambda_i| < 1$. Because of the term $-\mu\gamma$, it can be guaranteed even though one of the eigenvalues λ_i is zero, which is the aim of leaky LMS. However, for the large γ, it can be regarded as the distance $\mathbf{v}(n)$ is reduced in a large rate, which is equivalent to using large step size. Therefore, there will be large excess error when the large γ is used, resulting in the incorrect estimates of coefficients.

2.2 Adaptive Step Sizes

2.2.1 Task a: GASS Algorithm

In the gradient adaptive step-size (GASS) algorithms, the step size of LMS is dynamically updated at each step according to

$$\mu(n+1) = \mu(n) + \rho e(n)\mathbf{x}^T(n)\psi(n) \tag{41}$$

where the $\psi(n)$ can be calculated following three methods

$$\text{Benveniste:} \quad \psi(n) = \left[\mathbf{I} - \mu(n-1)\mathbf{x}(n-1)\mathbf{x}^T(n-1)\right]\psi(n-1) + e(n-1)\mathbf{x}(n-1) \tag{42}$$

$$\text{Farhang:} \quad \psi(n) = \alpha\psi(n-1) + e(n-1)\mathbf{x}(n-1), \quad 0 < \alpha < 1 \tag{43}$$

$$\text{Matthews:} \quad \psi(n) = e(n-1)\mathbf{x}(n-1) \tag{44}$$

In this part, the following real-value MA(1) process is considered.

$$x(n) = 0.9\eta(n-1) + \eta(n), \quad \eta \sim \mathcal{N}(0, 0.5) \tag{45}$$

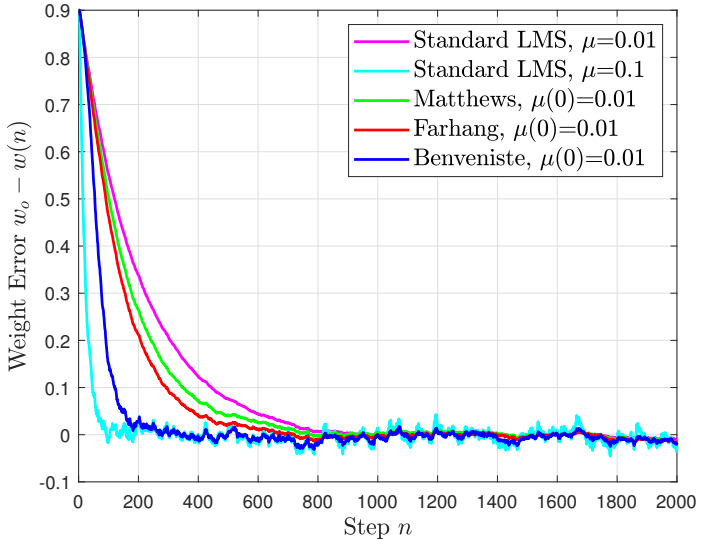


Figure 12: Comparison of standard LMS and GASS algorithm

The standard LMS and GASS algorithms are implemented in the system identification setting to identify the MA(1) system in (45). The input to the LMS is $\mathbf{x}(n) = [\eta(n-1)]^T$. The expression of error is $e(n) = x(n) - \mathbf{w}^T(n)\mathbf{x}(n)$. The standard LMS is implemented with step size of $\mu = 0.01$ and $\mu = 0.1$. For GASS algorithms, theri initial step sizes for weights update are all set to $\mu(0) = 0.01$, and the step size for $\mu(n)$ update is set to $\rho = 0.0005$.

Figure 12 shows the weight error in the standard LMS and GASS algorithms. The curves are plotted by averaging 100 trials of experiments. We can see that the standard LMS with $\mu = 0.1$ congerges fastest, but it has the largest steady state error. Benveniste's method also converge fast, which is only a little slower than standard LMS with $\mu = 0.1$, although its μ starts from a small value. Compared with standard LMS with $\mu = 0.1$, Benveniste's method has smaller steady state error. Both Farhang's and Matthews's method converge slower than the Benveniste's method, but they have better performance in terms of steady state error. The standard LMS with $\mu = 0.01$ converge slowest and its steady state error is also smaller than Benveniste's method.

In conclusion, the GASS outperforms LMS in terms of the convergence speed, but it may lead to higher steady state error.

2.2.2 Task b: NLMS Algorithm

In the normalized LMS (NLMS), the coefficients update is

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \frac{\beta}{\epsilon + \|\mathbf{x}(n)\|^2} e(n)\mathbf{x}(n) \tag{46}$$

which can be equivalently written in the form of

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu e_p(n)\mathbf{x}(n) \tag{47}$$

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

Proof *Firstly, define* $\Delta\mathbf{w}(n) = \mathbf{w}(n+1) - \mathbf{w}(n)$, *then we have*

$$\begin{aligned}
\Delta\mathbf{w}(n) &= \mathbf{w}(n+1) - \mathbf{w}(n) \\
&= \mu e_p(n)\mathbf{x}(n) \qquad (48) \\
&= \mu \left(d(n) - \mathbf{x}^T(n)\left(\mathbf{w}(n) + \Delta\mathbf{w}(n)\right)\right)\mathbf{x}(n) \\
&= \mu(\underbrace{d(n) - \mathbf{x}^T(n)\mathbf{w}(n)}_{e(n)} - \mathbf{x}^T(n)\Delta\mathbf{w}(n))\mathbf{x}(n) \\
&= \mu \left(e(n) - \mathbf{x}^T\Delta\mathbf{w}(n)\right)\mathbf{x}(n) \\
&= \mu e(n)\mathbf{x}(n) - \mu\mathbf{x}^T(n)\mathbf{x}(n)\Delta\mathbf{w}(n) \qquad (49)
\end{aligned}$$

By rearrange this expression, we have

$$(1+\mu\mathbf{x}^T(n)\mathbf{x}(n))\Delta\mathbf{w}(n) = \mu e(n)\mathbf{x}(n) \tag{50}$$

$$\begin{aligned}\Delta\mathbf{w}(n) &= \frac{\mu}{1+\mu\|\mathbf{x}(n)\|^2}e(n)\mathbf{x}(n)\\ &= \frac{1}{\frac{1}{\mu}+\|\mathbf{x}(n)\|^2}e(n)\mathbf{x}(n)\end{aligned} \tag{51}$$

By setting $\beta = 1$ *and* $\epsilon = 1/\mu$*, we get*

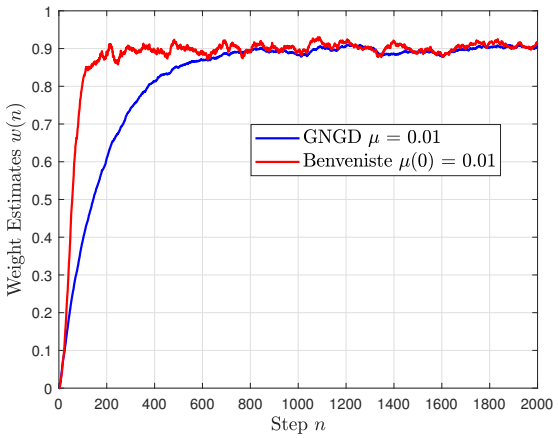
$$\begin{aligned}\mathbf{w}(n+1) &= \mathbf{w}(n) + \Delta\mathbf{w}(n)\\ &= \mathbf{w}(n) + \frac{\beta}{\epsilon+\|\mathbf{x}(n)\|^2}e(n)\mathbf{x}(n)\end{aligned} \tag{52}$$

Therefore, we prove that the (47) *is equivalent to* (46) *with* $\beta = 1$ *and* $\epsilon = 1/\mu$. ■

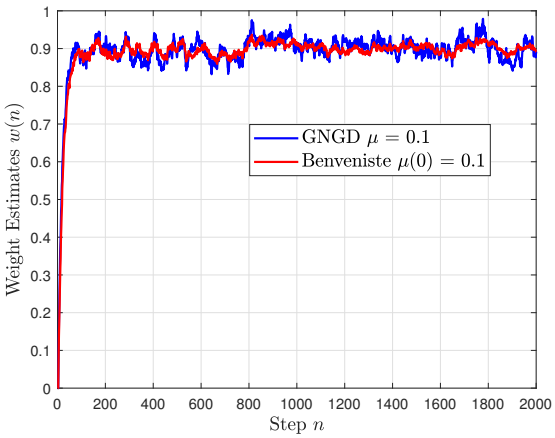
2.2.3 Task c: GNGD Algorithm

In Generalized Normalized Gradient Descent (GNGD) algorithm, the regularization facotr in NLMS is updated according to

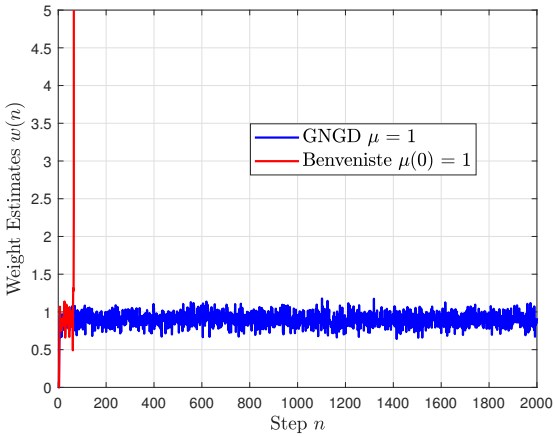
$$\epsilon(n+1) = \epsilon(n) - \rho\mu\frac{e(n)e(n-1)\mathbf{x}^T(n)\mathbf{x}(n-1)}{\left(\epsilon(n-1)+\|\mathbf{x}(n-1)\|^2\right)^2} \tag{53}$$



(a) $\mu = 0.01$



(b) $\mu = 0.1$



(c) $\mu = 1$

Figure 13: Comparison of GNGD algorithm and Benveniste's algorithm

In this part, the GNGD algorithm and Benveniste's algorithm are compared in identifying the MA(1) system (45). The value of μ in GNGD is set to the same value of the initial step size $\mu(0)$ in GASS. The value of ρ in GASS is set to $\rho = 0.0005$.

Figure 13 shows the weight estimates of MA(1) process in (45) using different μ. We can see that for the small value of $\mu = 0.01$, GNGD converges slower than Benveniste's algorithm but GNGD have better performance in steady state error. When μ increases to $\mu = 0.1$, both algorithms converge in almost the same speed, but the state error of GNGD is lager. For the very large value of μ (i.e., $\mu = 1$), the GNGD algorithm can still converge, but Benveniste's algorithm cannot.

In terms of complexity, for Benveniste's algorithm, it needs $3M^2+M$ multiplications and $2M^2+M$ additions to calculate $\psi(n)$, $2M+1$ multiplications and $M+1$ additions to update $\mu(n)$, and $M+1$ multiplications and M additions to update $\mathbf{w}(n)$. So there are totally $3M^2+4M+2$ multiplications and $2M^2+3M+1$ additions in each step n.

Similarly, for GNGD algorithm, it needs $M+6$ multiplications and $2M+2$ additions to update $\epsilon(n)$ as well as $2M+2$ multiplications and $2M+1$ additions to update $\mathbf{w}(n)$. So, there are totally $3M+8$ multiplications and $4M+3$ additions in each step.

As we can see, the computational complexity of Benveniste's algorithm is $O(M^2)$ and that of GNGD is $O(M)$, which indicates that the GNGD algorithm has lower computational complexity.

