Part IIA Project - SF1: Data Analysis - Final Report

Bailey Brookes — Corpus Christi — bdb31

February 5, 2019

Abstract

Data analysis has many key aspects including spectral analysis, noise reduction, parameter estimation and the reconstruction of data. There are several techniques that can be applied to these different signal processing tasks and these are investigated, applied to audio and discussed throughout the project. This report provides results to questions and tasks given in the handout, as well as explanations key interesting results.

Contents

1 Introduction

Digital signal processing (DSP) has numerous aspects. Spectral analysis, filtering, noise reduction, estimation and reconstruction all fall under the remit of DSP and each has there own techniques an phenomenon. It is therefore important to understand the techniques, how to implement them and optimize them, as well the phenomena caused by some techniques used.

The goal of the project was to learn about many techniques in DSP and implement them in code, then applying them to real life audio. While many techniques, such as filtering and spectral analysis, have functions that carry out the task it is important to understand how these functions work, their limitations and functions and techniques can be combined to improve results.

This report contains summaries of the first 2 weeks of the project and a more detailed look at the final two weeks. The report covers techniques of Fourier transforms, windowing, noise reduction, parameter estimation and signal reconstruction, all with an audio focus. The results are presented with a discussion on why they are significant. Where problems came up, they are addressed and explained. Conclusions are then drawn from the whole project.

For detail on the theory and task carried out in the project, see the SD1: Data Analysis £rd Year project handout.

2 Summary of previous work

2.1 Week 1 - Spectrum Analysis

Week 1 focused on spectrum analysis and basic digital signal processing techniques, which where then applied to real audio files. The following was learnt from week 1:

- The Fast Fourier transform (FFT) is much faster than the Discrete Fourier transform (DFT) as it makes use of redundancy and recursion to more efficiently compute the transform.
- The FFT is affected my many parameters of the input data sequence:
  - The resolution of the FFT increase with the length of the data sequence. This means zero padding (adding zeros to the end of the data) can be used to increase

resolution and efficiency if the padding makes the data length a power of 2.

  - Windowing removes and sharp cut-offs when a proportion of the data is taken for the FFT, reducing the effect of spectral leakage.

  - Amplitude modulation has different effects depending on the type of modulation used, with random noise having the same effect as additive noise, a linear increase having spectral leakage and periodic modulation having peaks at the signal and modulation frequencies.

- Spectrum analysis techniques where then used on music and speech data to compare and constant single notes and transients in music, and vowels and consonants in speech.

- All these techniques where then applied to a organ waveform to try and identify the notes present.

For more detailed explanation of week 1, see 'Part IIA Project - SF1: Data Analysis - First Interim Report'

2.2 Week 2 - Noise Reduction

Week 2 used the techniques of week 1 and the Wiener filter theory and applied it to noise reduction. The following was learnt from week 2:

- The filter `overlap_add.m` was analyzed, and technique of 'overlap add' was look as specification for a tool to be used later in developing a noise reduction filter.

- A noise reduction filter using the Wiener filter with overlap add was developed and tested on audio files. To filter however, the noise added to the speech had to be estimated. Two methods of noise estimation where used:

  - Using a silent part of the speech to estimate to noise floor.

  - Subtracting a smoothed version of the signal from the original in each frame and using that in each frame as the noise. This method results in more digital distortion (caused by filter coefficients being zero resulting in sharp jumps) but is more robust as doesn't require a silent portion length hard coded into the filter and for some files no silent portions exist.

- One big problem with the Wiener filter is the digital distortion caused by filter coefficients being 0 and sharp jumps ensuing. This was decreased by zero padding each frame to increase the resolution o the frequency-domain filter but did add more processing time as more padding was used. Hence there was a trade off of time vs sound quality.

- Mean Squared Error (MSE) serves as a good figure of merit for a filter but doesn't account well for the introduction of digital distortion.

For more detailed explanation of week 1, see 'Part IIA Project - SF1: Data Analysis - Second Interim Report'

3 Parameter Estimation

3.1 Simple linear model

For the simple linear model, a constant with additive Gaussian white noise $y = \theta + e_n$ where $e_n = N(e_n; 0, 1)$ meaning $y = N(y; \theta, 1)$. The Maximum likelihood (ML) and maximum a posterior (MAP) can either be computed by putting the following equation into matrix form with generator matrix G a column matrix of 1s and then using the forms of the estimates given in the handout, or by directly forming the likelihood and posterior functions and maximization, which is done here.

3.1.1 ML estimate

The likelihood function, $p(\mathbf{y}|\theta)$ is given by the product of each data point as the data is independent identically distributed.

$$p(\mathbf{y}|\theta) = \prod_{i=1}^{N} p(y_i|\theta) = \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi\sigma_e^2}} e^{\frac{1}{2\sigma^2}(y-\theta)^2}$$

Maximizing the likelihood function is the same as maximizing the log likelihood so by taking logs and differentiating with respect to θ and setting the result equal to zero, the ML estimate of θ is obtained:

$$L(\mathbf{y}|\theta) = \log\left(\prod_{i=1}^{N} p(y_i|\theta)\right) = \log\left(\prod_{i=1}^{N} \frac{1}{\sqrt{2\pi\sigma_e^2}} e^{\frac{1}{2\sigma_e^2}(y_i-\theta)^2}\right)$$

$$\therefore L(\mathbf{y}|\theta) = -\frac{N}{2}\log(2\pi\sigma_e^2) - \frac{1}{2\sigma_e^2}\sum_{i=1}^{N}(y_i - \theta^2)$$

$$\frac{dL(\mathbf{y}|\theta)}{d\theta} = \sum_{i=1}^{N}(y_i - \theta) = \sum_{i=1}^{N} y_i - N\theta = 0$$

$$\therefore \theta^{ML} = \frac{\sum_{i=1}^{N} y_i}{N}$$

That is, the ML estimate for θ is the mean of the data set y.

3.1.2 MAP estimate

To compute the map estimate, the likelihood function must be multiplied by a prior of θ. Throughout this report, we will assume a Gaussian prior on θ for simplicity:

$$\theta = N(\theta; m_\theta, \sigma_\theta^2)$$

Where m_θ, σ_θ^2 are the prior mean and variance on the θ parameter respectively. The MAP estimate is compute my forming the posterior density and then maximizing it in the same way as the ML estimate: differentiating and setting to zero.

$$p(\theta|y) \propto p(y|\theta)p(\theta) = \prod_{i=1}^{N} \frac{1}{2\pi\sigma_e^2} \frac{1}{2\pi\sigma_\theta^2} e^{-\frac{1}{2\sigma_e^2}(y_i-\theta)^2} e^{\frac{1}{2\sigma_\theta^2}(\theta-m_\theta)^2}$$

$$L(\theta|y) = \sum_{i=1}^{N} -\frac{1}{2\sigma_e^2}(y_i-\theta)^2 - \frac{1}{2\sigma_\theta^2}(\theta-m_\theta)^2 + Constant$$

$$\frac{dL(\theta|y)}{d\theta} = \sum_{i=1}^{N} y - N\theta + \frac{\sigma_e^2}{\sigma_\theta^2}\Bigg(N\theta + Nm_\theta\Bigg) = 0$$

$$\therefore \theta^{MAP} = \frac{\frac{\sum_{i=1}^{N} y}{N} + m_\theta \frac{\sigma_e^2}{\sigma_\theta^2}}{1 + \frac{\sigma_e^2}{\sigma_\theta^2}}$$

As the variance tends infinite, meaning the prior becomes spread out and each value of θ is equally likely, the MAP estimate becomes the ML estimate.

The true posterior probability is normalized by $p(x)$ which is found by using the sun=m rule of probability. However it is easier to calculate by using Matlab to normalized the posterior for us which gives the following result:

$$p(\theta|y) = N\Big(\theta; \theta^{MAP}, \frac{\sigma_e^2\sigma_\theta^2}{N\sigma_\theta^2 + \sigma_e^2}\Big)$$

Compared to the result for the likelihood:

$$p(y|\theta) = N(y; \theta^{ML}, \sigma_e^2)$$

3.1.3 Investigating these estimates

The Matlab script `Linear_model.m` investigates both of these estimates for different data length, noise variances and prior distributions on θ.

For small, medium and large N, the likelihood of θ remains unchanged but or larger N, the posterior get sharper as the variance of the estimate gets smaller. This is shown if Figures 1,2 and 3. The decrease in variance is obvious from the equation for the posterior distribution as as N grows large, the denominator of the variance grows as well shrinking the variance.

For increases in the error variance, both the ML and MAP estimates variance increase and the pdfs spread out as there is now more uncertainty in the measurement. This is shown in Figures 6 (small error variance) and 7 (high error variance) For MAP, this can be combated by using more data samples are discussed above.

Changing the prior also has an effect on the estimate. Figures 4 and 5 show that the posterior varies with the prior as expected. Priors with small variance and incorrect means bias the posterior away from the true parameter, a problem that can be fixed with a larger variance or more data points.

3.2 Linear Model with trend

Now with two parameters a multivariate model must be used, meaning that first the linear model of the model must be formed, and then the estimates of these parameter can be found using the results in the handout. The generator matrix G is:

$$G = \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ \vdots & \vdots \\ 1 & N-1 \\ 1 & N \end{bmatrix}$$

And for simplicity we will assume that the θ parameters are independently normally distributed with the following parameters:

$$\theta = \begin{bmatrix} a \\ b \end{bmatrix} \text{ with mean } \mathbf{m}_\theta = \begin{bmatrix} m_1 \\ m_2 \end{bmatrix} \text{ and covariance matrix } \mathbf{\Sigma}_\theta = \begin{bmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{bmatrix}$$

The ML and MAP estimates are then computed as they are in the handout. The Matlab script `Linear_trend.m` investigates this. Some examples are tabulated below.

