3 Spectral estimation and modelling

Contents

3.0 Spectral Estimation introduction

PSD calculation gives a representation of the power present in each frequency. However in real applications it is difficult to correctly estimate it as all the samples of the signal cannot be obtained. Instead we use a periodogram which estimates the PSD by pacing values in each frequency bin. A periodogram function was written:

```
function [P, k] = pgm(data)
%returns PSD estimate in P with the normalized frequencies in k
%input is a set of input values which form the signal in time domain
%implements the function: P_X(k) = 1/N |Σ_{n=0}^{N-1} x[n] e^{-j2πkn/N}|
N = length(data);
k = (0:1/N:(N-1)/N);
P=zeros(N,1);
for n = 1:N
    e = exp((-1i*2*pi*(n-1)).*(0:1:N-1)'./N)';
    P(n) = (abs(sum((data.*e)))).^2./N;
end
end
```

The code was used to generate Figure 1 – PSD Estimates of a AWGN input for different samples sizes which shows the psd estimate for various sample sizes.

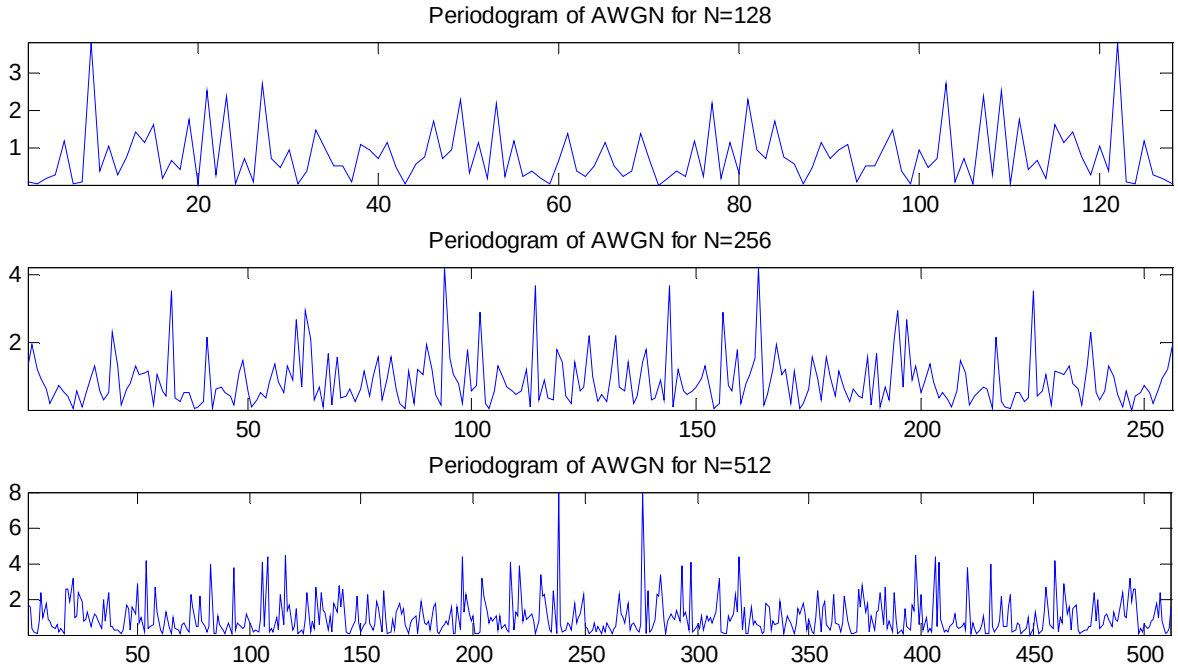

Figure 1 – PSD Estimates of a AWGN input for different samples sizes

These results seem reasonable if we remember that the PSD if the Fourier transform of the autocorrelation[1]. We have that $PSD_x(f) = \int_{-\infty}^{\infty} R_x(\tau) e^{-2\pi i f \tau}\, d\tau$ and due to having a Gaussian process we have $R_x(t) = {\sigma_x}^2 \delta(t)$ which means the expected $PSD_x(f)$ is

[1] http://mcise.uri.edu/chelidze/courses/mce567/handouts/psdtheory.pdf

$PSD_x(f) = \int_{-\infty}^{\infty} {\sigma_x}^2 \delta(t)\, e^{-2\pi i f \tau}\, d\tau = {\sigma_x}^2$. As the periodograms in Figure 1 are for very small sample sizes we can see that while we expect a uniform distribution which is somewhat visible –the main drawback is the very low number of samples.

3.1 Averaged Periodogram estimates

3.1.1 PSD estimate smoothing

As we expected a uniform distribution from the previous result it can be investigated in whether or not this is possible to obtain a 'better' uniform distribution by smoothing out the PSD estimate. This is what Figure 2 achieves. From it, it can be seen that the output does indeed look more like the expected uniform distribution. Additionally the result gets closer to the expected result the higher number of samples.