2.3 Adaptive Noise Cancellation

2.3.1 Task a & b: Adaptive Line Enhancer (ALE)

In this part, a noise-corrupted signal is considered, which is given as

$$s(n) = \sin(0.01\pi n) + \eta(n) \tag{54}$$

where $\eta(n)$ is the coloured noise which is given as

$$\eta(n) = v(n) + 0.5v(n-1), \quad v(n) \sim \mathcal{N}(0,1) \tag{55}$$

Figure 14 shows the noised signal

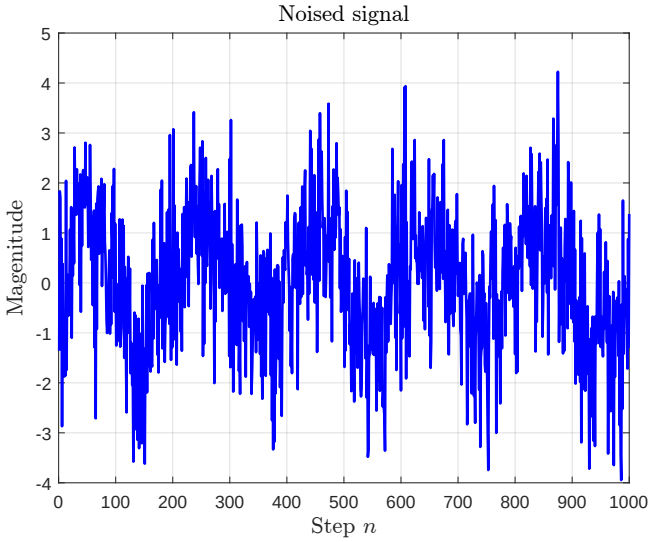


Figure 14: Noise-corrupted signal

In order to determine the minimum value of delay Δ, we can consider the Mean Square Error $\mathbb{E}\{(s(n) - \hat{x}(n))^2\}$, from which we can get

$$\begin{aligned}
\mathbb{E}\{(s(n)-\hat{x}(n))^2\} &= \mathbb{E}\{(s(n)-\hat{x}(n))(s(n)-\hat{x}(n))^T\} \\
&= \mathbb{E}\{(s(n)-\mathbf{w}^T(n)\mathbf{u}(n))(s(n)-\mathbf{u}^T(n)\mathbf{w}(n))\} \\
&= \mathbb{E}\{s(n)^2\} - 2\mathbb{E}\{s(n)\mathbf{u}^T(n)\}\mathbf{w}(n) + \mathbf{w}^T(n)\mathbb{E}\{\mathbf{u}(n)\mathbf{u}^T(n)\}\mathbf{w}(n)
\end{aligned} \tag{56}$$

If the noise in $s(n)$ and $\mathbf{u}(n)$ are uncorrelated, the second term in (56) becomes

$$-2\mathbb{E}\{s(n)\mathbf{u}^T(n)\}\mathbf{w}(n) = -2\mathbf{w}^T(n)\begin{bmatrix}\mathbb{E}\{x(n)x(n-\Delta)\}\\ \vdots \\ \mathbb{E}\{x(n)x(n-\Delta-M+1)\}\end{bmatrix} \tag{57}$$

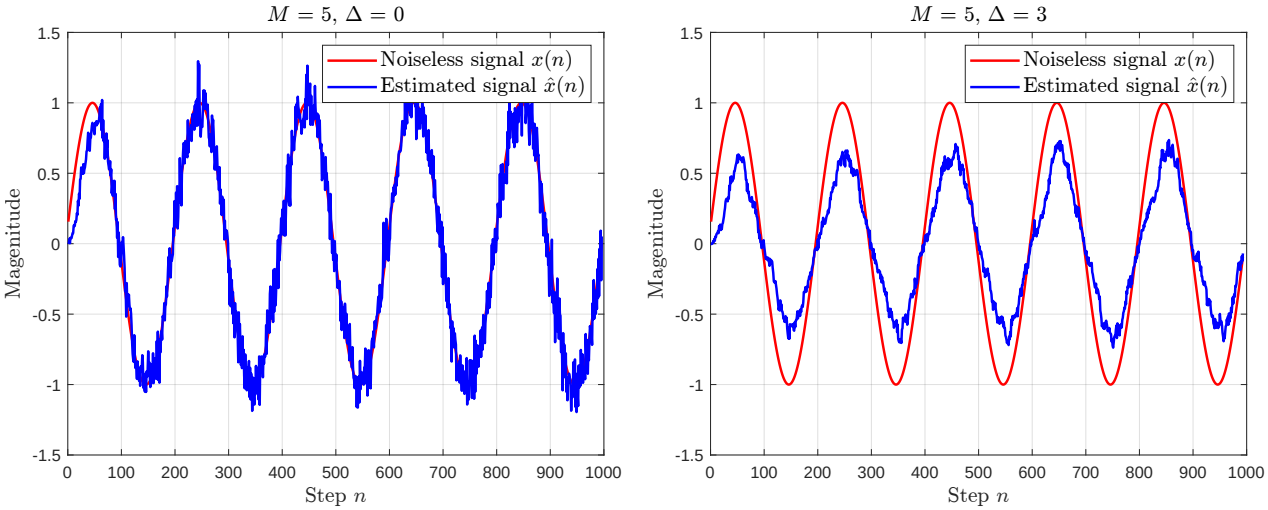
where there is only correlation of the pure signal $x(n)$. Therefore, the linear precoder will minimize the distance between the $x(n)$ and $\hat{x}(n)$ in this case. The error $e(n)$ will finally converge to $\eta(n)$ and $\hat{x}(n)$ will converge to $x(n)$. However, there is noise correlation in $-2\mathbb{E}\{s(n)\mathbf{u}^T(n)\}\mathbf{w}(n)$, the coefficients $\mathbf{w}(n)$ will also catch up the noise, and the both noise and signal will remain in the $e(n)$. Therefore, signal $x(n)$ in $\hat{x}(n)$ will be suppressed.

To ensure the noise in $s(n)$ and $\mathbf{u}(n)$ are uncorrelated, the delay need to be larger than the correlation length of $\eta(n)$, within which the correlation between the first and last sample is not zero. According to the expression $\eta(n) = v(n) + 0.5v(n-2)$, its correlation length is 2. Therefore, we have

$$\Delta > 2 \tag{58}$$

The minimum value of delay is 3.

Figure 15 comparison the results of ALE with and without delay. One visualize from the figure that the estimated signal is still noise-corrupted if there is no delay of the input signal, while the estimated signal is purer when $\Delta = 3$.



(a) $\Delta = 0$ (b) $\Delta = 3$

Figure 15: Comparison of different delay

The MSPE is calculated using ALE with different filter M orders and delays Δ to investigate the impact of them. The averaged MSPE over 1000 trials is calculated for each case. The step size μ is set to 0.01.

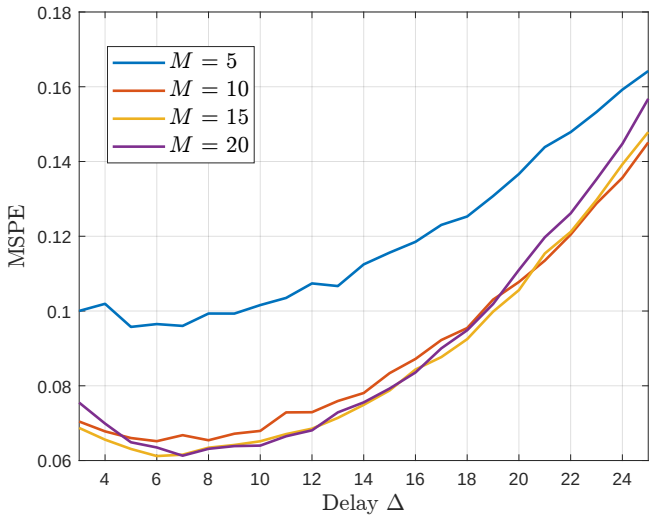


Figure 16: MSPE against different filter orders and delays

In Figure 16, we can see that the MSPE firstly decreases and then increases with the increase of the delay Δ. In terms of filter order, the MSPE dropped sharply when the order changed from 5 to 10. Nevertheless, as the order changes from 10 to 15 or 20, there is no significant drop of MSPE and it even increases sometimes. Given the computational cost increase with M, the best choice is $M = 10$.

2.3.2 Task c: Adaptive Noise Cancellation (ANC)

In this part, the ANC is implemented to compare with ALE, where the filter order M and step size μ are set to 10 and 0.01, respectively. The result is shown in Figure 17.

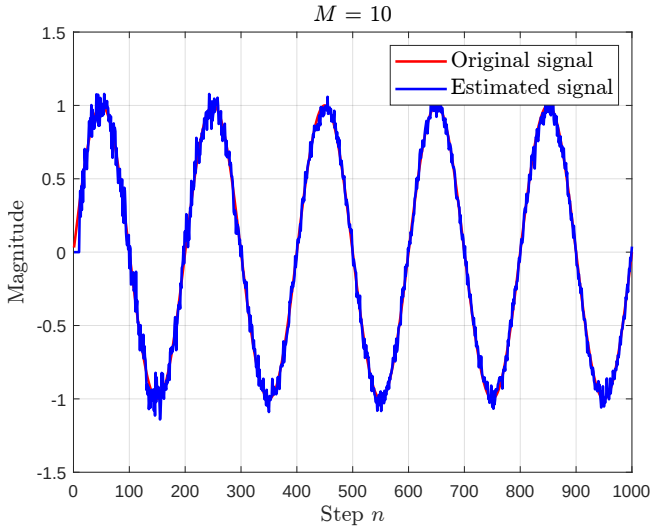


Figure 17: Noiseless signal and estimated signal by ANC

The MSPE in this case is 0.0038. For ALE, the minimum MSPE for $M = 10$ is achieved when $\Delta = 6$, which is 0.065. Therefore, the ANC outperforms ALE in de-nosing the noisy sinusoid signal.

2.3.3 Task d: Denoise EEG Data

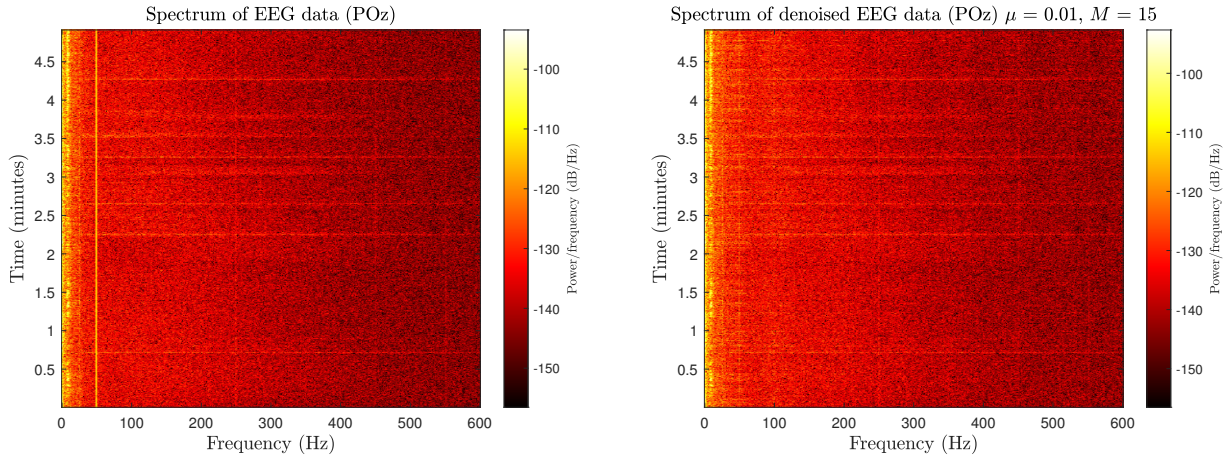
In the EEG data, there is a strong 50Hz component introduce by the mains. Figure 18(a) shows the spectrum of the corrupted EEG data using `spectrogram` function with window length of 1000 and overlap of 500. There is a bright yellow vertical line at 50Hz in this figure, which represents for the noise.