Table 1: Results from running `Linear_trend.m` for different priors for N = 100

Paramenters		Prior		ML Estimate		MAP Estimate	
DC	Trend	m_θ	σ_θ^2	DC	Trend	DC	Trend
1	2	1,2	1,1	0.85898	2.00135	0.86448	2.00127
1	2	2,1	1,1	0.96389	1.99852	1.00491	1.99791
1	2	5,5	0.1,0.1	0.94501	1.99981	2.10306	1.98261
1	2	5,5	100,100	0.98204	2.00121	0.98365	2.00119

3.3 Model Choice

Here 3 models are considered: Pure noise, a constant plus noise and a linear trend plus noise. The script `Model_choice.m` can generate data from either model, work out the MAP estimates for the parameters and also the marginal likelihoods of each of the models. This assumes the modes,a re all equally likely, a useful assumption for later use of this script. Some example results are tabulated below.

Table 2: Results from running `Model_Choice.m` for data generated from 3 different models and N = 10

Model	True Paramters values	Model 1	Model 2		Model 3		
		$p(M_1\|y)$	$p(M_2\|y)$	DC	$p(M_3\|y)$	DC	Trend
1	N/A	1.45E-06	2.65E-07	-0.0255	3.00E-09	0.028	-0.0107
2	DC = 2	1.18E-08	5.61E-07	1.1332	1.31E-04	0.7507	0.0583
3	DC = 1, Trend = 2	0	0	11.2379	2.47E-09	0.7065	2.0881

Many factors can effect the resulting with some many different parameters that can be varied including number of data points, the effect of priors and the effect of more/less noise. For coincidence full investigation into each of these is not fully described. The following is a summary that can be confirmed by running `Model_Choice` and changing the parameters:

- Increasing the variance of the noise does not affect the model probabilities as each estimate and model probability is affected by the noise by the same amount, as can be seen by the equations that calculate the model probabilities.
- Choice of prior is crucial. For example, if the prior for model 2 is set to $p(\theta) = N(\theta; 1, 1^2)$ then this is similar to model one, which is noise with mean zero. This can lead to missleading model probabilities and hence model choice.
- One interesting thing to note is that the results for the marginal likelihoods $p(x|M_i)$, which can be sued interchangeably with the posterior $p(M_i|x)$ as long as the models are equally likely and the priors on θ are Gaussian, produce very small values for even moderate values of N (for example 10). This is an examples of Lindley's paradox, where the frequentist and Baysian approach to model selection yield conflicting results.

3.3.1 Lindelys paradox

The paradox is best described by example, such as that given in [1]. Suppose an observer measures the position of a bead in the course of a biophysics experiment. A perturbation is applied and N measurements are made of the bead position. The N measurements are assumed to be independent and identically distributed (iid) in a normal distribution centered on the unknown true displacement μ with known variance σ 2 where $\mu = 0$ if the bead is unmoved and $\mu \neq 0$ otherwise.

In the Bayesian paradigm, we must specify priors over the parameters for the two models. Model zero (null hypothesis) is parameter free since $\mu = 0$,but model one (alternative hypothesis) is parameterized by unknown mean μ.

The true value μ_0 is unknown and to represent this ignorance, we use a vague conjugate prior, choosing a normal prior centered on zero with a large variance τ^2. As in the model selection above, we assume the competing models have equal prior probability. The model with the largest posterior probability is then selected as the true model. The experimental resolution for detecting a change in the bead position is then:

$$|\hat{\mu}| > \sigma_\mu \sqrt{\frac{2 \log \tau}{\sigma_\mu}}$$

while the frequentist rule of thumb (95% confidence level) for rejecting the null hypothesis is:

$$|\hat{\mu}| > 2\sigma_\mu$$

where $\sigma_\mu = \frac{\sigma}{\sqrt{N}}$ is the uncertainty in μ. The difference between the conditions defined by these 2 equations reveals that the paradigms may come to conflicting conclusions about model selection, as illustrated . Lindley emphasized this conflict by describing the following scenario: If the alternative hypothesis is true, for a suitable choice of τ and sample size N, the null hypothesis could be simultaneously rejected at a 95% confidence level and have 95% posterior probability [2]. This conflict between statistical paradigms has been called the Lindley paradox. and is caused by the two different methods being proportional and inversely proportional to the sample size N.

Applied to the model selection, this means that with more data points we can predict the model well 'by inspection' (is there a linear increase, is the noise centres around the mean or is it random) but the analytical Bayesian method produces smaller and smaller probabilities of each model, while with few data points its hard to see the model but the model probabilities give a good indication of the correct model.

The effect is demonstrated in Figure 8. The plot on the right is clearly a linear trend with trend value around 2 and mean (initial value) around 2, but all model probabilities are 0, while the left plot shows the same parameters but with fewer data points, where the model probabilities clearly show that it is a linear trend model.

3.4 The Auto-regressive (AR) Model

Auto-regressive (AR) data is one in which the next data points depends on a weighted sum of the previous. That is:

$$x_n = \sum_{i=1}^{P} a_i x_{n-i} + e_n$$

Where a_i are the filter coefficients of the all-pole filter and are referred to as the AR parameters. P is the number of coefficients and it known as the order of the AR process and e_n is the estimation error, model as AWGN as with all models in this project.

The AR process can be views as an all pole filter, with transfer function:

$$H(z) = \frac{1}{1 - \sum_{i=1}^{P} a_i z_{-i}}$$

AR data can then be generated by passing random Gaussian noise through the filter. The Matlab script `AR_gen.m` does this and some examples of this are plotted in the appendix, along with there power spectral density (PSD) and pole locations of the filter.

The PSD of an AR filter is given by:

$$S_x(e^{j\Omega}) = |H(e^{j\Omega})|^2 \sigma_e^2 = \frac{\sigma_e^2}{|1 - \sum_{i=1}^{P} a_i e^{-j\Omega i}|^2}$$

Where peaks in the PSD are at the normalized frequency equal to the argument of the poles, as inputs near this frequency will resonant with the poles. This also means that poles on or outside the unit circle are unstable and will cause he AR process to row without bound, as seen in Figure 10 where one of the pole magnitude is 1.005. Therefore when choosing an AR model, it is important the poles of the filter are stable (within the unit circle).

The magnitude squared the the filterers discrete Fourier transform (DFT) gives a good approximation of the PSD, as explained by the Einstein-Wiener-Khinchin Theorem which is fully derived in [3]:

$$\lim_{N \to \infty} \mathbb{E}\left(\frac{1}{2N+1} |X^N(e^{j\Omega})|^2 \right) = S_x(e^{j\Omega})$$

An example of this is shown in Figure 11

3.4.1 Parameter Estimation for AR

The same results for the ML and MAP estimates can be used for the AR model, meaning the same code can be used. All there is to do is form the linear model, specifically the generator matrix for an AR model where x is zero before the first sample is:

$$G = \begin{bmatrix} 0 & 0 & \cdots & 0 & 0 \\ x_1 & 0 & \cdots & 0 & 0 \\ x_2 & x_1 & \cdots & 0 & 0 \\ \vdots & & \ddots & & \vdots \\ x_{N-1} & x_{N-2} & \cdots & x_{N-P} & x_{N-P-1} \end{bmatrix}$$

3.4.2 Model Selection for AR

Now the AR model is in linear form, model selection can be done again. If ML estimation is used, model selection is only choosing the appropriate order of AR model. For MAP, this can be further tuned by selecting both an appropriate order and set of parameters.

Figure BLANK shows an original and reconstructed data of the music file using an AR(15) model. Speech and music can usually be approximated with model orders of around 10-20, as shown by the plot. The PSD plot in Figure BLANK also shows this.

When the estimator reaches the models true order, there is a sharp decrease in the mean squared error (MSE) as shown in Figure 12. It then stays roughly constant. This can be used to indicate the true model order as although a higher order results in a slightly smaller MSE, it can result in over fitting.

3.4.3 Predicting Missing data

When data is sent over a channel, it is possible that packets of data are lost. These can be modeled as zeros in the data. The AR model can be used to predict the values of the missing data by predicting the AR parameters from the data just before it which should be similar to the missing data. Then the model is run in forward and backward (anti-casual) prediction mode an then the prediction is a weighted up of the two prediction (see handout for full explanation).

The script `AR_Packets.m` attempts to do this for small amounts of missing data, and some sound files of the results are attached. The appendix shows plots of the original and predicted data. One issue during implementation was that despite the AR model predicting similar parameters for data around the gap to that of the overall sequence, the prediction tented to grow to huge values or shrink to tiny values unless a higher than usual model order was selected. Speech can usually be models with an AR process with an order of around 10 to 20, but varying model order had to be used to model speech and music correctly.

Attached are some sound files of attempts at reconstructing missing data. Some plots are also attached in the appendix for visual aid.

Also attached is an attempt at the changeling examples. Due to the problems faced in large values in the simple example, the code whilst able to find the missing packets and produce an estimate, produced an incorrectly large estimate for the missing data.

4 Conclusions

Spectrum Analysis

- FFT is more efficient for compute the Fourier transform then the DFT due to use of the DFT redundancy.
- The quality of the frequency response is covered by its input data length (which decided its resolution and can be improved by zero padding), windowing to avoid spectral leakages, and amplitude modulation which introduce additional frequency artifacts.
- Good spectral analysis technique are need to accurately extract vowels and consonants in speech, notes/transients in music and to identify notes present in a complex music file.

