Advanced Digital Signal Processing Coursework

Vijay S. Gami

CID: 00739377

4/9/2015

Contents

Random signals and stochastic processes 2
1.1 Statistical estimation 2
1.2 Stochastic process 4
1.3 Estimation of probability distributions 6
2 Linear stochastic modelling 8
2.1 ACF of uncorrelated sequences 8
2.2 ACF of correlated sequences 9
2.3 Cross-correlation function 9
2.4 Autoregressive modelling 10
2.5 Real world signals: ECG from iAmp experiment 13
3 Spectral estimation and modelling 15
3.1 Averaged periodogram estimates 15
3.2 Spectrum of autoregressive processes 16
3.3 Spectrogram for time frequency analysis: dial tone pad 18
3.4 Real world signals: Respiratory sinus arrhythmia from RR-intervals 20
4 Optimal filtering – fixed and adaptive 22
4.1 Wiener filter 22
4.2 The least mean square (LMS) algorithm 23
4.3 Gear shifting 25
4.4 Identification of AR processes 26
4.5 Speech recognition 27
4.6 Dealing with computational complexity: sign algorithms 29
5 A Real World Case Study: Vinyl De-noising 31

Random signals and stochastic processes

1.1 Statistical estimation

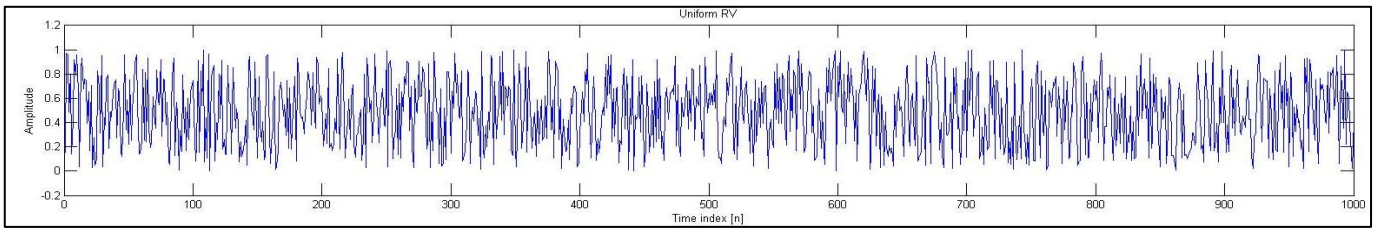

Figure 1: A realization of X_n for 1000 steps where $X_n \sim U(0,1)\ \forall\ n$.

1&2) Theoretical mean and standard deviation:

For $X_n \sim U(0,1),\ \forall n$ The theoretical mean is given by: $m = E\{X_n\} = \int_{-\infty}^{+\infty} x f_X\, dx = \int_0^1 x\, dx = 0.5,\ \forall n$. For a particular realization, **x**, MATLAB computes the sample mean to be 0.5100. The expected value and variance of this estimator are calculated below.

$$E\{\widehat{M}\} = E\left\{\frac{1}{N}\sum_{n=1}^{N} X_n\right\} = \frac{1}{N}\sum_{n=1}^{N} E\{X_n\} = \frac{N}{N}m = m = 0.5 \quad (1)$$

$$\mathrm{Var}\{\widehat{M}\} = \mathrm{Var}\left\{\frac{1}{N}\sum_{n=1}^{N} X_n\right\} = \frac{1}{N^2}\sum_{n=1}^{N} Var\{X_n\} = \frac{\sigma^2}{N} = \frac{1}{12N} \quad (2)$$

These show that the sample mean is an unbiased estimator for all N and all m, and the variance tends to zero as N increases. Therefore the sample mean estimator is consistent.

The standard deviation is given by $\sigma = \sqrt{E\{X - E\{X\}\}^2} = \sqrt{\int_{-\infty}^{+\infty}(x-m)^2 f_x\, dx} = \sqrt{\int_0^1 (x^2 - 2xm + xm^2)\, dx} = \sqrt{\frac{1}{12}}$. For a particular realization **x**, MATLAB computes the sample standard deviation to be 0.2890. The sample standard deviation is a biased estimator. This can be seen by noting that the sample standard deviation is unbiased and the expectation operator does not commute with the square root since the square root is nonlinear. Then using Jensen's inequality and the fact that the square root is a concave function, the sample variance is an underestimate. However it is asymptotically unbiased.

3) Distribution of sample means and sample standard deviations

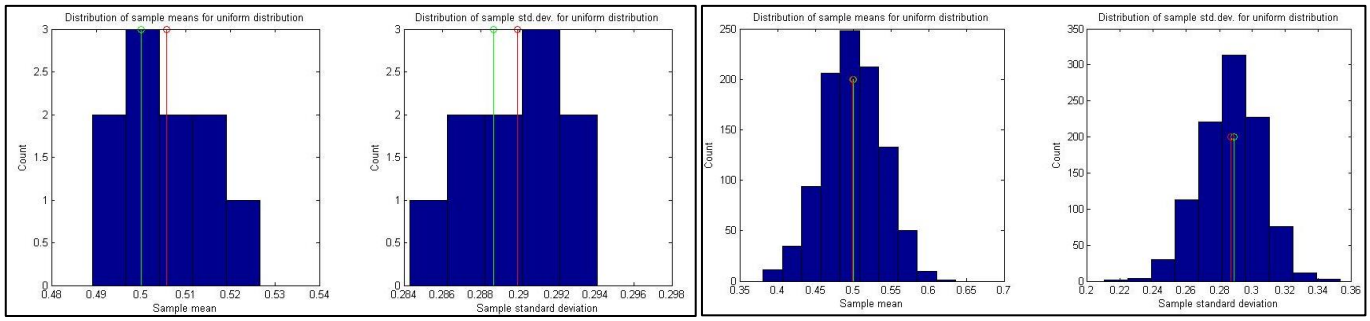

Figure 2: Left: histograms of 10 sample means and 10 sample variances, each computed from a sample size of 1000. Right: histograms of 1000 sample means and 1000 sample variances, each computed from a sample size of 50. Green stems show theoretical mean and variance. Red stem shows the average mean and variance for the samples chosen.

The figures on the right better show the bias in the sample standard deviation since the bias is larger for smaller sample size, and because more sets of samples are collected giving a more reliable mean of the estimator (averaging reduces variance). Note that as the sample size tends to infinite, the distribution for the sample mean tends to Gaussian, which can be explained by the central limit theorem. For the histograms on the right, the sample mean is 0.5001 on average, and the average sample variance is 0.2859 (bias of 0.0028). For a larger sample size of 50,000, the average sample mean and variance is 0.5000 and 0.2886 respectively. This confirms that they are both asymptotically unbiased estimators.

4) Aproximating the PDF with histograms.

The figures below approximate the PDF by area normalized histograms. They demonstrate that more bins are needed to increase the x-axis resolution, but increasing just the bin number and keeping the sample number the same worsens the estimated PDF. This is because there will be fewer samples per bin so the variance of the height of each bin is greater. For an accurate estimate the sample number should be 1000-10000 times greater than the number of bins as shown by the two figures on the right.

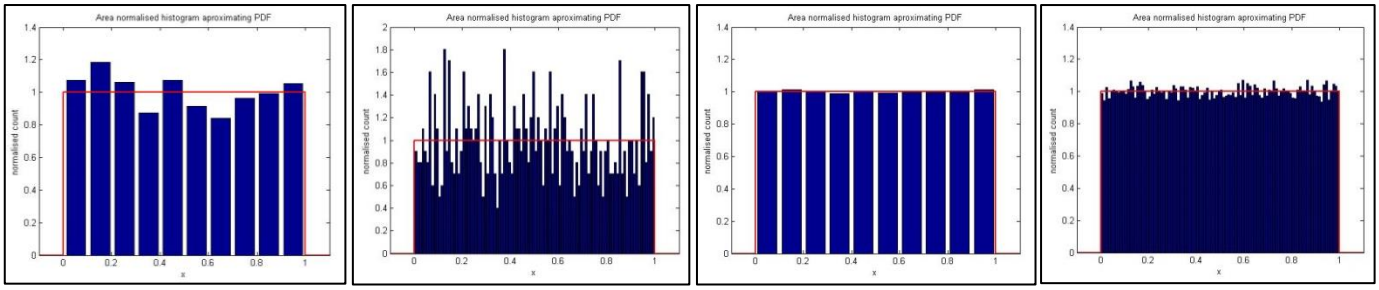

Figure 3: Area normalised histogram approximating the uniform PDF. From left to right, (sample number / Bins): (1000 / 10), (1000 / 100), (100,000 / 10), (100,000, 100). The red line indicates the theoretical PDF.

5) Repeating the above analysis but now with randn()

For $X \sim N(0,1)$, the theoretical mean is 0 and variance is 1. Once again the sample mean is unbiased and consistent. The sample standard deviation is an underestimate, but it is asymptotically unbiased. For a particular realization of **x** size 1000, MATLAB computes the sample mean and variance to be -0.0092 and 1.023 respectively. The histograms below show the distribution of the sample means and variances.

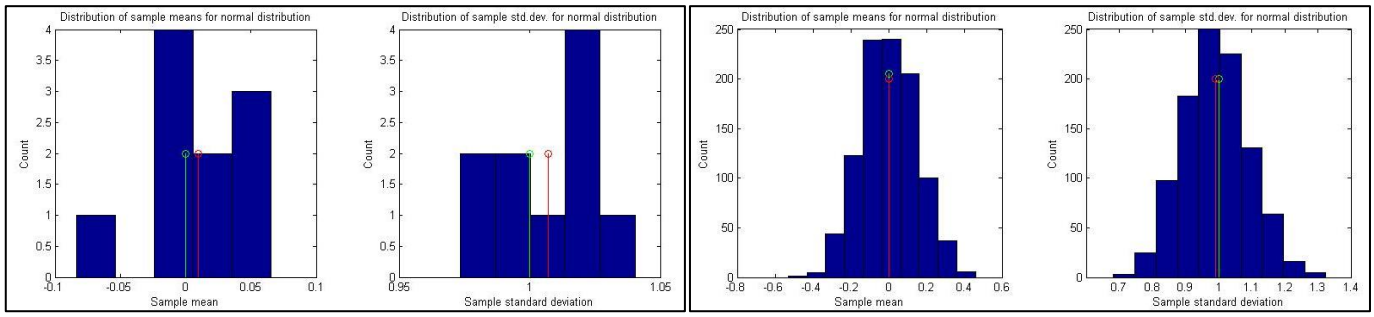

Figure 4: the left shows the distribution of 10 sample means and variances each calculated from a sample size of 1000. The right shows 1000 sample means and variances each calculated from a different sample size of 50.

Note that since the parent distribution is Gaussian, the distribution of the sample means is also Gaussian with mean 0 and variance 1/(sample size). This is more easily confirmed in the figure on the right since more sample means are collected. MATLAB calculates the average of the 10 ensemble mean and standard deviation to be 0.0104, and 0.9897 respectively. For the 100,000 ensemble case, the average calculated mean and standard deviation is -2.7e-5 and 0.9998.

Approximating the PDF with histograms:

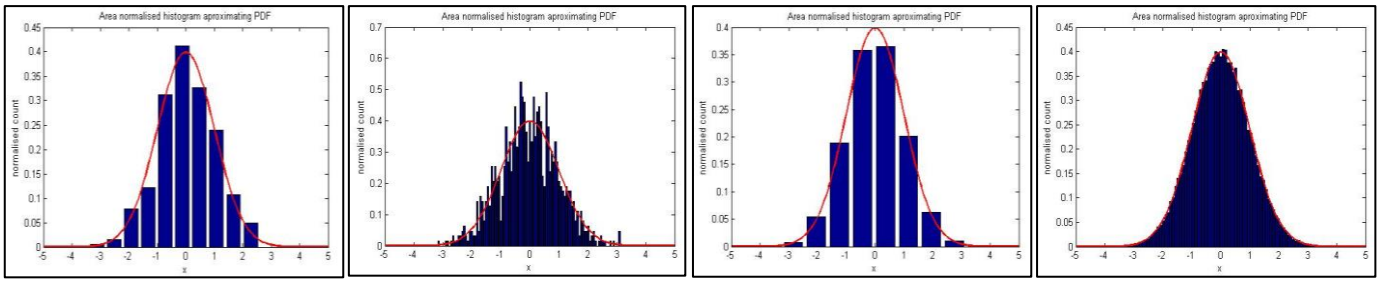

Figure 5 area normalised histogram approximating the ideal PDF which is shown in red. From left to right sample (number / Bins): (1000 / 10), (1000 / 100), (100,000 / 10), (100,000, 100).

These figures also demonstrate that the PDF estimate improves when more samples are used, and the x-axis resolution increases with bin number. More samples are needed when the bin number is higher such that the count for each bin is high in order to have good y axis resolution for each bin and reliable (low variance) amplitude.

1.2 Stochastic process

1) Ensemble mean and standard deviation for each process:

For a stationary process, a shift in the time origin would be impossible to detect. The figures below show that for RP2 and RP3, the mean and variance do not change with time, which means both RP2 and RP3 are stationary with respect to the mean and variance. Upon inspection of the generating functions for RP2 and RP3, we see each the samples are independent from each other. Since the mean is stationary, and the samples are independent, the autocorrelation function, ACF for depends only on the time difference. Hence these processes are at least wide sense stationary. However for RP1, the mean is increasing and the standard deviation has a half sinusoid shape, and so it is non-stationary. It is easy to see from the code that RP1 is not stationary since the generating mechanism changes with the time index. Both *'Mc'* (which determines the variance), and *'Ac'*, (which determines the mean) change with n.

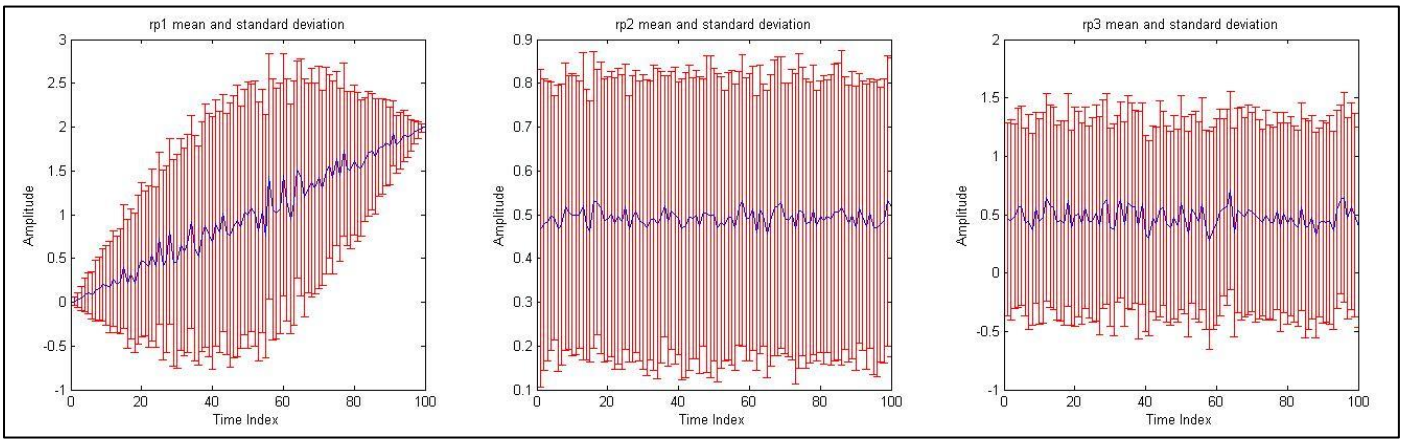

Figure 6: ensemble mean (blue) and standard deviation from the mean (red) for the three random processes with 100 members of the ensemble each 100 samples long. Obtained by the 'errorbar' function in MATLAB.

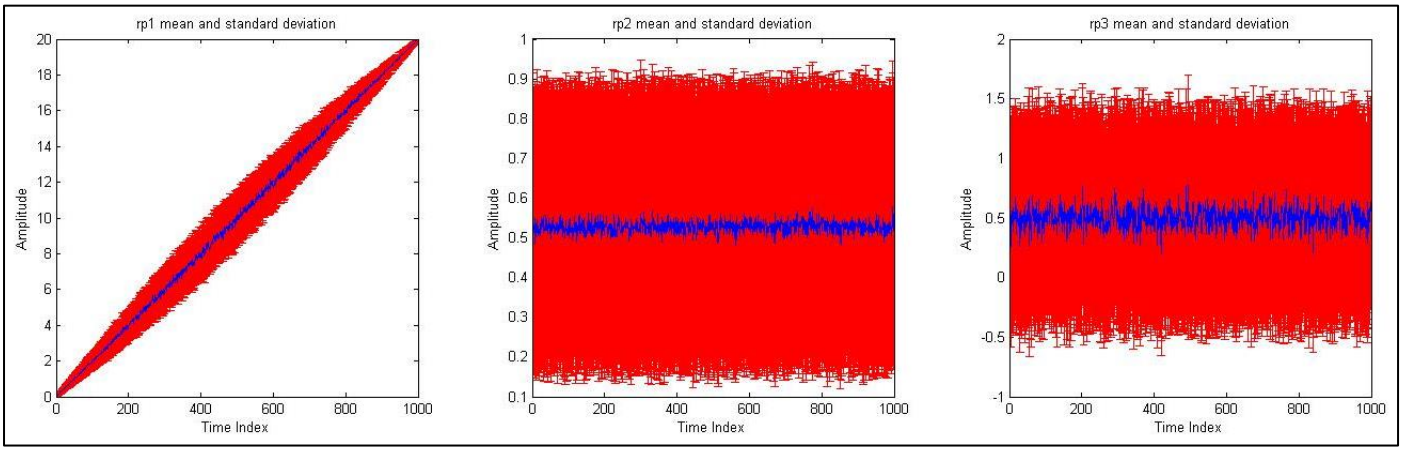

Figure 7: Ensemble mean and standard deviation for the three random processes with 100 members of the ensemble each 1000 samples long. This makes it clearer that the mean of rp1 is changing with time, and that rp2 and rp3 are stationary.

2) Comments on ergodicity of each process:

	rp1			rp2			rp3		
	μ	σ	σ^2	μ	σ	σ^2	μ	σ	σ^2
Realization 1	10.0002	5.8757	34.5234	0.6116	0.1778	0.0316	0.4582	0.8542	0.7296
Realization 2	10.0295	5.8816	34.5932	0.8707	0.2434	0.0592	0.4750	0.8644	0.7472
Realization 3	10.0470	5.9022	34.8355	0.0239	0.2758	0.0761	0.5424	0.8609	0.7411
Realization 4	9.9767	5.8599	34.3387	0.8311	0.0773	0.0060	0.5338	0.8538	0.7289

Table 1: Temporal mean, standard deviation and variance for each process:

Table 1 suggests RP3 to be ergodic with respect to the mean and variance. This is because the temporal mean and standard deviation are within a few percent (<10% deviation) of the ensemble mean and standard deviation. (Theoretical mean is 0.5 and theoretical variance is 0.75 as calculated in the next section). In contrast, the time statistics of RP2 do not match the ensemble statistics hence it is not ergodic. For example, the theoretical ensemble mean is 0.5, but the temporal mean calculated for the 4 realizations differ by up to 96%. RP1 cannot be ergodic since ergodic processes are a subset of stationary processes. This is confirmed in the table since the mean is 10.0, but Figure 7 clearly shows the mean to be varying with time from 0 to 20.

3) Mathematical description of rp1, rp2, rp3 and theoretical (ensemble) means and variances:

$$rp1:\ v_a[n] = (X_n)\left(5\sin\left(\frac{n\pi}{N}\right)\right) + 0.02n, \quad where \quad X_n \sim U(-0.5, 0.5) \tag{3}$$

$$rp2:\ v_b[n] = X_n Y + Y, \quad where \quad X_n \sim U(-0.5, 0.5),\ \ Y \sim U(0,1),\ \ (X \perp Y) \tag{4}$$

$$rp3:\ v_c[n] = 3X_n + 0.5, \quad where \quad X_n \sim U(-0.5, 0.5) \tag{5}$$

$$\mathrm{E}\{v_a[n]\} = \left(5\sin\left(\frac{n\pi}{N}\right)\right) \times \mathrm{E}\{(X_n)\} + \mathrm{E}\{0.02n\} = 0.02n \tag{6}$$

$$\mathrm{E}\{v_b\} = \mathrm{E}\{X_n Y\} + \mathrm{E}\{Y\} = \mathrm{E}\{X_n\}\mathrm{E}\{Y\} + \mathrm{E}\{Y\} = 0.5 \tag{7}$$

$$\mathrm{E}\{v_c\} = 3\mathrm{E}\{X_n\} + \mathrm{E}\{0.5\} = 0.5 \tag{8}$$

$$\mathrm{Var}\{v_a[n]\} = \left(5\sin\left(\frac{n\pi}{N}\right)\right)^2 \mathrm{Var}\{X_n\} = \frac{25\sin^2\left(\frac{n\pi}{N}\right)}{12} \tag{9}$$

$$Var\{v_b\} = Var\{X_n Y\} + Var\{Y\} = Var\{X_n\}Var\{Y\} + Var\{Y\}E\{X_n\}^2 + Var\{X_n\}E\{Y\}^2 + 1/12 = 1/9 \tag{10}$$

$$Var\{v_c\} = 3^2 Var\{X_n\} = 0.75 \tag{11}$$

Theoretical (temporal) means and variances:

$$\mathrm{E}\{v_a\} = \frac{1}{N}\sum_{n=1}^{N} v_a[n] = 0.02 \times \frac{(N+1)}{2} = 10.01 \qquad (on\ average\ for\ N = 1000) \tag{12}$$

$$\mathrm{E}\{v_b\} = \mathrm{E}\{X_n Y\} + \mathrm{E}\{Y\} = \mathrm{E}\{X_n\}\mathrm{E}\{Y\} + \mathrm{E}\{Y\} = \mathrm{y} \qquad \text{where } Y \sim U(0,1) \tag{13}$$

$$\mathrm{E}\{v_c\} = 3\mathrm{E}\{X_n\} + 0.5 = 0.5 \tag{14}$$

$$Var\{v_a\} = \frac{1}{N}\sum_{n=1}^{N}(v_a[n] - 10.01)^2 = \frac{1}{N}\sum_{n=1}^{N}\left(\left(X_n 5\sin\left(\frac{n\pi}{N}\right)\right)^2 + (0.02n - 10.01)^2\right)$$
$$= \left(\frac{1}{12}\sum_{n=1}^{N}\left(5\sin\left(\frac{n\pi}{N}\right)\right)^2\right) + 33.3333 = \frac{12.5}{12} + 33.3333 = 34.4 \tag{15}$$
$$(on\ average\ for\ N = 1000)$$

$$Var\{v_b\} = y^2 Var\{X_n\} = \frac{y^2}{12} \qquad \text{where } Y \sim U(0,1) \tag{16}$$

$$Var\{v_c\} = 3^2 Var\{X_n\} = 0.75 \tag{17}$$

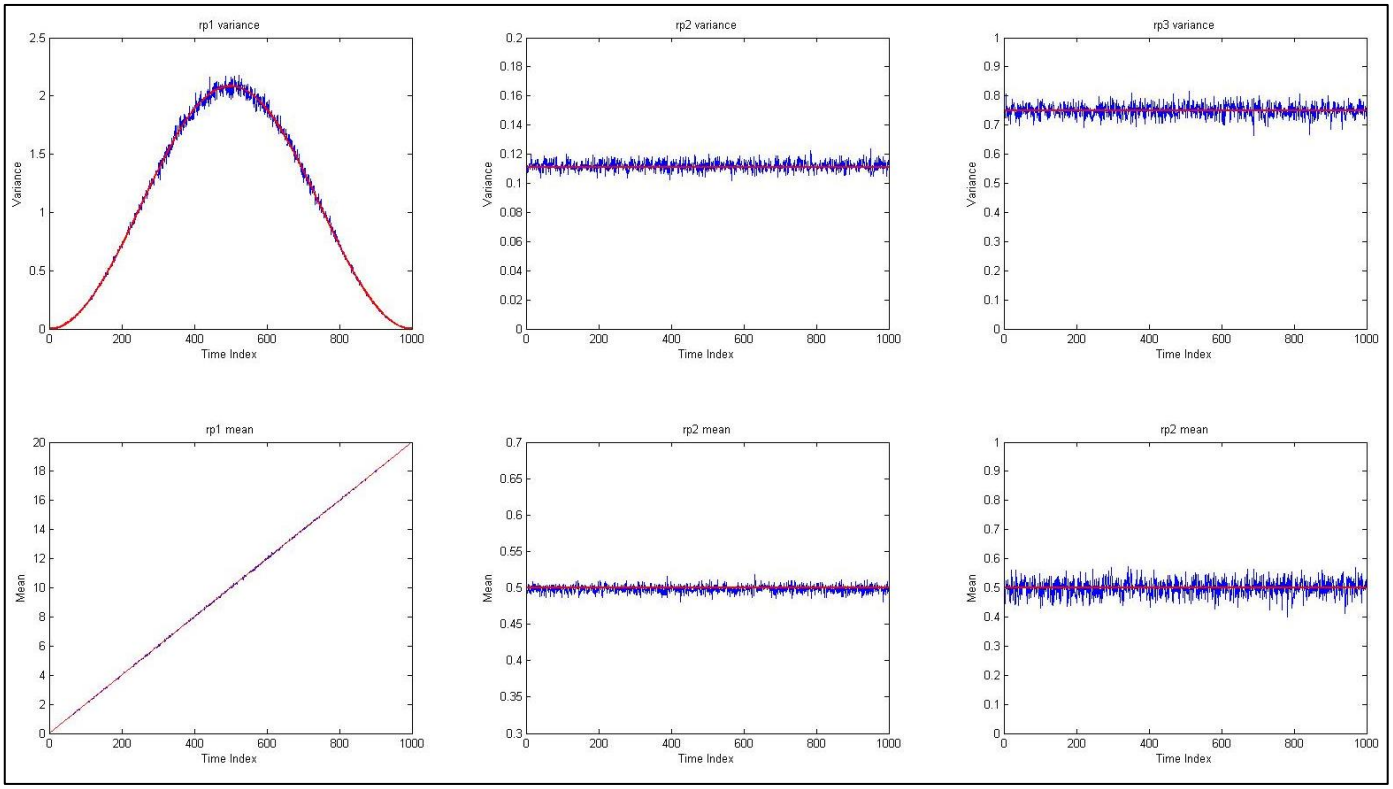

Figure 8: Comparing theoretical ensemble mean and variance (red) with the ones obtained by sample averaging (blue). 1000 sample functions of length 1000 were used. The figures show the theoretical values agree with the ensemble values.

1.3 Estimation of probability distributions

1) Testing the '*pdf*' function:

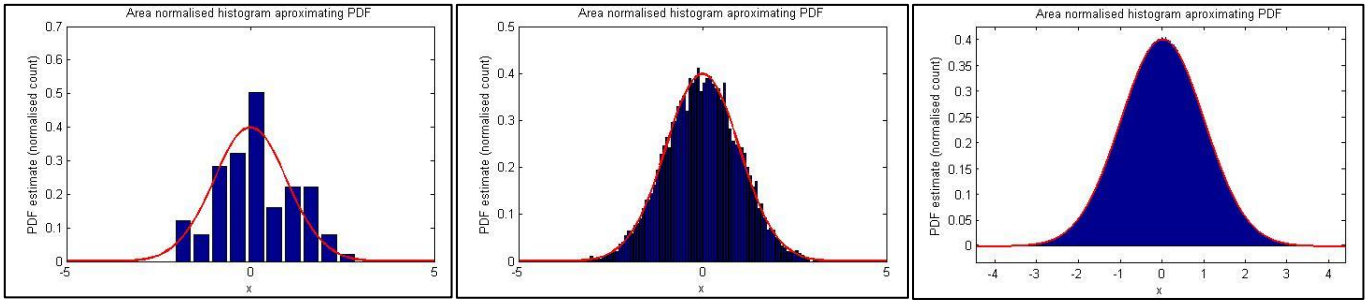

Figure 9: The pdf function output when input is WGN of different lengths. Ideal pdf is shown in red.

The function takes in 2 arguments: the first is the data, and the second is the number of bins to use in the histogram plot. The figures above show the estimated pdf for increasing data lengths and bin numbers from left to right. The higher sample length number along with higher bin number results in the best estimate. Within the function there is a switch to toggle the format of the output. When the switch is set to 1, the output graph is a line graph and not a histogram. This option is demonstrated in the next section.

