Advanced Signal Processing Coursework

Ata Yardimci

April 13, 2020

Contents

1 Random signals and stochastic processes

1.1 Statistical estimation

We generate and plot a 1000-sample realisation of the uniform random variable $X_n \backsim \mathcal{U}(0,1)$.

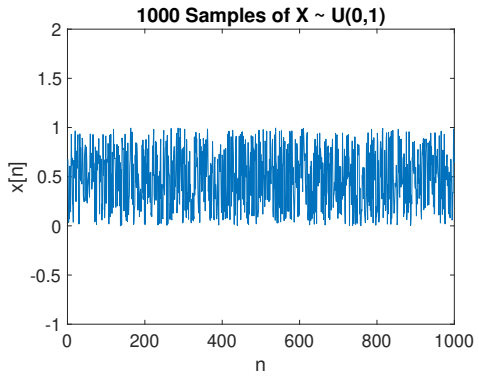


Figure 1: 1000-sample realisation of a stationary stochastic process $X_n \backsim \mathcal{U}(0,1) \quad \forall n$

(1) Theoretical mean:

$$m = \mathbb{E}\{X\} = \int_{-\infty}^{\infty} x \cdot p(x) \cdot dx = \int_0^1 x \cdot 1 \cdot dx = \left[\frac{x^2}{2}\right]_0^1 = 0.5 \tag{1}$$

Sample mean:

$$\widehat{m} = \texttt{mean(x)} = 0.4948 \tag{2}$$

We observe that the sample mean is very close to the theoretical mean.

(2) To find the theoretical value of the standard deviation, we can derive

$$\sigma^2 = \mathbb{E}\{(X - \mathbb{E}\{X\})^2\} = \mathbb{E}\{(X - \mu_X)(X - \mu_X)\} \tag{3}$$

$$= \mathbb{E}\{X^2 - 2X\mu_X + \mu_X^2\} = \mathbb{E}\{X^2\} - 2\mu_X\mathbb{E}\{X\} + \mu_X^2 \tag{4}$$

$$\sigma^2 = \mathbb{E}\{X^2\} - \mu_X^2 \tag{5}$$

We know $\mu_X = \text{m} = 0.5$ and

$$\mathbb{E}\{X^2\} = \int_0^1 x^2 \cdot p(x) \cdot dx = \left[\frac{x^3}{3}\right]_0^1 \approx 0.33 \tag{6}$$

So

$$\sigma^2 = 0.33 - 0.5^2 = 0.083 = \frac{1}{12} \tag{7}$$

and

$$\sigma \approx 0.289 \tag{8}$$

And for the sample estimate we get

$$\widehat{\sigma} = \texttt{std(x)} = 0.2880 \tag{9}$$

Like the sample mean, sample estimation of the standard deviation is very close to its theoretical value.

(3) We generate an ensemble of ten 1000-sample realisations of X and calculate their sample means and standard deviations. We plot them in figure 2. We observe that the ensemble means cluster around their theoretical value 0.5. Same applies to standard deviations which cluster around their theoretical σ value of 0.289. Realisation 5 has the highest bias in its mean estimation with 0.013 and realisation 1 has the highest bias in its σ estimation with 0.009. Both biases are very close to 0 as expected indicating unbiased estimator.

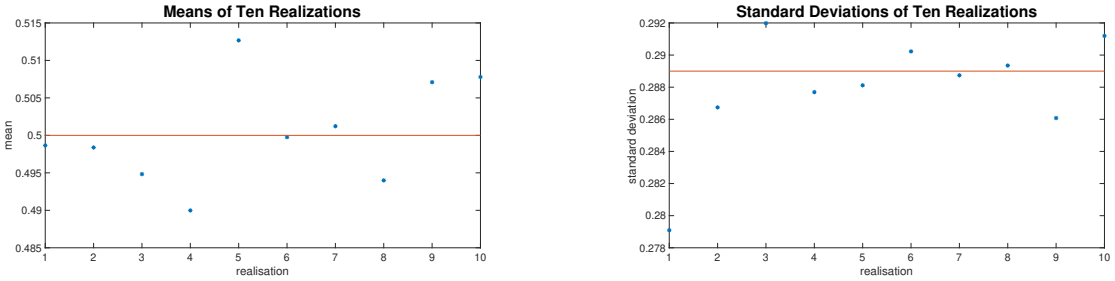
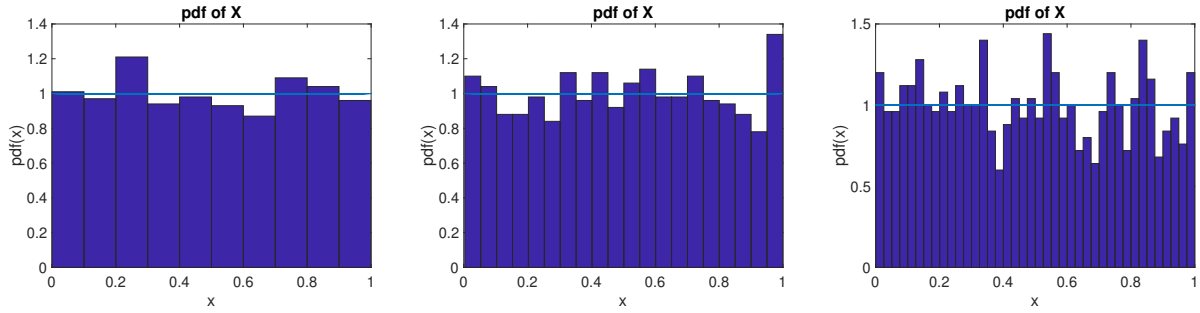


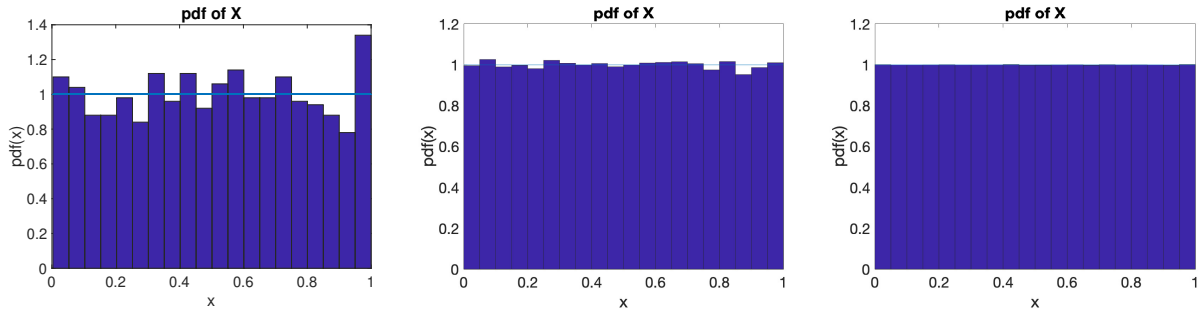
Figure 2: (Theoretical values given with a straight line)

(4) We approximate the probability density function of random variable X using histograms. The straight line gives the theoretical pdf of the uniform distribution between 0 and 1, and the bins are the estimates of this function using the samples generated with this distribution. We create the plots by using the `hist` function normalizing the bins by dividing their value by the number of samples times the width of the bins.



(a) 1000 samples, 10 bins (b) 1000 samples, 20 bins (c) 1000 samples, 40 bins

Figure 3: Changing the number of bins



(a) 1000 samples, 20 bins (b) 10^5 samples, 20 bins (c) 10^7 samples, 20 bins

Figure 4: Changing the number of samples

We observe in Figure 3 that increasing number of bins create more fluctuations around the theoretical value of the pdf. However, increasing number of samples allow the estimate to converge to its theoretical

value, which can be seen from Figure 4. Increasing the number of bins can be thought as improving the resolution, and increasing the number of samples can be seen as improving accuracy.

(5) We repeat the previous parts for zero-mean, unit standard deviation Gaussian random variables.

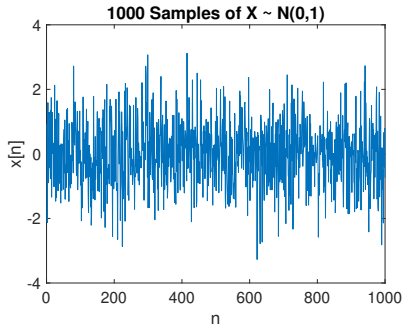


Figure 5: 1000-sample realisation of a stationary stochastic process $X_n \backsim \mathcal{N}(0,1), \quad \forall n$

Theoretical mean:

$$m = \mathbb{E}\{X\} = \int_{-\infty}^{\infty} x \cdot p(x) \cdot dx = 0 \tag{10}$$

We know that the pdf of a Gaussian distribution is symmetric around 0 (even function). We also know that x is an odd function, making the function inside the integral an odd function. Taking the integral of this function with such bounds, we get 0.

Sample mean:

$$\widehat{m} = \texttt{mean(x)} = 0.0355 \tag{11}$$

Just as before, we observe that the sample mean is very close to the theoretical mean. For the **theoretical standard deviation**, we know

$$\sigma^2 = \mathbb{E}\{X^2\} - \mu_X^2 \tag{12}$$

We know $\mu_X = \text{m} = 0$ and the pdf of the distribution

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \cdot e^{-\frac{(x-\mu)^2}{2\sigma^2}} = \frac{1}{\sqrt{2\pi}} \cdot e^{-\frac{x^2}{2}} \tag{13}$$

We need to find $\mathbb{E}\{X^2\}$ and do the following calculation:

$$\mathbb{E}\{X^2\} = \int_{-\infty}^{\infty} x^2 \cdot p(x) \cdot dx = \frac{1}{\sqrt{2\pi}} \cdot \int_{-\infty}^{\infty} x^2 \cdot e^{-\frac{x^2}{2}} \cdot dx \tag{14}$$

(Integrate by parts)

$$\mathbb{E}\{X^2\} = \frac{1}{\sqrt{2\pi}} \cdot \int_{-\infty}^{\infty} e^{-\frac{x^2}{2}} \cdot dx \tag{15}$$

Call I $= \int_{-\infty}^{\infty} e^{-\frac{x^2}{2}} \cdot dx = \int_{-\infty}^{\infty} e^{-\frac{y^2}{2}} \cdot dy$

$$I^2 = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} e^{-\frac{x^2+y^2}{2}} \cdot dxdy \tag{16}$$

We do a change of coordinates:

$$x = r \cdot cos(\theta) \qquad y = r \cdot sin(\theta) \qquad x^2 + y^2 = r^2 \qquad dxdy = rd\theta dr$$

$$I^2 = \int_{\theta=0}^{2\pi} \int_{r=0}^{\infty} e^{-\frac{r^2}{2}} \cdot r \cdot drd\theta \tag{17}$$

$$I = \sqrt{2\pi} \qquad \text{and} \qquad \mathbb{E}\{X^2\} = 1$$

So

$$\sigma^2 = \mathbb{E}\{X^2\} - \mu_X^2 \ = 1 - 0 = 1$$

$$\sigma = 1 \tag{18}$$

And for the **sample estimate** we get

$$\widehat{\sigma} = \texttt{std(x)} = 1.0224 \tag{19}$$

Like the sample mean, the sample estimation for the standard deviation is very close to its theoretical value, just as it was for the uniform distribution.

Ensemble of ten realizations: Similar observations as in part 1.1.3. We see that the ensemble means cluster around their theoretical value 0 and standard deviations cluster around their theoretical σ value 1. Realisation 2 has the highest bias in its mean estimation with 0.061, and realisation 1 has the highest bias in its σ estimation with 0.062. Both biases are very close to 0 as expected indicating unbiased estimator.

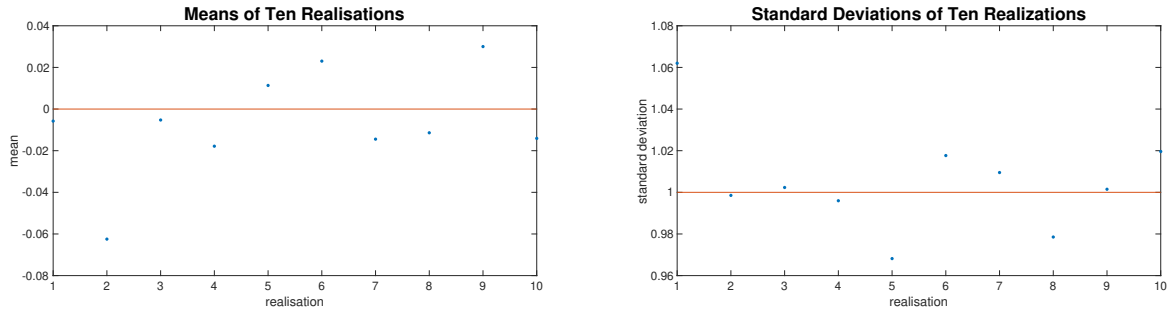
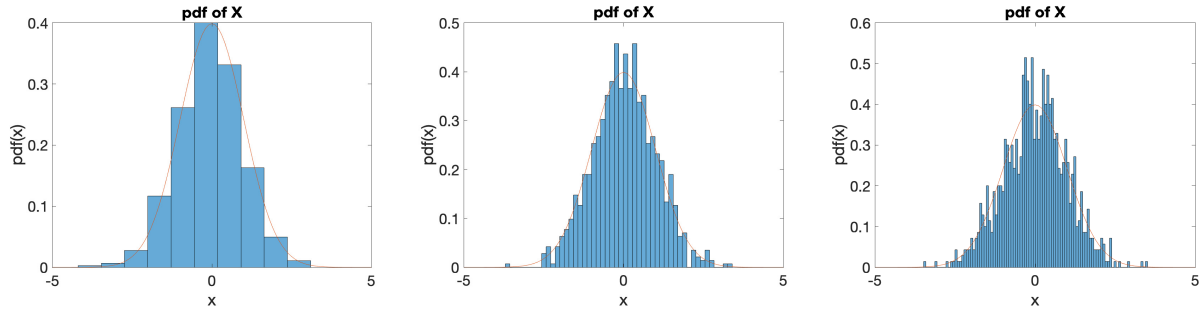


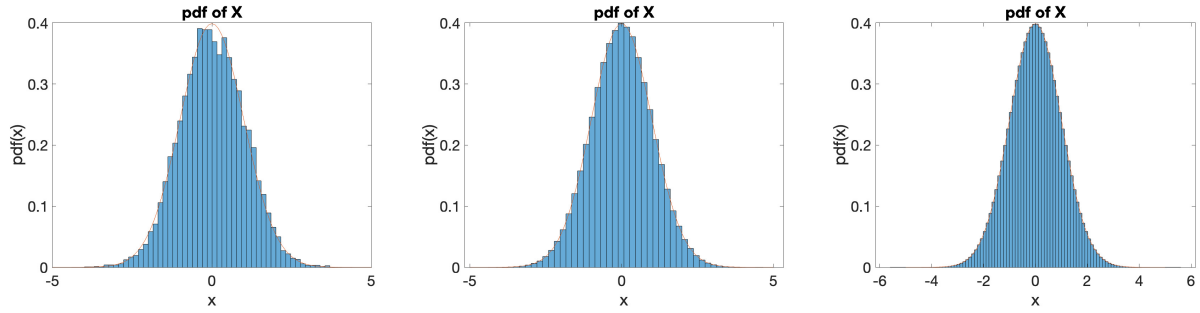
Figure 6: (Theoretical values given with a straight line)

The PDF of the Gaussian Distribution: We approximate the PDFs in figures 7 and 8. The red line is the theoretical pdf of the distribution while the bins are the estimates using the samples generated with this distribution. As before, with increasing number of samples, we observe convergence to the theoretical pdf, and increasing number of bins allow us to see more information about the data (increasing resolution).



(a) 1000 samples, 10 bins (b) 1000 samples, 50 bins (c) 1000 samples, 100 bins

Figure 7: Changing the number of bins



(a) 10^4 samples, 50 bins (b) 10^6 samples, 50 bins (c) 10^8 samples, 100 bins

Figure 8: Changing the number of samples

1.2 Stochastic Processes

(1) The time average of a signal is the averaged value of a realisation of a signal over a time interval. The ensemble average on the other hand is the averaged quantity of many identical signals at a certain time. We investigate these properties with the given MATLAB functions. In figure 9 are the ensemble means and standard deviations of the given processes using M = 100 members of the ensemble each of length N = 100.

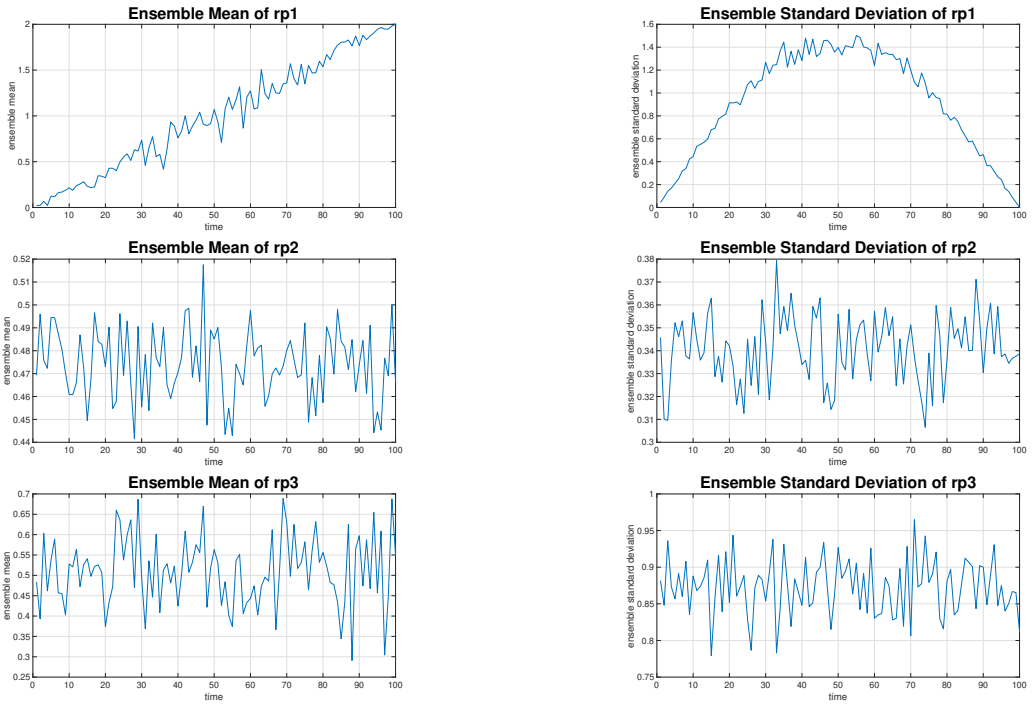


Figure 1: Ensemble means and standard deviations of the given random processes

We observe that the random process rp1 is not stationary as its mean and variance change over time. Processes rp2 and rp3 on the other hand seem to be stationary as their means and standard deviations cluster around the same value with a certain randomness regardless of time.

(2) Time Averages: We know generate M=4 realisations of length N=1000 for each process, and calculate the mean and σ of each realisation. Results are given in the tables below.

"A process is said to be ergodic if its statistical properties can be deduced from a single, sufficiently long, random sample of the process. [1]. Although we've stated that rp1 isn't a stationary process, it is along with

rp3 seem to be an ergodic process as their statistical properties stay consistent along different realizations. This means we can deduce their statistical properties from one of those realizations. However, looking at the results of Table 2, we cannot say that rp2 is ergodic as its statistics change erratically in different realizations.