Noise Reduction

- Wiener filtering is a filtering technique grounded in a theoretical basis. It is optimum for minimizing the mean squared error between the clean signal and the filtered signal.
- The addition of zero padding the signal before filtering, therefore increasing its resolution, reduces the amount of digital artifacts.

Parameter Estimation

- MAP and ML estimate formulate using the linear model are versatile and can be applied to a number of data sets including simple trends and AR models.
- Choices on priors is important to ensure MAP estimates are not biasing the posterior estimate in the wrong direction. This can be fixed by using a larger number of data points N.
- Lindelys paradox tells us that the Bayesian and frequensit model choices can contradict each other, as model probabilities decrease with the number of data points but the confidence interval increases.
- Model selection for AR models is done in the same way as for linear models, with AR parameters values and model orders being the parameters.
- The MSE of the model estimate decreases dramatically at the true model order, which can be used to indicate what model to chose, avoiding over-fitting that may be caused with higher order models.
- An Attempt was made to use AR modeling on speech and music to reconstruct missing data. While to code was written and work for small amounts of missing packets, it produced large estimates for the challenging examples. Both myself and demonstrators couldn't find the source of this error and this is a section that requires further work.

References

[1] Colin H. LaMont, Paul A. Wiggins *The Lindley paradox: The loss of resolution in Bayesian inference*. University of Washington, 16 Aug 2017.

[2] D. V. Lindley. *A statistical paradox. Biometrika*, 44(1/2):187–192, 1957.

[3] S. Godsill, *3F3 - Detection, Estimation and Inference for Signal Processing Part II - Estimation Theory and Inference Lecture Notes* Signal Processing and Communications Laboratory, Engineering Department, Cambridge, UK. Michaelmas 2016

A Final Report Appendix

A.1 Simple Linear Model

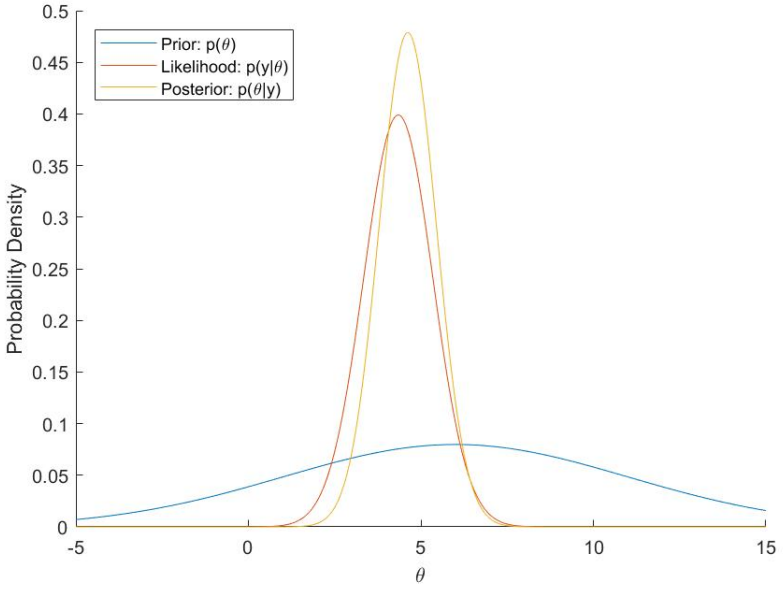

Figure 1: PDFs of the prior, likelihood and posterior for N = 1 data points

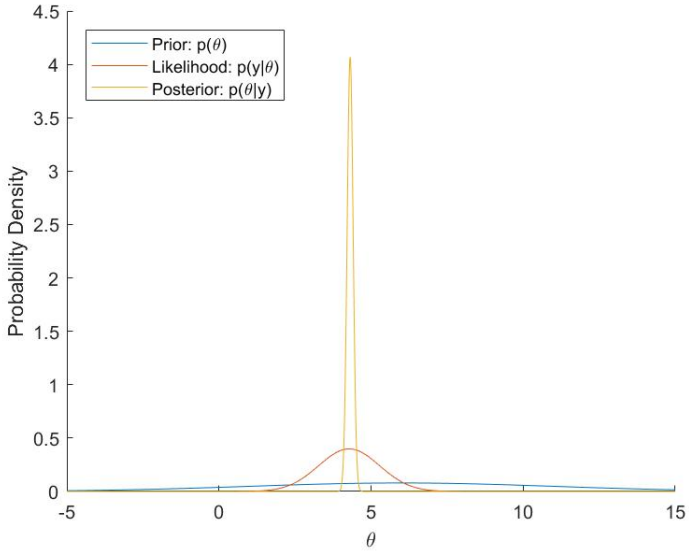

Figure 2: PDFs of the prior, likelihood and posterior for N = 10 data points

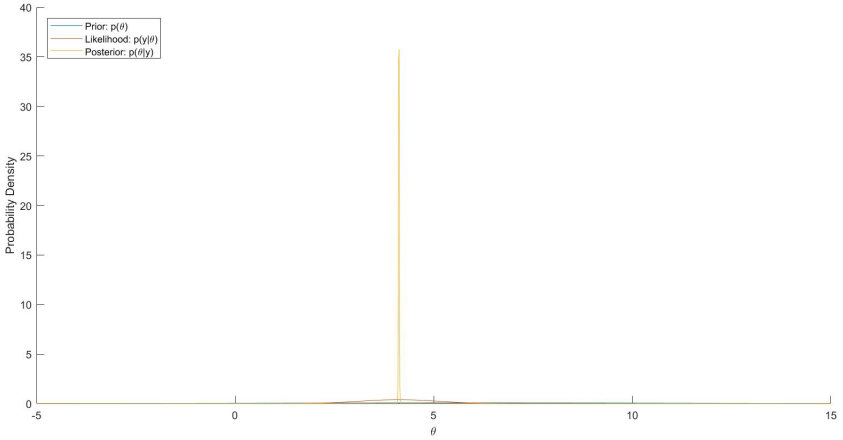

Figure 3: PDFs of the prior, likelihood and posterior for N = 100 data points

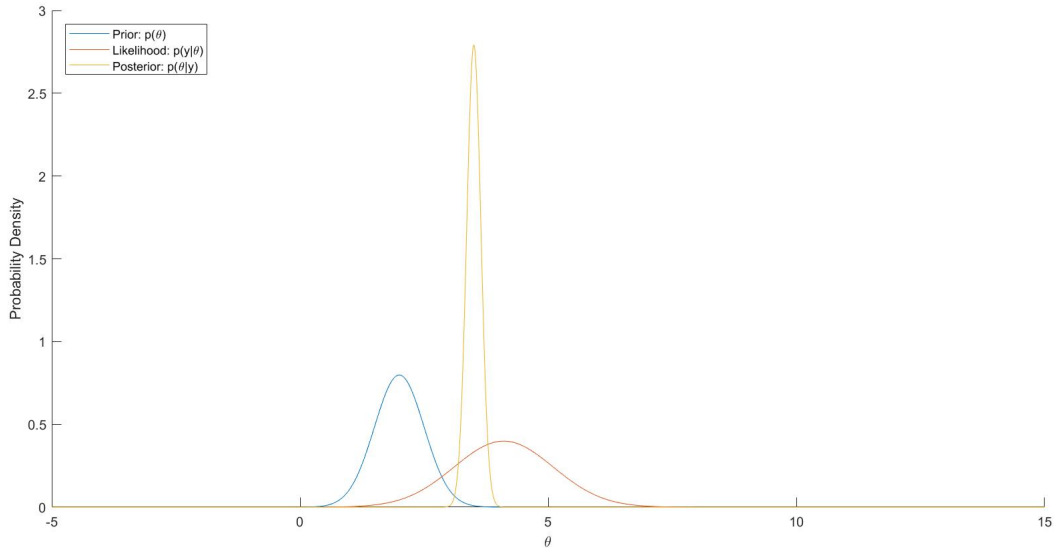

Figure 4: PDFs of the prior, likelihood and posterior for with a prior $N(\theta; 2, 0.5^2)$

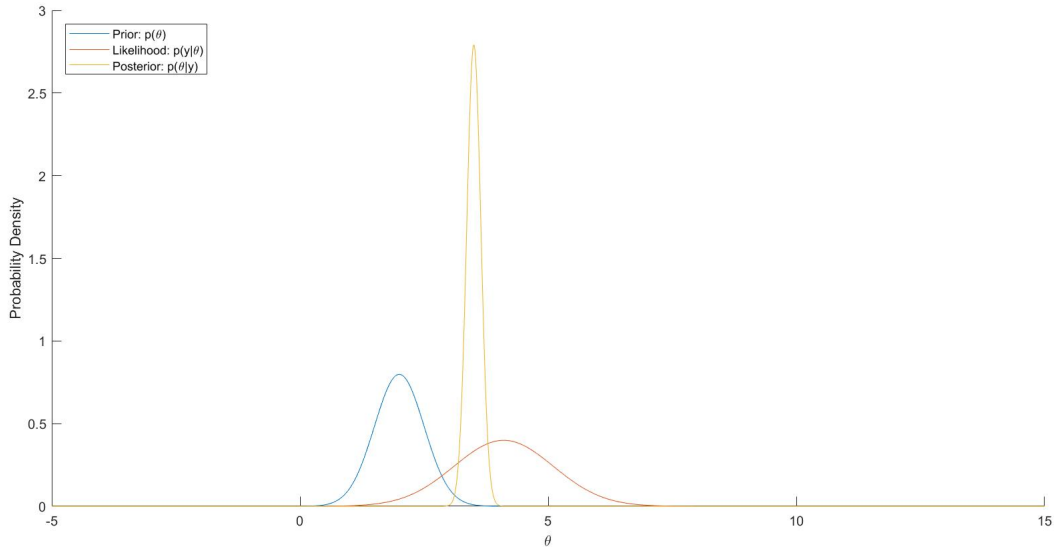

Figure 5: PDFs of the prior, likelihood and posterior for with a prior $N(\theta; 5, 3^2)$