2) Approximated PDF for the ergodic processes of section 1.2:

Only RP3 is ergodic with respect to the at least the mean and variance. The theoretical distribution of this process is given by equation 5, and is shown by the red line in the figure below. The figures below also show the approximate PDF (blue line). They have been obtained using sample lengths of 100, 1000, and 10,000. Increased N gives results which are closer to the theoretical.

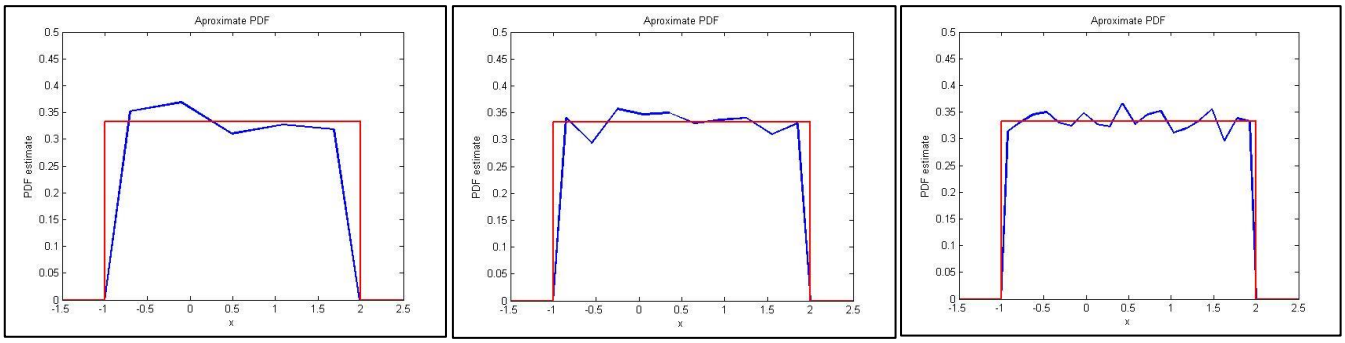

Figure 10: Estimated densities for random process RP3. From left to right, (Data length / Bins) : (100 / 5), (1000 / 10), (10,000 / 20)

3) Estimated PDF of non-stationary process:

If the process is non-stationary, the PDF is changing with time so estimating it over time will give smeared and inaccurate results. Figure 11 (left) shows the PSD estimate of a Gaussian process whose mean step changes from 0 to 1 after the 500th sample. The approximated PDF looks like the average of the two separate PDFs which are shown by the two figures on the right. These have been obtained by splitting the series into the two stationary halves.

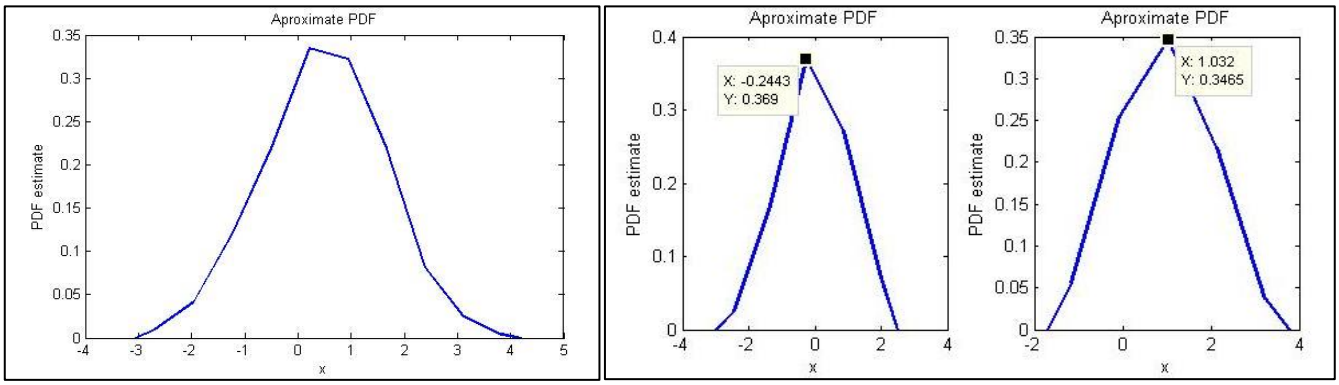

Figure 11: (Left) PDF estimate of the non-stationary process. (Right) the PDF estimate of the non-stationary process split into two stationary halves.

To deal with non-stationary we need short sliding windows which can effectively capture the instantaneous PDF. The length of the window needs to take into account the rate at which the process is moving. A slow process can have longer windows which contain more data and hence give a better estimate of the PDF. In short it is harder to estimate the instantaneous PDF of a fast moving process since smaller frames of data must be used. To demonstrate the concept of sliding windows, a Gaussian process with unit variance whose mean changes slowly from 0 to 10 from sample 0 to sample 500,000 has been created. The PDF estimate of the whole sample sequence is shown by Figure 12 (left), and this is clearly smeared due to the 'sliding' instantaneous PDF.

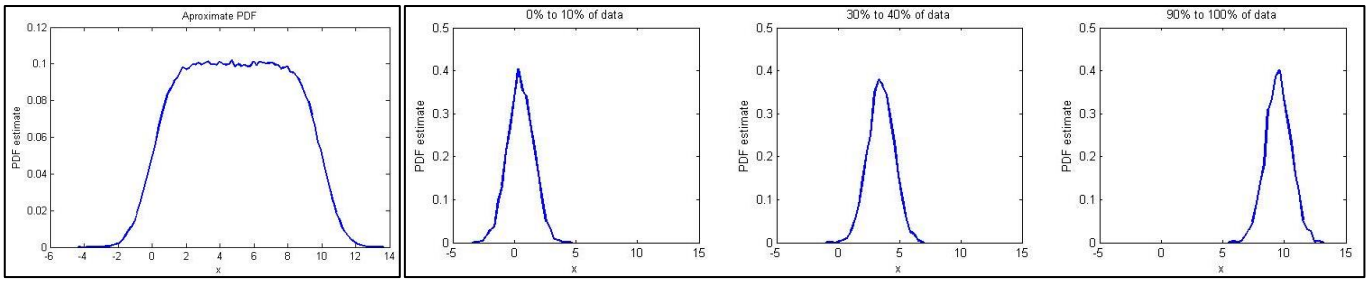

Figure 12: (left) PSD of no stationary process. (Right) PSD obtained by sliding window of length 10% of total data length.

Figure 12 (right) shows 3 'instantaneous' estimates of the PDF obtained by using only 10% of the data at a time. The frame of data is taken at three different time instants and now it is clear that the process is Gaussian with an increasing mean. A smaller window length would give better time resolution and a more instantaneous estimate, but a worse PDF estimate and so a compromise is needed.

2 Linear stochastic modelling

2.1 ACF of uncorrelated sequences

1) Unbiased estimate of the ACF for a WGN (white Gaussian noise) realization:

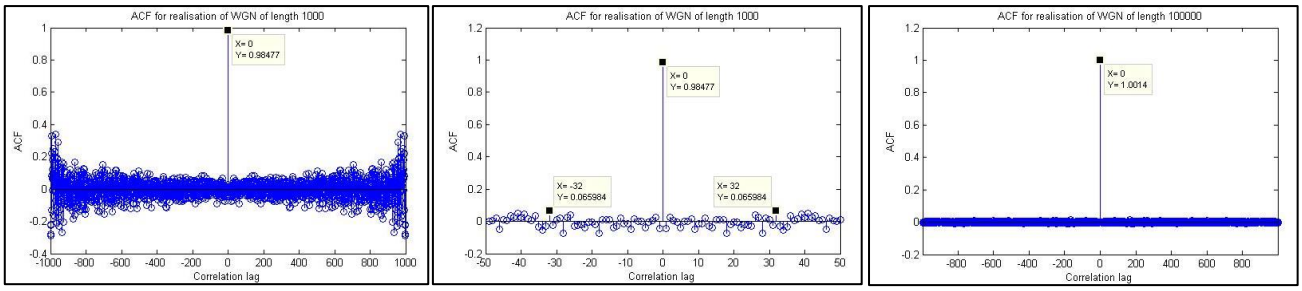

Figure 13: Unbiased estimate ACF for realisations of WGN (zero mean, unit variance). Left is obtained with 1000 samples, middle is close-up of the left figure and the right figure is obtained with 100,000 samples.

For a zero mean independent stochastic process, the ACF is a discrete Dirac function. Intuitively this is because knowing one sample gives no information on other samples due to independence. Mathematically this can be shown by: $R_X(n,m) = E\{X_n X_m\} = E\{X_n\}E\{X_m\} = 0\ (for\ n \neq m)$. For the case $n = m$, $R_X(n,n) = E\{{X_n}^2\} = \sigma^2$. For such a signal ergodic with respect to at least the second order averages (eg mean, variance, ACF), this is also the signal power. Figure 13 shows an unbiased estimate for the ACF of WGN. It shows amplitude 0.98477 for 0 lag, and mostly non-zero amplitude for non-zero lag. As more samples are used, the estimate accuracy increases since the variance of the estimator reduces as shown by Figure 13 (right). This one better resembles a discrete Dirac function.

Figure 13 (middle) shows the symmetry of the ACF function which can be proved simply from the definition and using the fact that for a WSS process, (ergodicity implies WSS), the ACF depends only on the time difference:

$$R_X(\tau) = R_X(n, n+\tau) = E\{X_n X_{n+\tau}\} = E\{X_{n+\tau} X_n\} = E\{X_i X_{i-\tau}\} = R_X(-\tau) \tag{18}$$

Proof that the unbiased ACF estimate is also symmetric:

$$\hat{R}_X(-\tau) = \frac{1}{N-|\tau|}\sum_{n=0}^{N-1-|\tau|} x[n]x[n-\tau] = \frac{1}{N-|\tau|}\sum_{i=\tau}^{N-1} x[i+\tau]x[i] = \frac{1}{N-|\tau|}\sum_{i=0}^{N-1-|\tau|} x[i]x[i+\tau] = \hat{R}_X(\tau) \tag{19}$$

2/3) Effects of large autocorrelation lag on the accuracy of the ACF estimate:

The larger the lag relative to N, the less accurate the ACF estimate. This is because fewer samples are used to compute the ACF i.e. only $N - |\tau|$ samples. This means greater variance in the estimate at higher lags. Figure 14 shows the ACF for large lags. The $|\mathrm{ACF}|$ at some lags is greater than 0.2, especially for lags above 970. Comparing to Figure 13 (middle), which only shows lags up to 50, all the values are within ± 0.15. The more samples that are used, the lower the variance of the estimator and hence the greater the accuracy since the estimator is unbiased. This is also illustrated in Figure 13 (left) which shows a greater spread in ACF at higher lags.

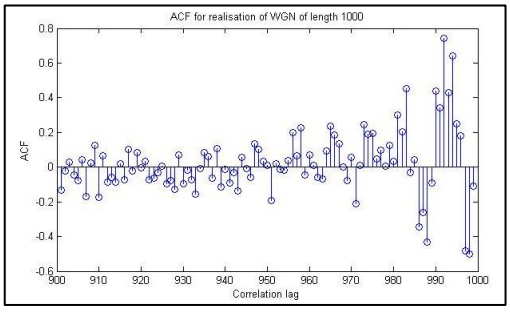

Figure 14: ACF shown for large lags

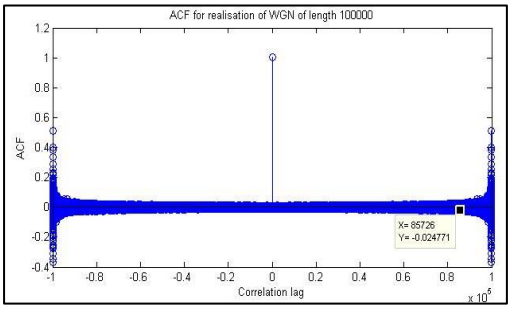

Figure 15: Determining empirical bound on τ

An empirical bound on τ could be related to the minimum number of samples required for a good estimate. For example, if 600 samples are required for a good estimate, and the number of samples is 1000, the max lag is 400. I used 600 as an example considering Figure 13 (left) based on the idea that a typical 15% maximum difference from the ideal Dirac would be statistically reliable. It can be more useful to consider the max lag relative to the number of samples. Figure 15 with 100,000 samples suggests the relative reliability decreases significantly after a lag greater than 85% of the sample number. I used the term relative reliability since the ACF will of course be more reliable if more samples are used. A final note is that this percentage will be lower for lower sample number, for example 65% considering Figure 13 (left) which was obtained with only 1000 samples.

2.2 ACF of correlated sequences

1/2) WGN through MA filter

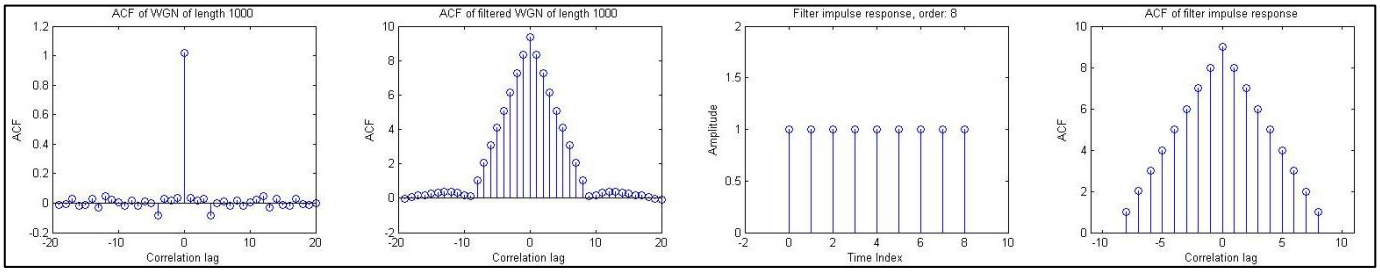

Figure 16: ACF of WGN through filter with impulse response as shown.

The ACF of the output stochastic process Y_n is given by $R_Y(\tau) = R_X(\tau) * R_h(\tau)$. The impulse response of the filter is shown in Figure 16 along with its ACF. Note that the unbiased ACF estimate formula was not used in this case since the data can be assumed infinite in duration, (infinite number of zeros following the rectangle). The ACF of the rectangular function is the triangle function since the convolution of two rectangular functions is a triangle. Since convolution with an impulse function simply means the function is copied and centered at the location of the impulse, the ACF of the output is also a triangle function centered at 0. We would expect the ACF of the output to become zero for lags greater than $q = 8$ = filter order, but this is not the case with this realization of WGN. This is because the impulse function is not ideal since a finite number of WGN samples are used to estimate the ACF. MATLAB shows the ACF for lags greater than q to tend to zero as length of the WGN is increased.

The filters impulse response is $q + 1$ samples long, and the length of its ACF of this is $2q + 1$. The height of the triangle is $q + 1$ since the rectangle function is of height 1. In summary, if the filter order is increased, the triangle part of the ACF of the output increases in height and width. However, since the ACF of the WGN is an estimate and the impulse is not ideal, the non-zero terms contribute to some of the main triangle shape. For much larger order this contribution is greater and is shown by a slightly distorted triangle as illustrated in Figure 17.

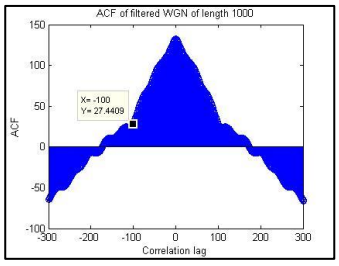

Figure 17: ACF of filtered WGN. Filter order is 100.

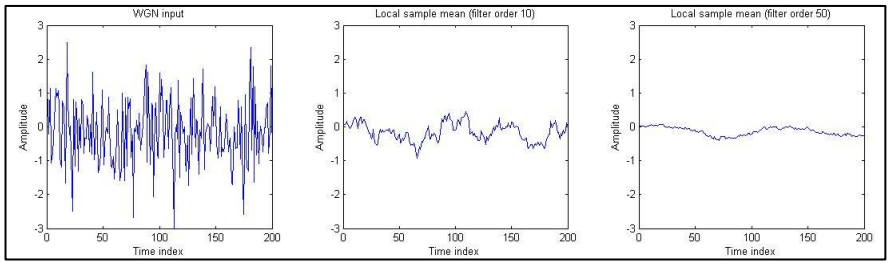

Figure 18: Local sample mean of WGN input computed by MA filter of order 10 (middle) and 50 (right).

It is possible to obtain a local sample mean by dividing the filter coefficients by $q + 1$. This is simply because the MA filter is just a sum of scaled current and q past inputs, so doing this sum then dividing by $q + 1$ would give the average of the current and past $q + 1$ inputs. This is illustrated in the Figure 18 which shows such an average to be calculated with filters of different orders. The greater the order the more accurate and consistent the local sample mean since the variance will be less.

If the input X_n is an uncorrelated sequence, its ACF is a Dirac. Since $R_Y(\tau) = R_X(\tau) * R_h(\tau)$, this means $R_Y(\tau)$ is simply the ACF of the filters/systems impulse response. The relationship $R_Y(\tau) = R_X(\tau) * R_h(\tau)$ can be proved as follows:

$$R_Y(\tau) = E\{y[n]y[n+\tau]\} = E\left\{\sum_i h[i]x[n-i]\sum_j h[j]x[n+\tau-j]\right\} = \sum_i h[i]\sum_j h[j]E\{x[n-i]x[n+\tau-j]\} \quad (20)$$

$$R_Y(\tau) = \sum_i h[i]\sum_j h[j]R_x(\tau-j+i) = \sum_k h[j+k]\sum_j h[j]R_x(\tau-k), \quad (substitute\ i = j+k) \quad (21)$$

$$R_Y(\tau) = \sum_k R_x(\tau-k)\sum_k h[j+k]\,h[j] = \sum_k R_x(\tau-k)R_h(k) = R_h(\tau) * R_x(\tau) \quad (22)$$

2.3 Cross-correlation function

1/2) CCF estimate between input and output, and system identification:

The CCF between input and output is given by $R_{XY}(\tau) = h(\tau) * R_X(\tau)$. This can be proved as follows:

$$R_{XY}(\tau) = E\{X_n Y_{n+\tau}\} = E\left\{\sum_{r=-\infty}^{\infty} h(r)X_{n+\tau-r}\,X_n\right\} = \sum_{r=-\infty}^{\infty} h(r)E\{X_{n+\tau-r}X_n\} = \sum_{r=-\infty}^{\infty} h(r)R_{XY}(\tau-r) = h(\tau) * R_X(\tau). \quad (23)$$

Since the input is WGN, the ACF is a Dirac function. This means the CCF will take the shape of the impulse response of the filter. Therefore an unknown systems impulse response can be identified by a inputting a sufficiently long WGN sequence and taking the CCF of the output and input. Long length is necessary to ensure a good estimate for the CCF in the sense of small variance.

This is demonstrated in Figure 19. The imperfections arise from the fact that the WGN input is of finite length so the CCF estimate has some variance. In an ideal case we would expect the CCF to be zero for lag < 0 and for lag > filter order.

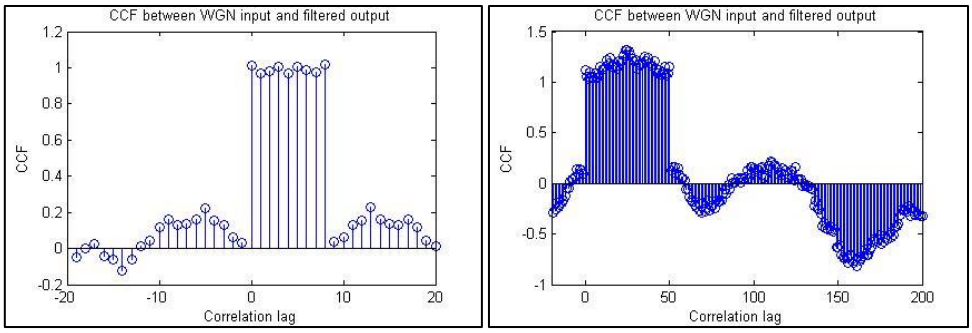

Figure 19: Left is CCF, R_{xy}, for the MA filter from section 2.3 (Figure 16) of order 8. Right is same filter but of order 50.

From this figure it is clear that increasing the order of the filter simply lengthens the impulse response and hence the CCF. For a greater order relative to the length of the input WGN, the output is more distorted than the ideal. This is because when the filters impulse response is long, more of the small non zero values on either side of the Dirac pulse are included in the summation. For reliable identification of the systems impulse response, the WGN input length should be much greater than the estimated order of the system.

2.4 Autoregressive modelling

1) Convergence of an AR(2) process:

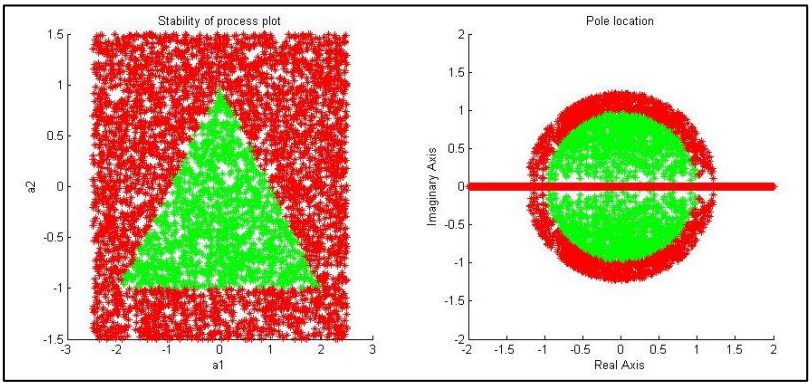

Figure 20: (Left), red shows unstable [a1, a2] of an AR(2) process. Green shows stable pairs. (Right), pole locations for stable and unstable systems. 5000 pairs have been plotted even though 100 were asked for since this gave a better visual

The pairs of (a1, a2) which preserve the stability of the AR(2) process are shown in Figure 20 (left). The two associated poles with each system are shown on the right, with the red pairs corresponding to an unstable system. The system is BIBO stable iff the infinite impulse response is absolutely summable. This statement is equivalent to all the poles lying within the unit circle. Figure 20 (right) confirms this since there are no green points outside the unit circle. The reason for red points on the real axis within the circle is due to the requirement that both the poles associated with each (a1, a2) pair lie within the circle. If one pole is within the circle and one is outside the system is still unstable, hence the reason for some red crosses within the circle. Note that this can only be the case when both the poles are real; therefore the red points within the circle are constrained to the real axis.

The stability triangle (Figure 20, left) can be derived considering the bilinear transform which can map a stable discrete time system to a stable continuous time system. First applying the bilinear transform we end up with the following polynomial in the S-domain:

$$S^2(1 - a_1 - a_2) + S(2 + 2a_2) + (1 + a_1 - a_2) \tag{24}$$

For stability of a continuous time system, the roots of this equation need to be in the LHP, which means the coefficients need to have the same sign. This results in the following constraints which agree with Figure 20 (left):

$$a_1 + a_2 < 1, \qquad -a_1 + a_2 < 1, \qquad a_2 > -1 \tag{25}$$

2) Sunspot time series ACF

The ACF of zero mean data gives an indication of the average frequency content of the signal. This is because the frequency content depends on the rate of change of the amplitude against time, and hence can be measured by correlating the sequence at index n and $n+\tau$. Some properties are preserved by the ACF of a zero mean sequence, for example if the signal is periodic, the ACF will have the same period.

However in the presence of a DC offset (the mean, m), this useful information can get obscured. The mean adds the following term to the ACF estimate at lag τ: $\frac{1}{N-|\tau|}\sum_{n=0}^{N-1-|\tau|}(x[n]\times m + x[n+\tau]\times m + m^2)$. The ACF of the sunspot data is shown in Figure 21 for different data lengths. The zero mean sunspot data ACF is also shown. Mostly it appears that the non-zero mean data's ACF is shifted up by the mean squared, and the period of oscillations can still be seen even from the nonzero mean ACF.

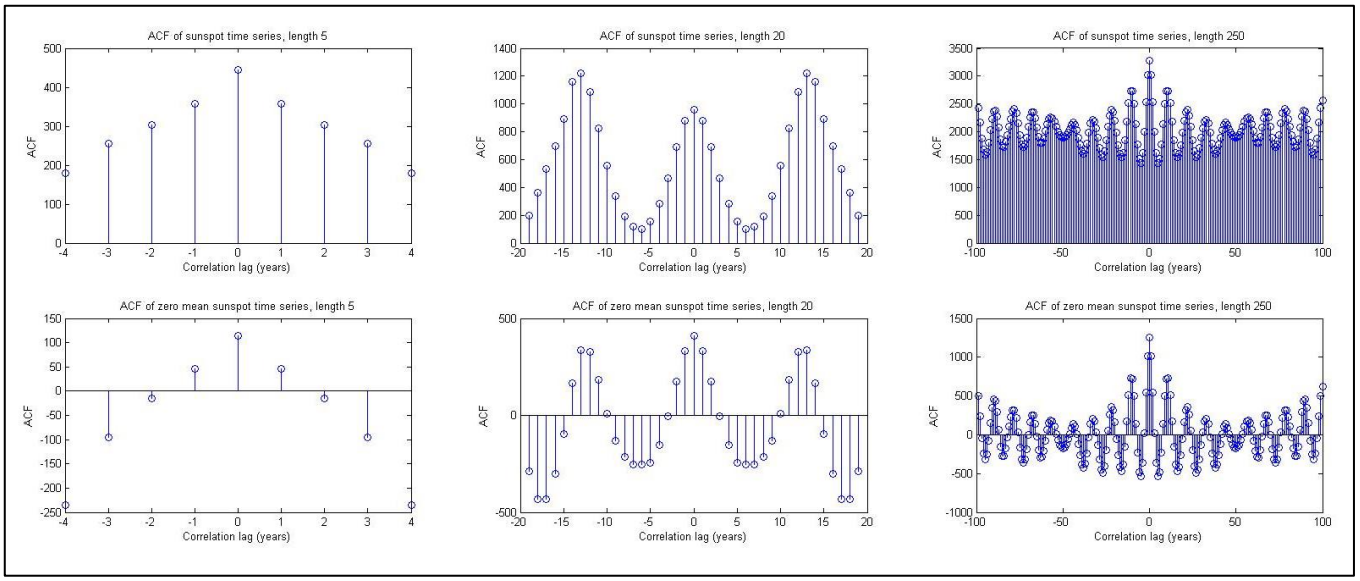

Figure 21: ACF estimate for sunspot data of different lengths. The lower three figures show the ACF estimate for the zero mean data.

The ACF above indicates periodicity with a period of 11 samples. This can also be seen by plotting the sunspot data and averaging the period over a few peaks and toughs as shown in the figure below.

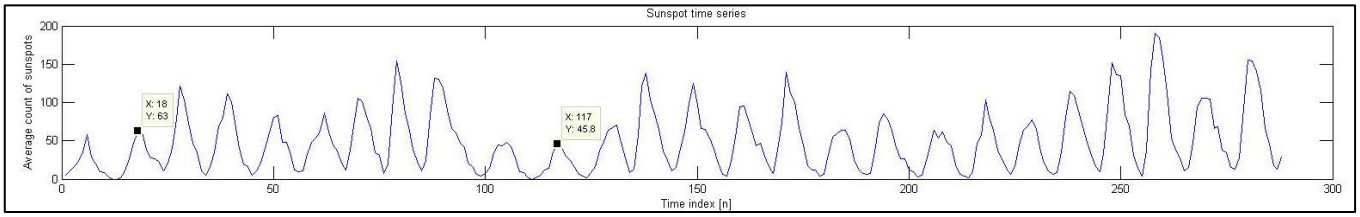

Figure 22: Sunspot time series. Note the sample period is 1 year.

3) PACF from Yule walker equations

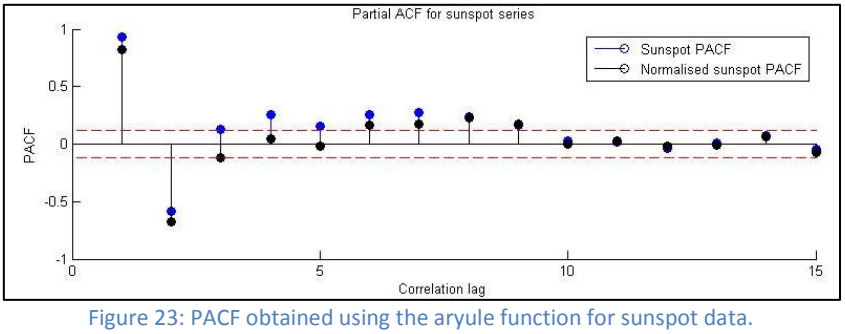

Figure 23: PACF obtained using the aryule function for sunspot data.