(rp1) Realisation	Mean	Standard Deviation
1	9.9983	5.8475
2	10.0053	5.8627
3	10.0799	5.8829
4	10.0408	5.8775

(rp2) Realisation	Mean	Standard Deviation
1	0.4306	0.2408
2	0.2156	0.2768
3	0.9795	0.1798
4	0.4920	0.0853

(rp3) Realisation	Mean	Standard Deviation
1	0.4911	0.8657
2	0.4861	0.8538
3	0.5219	0.8712
4	0.5268	0.8930

(3) We calculate the theoretical means and variances of each of the random processes. All random variables $\eta[n]$ in the equations below are distributed with $\mathcal{U}(0,1)$ We know from 1.1 that expectation of this uniform distribution is 0.5, and its variance is $\frac{1}{12}$.

rp1 mean:

$$x[n] = (\eta[n] - 0.5) \cdot 5sin(\frac{n\pi}{N}) + 0.02n \tag{1}$$

$$= \eta[n] \cdot 5sin(\frac{n\pi}{N}) - 2.5sin(\frac{n\pi}{N}) + 0.02n \tag{2}$$

$$m = \mathbb{E}\{x[n]\} = 5sin(\frac{n\pi}{N}) \cdot \mathbb{E}\{\eta[n]\} - 2.5sin(\frac{n\pi}{N}) + 0.02n \tag{3}$$

$$= 5sin(\frac{n\pi}{N}) \cdot 0.5 - 2.5sin(\frac{n\pi}{N}) + 0.02n \tag{4}$$

$$m = 0.02n \quad \textbf{(mean)} \tag{5}$$

One should be careful at this point. The theoretical mean we get here is the expected value at a certain time instant. For example at n = 100, we expect the value to be 2. This result agrees with the noisy straight line graph we had in 1.2.1 for rp1 ensemble mean. It also agrees with the sample mean values we got in 1.2.2 for rp1. We show this by calculating the average of a straight with the equation $x = 0.02n$ for n = [1, 1000].

$$\frac{0.02 \cdot 1 + 0.02 \cdot 1000}{2} \approx 10$$

rp1 σ:

$$VAR\{x[n]\} = VAR\{\eta[n] \cdot 5sin(\frac{n\pi}{N})\} + 0 = 5^2 \cdot sin^2(\frac{n\pi}{N}) \cdot VAR\{\eta[n]\} \tag{6}$$

$$\sigma^2 = 25sin^2(\frac{n\pi}{N}) \cdot \frac{1}{12} \tag{7}$$

$$\sigma = \frac{5\sqrt{3}}{6} \cdot sin(\frac{n\pi}{N}) \quad \textbf{(standard deviation)} \tag{8}$$

The result for σ again agrees with the ensemble standard deviation graph we had for rp1 in 1.2.1. For example, for N = 100 and n = 50, we have $\sigma = 1.443$ according to equation 8. We approximately have the same result in the related graph in 1.2.1 at time instant n = 50. Again, this value is related to the variance at a certain time instant. Since this is a non-stationary process, the calculation of the sample standard deviation (time average) for a single realization in 1.2.2 gives something totally different.

rp2 mean:

$$x[n] = (\eta_1[n] - 0.5) \cdot \eta_2[n] + \eta_3[n] = \eta_1[n] \cdot \eta_2[n] - 0.5 \cdot \eta_2[n] + \eta_3[n] \tag{9}$$

$$m = \mathbb{E}\{x[n]\} = \mathbb{E}\{\eta_1[n] \cdot \eta_2[n]\} - 0.5 \cdot \mathbb{E}\{\eta_2[n]\} + \mathbb{E}\{\eta_3[n]\} \tag{10}$$

Then we can calculate

$$\mathbb{E}\{\eta_1[n] \cdot \eta_2[n]\} = \int_0^1 \int_0^1 \eta_1 \cdot \eta_2 \cdot d\eta_1 d\eta_2 = 0.25 \tag{11}$$

Then

$$m = \mathbb{E}\{x[n]\} = 0.25 - 0.5 \cdot 0.5 + 0.5 \tag{12}$$

$$m = 0.5 \quad \textbf{(mean)} \tag{13}$$

The theoretical value for the mean agrees with the graph of its ensemble mean which clusters around this value. The sample mean values are also close to this value but not very consistent.

rp2 σ:

$$VAR\{x[n]\} = VAR\{\eta_1[n] \cdot \eta_2[n]\} + VAR\{0.5 \cdot \eta_2[n]\} + VAR\{\eta_3[n]\} \tag{14}$$

$$= \mathbb{E}\{\eta_1^2 \cdot \eta_2^2\} - \mathbb{E}^2\{\eta_1 \cdot \eta_2\} + 0.25 \cdot \frac{1}{12} + \frac{1}{12} \tag{15}$$

We calculate

$$\mathbb{E}\{\eta_1^2 \cdot \eta_2^2\} = \int_0^1 \int_0^1 \eta_1^2 \cdot \eta_2^2 \cdot d\eta_1 d\eta_2 = \frac{1}{9} \tag{16}$$

Then

$$\sigma^2 = \frac{1}{9} - \frac{1}{16} + \frac{1.25}{12} \tag{17}$$

$$\sigma \approx 0.39 \quad \textbf{(standard deviation)} \tag{18}$$

Theoretical σ agrees with its ensemble standard deviation graph as the values are close to the theoretical value, although they seem to cluster around 0.35. The sample standard deviation is, however, lower.

rp3 mean:

$$x[n] = (\eta[n] - 0.5) \cdot 3 + 0.5 = 3 \cdot \eta[n] - 1 \tag{19}$$

$$m = \mathbb{E}\{x[n]\} = 3 \cdot \mathbb{E}\{\eta[n]\} - 1 = 3 \cdot 0.5 - 1 \tag{20}$$

$$m = 0.5 \quad \textbf{(mean)} \tag{21}$$

The theoretical value for the mean agrees with the graph of its ensemble mean which clusters around this value. The sample mean values are also consistent with this value.

rp3 σ:

$$VAR\{x[n]\} = VAR\{3 \cdot \eta[n]\} = 9 \cdot VAR\{\eta[n]\} \tag{22}$$

$$\sigma^2 = 9 \cdot \frac{1}{12} \tag{23}$$

$$\sigma \approx 0.87 \quad \textbf{(standard deviation)} \tag{24}$$

The theoretical σ also agrees with its ensemble standard deviation graph as it clusters around the theoretical value. The sample standard deviation is also consistent with the theoretical value.

Ensemble averages average the statistics of different realisations of a process at particular time instants. Time averages, however, average different samples of a single realisation over a time interval. With this analysis, we support the previous claims on the ergodicity and the stationarity of the given random processes.

1.3 Estimation of probability distributions

(1) We create a function named `pdf.m` that estimates the pdf of a random variable using its samples and histograms. Matlab code given in Appendix [A]. We test it first for a Gaussian pdf.

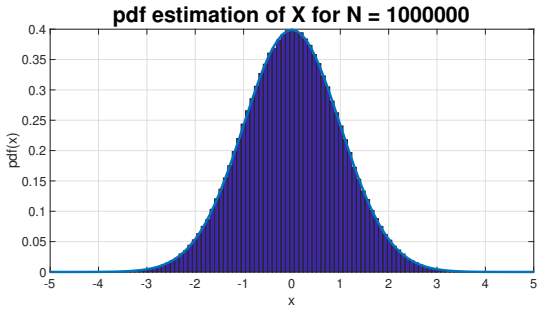


Figure 2: `x = randn(1,1000000)` with 100 bins

(2) Among the three processes in part 1.2, rp3 is the only one that is both stationary and ergodic. We estimate its pdf with our function and observe that as data N increases, the estimated pdf converges to the theoretical pdf. In fact when the number of data samples is 10^8, they're almost identical.

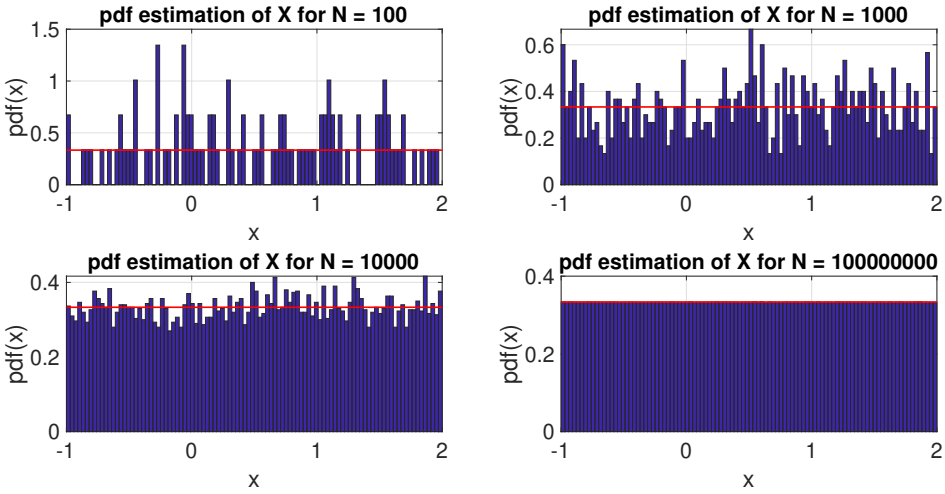


Figure 3: PDF estimations of rp3 (Theoretical pdf given in red)

(3) We cannot estimate the pdf of a non-stationary process with the function we declared as `pdf.m` since the pdf of the process changes with time. Our function would only give the probability of selecting a certain element from the pool of all samples generated in the given time.

For example, if the mean of a 1000-sample long process changes from 0 to 1 at sample point n=500, this would mean that the pdf of the process is different for the first 500 samples and last 500 samples and should be analyzed accordingly. Meaning, we would use our function `pdf()` first for only the first 500 data samples and generate a pdf; then we would use the last 500 samples and generate another pdf for the second half of the process. If the mean is the only statistics that has changed for the process, we would observe that the pdf gets shifted towards the new mean.

2 Linear stochastic modelling

2.1 ACF of uncorrelated and correlated sequences

(1) In figure 1 we estimate the autocorrelation function of a 1000-sample realisation of WGN and observe the results for lag τ between -999 and 999. The graph emulates a Dirac function at 0 since different realisations are independent, and thus uncorrelated, from each other. The auto correlation values are very close to 0 except for at $\tau = 0$. However, we also recognize that the AC values start to diverge for large values of $|\tau|$ due to signal being windowed. We also note the symmetric property of the function across $\tau = 0$.

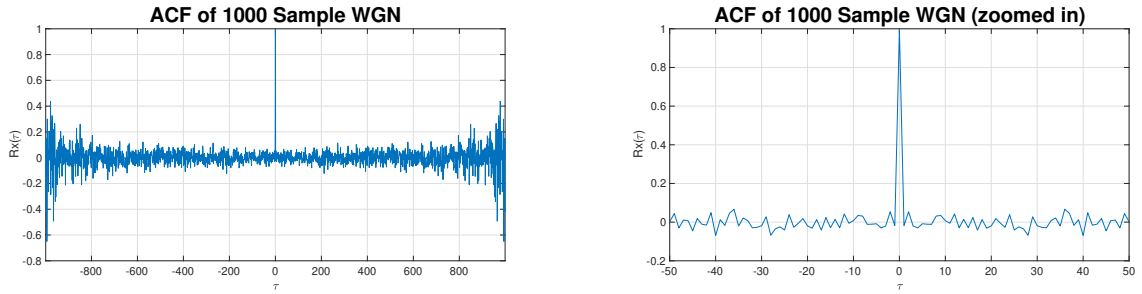


Figure 1: ACF estimation of 1000 sample WGN

(2) If we zoom into the values of $|\tau| < 50$, we get the second plot in Figure 1. We observe a very small divergence from the theoretical auto-correlation value of 0 for $\tau \neq 0$. However, for large τ, after $|\tau| > 500$, we observe a larger and larger divergence of the function from its theoretical value.

(3) The MATLAB function we use to estimate the autocorrelation of the WGN realisation calculates the values according to the equation 1:

$$\widehat{R}_X(\tau) = \frac{1}{N - |\tau|} \sum_{n=0}^{N-|\tau|-1} x[n]x[n+\tau], \qquad \tau = -N+1, ..., N-1 \tag{1}$$

From the equation, we see that as the value of $|\tau|$ increases, number of samples we use to estimate the auto-correlation function decreases. This causes the high oscillations and the divergence of the values and the low accuracy of the estimates for large τ. We conclude that the estimates aren't reliable for large $|\tau|$ and looking at the plots in Figure 1, we suggest an empirical bound of $|\tau| < 500$ for reliable estimation of the autocorrelation function of this 1000-sample realisation of WGN.

(4) We filter a 1000-sample WGN with a moving average (MA) filter with unit coefficients of order 9 and plot the ACF estimate of the filtered signal in Figure 2. The Dirac function we had around $\tau = 0$ turns into a triangular shape after the filter is applied. The applied filter can be viewed by equation 2 or the generated signal by equation 3,

$$Y(z) = \frac{1 + z^{-}1 + ... + z^{-}8}{1} X(z) \tag{2}$$

$$y[n] = x[n] + x[n-1] + x[n-2] + ... + x[n-8] \tag{3}$$

where x[n] represents WGN. The filtered response of the process depends on 9 different realisations of WGN. We know that WGN realisations correlate with each other and with each other only – different realisations do not correlate – so after taking the auto-correlation estimate of the response, we see at $\tau = 0$ the ACF value is equal to the order number as expected. At $|\tau| = 1$, one less realization is correlated and value decreases to 8, and then to 7 and so on until $|\tau|$ reaches the order number and none of the realisations start to correlate after that much lag. The situation explained above can be seen in figure 3 where the filter order numbers are changed into 3 and 15 respectively.

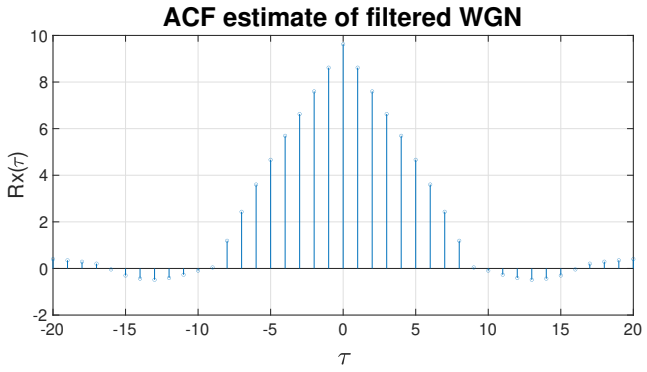
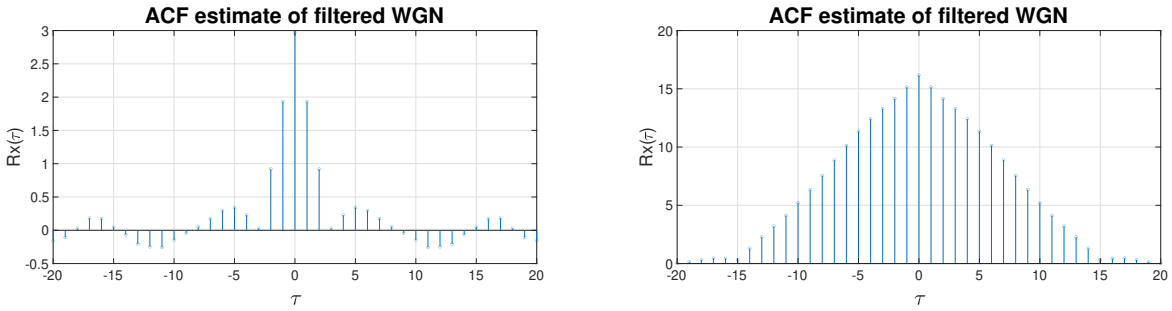


Figure 2: ACF Estimate of Filtered WGN with Unit MA Coefficients of Order 9

From equation 3, we see that a sample in the filtered response is the unweighted sum of 9 (order) realisations (1 current, 8 previous) of the input noise signal. So dividing the output of the filtered response by the order number would give us the (local) sample mean. (i.e dividing the sum of nine samples by 9 would give us the sample mean of those nine samples).



(a) Filter of Order 3

(b) Filter of Order 15

Figure 3: ACF estimates of the filtered signal with different order MA filters

(5) Given

$$R_Y(\tau) = R_X(\tau) * R_h(\tau) \tag{4}$$

If X_n is an uncorrelated process, its auto-correlation function would give a scaled Dirac function at $\tau = 0$. For this reason R_Y would represent the convolution of R_h with a Dirac function, which is seen as identity for the convolution operation. So $R_Y(\tau)$ would be equal to a scaled version of $R_h(\tau)$ (Scaled with σ_X^2).

2.2 Cross-correlation function

(1) From the previous part, we know that process X is an uncorrelated stochastic process, and for this reason $R_X(\tau)$ is a Dirac function around $\tau = 0$. For the same reasons mentioned in 2.1.5, the cross-correlation function of X and Y will be a scaled version of the filter's impulse response according to equation 5.

$$R_{XY}(\tau) = h(\tau) * R_X(\tau) \tag{5}$$

We know that the filter has order 9 moving average unit coefficients, and as expected, we observe that R_{XY} has the same shape as the impulse response. CCF estimate for sequences $\mathbf{x}$ and $\mathbf{y}$ is plotted alongside the ACF estimate of $\mathbf{x}$ in Figure 4.

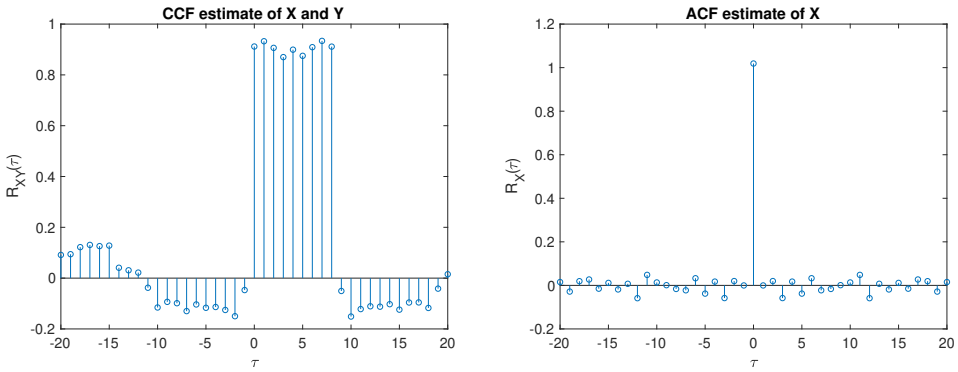
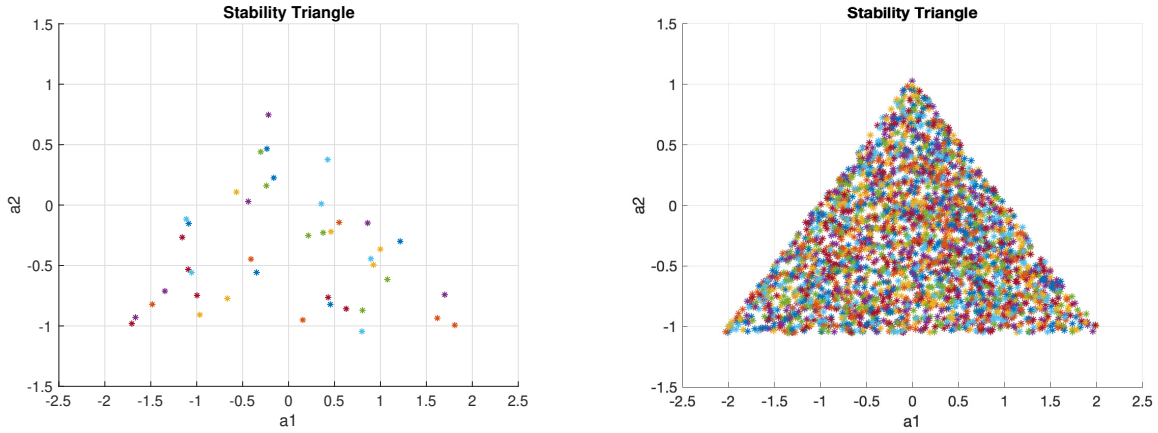


Figure 4: Cross-correlation and Autocorrelation estimates

(2) We know that equation 5 holds. We also know that the autocorrelation function of an uncorrelated stochastic process such as WGN is a Dirac function. Lastly, we know that convolving a function with a Dirac function only scales a that function according to the power of the Dirac. Using these facts, we can identify a system's impulse response by providing a known uncorrelated stochastic process in the input of the system and then by cross-correlating the input with the output of the system as in the example above. So, we can also conclude that for the example above, the order of the filter controls the duration of the impulses in the cross-correlation function.

2.3 Autoregressive modelling

(1) In Figure 5, we only plot a_1 and a_2 values that give a stable filter response. Looking at the plots, we observe that those values that are inside the so called "Stability Triangle" are the ones that give a stable response. The triangle is better visualised in figure (b) where 10000 pairs of random a_1 and a_2 were tested.