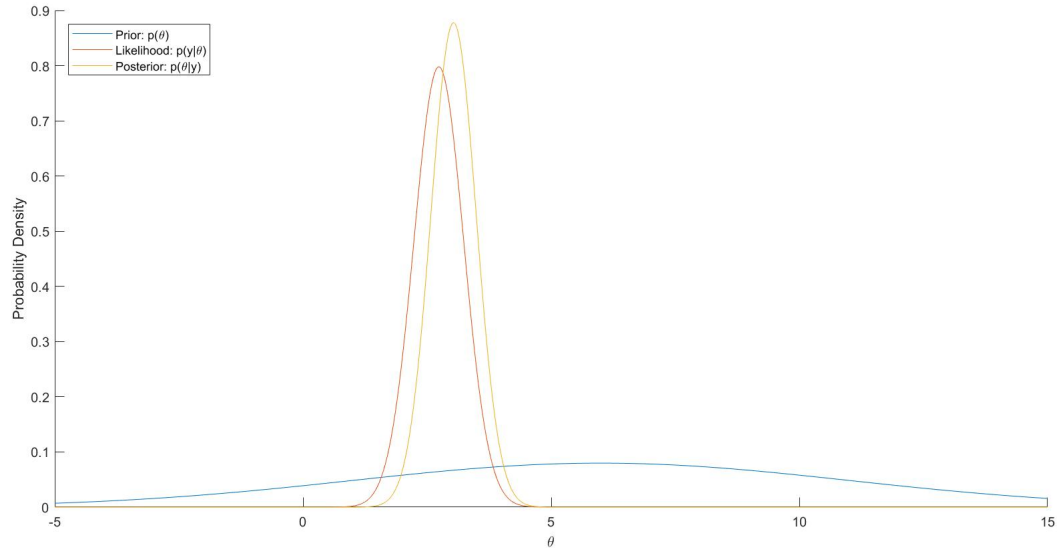

Figure 6: PDFs of the prior, likelihood and posterior for $\sigma_e = 0.5$

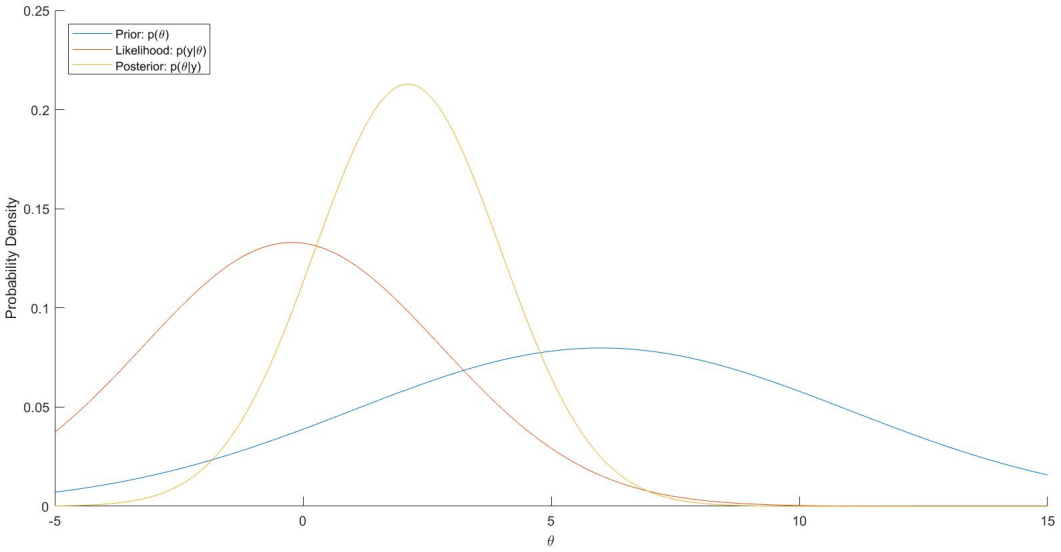

Figure 7: PDFs of the prior, likelihood and posterior for $\sigma_e = 3$

A.2 Lindelys Paradox

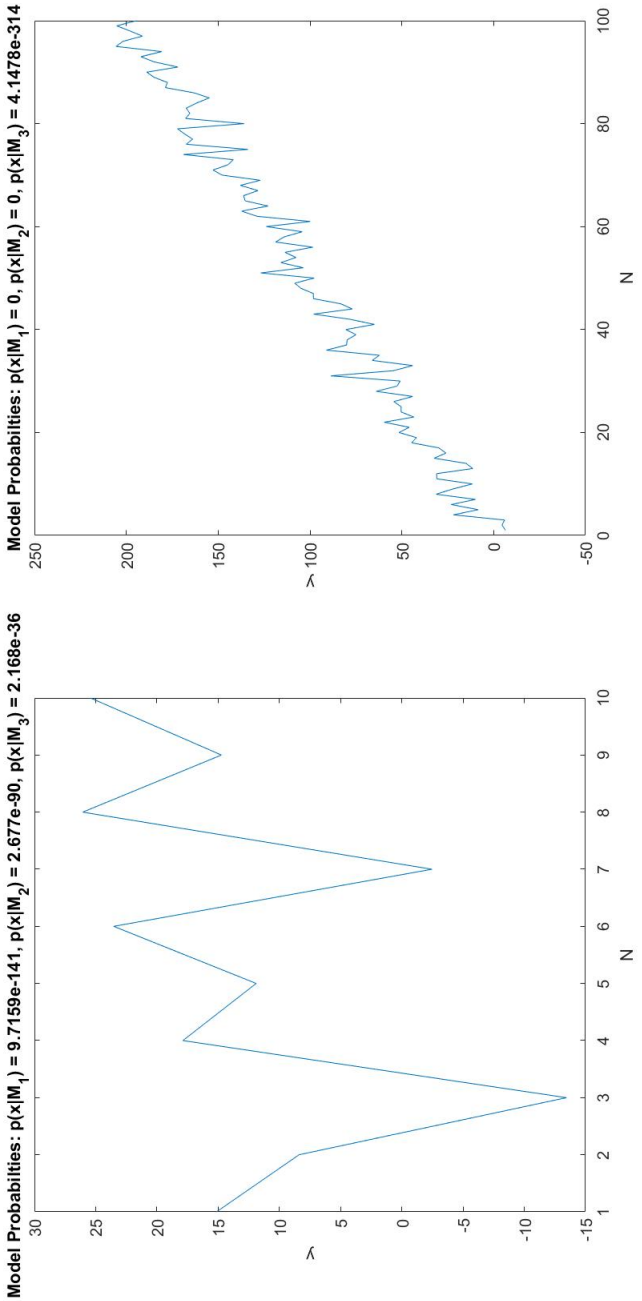

Figure 8: Plot of the linear trend model and the model likelihood probabilities (M_1 is random noise, M_2 is the linear model and M_3 is the linear model with trend) for differ N to demonstrate lindleys paradox