Assuming the sunspot time series to be generated by an AR(p) process, then the PACF for lags greater than p follow a normal distribution with zero mean and variance 1/(data length). The red line shows a 95% confidence interval, and for both the normalized and original data, the PACF at lags 1 and 2 exceeds the confidence interval significantly. This implies a model order of 2 to be sufficient. However, some lags below and including lag 9 also exceed the confidence interval slightly. Therefore the order could also be considered to be 9 depending on the trade of between simplicity and model accuracy. For larger lags, the PACF for the normalized and non-normalized data converges to the same.

4) MDL and AIC for practical order selection

In general, the greater the models order the greater the accuracy, (although there are some exceptions for example over modeling can cause spectral line splitting.) In practice there is a tradeoff between computational complexity and model accuracy. The minimum description length criterion (MDL) and Akaike information criterion (AIC) introduce a penalty for higher order and so help determine optimal order selection.

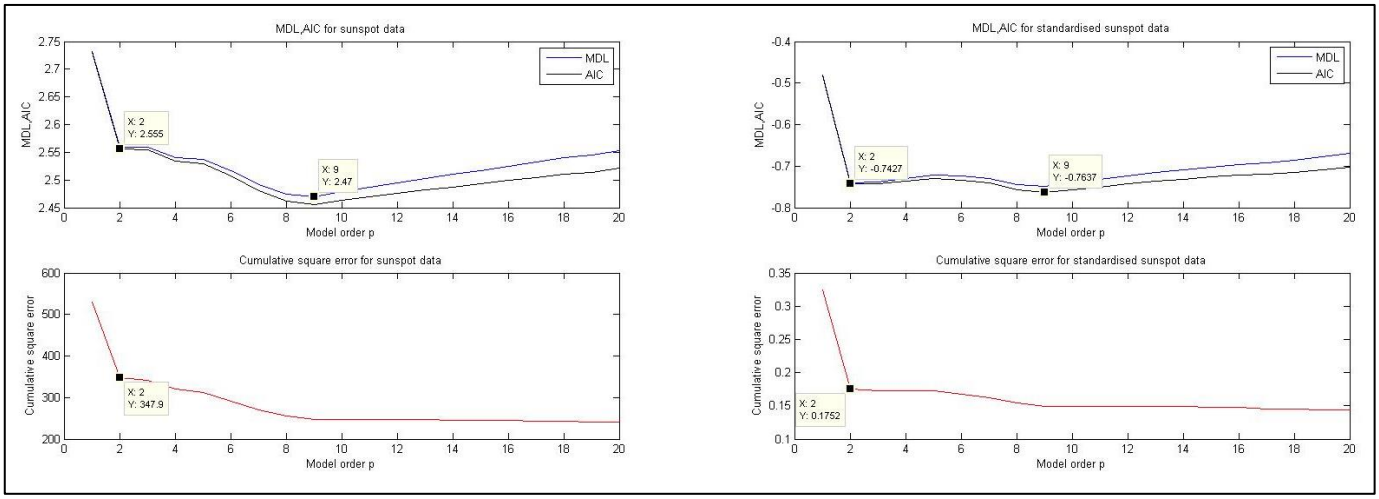

Figure 24: MDL, AIC and cumulative square error for sunspot data and normalised sunspot data

Figure 24 shows the cumulative square error to be a monotonically decreasing function as expected. Note the large difference in the error plot for the normalized and non-normalized data. This is due to the much larger variance of the original data which essentially scales the error (original variance = 1554). Both the MDL and AIC have a smallest minimum at model order 9 for both the sets of data. This implies a model order of 9 is best based on the tradeoff between computational complexity and model accuracy. However the model order could also be considered to be 2 since this is the location of the first minimum for the standardized data, and because this minimum is only 2.7% higher than other.

5) Sunspot forecasting with the AR(p) model:

If we are restricted to using past outputs to predict future outputs, the best linear predictor in a mean squared error sense is given by is given by $\hat{x}[n] = a_1 x[n-1] + a_2 x[n-2] + \cdots + a_p x[n-p]$, where p is the optimal order of the AR process. This is basically the equation describing a $AR(p)$ process with the $w[n]$ term removed. The reason for this is because we are forecasting for the excepted value and $E\{w[n]\} = 0$. Note that this is optimal regardless if the process is AR or not. This formula can be used to forecast an arbitrary number of steps ahead. To forcast more than one step ahead, the previous forecasts must first be estimated, and these are re used in the formula to compute the next step.

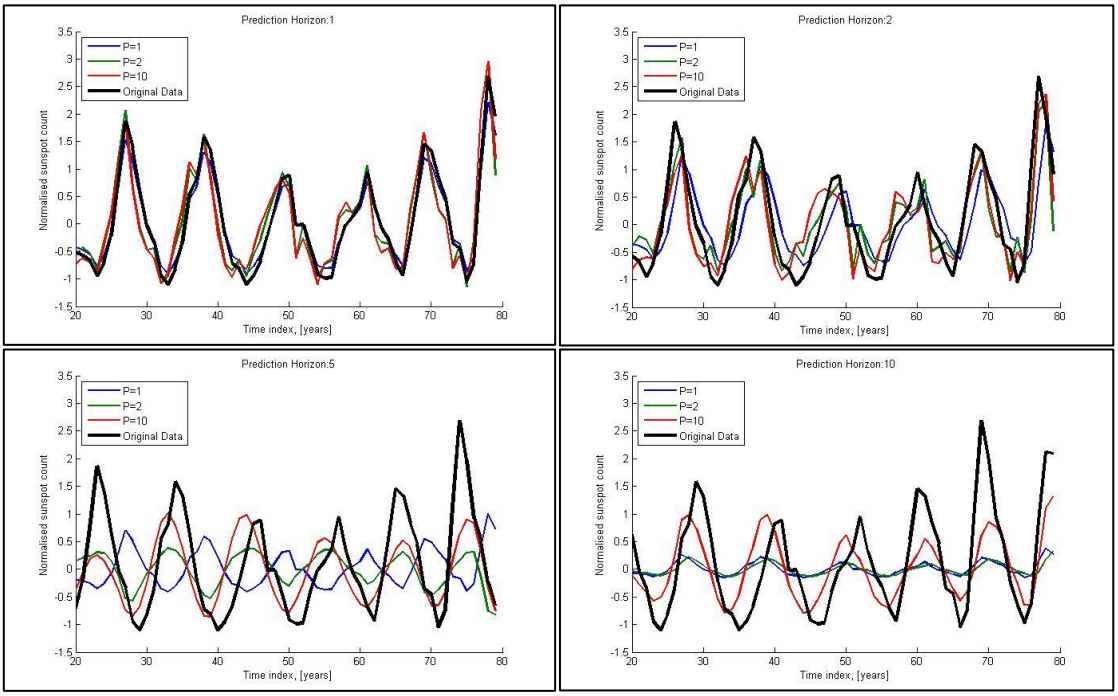

Figure 25: AR based forecasting for normalised sunspot time series for different prediction horizons and with different model orders.

Figure 25 shows the highest model order to give best results, particularly for forecasting further into the future. For a horizon greater than 2, the AR(1) and AR(2) models were inaccurate whereas the AR(10) model still followed the general periodic shape. For forecasting only 1 sample ahead, the AR(1) model seems to be sufficient and in some places better than the higher order models. This suggests that for near future forecasting there is a tradeoff between model error and prediction error. I.e. a higher model order gives a small model error, but for near future forecasting a lower order can be better in terms of prediction error.

2.5 Real world signals: ECG from iAmp experiment

A/B) PDE of heart rate data:

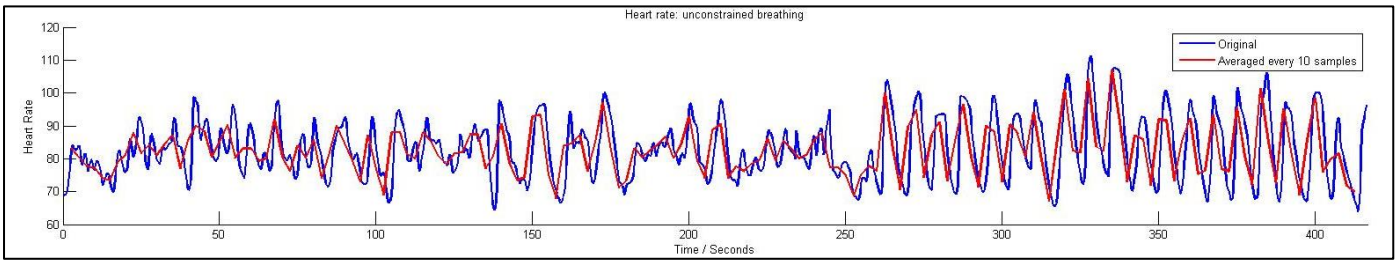

Figure 26: Heart rate data from trial 1: unconstrained breathing.

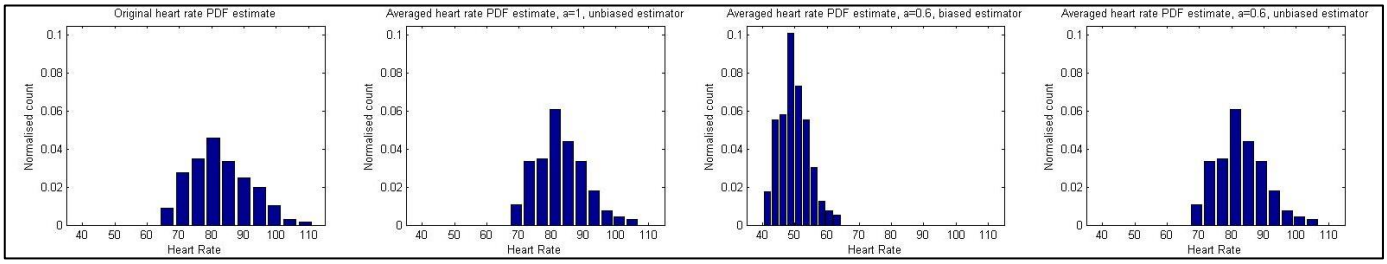

Figure 27: PDE for original and averaged heart rates (BPM). (Mean/variance) for the estimates from left to right: (82.5/82.5), (82.5/57.9), (49.5/20.9), (82.5, 57.8).

Averaging has been performed by $\widehat{h} = (1/10)\sum_{i=1}^{10}(a \times h[i])$, and the effect is to reduce the variance. If the samples were uncorrelated, we would expect the variance to reduce by 10 when the averaging with $a = 1$ is performed. In fact it only reduces by 1.4 and this is because the samples are obviously not uncorrelated. For $a = 0.6$, we would expect the variance to reduce by $10/0.6^2 = 27.8$, but in reality the reduction is by 3.95. From Figure 27 we see that averaging has reduced the spread of the data in all cases. For $a = 0.6$, the mean has also been scaled by 0.6 and the estimator is biased. To correct the bias, the estimator can be divided by 6 rather than 10 and this is shown by the rightmost figure. Mathematically this is the exact same as the estimator with $a = 1$, and dividing by 10 as in the second leftmost figure.

C/D) AR modeling of heart rate:

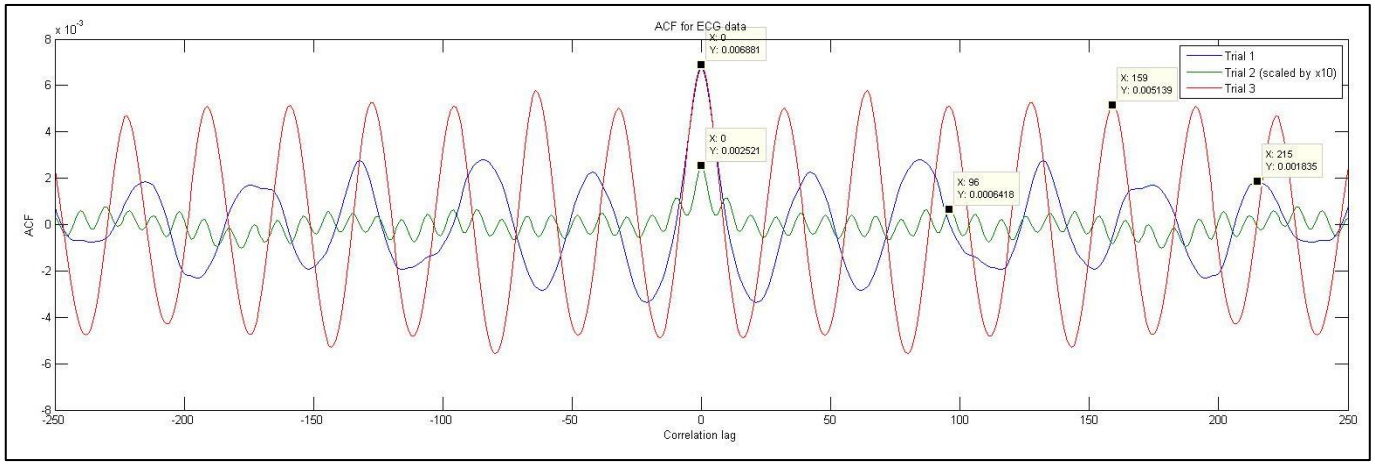

Figure 28: ACF estimate for the three trials. Trial 2 is scaled by 10 to show on same axis.

Figure 28 shows the estimated ACF for the three trials zero mean RRI data. Trial 2 has been obtained from a different heart and so its ACF amplitude differs greatly from the others. The ACF implies the RRI signal to be modeled by an AR model rather than a MA one since it appears to be infinite in duration. Furthermore, taking the FFT gives the PSD which is peaky and hence implies an AR model since a MA model would struggle to model a peaky PSD since it has an all zero transfer function.

The ACF shows trial 1, 2 and 3 to have a period of 43, 9.6, and 31.8 samples respectively. These correspond to a heart rate modulation frequency of 0.09, 0.417, and 0.126 Hz. According to the trail conditions, we expect this frequency modulation to be

0.417 Hz for trial 2 and 0.125 Hz for trial 3, which agrees with the test data to within 0.5%. The data shows the unconstrained breathing rate to be 10.8 beats per minute on average.

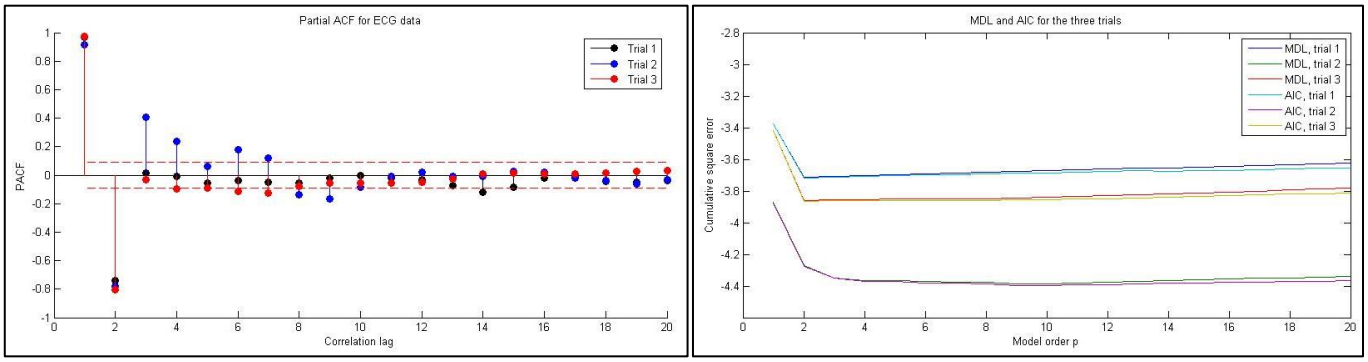

Figure 29: Determining the optimal model order with PACF, MDL and AIC.

The MDL and AIC criterion suggest the trial 1 and 3 to have model order 2. Additionally, for these two trials the PACF is smaller than the 95% confidence interval for lags greater than 2 which also implies a model order of 2. For trial 2 the MDL and AIC have a steep drop until order 2. Beyond this there is a slow drop and a minimum at 9. The PACF also suggest the model order to be 9, however 3 or 4 also seems reasonable. This is because the minimum of the MDL and AIC at 9 is not much different to that at 3 or 4. Additionally, the PACF after lags 4 are much smaller, indicating 4 is sufficient.

3 Spectral estimation and modelling

1) Testing the pgm function

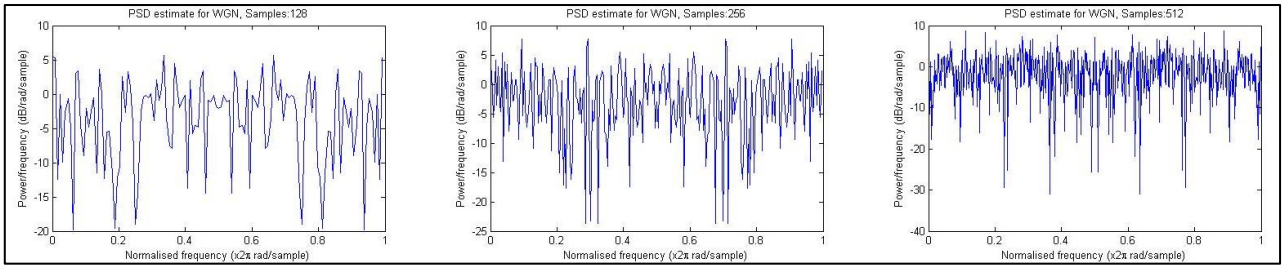

Figure 30: The plotted results of pgm function used with different lengths of WGN.

The PSD for a wide sense stationary process is the ensemble average of all the PSDs of all the sample functions. Hence it tells us about the average distribution of power across all frequencies. By the Wiener–Khinchin theorem, this definition is equivalent to the Fourier transform (FT) of the ACF. For zero mean WGN, since all the samples are independent, the ACF is a discrete Dirac function. The FT of this is a constant value with height the same as the Dirac. Therefore there is equal power in all frequencies and hence the name white noise.

Figure 30 shows the PSD estimate for realizations of WGN for different lengths. This estimate can be improved if more sample functions are used and the PSDs are averaged, and/or if a longer sample function is used (due to ergodicity). The PSD is approximately constant for each realization, and on average MATLAB calculated the average to be 1±10%, which is expected since the Dirac is height 1 since variance of the noise is 1.

The average power is equal to the sum of all frequencies divided by the sample number. (This is the same as the inverse DFT evaluated at zero, and the inverse DFT gives the ACF which gives power if evaluated at zero.) MATLAB computed this to be 1±10% as expected since the input has variance approximately 1 and is approximately zero mean. The differences between the theoretical PSD of WGN and the estimate as in Figure 30 is due to the finite lengths which mean there is some variance in the estimate. However on average this estimate will match the theoretical since it is an unbiased estimator.

3.1 Averaged periodogram estimates

1) Smoothing by zero phase FIR filter of order 4:

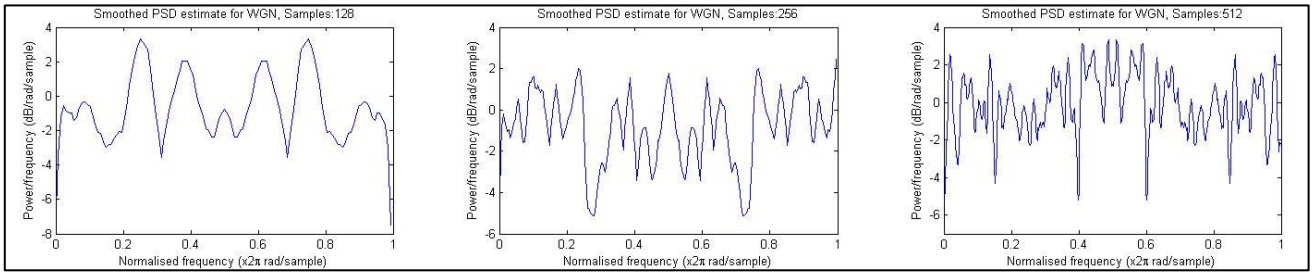

Figure 31: The same PSD as in Figure 30 but smoother by a FIR zero phase filter of order 4.

This improves the apparent PSD estimate since it is closer to the ideal. The fast variations of PSD between frequency bins are smaller due to the averaging, so it is closer to the ideal straight line. A higher order filter would give better results. Note that in the case where the PSD is not supposed to be a smooth curve, filtering in this manor would give inaccurate results.

2/3) Averaged periodogram

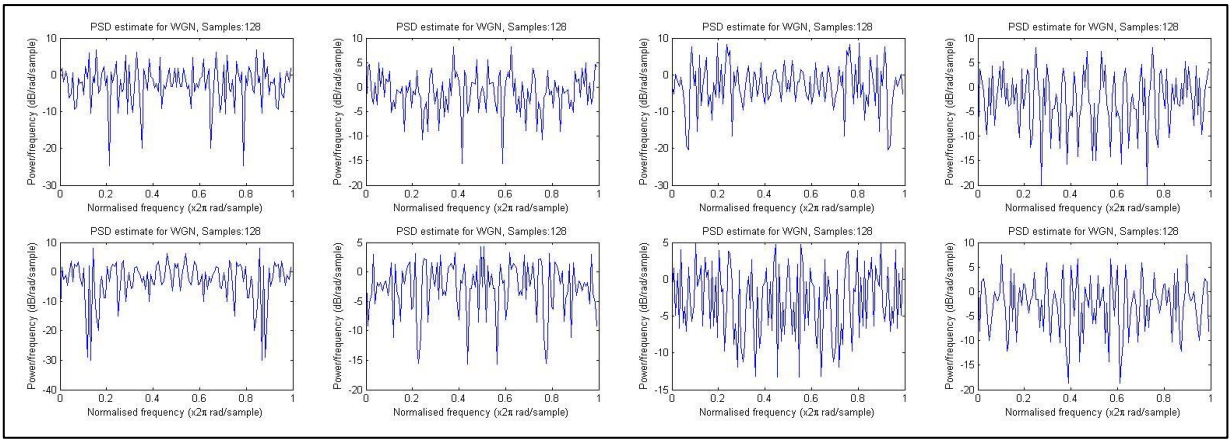

Figure 32: PSD of the 8 segments of the 1024 sample length WGN

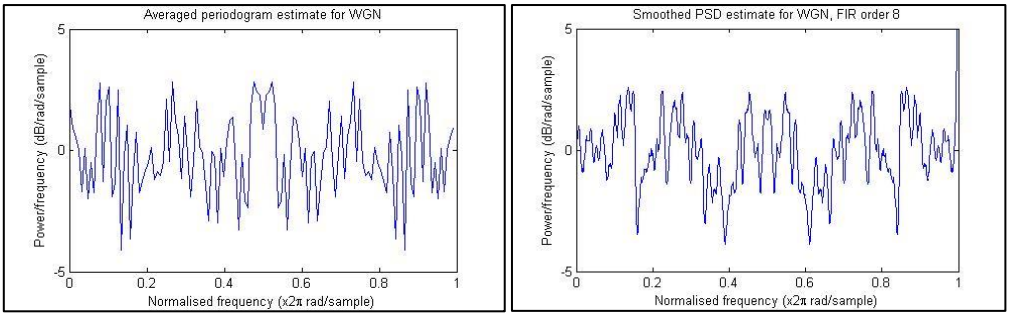

Figure 33: (Left) 8 periodograms of length 128 averaged. (Right) 8*128 length periodogram smoothed by FIR filter of order 8

Figure 33 (left) is the averaged periodogram of the 8 in Figure 32. It has a much smaller spread with the mean 0.163dB/rad/sample and standard deviation -8.965dB/rad/sample. For the separate periodograms the standard deviations varied from -2 to 2dB/rad/sample and were 0.15dB/rad/sample on average. The variance has reduced by around 8 (-9dB) in the averaged periodogram as expected. This is because the PSD at each frequency bin is a RV, and averaging N independent RV's reduces the variance by N. In this case each bin follows a chi-squared distribution since the PSD is a sum of 128 squared (and scaled) independent samples which follow a normal distribution. Figure 33 (right) shows the PSD estimate of the same realization of WGN. This has been smoothed by a FIR filter of order 8 and has mean 0.15dB/rad/sample and standard deviation -9.01dB/rad/sample which is comparable to the left figure. In general it is best to average PSD frames rather than subsequent bins due to the reason mentioned in 3.1.1.

3.2 Spectrum of autoregressive processes

1/2/3) spectral estimate of filter output and windowing:

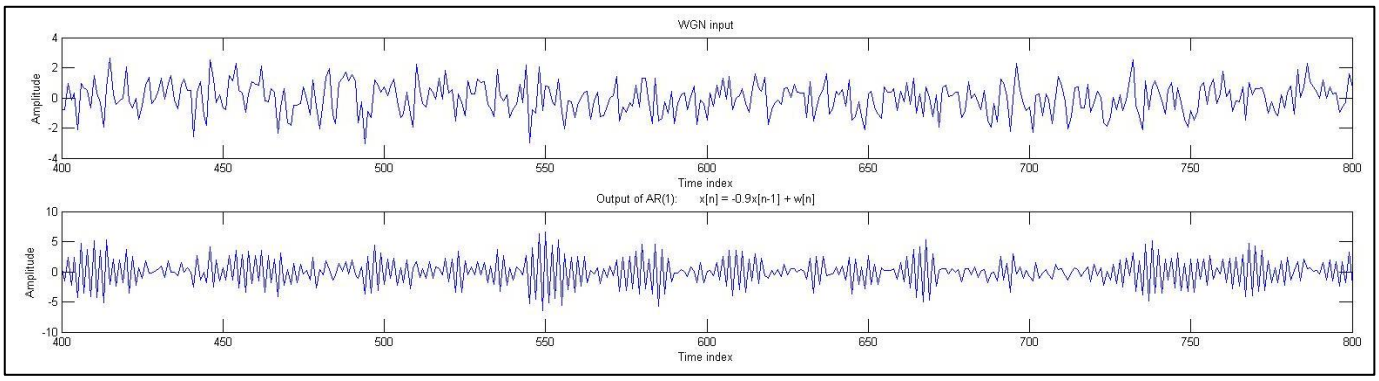

Figure 34: Synthesis of AR(1) process.: (x[n]=-0.9x[n-1]+w[n]).

Figure 34 shows the synthesis of an AR(1) process, with w[n] as WGN with unit variance. This appears to have less of a random frequency variation as compared to the input. It looks mostly composed of a sinewave of a single frequency with randomly varying amplitude. This is expected since an AR(1) filter would amplify one frequency due to its single pole.

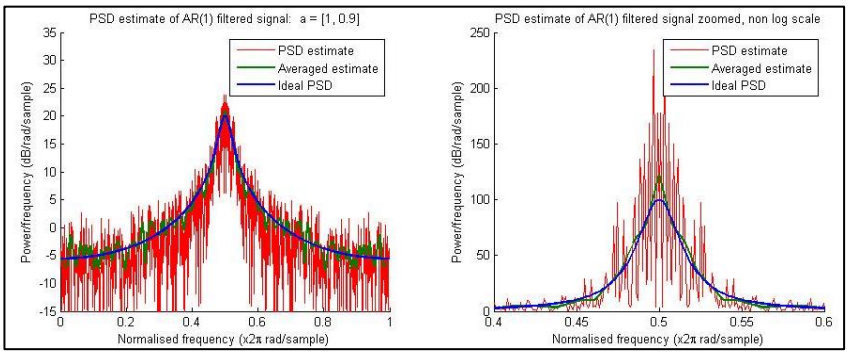

Figure 35: PSD estimate of filter output and exact ideal PSD

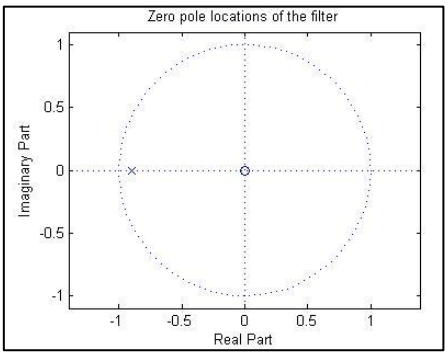

Figure 36: Poles and zeroes of filter

Figure 35 shows the PSD estimate of the output of the filter when the input is WGN of length 1064. The first 40 output samples are not used in the estimate due to the transient effects of the filter. The ideal PSD is shown in blue and the differences between the red estimate and the ideal are mostly due to the variance in the estimator since only one PSD is taken and not the average of many, and because of finite data length. Averaging as explained in 3.1.2 reduces the variance and hence is closer to the ideal PSD. The shape of the ideal PSD can be explained by considering the location of the pole (Figure 36). The pole is close to the unit circle at an angle of π which corresponds to half the sampling rate, or π radians/ sample in normalized frequency. Consequently, at this frequency there will be amplification and hence the PSD has a peak at these π radians/ sample. If the pole were closer to the circle, e.g. if the coefficients were $[1,0.95]$, then the peak would be higher and narrower.