The ANC is exploited to suppress the 50Hz component in EEG data. The reference noise signal is genereated according to

$$r(n) = sin(2\pi \times 50 \times T_s \times n) + w(n) \tag{59}$$

where T_s is the sampling interval and $w(n) \sim \mathcal{N}(0, 1)$ is the white Gaussian noise. The sampling frequency of EEG data is $f_s = 1200$Hz, so $T_s = 1/f_s = 1/1200$s. In ANC, the step size is set to $\mu = 0.01$ and the filter length is set to $M = 15$.

Figure 18(b) shows the spectrum of the EEG data denoised by ANC, where we can see that the bright yellow line at 50Hz in Figure 18(a) disappears and other frequency components are not affected. Therefore, the noise introduced by the mains is almost totally removed.



(a) Corrupted

(b) Denoised

Figure 18: Spectrum of corrupted and denoised EEG data

3 Widely Linear Filtering and Adaptive Spectrum Estimation

3.1 Complex LMS and Widely Linear Modelling

3.1.1 Task a: Comparison of CLMS and ACLMS

In the complex LMS (CMLS) algorithm, the weights update procedure is given as

$$\begin{aligned} \hat{y}(n) &= \mathbf{h}^H(n)\mathbf{x}(n) \\ e(n) &= y(n) - \hat{y}(n) \\ \mathbf{h}(n+1) &= \mathbf{h}(n) + \mu e^*(n)\mathbf{x}(n) \end{aligned} \tag{60}$$

In the argumented CLMS (ACLMS) algorithm, the procedure is given as

$$\begin{aligned} \hat{y}(n) &= \mathbf{h}^H(n)\mathbf{x}(n) + \mathbf{g}^H(n)\mathbf{x}^*(n) \\ e(n) &= y(n) - \hat{y}(n) \\ \mathbf{h}(n+1) &= \mathbf{h}(n) + \mu e^*(n)\mathbf{x}(n) \\ \mathbf{g}(n+1) &= \mathbf{g}(n) + \mu e^*(n)\mathbf{x}^*(n) \end{aligned} \tag{61}$$

To compare CLMS and ACLMS, a WLMA(1) process is considered, which is given as

$$y(n) = x(n) + b_1 x(n-1) + b_2 x^*(n-1), \quad x \sim \mathcal{CN}(0,1) \tag{62}$$

where $b_1 = 1.5 + 1j$ and $b_2 = 2.5 - 0.5j$. Both CLMS and ACLMS algorithms are implemented in the system identification setting to identify the WLAM model. The input to the model and CLMS/ACLMS is $\mathbf{x}(n) = [x(n-1)]^T$.

Figure 19 shows the leaning curves, $10\log_{10}|e(n)|^2$, of CLMS and ACLMS by averaging the results of 100 trials. For CLMS, the square error $|e(n)|^2$ is always around 7dB. However, for the ACLMS, its square error finally converges to around -3dB, which is much lower than that in CLMS. Therefore, ACLMS have better performance in terms of steady state error when learning the weights of WLMA(1) model.

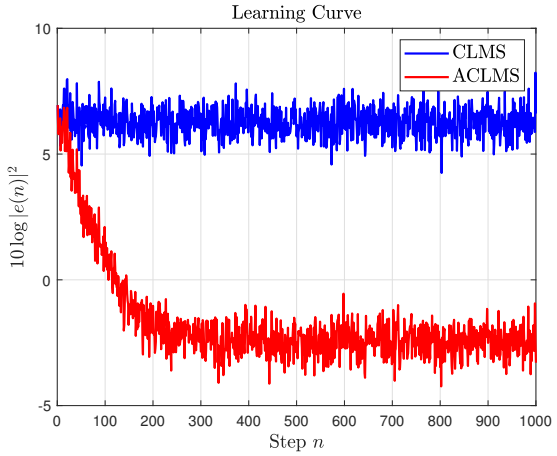


Figure 19: Learning curves of CLMS and ACLMS

3.1.2 Task b: Processing Wind Data

Figure 20 shows the scatter diagrams of wind data. Based on observation, the three complex wind signals are all non-circular, because the properties of signals would be changes if they are rotated. Their circularity can also be analysed quantitatively using circularity coefficient, which is defined as

$$|\rho| = \frac{|p|}{c}, \quad c = \mathbb{E}\{zz^*\}, \quad p = \mathbb{E}\{zz^T\} \tag{63}$$

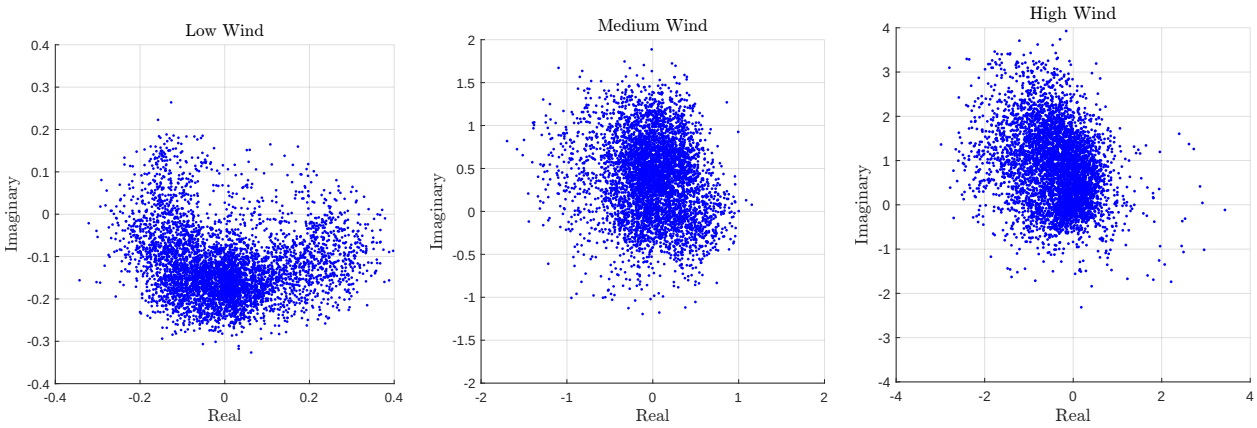
where c and p are the covariance and pseudocovariance of z, respectively. In practice, they can be approximated by

$$c = \frac{1}{N}\mathbf{z}^H\mathbf{z}, \quad p = \frac{1}{N}\mathbf{z}^T\mathbf{z} \tag{64}$$

where N is the number of data samples and $z = [z(0), ..., z(N-1)]^T$. The circularity coefficients of the low, medium and high wind data is calculated based on (63) and (64).

$$\begin{aligned} \text{low}: &\quad |\rho_l| = 0.1592 \\ \text{medium}: &\quad |\rho_m| = 0.4543 \\ \text{high}: &\quad |\rho_h| = 0.6237 \end{aligned} \tag{65}$$

According to circularity coefficients in (65), the circularity of low wind signal is highest and that of high wind data is lowest.



(a) Low wind (b) Medium wind (c) High wind

Figure 20: Scatter diagrams of wind data

The CLMS and ACLMS filter are configured in a prediction setting to perform a one-step ahead prediction, i.e., using the past wind data to predict one future win data. The wind signals in east-west and north-south directions are firstly combined to a complex signal

$$v(n) = v_{\text{east}}(n) + jv_{\text{north}}(n) \tag{66}$$

The signal input to the CLMS and ACLMS is $\mathbf{x}(n) = [v(n-1), ..., v(n-M)]^T$, which is used to predict $v(n)$. The prediction gain R_p is exploited to evaluate the performance of CLMS and ACLMS. The prediction gain is defined as

$$R_p = 10 \log_{10} \frac{\sigma_v^2}{\sigma_e^2} \tag{67}$$

where $\sigma_v^2 = \mathbb{E}\left\{|v(n)|^2\right\}$ is the signal power and $\sigma_e^2 = \mathbb{E}\left\{|e(n)|^2\right\}$ is the error power.

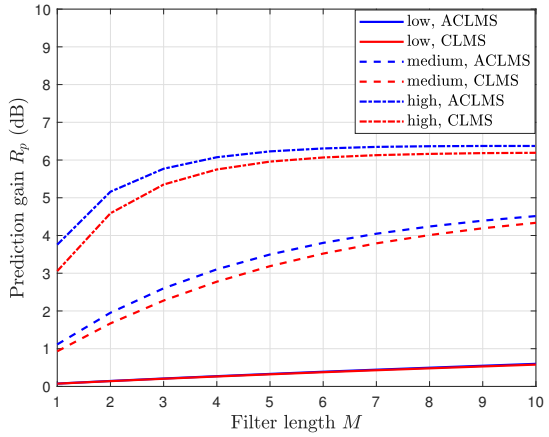


Figure 21: Comparison of CLMS and ACLMS for different wind regimes

Figure 21 evaluates CLMS and ACLMS in terms of prediction gain. For the low wind signal, which has high circularity, CLMS and ACLMS has almost the same performance. For the wind signals that have low circularity (i.e., medium and high wind), the ACLMS outperforms the CLMS algorithm.

3.1.3 Task c: Circularity Diagrams of Complex α-β Voltages

In this section, the balanced and unbalanced three-phase voltages are generated with system frequency $f_o = 50\text{Hz}$ and sampling frequency $f_s = 1000\text{Hz}$.

Figure 22 shows the circularity diagram of balanced system, where all the voltages locate at several points which make up a circle. The amplitude of all the complex voltages are the same. Therefore, the balanced complex voltages have very high circularity. When the phase ϕ is changed, the points rotate angle of ϕ around the origin in counterclockwise.

Figure 23 shows the circularity diagram of unbalanced systems with $f_o = 50\text{Hz}$, $f_s = 1000\text{Hz}$ and $\phi = 0$. The magnitudes (V_b, V_c) and phases (Δ_b, Δ_c) are changed to explore their impact. In the unbalanced systems, all the voltages also locate at several points, but these points become non-circular, which can be a criterion to identify whether there is a fault.