A.3 AR Model

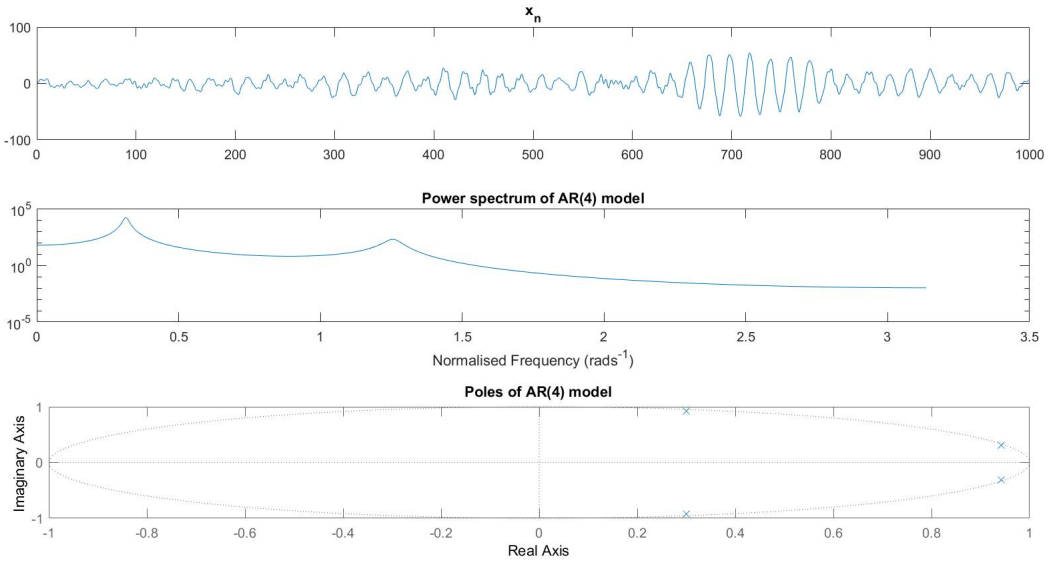

Figure 9: An AR(4) process, its PSD and the location of its poles

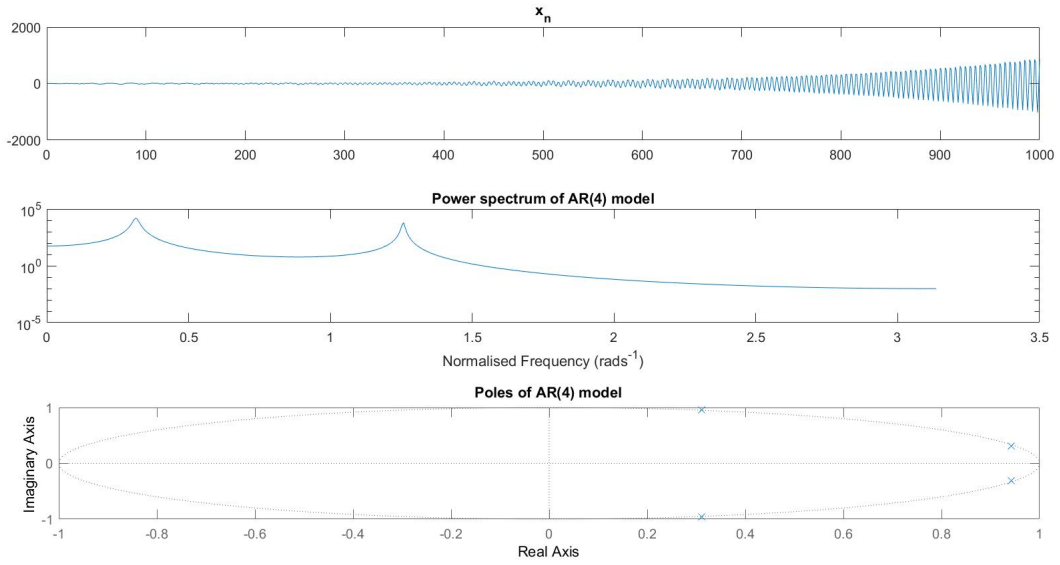

Figure 10: An unstable AR(4) process, its PSD and the location of its poles

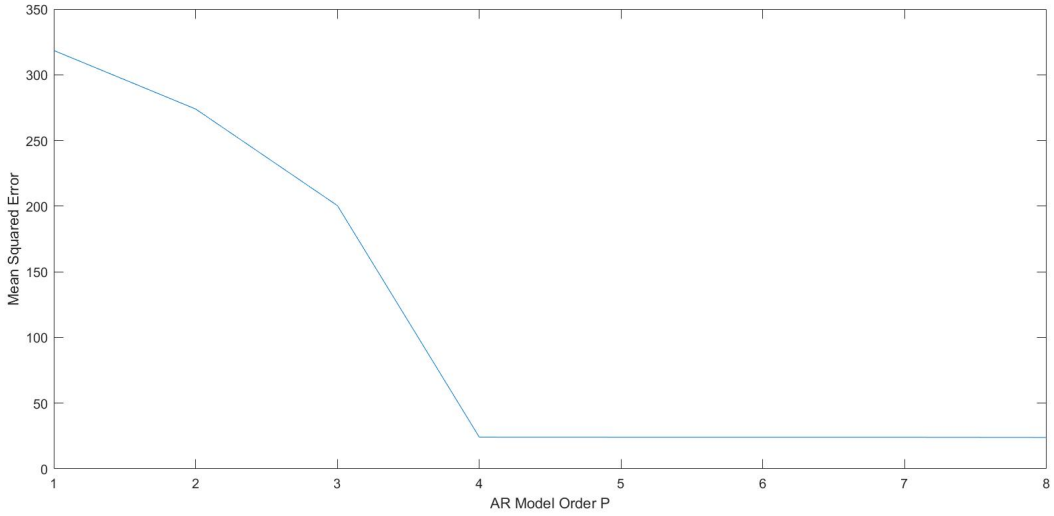

Figure 12: Mean squared error plot for the AR order used to model the data, showing the sharp decrease at the true model order

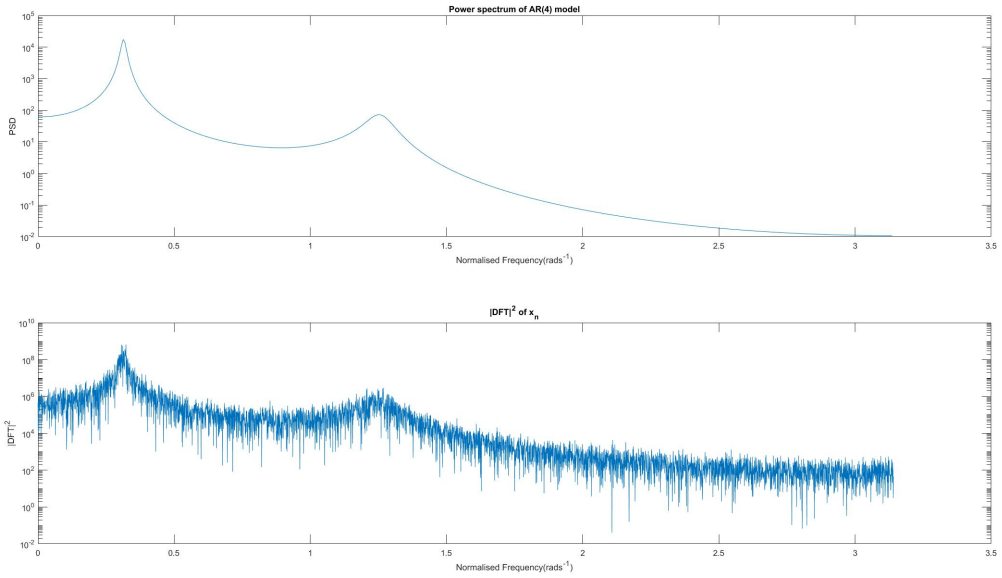

Figure 11: Plot of the PSD and magnitude squared of the DFT of an AR(4) process

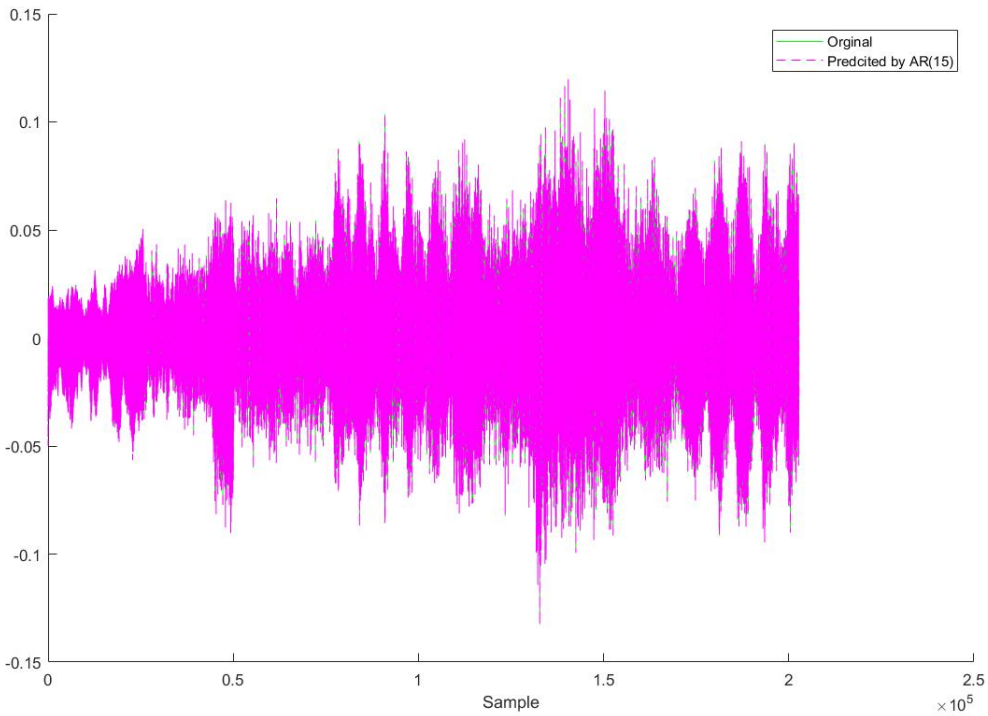

Figure 13: Plot showing the original and reconstructed signal using an AR(15) model