(a) 100 pairs

(b) 10000 pair

Figure 5: Stability Triangle

To support this empirical result with a theoretical justification, we do the following calculation for stability conditions. We know the filter response to be:

$$x[n] = a_1 x[n-1] + a_2 x[n-2] + w[n], \qquad w[n] \backsim \mathcal{N}(0,1) \tag{6}$$

$$X(z) = (a_1 z^{-1} + a_2 z^{-2}) X(z) + W(z) \tag{7}$$

$$H(z) = \frac{X(z)}{W(z)} = \frac{1}{1 - a_1 z^{-1} - a_2 z^{-2}} = \frac{z^2}{z^2 - a_1 z - a_2} \tag{8}$$

For stability, we know that the poles of this filter should be inside the unit circle. So,

$$| z_{1,2} |= \left| \frac{a_1 \pm \sqrt{a_1^2 + 4a_2}}{2} \right| < 1 \tag{9}$$

$$-2 < a_1 \pm \sqrt{a_1^2 + 4a_2} < 2 \tag{10}$$

Then we can do,

$$a_1 + \sqrt{a_1^2 + 4a_2} < 2 \tag{11}$$

$$a_1^2 + 4a_2 < (2 - a_1)^2 \Rightarrow a_1^2 + 4a_2 < 4 - 4a_1 + a_1^2 \tag{12}$$

$$a_1 + a_2 < 1 \quad \textbf{(Condition 1)} \tag{13}$$

We also have

$$-2 < a_1 - \sqrt{a_1^2 + 4a_2} \tag{14}$$

$$a_1^2 + 4a_2 < (a_1 + 2)^2 \tag{15}$$

$$a_2 - a_1 < 1 \quad \textbf{(Condition 2)} \tag{16}$$

We can write the denominator of the right hand side of equation 8 as $(z - z_1)(z - z_2)$. Then $-a_2 = z_1 \cdot z_2$. We know from the stability conditions above that $|z_{1,2}| < 1$. Then we have $|a_2| < 1$. **(Condition 3)**

(2) We plot the ACF of the real world sunspot series. Without removing the DC level in the signal, we get the ACF plots in Figure 6 for different data lengths N = 5, 20, 250. We observe that the shapes are similar for different data lengths with same periodicity as the original time series. However, as the data length increases the shape of the ACF becomes more clear. Due to the DC level, however, we cannot view the true periodic behaviour of the signal well and observe an overall triangular shape.

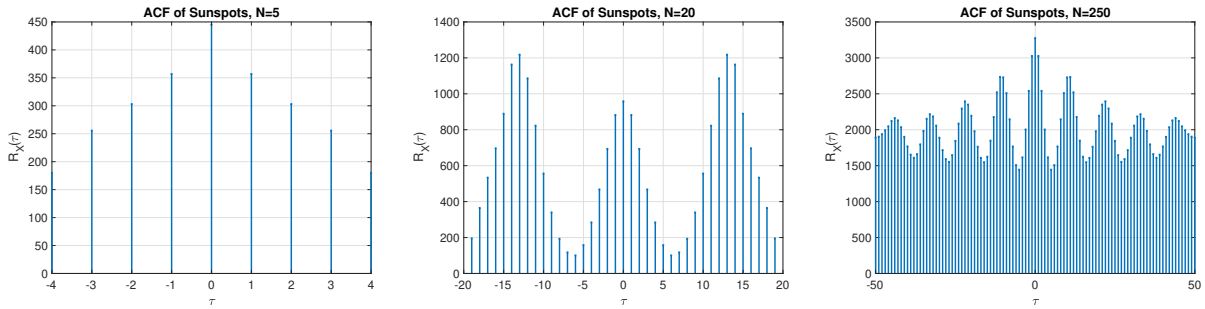


Figure 6: Auto-Correlation Function of the Sunspot Time Series for Different Data Lengths

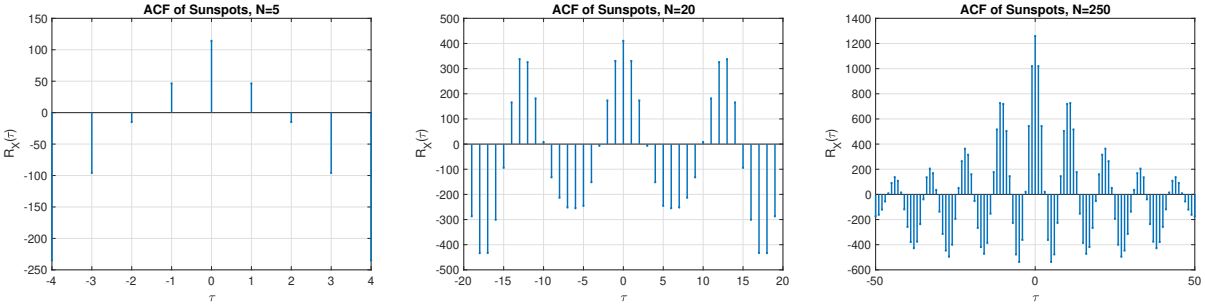


Figure 7: ACF for the zero-mean (centered) version of the sunspot series

After removing the DC level from the signal, we get the ACF plots in Figure 7 which gives a better representation of the signal and its periodicity showing how detrimental the DC level is for processing a signal. We observe a psuedo-periodic ACF for the sunspot series.

(3) We determine the most likely model order for the sunspot time series using the Yule-Walker equations.

$a_1 = [0.9295]$
$\boldsymbol{a_2 = [1.4740, -0.5857]}$
$a_3 = [1.5492, -0.7750, 0.1284]$
$a_4 = [1.5167, -0.5788, -0.2638, 0.2532]$
$a_5 = [1.4773, -0.5377, -0.1739, 0.0174, 0.1555]$
$a_6 = [1.4373, -0.5422, -0.1291, 0.1558, -0.2248, 0.2574]$
$a_7 = [1.3669, -0.4807, -0.1718, 0.1912, -0.0764, -0.1359, 0.2736]$
$a_8 = [1.3016, -0.4483, -0.1535, 0.1456, -0.0355, -0.0212, -0.0523, 0.2384]$
$a_9 = [1.2615, -0.4395, -0.1500, 0.1515, -0.0599, 0.0046, 0.0230, 0.0197, 0.1680]$
$a_{10} = [1.2573, -0.4400, -0.1506, 0.1514, -0.0584, 0.0007, 0.0268, 0.0308, 0.1362, 0.0252]$

The last coefficient in a certain model with order k is also represented as a_{kk} (kth coefficient in an order k model) and is called the **partial autocorrelation function (PAC)**, and we measure the significance of a_{kk} by comparing it to a threshold. We observe that starting from $k = 3$ PAC is very small and the first two coefficients doesn't seem to change much, indicating most likely order to be 2 and that the higher orders may be over-fitting. The sunspot model is then:

$$x[n] = 1.4740x[n-1] - 0.5857x[n-2] + w[n] \tag{17}$$

As we do not wish to model the DC offset but the stochastic component of the signal, we repeat the procedure for standardised zero mean and unit variance signal and get:

$a_1 = [0.8212]$
$\boldsymbol{a_2 = [1.3783, -0.6783]}$
$a_3 = [1.2953, -0.5097, -0.1223]$
$a_4 = [1.3011, -0.4856, -0.1836, 0.0473]$
$a_5 = [1.3018, -0.4885, -0.1912, 0.0676, -0.0156]$
$a_6 = [1.3044, -0.4994, -0.1602, 0.1469, -0.2269, 0.1623]$
$a_7 = [1.2760, -0.4597, -0.1859, 0.1749, -0.1394, -0.0661, 0.1751]$
$a_8 = [1.2361, -0.4447, -0.1541, 0.1351, -0.0971, 0.0385, -0.1154, 0.2276]$
$a_9 = [1.1959, -0.4243, -0.1609, 0.1523, -0.1210, 0.0658, -0.0368, 0.0093, 0.1766]$
$a_{10} = [1.1952, -0.4243, -0.1608, 0.1520, -0.1205, 0.0652, -0.0362, 0.0109, 0.1721, 0.0038]$

The coefficients are only slightly different. (Note that the first two coefficients are slightly smaller). Again the most likely order is 2 with a slight change in coefficients. Model for the standardised series is then:

$$x[n] = 1.3783x[n-1] - 0.6783x[n-2] + w[n] \tag{18}$$

(4) We estimate the data with different AR model orders. Comparing the estimates with the actual data, we get the following performance in Figure 8 for different criterion. It's easy to see that the plots take a sharp turn after model order 2. Therefore, we determine the model order to be 2 which is the simplest model order adequate for the data.

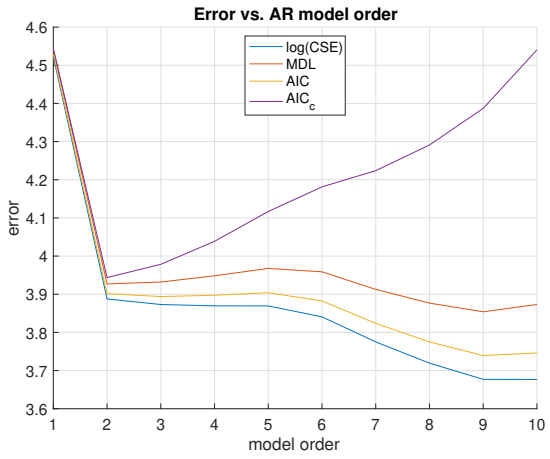


Figure 8: AR model order selection according to different principles

(5) We use three different models (order = 1,2,10) to predict the sunspot time series using different prediction horizons m, where the prediction for different m values is given by the following formula.

$$\hat{x}[n+m] = a_1 x[n+m-1] + ... + a_p x[n+m-p] \tag{19}$$

So for example, a prediction with AR(2) model where m = 5 can be made by following the algorithm:

$$\hat{x}[n+5] = a_1 \hat{x}[n+4] + a_2 \hat{x}[n+3] \tag{20}$$

$$\hat{x}[n+4] = a_1 \hat{x}[n+3] + a_2 \hat{x}[n+2] \tag{21}$$

$$\hat{x}[n+3] = a_1 \hat{x}[n+2] + a_2 \hat{x}[n+1] \tag{22}$$

$$\hat{x}[n+2] = a_1 \hat{x}[n+1] + a_2 x[n] \tag{23}$$

$$\hat{x}[n+1] = a_1 x[n] + a_2 x[n-1] \tag{24}$$

In Figure 9, we can see that as the prediction horizon increases, the error in the predictions increase as well. (An exception to this is going from prediction with AR(1) for m=5 to m=10. This is most probably particular to this data set). The reason for the increase in error is that as we increase the prediction horizon, instead of using actual values to make a prediction, we use predicted values to make another prediction. As the horizon increases this bootstrapping length also increases and the predictions become less reliable.

The AR(10) model is more complex and has more capacity than the other two models. For this reason, we observe that the predictions with this model is closer to the actual values. We also observe as the prediction horizon increases, models AR(1) and AR(2) predict similar values to each other which are smoother due to their limited capacity. We even observe that the MSE for AR(1) is slightly less than that of AR(2) for the first time when the prediction horizon is 10.

As the prediction horizon increases from 5 to 10, we observe the variance of error to stabilize for AR(2) model. While AR(10) model seems to perform better in predicting the actual values, we should keep in mind that these predictions are in-sample predictions, i.e. the model is trained with the samples that were predicted. The true performance of the models can be evaluated when we forecast data points where the models don't have any idea what the true values are. Because AR(10) is too complex for this particular set of data, it won't generalize well out of sample (over modelling), but we would expect AR(2) to perform better since it has enough capacity for representation with enough simplicity for good generalization.

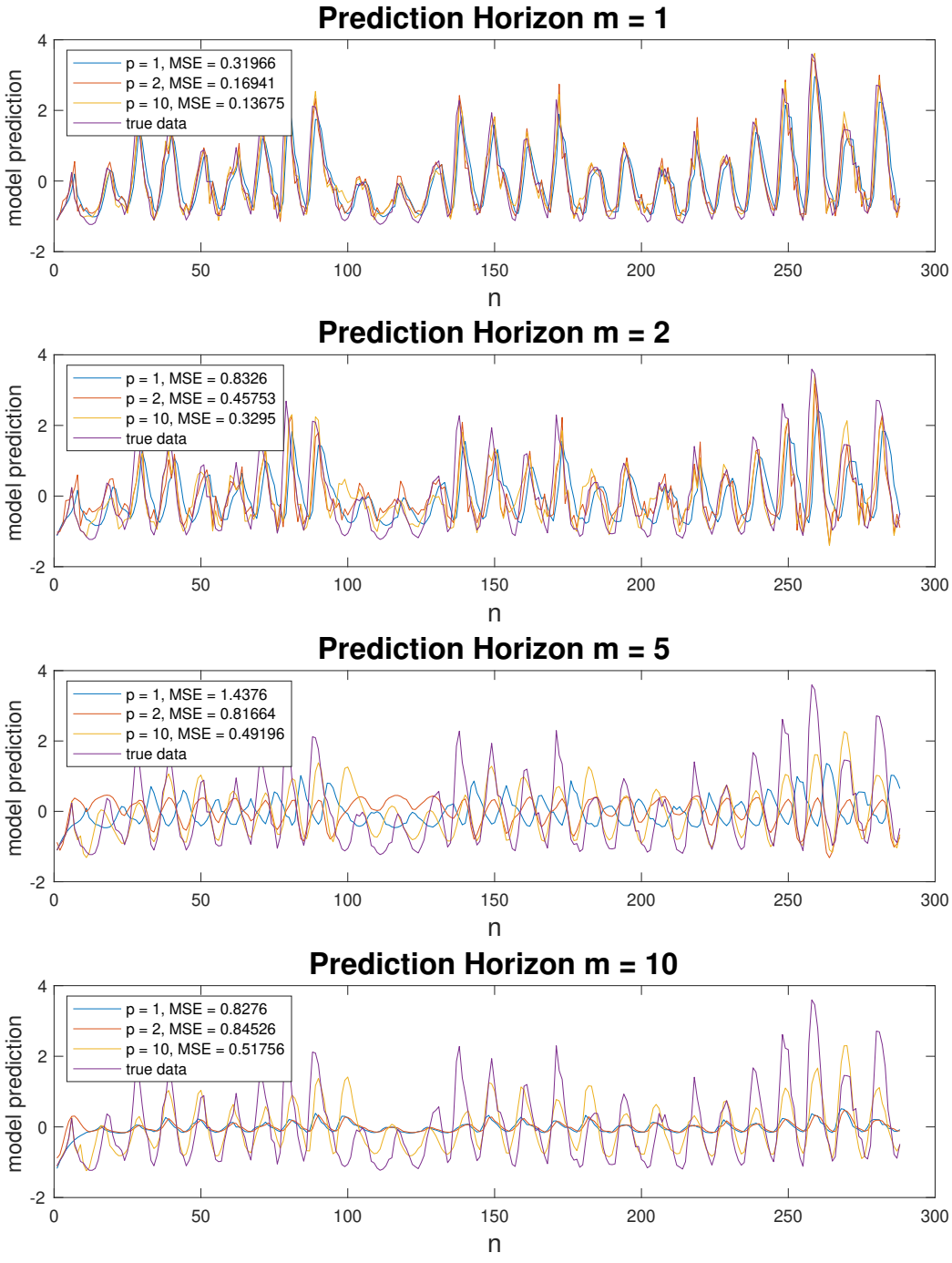


Figure 9: Predictions with different Models with different prediction horizons (Mean squared error of each prediction is given in legends)

2.4 Cramer-Rao Lower Bound

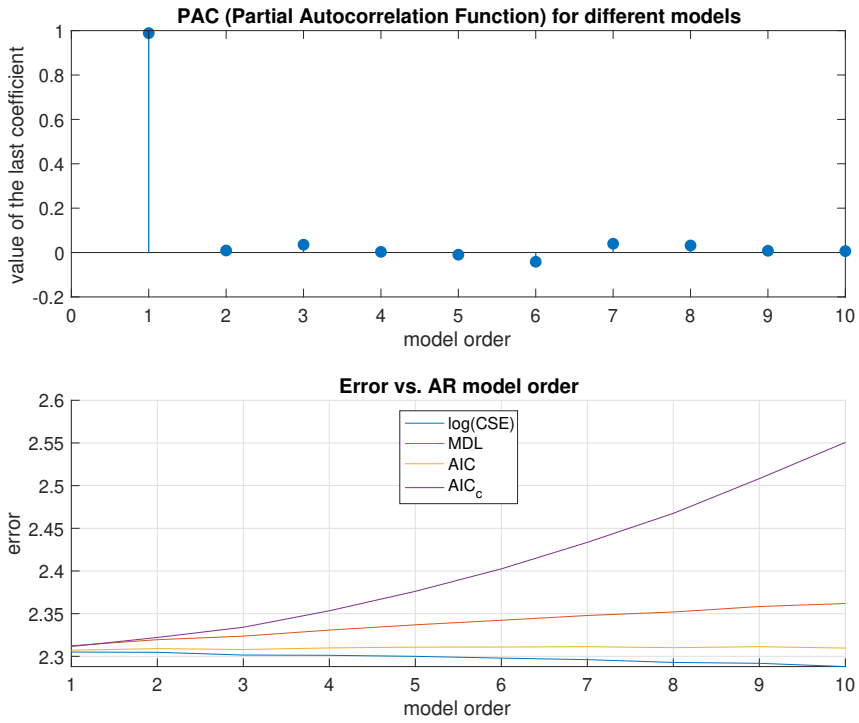


Figure 10: PAC vs. model order and Error vs. model order

(a) When we calculate the Partial Auto-correlation functions (i.e. the last coefficient of an AR model) of different AR models to represent the given NASDAQ data set, we get the first graph in Figure 10. We note that after model order 1, the coefficients become very small so we determine model order 1 to be sufficient to represent this data. The second graph of Figure 10 supports this claim as the error plots don't show any significant decrease, and even increase, after model order 1.

(b) In part (a) we determined the order of the AR model to be 1. Here we have its power spectrum as

$$\hat{P}_X(f;\theta) = \frac{\hat{\sigma}^2}{\mid 1 - \hat{a}_1 e^{-j2\pi f} \mid^2} \tag{25}$$

where $\theta = [a_1, \sigma^2]$. Then,

$$\ln\left[\hat{P}_X(f;\theta)\right] = \ln\left[\hat{\sigma}^2\right] - 2\ln\left[1 - \hat{a}_1 e^{-j2\pi f}\right] \tag{26}$$

Given the Fisher information matrix to be

$$[\mathbf{I}(\theta)]_{ij} = \frac{N}{2}\int_{-\frac{1}{2}}^{\frac{1}{2}} \frac{\partial \ln[\hat{P}_X(f;\theta)]}{\partial\theta_i} \cdot \frac{\partial \ln[\hat{P}_X(f;\theta)]}{\partial\theta_j} df \tag{27}$$

we compute $[\mathbf{I}(\theta)]_{22}$ by doing the following calculation.

$$[\mathbf{I}(\theta)]_{22} = \frac{\partial \ln[\hat{P}_X(f;\theta)]}{\partial\sigma^2} = \frac{1}{\sigma^2} \Longrightarrow \frac{N}{2}\int_{-\frac{1}{2}}^{\frac{1}{2}} \frac{1}{\sigma^4} df = \frac{N}{2\sigma^4} \tag{28}$$

Since we're given the rest of the elements of the matrix, we state

$$[\mathbf{I}(\theta)] = \begin{bmatrix} \frac{Nr_{xx}(0)}{\sigma^2} & 0 \\ 0 & \frac{N}{2\sigma^4} \end{bmatrix} \tag{29}$$

(c) We can get an expression for the variance of the AR(1) process by doing the following calculation.

$$x[n] = a_1 x[n-1] + w[n] \tag{30}$$

multiplying both sides by x[n-k] we get $r_{xx}(k) = a_1 r_{xx}(k-1)$ for $k > 0$. Then

$$r_{xx}(1) = a_1 r_{xx}(0) \tag{31}$$

and dividing by $r_{xx}(0)$

$$\rho(1) = a_1 \tag{32}$$

And for k=0

$$r_{xx}(0) = a_1 r_{xx}(1) + \sigma_w^2 \tag{33}$$

$$1 = a_1 \rho(1) + \frac{\sigma_w^2}{\sigma_x^2} \tag{34}$$

$$\sigma_x^2 = \frac{\sigma_w^2}{1 - a_1^2} \implies \frac{\sigma_w^2}{r_{xx}(0)} = 1 - a_1^2 \tag{35}$$

Now we know

$$var(\hat{\theta}_i) \geq [\mathbf{I}^{-1}(\theta)]_{ii}, \quad \text{and} \quad [\mathbf{I}^{-1}(\theta)] = \begin{bmatrix} \frac{\sigma^2}{Nr_{xx}(0)} & 0 \\ 0 & \frac{2\sigma^4}{N} \end{bmatrix} \tag{36}$$

So,

$$var(\hat{\theta}_1) = var(\hat{a}_1) \geq \frac{\sigma^2}{Nr_{xx}(0)} = \frac{1}{N}(1 - a_1^2) \tag{37}$$

$$var(\hat{\theta}_2) = var(\hat{\sigma}^2) \geq \frac{2\sigma^4}{N} \tag{38}$$

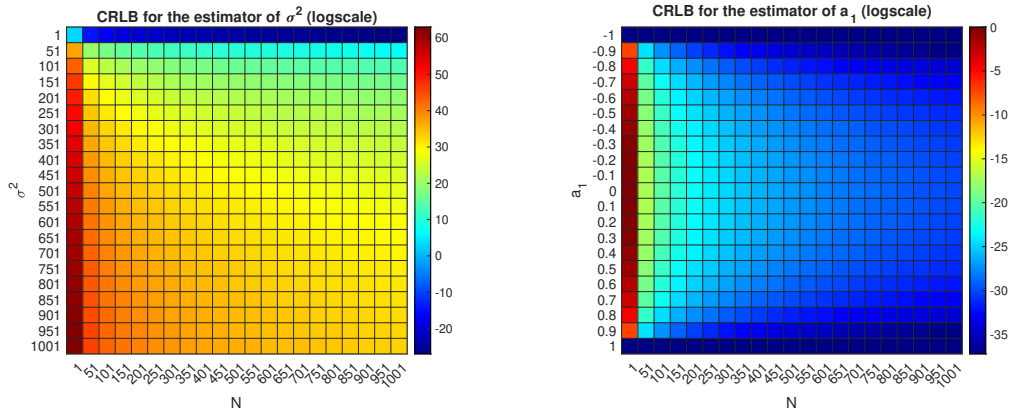


Figure 11: Cramer-Rao Lower Bound for $\hat{\sigma}^2$ and $\hat{a}_1$

In Figure 11, we plot the Cramer-Rao Lower Bound for the relevant estimators. The AR(1) model we had for the financial data set with N = 924 (number of data points) investigated in part (a) had its coefficient $a_1 = 0.9887$. Considering this is the true value of the coefficient for this model, we find the CRLB on its estimator from equation 37 as

$$var(\hat{a}_1) \geq \frac{1}{N}(1 - a_1^2) = \frac{1 - 0.9887^2}{924} = 2.4 * 10^{-5} \tag{39}$$

We see from equation 37 that as a_1 approaches unity, the variance of its estimator goes to zero. With a_1 approaching unity, the pole of the characteristic function approaches the unit circle, right at the boundary of being unstable. For this reason, signals that are not correlated or at the edge of being unstable would have a_1 most probably very close to 1 with very little variability in its estimation.