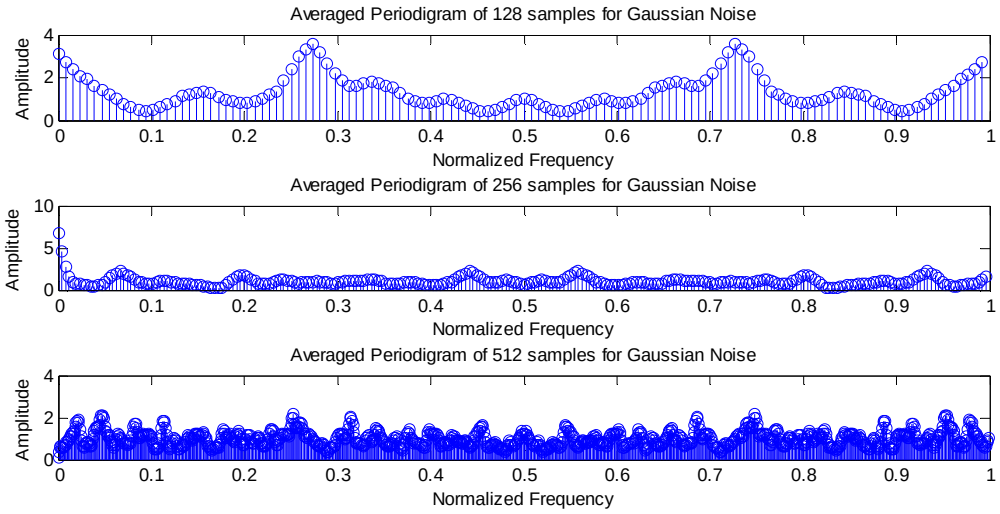

Figure 2 -Averaged Periodogram of Gaussian noise notice the contrast with Figure 1 and how this graph looks more like a uniform distribution – the expected result.

3.1.2 PSD estimate splitting

A 1024 sample sequence was generated using randn and divided into 8 sample segments as seen from Figure 3. As all values are independent splitting it up into portions of N=128 has not changed the overall shape compared to the 128 sample instance found in Figure 1. This means that the same inaccuracy remains and that the large variation is caused by the small sample size.