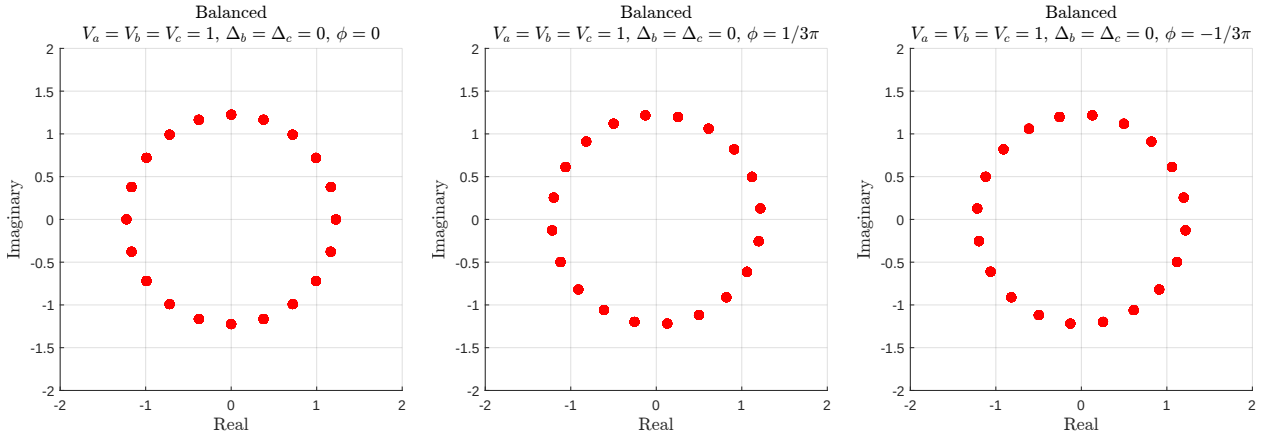


Figure 22: Circularity diagrams balanced complex voltages

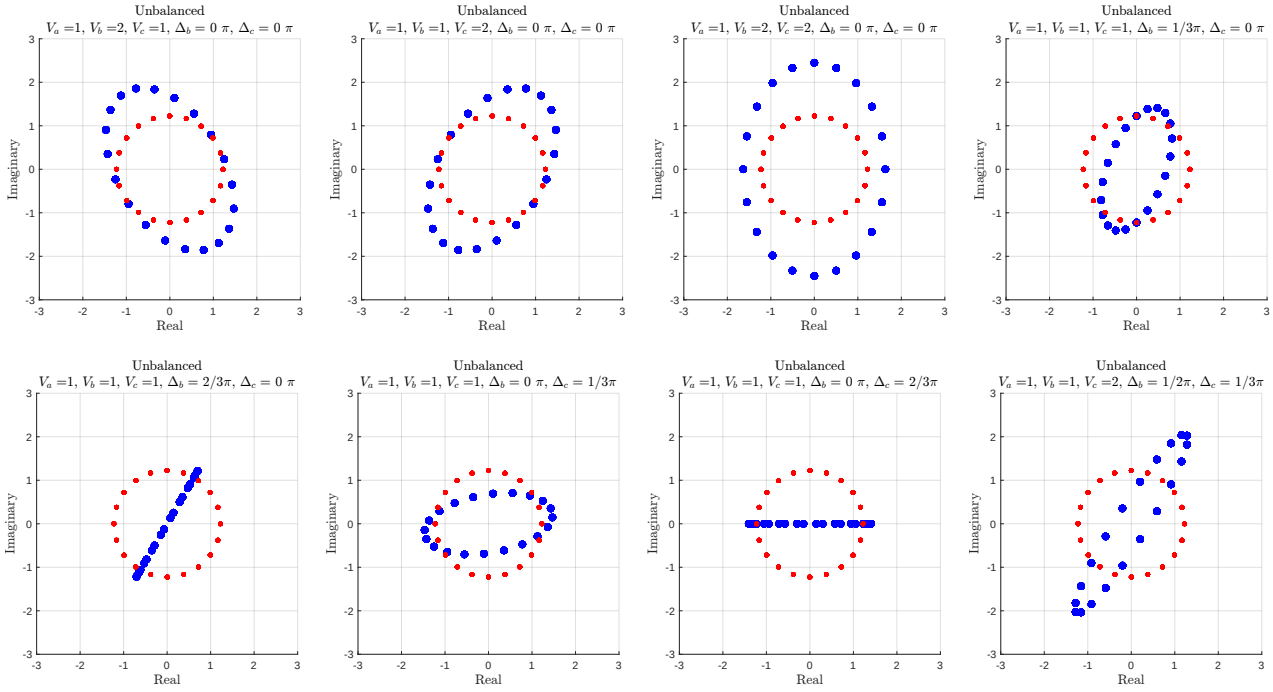


Figure 23: Circularity diagrams unbalanced complex voltages with different magnitudes and phases (blue points are "unbalanced"; red points are "balanced")

3.1.4 Task d & e: Estimates of System Frequency

Consider the strictly linear autoregressive model, given by

$$v(n+1) = h^*(n)v(n) \tag{68}$$

The frequency of balanced complex α-β voltage can be expressed as

$$f_o(n) = \frac{f_s}{2\pi} \arctan\left\{\frac{\Im\{h(n)\}}{\Re\{h(n)\}}\right\} \tag{69}$$

Proof *In balanced system, the complex voltage is expressed as*

$$v(n) = \sqrt{\frac{3}{2}} V e^{j\left(2\pi \frac{f_o}{f_s} n + \phi\right)} \tag{70}$$

According to (68)*, we have*

$$h^*(n) = \frac{\sqrt{\frac{3}{2}} V e^{j\left(2\pi \frac{f_o}{f_s}(n+1) + \phi\right)}}{\sqrt{\frac{3}{2}} V e^{j\left(2\pi \frac{f_o}{f_s} n + \phi\right)}} \tag{71}$$

$$h^*(n) = e^{j2\pi \frac{f_o}{f_s}} \tag{72}$$

The two side of (72) *must be co-phased. According to Nyquist sampling theory, we must have* $0 < 2\pi \frac{f_o}{f_s} \leq \pi \quad (f_s \geq 2f_o)$*. Therefore, we have*

$$2\pi \frac{f_o}{f_s} = \arctan\left\{\frac{\Im\{h(n)\}}{\Re\{h(n)\}}\right\} \tag{73}$$

$$f_o = \frac{f_s}{2\pi} \arctan\left\{\frac{\Im\{h(n)\}}{\Re\{h(n)\}}\right\} \tag{74}$$

■

Similarly, if we consider the following widely linear autoregressive model

$$v(n+1) = h^*(n)v(n) + g^*(n)v^*(n) \tag{75}$$

the frequency of unbalanced voltage is

$$f_o(n) = \frac{f_s}{2\pi} \arctan\left\{\frac{\sqrt{\Im^2\{h(n)\} - |g(n)|^2}}{\Re\{h(n)\}}\right\} \tag{76}$$

Proof *In unbalanced system, the complex voltage is given as*

$$v(n) = Ae^{j\left(2\pi \frac{f_o}{f_s} n + \phi\right)} + Be^{-j\left(2\pi \frac{f_o}{f_s} n + \phi\right)} \tag{77}$$

According to (75)*, we have the following equivalent equations.*

$$g^*(n) = \frac{v(n+1) - h^*(n)v(n)}{v^*(n)} \tag{78}$$

$$g(n)g^*(n) = \frac{\Big(v(n+1) - h^*(n)v(n)\Big)\Big(v^*(n+1) - h(n)v^*(n)\Big)}{v(n)v^*(n)} \tag{79}$$

$$|g(n)|^2 = \frac{v(n+1)v^*(n+1)}{v(n)v^*(n)} - \frac{h(n)v(n+1)}{v(n)} - \frac{h^*(n)v^*(n+1)}{v^*(n)} + h(n)h^*(n) \tag{80}$$

$$|g(n)|^2 = \left(e^{j2\pi \frac{f_o}{f_s}}\right)^2 - (h(n) + h^*(n))e^{j2\pi \frac{f_o}{f_s}} + |h(n)|^2 \tag{81}$$

$$|g(n)|^2 = \left(e^{j2\pi \frac{f_o}{f_s}}\right)^2 - 2\Re\{h(n)\}e^{j2\pi \frac{f_o}{f_s}} + \Re^2\{h(n)\} + \Im^2\{h(n)\} \tag{82}$$

$$|g(n)|^2 - \Im^2\{h(n)\} = \left(e^{j2\pi \frac{f_o}{f_s}} - \Re\{h(n)\}\right)^2 \tag{83}$$

$$j\sqrt{\Im^2\{h(n)\} - |g(n)|^2} = e^{j2\pi \frac{f_o}{f_s}} - \Re\{h(n)\} \tag{84}$$

$$e^{j2\pi \frac{f_o}{f_s}} = \Re\{h(n)\} + j\sqrt{\Im^2\{h(n)\} - |g(n)|^2} \tag{85}$$

$$2\pi\frac{f_o}{f_s} = \arctan\left\{\frac{\sqrt{\Im^2\{h(n)\} - |g(n)|^2}}{\Re\{h(n)\}}\right\} \tag{86}$$

$$f_o = \frac{f_s}{2\pi}\arctan\left\{\frac{\sqrt{\Im^2\{h(n)\} - |g(n)|^2}}{\Re\{h(n)\}}\right\} \tag{87}$$

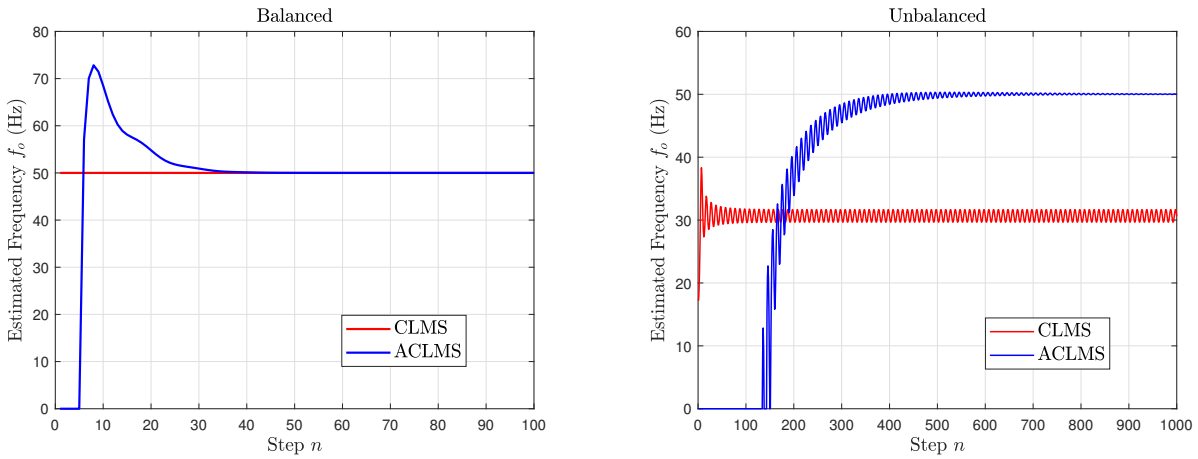
■

Therefore, based on (68), (69), (75) and (76), we can use CLMS and ACLMS to estimate the frequency of balanced and unbalanced system. The parameters of balanced system and unbalanced system are shown below.

Table 5: Parameters of balanced and unbalanced systems

	Balanced	Unbalanced
f_o	50Hz	50Hz
f_s	1000Hz	1000Hz
V_a	1	1
V_b	1	1
V_c	1	2
ϕ	0	0
Δ_b	0	0
Δ_c	0	$\frac{1}{3}\pi$

Figure 24 shows the frequency estimates for balanced and unbalanced systems. For the balanced system, the CLMS algorithm is capable of getting the correct frequency from the first step and the ACLMS algorithm can also converge to the correct frequency after several steps. For the unbalanced system, the ACMLS can estimate the frequency correctly. However, the CLMS algorithm cannot converge to the correct frequency. The main reason is that the complex voltages of unbalanced system are non-circular, on which the CLMS performs bad because of the conjugate component.