(d) From the CRLB, we know that

$$var(\hat{P}_X(f;\theta)) \geq \frac{\partial \hat{P}_X(f;\theta)}{\partial \theta}^T \mathbf{I}^{-1}(\theta) \frac{\partial \hat{P}_X(f;\theta)}{\partial \theta} \tag{40}$$

From equation 25, we first compute

$$\frac{\partial \hat{P}_X(f;\theta)}{\partial a_1} = \frac{\partial}{\partial a_1} \frac{\sigma^2}{(1 - a_1\cos(2\pi f))^2 + (a_1 \sin(2\pi f))^2} \tag{41}$$

$$= \frac{2\sigma^2(\cos(2\pi f) - a_1)}{\mid A(f) \mid^2} = \frac{2\sigma^2(1 - Re(A(f)) - a_1^2)}{a_1 \mid A(f) \mid^2} \tag{42}$$

$$\frac{\partial \hat{P}_X(f;\theta)}{\partial \sigma^2} = \frac{1}{\mid A(f) \mid^2} \tag{43}$$

where $A(f) = 1 - a_1 e^{-j2\pi f}$. We know $\mathbf{I}^{-1}(\theta)$ from equation 36. Then we compute the bound as

$$var(\hat{P}_X(f;\theta)) \geq \begin{bmatrix} \frac{2\sigma^2(1-Re(A(f))-a_1^2)}{a_1|A(f)|^2} & \frac{1}{|A(f)|^2} \end{bmatrix} \begin{bmatrix} \frac{\sigma^2}{Nr_{xx}(0)} & 0 \\ 0 & \frac{2\sigma^4}{N} \end{bmatrix} \begin{bmatrix} \frac{2\sigma^2(1-Re(A(f))-a_1^2)}{a_1|A(f)|^2} \\ \frac{1}{|A(f)|^2} \end{bmatrix} \tag{44}$$

$$= \frac{\sigma^2}{Nr_{xx}(0)} \cdot \left[\frac{2\sigma^2(1 - Re(A(f)) - a_1^2)}{a_1 \mid A(f) \mid^2}\right]^2 + \frac{2\sigma^4}{N} \cdot \left[\frac{1}{\mid A(f) \mid^2}\right]^2 \tag{45}$$

2.5 Real world signals: ECG from iAmp experiment

(a) Using the RRI signal for unconstrained breathing, we obtain the heart rate signal $h[n]$. Then to obtain a smoother estimate of the heart rate, we average every 10 samples in our signal with a weighting α and get $\hat{h}[n]$. The probability density estimates of the original and averaged heart rate is given in Figure 12 for different α values.

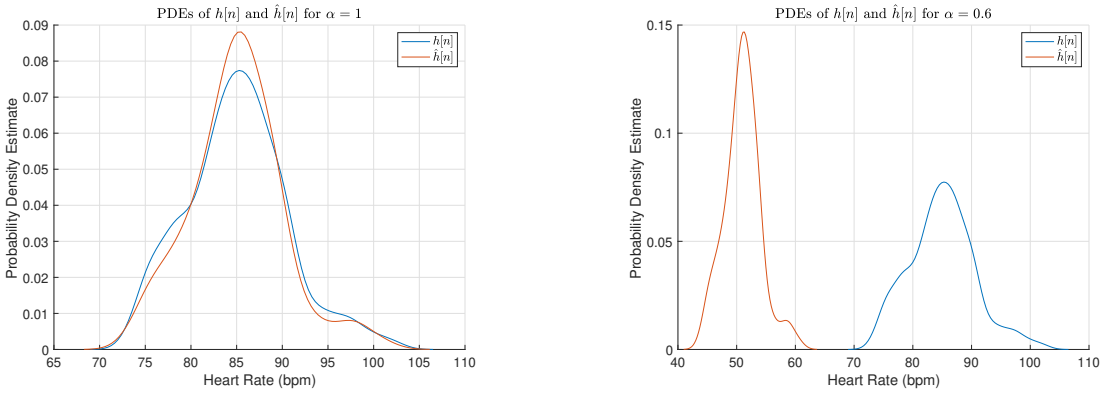


Figure 12: Probability Density Estimates for Original and Averaged Heart Rates with different α

(b) When α is 1, we observe that the shapes of the PDEs are quite similar to each other. The PDE of the averaged heart rates is slightly narrower and smoother than that of the the original. When α is 0.6, we average the heart rate samples in a way that it decreases their true average. For this reason, the PDE plot for $\hat{h}[n]$ is shifted to left, with the gap between the maximum and the minimum values decreasing by the same ratio. This also causes the plot to be even narrower as it can be seen in the second graph in Figure 12.

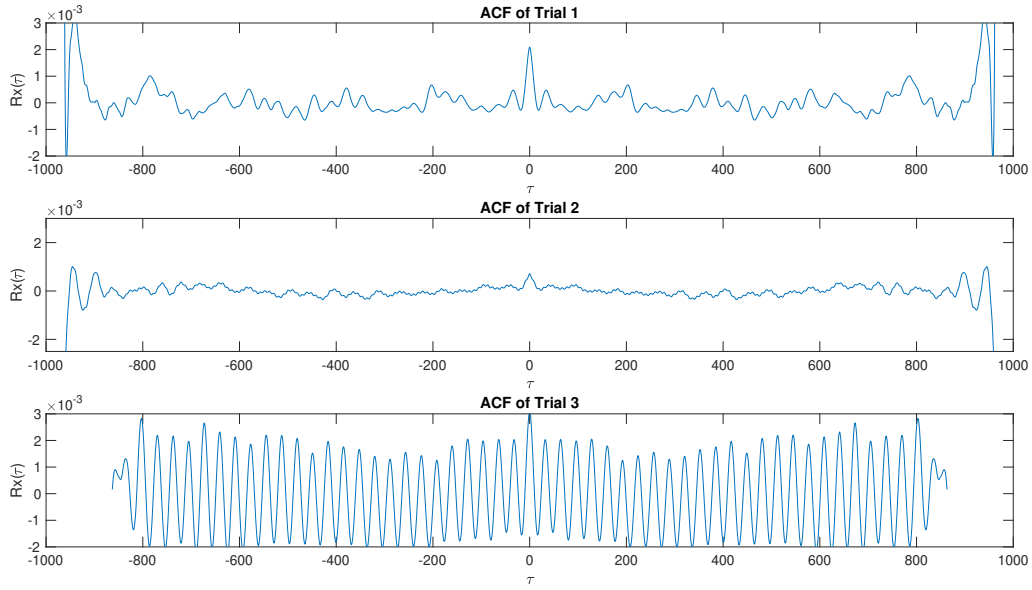
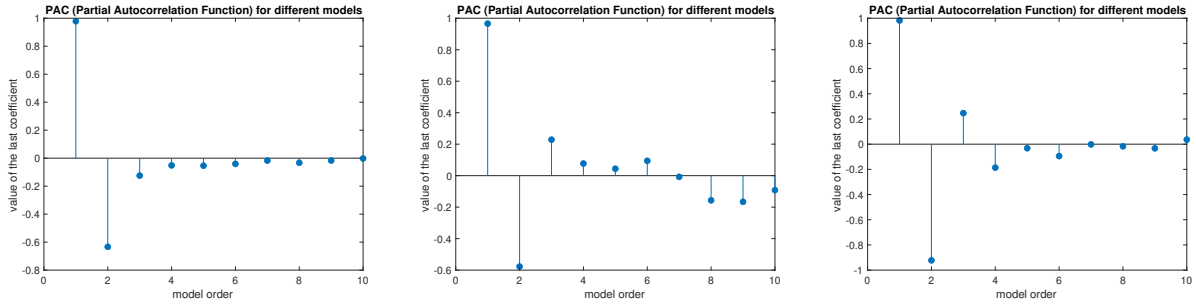


Figure 13: Autocorrelation sequences of the RRI data for the three trials – unconstrained breathing, constrained breathing at 50 bpm, and constrained breathing at 15 bpm

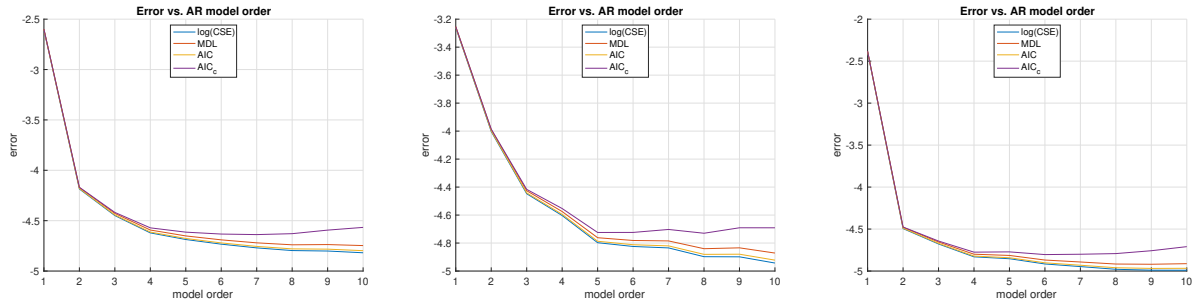
(c) The autocorrelation sequences of the RRI data for the three trials are given in Figure 13. We know that a finite MA(p) process has an ACF zero beyond p. (See 2.1.4) For an AR process however, the ACF is infinite in length and consists a mixture of damped exponentials and/or damped sine waves. [2]. Looking at Figure 13, we can safely say that the RRI data is an AR process. We note that this sinusoidal behaviour of the ACF is due to RRI data modulating during inspiration and expiration with heart rate increasing and decreasing respectively.



(a) PAC for RRI data of Trial 1 (b) PAC for RRI data of Trial 2 (c) PAC for RRI data of Trial 3

Figure 14: Partial Autocorrelation Functions for determining the model order

(d) Looking at Figures 14 and 15 and following a similar argument we followed in determining the model order for the sunspot data in (2.3.4), we determine the optimal AR model order for this data to be 2. The argument is that the PAC functions become considerably insignificant and the error plots don't show significant improvement after model order p is larger than 2. We could also argue that PAC values are significant until $p > 4$ and the error plots take a turn for the worse after model order 4. However, we argue that the optimal model order for this particular data would be 2 for good generalization.



(a) Error for RRI data of Trial 1 (b) Error for RRI data of Trial 2 (c) Error for RRI data of Trial 3

Figure 15: Error plots for estimation of RRI data with different model orders

3 Spectral estimation and modelling

1. We write a matlab function that calculates the periodogram according to equation 1. See Appendix B

$$\hat{P}_X(f) = \frac{1}{N}\left|\sum_{n=0}^{N-1} x[n]e^{-j2\pi f \frac{n}{N}}\right|^2 \tag{1}$$

Testing the function on WGN we get the plots in Figure 1. (f is the normalised frequency)

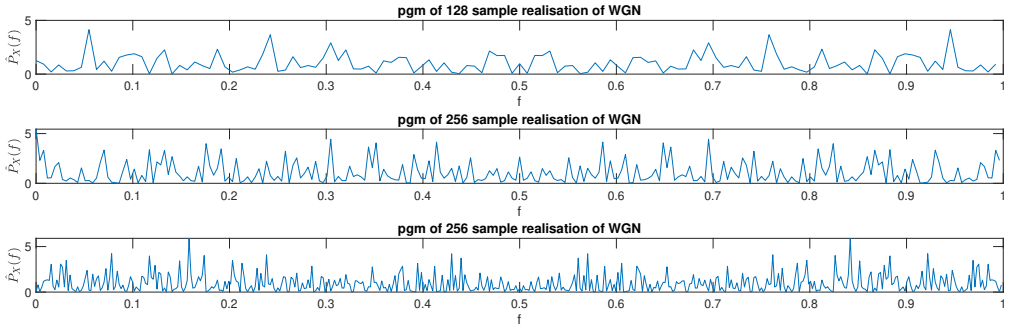


Figure 1: Testing `pgm.m` with different realisations of N-sample White Gaussian Noise

We expect WGN to have a constant spectrum independent of frequency with a value equal to the variance of the noise – in this case $\sigma^2 = 1$. However, as it can be seen from Figure 1, our estimates aren't constant at 1 and instead varies around that value. The reason for this is partly due to the finite lengths of the realisations. Another fact we have to consider is that we're estimating the PSD at N discrete frequencies in each of the plots. These estimates at each frequency are random variables.

We're sampling x[n] from a Gaussian distribution $\mathcal{X} \sim (0, 1)$, and for this reason the estimate of its PSD at each frequency is distributed with $\mathcal{X}^2$, which is a distribution with mean 1 and variance 2 (See literature for proof). For this reason, we do expect the plots to move around 1 a lot.

3.1 Averaged periodogram estimates

(1) We use a FIR filter with impulse response 0.2*[1 1 1 1 1] to smooth the PSD estimate in the previous exercise and get the plots in Figure 2. We observe that plots are now closer to the ideal PSD with less variation around 1. However, the variation is still very high.

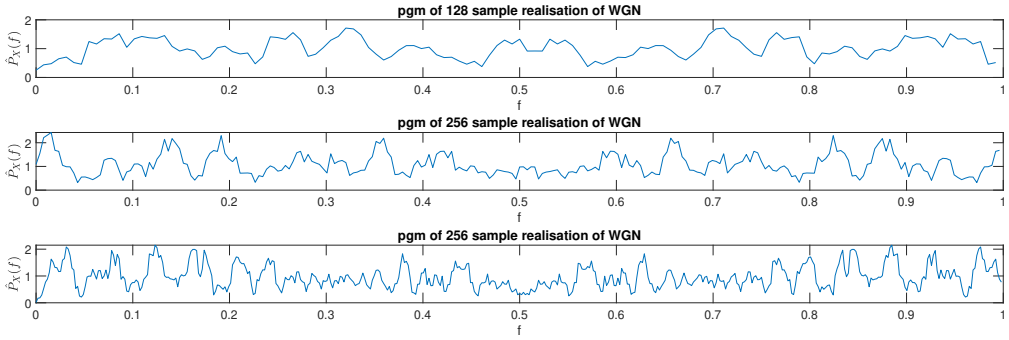


Figure 2: Smoothed PSD estimate (periodogram)

(2) We then generate a 1024-sample sequence of WGN and subdivide it into 128-sample segments. We plot the PSD estimate of each segment individually in Figure 3. We observe as in the first plot of Figure 2 that the plots vary around value 1 in different ways.

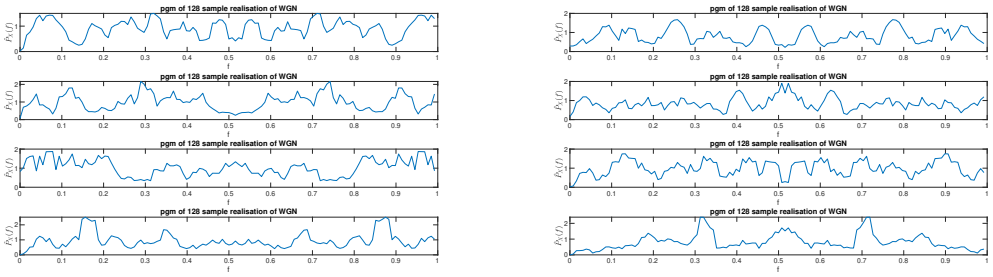


Figure 3: Individual PSD estimates of the 8 segments

(3) We then average these 8 segments to get the *averaged periodogram* (Figure 4). We observe that this plot is much smoother and also closer to the ideal PSD where the plot should be constant at 1 for all frequencies. By averaging the results of different segments, we decrease the variance of the plot around 1. Note the transient behaviour of the filter at small frequencies.

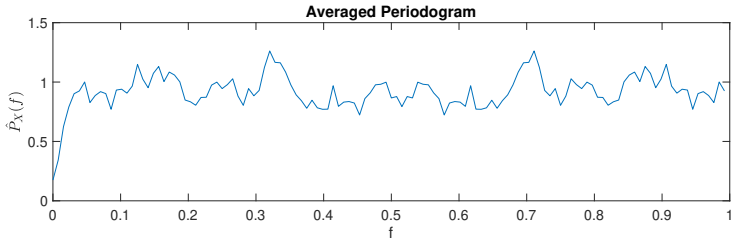


Figure 4: Averaged Periodogram

3.2 Spectrum of auto-regressive processes

(1) In figure 5, we have a WGN sequence and its filtered output with coefficients b=1 and a=[1, 0.9]. We observe a high pass behaviour from the filter when we view the filtered signal as rapid changes are still apparent while some low frequency behavior has vanished. We support this claim with the PSD plot in

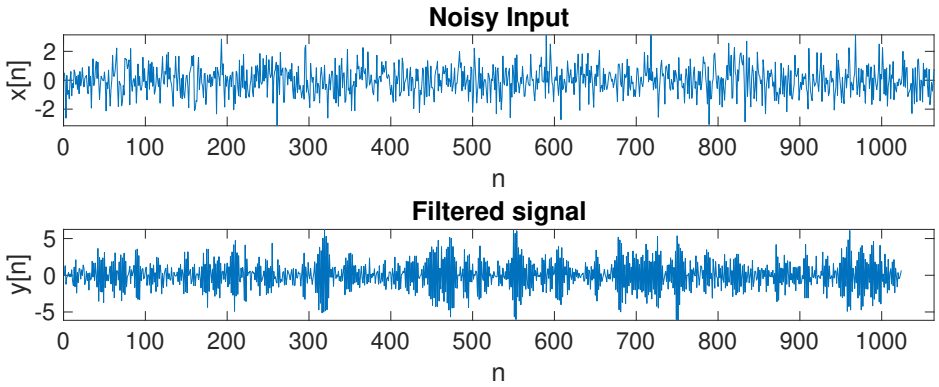


Figure 5: WGN signal and the filtered signal

Figure 6 (in blue) which verifies the high pass behaviour of the filter response. The plot follows equation 2 which achieves its maximum value when the normalized frequency f is equal to 0.5. For this reason we claim that the cut-off frequency is around 0.5.

$$P_Y(f) = \frac{1}{|\ 1 + 0.9e^{-j2\pi f}\ |^2} \tag{2}$$

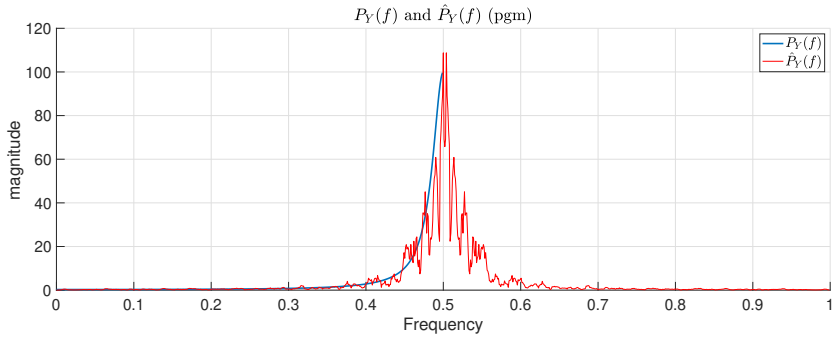


Figure 6: Theoretical PSD of the filter and the estimated PSD of the filtered signal

(2) We see in Figure 6 that the shape of the estimated PSD with the `pgm` function is very similar to the theoretical response of the filter as expected. Note that the estimated PSD shows the double-sided frequency response. The variant behaviour is due to the reasons explained earlier. We take a closer look between the interval $f = [0.4, 0.5]$.