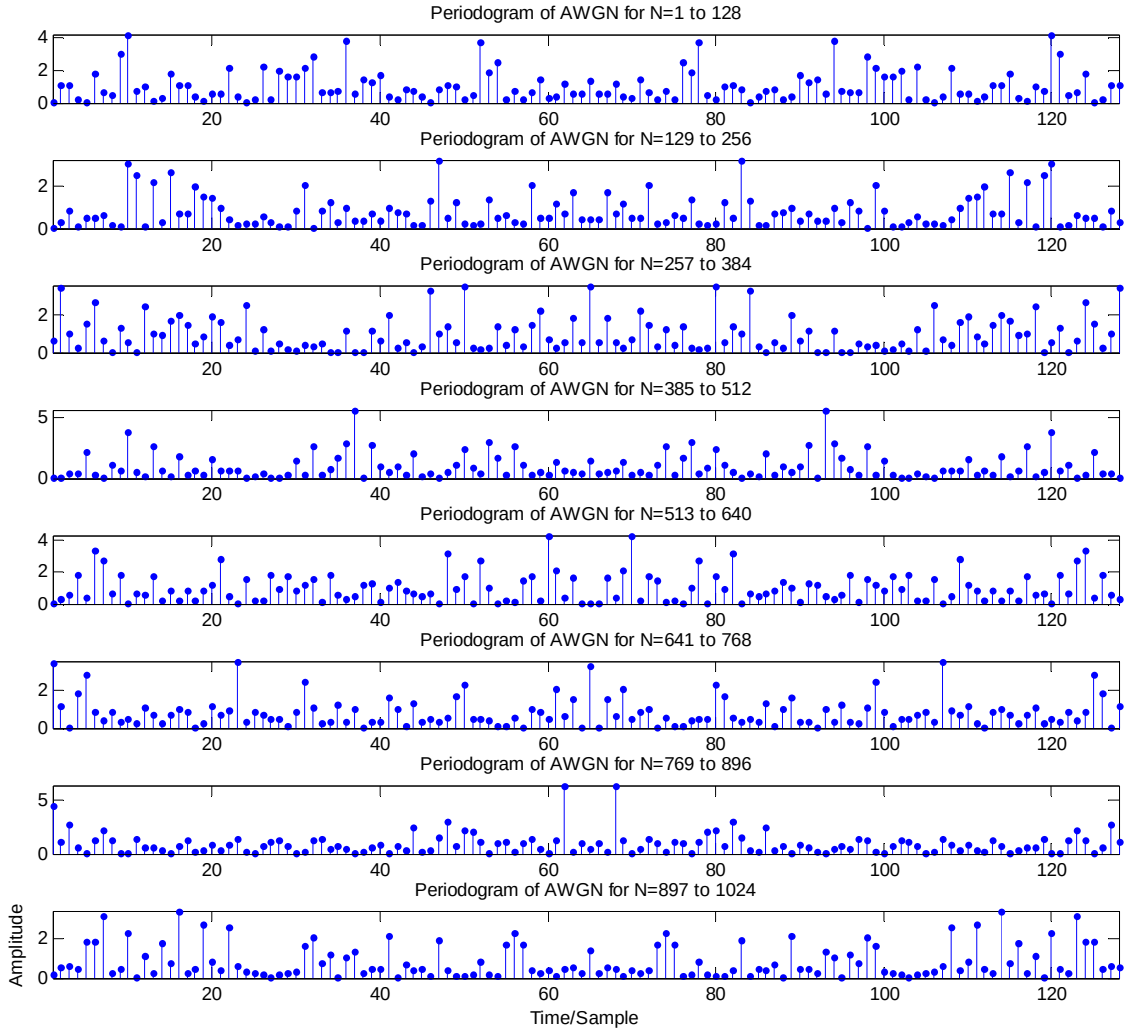

Figure 3 – 8-split of 1024 sample Gaussian noise data.

```
x=randn(1024,1);
for i=1:8
    subplot(8,1,i)
    [a,b]=pgm(x((i-1)*128+1:128*i)');
    plot(a)
    axis tight;
    str = sprintf('Periodogram of AWGN for N=%d to %d',(i-1)*128+1,128*i);
    title(str);
end
```

3.1.2 PSD estimate splitting and smoothing

In this part we take the average of all individual bins of Figure 4 and take the spectrogram. We see that the mean and standard deviation are now much closer to what is expected. Thus we can infer that the PSD splitting and smoothing leads to a better PSD estimate than the previous periodograms.

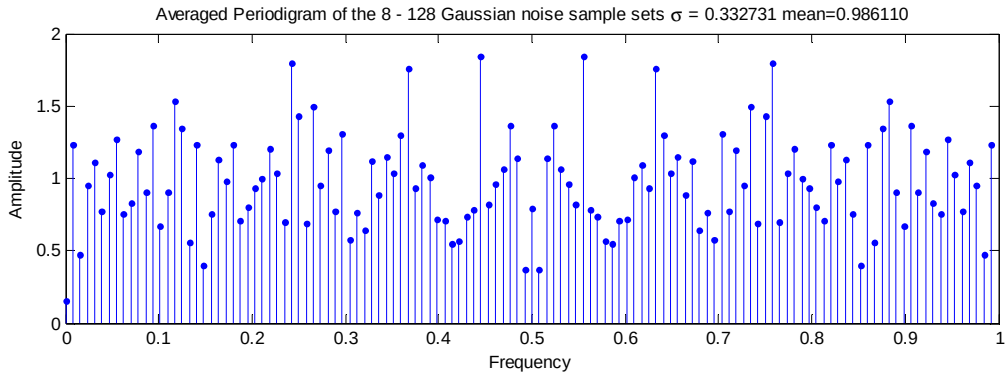

Figure 4 - Periodogram resulting from the average of the 8 periodograms of Figure 3.

3.2 Spectrum of autoregressive processes

In this section we look at a 1024-sample (technically 1064 with 40 subtracted) random noise passed through an AR1 process. This essentially acts as a high pass filter of the input and is apparent from Figure 5 where the output appears to be more "spiky" a feature of high frequency signals.