(a) Balanced

(b) Unbalanced

Figure 24: Comparison of CLMS and ACLMS for frequency estimation

3.2 Adaptive AR Model Based Time-Frequency Estimation

The conventional AR model can only estimate the spectrum of stationary signals. In this part, the adaptive AR model is investigated to estimate the non-stationary signal.

The following frequency modulated signal is considered.

$$y(n) = e^{j\left(\frac{2\pi}{f_s}\phi(n)\right)} + \eta(n) \tag{88}$$

where $\eta(n) \sim \mathcal{CN}(0, 0.05)$ and $\phi(n) = \int f(n)dn$. $f(n)$ is a frequency signal which is given as

$$f(n) = \begin{cases} 100, & 1 \leq n \leq 500 \\ 100 + \frac{n-500}{2}, & 501 \leq n \leq 1000 \\ 100 + \left(\frac{n-1000}{25}\right)^2, & 1001 \leq n \leq 1500 \end{cases} \tag{89}$$

Figure 25 shows the plot of $f(n)$. As the maximal frequency is 500Hz, the sample frequency is set to $f_s = 1000$Hz.

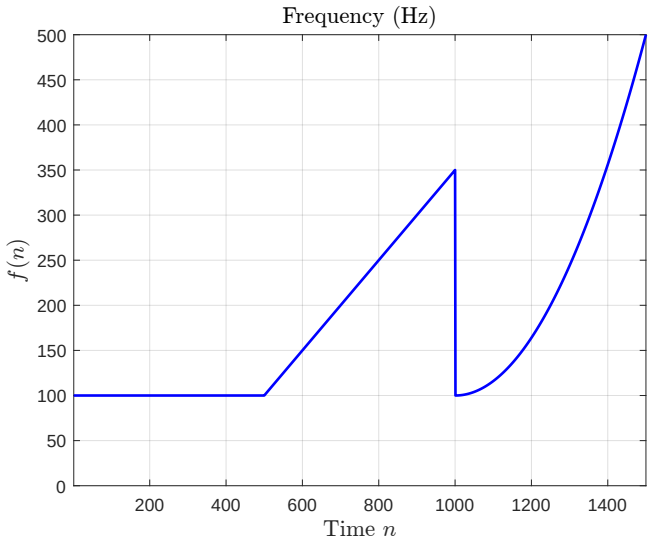


Figure 25: Frequency signal $f(n)$

The conventional AR(1) model is firstly used to estimate the spectrum of $y(n)$. The MATLAB function `aryule` is used to calculate the AR(1) coefficient and the function `freqz` is used to calculated the spectrum.

Figure 26 shows the spectrum estimated by the AR(1) model. Obviously, this method cannot capture the changes of frequency in $f(n)$.

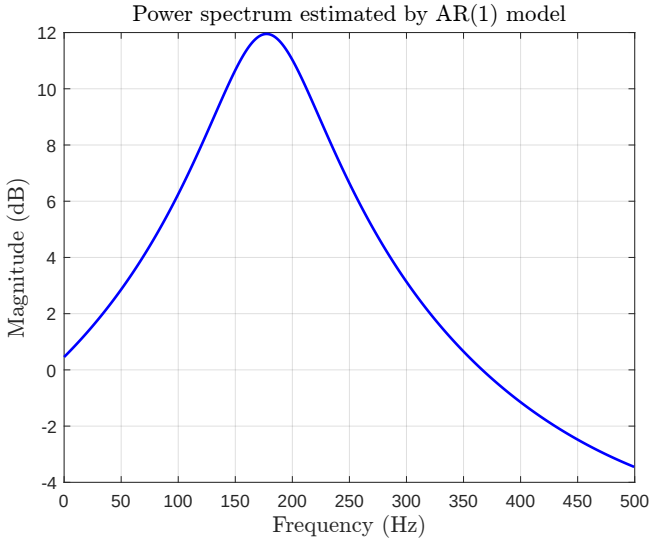


Figure 26: Spectrum estimated by AR(1) model

In order to capture the frequency changes, we can use the adaptive AR(1) model, where the CLMS algorithm is exploited to estimate the time-frequency spectrum. The adaptive AR(1) model follows (68).

Figure 27 shows the time-frequency estimated by adaptive AR(1) model. The bright parts in this figure represent the frequency components. We can see that the time-varying frequencies match the plot of frequency $f(n)$ in Figure 25. Therefore, the CLMS-based adaptive AR model is capable of estimating the frequency of a non-stationary signal, while the conventional AR spectrum is only suitable for stationary signal.

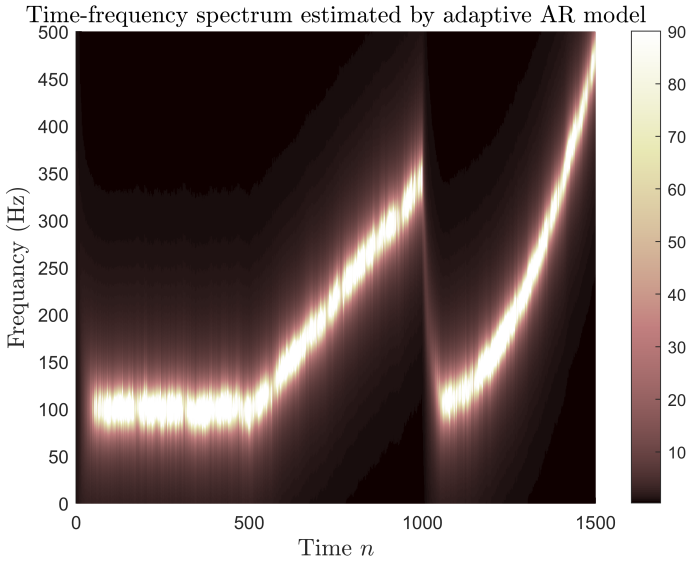


Figure 27: Time-frequency spectrum estimated by adaptive AR(1) model

3.3 A Real Time Spectrum Analyser Using Least Mean Square

3.3.1 Task a & b: Least Squares Solution and Discrete Fourier Transform

Consider the signal $\hat{\mathbf{y}} = \mathbf{F}\mathbf{w}$, which is an estimate of the actual signal $\mathbf{y}$. The matrix $\mathbf{F}$ is given as

$$\mathbf{F} = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & e^{j\frac{2\pi}{N}(1)(1)} & \cdots & e^{j\frac{2\pi}{N}(1)(N-1)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & e^{j\frac{2\pi}{N}(N-1)(1)} & \cdots & e^{j\frac{2\pi}{N}(N-1)(N-1)} \end{bmatrix} \tag{90}$$

The least squares problem is solved by minimising the cost function

$$\min_{\mathbf{w}} \|\mathbf{y} - \hat{\mathbf{y}}\|^2 \tag{91}$$

The solution of problem (91) is given as

$$\mathbf{w} = \left(\mathbf{F}^H\mathbf{F}\right)^{-1}\mathbf{F}^H\mathbf{y} \tag{92}$$

Proof *By expending the* (91)*, we have*

$$\begin{aligned} \|\mathbf{y} - \hat{\mathbf{y}}\|^2 &= (\mathbf{y} - \hat{\mathbf{y}})^H(\mathbf{y} - \hat{\mathbf{y}}) \\ &= (\mathbf{y} - \mathbf{F}\mathbf{w})^H(\mathbf{y} - \mathbf{F}\mathbf{w}) \\ &= \mathbf{w}^H\mathbf{F}^H\mathbf{F}\mathbf{w} - 2\mathbf{F}^H\mathbf{y}\mathbf{w} + \mathbf{y}^H\mathbf{y} \end{aligned} \tag{93}$$

It is obvious that $\mathbf{F}^H\mathbf{F}$ *is a semi-definite matrix, so the problem* (91) *is an unconstrained convex optimization problem, the solution of which is given by* $\frac{\partial\|\mathbf{y}-\hat{\mathbf{y}}\|^2}{\partial\mathbf{w}} = 0$*. Based on* (94)*, we have*

$$\begin{aligned} \frac{\partial\|\mathbf{y} - \hat{\mathbf{y}}\|^2}{\partial\mathbf{w}} &= (\mathbf{F}^H\mathbf{F} + \mathbf{F}\mathbf{F}^H)\mathbf{w} - 2\mathbf{F}^H\mathbf{y} \\ &= 2\mathbf{F}^H\mathbf{F}\mathbf{w} - 2\mathbf{y}\mathbf{F} \end{aligned} \tag{94}$$

By setting (94) *to zero, we have*

$$2\mathbf{F}^H\mathbf{F}\mathbf{w} - 2\mathbf{F}^H\mathbf{y} = 0 \tag{95}$$

$$\mathbf{F}^H\mathbf{F}\mathbf{w} = \mathbf{F}^H\mathbf{y} \tag{96}$$

$$\mathbf{w} = \left(\mathbf{F}^H\mathbf{F}\right)^{-1}\mathbf{F}^H\mathbf{y} \tag{97}$$

■

The solution (92) has a strong relationship with the discrete Fourier transform (DFT). Denote the DFT of $\mathbf{y}$ by $\mathbf{d} = [d(0), ..., d(N-1)]^T$.

$$y(n) = \frac{1}{N} \sum_{k=0}^{N-1} d(k) e^{j\frac{2\pi}{N}nk}, \quad n = 0, ..., N-1 \tag{98}$$

Observing the structure of matrix $\mathbf{F}$ and equation (98), $\mathbf{y}$ can be rewritten as

$$\mathbf{y} = \frac{1}{N}\mathbf{F}\mathbf{d} \tag{99}$$

Substitute it into equation (92), we have

$$\mathbf{w} = \frac{1}{N}(\mathbf{F}^H\mathbf{F})^{-1}\mathbf{F}^H\mathbf{F}\mathbf{d} = \frac{1}{N}\mathbf{d} \tag{100}$$

Therefore, the solution $\mathbf{w}$ of least squares is the DFT of $\mathbf{y}$ divided by N.

Based on the least squares interpretation of DFT, we can explain the Fourier transform in another perspective. Consider a function $f(t), t \in (t_1, t_2)$ and a set of orthonormal basis $\{B_k\}$. We can try to represent $f(t)$ using $\{B_k\}$, and we can get a approximation $\hat{f}(t)$,which is given as

$$\hat{f}(t) = \sum_k w_k B_k \tag{101}$$

In order to find $\{w_k\}$ that best approximate the function $f(t)$, the square error between $\hat{f}(t)$ and $f(t)$ needs to be minimized. The square error is given as

$$\begin{aligned}
\varepsilon^2 &= \int_{t_1}^{t_2} \left(f(t) - \hat{f}(t)\right)^2 dt \\
&= \int_{t_1}^{t_2} \left(f(t) - \sum_k w_k B_k\right)^2 dt \\
&= \int_{t_1}^{t_2} f^2(t) - 2f(t)\sum_k w_k B_k + \left(\sum_k w_k B_k\right)^2 dt \\
&= \int_{t_1}^{t_2} f^2(t) - 2f(t)\sum_k w_k B_k + \sum_k (w_k B_k)^2 dt
\end{aligned} \tag{102}$$

where the final step is due to the orthogonality of $\{B_k\}$.