(3) We use the periodogram estimator to estimate the PSD of a finite sequence discrete signal, which can also be seen as an infinite signal windowed with a rectangular function r[n]. This is described below.

$$y[n] = x[n] \cdot r[n] \qquad \text{where } r[n] = \begin{cases} 1, & 0 \leq n \leq N-1 \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

Then taking the Fourier Transform of both sides we get

$$Y(w) = \frac{1}{2\pi} X(w) * R(w) \tag{4}$$

where $R(w)$ is a sinc function with a main lobe and side lobes. For this reason, in the estimated PSD in Figure 7, we observe a "loby" behavior as well as some spectral leakage coming from convolution with a sinc function.

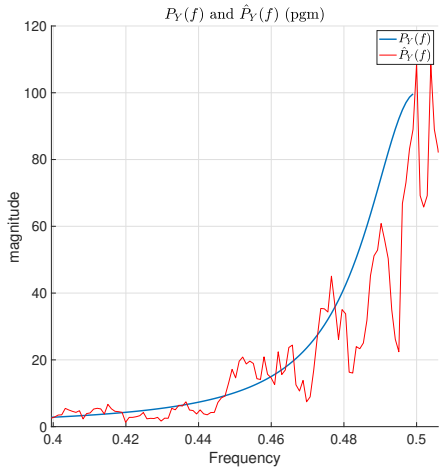


Figure 7: Zooming in...

(4) We have for model based PSD estimate

$$\hat{P}_Y(f) = \frac{\hat{\sigma}_X^2}{|\ 1 - \hat{a}_1 e^{-j2\pi f}\ |^2} \tag{5}$$

By following the reasoning below

$$y[n] = \hat{a}_1 y[n-1] + x[n] \tag{6}$$

$$R_y(k) = \begin{cases} \hat{a}_1 R_y(k-1), & k > 0 \\ \hat{a}_1 R_y(1) + \hat{\sigma}_X^2, & k = 0 \end{cases} \tag{7}$$

Then

$$\hat{a}_1 = \frac{R_y(1)}{R_y(0)} \tag{8}$$

and

$$\hat{\sigma}_X^2 = R_y(0) - \hat{a}_1 R_y(1) \tag{9}$$

Then from the autocorrelation function of y, we estimate $\hat{a} = -0.9097$ and $\hat{\sigma}_X^2 = 1.0404$ which are both very close to their theoretical values. Then the model based PSD according to equation 5 is given in Figure 8. The shape of the plot is very similar to the periodogram estimate and the theoretical PSD. However, the values are slightly higher which is due to the estimation variance of $\hat{a}_1$ and $\hat{\sigma}_X^2$.

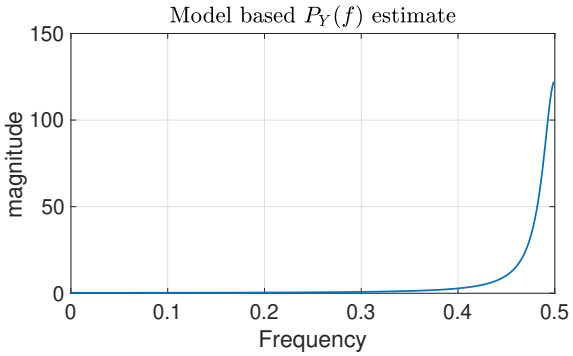


Figure 8: Model Based PSD Estimate

(5) In figure 9, we estimate the Power Spectral Density of the sunspot time series according to the model based PSD estimate algorithm, but also using our pgm.m function. For the model-based estimation, we observe that when the model order is low, the spectrum is very smooth and curvature doesn't change much. As the model order increases, we can observe a cut-off frequency around f = 0.1, and this becomes more apparent as we continue to overmodel. We also note that the periodogram estimate is most similar to the PSD estimate for the highest model order, 10. We can observe two poles in their plots – a strong one at $f = 0.1$ and a weak one between $f = [0.15, 0.20]$.

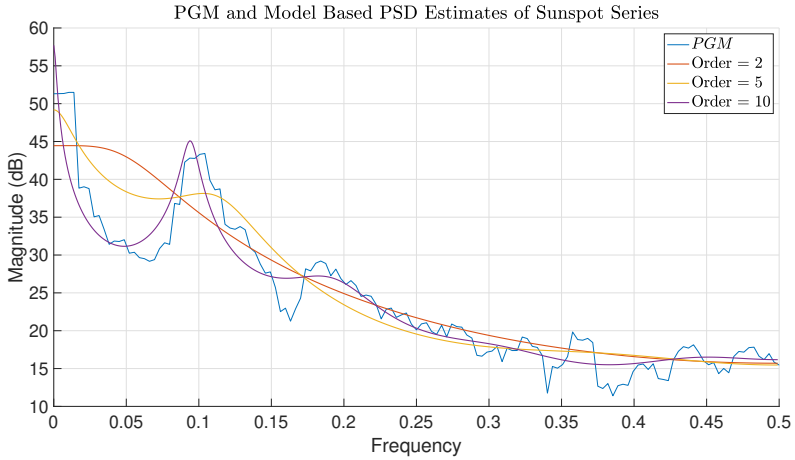


Figure 9: Power Spectral Density estimates of the sunspot time series using the periodogram and the model-based PSD method with different model orders

3.3 The Least Squares Estimation (LSE) of AR Coefficients

(1) The biased autocorrelation function of an AR(p) process can be written as

$$\hat{r}_{xx}[k] = \sum_{i=1}^{p} a_i \hat{r}_{xx}[k-i] + \epsilon[k], \qquad \text{for } i \geq 1 \tag{10}$$

which can be represented in vector form as $\mathbf{x} = \mathbf{s} + \epsilon$, where $\mathbf{s} = \mathbf{Ha}$. We show this in more detail as

$$\underbrace{\begin{bmatrix} \hat{r}_{xx}[1] \\ \hat{r}_{xx}[2] \\ \vdots \\ \hat{r}_{xx}[p] \\ \vdots \\ \hat{r}_{xx}[M] \end{bmatrix}}_{\mathbf{x}} = \underbrace{\begin{bmatrix} \hat{r}_{xx}[0] & \hat{r}_{xx}[1] & \dots & \hat{r}_{xx}[p-1] \\ \hat{r}_{xx}[1] & \hat{r}_{xx}[0] & \dots & \hat{r}_{xx}[p-2] \\ \vdots & \vdots & \ddots & \vdots \\ \hat{r}_{xx}[p-1] & \hat{r}_{xx}[p-2] & \dots & \hat{r}_{xx}[0] \\ \vdots & \vdots & \ddots & \vdots \\ \hat{r}_{xx}[M-1] & \hat{r}_{xx}[M-2] & \dots & \hat{r}_{xx}[M-p] \end{bmatrix}}_{\mathbf{H}} \underbrace{\begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_p \end{bmatrix}}_{\mathbf{a}} + \underbrace{\begin{bmatrix} \epsilon[1] \\ \epsilon[2] \\ \vdots \\ \epsilon[p] \\ \vdots \\ \epsilon[M] \end{bmatrix}}_{\epsilon} \tag{11}$$

where ϵ is the error due to the effects of errors on ACF estimation. Then we can represent the cost function $\mathrm{J} = \mid \epsilon \mid^2$ as $|\mathbf{x} - \mathbf{Ha}|^2 = (\mathbf{x} - \mathbf{Ha})^T(\mathbf{x} - \mathbf{Ha})$. Then to find the Least Squares estimates of the unknown coefficients $\mathbf{a}$, we take the derivative of this loss function with respect to $\mathbf{a}$ and make it equal to 0.

$$J = (\mathbf{x} - \mathbf{Ha})^T(\mathbf{x} - \mathbf{Ha}) \tag{12}$$

$$\frac{\partial J}{\partial \mathbf{a}} = 2(\mathbf{x} - \mathbf{Ha}) = 0 \tag{13}$$

$$\mathbf{x} = \mathbf{Ha} \tag{14}$$

$$\mathbf{H}^T\mathbf{x} = \mathbf{H}^T\mathbf{Ha} \tag{15}$$

$$\mathbf{a} = (\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\mathbf{x} \tag{16}$$

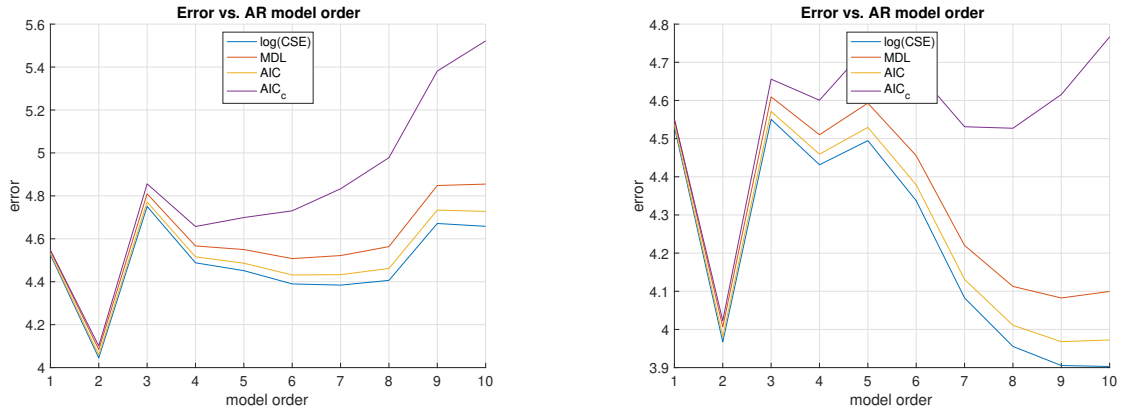
The coefficients obtained this way is similar but different to the method we used when we estimated them with Yule-Walker equations. Here we use a tall $\mathbf{H}$ matrix where $M > p$, where as in the Yule-Walker estimates we used the same equations with M = p. For this reason, the matrix $\mathbf{H}$ was an invertible positive definite (Toeplitz) matrix and we would obtain the coefficients by

$$\mathbf{a} = \mathbf{H}^{-1}\mathbf{x} \tag{17}$$

(2) The observation matrix $\mathbf{H}$ is made of auto-correlation values of a stochastic AR process. For this reason $\mathbf{H}$ itself is a stochastic matrix.

(3) Following equation 16, we calculate coefficients $\mathbf{a} \in \mathbb{R}^{p\times 1}$ for different AR(p) model orders for the standardised sunspot time series. Using M = 287, which is the maximum length we could use, we get the following coefficients:

$$\begin{aligned}
a_1 &= [0.8543] \\
a_2 &= [1.6833, -0.9569] \\
a_3 &= [2.4541, -2.3054, 0.7958] \\
a_4 &= [2.2139, -1.6142, 0.0634, 0.2967] \\
a_5 &= [2.1627, -1.6286, 0.3479, -0.0882, 0.1721] \\
a_6 &= [2.1434, -1.6144, 0.2841, 0.1385, -0.1056, 0.1229] \\
a_7 &= [2.1303, -1.6081, 0.2895, 0.0436, 0.1773, -0.1999, 0.1394] \\
a_8 &= [2.0923, -1.5538, 0.2156, 0.1339, -0.1213, 0.5135, -0.6326, 0.3308] \\
a_9 &= [2.1763, -1.6992, 0.3313, 0.0976, -0.1074, 0.7069, -1.2624, 1.0490, -0.3176] \\
a_{10} &= [2.1718, -1.6848, 0.3148, 0.1080, -0.1122, 0.7093, -1.2491, 1.0085, -0.2736, -0.0186]
\end{aligned}$$



(a) Error vs. model order for M=287

(b) Error vs. model order for M=50

Figure 10: Approximation error to the original data

(4) In figure 10, we plot the approximation errors for different model orders as we did in 2.3.4. Looking at the plots, we again determine the optimal model order to be 2 as we did before. When we check the coefficients above, we realize after model order 2, the first coefficient of $\mathbf{a}$ becomes much larger and we note the sudden jump in error from order 2 to 3 in the plots in relation to this.

When compared with the coefficients we got in 2.3.3 from the Yule-Walker equations, we can say that especially the first two order coefficients are very similar. However, after model order two, there are slight

differences in coefficient values, yet the signs are mostly the same and the values are not too far away from the ones we got with the Yule-Walker equations.

(5) We plot the power spectral densities in Figure 11 associated with different AR(p) models we have given the coefficients for above. (Note that we don't consider the power of the noise input). Shape of the plots are quite similar to the ones we had when we used the aryule function to get the coefficients in Figure 9. Again, a pole at $f = 0.1$ is apparent in the PSD plots.

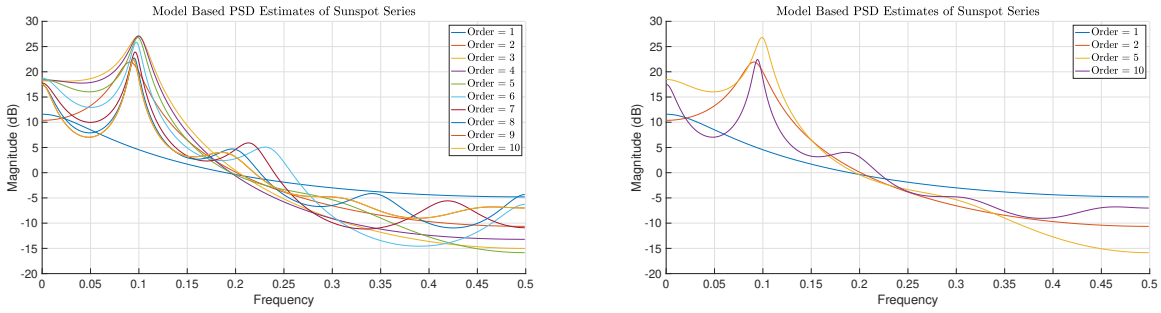


Figure 11: Power Spectra Associated with different AR(p) models

(6) We label the last 38 data points of the 288-sample sunspot data as the *test set*. Then using the first N = [10:5:250] number of data points, we estimate the coefficients of the AR(2) model (optimal model order). Then for each model, we get the approximation error on the test set to compare different models trained on different data lengths. We plot the results in Figure 12. We observe that the error oscillates for training with number of data points less than 80, then it achieves a steady value of 670. For this reason, we suggest that we train with more than 100 data points to model this data with an AR model.

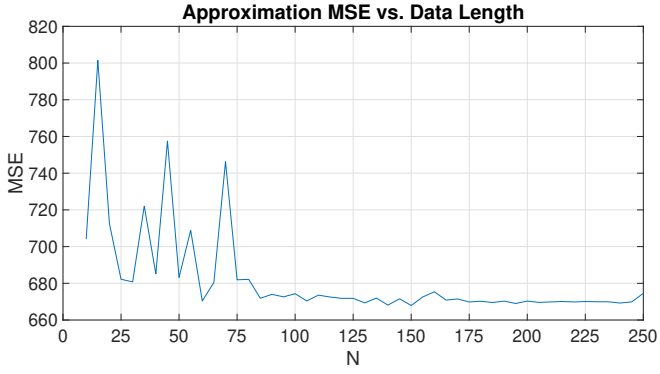


Figure 12: Mean Squared Error vs. data length used to model

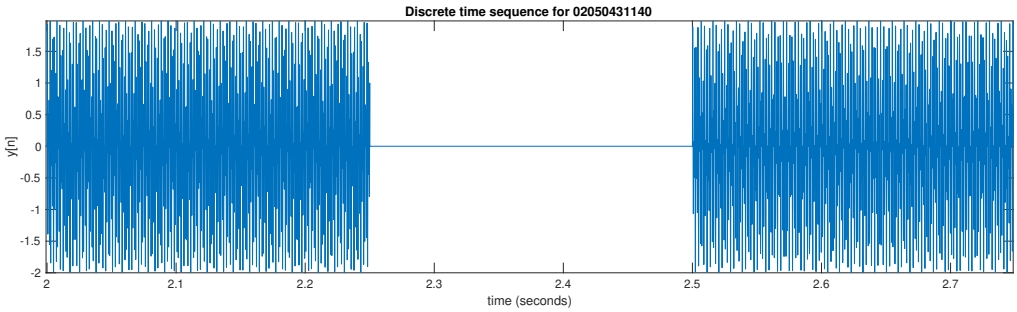
3.4 Spectrogram for time-frequency analysis: dial tone pad

(1) We generate two random London landline numbers and compute their discrete time sequences **y** according to equation 18 and Table 1. We use a sampling rate of 32768 Hz which is much higher than double the highest frequency present in the signal (Nyquist rate). We also note that 0.25s (length of a tone) divides by 1/32768 (time between two samples) exactly giving 8192. For these reasons, it is appropriate to sample at this rate. We have the tones for different keys in Figure 13.

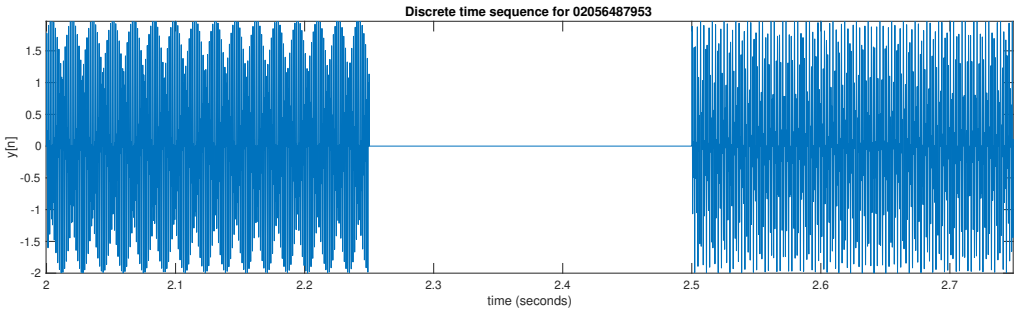
$$y[n] = sin(2\pi f_1 n) + sin(2\pi f_2 n) \tag{18}$$

$f_1 \backslash f_2$	1209Hz	1336Hz	1477Hz
697Hz	**1**	**2**	**3**
770Hz	**4**	**5**	**6**
852Hz	**7**	**8**	**9**
941Hz	*	**0**	#

Table 1: Dial pad frequencies



(a) Tone for 0 - idle - 4



(b) Tone for 6 - idle - 4

Figure 13: Discrete time signal y for time = (2, 2.75)s

(2) We then use the `spectrogram` function to analyse the spectral components of sequence **y**. We divide the signal into 21 segments of 8192 samples and window them with a Hanning window without any overlaps between the segments. We can see very clearly the 21 different segments and the frequency values present at these segments in Figure 14. The idle times are completely dark where there is no signal present, and the segments for keys have very clear two separate strong frequency values in yellow. For example, for time between 0 and 0.25s, we have very strong frequency values around 1336Hz and 941Hz which represents key 0 as expected. The frequencies are very distinguishable at this stage since there is no noise present.

(3) It is very possible to identify the sequences generated using the spectrum (by using the frequencies with high power at a slot) and the key values f_1 and f_2. For example, in the sequence of Figure 14, the third slot between time 0.5 and 0.75s has high power around $f = 1.33$kHz and $f = 0.70$kHz, which corresponds to the frequency values of the tone for key 2. Then comes an idle slot with no signal, thus no power at any frequency present. Therefore, we could easily retrieve the sequence by classifying the keys accordingly.

(4) We then corrupt the sequence **y** with WGN with different powers. We can see how sequences are corrupted in Figure 15. With noise variances of 16 and 64, we observe that the original signal is visually completely immersed in noise.