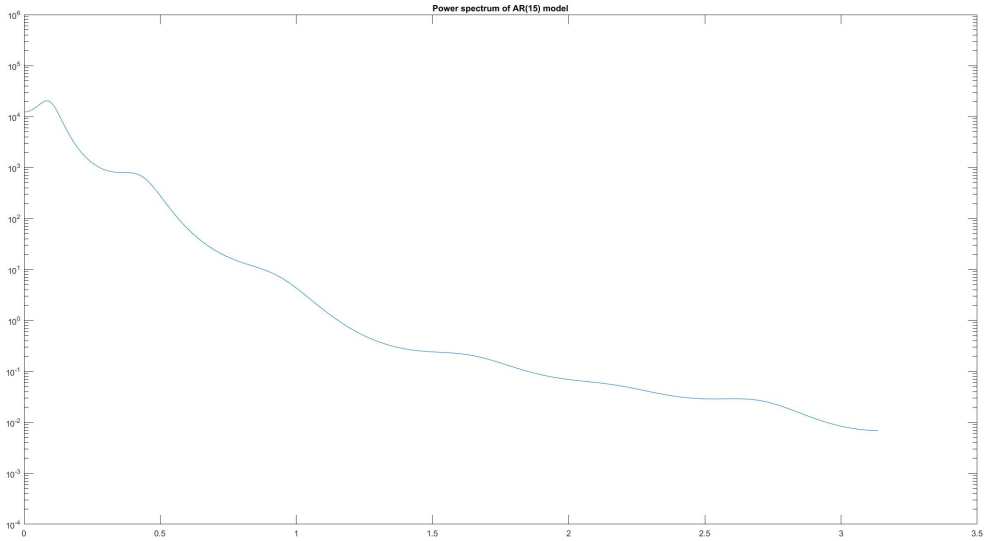

Figure 14: PSD or predicted AR(15) model

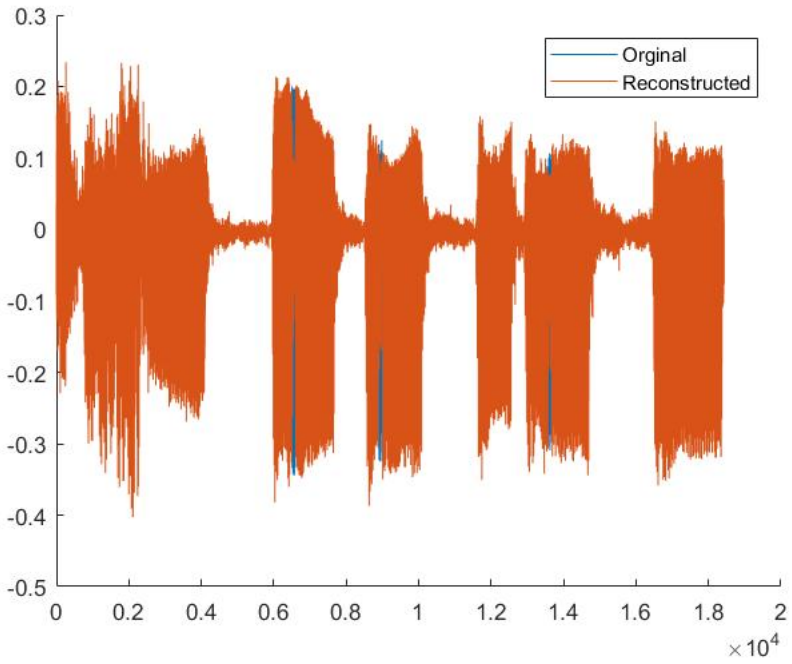

Figure 15: Reconstruction of audio with missing sections

B First Interim Report

B.1 DFT vs FFT

While the DFT is fairly simple to compute, it has redundancy in it which causes it to be slow. It calculates the same exponential base (weights) numerous times. The FFT uses this recursion and redundancy to speed up the process. While the DFT requires $2N + 2N$ operations to calculate each point X_p and $X_{p+\frac{N}{2}}$, giving it complexity $O(N^2)$, the FFT has $2N + 2 + 2$ operations, meaning it has complexity $O(N \log_2 N)$. Figure 16 shows how the reduction in complexity results in a much faster computing time for longer data length.

B.2 Effects on the FFT

B.2.1 Data Length

FFT resolution increases with data length N and there are more frequency bins meaning a smaller frequency difference between bins. This increase in sharpness in shown in Figure 17. This is why signals are sometimes 'zero padded'. Adding zeros to the end of the signal achieves two things:

1. By adding enough zeros to make the data length a power of two, the FFT becomes much more efficient.

2. The greater the length of data, the higher the resolution.

Zero padding also does not change the frequency content of the signal, making this technique extremely useful.

B.2.2 Windowing

Windowing reduced the effect of spectral leakage caused by the sharp cut off caused by finite sampling. Figure 18 shows this leakage for a rectangular (unwindowed) sinewave as well as a hann, hamming and triangle window. Windowing reduces the signal to zero at the end of the sampling times so there is no sharp frequency change when the sample is repeated in order to compute the FFT. Sharp charges in time result in broad frequency spectrum. This, as well as the true signal being spilt across multiple frequency bins if the frequency of the signal does not perfectly fit into one bin, is the spectral leakage.

B.2.3 Noise

The addition of Gaussian noise to the pure tone decreases the relative size of the spectral peak with respect to the surrounding noise. This is because white noise, such as Gaussian white noise, is flat across all frequencies, meaning the base level is higher for more and more noise making the size of the peak for the tone relatively smaller, as shown in Figure 19.

B.2.4 Amplitude Modulation

Figure 20 shows the effect 3 different modulation schemes have on a pure tones fft.

Random noise modulation has a similar effect to the additive Gaussian noise from before, decreasing the relative size of the main spectral peak dramatically.

Linear increase modulation causes a large increase in spectral leakage. This is due to to effects of windowing,as described above. The sharp change from a high amplitude back to zero when the signal is repeated has a broad frequency spectrum, resulting in this large leakage. This is similar to the effects of using a rectangular window.

The periodic modulation similar to that used in amplitude modulation (AM) for radio, resulting in peaks at both the tones frequency (in this case, the higher frequency 'carrier') and peaks either side which are the upper and lower side-band frequencies. The spacing between them is the modulation frequency itself of 0.01Hz.

B.3 FFT on Real Data

B.3.1 FFT on Music

The file piano_clean.wav was loaded into Matlab and an attempt made to isolate its six notes in the spectrum. This was done by 'eyeballing' the location of the notes, windowing each note with a hann window, which gave the sharpest roll off in frequency as found out before. Results from this are shown in Figure 22, which can be compared to the full frequency spectrum in Figure 21.

The sound file has a scale of increasing pitch which is clearly seen in Notes 4 through 6. Note 1 shows the longer held note and has clearly defined peaks. Three peaks suggest a chord is being played.

B.3.2 FFT on Speech

The file f1lcapae.wav was loaded into Matlab and an attempt made to isolate individual syllables of the first phrase. This is much harder due to the more 'continuous' nature of speech, rather than the clear separate notes in the piano file. Figure 23 shows the full spectrum and attempts to isolate individual syllables, again using a hann window to window to syllable of interest.

Sounds 4 and 6 are sharp 't' sounds and have a broad spectrum while 2 is an 'o' and 7 a 'y' sound which have quite peaky spectrum, suggesting vowels have sharper peaks in there spectrum and consonants a more broader spectrum.

B.3.3 Challenge Question: Organ.wav

Organ.wav was loaded into Matlab and its first channel windowed with a hann window and then zero padded with 1×10^6 zeros to produce the frequency spectrum in Figure 25. The sound file is messy with many notes merged into one, as can be seen in the time domain plot of the organ in Figure 24 and in the large number of peaks in the frequency spectrum. By measuring the frequency at which these peaks occur and comparing them to the frequencies of musical notes (such as the table of them from https://pages.mtu.edu/ suits/notefreqs.html) the musical notes present can be extracted.Taking 'Middle C as C_4 and A at 440Hz, a few examples are:

- Main peak at 468.4Hz, corresponding to $A_3^{\#}$
- Peak at 156.8, corresponding to $D_3^{\#}$
- Peak at 627.4, corresponding to $D_4^{\#}$, and an octave higher than the previous note.

Overall there are 10 large peaks suggesting louder notes with several small peaks suggesting other quieter notes, such as sustained notes from before the recording was taken, are present.

B.4 First Interim Report Appendix

B.4.1 Plots

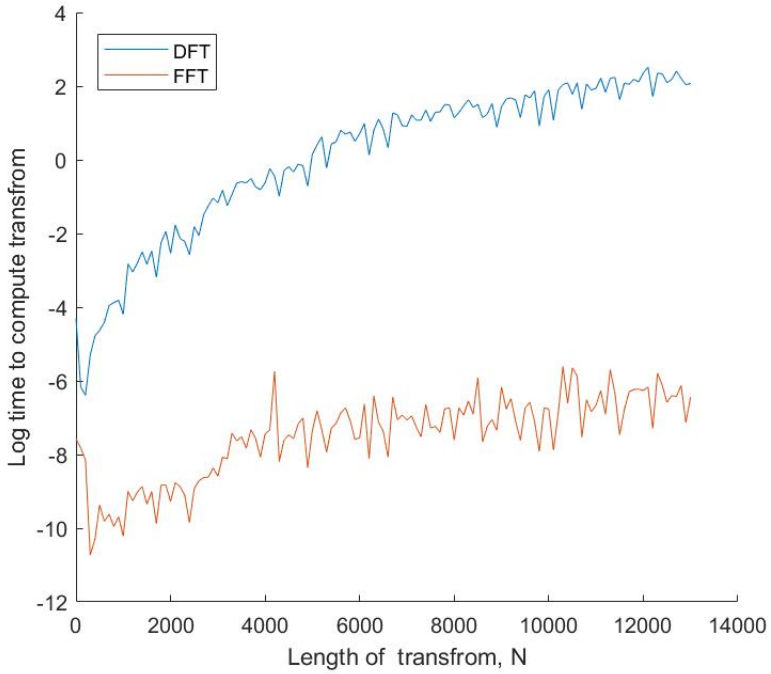

Figure 16: Comparison of the speed of the DFT and FFT, measured by the log of the time taken to compute with N samples. It is clear to see the DFT is slower and getting slower much faster than the FFT