The solution of $\{w_k\}$ is given by the least squares, i.e.,

$$\min_{\{w_k\}} \varepsilon^2 \tag{103}$$

Therefore, w_k need to satisfy $\frac{\partial \varepsilon^2}{\partial w_k} = 0$.

$$\frac{\partial \varepsilon^2}{\partial w_k} = -2\int_{t_1}^{t_2} f(t)B_k dt + 2w_k \int_{t_1}^{t_2} B_k^2 dt \tag{104}$$

By setting this expression to zero, we have

$$\begin{aligned}
w_k &= \frac{\int_{t_1}^{t_2} f(t)B_k dt}{\int_{t_1}^{t_2} B_k^2} \\
&= \frac{< f, B_k >}{< B_k, B_k >}
\end{aligned} \tag{105}$$

where $< a, b >$ is the inner product of a and b. As $\{B_k\}$ is orthonormal, $< B_k, B_k >= 1$. Therefore, the best approximation of $f(t)$ using the orthonormal basis $\{B_k\}$ is

$$\hat{f}(t) = \sum_k < f, B_k > B_k \tag{106}$$

Actually, the inner product $< f, B_k >$ is the projection of $f(t)$ onto B_k.

Now, we can consider the Fourier transform in terms of basis and projection. Fourier transform uses a complex orthonormal basis set $\{e^{j2\pi\xi t}\}$ with infinity size and weights $\{F(\xi)\}$ to approximate the signal $f(t), t \in (-\infty, +\infty)$, i.e.,

$$f(t) = \int_{-\infty}^{\infty} F(\xi) e^{j2\pi\xi t} d\xi \tag{107}$$

where the integral replace the sum in (101) because the size of basis set is infinite. According to (106), the optimal $F(\xi)$ is given by the projection of $f(t)$ on to basis, i.e.,

$$\begin{aligned} F(\xi) &=< f(t), e^{j2\pi\xi t} > \\ &= \int_{-\infty}^{+\infty} f(t) e^{-j2\pi\xi t} dt \end{aligned} \tag{108}$$

We can find that this expression is definition of Fourier transform.

3.3.2 Task c: DFT-CLSM Algorithm

As the DFT can be treated as the solution of least squares, the signal can be estimated by

$$\hat{y}(n) = \mathbf{w}^H \mathbf{x}(n) \tag{109}$$

where $\mathbf{x}$ represents for the DFT basis, which is given as

$$\mathbf{x}(n) = \frac{1}{N} \left[1, e^{j\frac{2n\pi}{N}}, ..., e^{j\frac{2n(N-1)\pi}{N}}\right]^T \tag{110}$$

Each element of $\mathbf{w}(n)$ represents the magnitude of each frequency components at time n. In $\mathbf{w}(n)$, the element corresponds to $e^{j\frac{2nk\pi}{N}}$ in $\mathbf{x}(n)$ is denoted by $w_k(n)$. The signal is assumed to be sampled by f_s. Then, the magnitude of $w_k(n)$ represents the power at frequency $\frac{k}{N} f_s$ when $k < \frac{N}{2}$, and represents power at frequency $\frac{k-N}{N} f_s$ when $k > \frac{N}{2}$.

The frequency modulated signal (88) in Part 3.2 is considered, and the CLMS algorithm is applied to get the weights $\mathbf{w}(n)$.The time-frequency spectrum can be plotted based on $\mathbf{w}(n)$. The sampling frequency is set to $f_s = 1000$Hz.

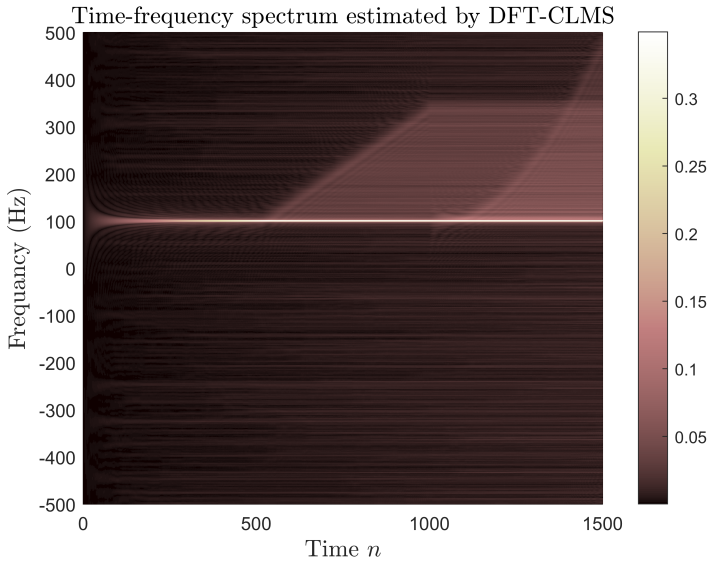


Figure 28: Time-frequency spectrum estimated by DFT-CLMS

Figure 28 shows the time-frequency spectrum estimated by DFT-CLMS. The elements in $\mathbf{w}(n)$ are reordered to show the true frequency. There is no negative frequency components because the signal is complex. In this figure, the bright part is not the line shown in Figure 25, but the area enclosed by the line and coordinate axis. It means that once a frequency component appears at time instance k, it exists at the all the subsequent time instances. The reason is that the weight $\mathbf{w}(k)$ in DFT-CLMS algorithm is the DFT of the sequence $\mathbf{y}_k = \mathbf{y}(0:k) = [y(0), ..., y(k)]^T$. Therefore, $\mathbf{w}(k)$ will reveal all the frequency components that appear before time k, resulting in a bright area instead of bright line in the estimated time-frequency spectrum.

3.3.3 Task d: DFT-CLSM for EEG Signal

The DFT-CLSM algorithm is also applied to estimate the time-frequency spectrum of EEG signal.

Figure 29 shows the time-frequency spectrum estimated by DFT-CLMS. One visualize from the figure that there are only bright lines in the spectrum. This is because that the EEG signal is a stationary signal, the frequency components of which do not change with time.

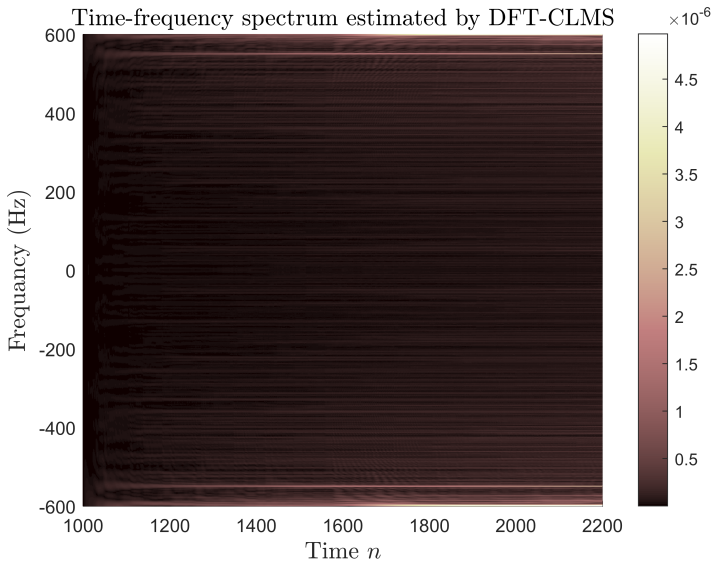


Figure 29: Time-frequency spectrum of EEG signal estimated by DFT-CLMS

4 From LMS to Deep Learning

4.1 Task 1: One-step Ahead Prediction for Time-series

In this part, the one-step ahead prediction is performed for the time-seires from the 'time-series.mat' file using AR(4) model and LMS algorithm. The step size is set to $\mu = 1 \times 10^{-5}$.

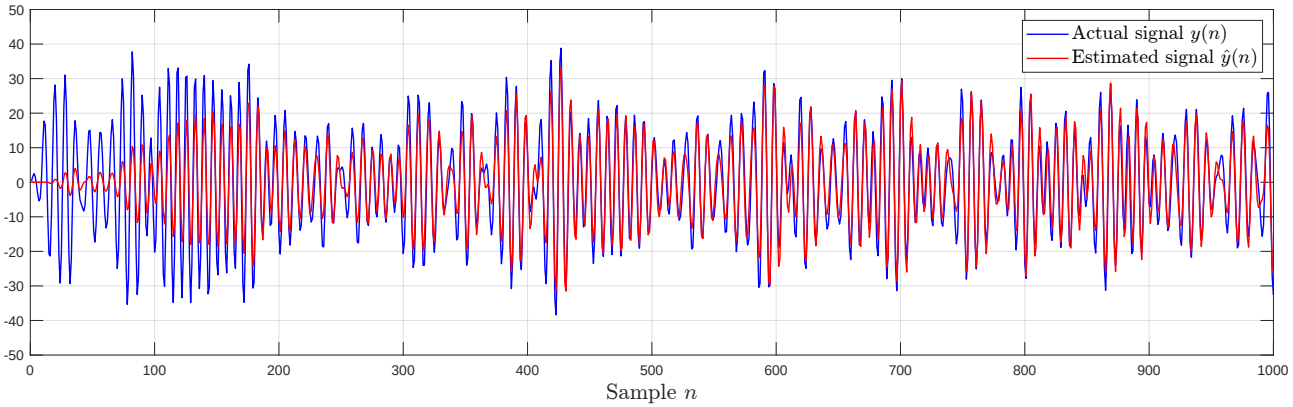


Figure 30: One-step ahead prediction using LMS

Figure 30 shows the actual signal $y(n)$ and the estimated signal $\hat{y}(n)$. We can see that the estimated signal starts from a very small value and gradually reaches to the close values of actual signal. The mean square error (MSE) and prediction gain is shown below.

Table 6: LMS one-step ahead prediction

MSE	40.094
Prediction Gain	5.197

4.2 Task 2: Activation Function

In this part, the `tanh` activation function is used in LMS to perform the one-step prediction for the signal in Part 4.1.

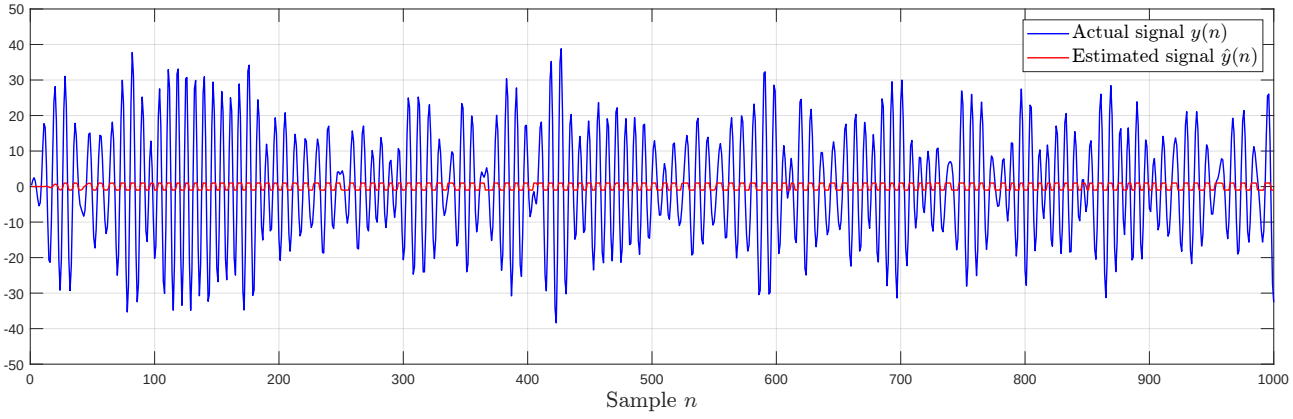


Figure 31: One-step ahead prediction using LMS and activation function `tanh`

Obviously, the estimated signal in Figure 31 only oscillates between -1 and 1. Therefore, this activation function is not appropriate.