We then take a look at their spectra in Figure 16. Since white noise has constant spectra across all frequencies, the power of all frequencies at each segment increases at a similar way. We note that while the noise variance is low, it is still easy to classify each segment correctly. However, we observe that when

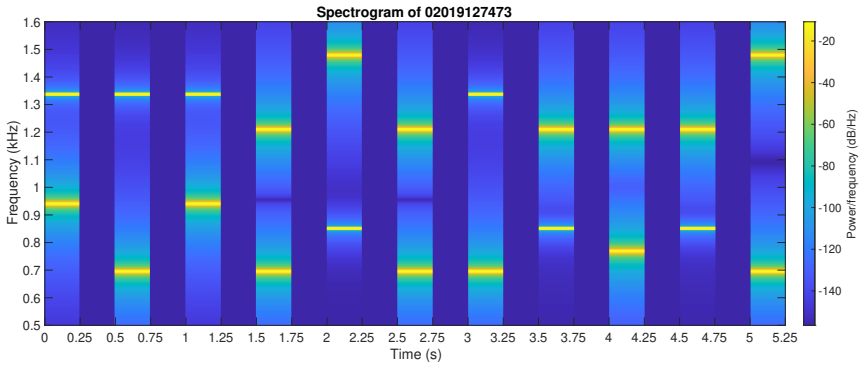


Figure 14: Spectrogram of the signal for 02019127473

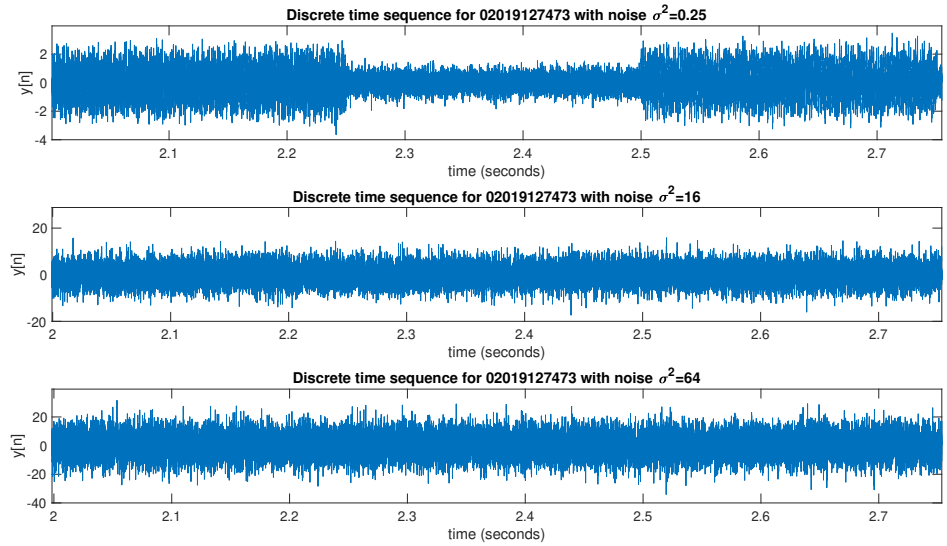


Figure 15: Discrete time sequence $\mathbf{y}$ corrupted by noise

the variance is 16, it gets much harder to classify the key, yet it is still possible. However, when the noise variance is 64, it is at least visually not quite possible to classify the keys correctly since all frequencies in the spectra have very high power, and the power of signal $\mathbf{y}$ just gets immersed in it.

3.5 Real world signals: Respiratory sinus arrhythmia from RR-Intervals

(a) In Figure 17, we estimate the Power Spectral Density of the RRI data we have from three trials. We obtain the PSD estimates using our standard `pgm.m` function as well as the averaged periodogram method with different window lengths. The plots are in normalized frequency and we would need to multiply by the sampling rate, 4Hz to get the actual frequency values. We also note that the plots are zoomed in between 0 and 0.2 since rest of the frequencies have power near zero.

(b) We notice a stronger peak in Trial 3 compared with other trials which is around $f = 0.035$. This means trial 3 has a peak around 0.14Hz (0.035x4) for the frequency of respiration. Since Trial 3 was breathing at regular rates, the behavior was very sinusoidal, and this may have caused the greater "peakiness".

Trial 2 was also breathing at a regular but much faster rate. This may have caused the three different but harmonic peaks we observe in the plots. Trial 1 on the other hand wasn't breathing in a regular rate but rather was breathing in an unconstrained way. For this reason, the spectra isn't very peaky but is spread around low frequency values.

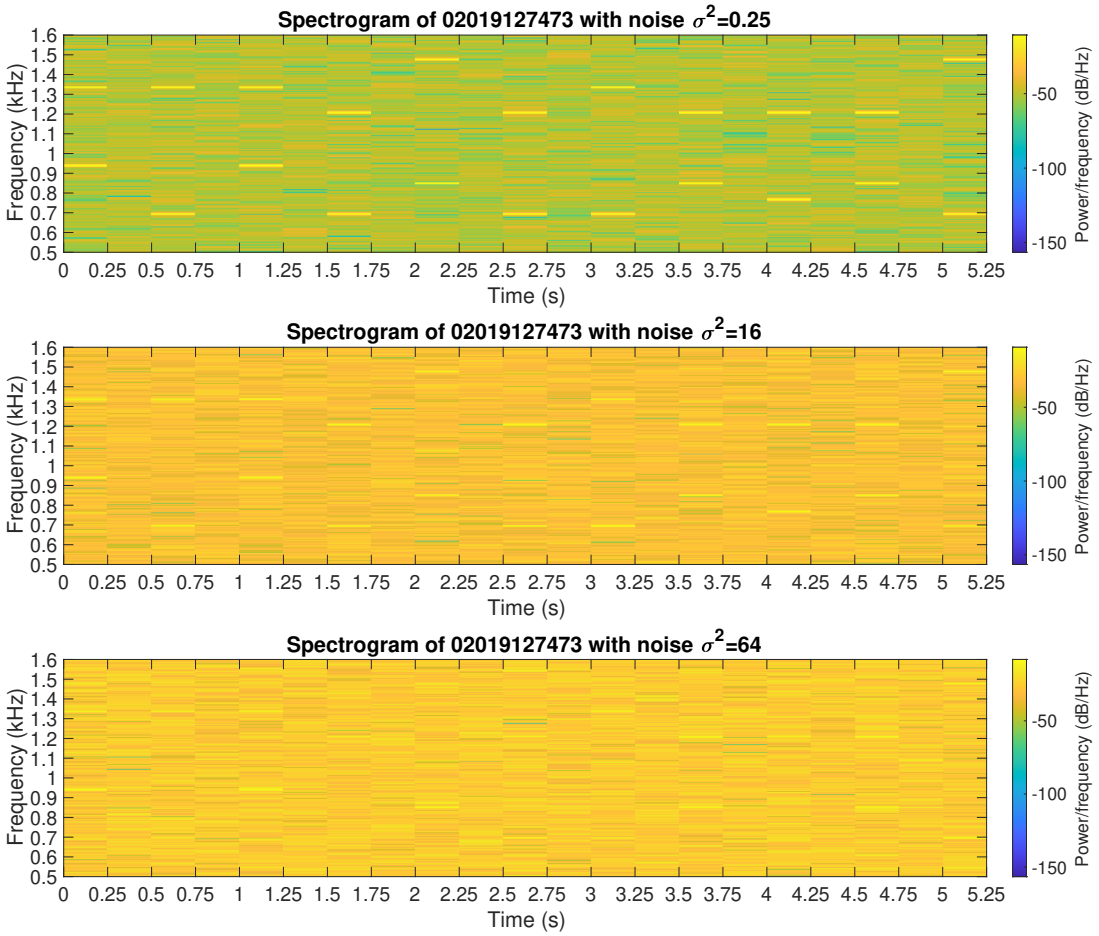


Figure 16: Spectra of the corrupted signals

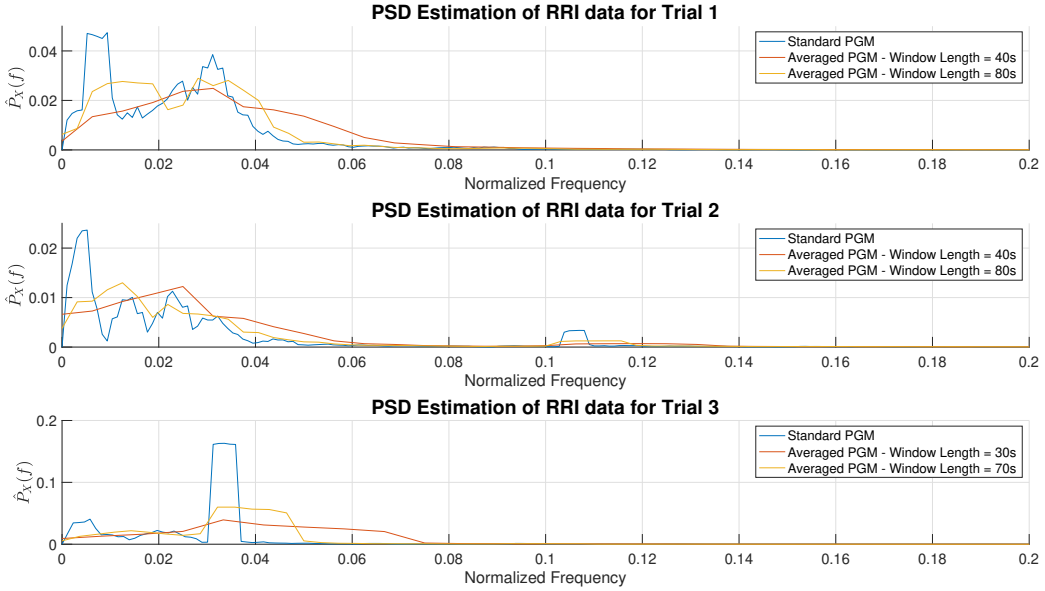


Figure 17: Power Spectral Density estimation of the RRI data

4 Optimal filtering - fixed and adaptive

4.1 Wiener filter

(1) We create an FIR filter with coefficients b = [1,2,3,2,1], a = 1 and feed it with WGN in its input. We then normalise the output y[n] – so it has unit variance – and add another WGN η[n] with $\sigma_\eta^2 = 0.01$ to corrupt the signal and generate z[n]. The SNR for z[n] is then $10log(\frac{1}{0.01}) = 20$dB. We use the Wiener filter to identify the filter coefficients following the Wiener-Hopf equation:

$$\mathbf{w}_{opt} = \mathbf{R}_{xx}^{-1}\mathbf{p}_{zx} \tag{1}$$

where

$$\mathbf{p}_{zx} = \mathbb{E}\{z[n]\mathbf{x}[n]\} = \begin{bmatrix} \mathbb{E}\{z[n]x[n]\} \\ \mathbb{E}\{z[n]x[n-1]\} \\ \vdots \\ \mathbb{E}\{z[n]x[n-N_w]\} \end{bmatrix} = \begin{bmatrix} r_{zx}(0) \\ r_{zx}(-1) \\ \vdots \\ r_{zx}(-N_w) \end{bmatrix} \tag{2}$$

$$\mathbf{R}_{xx} = \mathbb{E}\{\mathbf{x}[n]\mathbf{x}^T[n]\} = \begin{bmatrix} r_{xx}(0) & r_{xx}(-1) & \dots & r_{xx}(-N_w) \\ r_{xx}(1) & r_{xx}(0) & \dots & r_{xx}(-N_w+1) \\ \vdots & \vdots & \ddots & \vdots \\ r_{xx}(N_w) & r_{xx}(N_w-1) & \dots & r_{xx}(0) \end{bmatrix} \tag{3}$$

Considering that the order of the filter is 5 with $N_w = 4$, we calculate the 5x5 $\mathbf{R}_{xx}$ matrix and 5x1 $\mathbf{p}_{zx}$ vector to find the optimal coefficients of the filter to be $\mathbf{w}_{opt}$ = [0.23, 0.45, 0.68, 0.45, 0.22]. We realise that these coefficients are just a scaled down version of b, and the reason for it is that we normalise y[n]. If we scale the coefficients up using the same factor that we normalised y[n] with (i.e. σ_y), we get $\mathbf{w}_{opt}$ = [1.01, 2.01, 3.02, 2.00, 1.00] which is almost identical to b.

(2) We vary the scaling of the additive noise η[n] to investigate its effects on the Wiener solution. (We provide the coefficients after scaling them back up). We have:

For $\sigma^2 = 0.1$, SNR = 10dB, $\mathbf{w}_{opt}$ = [0.98, 2.08, 2.97, 1.98, 1.02]
For $\sigma^2 = 0.5$, SNR = 3dB, $\mathbf{w}_{opt}$ = [1.02, 1.91, 2.95, 1.95, 1.06]
For $\sigma^2 = 1$, SNR = 0dB, $\mathbf{w}_{opt}$ = [1.03, 2.08, 2.77, 1.84, 1.16]
For $\sigma^2 = 2$, SNR = -3dB, $\mathbf{w}_{opt}$ = [0.84, 1.67, 3.25, 1.74, 0.82]
For $\sigma^2 = 5$, SNR = -7dB, $\mathbf{w}_{opt}$ = [1.37, 2.03, 2.98, 2.40, 0.90]
For $\sigma^2 = 10$, SNR = -10dB, $\mathbf{w}_{opt}$ = [0.53, 1.56, 2.28, 1.47, 0.94]

We observe that the coefficients start to diverge from filter coefficients b = [1,2,3,2,1] as noise power increases, which is expected. But even for SNR of -10dB, we have coefficients that are somewhat similar to b. When we did this analysis, we assumed we knew the filter order to be 5 or $N_w = 4$. We know check what happens if we assume N_w to be 5 or 6. We would then need to add elements to the $\mathbf{R}_{xx}$ matrix and $\mathbf{p}_{zx}$ vector. We move back the noise variance to 0.01 and get the following coefficients:

For $N_w = 5$, $\mathbf{w}_{opt}$ = [0.99, 2.01, 2.98, 1.99, 1.00, 0.01]
For $N_w = 6$, $\mathbf{w}_{opt}$ = [1.02, 2.01, 2.99, 2.00, 1.01, 0.01, 0.01]

We observe that the first five coefficients are similar to b and the 6th and 7th coefficients are negligible. So even if we assumed N_w to be greater than the actual model order, we could correct our assumption after observing the negligible coefficients. (With increasing error power, it might be harder to classify a coefficient as negligible).

(3) To estimate the number of multiplications and additions in the calculation of the Wiener solution, we need to consider couple things. First the computation of the auto-correlation and the cross-correlation functions. We calculate 2N-1 values for those functions (where N is the length of the input signal), and for each value we do N - τ multiplications and additions – on average N/2. So we do $(\frac{N}{2}+\frac{N}{2})\times(2N-1) = 2N^2-N$ operations for creation of each correlation function and $4N^2 - 2N$ for both.

If we assume we're given the correlation values and instead just estimate the number of operations needed to compute the optimal coefficients with the given matrix and vector, we have to consider:

1. Inverting the matrix $\mathbf{R}_{xx} \Longrightarrow (N_w + 1)^3$
2. $\mathbf{R}_{xx} \times \mathbf{p}_{zx} : (N_w + 1)^2$ operations for each of the $(N_w + 1)$ coefficients in $\mathbf{w}_{opt} \Longrightarrow (N_w + 1)^3$

Then in total, we require $2(N_w + 1)^3$ operations to do the matrix calculations. This number equals to 250 for the example we have done.

4.2 The least mean square (LMS) algorithm

(1) We create the function `lms.m` which follows the following algorithm to compute e[n], $\hat{y}$[n], and the $\mathbf{W}$ matrix which includes the evolution of the adaptive weights in its columns. (MATLAB code in Appendix C)

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu e[n]\mathbf{x}(n) \text{ and } \mathbf{W}(n+1) = [\mathbf{W}(n), \mathbf{w}(n+1)] \tag{4}$$

$$\hat{y}[n] = \mathbf{w}^T(n)\mathbf{x}(n) \tag{5}$$

$$e[n] = z[n] - \hat{y}[n] \qquad n = 0, 1, ... \tag{6}$$

We apply this algorithm to $\mathbf{x}$ and $\mathbf{z}$ signals in 4.1 and observe the evolution of the weights to b = [1,2,3,2,1] as expected in time with e[n] decreasing to 0. Instead of a block-based approach as in 4.1, with this algorithm we compute the weights sequentially with each x[n] and z[n] value we receive.

(2) We plot the evolution of coefficients and the error power in Figure 1. We use 4 different adaptation gains to see its effect. We observe that when $\mu = 0.01$ the adaptive filter coefficients converge to the Wiener solution at about time step 400. When we decrease μ to 0.002, because it represents the "learning rate", the adaptive filter learns and adapts its coefficients very slowly that it still doesn't converge at time step 1000. We expect it to converge later on, however, it takes time. (There is also a possibility that it might get stuck at a local minima). When we do the opposite and increase μ to 0.1, we observe very fast convergence, at about time step 50. the error plots also behave accordingly to the convergence of the coefficients as expected.

However, when $\mu = 0.4$ we observe that the system becomes unstable. This is because the learning rate is too high and above the so called $\mu_{critical}$ which defines a stability bound for the learning rate. We also note that when μ approaches this value but stays less than it, the trajectory becomes oscillatory or overdamped.

(3) The Wiener filter can be computationally demanding since we have to calculate the inverse of a possibly large correlation matrix $\mathbf{R}_{xx}$. Rather then taking a block-based approach, the LMS uses the instantaneous estimates of the auto-correlation and cross-correlation functions and updates the weights in an iterative fashion. To calculate the number of operations required at each iteration we consider:

1. calculation of $\hat{y}[n]$: $(N_w + 1)$ multiplications and then $(N_w + 1)$ additions $\Longrightarrow 2(N_w + 1)$ operations
2. calculation of e[n] $\Longrightarrow$ 1 operation
3. update of $\mathbf{w}$: $2(N_w + 1)$ multiplications and $(N_w + 1)$ additions $\Longrightarrow 3(N_w + 1)$

So in total we require $5N_w + 6$ operations in each iteration of the LMS algorithm. This number equals to 31 for the example we have done.

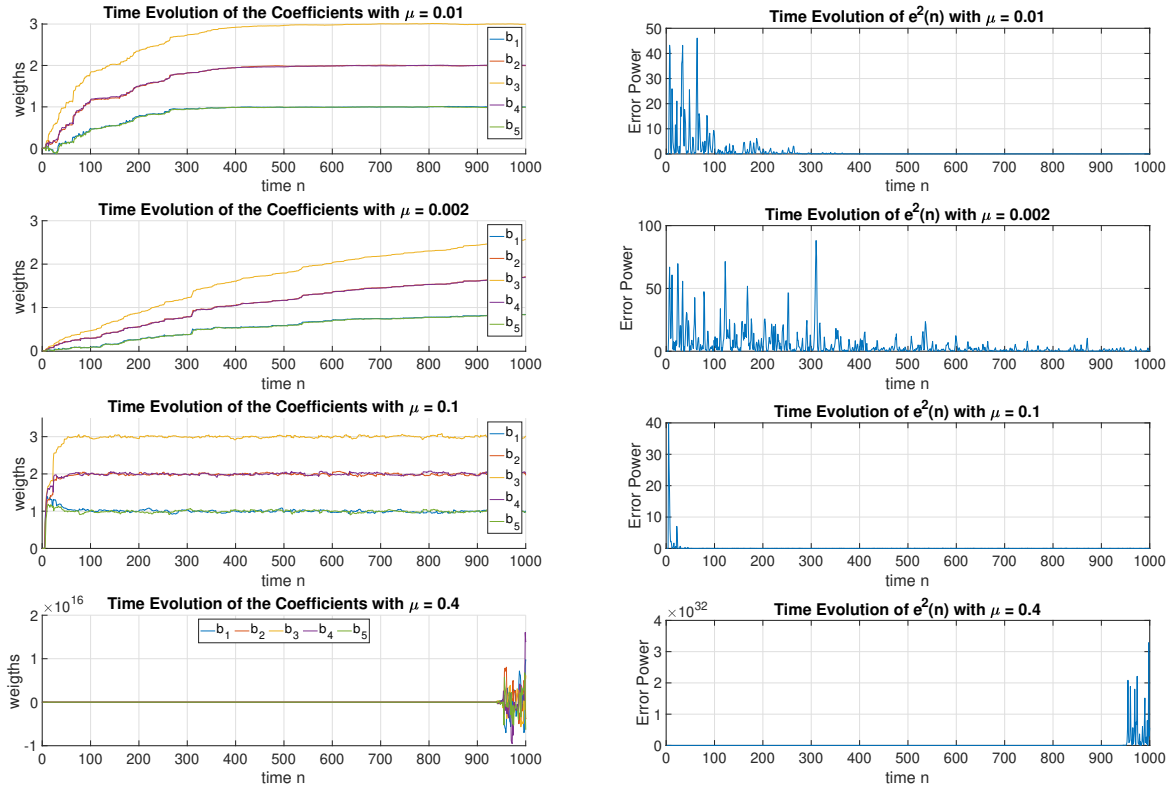


Figure 1: Evolution of the Adaptive Filter Coefficients and the Error Power using the LMS algorithm

4.3 Gear shifting

To obtain improved estimates in the steady state, we try decreasing the adaptation gain in time. We make $\mu = 0.1 \times 0.99^n$ so that as time increases, the updates to the weight vector becomes smaller. When we compare Figure 2 with the plot in Figure 1 for $\mu = 0.1$, we can see the difference in behavior in the steady state – it is now very flat. By decreasing the update rate in time, we get the benefit of converging fast with large updates at the beginning and staying steady with small updates later in time.