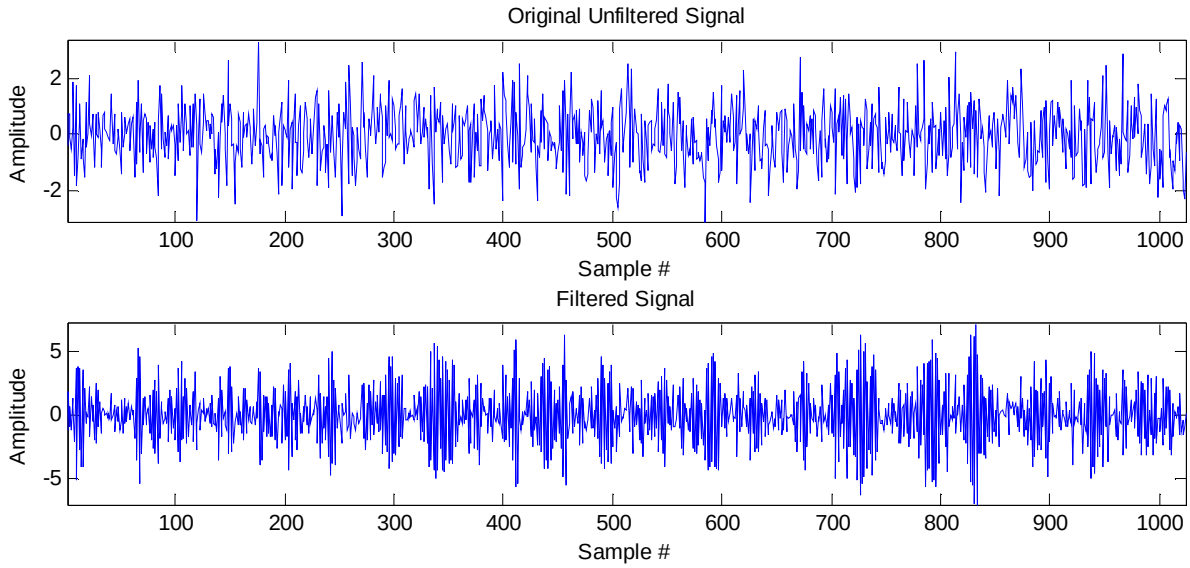

Figure 5 – Gaussian Noise data passed through a filter y=filter([1],[1 0.9],x)

The reason for the increase is due to the filter features. Indeed the relation $\sigma_y{}^2 = \frac{\sigma_x{}^2}{1-a_1{}^2}$ shows how the variances are related to each other. From this we can derive the expected variance of $\sigma_y{}^2 = \frac{1}{1-0.9^2} \cong 5.26$ which agrees with our findings.

3.2.1 Ideal PSD response

We investigate the nature of the PSD ideal estimate.

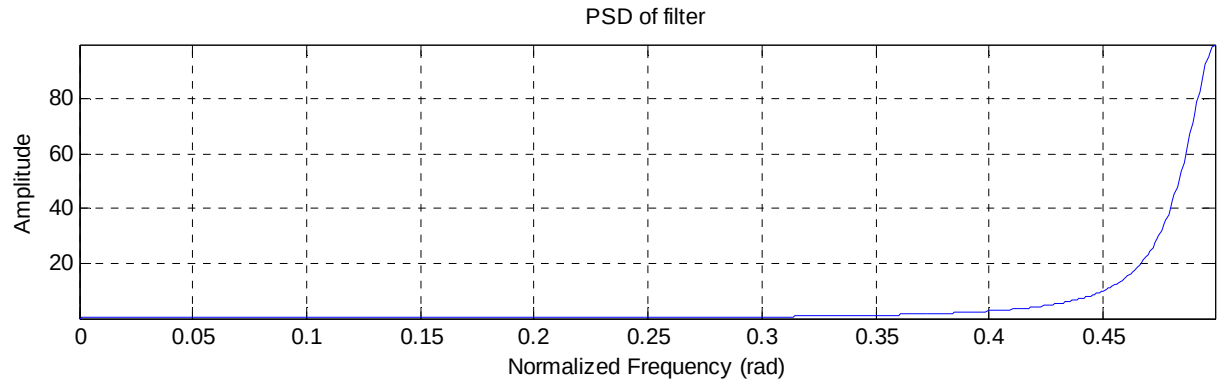

Figure 6 – PSD estimate of Y, generated by an AR1 process.

From the above graph of the PSD of Y ($P_Y(f) = \frac{1}{|1+0.9e^{-j2\pi f}|^2}$) , generated by an AR(1) process, it can be noted that most power is present in higher frequencies and that virtually none exists in frequencies under 0.3 rad (normalized frequency) and most is present past 0.45 rad.

This due to the nature of $P_Y(f)$ which has a max for f (normalized) = 0.5.

3.2.2 Comparison to obtained data

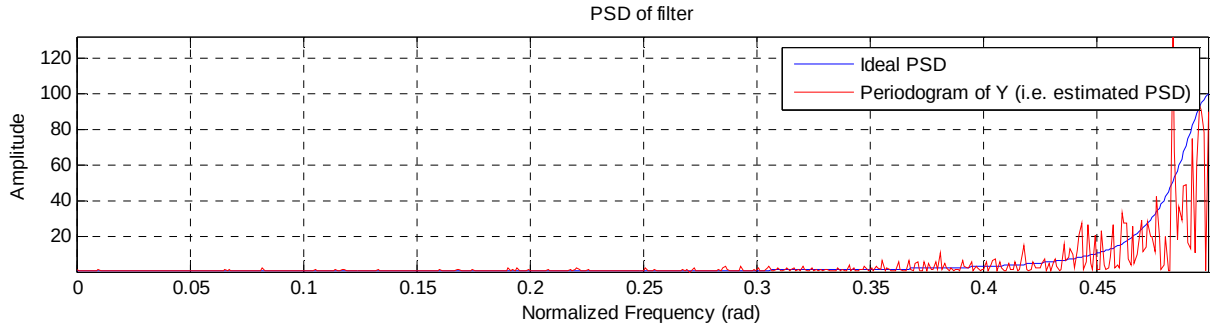

Figure 7 – Periodogram of Y plotted against the ideal PSD. The output periodogram can be compared to its input which would be a Gaussian signal similar to Figure 1.