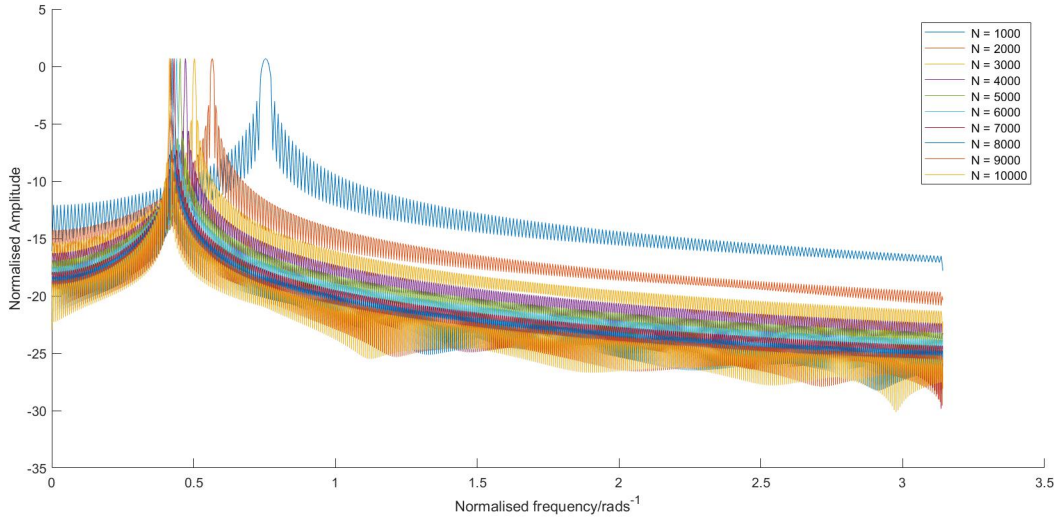

Figure 17: Log spectral magnitude plots for different lengths of FFT.

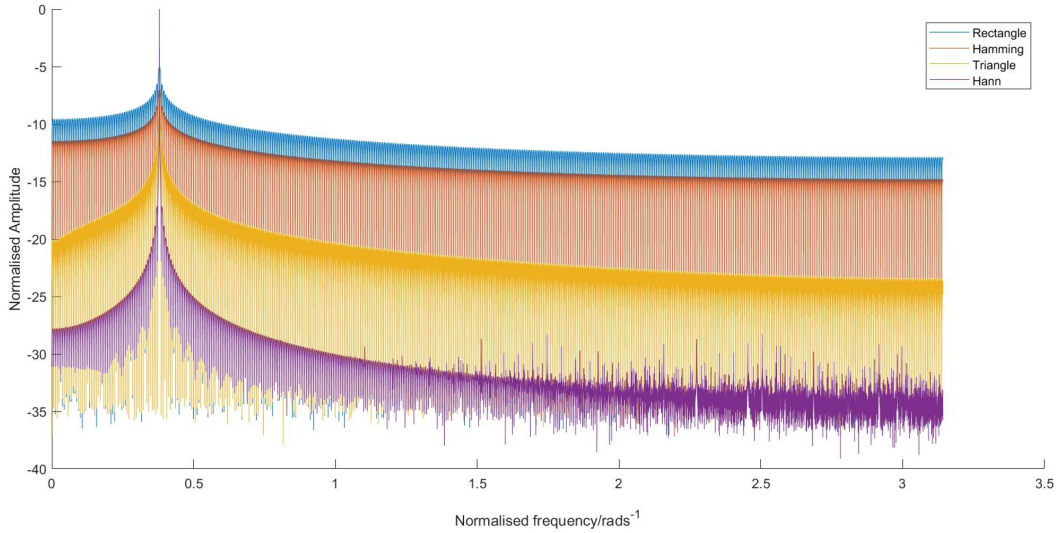

Figure 18: Log spectral magnitude plots for different lengths of FFT.

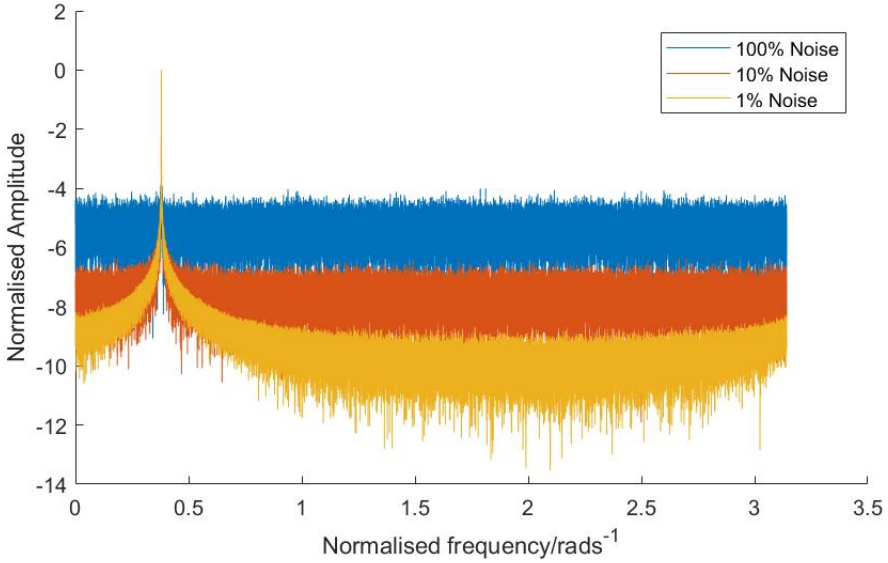

Figure 19: Normalised Log spectral magnitude plots for different windows on a pure sine tone