4.3 Task 3: Scaled Activation Function

In order to solve the problem in Part 4.2, we can use the scaled activation function (i.e., $a \cdot$ `tanh`). For the signal in Part 4.1, the scaling coefficient α need to be larger than the maximum oscillation range of the signal.

The maximum value of the signal $y(n)$ is 38.8115, so $a = 40$ is selected.

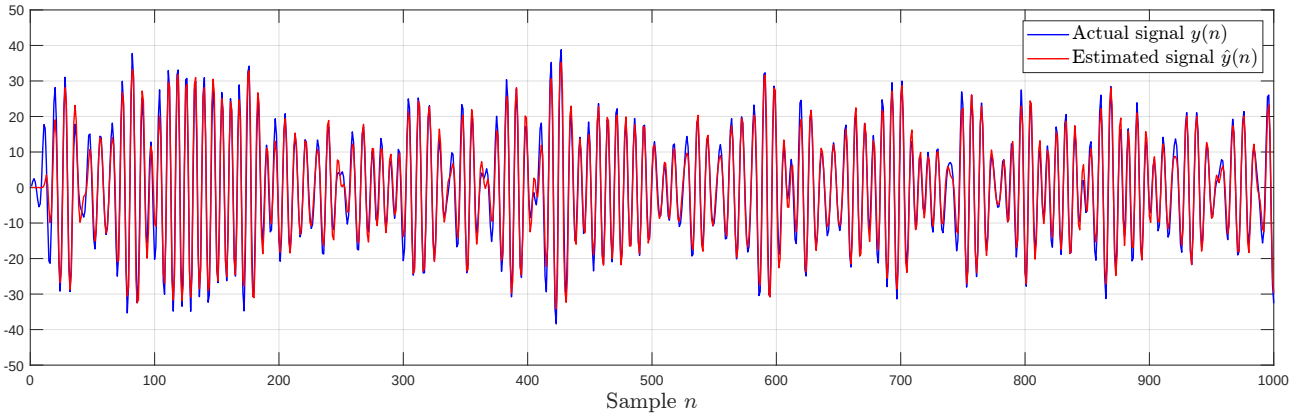


Figure 32: One-step ahead prediction using LMS and activation function $40 \cdot \texttt{tanh}$

Figure 32 shows the signal estimated by LMS using activation function $40 \cdot \texttt{tanh}$. One visualize from this figure that the estimated signal $\hat{y}(n)$ is capable of converging to the close values of the actual signal from the beginning, instead of waiting for a lot of steps as in standard LMS. The MSE and prediction gain of this method is shown below.

Table 7: Scaled activation function for time-series with zero-mean

MSE	8.142
Prediction Gain	14.087

Compared with the MSE and prediction gain of standard LMS in Table 6, the method used in this part has much lower MSE and much higher prediction gain, which shows better performance.

4.4 Taks 4: Time-series with Non-zero Mean

To process the time-series with non-zero mean, we can add a bias input to the model (i.e., $\phi(\mathbf{w}^T\mathbf{x} + b)$, where $\phi(\cdot)$ is the activation function), which can be achieved by argumented input $[1, \mathbf{x}]^T$.

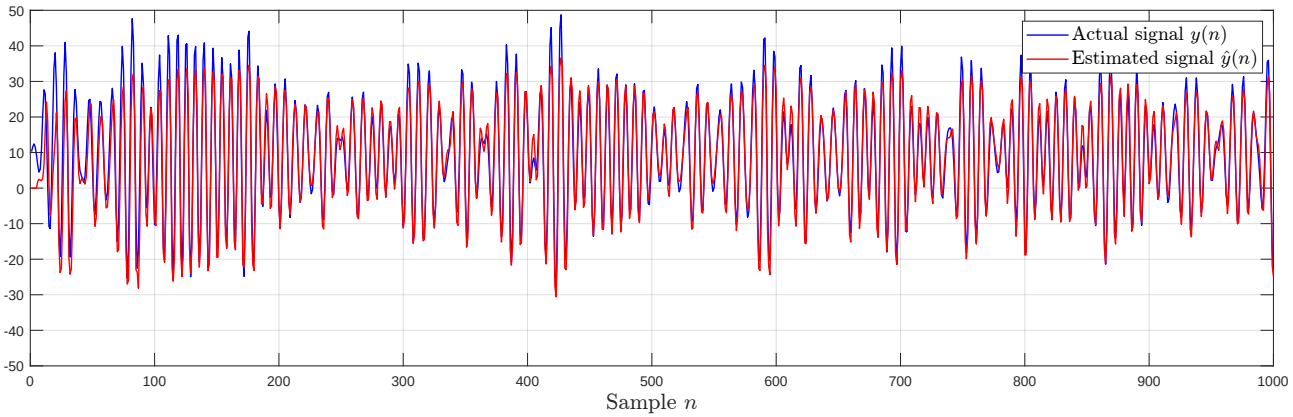


Figure 33: One-step ahead prediction using LMS, activation function $40 \cdot \texttt{tanh}$ and bias input

We can see from Figure 33 that the mean of the estimated signal is zero at the beginning, but it gradually shifts to the mean of the actual signal. The estimated signal also converge to the close values of the actual signal. Table 8 shows the MSE and prediction gain of the method in this part.

Table 8: Bias input for time-series with non-zero mean

MSE	16.179
Prediction Gain	11.509

4.5 Task 5: Pre-trained Weights

For the method in Part 4.4, it need to take a lot of steps to converge to the close values of actual signal. To solve this problem, we can pre-train the weights $\mathbf{w}$ on a small number of samples and use the pre-trained weights as the initial weights.

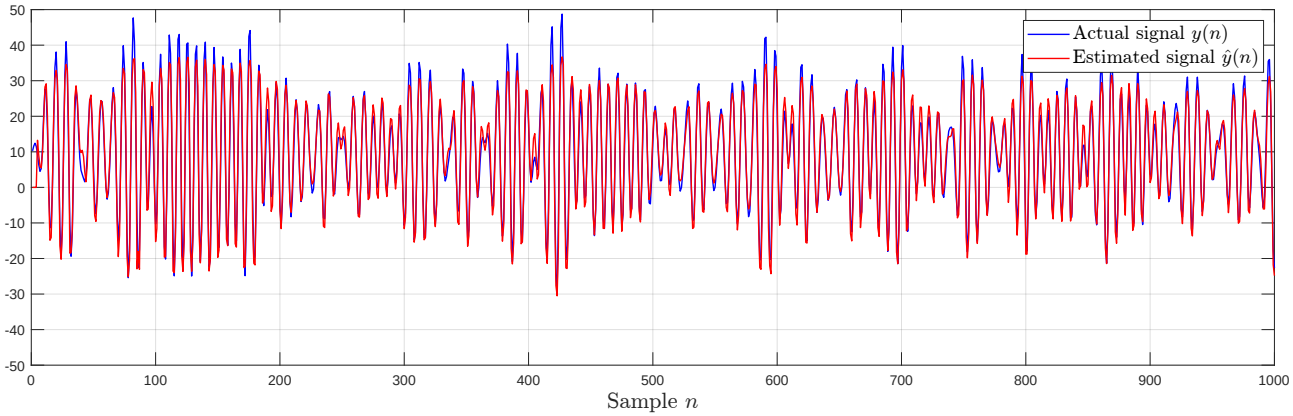


Figure 34: One-step ahead prediction using LMS, activation function $40 \cdot \texttt{tanh}$, bias input and pre-trained weights

One can visualize from Figure 34 that the signal estimated by the method in this part converge to the reasonable value fast. Table 9 shows the MSE and prediction gain of the method using pre-trained weights. Compared with the method in previous part (Table 8), it has better performance.

Table 9: Pre-trained weights for time-series with non-zero mean

MSE	9.173
Prediction Gain	13.954

4.6 Task 6: Backpropagation in Deep Network

A deep network can be trained using backpropagation algorithm, which is investigated in this part by taking the deep network in Figure 35 as example.

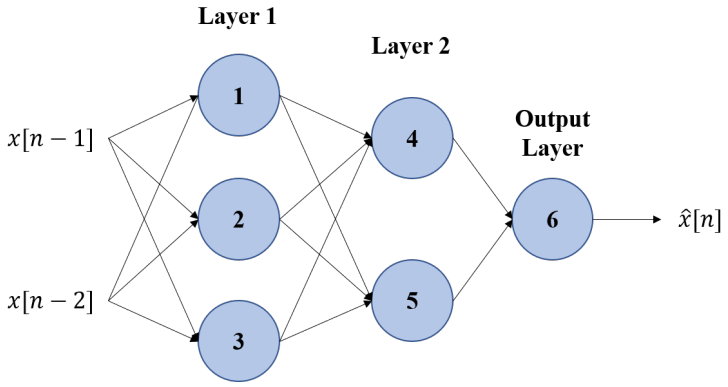


Figure 35: Example of a deep network

Denote the weight between $x[n-i]$ and the j-th neuron in Layer 1 by $w_{n-i,j}$, e.g., the weight between $x[n-1]$ and neuron 1 is $w_{n-1,1}$. Denote the weight between neuron i and neuron j by $w_{i,j}$, e.g, the weight between neuron 1 and neuron 4 is $w_{1,4}$. The activation function at neuron i is denoted by $f_i(\cdot)$. The value and error at neuron i are denoted by v_i and e_i, respectively. The Backpropagation algorithm trains the deep network thorough the following steps.

- **Step 1: Value Forward Propagation:** In this step, the value at each neuron is calculated in the

forward direction. The values of the Layer 1 are firstly calculated. Taking neuron 1 as example,

$$v_1 = f_1(w_{n-1,1}x(n-1) + w_{n-2,1}x(n-2)) \tag{111}$$

The value of v_2 and v_3 are calculated similarly. Then the value of Layer 2 is calculated, e.g.,

$$v_4 = f_4(w_{1,4}v_1 + w_{2,4}v_2 + w_{3,4}v_3) \tag{112}$$

Finally, the output is given as

$$\hat{x}(n) = v_6 = f_6(w_{4,6}v_4 + w_{5,6}v_5) \tag{113}$$

- **Step 2: Error Back Propagation:** In this step, the error at each neuron is calculated in the back direction. The overall error is firstly calculated, which is given as

$$e = e_6 = x(n) - \hat{x}(n) \tag{114}$$

Then the errors at Layer 2 are calculated. The errors at neuron 4 and neuron 5 are given as

$$e_4 = w_{4,6}e_6 \tag{115}$$
$$e_5 = w_{5,6}e_6 \tag{116}$$

Finally, the errors at Layer 1 are calculated. For example, the error at neuron 1 is given as

$$e_1 = w_{1,4}e_4 + w_{1,5}e_5 \tag{117}$$

The erros at neuron 2 and 3 are calculated similarly.

- **Step 3: Weight Update:** In this step, the weights are updated. For example, the weight $w_{1,4}$ update is

$$w_{1,4}(n+1) = w_{1,4}(n) + \mu e_4 x_1, \tag{118}$$

which is similar to the weights update in LMS.