Another source of the difference between the ideal PSD and the estimate is due to windowing effects. By using a finite amount of data, we are essentially multiplying by a rectangular window. This windowing corresponds to convolution with a Sinc function in frequency domain, and therefore the bandwidth is increased by the bandwidth of the window. The bandwidth of a Sinc is technically infinite, but for real-world purposes it is simply proportional to the inverse of the window length. Due to this convolution, there is spectral smearing and leakage. The spectral spread occurs because of the main lobe width, and is equal to the practical bandwidth of the window function. The leakage due to the side lobes can be reduced by windowing with a smooth window; the smoother the window, the faster the decay of the side lobes. A smooth window is one with many continuous derivatives.

4) Estimated model based PSD

Another way to obtain the estimated PSD is to estimate the model, then find the ideal PSD of that estimated model. The coefficients and input noise variance can be estimated using the unbiased ACF estimate. Using the equations (20) and (21) from the coursework handout gives $\hat{a}_1 = 0.9006$, and $\hat{\sigma}_w = 1.0072$. These are within 0.07% of the actual values and these errors reduce when the length of y[n] increases. (These were computed with a sample length of 1000). An alternate method is to use the aryule function which returns $\hat{a}_1 = 0.8998$, and $\hat{\sigma}_w = 1.0156$. Note that the previous method gives these same results if the biased ACF estimate is used. The reason the error reduces with length is because the variance of the ACF estimate reduces with length. Since these are close to the actual values, we expect the estimate PSD to be close to the ideal one. This is confirmed in Figure 37 below.

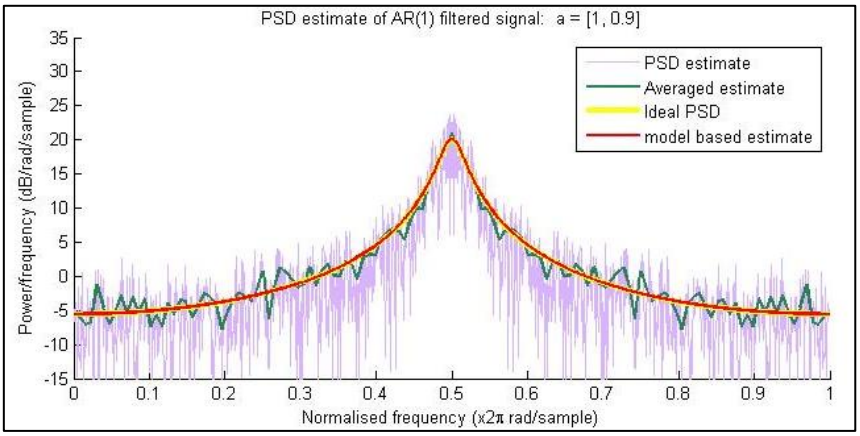

Figure 37: Model based PSD estimate of filter output and ideal PSD.

5) Estimated model based PSD for sunspot series

The aryule function has been used to estimate the model for the sunspot time series and then PSD generated by the freqz function. This is shown in the figures below for both the mean centered and non-mean centered versions of sunspot data. The effect of the mean is to basically increase the power of frequency 0 Hz. The PSD obtained by the pgm function is shown along with the one averaged across shorter frames of length 48. Various model based PSD estimates are also shown to illustrate over and under modeling. With an order too low (p=1), there isn't enough freedom in the spectra since there can only be one peak, therefore the accuracy is low and it differs greatly from the averaged PSD estimate. Over modeling causes too many peaks which may or may not be accurately portraying the system. It is likely to be inaccurate since earlier analysis suggested a model order of 9 was sufficient, so the extra peaks are unnecessary. This spectral splitting continues to increase with higher order.

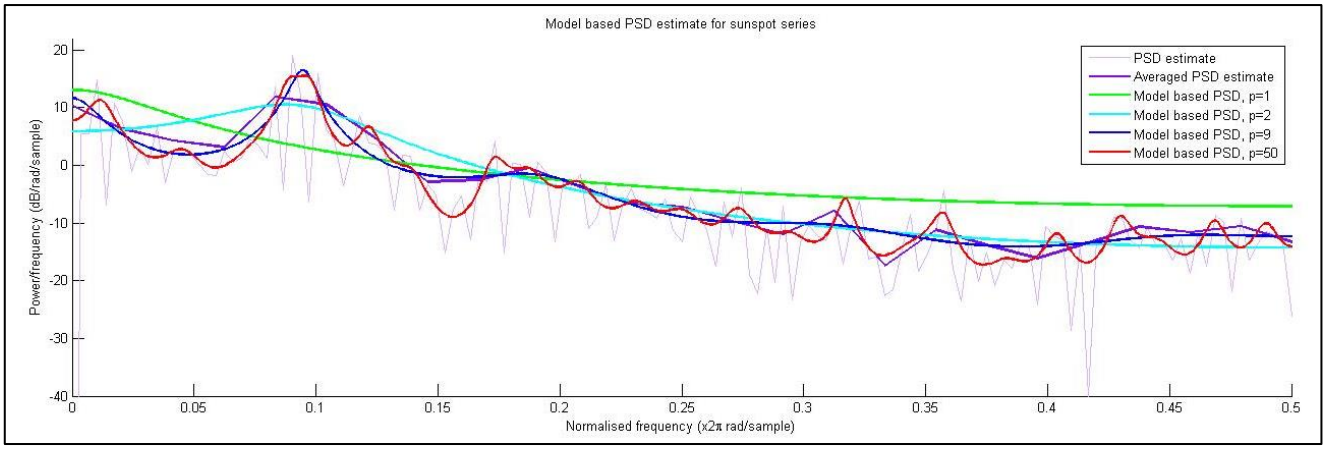

Figure 38: Model based PSD estimate of normalised sunspot series with zero mean and unit variance. Different model orders are shown.

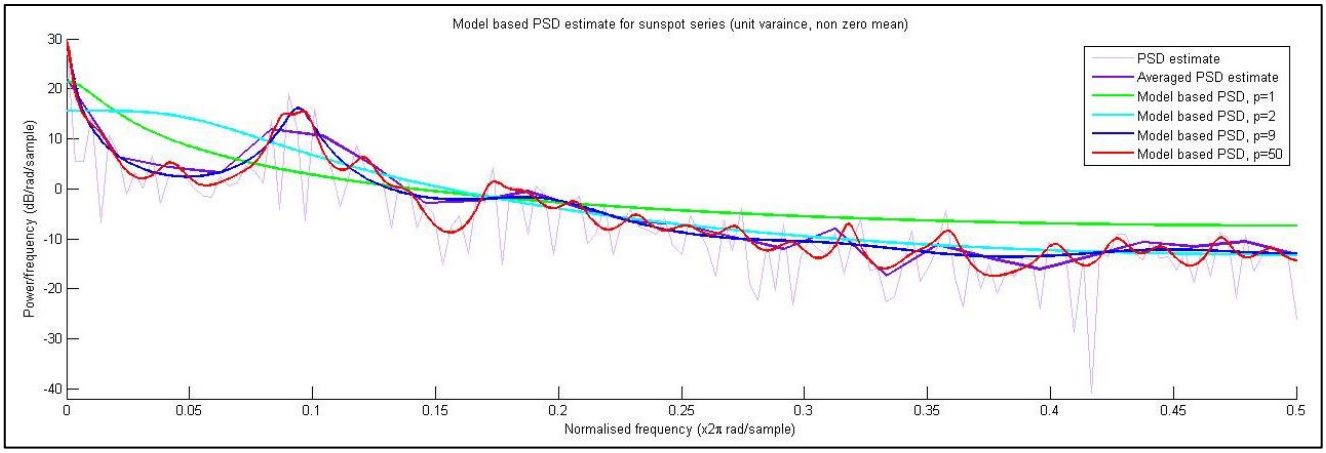

Figure 39: Model based PSD estimate for sunspot series with non-zero mean and unit variance. Different model orders are shown

The coefficients and input noise power could alternately be calculated with the following formulas:

$$\hat{\boldsymbol{a}} = \boldsymbol{R}_{xx}^{-1}\boldsymbol{r}_{xx} \tag{26}$$

$$\hat{\sigma}_w = \sigma_x\left(1 - \rho_1\widehat{a_1} - \rho_2\widehat{a_2} \ldots. - \rho_{\mathrm{p}}\widehat{a_{\mathrm{p}}}\right) \tag{27}$$

Here ρ_i are the normalized correlation coefficients, and p is the model order, and $\boldsymbol{R}_{xx}$ is the ACF matrix of the signal to be modeled and is of dimension p × p.

3.3 Spectrogram for time frequency analysis: dial tone pad

1) Generating a DTMF, (Dual Tone Multi Frequency) time sequence corresponding to a London landline number:

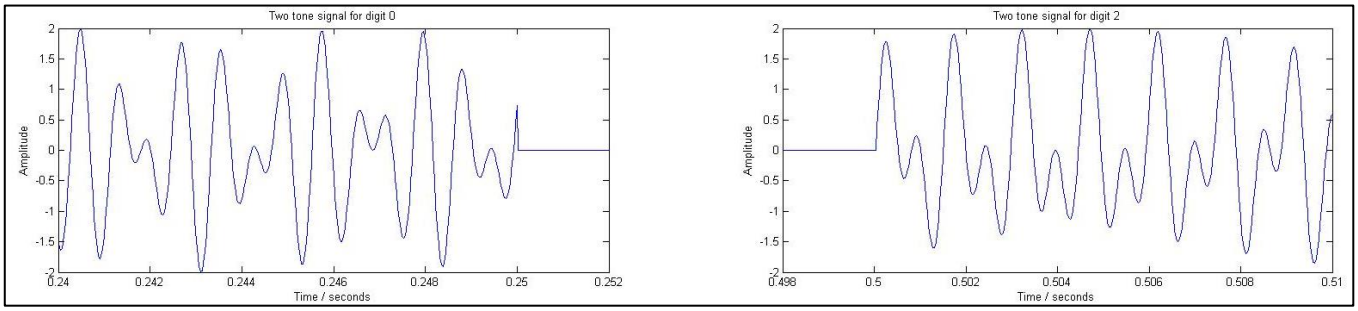

Figure 40: DTMF, system showing 2 digits (tones) and the idle time in-between.

A function DTMF returns the time two tone signal corresponding to the inputted digit, duration of tone, and sample frequency. This has been used to generate a DTMF sequence for a London land line number and two segments corresponding to the digits 0 and 2 are shown in Figure 40. The sampling rate of 32768 Hz is appropriate since it is greater than the nyquist rate (2418 Hz) for the highest sinusoid frequency. In industry, this can be generated very accurately with a correctly shaped quartz crystal and since it is a power of 2, $(32768 = 2^{15})$, it can be divided down to 1 Hz with simple digital electronics which can be useful, particularly for time keeping applications. Finally, more relevant to our case, having a digit pressed for 2.5 seconds means 8192 samples which is a power of 2 which means the efficient FFT can be used over a period of digit pressing.

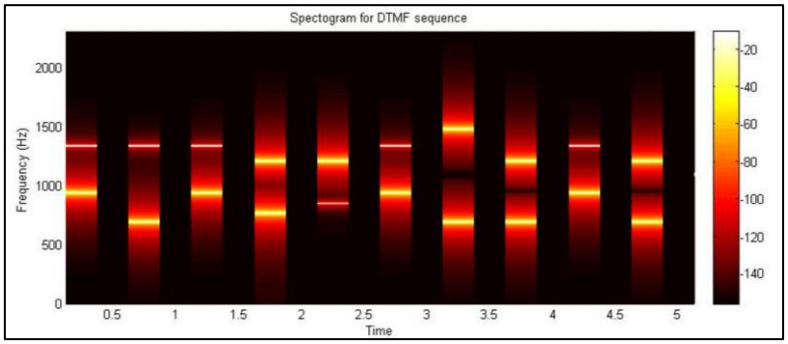

Figure 41: Spectrogram of the DTMF sequence. Colour key shows power in dB.

The spectrogram above clearly shows the two frequencies (high power strips of bright while/yellow color) during periods of key pressing. The spectral power is zero for periods of non-key pressing and is shown as black.

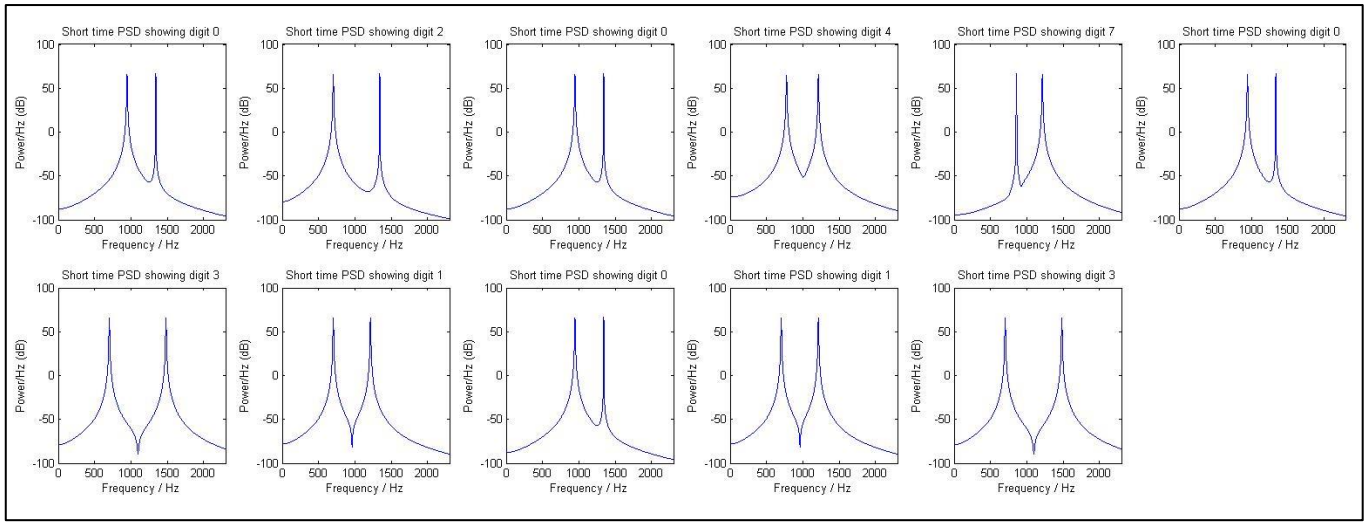

Figure 42: The PSD segments returned by the spectrogram function for the periods where a key is pressed. The sequence of frames is from left to right, top to bottom. The pairs of peaks correspond to the pairs of sinusoids.

3/4) identifying and decoding a noisy signal:

The sequence can be easily decoded back into digits using the spectrum of the segments (Figure 42). The peaks correspond to the frequencies of the two tones and once the two frequencies are determined, they can be looked up from the table (table 1 from c/w handout) to determine the digit. A function called '*decode'* has been written to do this. This first finds the PSD for short windowed frames, and then identifies the two largest peaks within each frame. The peaks are then rounded to the nearest frequency in the lookup table, and the table is used to determine the digit.

The assumption is that the length of the tone duration is known and this is input to the function. This allows one windowed frame to cover one tone and hence simplifies the algorithm. If this information wasn't known beforehand, much shorter frames, (possibly overlapping) could be used and the tone digit decoded from each frame. In this case, the output decoded sequence would be something like this: [1,1,1,1,1,1,x,x,x,x, x,3,3,3,3,3,3,..] where x is a random number and is the functions output during the period of silence between tones. This is because the frames would be shorter than the duration of tone pressing so the same tone would be decoded repeatedly. Then this sequence could be post processed to obtain the sequence [2,3]. The reliability of this method would be less due to the shorter frames and hence less spectral resolution.

The function 'decode' has been tested with a noisy input. It manages to correctly decode a sequence with additive WGN with power up to 75. Note the signal power during tones is 1, so SNR in this case is -18.7dB. The reason we can still decode at even such low SNR is because the energy of the noise is spread over all the frequencies, whereas the energy of the tone is at 2 distinct frequencies, so even if SNR is low, the peaks due to the tone frequencies can be much higher than the noise. This is illustrated in Figure 43 which shows the spectra of the segments for a SNR of -18.7dB. If the noise was perfectly white, and the PSD estimate was perfect, we would be able to decode the signal for any noise power. This is because it is the variance in the 'white' noise spectrum which causes the noise peaks which are confused with the sinewave peaks. Below a SNR of -19.5dB it is no longer possible to hear the tones, or visually distinguish the peaks in the spectra of the segments.

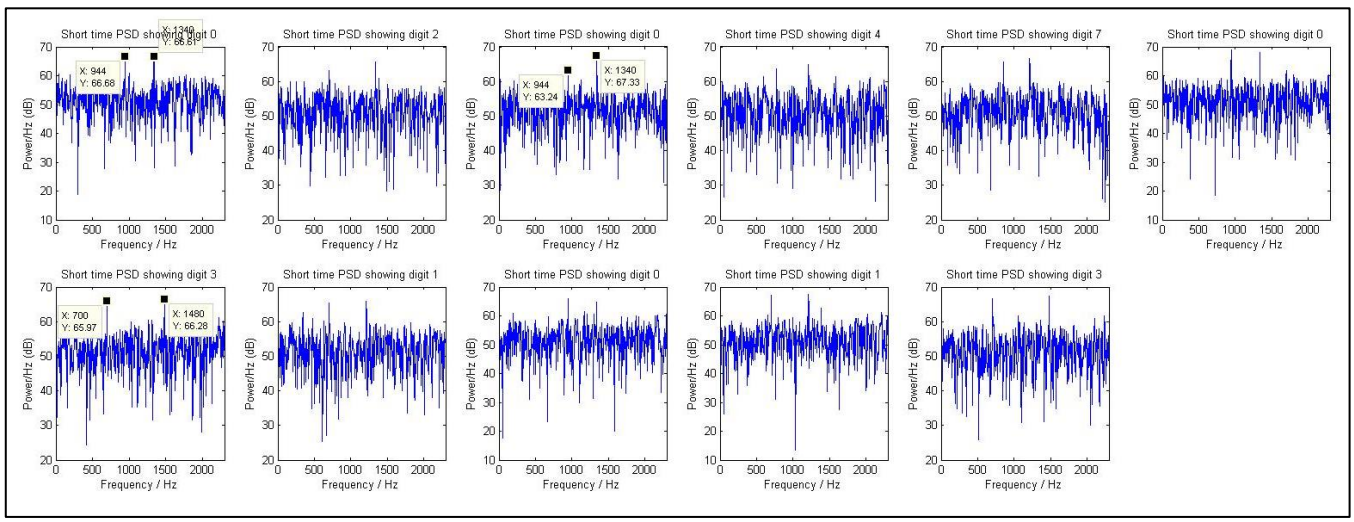

Figure 43: PSD segments in the presence of additive WGN. The tone peaks are still visible even though the SNR is -18.7dB. (Noise variance 75)

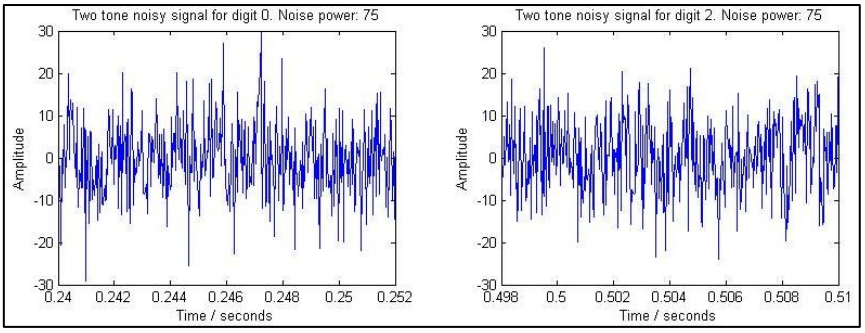

Figure 44: Figure 45: DTMF, system showing 2 digits (tones) and the idle time in-between in the presence of noise. SNR is -18.7dB.

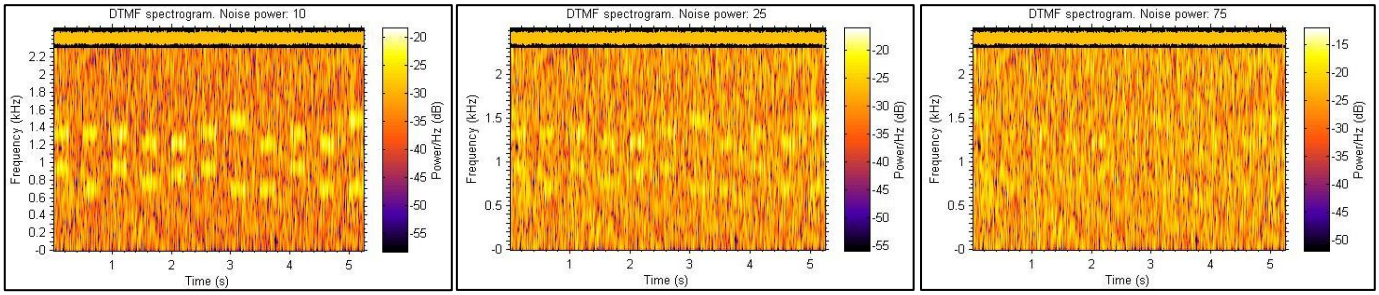

Figure 46: Spectrograms of the same DTMF signal but with different noise powers.

From the spectrograms it is difficult to distinguish the tone frequencies from the noise beyond noise power of 25. This is because the colors are more difficult to interpret than the amplitudes of Figure 43. The table below summarizes the performance of the decoder for various SNR.

Noise power	SNR (dB)	Decoded number. (Target sequence = [0, 2, 0, 4, 7, 0, 3, 1, 0, 1, 3])
75	-18.7	[0, 2, 0, 4, 7, 0, 3, 1, 0, 1, 3]
80	-19	[0, 0, 0, 4, 7, 0, 3, 1, 0, 1, 3]
90	-19.5	[0, 0, 0, 4, 7, 0, 3, 1, 0, 1, 3]
95	-19.8	[0, 0, 0, 4, 7, 0, 3, 1, 0, 1, 3]
100	-20	[0, 0, 0, 0, 7, 0, 3, 1, 0, 1, 3]
150	-21.8	[0, 0, 0,0, 0, 0, 3, 0, 0, 1, 3]

Table 2: showing the performance of the decoder for various SNR.

3.4 Real world signals: Respiratory sinus arrhythmia from RR-intervals

A/B) PSD estimates for RRI data for the three trials:

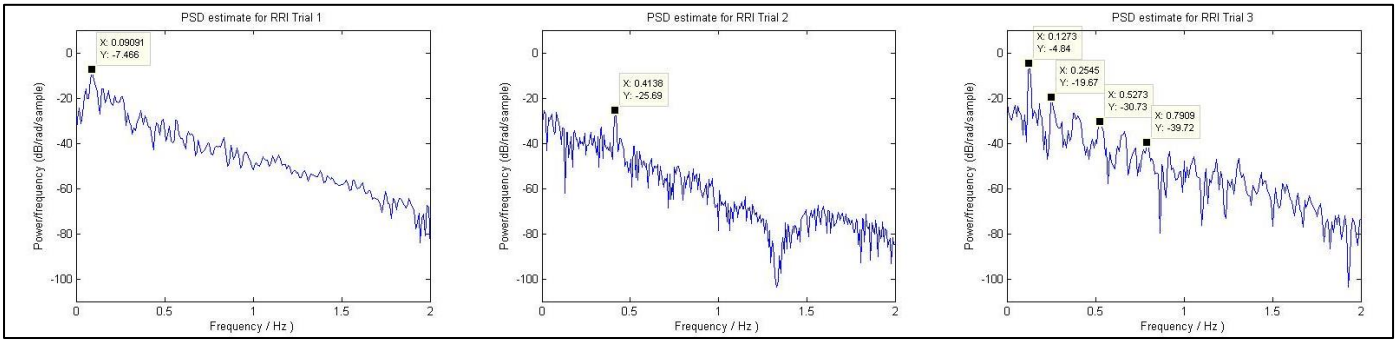

Figure 47: PSD for RRI data for the three trials

The peaks corresponding to the heart rate modulation are clearly visible in Figure 47, and there are close to the expected values of 0.417 Hz for trial 2 and 0.125 Hz for trial 3; the difference is less than 1.6%. The PSD estimate shows the unconstrained breathing rate to be 0.09 which is the same as shown by the ACF in Figure 28. To obtain a better estimate, the PSD has been averaged over shorter consecutive frames. This is illustrated below for frame lengths of 50, 100, and 150. 150 gives the best results since the shorter frames resulted in poor spectral resolution. A Hanning window function was used to help deal with the problems of rectangular windowing. For trial 3 we can see some evidence of the harmonics of the breathing rate. This is most evident in Figure 47 and Figure 48 (bottom). The harmonics measured using Figure 48 (bottom) appear at 0.2666, 0.4000, 0.5334 and 0.6667 Hz ...etc. The harmonics are expected since the modulation of heart rate by the periodic breathing is not a perfect sinusoid shape, (more of a pointy triangular like sinusoid shape), hence by the Fourier series there will be harmonic components.

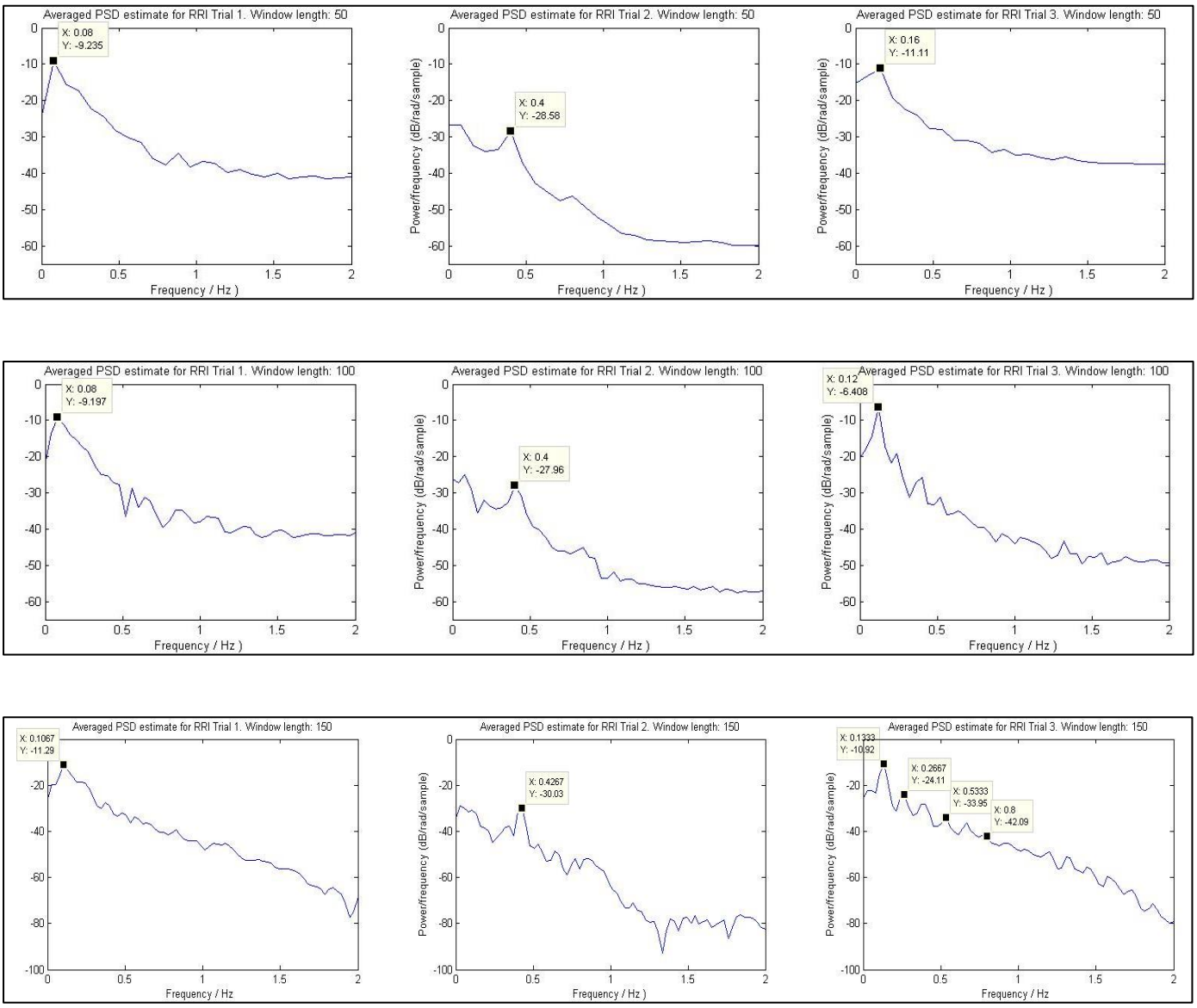

Figure 48: Averaged PSD estimate for the three trials for different window lengths

4 Optimal filtering – fixed and adaptive

4.1 Wiener filter

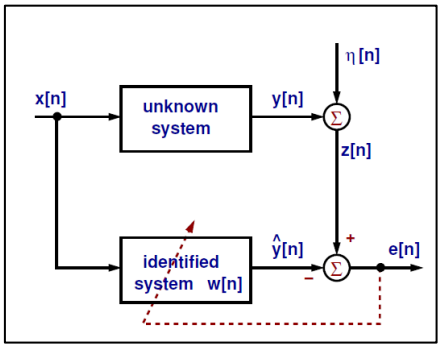

Figure 49: Real world system identification. Source: c/w hand-out pg. 14 figure 4.