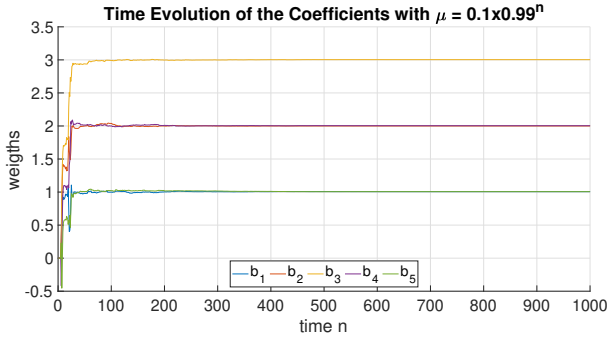


Figure 2: Evolution of the coefficients with decreasing μ

Then we try to improve the use of time-varying adaptation gain further. Our aim is to minimise the error which we can represent by

$$e[n+1] = e[n] + \sum_{m=0}^{N_w} \frac{\partial e[n]}{\partial w_m(n)} \Delta w_m(n) \tag{7}$$

Since $e[n] = z[n] - \mathbf{w}^T\mathbf{x}(n)$, we get that $\frac{\partial e[n]}{\partial w_m(n)} = -x(n-m)$. Also knowing $\Delta w_m(n) = \mu e[n]x(n-m)$,

$$e[n+1] = e[n]\left[1 - \mu \sum_{m=0}^{N_w} x^2(n-m)\right] = e[n](1 - \mu \mid\mid \mathbf{x}(n) \mid\mid_2^2) \tag{8}$$

To minimise the error we want $e[n+1] = 0$. Then

$$0 = 1 - \mu \mid\mid \mathbf{x}(n) \mid\mid_2^2 \tag{9}$$

$$\mu = \frac{1}{\mid\mid \mathbf{x}(n) \mid\mid_2^2} \quad \text{but for practical reasons we use} \quad \mu = \frac{\mu_{const}}{\mid\mid \mathbf{x}(n) \mid\mid_2^2 + \epsilon} \tag{10}$$

where μ_{const} is the inital value we provide and ϵ is a small constant to avoid division by 0 for small values of $\mid\mid \mathbf{x} \mid\mid$. [3] We implement this subroutine to `lms.m` function with $\epsilon = 0.001$. (Note that the code in Appendix C is for this implementation). We initialize μ_{const} to 0.5 and apply the decay rate 0.99 and get the plot in Figure 3. This helps the convergence of the algorithm immensely letting it use an initially high μ without divergence. As a result, rise time for this algorithm is pretty quick, which is about 10 time steps.

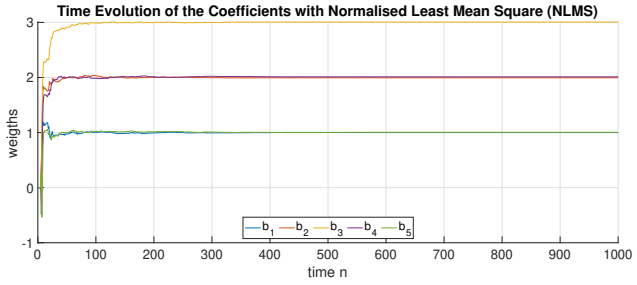


Figure 3: Evolution of the coefficients with Normalized Least Mean Square Algorithm

4.4 Identification of AR processes

(1) We generate a signal with an AR(2) filter of coefficients $a_1 = -0.9$ and $a_2 = -0.2$ using 3000-sample WGN as its input. We then analyse the generated signal. Since we know the generated signal is an AR signal of order 2, it should follow the equation

$$x[n] = a_1 x[n-1] + a_2 x[n-2] + w[n] \tag{11}$$

where w[n] is the WGN. Then to identify the coefficients, we use the signal x[n] in the following way

$$\hat{x}[n] = \hat{a}_1 x[n-1] + \hat{a}_2 x[n-2] \tag{12}$$

where $\hat{a}_1$ and $\hat{a}_2$ are the estimated filter coefficients using the adaptive LMS algortihm in order to make $e[n] = x[n] - \hat{x}[n]$ as small as possible. Following a similar argument in 4.3, we use an adaptive time-varying μ as in equation 10. If successful, we expect the filter coefficients to converge to $\hat{a}_1 \approx -0.9$ and $\hat{a}_2 \approx -0.2$.

(2) Using a decay rate of 0.998 and initializing $\mu = [0.01, 0.02, 0.05, 0.1]$, we observe the results in Figure 4. When $\mu = 0.01$, we observe that after a certain time step the coefficients get stuck at non-ideal values since the updates are no longer large enough due to low μ and the decaying rate. They get stuck at a local minima end can't get away from it.

When $\mu = 0.05$ or 0.1, the updates present a divergent behavior suggesting that the adaptation gains are too high – coefficients get away from their theoretical values. As the decaying rate making the updates smaller and smaller, they settle down at a minima but now at another local minima due to their initial divergent behavior, and they can't get to the global minima due to very small updates at that point in time.

With these observations, we argue the importance of selection of the adaptation gain μ and that one should be careful while doing it. When $\mu = 0.02$, we observe that it is initially big enough to discover the global minimum, but not too big to diverge from it. We report this as our best result and observe the convergence of the coefficients to their theoretical values.

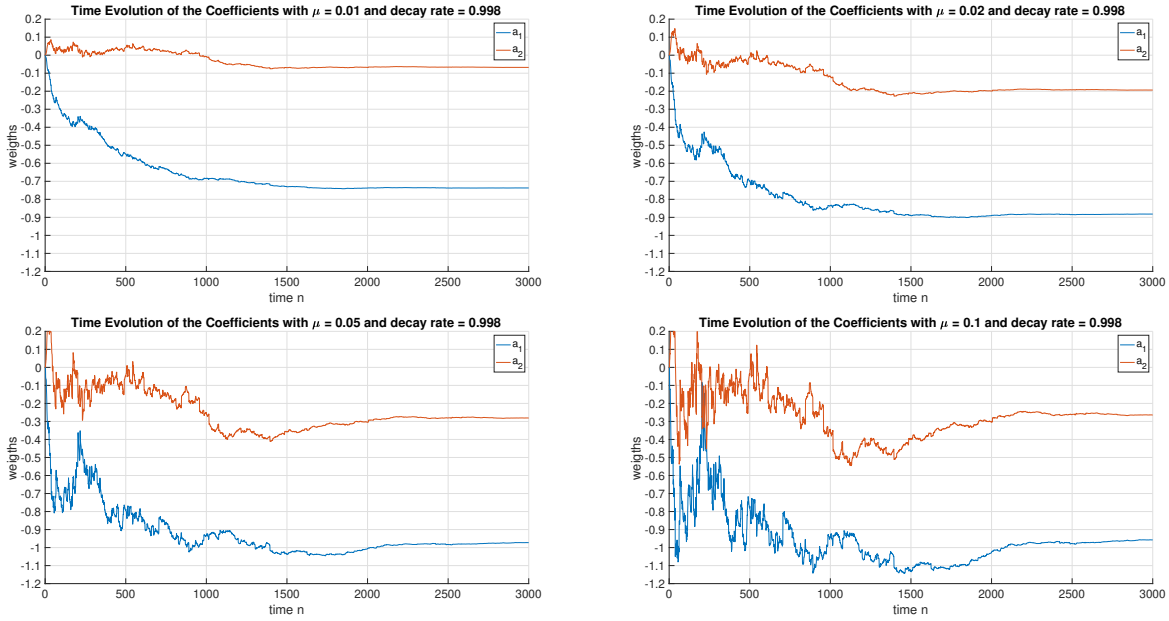


Figure 4: Evolution of the Adaptive Filter Coefficients with different adaptation gains

4.5 Speech Recognition

(1) We test the performance of the predictor used in 4.4 on real-world signals. Saying the letters "e", "a", "s", "t", "x" and sampling our voice with $f_s = 44.1$kHz and using 1000 sample from each recording, we investigate the predictor. First, we have to consider a viable adaptation gain and select the order of the filter. From previous investigations, we know $\mu = 0.5$ could be practical. We start off with this value and change it during experimentation for a more optimal value. We also implement "gear shifting" as in 4.3.

(2) On top of some heuristic approaches, we follow a more analytical method to find the optimal filter order p. We run the algorithm for p = 1 to 10 – all of which somewhat converge to a minima – and after obtaining the coefficient estimates check the approximation error using MDL, AIC, AICc criterion as in 2.3.4. On top of this, we use the prediction gain defined by $R_p = 10 \log_{10} \frac{\sigma_x^2}{\sigma_e^2}$ and plot it against the model order. We also consider the strength of the Partial Autocorrelation Function (p^{th} coefficient of order p model). An example of this process for letter "a" is given in Figure 5, which suggests an optimal order of 2.

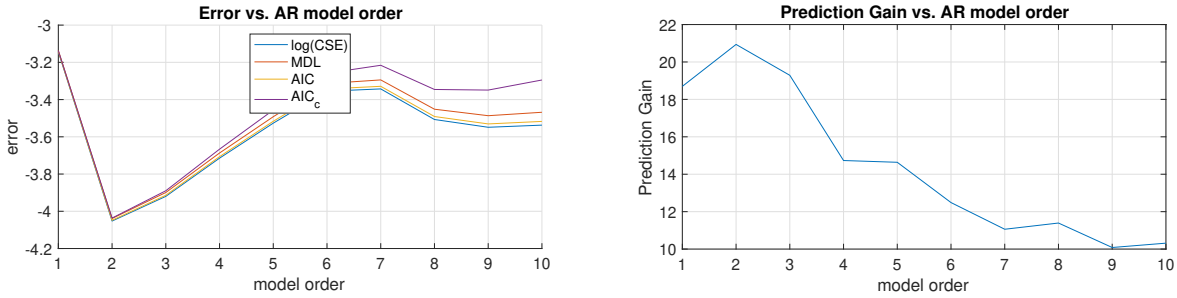


Figure 5: Finding the optimal model order for "a"

In Figure 6, we report the evolution of the coefficients following the methods mentioned. (R_p values provided with the title). We observe that all filters converge well in the given time except for sound "e". Although providing us with a high R_p value, we can see that the coefficients are still adapting steadily. We argue that we would need a longer input sample and train longer in order to converge.

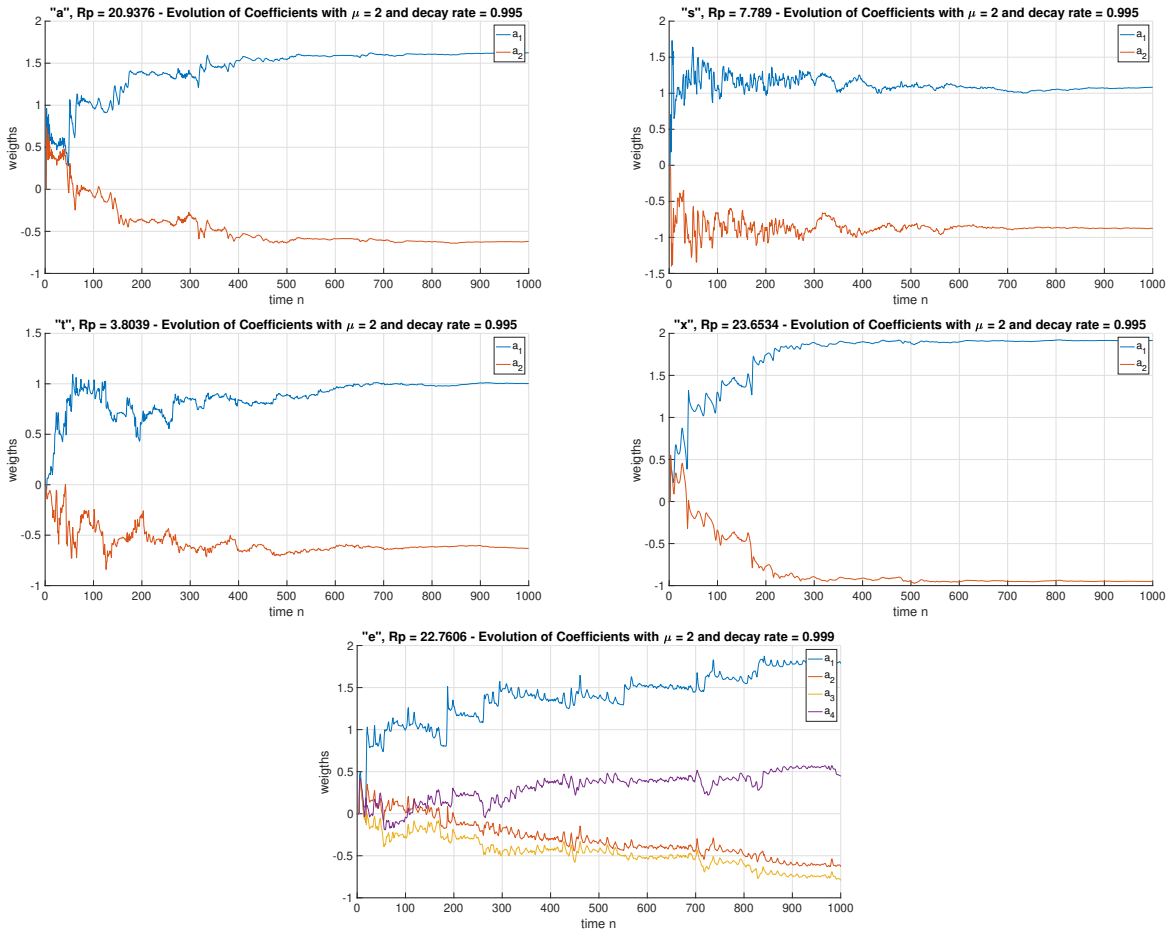


Figure 6: Evolution of the Adaptive Filter Coefficients for different sounds

(3) Note that we chose a model order 2 for each of the recordings except for sound "e" for which the analysis suggested a higher order model. We have done the recordings at a sampling rate of $f_s = 44.1$kHz. We now do the same procedure for recordings of the same sounds with $f_s = 16$kHz. Using a lower sampling frequency and a same number of samples N=1000, we realize that we use a larger portion of the sound – timewise – at a lower resolution. (Before we used $\frac{1}{44100} \times 1000 = 0.023$s portion, and now we use $\frac{1}{16000} \times 1000 = 0.063$s portion). The evolution of the coefficients for 16kHz sampled signals is in Figure 7.

First thing we observe is that we can predict the signals with higher model orders. Despite having a lower resolution signal (lower f_s), since we now have samples from the signal spread along a longer time period, we could say we have more information about the signal. Therefore, this "more information" lets us represent the signals with more complex (higher order) models.

Before, we also had trouble with the convergence of the quasi-stationary vowel sound of "e". Vowel sounds take a longer time to deliver compared with the consonant sounds, and therefore it makes sense that we would need samples along a longer time period, which we have with 1000 samples sampled at 16kHz. So, we note that in order for a model to converge and predict accurately, we should have samples for a sufficient time period with sufficient resolution. We now observe that the coefficients for signal "e" converge smoothly.

Another thing we observe is the lower prediction gain values for each signal compared to 44.1kHz sampling. This is, as said before, because of the lower resolution of the signals that were processed. Before, we were trying to predict a small chunk of the signal using 1000 samples. Now we're predicting a bigger chunk with the same number of samples which causes the decrease in the prediction gain.

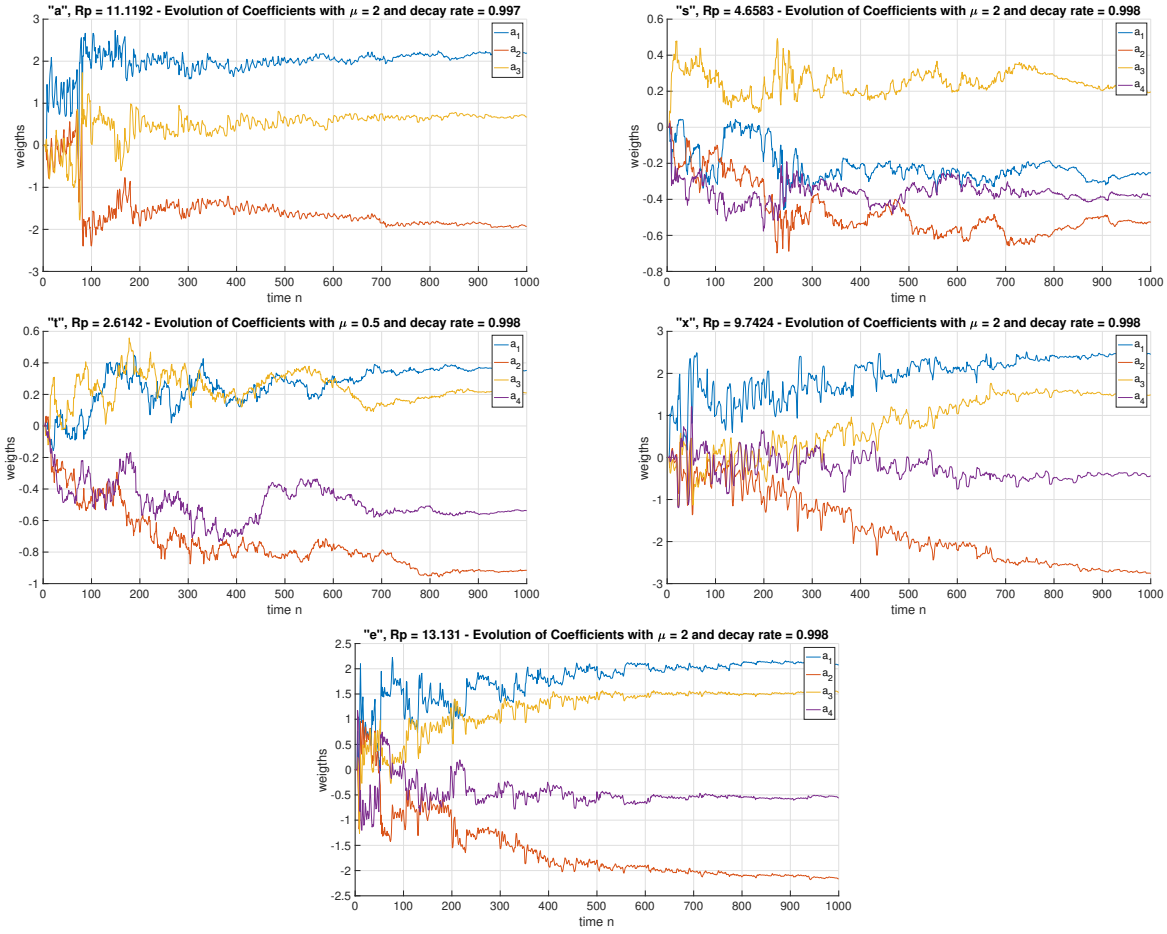


Figure 7: Evolution of the Adaptive Filter Coefficients for different sounds (16kHz sampling rate)

4.6 Dealing with computational complexity: sign algorithms

A simplified version of the LMS algorithm are the sign LMS algorithms:

$$\begin{aligned} \text{signed - error} \quad & \mathbf{w}(n+1) = \mathbf{w}(n) + \mu \text{sign}(e[n])\mathbf{x}(n) \\ \text{signed - regressor} \quad & \mathbf{w}(n+1) = \mathbf{w}(n) + \mu e[n]\text{sign}(\mathbf{x}(n)) \\ \text{sign - sign} \quad & \mathbf{w}(n+1) = \mathbf{w}(n) + \mu \text{sign}(e[n])\text{sign}(\mathbf{x}(n)) \end{aligned}$$

We simulate these algorithms first with the AR parameter identification in 4.4 and then for the recording of "e". The plots of the evolution of the coefficients are given in figures 8 and 9. In the equation for the signed-error update, we see that instead of using the strength of the error signal, we just use its direction for the update. For this reason, even if the error is too small, we could still end up in making large updates. This can be seen in the relevant graph in Figure 9.

A similar argument for the signed-regressor algorithm, it only considers the direction of the input instead of its strength. However, in this case we still consider the strength of the error in the updates, so when it becomes small, steady state is actually steadier. We can observe this in the relevant plot in Figure 9 and observe this algorithm's effect on the fast convergence.

In the audio signal example, the input signal's strength and accordingly the error strength was very low. So, using the sign function increases their value a lot, which makes the updates larger. Therefore, we had to use a much smaller μ value for convergence, and in the sign-sign case, since both the values for $\mathbf{x}$(n) and e[n] are boosted, we had to use even a smaller value – values provided with the titles.