The above graph compares the ideal PSD to the periodogram of Y. The first observation is that the AR1 does succeed in exaggerating high frequency components while minimizing the components in lower frequencies.

The differences are due to the collected data being performed on AWGN and that the data was simply taken as a whole to make the periodogram. Previous methods such as segmenting then averaging would give better estimates – this assigns the power more accurately to the correct frequencies (as a trade-off less frequency resolution is available).

The periodogram and the ideal while somewhat similar disagree to a certain extent.

This is due to 2 inter-related reasons:

- The periodogram is only an approximate of PSD
- The input is a discrete and limited in sample size whereas for an ideal PSD an infinite number of samples would be needed.

3.2.3 Effect of rectangular windowing

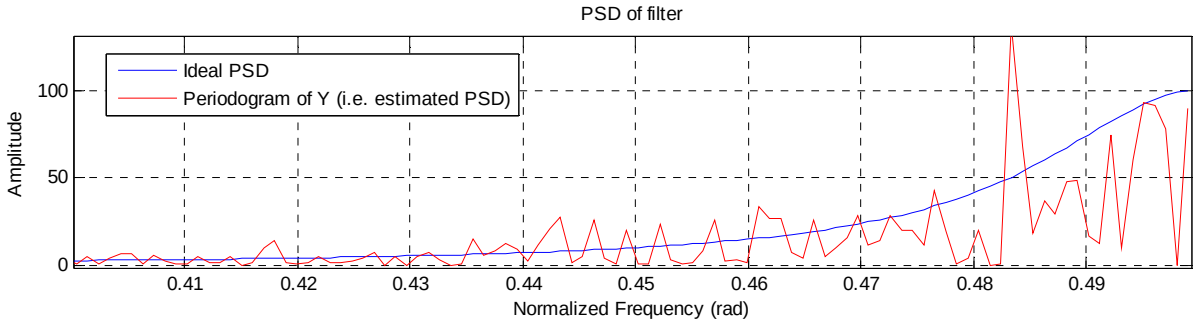

Figure 8 – Periodogram of Y plotted against the ideal PSD for a normalized frequency range of 0.4 to 0.5

Around 0.4 to 0.5 normalized frequency it can be seen that the filter is far from ideal. This would be due to the way that the periodogram calculates the output. This is first done by windowing the data in time domain by a rectangular window. However in frequency domain this is equivalent to convoluting with a sinc function as $\mathcal{F}_{rect\left(\frac{t}{a}\right)}(f) = asinc(af)$. This causes the peak, theoretically at 0.5 normalized frequency , to be more spread out For example in Figure 8 it can be see that the peak it at around 0.48 – this is due to the combination of the sample size being small and the effect of rectangular windowing in time domain.

Additionally we can see that a small rect in time leads to a very large sinc in frequency – which is another reason larger sample sizes are needed

3.2.4 Model Estimation

Calculating the PSD by periodogram is possibly not the best estimation and another model based method is attempted. The idea of this model is to use observed values of the sampled data. This is because the auto-correlations can be expressed from the Yule-Walker equations as

$$r_{xx}(0) = -a_1 r_{xx}(1) + {\sigma_x}^2$$

$$r_{xx}(1) = -a_1 r_{xx}(0)$$

Rearranging these equations we get

$$\widehat{a_1} = \frac{-\hat{r}_{xx}(1)}{\hat{r}_{xx}(0)}$$

$$\hat{\sigma}_x^2 = \hat{r}_{xx}(0) + \widehat{a_1}\hat{r}_{xx}(1)$$

These values can be used to obtain and estimate of P_y i.e. $\hat{P}_y$.

This led to the following code:

```
corry = xcorr(y,'unbiased'); %% plot estimate
```

```
corry = xcorr(y,'unbiased');
%calculate a and sigma
a1 = -corry(2)/corry(1);
sigma_x = corry(1)+a1*corry(2);
%sigma_x = var(x); %alternative to line above
figure(3);
%get the data
[h,w]=freqz(sigma_x,[1 a1],512);
%plot estimate with model
plot(w/(2*pi),abs(h).^2); hold on;
plot(w/(2*pi),py(1:512),'r');hold off;
legend('Estimated','Periodogram of Y (i.e. estimated PSD)')
xlabel('Normalized Frequency (rad)')
ylabel('Amplitude')
axis tight;
grid on;
str = sprintf('Estimated PSD of AR(1)\n \\sigma_x = %f a_1=%f',sigma_x,a1);
title(str);
```