Assuming x[n], and η [n] are uncorrelated stationary stochastic processes and the unknown system is MA and the order is the same as the unknown system, the optimal filter coefficients are given by $\boldsymbol{w}_{opt} = \boldsymbol{R}_{xx}^{-1}\boldsymbol{p}_{zx}$. This is optimal in the sense that the mean square error, $MSE = E\{(z[n] - \hat{y}[n])^2\}$ is minimized. This cost function is a quadratic in the error and hence also in the weights. The error surface which is the plot of the cost function against the weights is shaped like a bowl and has a global minimum. This is derived as follows: note that x[n] is the history of the input in the filter.

$$E\{e^2[n]\} = E\{(z[n] - \hat{y}[n])^2\} = E\left\{\left(z[n] - \boldsymbol{w}^{\boldsymbol{T}}{}_{opt}\boldsymbol{x}[n]\right)^2\right\} = E\left\{z^2[n] - 2z[n]\boldsymbol{w}^{\boldsymbol{T}}{}_{opt}\boldsymbol{x}[n] + \boldsymbol{w}^{\boldsymbol{T}}{}_{opt}\boldsymbol{x}[n]\boldsymbol{x}^{\boldsymbol{T}}[n]\boldsymbol{w}_{opt}\right\} \quad (28)$$

$$E\{e^2[n]\} = E\{z^2[n]\} - 2\boldsymbol{w}^{\boldsymbol{T}}{}_{opt}E\{z[n]\boldsymbol{x}[n]\} + \boldsymbol{w}^{\boldsymbol{T}}{}_{opt}E\{\boldsymbol{x}[n]\boldsymbol{x}^{\boldsymbol{T}}[n]\}\boldsymbol{w}_{opt} \quad (29)$$

$$E\{e^2[n]\} = E\{z^2[n]\} - 2\boldsymbol{w}^{\boldsymbol{T}}{}_{opt}\boldsymbol{r}_{zx} + \boldsymbol{w}^{\boldsymbol{T}}{}_{opt}\boldsymbol{R}_{xx}\boldsymbol{w}_{opt} \quad (30)$$

Differentiating the last result with respect to $\boldsymbol{w}_{opt}$ and equating to zero, and then solving the resulting q+1 simultaneous equations, where q Is the order of the filter ($\boldsymbol{w}_{opt}$ is (qx1)) gives the Weiner solution. In the compact matrix form the solution is given by $\boldsymbol{w}_{opt} = \boldsymbol{R}_{xx}^{-1}\boldsymbol{p}_{zx}$. This shows there to be a unique minimum provided the ACF matrix is full rank.

1) Optimal filter coefficients calculation:

With noise power 0.01, the SNR = 20dB. The SNR for zero mean $y[n]$ and $\eta[n]$ is simply $10log10(\sigma_\eta^2/1)$, under the assumption that y is normalized to unit power as requested. For a sample length 1000, $\boldsymbol{w}_{opt}$ is calculated to be $[1.0054, 1.9967, 3.0075, 2.000, 1.0190]^T$. This is after it has been rescaled by the variance of the output $y[n]$ before it was normalized to unit power. The minimum mean square error calculated is 0.01 and is equal to the noise power as expected. This is because the smallest the MSE can ever be is equal to the variance of the additive noise on the output. This can be easily seen in the case that the unknown system and Weiner filter are exactly equal since in the error is then just exactly $\eta[n]$. The percentage difference in the actual unknown system coefficients and optimum filter coefficients is 0.57% on average, and the MSCE, (mean squared coefficient error) is -40.4dB. $MSCE = 10log10\left(\|b - \hat{b}\|^2\right)$

2) SNR and the Weiner solution

The graphs below have been obtained by varying the noise power from 0.1 to 10. A log scale is used on the y-axis to show relative changes in MMSE, (minimum mean square error) and average coefficient error against SNR.

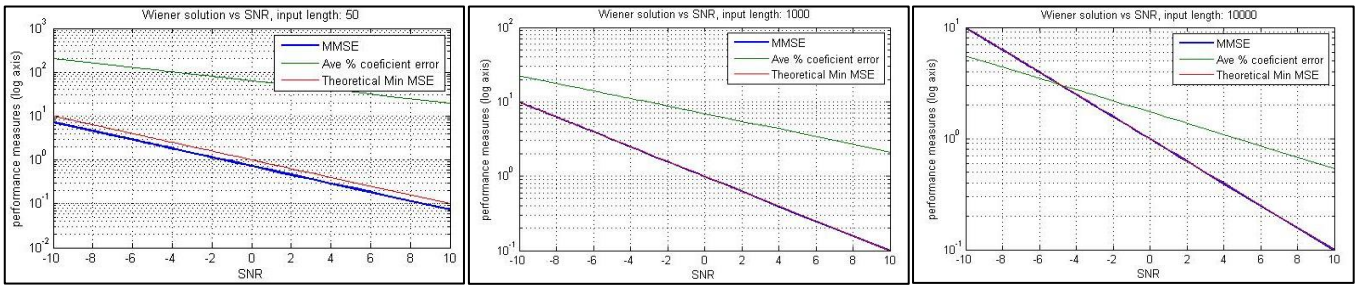

Figure 50: Effects of different noise (SNR) on the Weiner solution for different input lengths.

Figure 50 shows the MMSE reduces with SNR as expected. The theoretical minimum MMSE is given by the variance of the additive noise. The cause of the difference between the theoretical minimum and actual minimum MSE for the figure on the left is due to the autocorrelation and cross-correlation estimates of $\boldsymbol{R}_{xx}$, and $\boldsymbol{r}_{zx}$. For accurate ACF estimates (low variance) much more samples than 50 are needed as discussed in section 2.1. Note that all these estimates assume (jointly) ergodicity WRT the second order statistics. The middle figure shows that 1000 samples are sufficient. Increasing the input sample length further results in negligible difference in MMSE, but the percentage error in the coefficients reduces further.

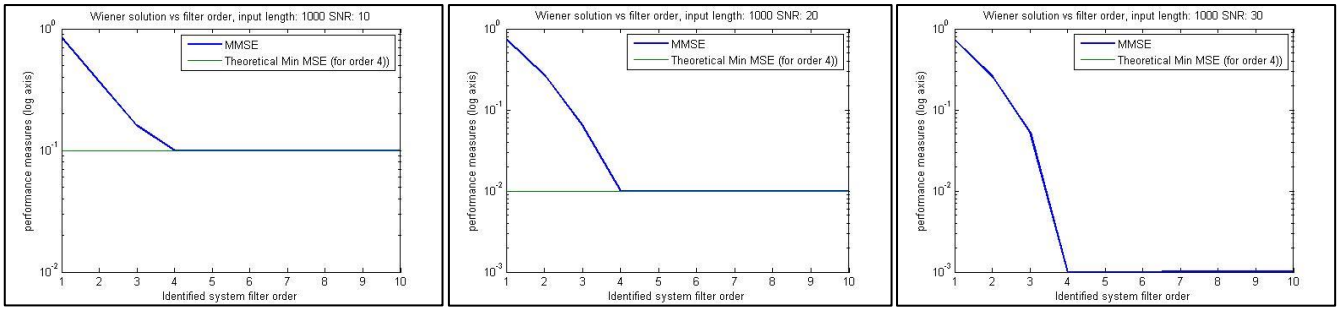

Figure 51: MMSE vs. optimal filter order for different SNR values.

Figure 51 shows the effect of optimal filter order on the Wiener solution. The MMSE is reduced to the expected value of the noise variance only for orders higher than 4. This is expected since a filter with order less than 4 does not have sufficient orders of freedom to replicate output of the unknown system which is order 4. As an example, for order 2, the coefficients were $[0.9304, 2.0835, 3.0249]^T$. At higher orders than 4, there is no noticeable difference in MMSE. In fact the coefficients estimated after the fifth after are 0 ± 0.004 and are negligible. A disadvantage of this over-modelling would be computation time and memory usage.

3) Computational complexity:

The approximate computational complexity of the Weiner solution in terms of additions, multiplications and memory usage is shown in the table below. The order of the optimal filter is q, and N is the length of the WGN input. Since the ACF is symmetric, it is assumed only half is computed and the rest made up from symmetry. This is not the case for the CCF. Finally, the ACF/CCF is only computed up to a '*maxlag*' of q since this is all that is required for the Weiner solution.

Step in solution	Approx. no. of Multiplications	Approx. no. of Additions	Final Memory used
$\boldsymbol{R}_{xx}$ (calculate ACF)	$N \times (q+1)$	$N \times (q+1)$	$(q+1)^2$
$\boldsymbol{r}_{zx}$ (calculate CCF)	$N \times (2(q+1)-1)$	$N \times (2(q+1)-1)$	$q+1$
$\boldsymbol{R^{-1}}_{xx}$ (inverse matrix)	General matrix inverse is $O(N^3)$, but a symmetric toeplitz positive definite matrix can be $O(Nlog^2N)$ or faster to date.		Not stored
$\boldsymbol{R}_{xx}^{-1}\boldsymbol{p}_{zx}$ (multiply matrix)	$(q+1)^2$	$(q+1)(q)$	$q+1$

Table 3: Approximate computational complexity of the Wiener solution. See the note below for an optimisation.

Note that rather than explicitly computing the inverse of the ACF matric then multiplying it with the CCF, we can simply do '`w = Rxx\pzx`' in MATLAB. This produces the solution via Gaussian elimination and doesn't explicitly compute an inverse. This is up to 2 to 3 times as fast as computing $\boldsymbol{R}_{xx}^{-1}\boldsymbol{p}_{zx}$.

4.2 The least mean square (LMS) algorithm

The Weiner solution is computationally intensive and assumes statistical stationary. One solution is to approximate the Weiner solution in a recursive fashion. If the weights are allowed to vary with time, they can be adjusted in an iterative way along the error surface such that the error is minimized. This has the advantage that the solution can track a moving unknown system provided it converges relatively fast enough. The direction of movement is opposite to the gradient vector, i.e. in the direction of steepest descent, and the step size which is controlled by μ determines stability. Note that this method is exact in the sense that no assumptions are made, and if the step is chosen properly, the result can be the same as the Wiener filter in 4.1.

1/2) Testing the '*lms*' function with different adaptation gains:

Figure 52 shows the time evolution of MSE for different step sizes averaged over 1000 trials. The greater the step the smaller the time required to reach the minimum error. For the Weiner filter, the minimum possible error is -20dB (since the noise power is 0.01), and if the step is small enough, the lms algorithm converges to this. This is shown by the blue and red plots in Figure 52 (left). For larger μ, e.g. 0.1, the final MSE is -18.9db which is greater. This occurs because the step is too large and there are over corrections occurring at each step, so the weights jump around the minimum of the error surface. Theoretically, the excess error in excess of the MSE is given by:

$$\frac{\mu \times MSE}{2}\sum_{i=0}^{p}\lambda_i\,, \qquad where\ \lambda_i\ are\ the\ (p+1)\ eigenvalues\ of\ R_{xx} \tag{31}$$

According to this formula, the final error with $\mu = 0.1$ should be $10\log10(0.01 + 0.0025) = -18.9\text{dB}$, which is confirmed by Figure 52. The requirement for convergence in the MSE sense is that $0 < \mu < 2/\text{tr}[\text{R}_{\text{xx}}]$, so $0 < \mu < 0.4$. This stability requirement is illustrated in Figure 52 (right), which shows the plots for $\mu = 0.4$ and $\mu = 0.5$ to diverge.

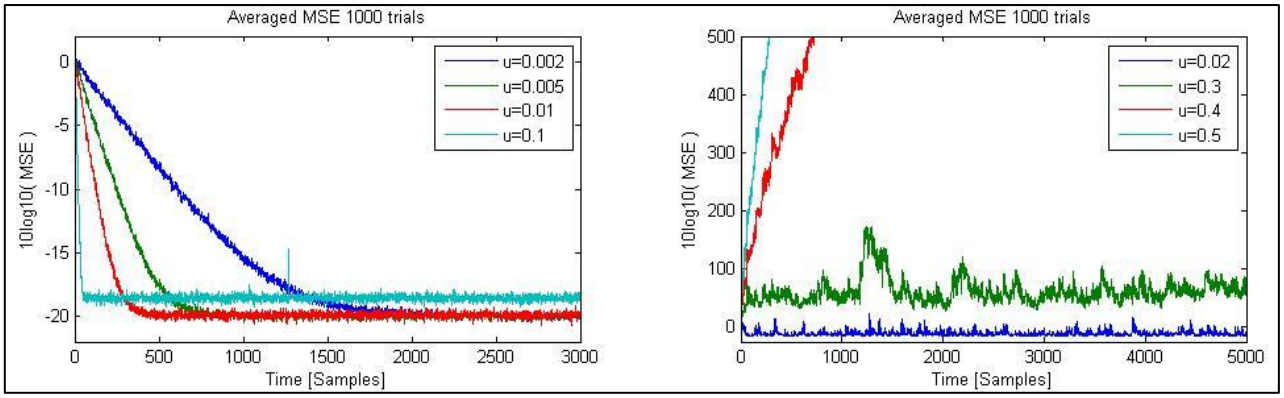

Figure 52: MSE averaged over 1000 trials for different adaptation gains. Two graphs shown due to different y-axis scale

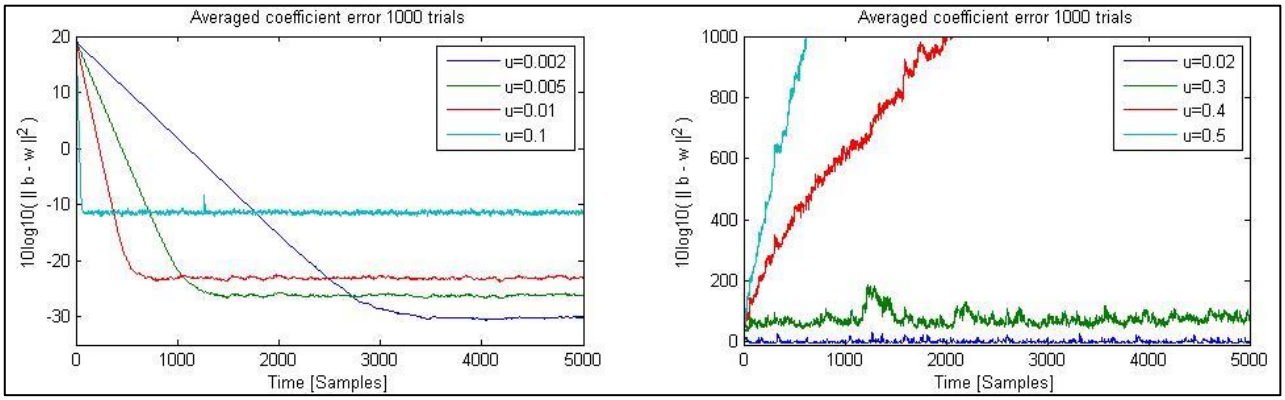

Figure 53: MSCE averaged over 1000 trials for different adaptation gains. Two graphs shown due to different y-axis scale

Figure 53 shows the MSCE averaged over 1000 trials. A smaller step results in a smaller MSCE, but it takes longer to reach the final value. For comparison, for the Wiener filter with input of length 1000, the MSCE was -40dB. Figure 54 shows the evolution of the coefficients for different step sizes. In summary, a smaller step size means the coefficients take longer to settle, but are more stable once settled. Too large a step and the coefficients don't converge ($\mu < 0.4$ for convergence).

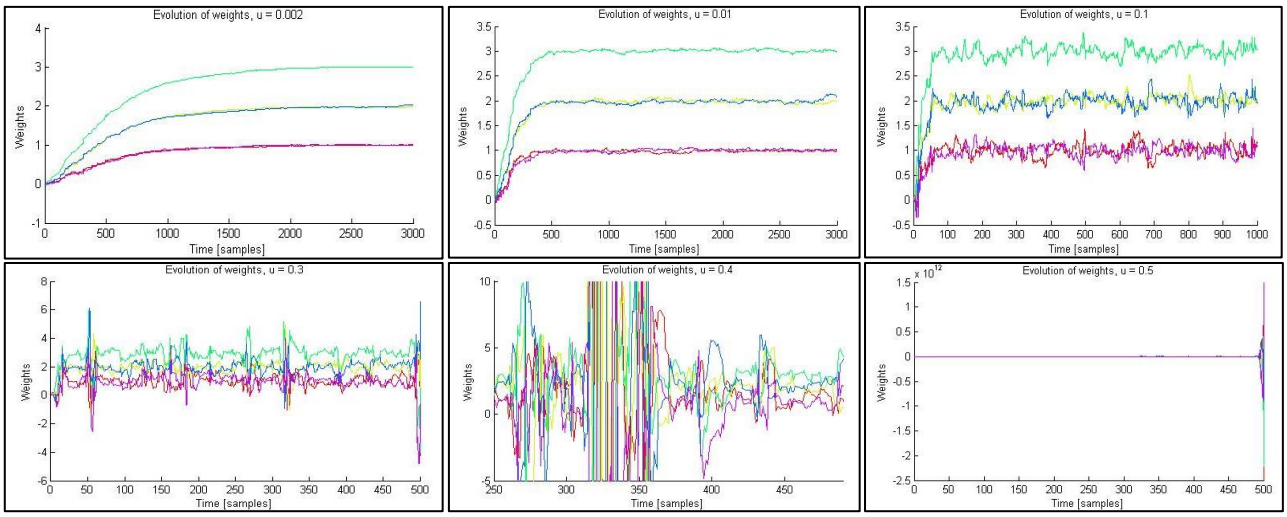

Figure 54: Evolution of the estimated weights for different adaptation gains

Because the instantaneous time statistics are used to estimate the ACF the trajectory of the weights along the error surface is not smooth, and μ must be small enough to ensure these jumps are small so that after many small steps it converges to the minimum (on average the steps are in the correct direction) without constantly overshooting or moving in incorrect directions. This explains why the lines are not smooth for large adaptation gain, e.g. $\mu = 0.1$. The reason the MCE and MSCE graphs are averaged over 1000 sample functions is so the variance of the measurement noise doesn't make the plots too noisy such that the trends can be seen and compared.

3) Computational complexity of the lms algorithm:

The approximate computational complexity of the '*lms*' function in terms of additions, multiplications and memory usage is shown in the table below. The order of the optimal filter is q, and N is the length of the WGN input.

Step in solution		Approx. no. of Multiplications in total	Approx. no. of Additions in total
$y[n]$	(estimate output)	$(q+1) \times N$	$q \times N$
$e[n]$	(compute error)	0	N
$W[n+1]$	(update weights)	$(q+1) \times (2N)$	$(q+1) \times (N)$

Table 4: Approximate computational complexity of the LMS algorithm

The main advantage of this is that there is no matrix inverse computed which is significant for high order estimation as explained in 4.1.3. The memory usage is also much less since no $\boldsymbol{R}_{xx}$ matrix or its inverse needs to be stored, and the $\boldsymbol{y}$, $\boldsymbol{e}$ and $\boldsymbol{W}$ can be overwritten each iteration if their history is not required.

4.3 Gear shifting

As explained in 4.2.2, different adaptation gains have different advantages. High gain means a fast rise time and low gain means good steady state accuracy. If the gain were to change with time, we could benefit from both these advantages. Gear shifting can be used to change the gain according to the error. The gain should be large for large error so the rise time is short. For a small error, the gain should be small which would ensure good stead state accuracy.

Two methods of gear shifting were devised and the normalized LMS algorithm is used to compare the results. The table below compares approximate rise and settle time for the three examples in Figure 55. Note that these results change with realization of the WGN and so are only a rough guideline on performance. The '*gear2*' algorithm has some overshoot, but this is less than the required 20% of the true value.

Algorithm	**Rise time (10%-90%) / samples**	**Settle time (5%) / samples**
NLMS, $\mu = 1$	40	(does not settle within 5% range)
NLMS, $\mu = 0.5$	50	70
NLMS gear 1 $\mu = 1$	20	40
NLMS gear 2, $\mu = 1$	<8	55

Table 5: Approximate rise and settle times for the three different algorithms

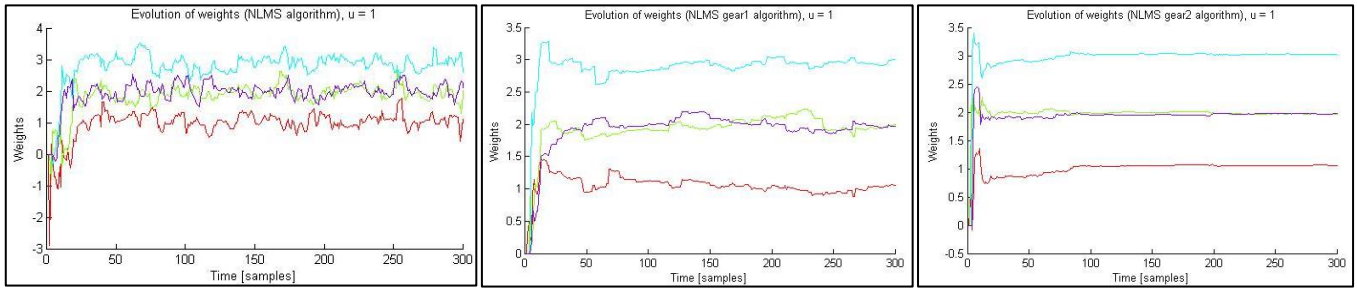

Figure 55: Evolution of weights for two different methods of gear shifting, and the NLMS method for comparison

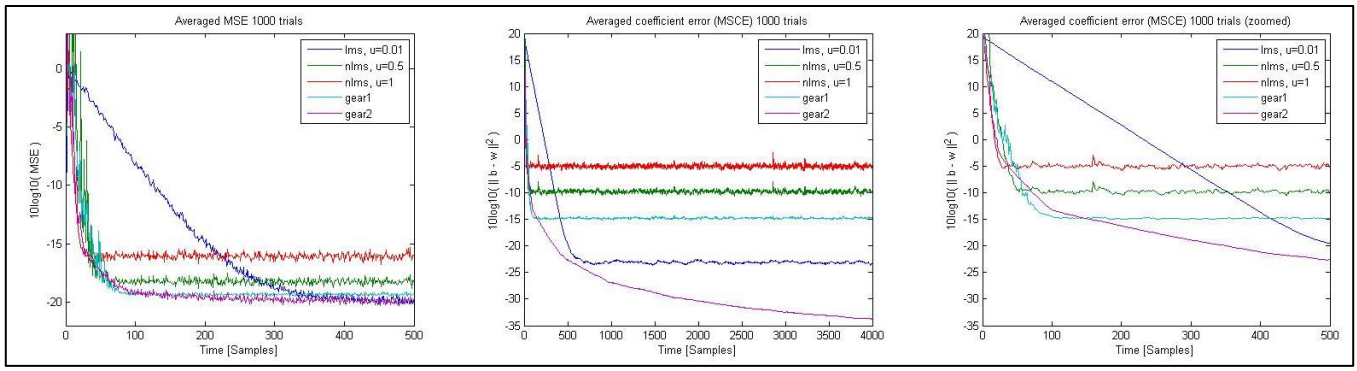

Figure 56: MSE and MSCE for two different methods of gear shifting, and the NLMS, LMS methods for comparison

Figure 56 shows the MSE and MSCE for the different algorithms implemented. The '*gear2*' settles with the smallest MSE and MSCE, and is the second fastest for its MSE to drop below -15dB, just slightly slower than the NLMS with $\mu = 1$.

Explanation of NLMS algorithm:

The NLMS method simply normalizes the adaptation error to the power of the signal in the memory of the filter. This essentially ensures that the step is not too high in the case when the signal power in the filter memory is too high, and hence increases stability. For minimal error, $\mu[n] = 1/\ \|x[n]\|^2$, but in practice we use $\mu[n] = \mu/(\ \|x[n]\|^2 + \varepsilon)$ where ε is a small constant used to prevent division by zero errors, and $\mu \in (0,2)$ for convergence.

Explanation of gear1 algorithm:
This algorithm builds on the NLMS, except now the step is proportional to the absolute error. A cap on the maximum step is placed to ensure stability, and since μ is normalized; the cap is chosen to be 1.5. (2 is the limit of instability).

Explanation of gear2 algorithm:
This algorithm ensures the step tends to zero when the error squared tends to the minimum possible mean squared error which means improved steady state accuracy; the step is made proportional to the absolute value of the error squared minus the estimated minimum MSE. If the estimate is not known, it is set to zero and then the performance is similar to the previously described algorithm.

Additionally, this algorithm also has distinct 'gears'; the error squared is compared to a scaled version of the estimated MMSE and if it is smaller, the step is scaled further by another fraction. This is repeated 4 times so there are 4 distinct 'gears'. Essentially, this ensures a smaller step for a smaller error. In order to reliably compare the current error with the estimated MMSE, the error needs to first be time smoothed since it has a large variance. This is implemented by a first order recursive filter as follows:

$$e_{smooth} = (1-k) \times e[n] + k \times e_{smooth} \qquad where \quad k = e^{-1/(\tau f_s)}, \quad and\ \tau\ is\ the\ time\ constant \tag{32}$$

K = 0.93 is sufficient, and the time constant associated with this is 13.7 samples, or 0.3ms for a sample rate of 44.1 KHz. A larger K of 0.95 gave better steady state results, but 0.93 was chosen as a compromise since less error smoothing means a faster reaction time. This is important for a non-stationary process.

4.4 Identification of AR processes

1/2) linear adaptive predictor tested with different adaptation gains:

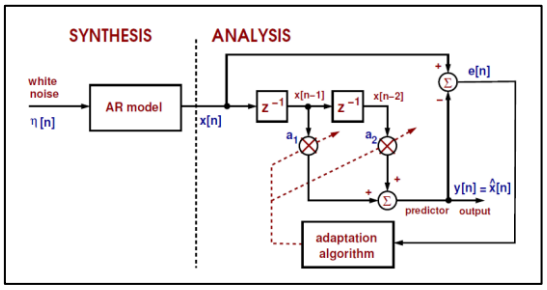

Figure 57: Synthesis and analysis structure for AR(2) model. Source: c/w hand-out pg. 16 figure 5

Figure 57 shows a linear adaptive predictor. The error is minimized in the mean square sense in the same way as before which means the coefficients will converge to a value such to minimize the MSE. Since the AR synthesis model has $a1 = -0.9$ and $2 = -0.2$, $(or\ \ a = [1, 0.9, 0.2]\ in\ MATLAB\ notation)$, $a1$ will converge to -0.9 and $a2$ to -0.2. This is illustrated in Figure 58 which shows the evolution of these weights for different adaptation gains. For the case of ideal convergence, the error is exactly $\eta[n]$. This means the minimum possible MSE is just the variance of the input WGN, $\eta[n]$. This is called a predictor of order 2 since it uses the past 2 outputs to predict the next output. It is called adaptive because unlike the model used to predict the sunspot time series this one uses the error to re adjust the weights with time.