4.7 Task 7 & 8: Deep Learning for Prediction

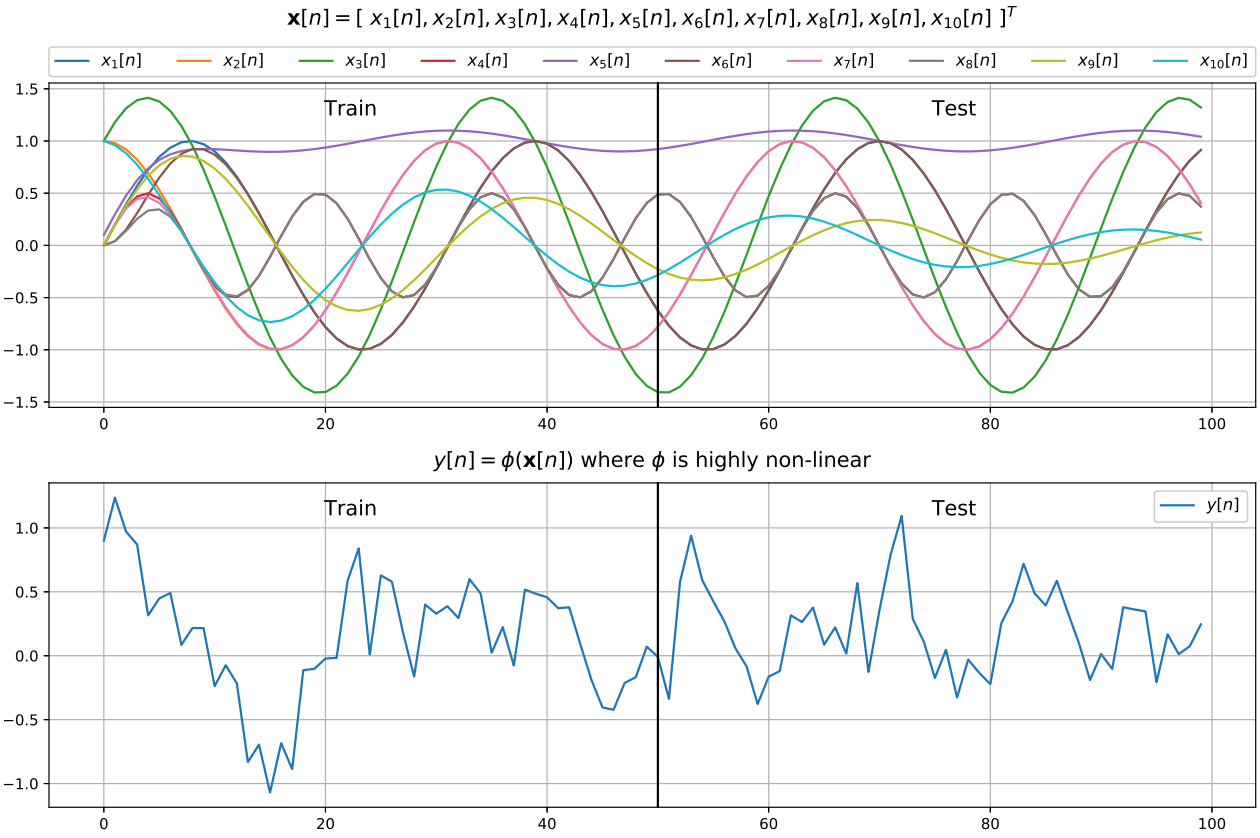


Figure 36: Input signal (top) and target signal (bottom)

In this part, the deep learning method is used for predicting the signal $y(n)$ shown in Figure 36. The deep network used is consist of 4 hidden layers, which is compared with a linear single neuron and a non-linear neuron actived by `tanh`. The learning rate is set to 0.01 and the noise power is 0.05.

Figure 37 shows the outputs of deep network and single neuron against target signal $y(n)$. We can see that the deep network is capable of fitting the target signal better after 20,000 epochs. Figure 38 shows the loss of three methods against epoch number. One can visualize that the test loss of linear and non-linear single neuron starts from a low value (around 0.2), but the initial test loss of deep network is high (around 0.7). Nevertheless, with the increase of epoch number, the test loss of deep network finally converge to a lower value (around 0.08), which is better than that of single neuron (more than 0.1).

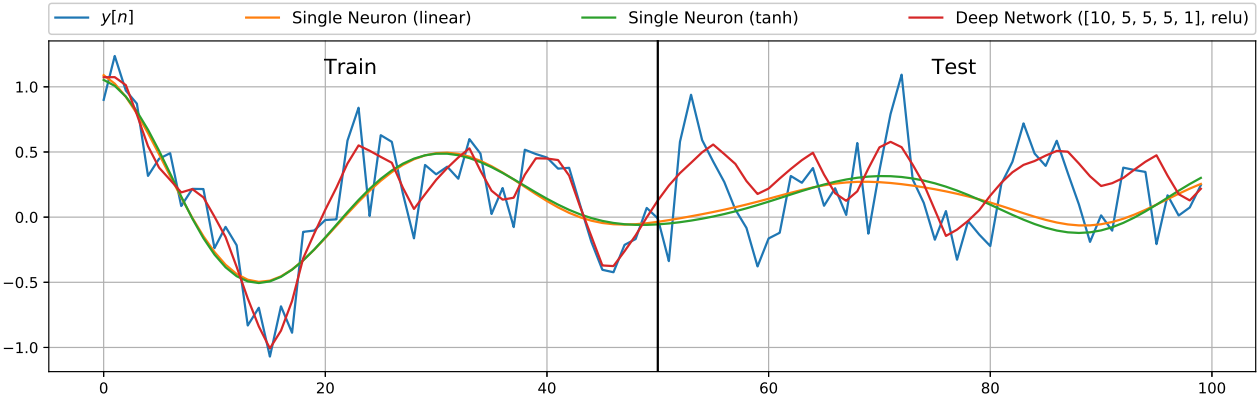


Figure 37: Outputs of deep network and single neuron

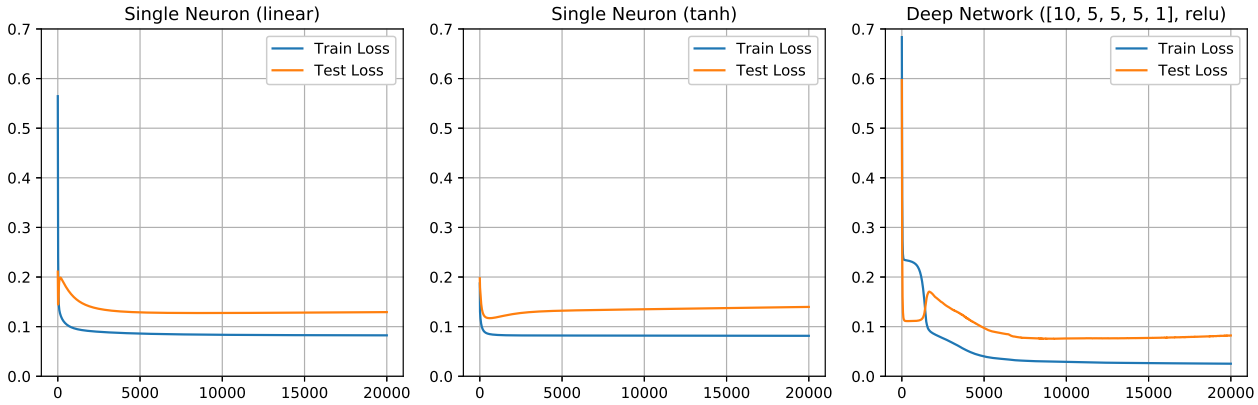


Figure 38: Loss of different methods on train set and test set (noise power is 0.05)

However, the deep network is not suitable for predicting signal with high noise power. Figure 39 and Figure 40 show the train loss and test loss of the three methods for the signal with noise power of 0.5 and 1, respectively. For the high noise power, the single neuron model can still converge. But the deep network can not converge in both train loss and test loss. The test loss of deep network even increase with the epoch number and is worse than the single neuron, which means that there is an over fit in the deep network. This drawback of deep network gets worse when the noise power is higher, i.e., the randomness of the signal gets higher.

Therefore, the deep network cannot predict the random signal like noise.

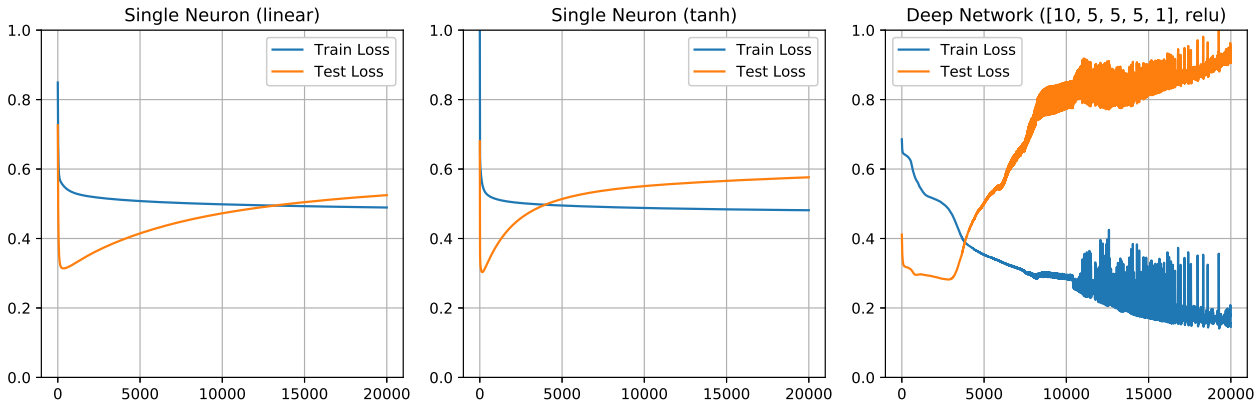


Figure 39: Loss of different methods on train set and test set (noise power is 0.5)

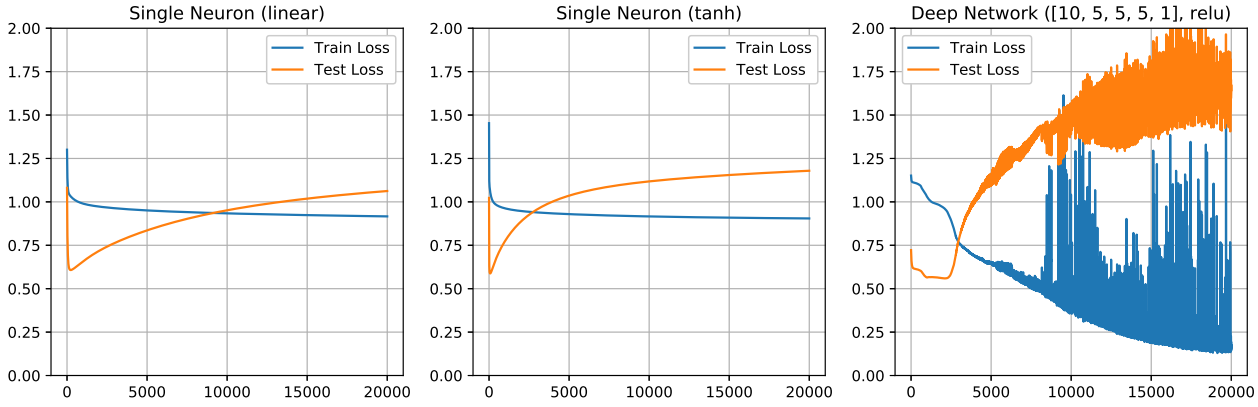


Figure 40: Loss of different methods on train set and test set (noise power is 1)