3.2.5 Model Comparison

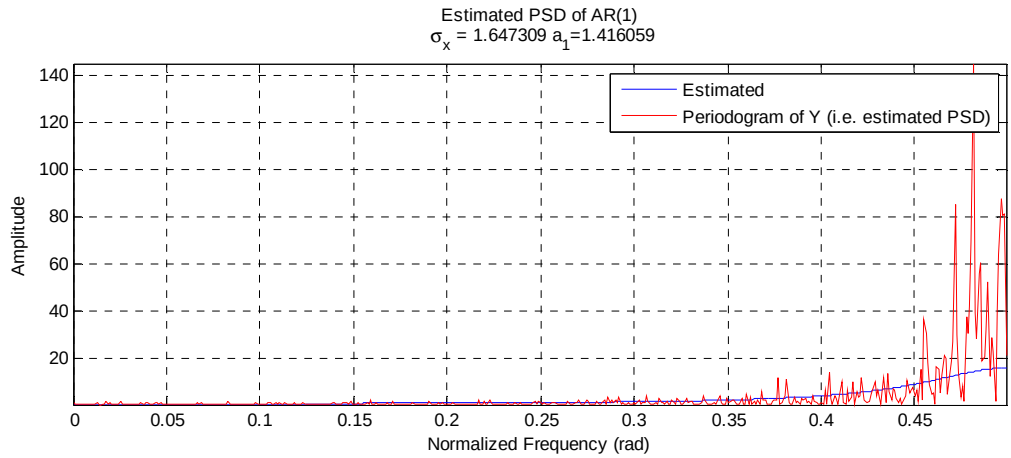

Figure 9 - Periodogram of Y plotted against the estimated PSD

As can be seen the estimated PSD from $\hat{P}_Y(f) = \frac{\hat{\sigma}_x{}^2}{|1+\hat{a}_1 e^{-j2\pi f}|^2}$ in Figure 9 the estimated model has the same shape as the periodogram but with a much reduced amplitude. This is due to the scaling heavily relying on the variance estimate $\hat{\sigma}_x{}^2$ which is extremely sensitive due to the way it relies on a set of data with very small values. Thus this model while good at estimating the shape (due to a_1 being reasonable accurate) does not work very well in estimating the amplitude.

3.3 Spectrogram example: dial tone pad

Here we generate the sounds generated when using old-style phones which use the DTMF (Dual Tone Multi-Frequency) dialling system. In this system each number is assigned pair of frequencies as seen in Figure 10. After generation there is hope of being able to decode the frequencies and correctly determine which symbol was being represented.

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

Figure 10 – DTMF Frequency pairs for different keys

3.3.1 Number generation

N random London phone numbers were generated by the following code in matlab:

```
l_phone = floor(10*rand(N,11));
l_phone(:,1:3) = ones(N,1)*[0 2 0];
```

From this it was possible to generate the discrete time samples of a London phone number:

```
N = 1;
Fs = 32768; %set sampling frequency
%generate phone numbers
l_phone = floor(10*rand(N,11));
l_phone(:,1:3) = ones(N,1)*[0 2 0];
f1 = zeros(11,1);
f2 = zeros(11,1);
% assign frequencies
for i=1:11
    switch mod(l_phone(1,i),3)
        case 0
            f1(i) = 1477;
        case 1
            f1(i) = 1209;
        case 2
            f1(i) = 1366;
    end
    switch floor((l_phone(1,i)-1)/3)
        case 0
            f2(i) = 697;
        case 1
            f2(i) = 770;
        case 2
            f2(i) = 852;
    end
    %handle exception
    if l_phone(1,i) == 0
        f1(i) = 1336;
        f2(i) = 941;
    end
end
tone = zeros(1,180224);
%generate sound using y[n] = sin(2πf1n) + sin(2πf2n)

for i=1:11

    for t=(1+Fs/2*(i-1)):(Fs/4*i + Fs*(i-1)/4)
        tone(t) = (sin(2*pi*f1(i)*(t)/Fs)+sin(2*pi*f2(i)*t/Fs))/2;
    end
    %silent section for 0.25 seconds
```

```
    for t=((1+Fs/4*i)+ Fs*(i-1)/4):Fs/2*i
        tone(t) = 0;
    end
end
```