Figure 58 shows the evolution of weights over time. If the adaptation gain is chosen correctly, the weights converge to the ideal values. There is a lot more variance in the weights for similar adaptation gains compared to Figure 54. This is because WGN input which generates the AR process has variance 1 which means even if the coefficients are ideal, the error will still have variance 1. Compared to Figure 54, if the coefficients are ideal, the error will be equal to the measurement noise which was set to have a variance of 0.01, hence the smoother evolution of the weights.

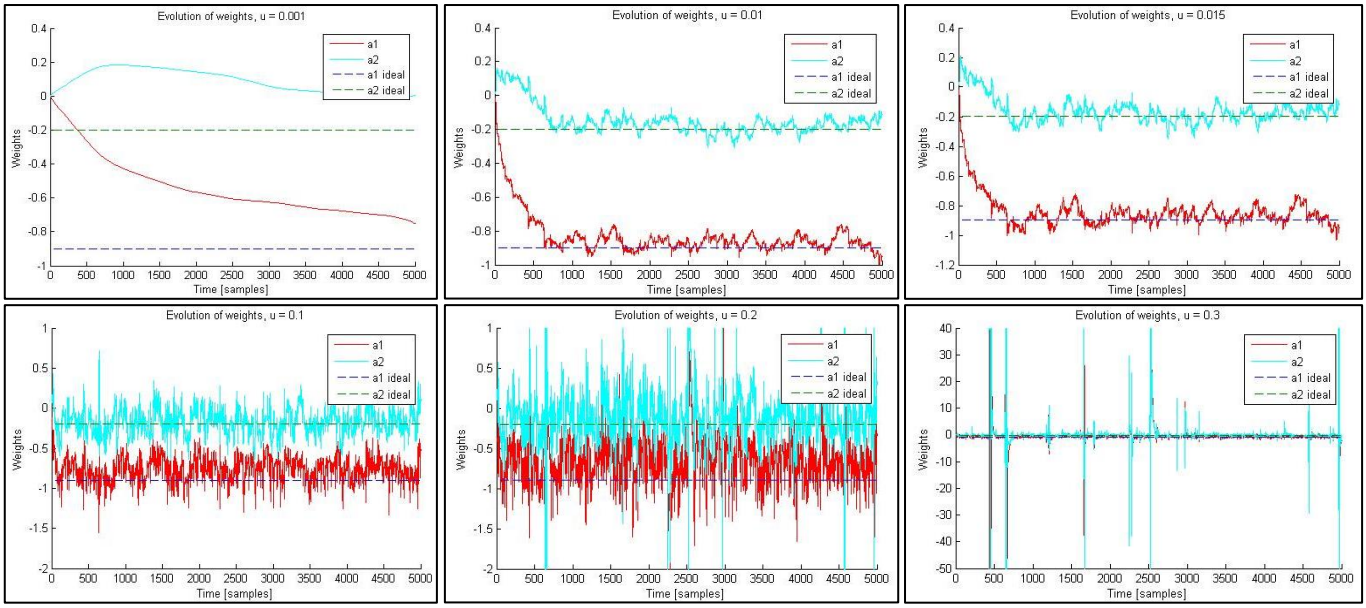

Figure 58: Evolution of weights for different adaptation gains.

Figure 59 below shows the MCE and MSCE for different adaptation gains. Beyond a gain of 0.2, the MCE and MSE are out of the scale and hence not included in the figures. For small enough adaptation gain, 0.01 and 0.001, the MSE converges to 0dB as expected. (The variance of $\eta[n]$ is 1 = 0dB) However if the gain is too small, it is slow to reach its final value. The figure on the right shows that the smaller the gain the smaller the final MSCE which is expected and the same result as the previous adaptive filters.

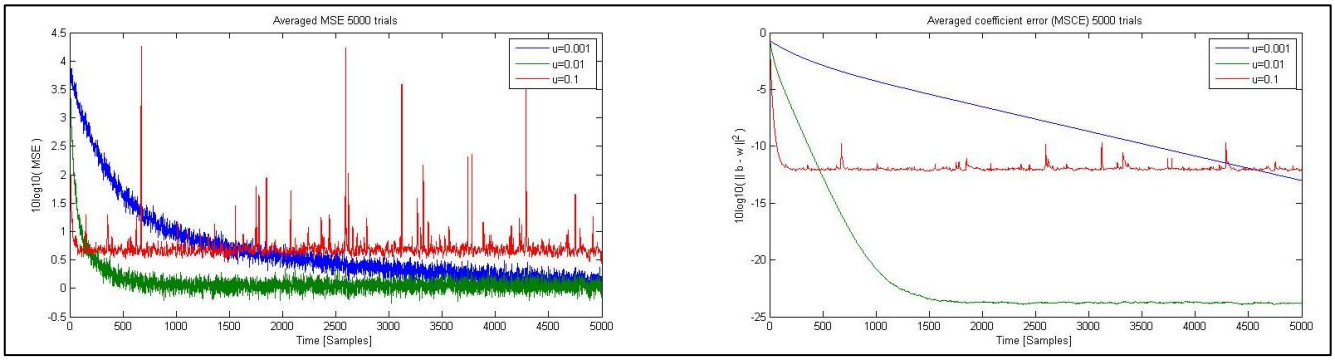

Figure 59: MSE and MSCE for different adaptation gains.

4.5 Speech recognition

1) Performance of the linear predictor:

The figures below show the evolution of weights, the predicted output along with the actual signal, and the prediction error for two example speech sounds ‘a’, and ‘s’. The NLMS algorithm has been implemented to ensure stability and robust performance since the amplitude of the sounds are different and they vary with time. The order 3 was considered here for a starting point since it seemed to give the least MSE under initial investigations (3e-5 and 6e-5 for ‘a’ and ‘s’ respectively). The optimal order is discussed more in the next section.

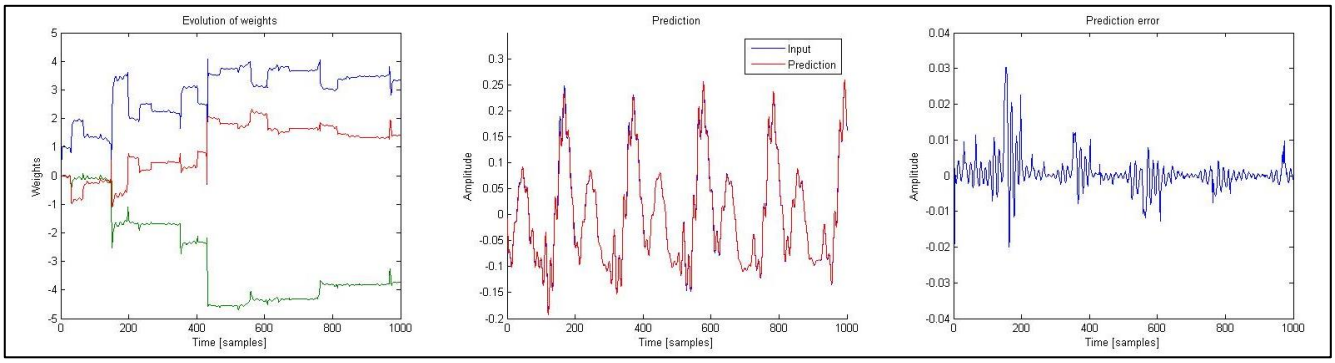

Figure 60: evolution of weights, prediction compared with input, and prediction error for sound ‘a’

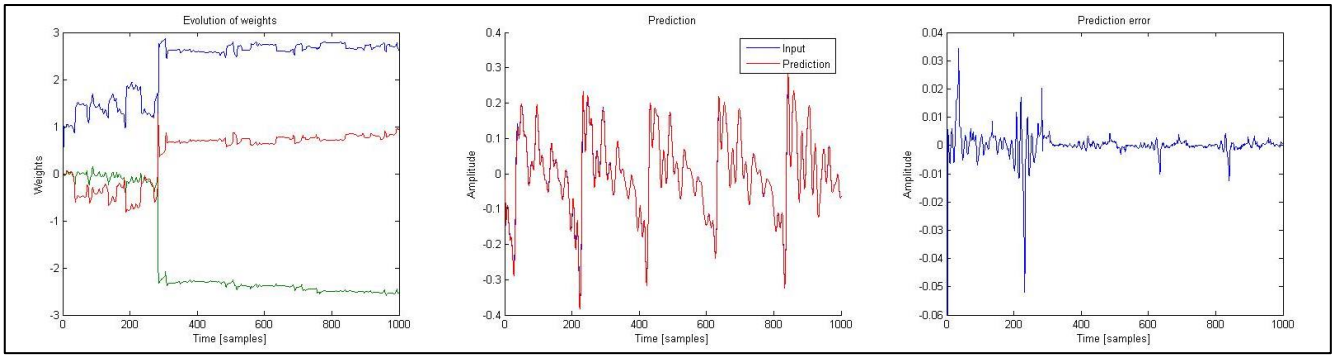

Figure 61: Evolution of weights, prediction with input, and prediction error for sound 's'

2/3) Optimal order selection and assessment of performance:

The optimal order could be selected by using the MDL and AIC or PACF tools applied to a short relatively stationary segment of the speech sound, eg 1000 samples at 44.1 KHz. A short segment is required since to model since the whole non stationary signal would require a much more complex (higher order) model which may not be as accurate when used in an adaptive manor. As an example the MDL and AIC are evaluated for two example sounds and the results shown below. The optimal order is typically in the 10 to 15 range for all recorded sounds according to the MDL and AIC.

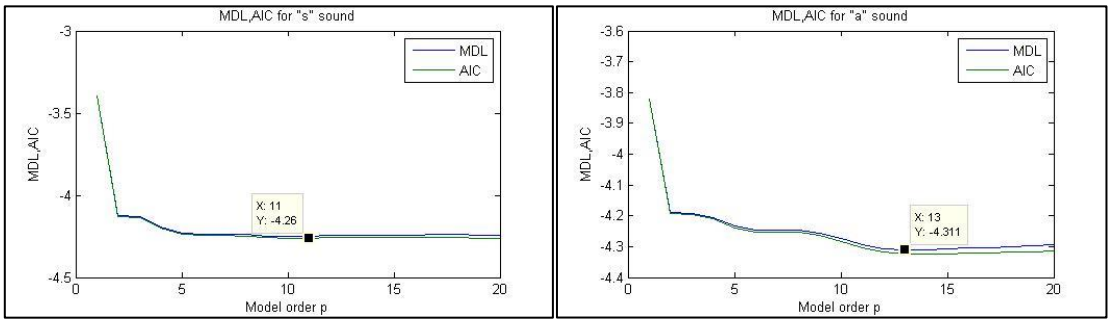

Figure 62: MDL and AIC for 2 sounds 's' and 'a'.

Another method is more heuristic, and probably gives more useful results. The MSE and prediction gain $\left(PG = 10 \times log10(\sigma_{input}^2/\sigma_e^2)\right)$ have been plotted for all the voice samples for a range of orders. The prediction gain simply measures the error power relative to input power, and a higher number is better. The results are shown in Figure 63. This shows that order 3 is optimal in MSE sense for sounds 't' and 's', whereas for the other sounds error continues to reduce slightly with increasing order.

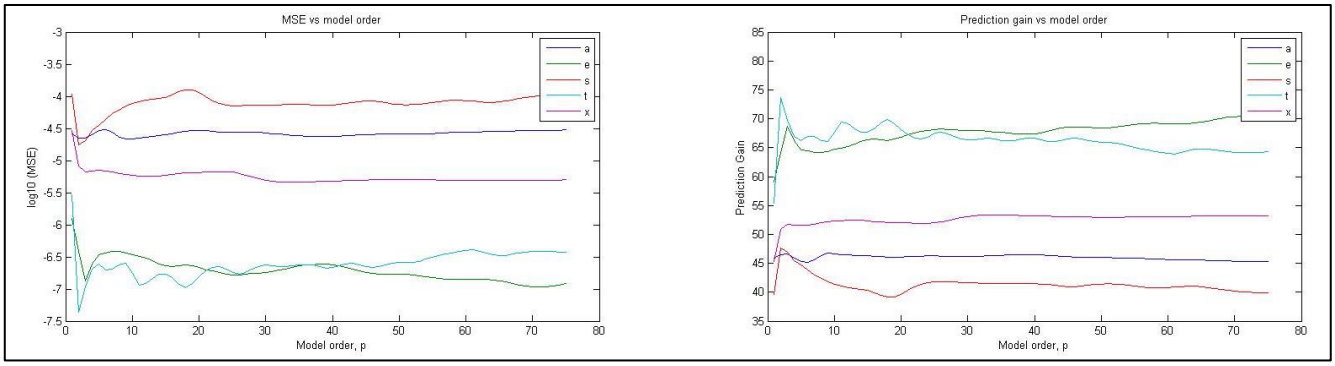

Figure 63: PG and MSE plotted against model order for all speech recordings. Adaptation gain is 1 for all cases.

To assess the performance of the predictor for different sample rates, the figure below has been created. It shows the MSE and PG for different model orders for all audio samples at both sample rates. A segment of 1000 sample right in the middle of the speech has been used to generate the results. In summary it shows higher MSE and lower PG for the lower sample rates for all orders. This is expected since the lower sample rate essentially means the data is less stationary so the error is larger. This is also illustrated in Figure 65 which shows the coefficients adapting for 1000 sample length segment for the sound 'x' for both sample rates. The lower sample rate is clearly less stationary. Therefore the weights spend more time adapting and the thus the error is greater. Quantitatively it is moving 44.1/16 = 2.75 times faster than the one sampled at 44.1 KHz.

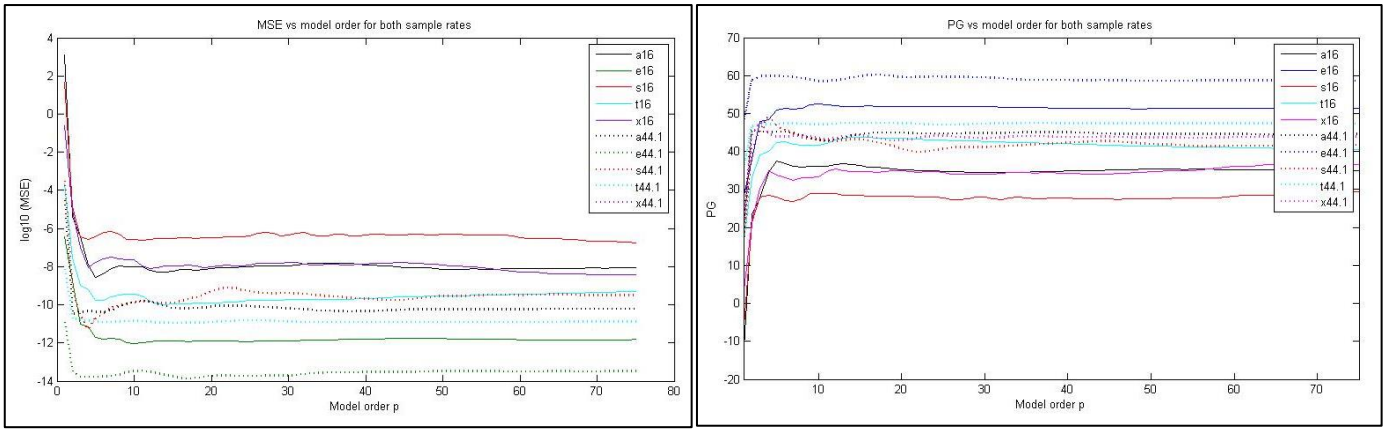

Figure 64: MSE and PG vs model order for 1000 sample segments of all recorded audio for both sample rates.

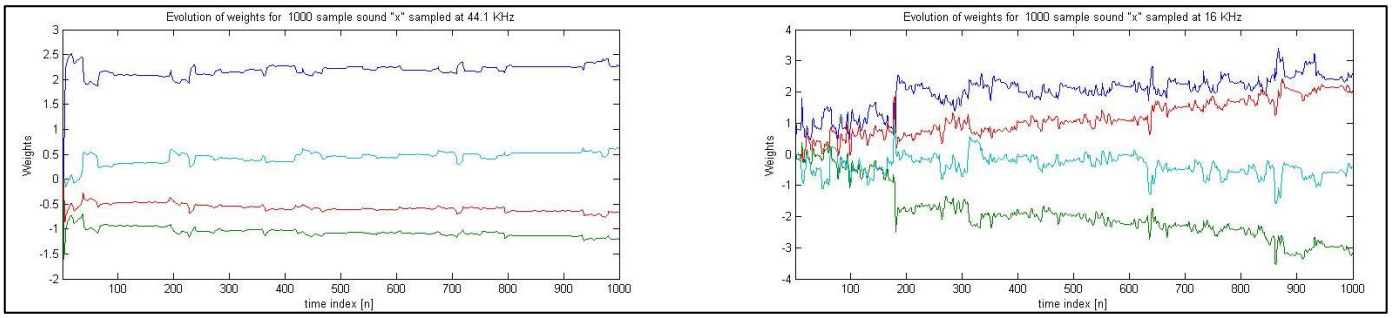

Figure 65: evolution of weights for sound 'x' for 2 different sampling rates. (Starting at the same point in time for 1000 samples)

4.6 Dealing with computational complexity: sign algorithms

To further reduce computational complexity, various sign algorithms can be used. These are essentially simplified versions of the original '*lms*' algorithm. For the *'signed error'* algorithm, the error is replaced by its sign, which saves q $(=$ filter order$)$ multiplications per iteration, hence $q \times N$ multiplications in total for an input of length N samples. The '*signed regressor*' algorithm has the $\boldsymbol{x}$ vector replaced by a vector of signs. This also has the same computational saving as the previous 'sign' method. The '*sign-sign'* algorithm is a combination of the two above and hence saves $2q \times N$ multiplications in total. Table 6 shows the computation time for each algorithm. The '*lms'* is slowest as expected and the '*signed error*' and '*signed regressor*' are similar and around 4 seconds faster than '*lms'*. The '*sign-sign'* is a further 4 seconds faster. This implies that it takes MATLAB 4 seconds to do $q \times N \times 5000 = 20\ million$ multiplications.

Algorithm	Computation time for 5000 trials of length 2000 / seconds
Lms	137.8
Signed - error	133.9
Signed - regressor	134.1
Sign sign	130.3

Table 6: Computation time comparison for the different algorithms

The learning curves for these algorithms are shown in Figure 66. These show the '*sign-sign'* to converge fastest and settle with the smallest MSCE. The '*signed regressor*' is the slowest to converge out of all the sign algorithms, but it is still faster than the original lms. Note that these results were all obtained with the same adaptation gain of 0.01.

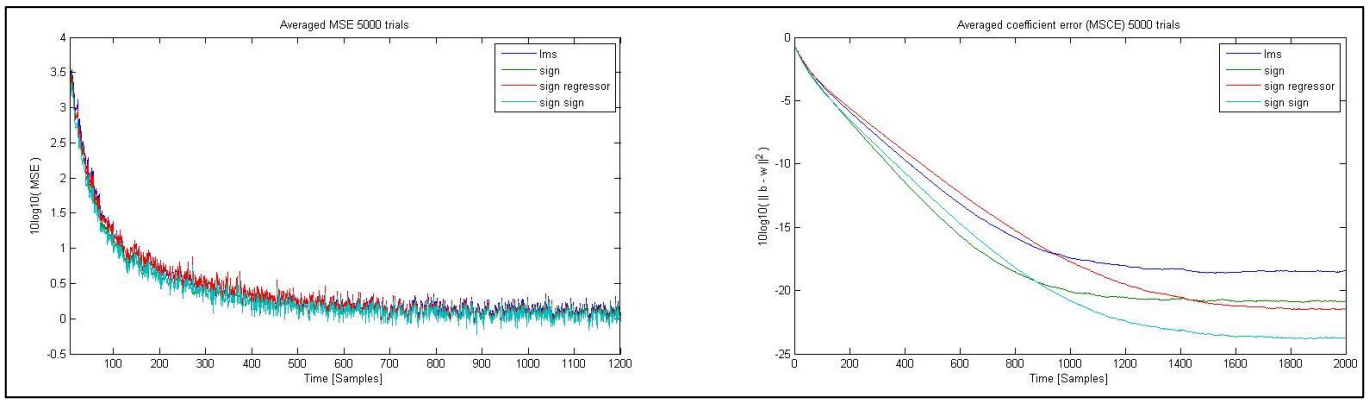

Figure 66: MSE and MSCE for different sign algorithms (adaptation gain is 0.01 for all)

Figure 67 shows the evolution of the weights for the three different algorithms with the input sound 't'. It shows that for the same adaption gain, the sign algorithms are much slower to rise. Out of all the sign algorithms, the 'sign-sign' is the fastest again.

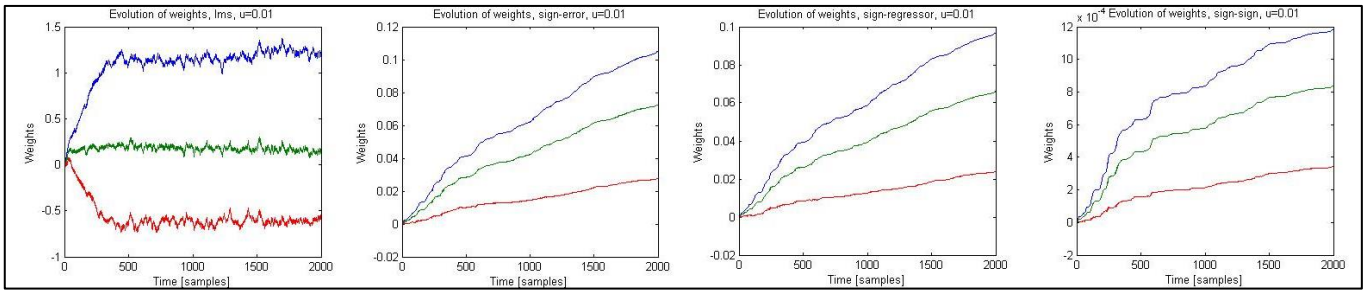

Figure 67: Evolution of weights for the speech 't' for the different sign algorithms, each with the same adaptation gain

Figure 68 shows the same plots but now with optimized adaption gains for each algorithm. The '*sign-sign'* settles with the smoothest line and second fastest rise time, just slightly slower than the '*sign error*'.

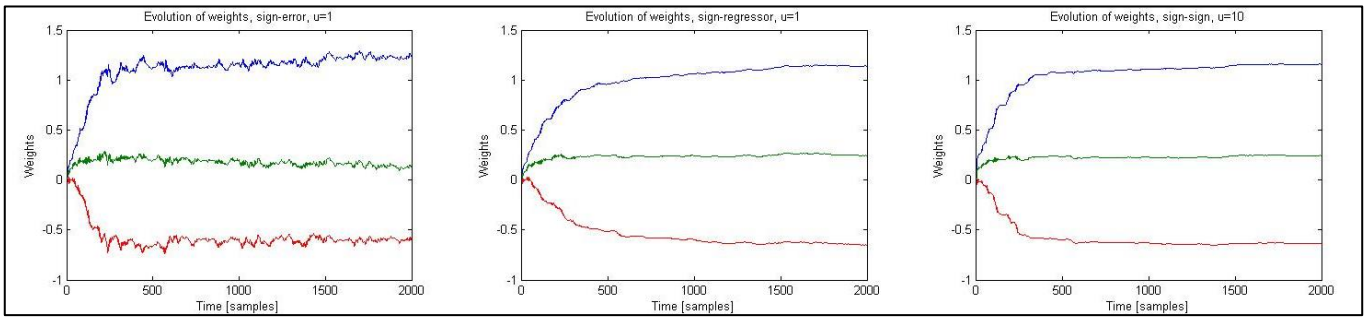

Figure 68: Evolution of weights for the speech 't' for the different sign algorithms, each with optimised adaptation gains

5 A Real World Case Study: Vinyl De-noising

1/2) PSD analysis of corrupted and clean audio segments:

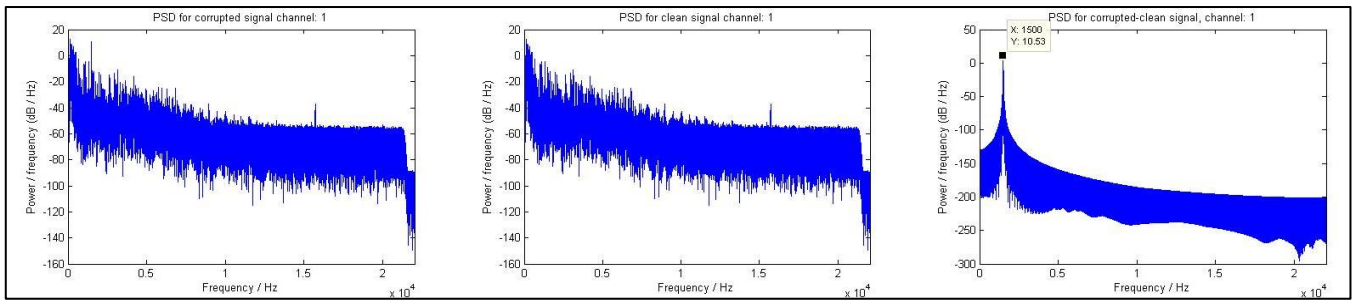

Figure 69: PSD of the corrupted, clean, and corrupted-clean signals. Channel 1 shown.

This section uses the pgm function to analyze the PSD of clean and corrupted signals for both channels. Looking at the PSD of the corrupted segment alone (Figure 69 left) makes it difficult to distinguish the spectral components of the tick and the song. Possibly if the tick were louder we would see peaks of bands of frequencies corresponding to the ticks. Even comparing the corrupted and clean signals' PSD (middle figure), is difficult to see any differences. This is partly due to the non-stationary nature of the tick and the song and the high spectral resolution which makes the PSD crowded. Subtracting the PSD gives a bit more information; the figure on the right shows that the frequencies associated with the tick for channel one are mostly at 1500 Hz. The figure below shows the same information for channel 2. It shows there are two major frequency components, 200 Hz and 1500 Hz, associated with the tick.

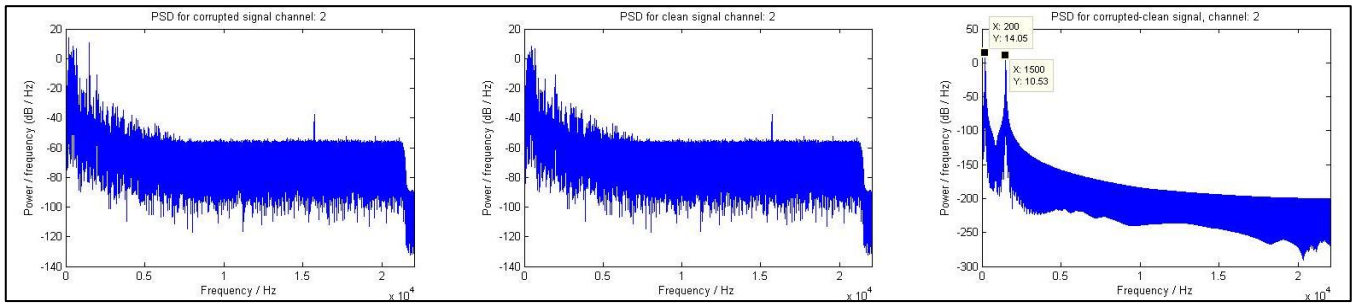

Figure 70: PSD of the corrupted, clean, and corrupted minus clean signals. Channel 2 shown.