Figure 20: Effect different modulation schemes have on the spectral content

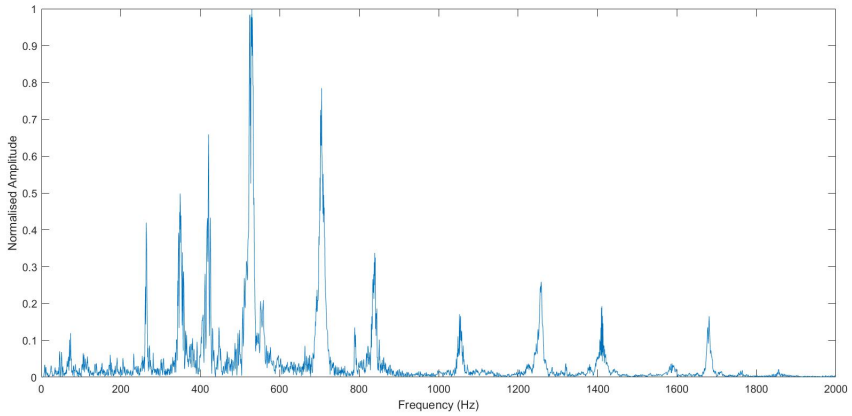

Figure 21: Spectrum of piano_clean.wav

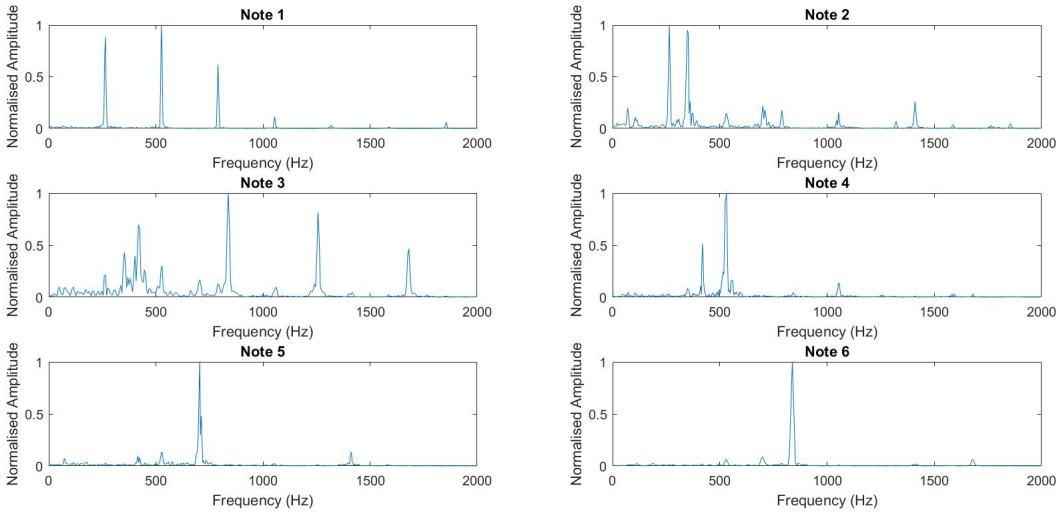

Figure 22: Spectrum's of piano_clean.wav individual notes

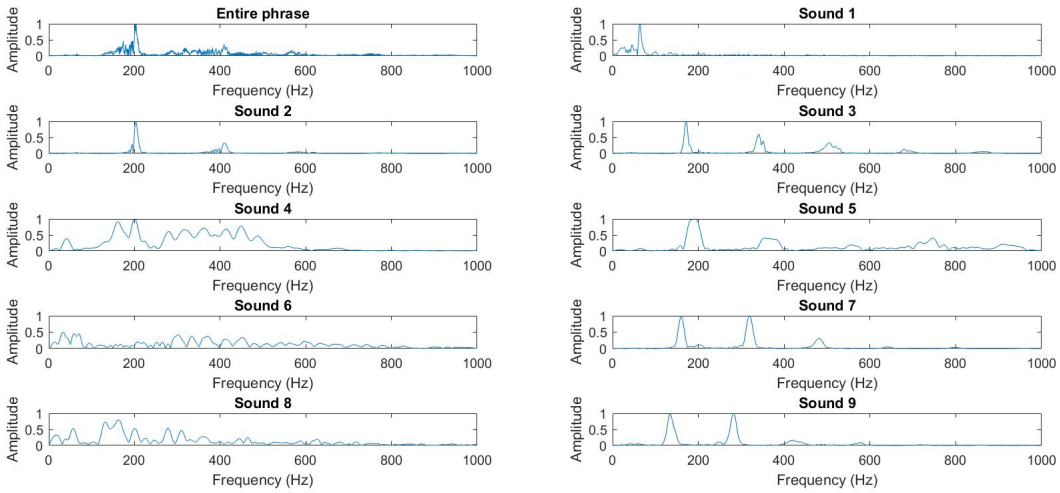

Figure 23: Spectrum's of f1lcapae.wav

Figure 24: Time domain plot of organ.wav

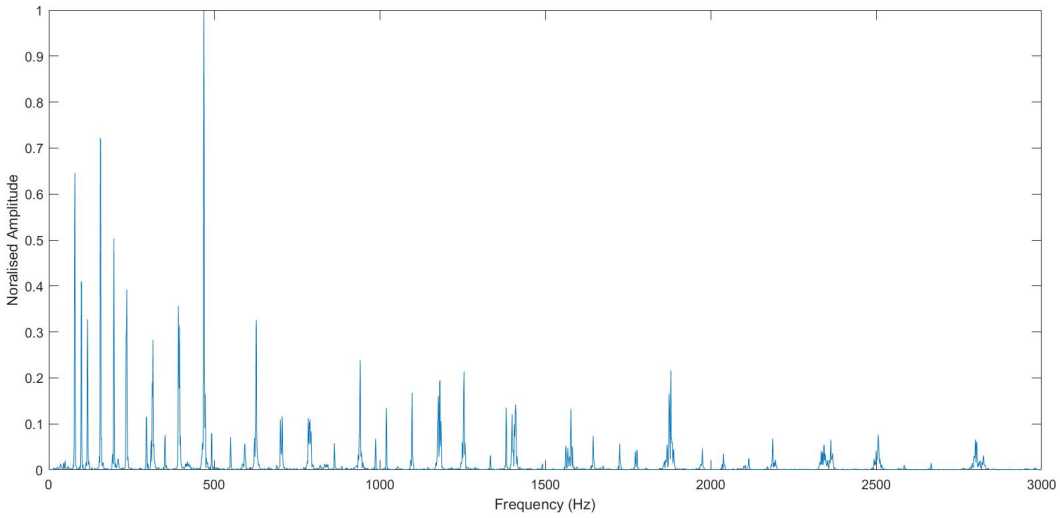

Figure 25: Spectrum's of organ.wav

C Second Interim Report

C.1 `overlap_add.m`

Below are answers to the questions posed in the handout about the `overlap_add.m` file.

(a) This script is a simple attempt at the Wiener filter but with constant filter gains of 0.1 and 0.2. Also, this filter use the overlap add technique.

(b) Weiner filters can be implemented in time or frequency domain. The multiplication of gains in frequency is the same as convolving them in time, meaning filtering in frequency or time produces the same results.

(c) Direct convolution filtering has complexity of $O(N^2)$ while FFT convolution has complexity of $O(N \log N)$. When the number of samples, N, per frame is 516 this means FFT convolution is:

$$\frac{N^2}{N \log N} = \frac{516^2}{516 \log 516} \approx 57.3 \text{ faster}$$

(d) `conj` takes the complex conjugate of a complex number so as the FFT is symmetric, only one half needs to be filters (0 to π) and then the second half (π to 2π) is the complexity conjugate of the first half.

C.2 Noise Reduction

C.2.1 Basic de-noising: `bdb31_noiseReduction.m`

`bdb31_noiseReduction.m` is the script that de-noises the required 4 audio files using a basic overlap add Weiner filter. It predicts the power spectrum of the noise using the initial silent portion of the audio, and zero pads each frame to increase precision without adding too much run time. Zero padding reduces the digital distortion resulting from filter gains of zero, which result in sharp jumps in the time domain which is head as digital noise.he resulting filtered audio is good, with minimal digital noise and almost all the random noise from the file removed.

The mean squared error (MSE) is calculated and presented after each run. The MSE however doesn't fully convey the quality of the filtered audio. For example, before zero padding there is a lot of digital distortion but the MSE result is very small, even though the audio is of a very poor quality. Furthermore, smoothing the signal using Matlab `smooth` also decreases the MSE but the audio becomes considerably quieter and 'merky', like it's being heard through a wall. Hence, zero padding is used without smoothing.

In deciding the parameters, changing the frame length has an interesting effect on the MSE, as show in Figure 27. MSE is almost periodic and overall lower at shorter frame lengths. However to smaller frame length result in audible distortion so a trade off is need. For my filter, a frame length of 256 and overlap 128 is used a hann. window

Time varying noise For the two time varying noise files, the basic Weiner filter works well. This is because the power spectrum of the noise is calculated from the variance of the initial silent bit of the audio, which takes into the account the changing noise level. This is very basic and a better filter would calculate the power spectrum of the noise for each frame and use that for the filter coefficients for each frame.

C.2.2 De-noising files without a silent portion: `bdb31_noSilenceReduction.m`

Some files, such as `Fast_N.wav` don't have a silent portion from which the variance of the noise can be estimated. A different method therefore must be used.

One method is to smooth the original data and then subtract this smoothed data from the original data to get an estimate of the noise. This is done for each individual frame and produces a file with the noise removed but more digital noise is added than in the simple de-noising case that estimates the noise level using the silent portion.

C.2.3 Time-varying noise

The time varying noise files where also filtered using this method for comparison with the silent estimate case. While this method estimates the noise better, there is more digital distortion caused using the smoothed technique. Both are presented so methods can be compared.

C.3 Second Interim Report Appendix

C.3.1 How audio files where filtered

The file are saved using the following naming convention so the filter used is clear:

`orginal_file_name.wav_filtered_method.wav`

Where `method` tells you how the noise was estimated either using the silent portion method, denoted `silent`, or smoothing each frame and subtracting the smoothed version from the original to estimate the noise, denoted `smooth`.

C.3.2 Plots

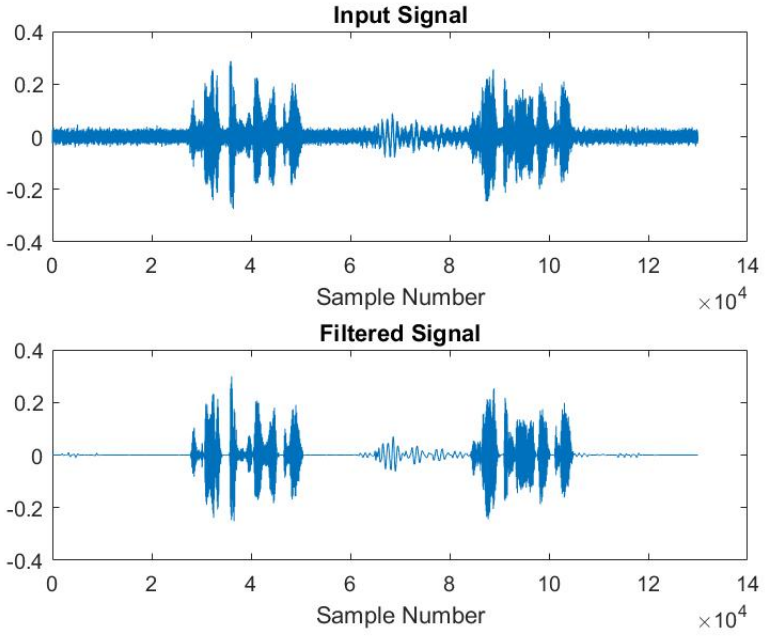

Figure 26: Plot of Input and filtered signal of male_speech_noisy_loud.wav. The script also produces plots for all files it filters.

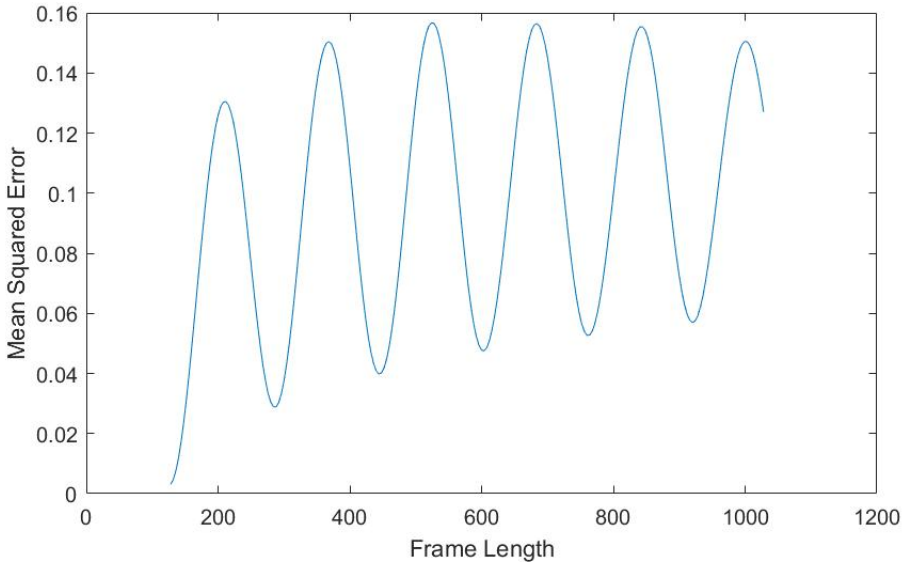

Figure 27: Plot of Mean squared error vs frame length for the overlap add Weiner filter