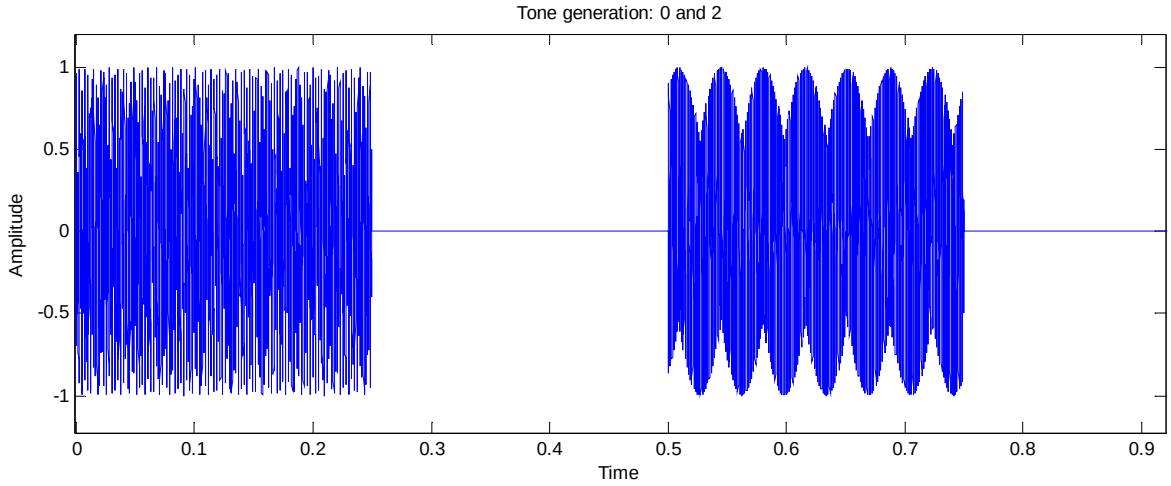

Figure 11 – Time plot of first two key presses of a London number, 0 and 2

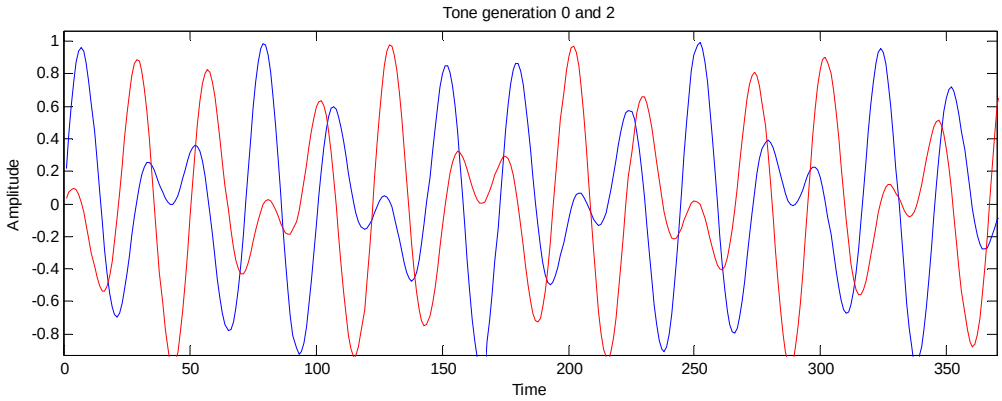

Figure 12 – In blue is the time domain plot of the tone for 0(941 and 1336Hz) and in red is the plot for 2(697 and 1336Hz). It can be seen that they have different frequency components but identification of which is which is not simple.

3.3.2 & 3.3.3 Frequency – Spectogram analysis

Note – while matlab provides a spectrogram function Mike Brookes has written a possibly more difficult to use but way more configurable spectrogram function as part of his voicebox toolbox[2].

Figure 13 and Figure 14 show the spectrogram of 2 random London telephone numbers with different sensitivity settings. From these we can see that the frequencies can be accurately separated and identified, leading to key identification being possible. Indeed the first frequency pair can be seen in Figure 14 to be just under 950hz and just above 1300 Hz, clearly indicating that this was a 0.

[2] http://www.ee.ic.ac.uk/hp/staff/dmb/voicebox/voicebox.html

Thus key classification in frequency domain is a definite possibility and explains why this standard is so widespread.