It can be more beneficial to look at the averaged PSD of shorter frames. This is because the tick is clearly not continuous and not perfectly periodic, (the frequency of ticks reduces with time as shown in spectrograms later). Looking at the PSD of the whole signal will show the frequencies required to synthesize the whole sequence of ticks, which would be slightly different to the spectrum of a single tick. This is important since a filter designed to remove a single tick rather than the sequence of ticks is likely to have a narrower stop band, yet it may still be sufficient to remove the sound of the ticks. This would mean less of the music frequencies are removed and hence better overall quality result. If short successive frames are considered, some of these frames will contain a full tick sound from frame beginning to frame end. Since the tick is relatively high power (it sounds loud compared to the music), the frames containing the tick will have distinct high power peaks in their PSD. These will bring the average up in the averaged PSD and the precise components of the isolated ticks will be easily visible.

The figures below show the averaged periodogram obtained by averaging the PSD of smoothly windowed intervals of length 1024 samples (23ms). Hanning windows are used to reduce the spectral effects of rectangular windowing. 23 ms was chosen since the signal is relatively stationary during this interval and the tick has duration of 93ms. Furthermore, 1024 frequency bins also provide adequate spectral resolution for analysis in the frequency domain. As mentioned previously, now the frequencies associated with the tick are more clearly visible in the corrupted spectra even without having to subtract the corrupted spectra from the clean spectra. As a general comment, the fast cut-off close to half the sample frequency (22050 Hz) is due to the antialiasing filter.

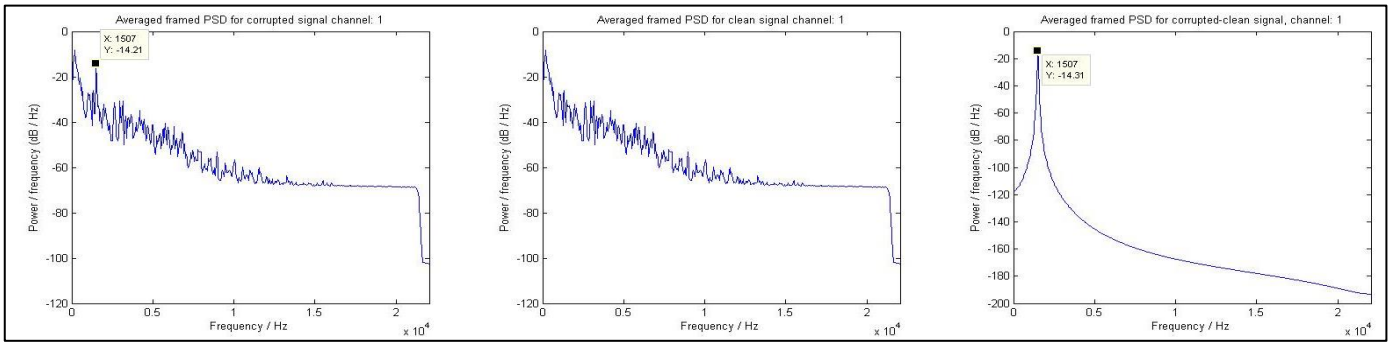

Figure 71: Averaged periodogram of corrupt, clean and corrupt minus clean signals for channel 1. Window is of length 1024 = 23ms.

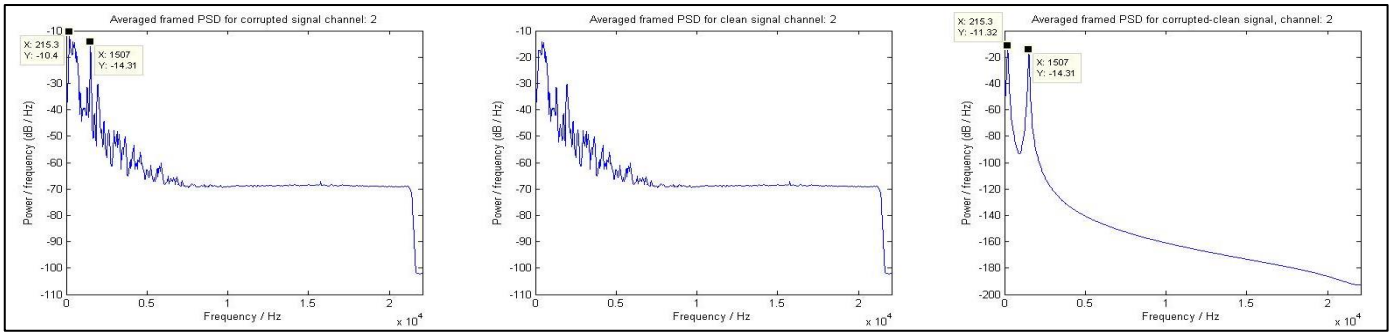

Figure 72: Averaged periodogram of corrupt, clean and corrupt minus clean signals for channel 2. Window is of length 1024 = 23ms.

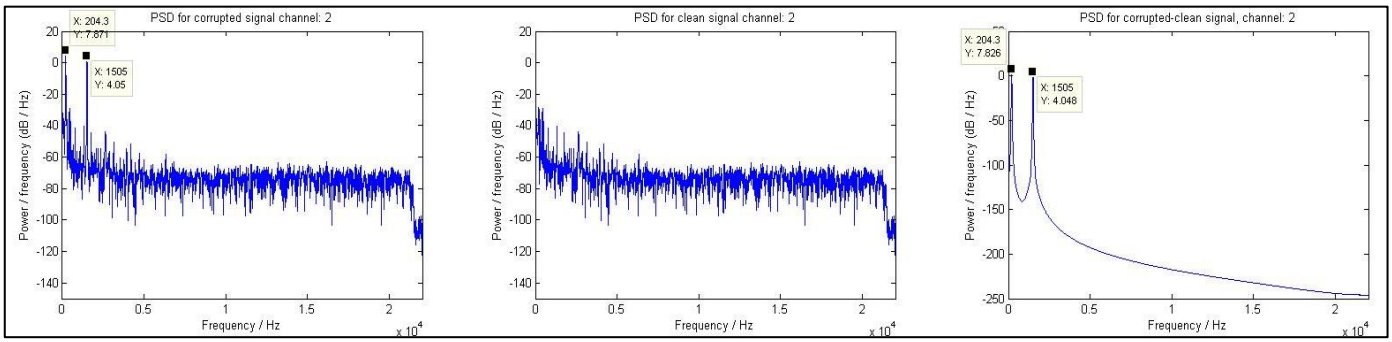

Figure 73: Periodogram over the interval 1.005-1.098 seconds for channel 2. This interval contains a full 'tick'.

Furthermore, analyzing small relatively stationary segments would make it easier to see any differences, especially if the segment contains the tick. Figure 73 shows the PSD of a segment from channel 2 containing a full tick which lasts 93ms. From this figure, even without subtracting the clean audio from the corrupted audio it is clear which frequencies are caused by the tick noise: 204.3, and 1505 Hz.

The spectrograms below show the clean and corrupt signals. Figure 76 shows the spectrogram for the corrupt signal minus the clean signal, i.e. the noise spectrum. The yellow signal at the top of each diagram shows the time domain waveform. The tick noise can be seen as the repetitive yellow high power spot that occurs at around 1.5 KHz and also at 0.2 KHz for channel 2. From these figures it is clear that the frequency of occurrence of the tick noise is reducing with time.

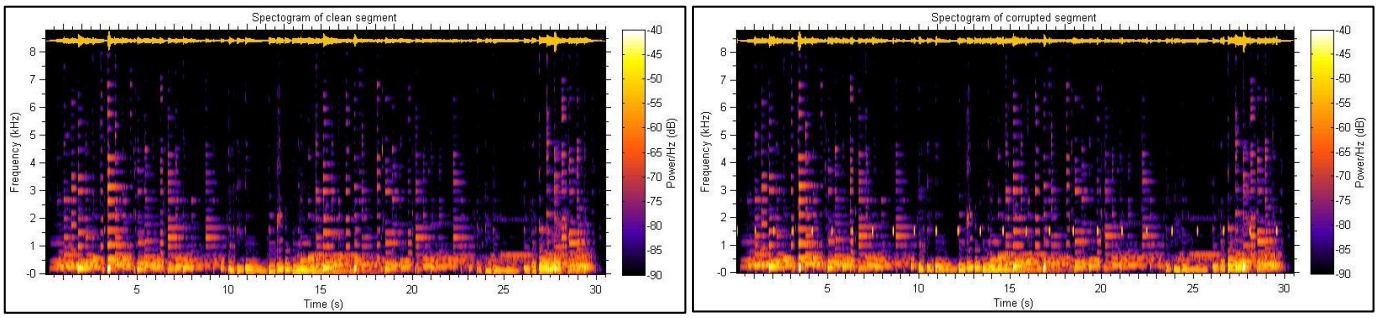

Figure 74: Channel 2, spectrogram of clean (left) and corrupted segments (right)

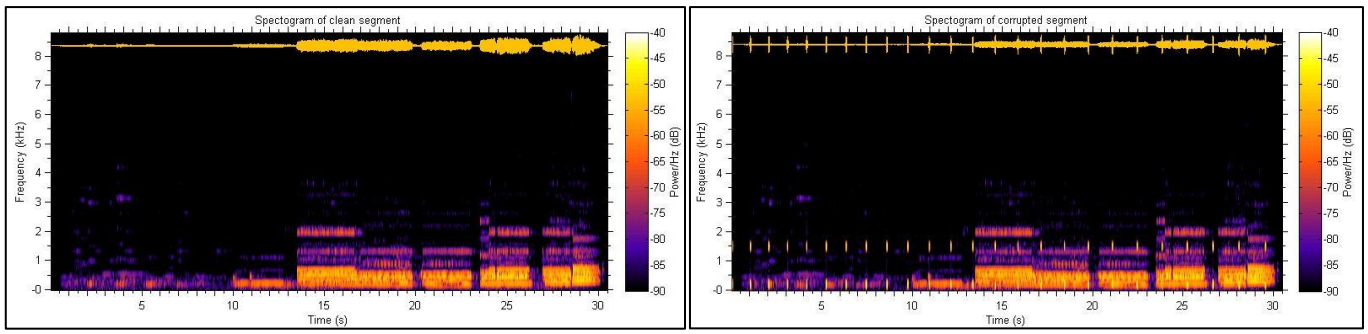

Figure 75: Channel 2, spectrogram of clean (left) and corrupted segments (right)

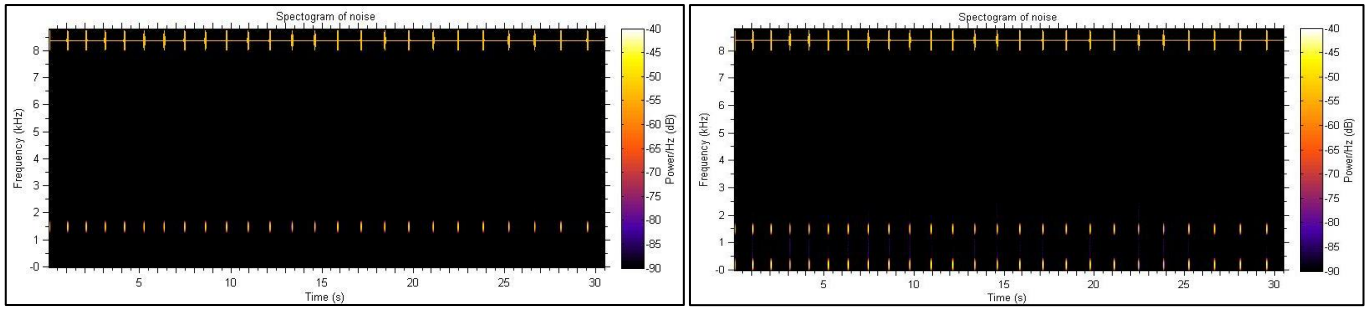

Figure 76: spectrogram of ticking noise. Left is channel 1, right is channel 2.

3) Fixed coefficient filters to remove the distortion:

From the previous section we can see that for channel 1 it seems appropriate to attenuate the range 1450 to 1550 Hz by around 40dB. This will essentially suppress the big peak at 1500 Hz shown in Figure 71 (right) and suppress the nearby energies as well which also probably contribute to the tick sound. Once the peak is suppressed, the remaining difference in the PSD is small and less than 100dB, so it will hardly make a difference in listening.

For channel 2, the frequency bands 175 to 265 Hz and 1450 to 1550 Hz should be attenuated by around 40dB. This will suppress the peaks shown in Figure 72 (right). FIR linear phase filters in band stop configuration are chosen. This is because linear phase is important in filtering audio since our ears are sensitive to phase distortion. The filter is also equi-ripple, and the ripples in the passband are chosen to be as small as MATLAB allows before convergence issues occur in the generation of the coefficients. The full specification is detailed in the table below, and the frequency responses are shown in Figure 77. Note that two filters in series are needed for channel 2 since there are two peaks to suppress.

Specification:	**Channel 1, filter 1:**	**Channel 2, filter 1:**	**Channel 2, filter 2:**
Passband ripple in dB	0.025	0.025	0.1
Stopband ripple in dB	40	40	40
Cut-off frequencies in Hz	[1350, 1450, 1550, 1650]	[1350, 1450, 1550, 1650]	[55, 175, 265, 365]

Table 7: Specifications for the three FIR equi-ripple linear phase filters. Designed using heuristic methods and judged by listening tests.

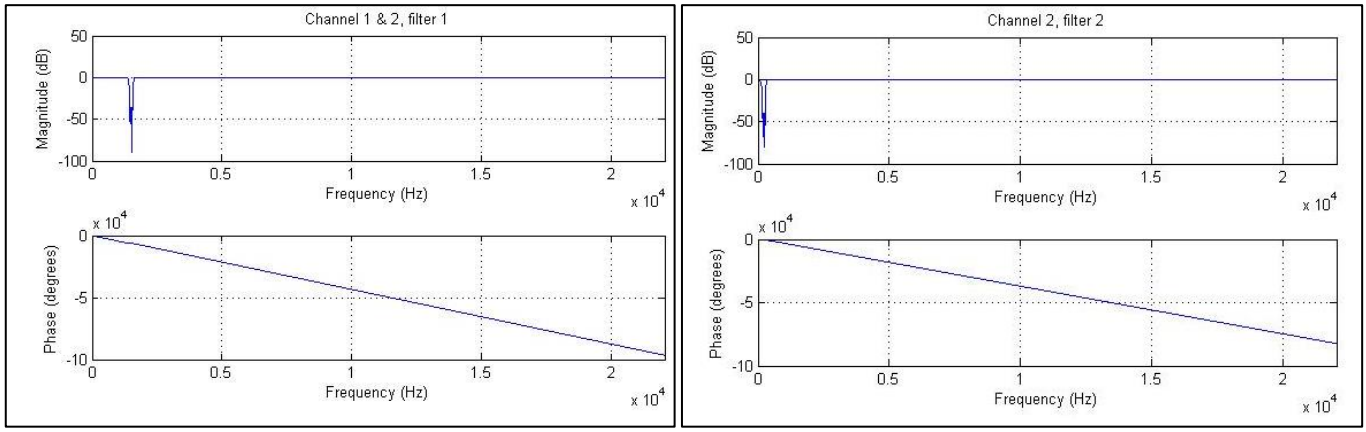

Figure 77: Frequency response of the filters. Note channel 1 and 2 have a common filter so only 2 graphs shown.

The delay introduced to channel 1 is 540 samples, and 1001 samples for channel 2. This delay will need to be considered when computing some time domain based quantitative measures such as the prediction gain, (PG).

Listening to channel 1 shows the tick noise to be almost completely removed. When the volume is increased a very faint tick can be heard during periods of quiet music. If the attenuation is increased further than 40db, the transition bands need to be made wider due to MATLAB convergence reasons. In this case the music loses some quality since more music frequencies are disturbed. For channel 2 it is easier to hear a faint tick since the music is quieter and there are two ticks. Increasing the attenuation has the same effect described previously, i.e. an overall reduction in music quality. Overall based on listening tests the filter specifications as in Table 7 appear to be optimal. As a general comment, an alternative to this time domain processing would be to directly adjust the peaks in the frequency domain. This could also be done in shorter successive frames (using the windowing and overlap add method). This would have the advantage that only the frames containing the noise peaks would have the peaks attenuated which would help preserve some music quality.

4) Comparison between filtered corrupt signal and ideal signal

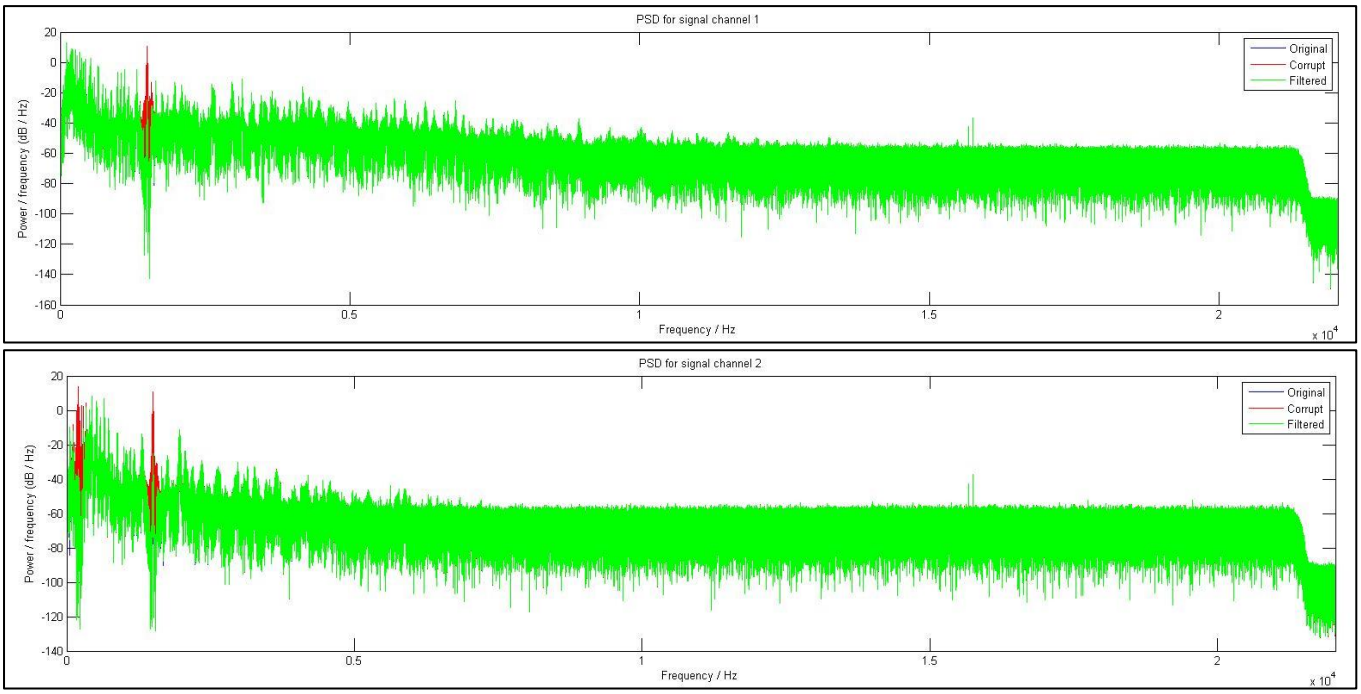

Figure 78: PSD for pre and post filtered signals for channel 1 and 2. The original PSD (blue) is hidden behind the corrupt and filtered PSD

Figure 78 shows the PSD of the original, corrupt and filtered signals for both channels. The places where red is showing show the effect of the band stop filter. Comparison with the original signal is difficult since it is hidden behind the other colors. Figure 80 shows the same signals, but with PSD averaged over consecutive windowed frames of length 1024. From this is it clearer that the attenuation of the band stop filters is too high since the filtered PSD is below the original PSD at the band stop regions. However this extra attenuation was necessary to remove the ticking sufficiently so that it could barely be heard. This is because this figure shows the average spectra and the peaks associated with the tick are not continuously occurring. They have high power but short duration, so the average is an underestimate of the peak, and the peak is what needs to be removed for good audio quality. The small differences between the filtered and original signals PSD at the passband frequencies is due to the small (equi-ripple) passband ripple of the filter.

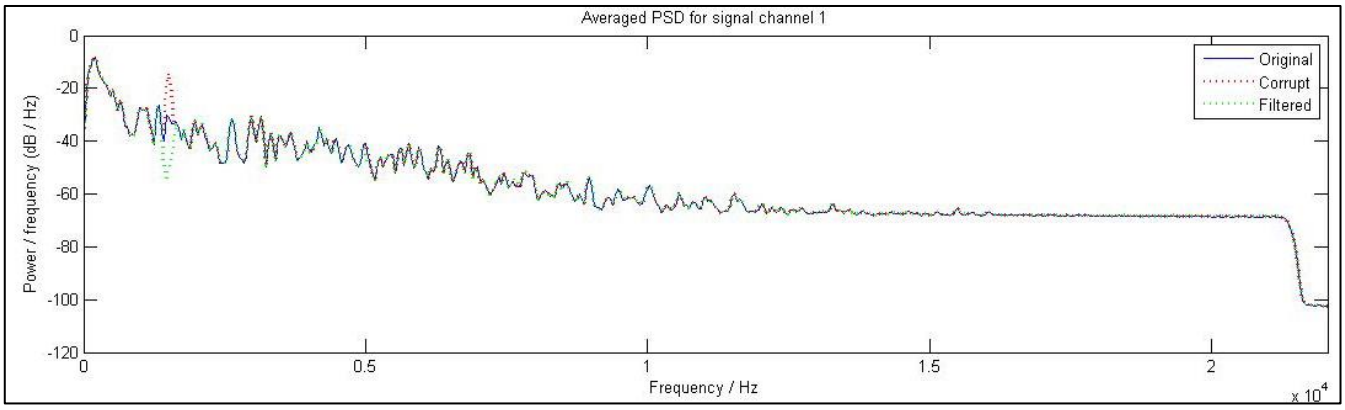

Figure 79: Averaged PSD for pre and post filtered signals for channel 1. The original signal is shown in blue.

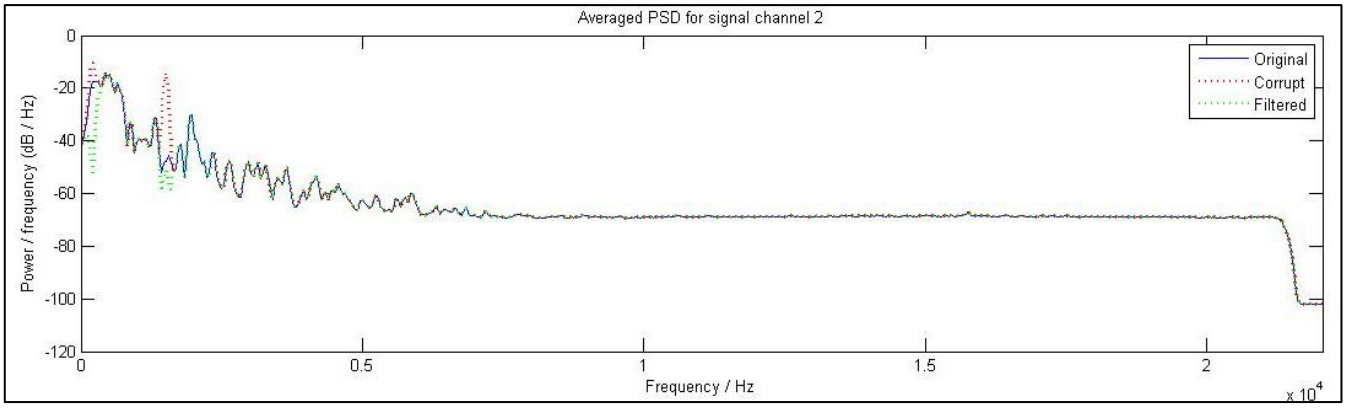

Figure 80: Averaged PSD for pre and post filtered signals for channel 2. The original signal is shown in blue.

The spectrograms below show the effects of the filters on the two channels. Comparing these to Figure 74 (right) and Figure 75 (right) we can see the repetitive tick sounds to be removed. However the cost is that the tick frequencies are removed for all 30 seconds, which means there is some loss in audio quality since music frequencies are also removed. The effect of this is clearer in Figure 81 (right) which shows two narrow black horizontal segments at 200 and 1500 Hz. Comparing to Figure 75 (left) which shows the ideal spectrogram for channel 2 it is clear some music frequencies are removed as well. Comparing the time domain waveform shown at the top of Figure 75 (right) and Figure 81 (right) also confirms the repetitive tick to be mostly removed.

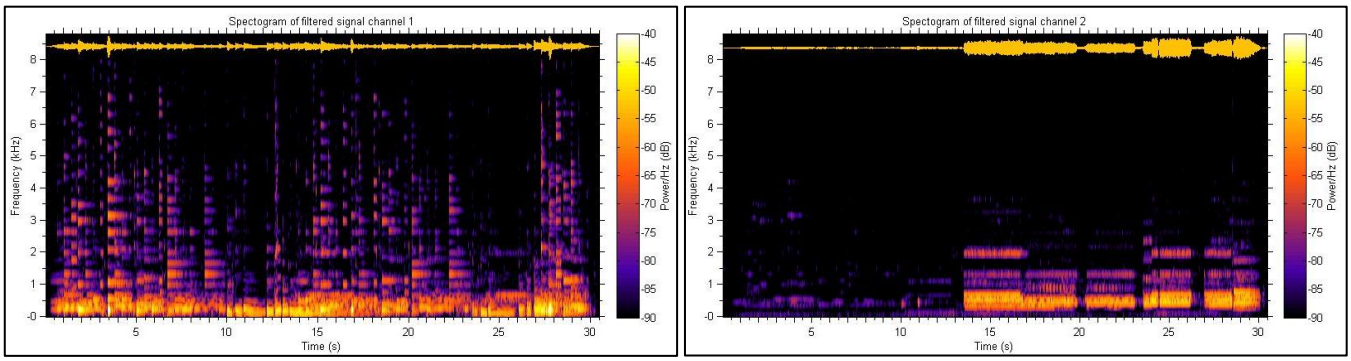

Figure 81: Spectrograms for the filtered signals for both channels.

Quantitative measures:

A quantitative performance measure of the relative error, (RE) is given by $\mathrm{RE} = \left\|\mathrm{P_c} - \widehat{\mathrm{P}}_\mathrm{c}\right\| \,/\, \|\mathrm{P_c}\|$, where $\mathrm{P_c}$ is the PSD of the original channel c, and the hat denotes the filtered corrupt signals PSD. The table below compares this performance measure with the non-filtered corrupt signal and the corrupt signal to see the improvements. The prediction gain, (PG) is also shown as a performance measure.

Channel	RE of filtered signal	RE of corrupt signal	PG of filtered signal	PG of corrupt signal
1	0.0083	0.2275	5.41dB	9.63 dB
2	0.3640	1.3296	6.69 dB	1.01 dB

Table 8: Quantitative performance measures (relative error and prediction gain) of pre and post filtered signals.

The RE should ideally be zero since it shows the relative difference between the ideal and corrupt signals. For both channels the RE reduces, however for channel one the reduction is 96%, whereas it is only 73% for channel 2. Initially for channel 2, the RE is also higher, and this is because the music is initially quieter for the first 13.5 seconds, and there are two major frequencies associated with the ticking noise. This essentially means the SNR is lower hence RE is greater. The reason the percentage decrease in RE for channel 2 is less is because two bands of frequencies are attenuated. This means more music frequencies are removed and hence RE cannot improve as much as in channel 1.

Before the PG was calculated, the filtered signal and ideal signal were time aligned. This is because the filter introduces delay which needs to be accounted for when comparing the ideal and filtered signals in the time domain. The higher the prediction gain, the better since it means the error of prediction is relatively small. For channel 2, the prediction gain increases by 5.68dB, meaning the error power reduced by a factor of 3.67. In the case of channel 2, the prediction gain actually worsens by 4.2dB. This is likely because the filter attenuated important music frequencies so much so that overall error increased. This is not the case for channel 2 since for the first 13.5 seconds there is hardly any music playing so during this time the noise is removed without removing significant music, which essentially helps improve the average PG.