Since the μ value we used for the AR parameter identification was small, and the input and error powers were relatively closer to unity, we didn't have problem with convergence. We even observe that signed-regressor and sign-sign algorithms made the evolution smoother. Overall, the sign LMS algorithms were quite accurate as they yielded very similar results to that of the basic algorithms.

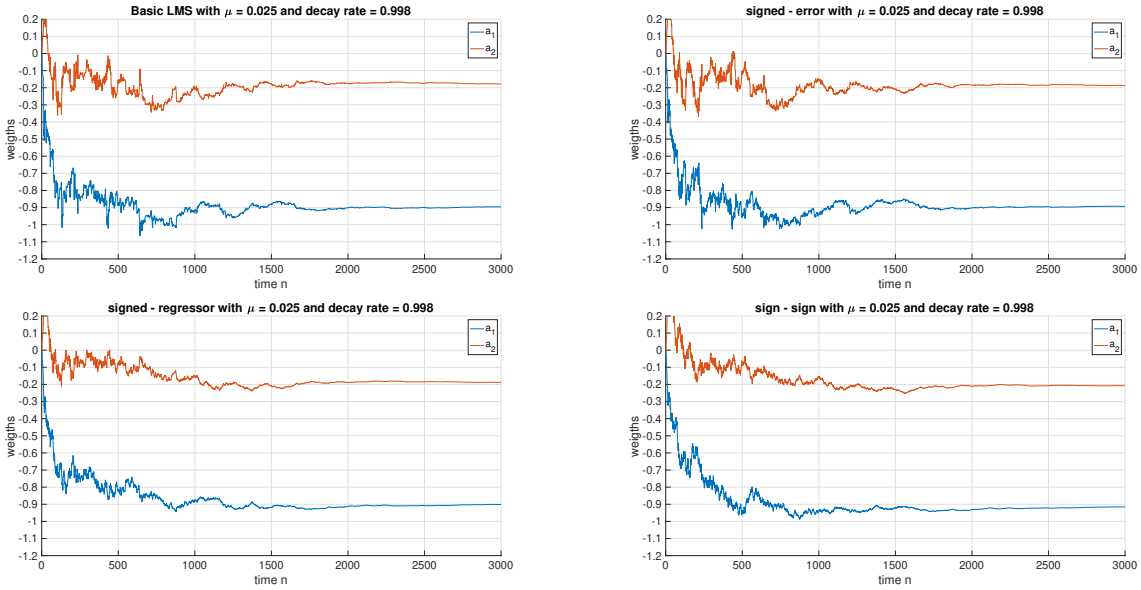


Figure 8: Evolution of Filter Coefficients with sign LMS algorithms (AR parameter identification)

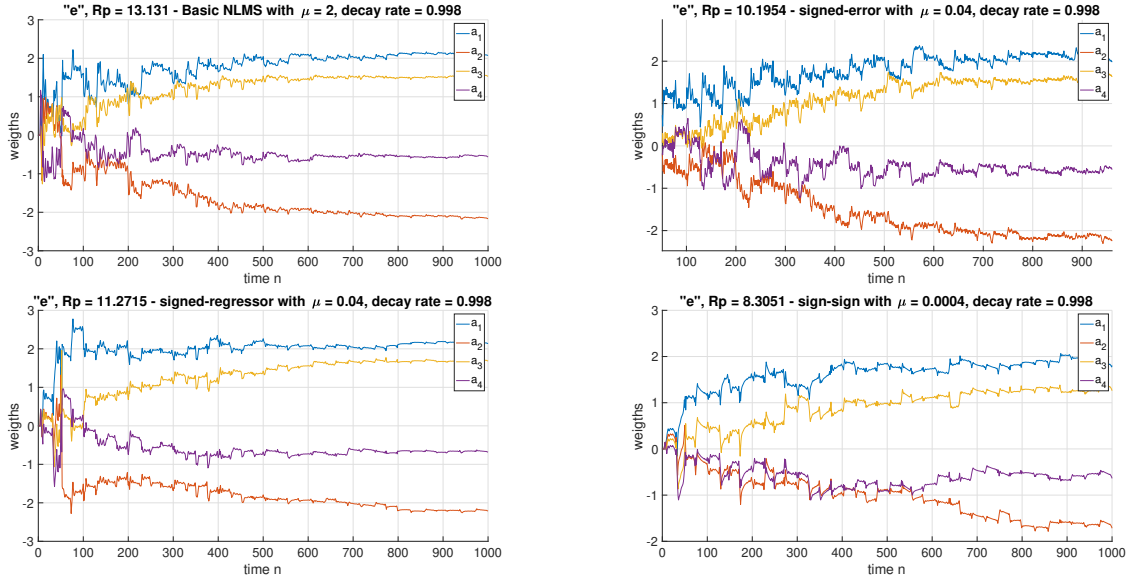


Figure 9: Evolution of Filter Coefficients with sign LMS algorithms (Audio signal "e")

5 MLE for the Frequency of a Signal

(1) The pdf of N sample signal $\mathbf{x}$ parametrized by $\boldsymbol{\theta} = [A, f_0, \phi]^T$ is given as

$$p(\mathbf{x};\boldsymbol{\theta}) = \frac{1}{(2\pi\sigma^2)^{\frac{N}{2}}} \exp\left(-\frac{1}{2\sigma^2}\sum_{n=0}^{N-1}(x[n] - A\cos(2\pi f_0 n + \phi))^2\right) \tag{1}$$

In order to maximize the parametrized probability, we have to minimize the function

$$J(\boldsymbol{\theta}) = \sum_{n=0}^{N-1}[x[n] - A\cos(2\pi f_0 n + \phi)]^2 \tag{2}$$

$$= \sum_{n=0}^{N-1}[x[n] - \underbrace{A\cos(\phi)}_{\alpha_1}\cos(2\pi f_0 n) + \underbrace{A\sin(\phi)}_{-\alpha_2}\sin(2\pi f_0 n)]^2 \tag{3}$$

The function is non-quadratic in A and ϕ. So, we let $\alpha_1 = A\cos\phi$ and $\alpha_2 = -A\sin\phi$ with $A = \sqrt{\alpha_1^2 + \alpha_2^2}$ and $\phi = -\tan^{-1}(\frac{\alpha_2}{\alpha_1})$. Equation 3 becomes

$$J(\boldsymbol{\theta}) = \sum_{n=0}^{N-1}[x[n] - \alpha_1\cos(2\pi f_0 n) - \alpha_2\sin(2\pi f_0 n)]^2 \tag{4}$$

To convert this equation into its vector form, we introduce the vectors of sampled cos and sin – i.e. $\mathbf{c} = [1, \cos(2\pi f_0), ..., \cos(2\pi f_0(N-1))]^T, \mathbf{s} = [1, \sin(2\pi f_0), ..., \sin(2\pi f_0(N-1))]^T$. Then $J(\boldsymbol{\theta}) \equiv$

$$J'(\alpha_1, \alpha_2, f_0) = (\mathbf{x} - \alpha_1\mathbf{c} - \alpha_2\mathbf{s})^T(\mathbf{x} - \alpha_1\mathbf{c} - \alpha_2\mathbf{s}) \tag{5}$$

$$J'(\boldsymbol{\alpha}, f_0) = (\mathbf{x} - \mathbf{H}\boldsymbol{\alpha})^T(\mathbf{x} - \mathbf{H}\boldsymbol{\alpha}) \tag{6}$$

where $\boldsymbol{\alpha} = [\alpha_1, \alpha_2]^T$ and $\mathbf{H} = [\mathbf{c}, \mathbf{s}]$.

(2) We want to minimize equation 6, so we differentiate it with respect to $\boldsymbol{\alpha}$ and equate it to 0.

$$J'(\boldsymbol{\alpha}, f_0) = \mathbf{x}^T\mathbf{x} - 2\mathbf{x}^T\mathbf{H}\boldsymbol{\alpha} + \boldsymbol{\alpha}^T\mathbf{H}^T\mathbf{H}\boldsymbol{\alpha} \tag{7}$$

$$\frac{\partial J'(\boldsymbol{\alpha}, f_0)}{\partial \boldsymbol{\alpha}} = -2\mathbf{H}^T\mathbf{x} + 2\mathbf{H}^T\mathbf{H}\boldsymbol{\alpha} \tag{8}$$

$$0 = -\mathbf{H}^T\mathbf{x} + \mathbf{H}^T\mathbf{H}\boldsymbol{\alpha} \tag{9}$$

$$\mathbf{H}^T\mathbf{H}\boldsymbol{\alpha} = \mathbf{H}^T\mathbf{x} \tag{10}$$

$$\boldsymbol{\alpha} = (\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\mathbf{x} \quad \textbf{(minimizing solution)} \tag{11}$$

Then we can write $J'(\boldsymbol{\alpha}, f_0)$ as

$$J'(\boldsymbol{\alpha}, f_0) = \mathbf{x}^T\mathbf{x} - 2\mathbf{x}^T\mathbf{H}(\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\mathbf{x} + \mathbf{x}^T\mathbf{H}(\mathbf{H}^T\mathbf{H})^{-1}(\mathbf{H}^T\mathbf{H})(\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\mathbf{x} \tag{12}$$

$$= \mathbf{x}^T\mathbf{x} - 2\mathbf{x}^T\mathbf{H}(\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\mathbf{x} + \mathbf{x}^T\mathbf{H}(\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\mathbf{x} \tag{13}$$

$$= \mathbf{x}^T\mathbf{x} - \mathbf{x}^T\mathbf{H}(\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\mathbf{x} \tag{14}$$

so from equation 14, we can say that minimizing $J'(\boldsymbol{\alpha}, f_0)$ is equivalent to maximizing the subtracting term $\mathbf{x}^T\mathbf{H}(\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\mathbf{x}$.

(3) We have $\mathbf{x} = [x[0], x[1], ..., x[N-1]]^T$ and $\mathbf{H} = [\mathbf{c}, \mathbf{s}]$, where $\mathbf{c}$ and $\mathbf{s}$ are column vectors of length N as well. Then we can represent $\mathbf{x}^T\mathbf{H}(\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\mathbf{x}$ as

$$\begin{bmatrix}\mathbf{x}^T\end{bmatrix}\begin{bmatrix}\mathbf{c} & \mathbf{s}\end{bmatrix}\left(\begin{bmatrix}\mathbf{c}^T\\ \mathbf{s}^T\end{bmatrix}\begin{bmatrix}\mathbf{c} & \mathbf{s}\end{bmatrix}\right)^{-1}\begin{bmatrix}\mathbf{c}^T\\ \mathbf{s}^T\end{bmatrix}\begin{bmatrix}\mathbf{x}\end{bmatrix} \tag{15}$$

$$= \begin{bmatrix}\mathbf{x}^T\mathbf{c} & \mathbf{x}^T\mathbf{s}\end{bmatrix}\begin{bmatrix}\mathbf{c}^T\mathbf{c} & \mathbf{c}^T\mathbf{s}\\ \mathbf{s}^T\mathbf{c} & \mathbf{s}^T\mathbf{s}\end{bmatrix}^{-1}\begin{bmatrix}\mathbf{c}^T\mathbf{x}\\ \mathbf{s}^T\mathbf{x}\end{bmatrix} \tag{16}$$

Here we note the orthogonality of the columns of $\mathbf{H}$, $\mathbf{c}$ and $\mathbf{s}$. Doing the following rationale,

$$\sum_{n=0}^{N-1}\cos(2\pi f_0 n)\sin(2\pi f_0 n) = 0 \tag{17}$$

$$\sum_{n=0}^{N-1}\cos(2\pi f_0 n)\cos(2\pi f_0 n) = \frac{N}{2} \tag{18}$$

$$\sum_{n=0}^{N-1}\sin(2\pi f_0 n)\sin(2\pi f_0 n) = \frac{N}{2} \tag{19}$$

for f_0 not close to 0 or 1/2. Then the expression 16 becomes

$$\begin{bmatrix}\mathbf{x}^T\mathbf{c} & \mathbf{x}^T\mathbf{s}\end{bmatrix}\begin{bmatrix}\frac{N}{2} & 0\\ 0 & \frac{N}{2}\end{bmatrix}^{-1}\begin{bmatrix}\mathbf{c}^T\mathbf{x}\\ \mathbf{s}^T\mathbf{x}\end{bmatrix} \tag{20}$$

(4) If f_0 were close to 0, then equation 19 would be ≈ 0 and equation 18 would be $\approx N$. If it were close to 1/2, same thing would happen. For this reason for good estimates, we need f_0 to be away from 0 or 1/2. We can then show that equation 20 becomes

$$\begin{bmatrix}\mathbf{x}^T\mathbf{c} & \mathbf{x}^T\mathbf{s}\end{bmatrix}\begin{bmatrix}\frac{2}{N} & 0\\ 0 & \frac{2}{N}\end{bmatrix}\begin{bmatrix}\mathbf{c}^T\mathbf{x}\\ \mathbf{s}^T\mathbf{x}\end{bmatrix} \tag{21}$$

$$= \begin{bmatrix}\frac{2}{N}\mathbf{x}^T\mathbf{c} & \frac{2}{N}\mathbf{x}^T\mathbf{s}\end{bmatrix}\begin{bmatrix}\mathbf{c}^T\mathbf{x}\\ \mathbf{s}^T\mathbf{x}\end{bmatrix} \tag{22}$$

$$= \frac{2}{N}\mathbf{x}^T\mathbf{c}\cdot\mathbf{c}^T\mathbf{x} + \frac{2}{N}\mathbf{x}^T\mathbf{s}\cdot\mathbf{s}^T\mathbf{x} \tag{23}$$

since $\mathbf{x}^T\mathbf{c}$ is a scalar $\mathbf{x}^T\mathbf{c} = \mathbf{c}^T\mathbf{x}$. (And $\mathbf{x}^T\mathbf{s} = \mathbf{s}^T\mathbf{x}$). Then equation 23 becomes

$$\frac{2}{N}\left[(\mathbf{x}^T\mathbf{c})^2 + (\mathbf{x}^T\mathbf{s})^2\right] \tag{24}$$

$$= \frac{2}{N}\left[\left(\sum_{n=0}^{N-1}x[n]\cos(2\pi f_0 n)\right)^2 + \left(\sum_{n=0}^{N-1}x[n]\sin(2\pi f_0 n)\right)^2\right] \tag{25}$$

to write equation 25 in periodogram notation, we first show that it is equivalent to equation 26.

$$\frac{2}{N}\left|\sum_{n=0}^{N-1}x[n]\exp(-j2\pi f_0 n)\right|^2 \tag{26}$$

$$= \frac{2}{N}\left|\sum_{n=0}^{N-1}x[n]\cos(2\pi f_0 n) - jx[n]\sin(2\pi f_0 n)\right|^2 \tag{27}$$

Getting the absolute value of a complex number is done by:
1. First adding all its real components together and the all its imaginary components together separately.
2. Then by squaring those two sums separately and adding them together.

3. Then by taking the square root of the final sum.

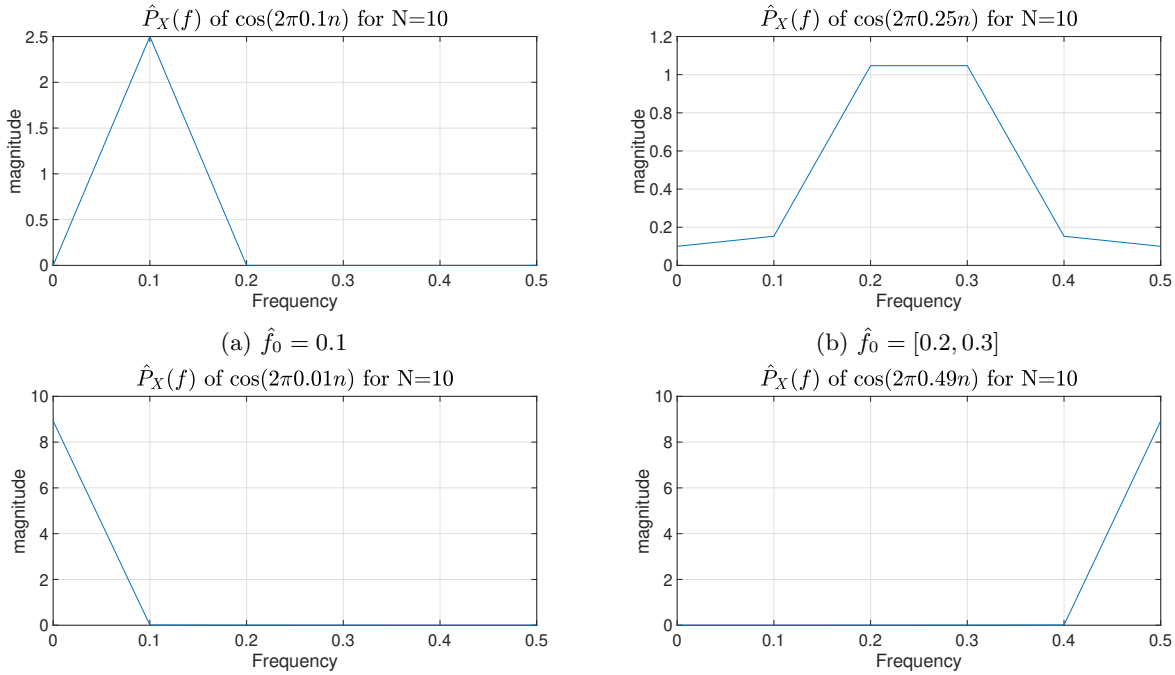
If we follow this, we get that equation 27 is equal to equation 25. By this, we prove that we can write the function we want to maximize as

$$\frac{2}{N}\left|\sum_{n=0}^{N-1} x[n]e^{-j2\pi f_0 n}\right|^2 \tag{28}$$

Our aim is to find the "maximum likelihood estimate" of f_0 which is between 0 and 1/2. To find $\hat{f}_0$ that maximizes equation 28, we can use the periodogram function which was defined by (note the similarity to equation 28)

$$\hat{P}_X(f) = \frac{1}{N}\left|\sum_{n=0}^{N-1} x[n]e^{-j2\pi \frac{f}{N} n}\right|^2 \tag{29}$$

Here $\frac{f}{N}$ represents the normalized value of the frequencies. Therefore, we find $\hat{f}_0$ by finding the f value that maximizes the periodogram and divide it by N.



(a) $\hat{f}_0 = 0.1$

(b) $\hat{f}_0 = [0.2, 0.3]$

(c) $\hat{f}_0 = 0$

(d) $\hat{f}_0 = 0.5$

Figure 1: Periodogram and the MLE estimates for x[n] = cos($2\pi f_0 n$)with varying f_0

We estimate the true value of f_0 when it is equal to 0.1. However, since N = 10, we have low resolution in our estimates. For example when $f_0 = 0.25$, we cannot estimate as accurately as we would wish, but the true value does lie within our estimate interval. When f_0 approaches 0 or 1/2, we observe that the magnitude of the plots increases. The reason this happens is that the other side of the periodogram merges with this side, and as a result, the plots show a higher value. As a note, we add that it can be shown that the Cramer-Rao Lower Bound for estimation of f_0 rises rapidly as it approaches 0 or 1/2.

References

[1] A. Golan, "Ergodicity, non-ergodicity and aging processes."

[2] D. Mandic, "Asp linear stochastic processes," 2020.

[3] ——, "Asp adaptive filters," 2020.

Appendix A Matlab Code for `pdf()`

```
function pdf(x)

num_bins = 100;
num_samples = length(x);

[freq, bin_value] = hist(x, num_bins);
width = bin_value(2) - bin_value(1);

bar(bin_value, freq./(num_samples.*width), 'hist')

xlabel('x')
ylabel('pdf(x)')
title(['pdf estimation of X for N = ' num2str(num_samples)])
```

Appendix B Matlab Code for `pgm()`

```
function Px = pgm(x)

    N = length(x);
    Px = zeros(N, 1);

    for f = 0:N-1
        s = 0;
        for n = 0:N-1
            s = s + x(n+1) * exp(-1j*2*pi*(f)*(n/N));
        end
        tmp = (abs(s)^2) / (N);
        Px(f+1) = tmp;
    end
end
```

Appendix C Matlab Code for `lms()`

```
function [yhat, e, W] = lms(x, z, mu, p)

w = zeros(p, 1);
W = zeros(p, p);

N = length(x);

yhat = zeros(N, 1);
e = zeros(N, 1);
for n = p:N

    x_tmp = flip(x(n-p+1:n));
    yhat(n) = dot(w, x_tmp);

    e(n) = z(n) - yhat(n);

    mu_tmp = mu / (dot(x_tmp,x_tmp) + 0.001);

    w = w + x_tmp.* (mu_tmp * e(n));
    W = [W w];

    mu = mu*0.99;
end

end
```