5) Supervised adaptive denoising algorithm:

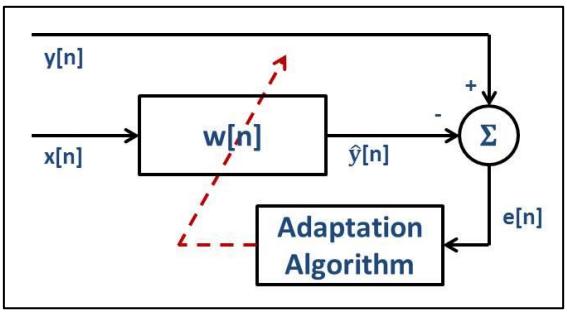

Figure 82: Structure for the supervised NLMS denoising algorithm

Figure 82 shows the structure used to denoise the corrupt channels. $\mathrm{x[n]}$ is the noisy input and $\mathrm{y[n]}$ is the teaching signal with $\hat{\mathrm{y}}\mathrm{[n]}$ the denoised signal. The Adaptive algorithm is the NLMS algorithm with the '*gear2*' modification as described in 4.3. However, some further modifications have been made. Firstly, the estimated minimum MSE is now set to zero since it is unknown, (and because this structure actually allows the MSE to tend to zero for some periods of time, ie during tick free

periods of corrupt input x[n]). Additionally, the scaling factors have been adjusted for optimal performance (in the sense of RE), and the explicit gears have been removed. This is because they made no noticeable difference and they just add computational load. Now there is only a single gear, i.e. adaptation gain is just proportional to the squared error with a single proportional constant. (Previously this constant changed based on squared error, hence the explicit gears).

Note that although the coursework handout says $\hat{y}[n] = \boldsymbol{w}_n^T \boldsymbol{x}_n$, with $\boldsymbol{x}_n = \left[x[n-1], \ldots, x[n-p]\right]^T$, , the structure works best with $\boldsymbol{x}_n = \left[x[n], x[n-1], \ldots, x[n-p]\right]^T$. The denoising can be explained by recognizing that the FIR filter will be adjusted such that the MSE is minimized. This means when a tick noise occurs, the filter will adapt into a band stop filter to remove the tick. Once the tick disappears, the filter will go back into an all pass. This will therefore give better results that the constant coefficients filter since this doesn't attenuate music frequencies during non-tick noise periods. The evolution of the coefficients with time for a 100 order filter used for channel 1 is shown in Figure 83. This shows the coefficients to change periodically when the tick noises occur (e.g. at sample number 17000 and 23000). The average of these coefficients over a short time period has been taken for a tick period and a non-tick period, and their frequency response plotted. This is shown in Figure 84, which confirms the coefficients to form into a band stop during a tick period and an all pass during a noise free period.

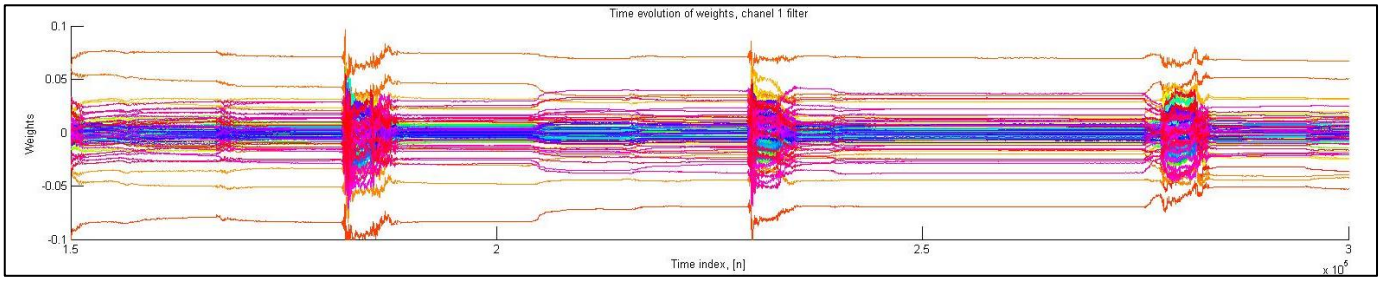

Figure 83: Time evolution of coefficients for channel 1. (Filter is order 100). Abrupt changes during tick periods are clear.

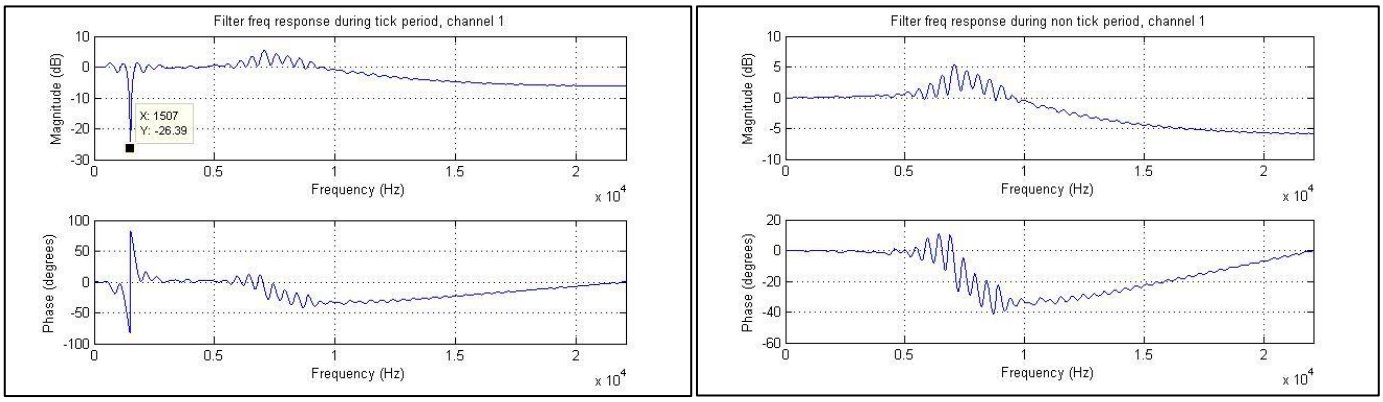

Figure 84: (Left), frequency response of the adaptive filter during tick period. Frequency 1507 is attenuated by -26.4dB. (Right), frequency response during a noise free period; It is roughly all pass with small phase & amplitude distortion for the important music frequencies.

The optimal order in terms of RE for channel 1 and 2 is 100 and 450 respectively. This was determined by heuristic methods and the graph for RE and PG for different orders is shown in Figure 85. An adaptation gain of 1.5 is also optimal in terms of smallest RE and largest PG. Table 9 summarizes the optimal results and compares the performance relative to the fixed coefficient filter. The reason channel 2 needs a higher order is because there are two bands of frequencies which need attenuating during noise periods.

Channel	RE of filtered signal	PG of filtered signal	Improvement in RE	Improvement in PG
1	0.00116	32.16dB	86.0%	26.75 dB
2	0.00474	29.88dB	98.7%	23.19 dB

Table 9: Quantitative performance measures for ch1 and ch2 respectively, and their improvements compared to the fixed coefficient filter

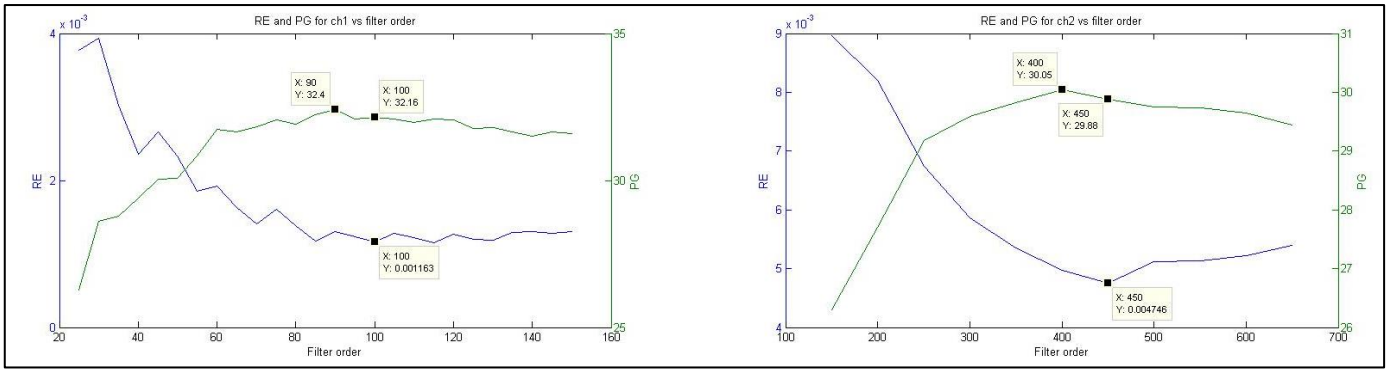

Figure 85: RE and PG for vs filter order for both channels. (Adaptation gain is 1.5)

The figures below show the final spectral results. Figure 86 and Figure 87 show the filtered signals PSD to be closer to the ideal one at the noise frequencies (1500 and 200Hz) compared to the fixed coefficient filter which over attenuates (Figure 79, Figure 80). Although the green line differs from the blue line at high the frequencies above 7 KHz compared to the fixed coefficient filter, this is unimportant. This is because the power of these frequencies is negligible at less than -60dB, so this doesn't reduce audible music quality or RE.

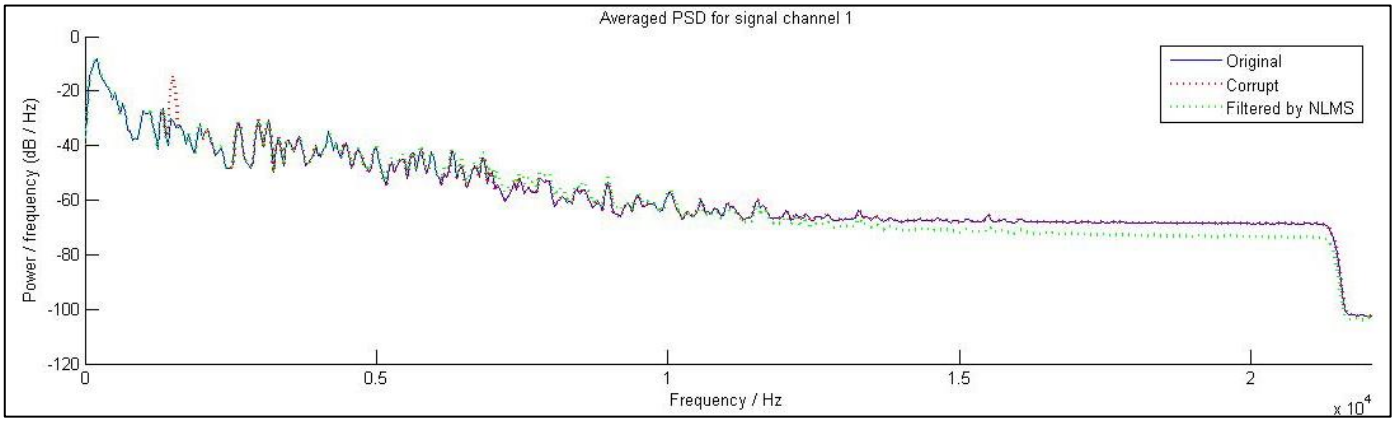

Figure 86: Averaged PSD (1024 samples is frame size) for pre and post filtered signals for channel 1. The original signal is shown in blue

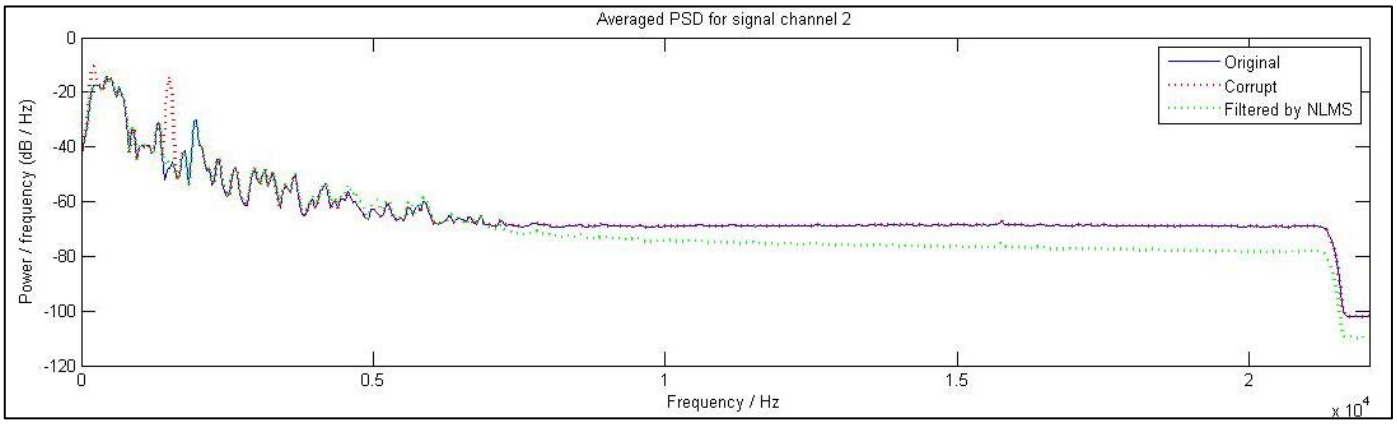

Figure 87: Averaged PSD (1024 samples is frame size) for pre and post filtered signals for channel 2. The original signal is shown in blue.

The spectrograms below of the filtered signals are closer to the ideal ones shown in Figure 74 and Figure 75. They show that the frequencies around 1500Hz and 200 Hz are not always attenuated unlike the fixed coefficient filter (Figure 81). In terms of audio quality, the filtered signal sounds identical to the original and no hint of a background tick can be heard in either channel regardless of volume.

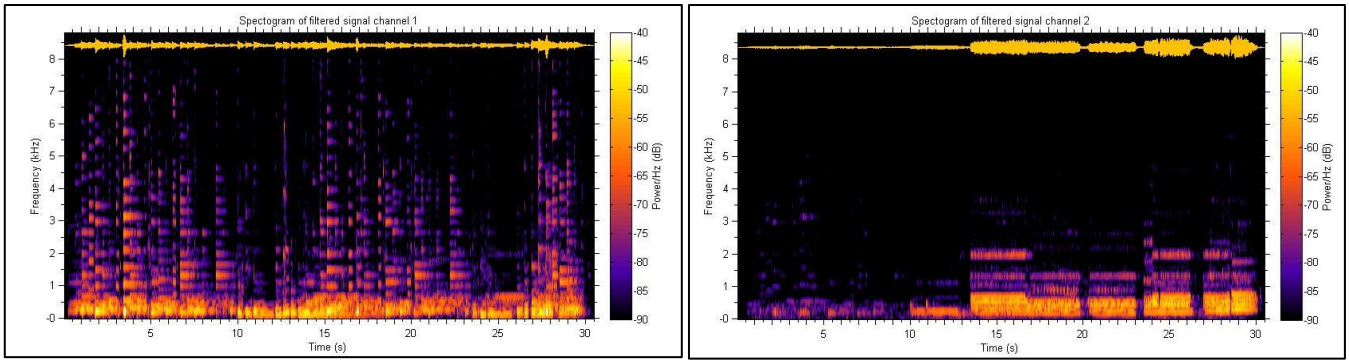

Figure 88: Spectrograms of both channels for the filtered signals.

The lms algorithm was also investigated. For this to work correctly, the adaptation gain was changed from 1.5 to 0.01, and the best results it gave were a RE and PG of 0.0194 and 17.2dB for channel 1, and 0.0383 and 19.0dB for channel 2. This is significantly worse than previously and in fact the RE for channel 1 is higher than the fixed coefficient filter. Furthermore, a fainter and slightly distorted tick can still heard in the background. The reason the NLMS algorithm performs better than the LMS is because the NLMS adjusts the adaptation gain according to the power of the signal in the filters memory. We know that for convergence in the MSE sense, the gain must be less than $2/(signal\ power)$. In the case of music, signal power varies with time, e.g. for channel 2 first the music is quiet for the first 13.5 seconds, and then much louder. This means a smaller adaptation gain needs to be chosen to ensure convergence for all parts of the signal, and hence poor rise times and poor RE. In the case of the NLMS, the signal power is factored into the equation so does not cause these convergence problems, hence better RE performance.

6) Denoising the LTE vinyl:

The same supervised denoising algorithm manages to remove the tick from the LTE recording. However, this recording is much richer in spectral content because of the multiple layers of each instrument and the harmonics of the distorted guitars used. Therefore greater error is expected. The table below compares the RE and PG for both channels and as s a benchmark, the corrupt signal is also compared. The table below confirms greater error compared to the results shown in Table 9.

Channel	RE of filtered signal	RE of corrupt signal	PG of filtered signal	PG of corrupt signal
1	0.0066	0.1892	28.7 dB	14.9 dB
2	0.0105	0.4501	28.6 dB	9.83 dB

Table 10: Quantitative performance measures (relative error and prediction gain) of pre and post filtered signals

The spectral distortion/loss as a result of removing the ticks is shown in the averaged periodograms below which compare the ideal, corrupt and filtered signals. For channel 2 we see some difference in the filtered and ideal PSD around 1.8 KHz. The powers are not insignificant here at around -40dB so this distortion results in a slightly greater relative error. However there seems to be no audible difference in the filtered and ideal signals. Above 2 kHz the distortion is negligible since the powers are small at around -90dB.

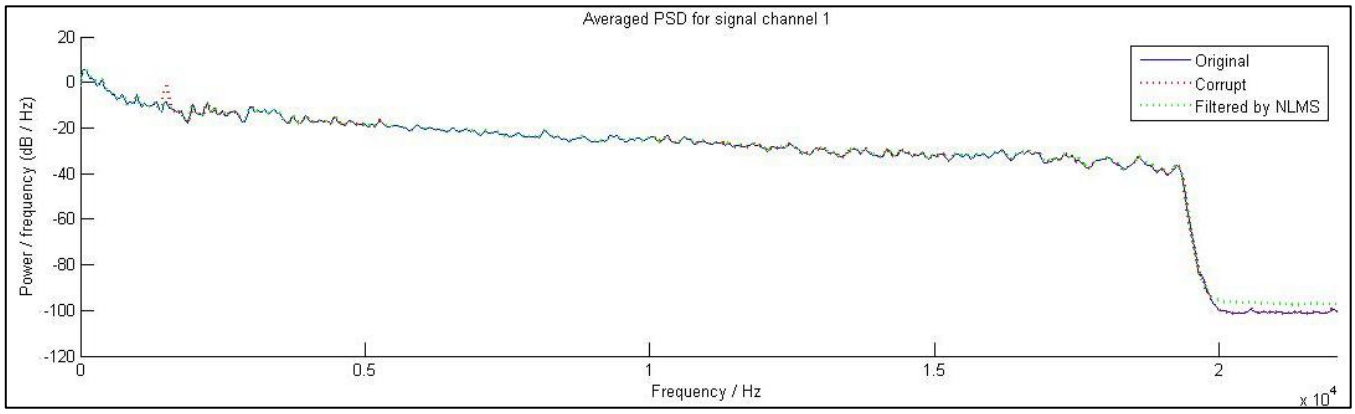

Figure 89: Averaged PSD (1024 samples is frame size) for pre and post filtered signals for channel 1. The original signal is shown in blue.

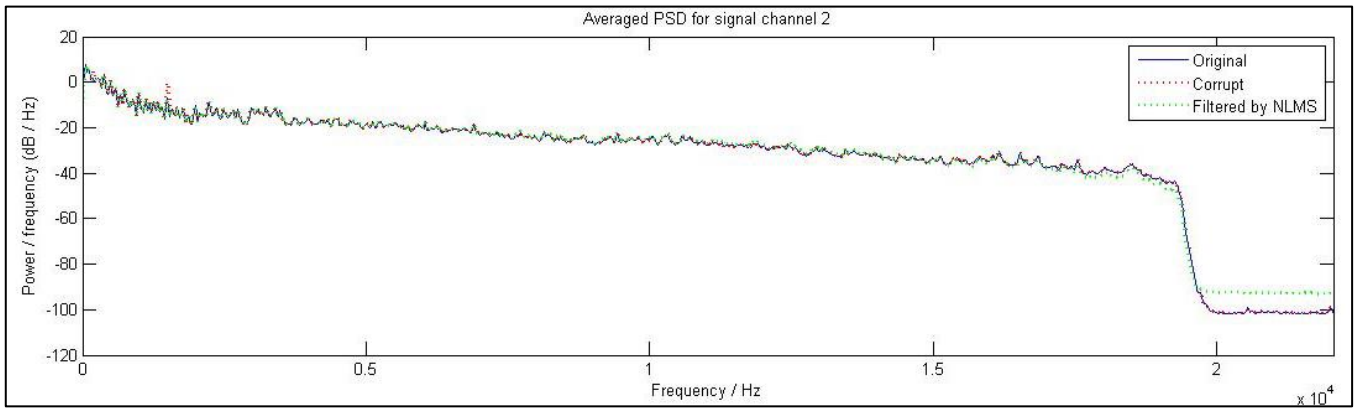

Figure 90: Averaged PSD (1024 samples is frame size) for pre and post filtered signals for channel 2. The original signal is shown in blue.

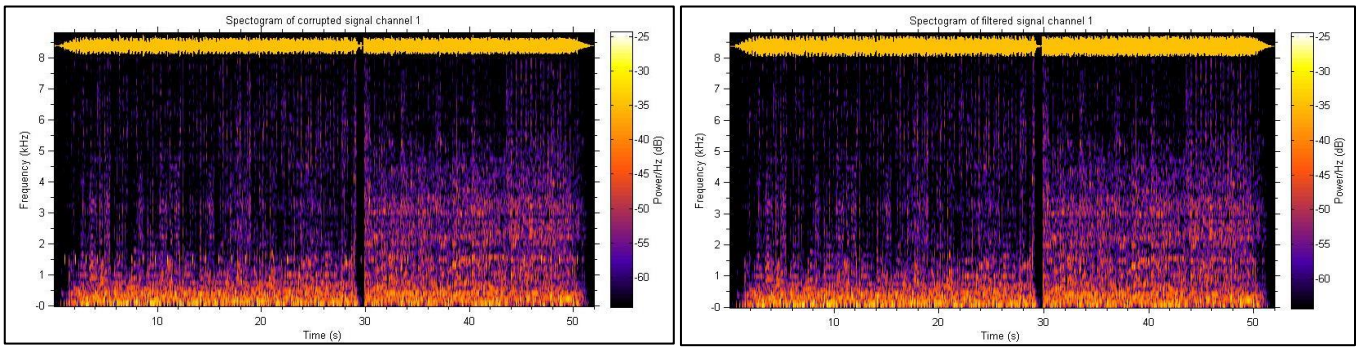

Figure 91: The corrupt and filtered spectrograms for channel 1

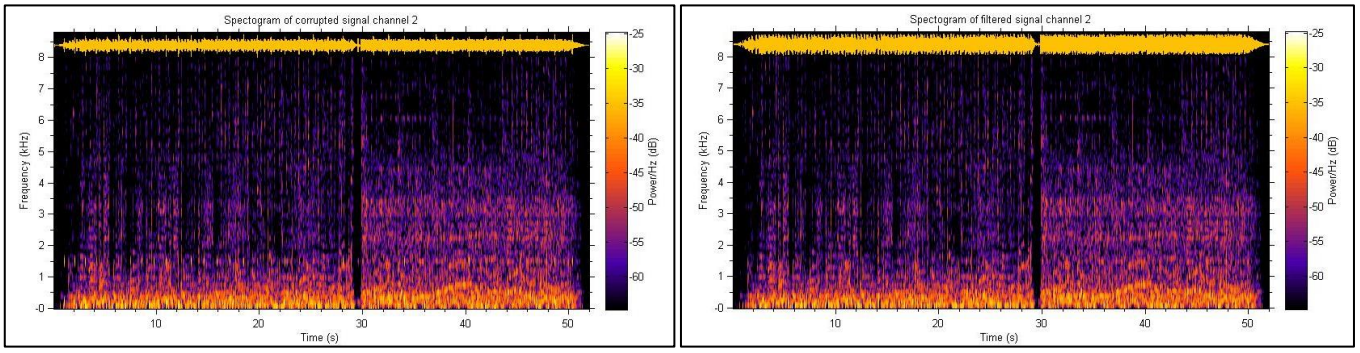

Figure 92: The corrupt and filtered spectrograms for channel 2

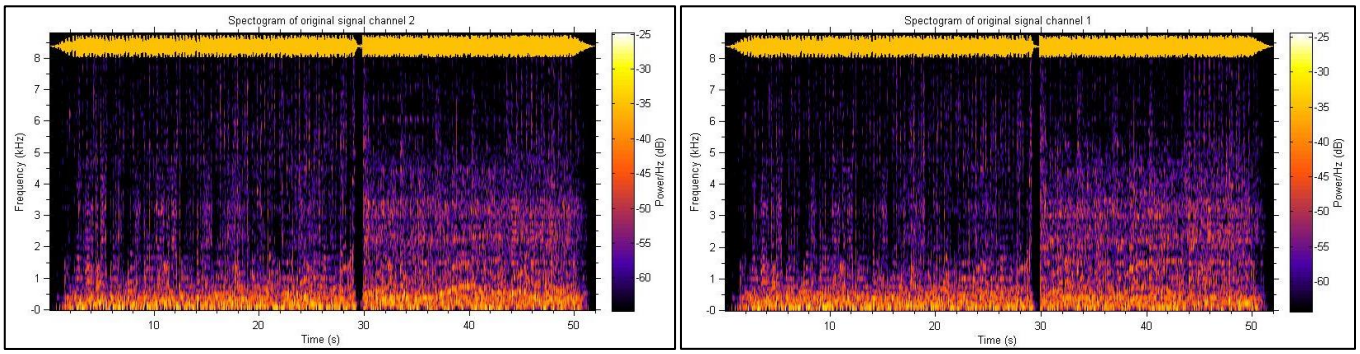

Figure 93: The ideal spectrograms as a reference

From Figure 91 and Figure 92 we can just about see that the yellow repetitive spots corresponding to the tick noise are removed by the filter. The filtered spectrograms look identical to the original ones as shown in Figure 93, however it is difficult to compare fully due to the high spectral content. The main point is that the adaptive filter does not remove the music at the tick frequencies for all 50 seconds, but rather just when the ticks occur. In terms of music quality, no noticeable difference can be heard from the filtered and ideal signals for both channels, and the tick is completely removed. As a comparison, the spectrogram of channel 2 filtered by the fixed coefficient filter is shown in Figure 94. This shows the spectral loss caused by filtering the noise frequencies for the full 50 duration of the music.

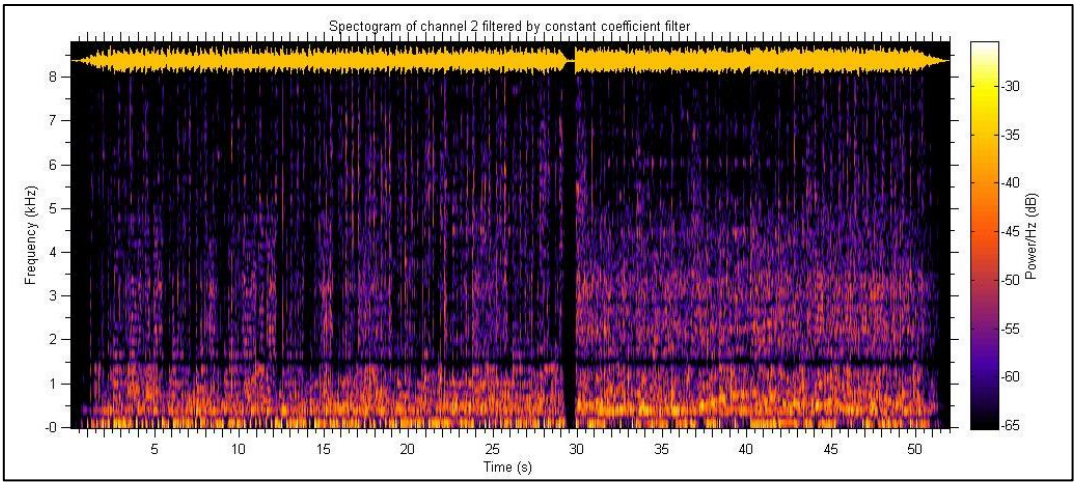

Figure 94: Channel 2 filtered by fixed coefficient filter. Spectral loss at 200 and 1500 Hz is clear.

In conclusion, adaptive filters are far more flexible than fixed coefficient filters. Their filtering ability is greater and their flexibility makes them applicable to a wide range of scenarios from adaptive noise filtering or linear prediction or adaptive system identification. The NLMS algorithm is a stable and effective method for minimizing the MSE in all these configurations. Simple modifications such as gear shifting can be made to improve the transient characteristics of the filter.