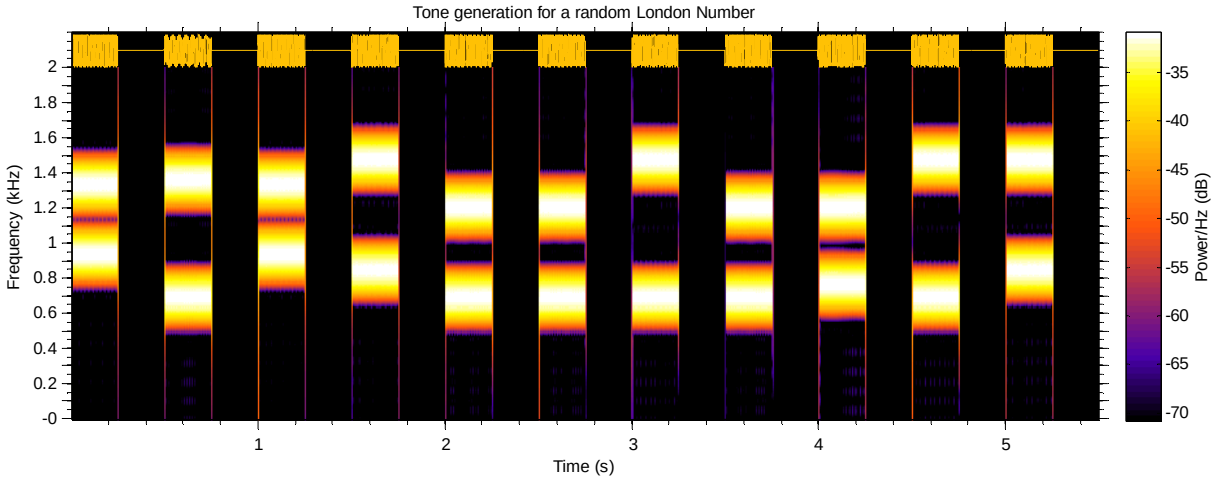

Figure 13 – Spectrogram of a random London telephone number using amplitude sensitive settings.

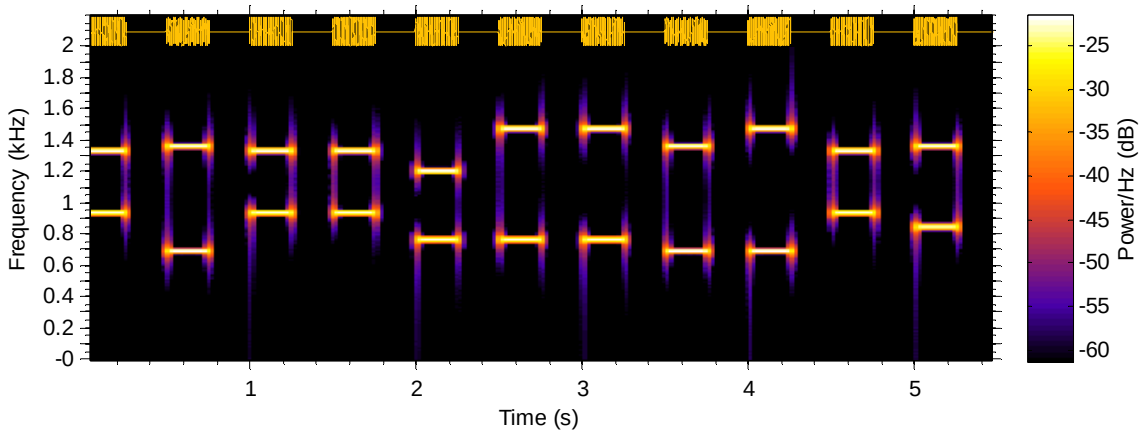

Figure 14 - Spectrogram of a random London telephone number using less amplitude sensitive settings to more clearly distinguish the frequencies between each other.

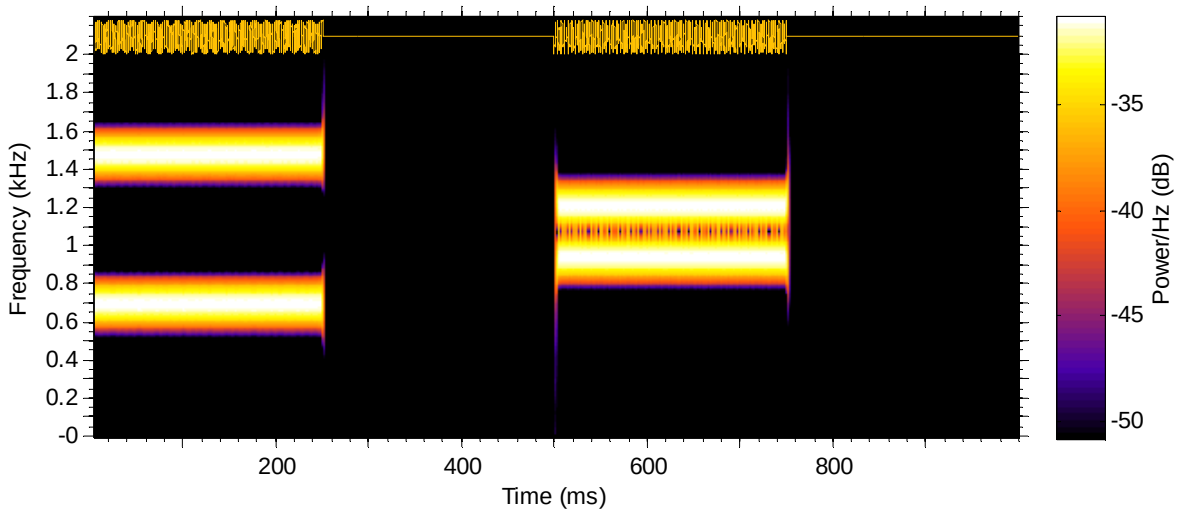

Figure 15 – Spectrogram of first two keys – 0 and 2

3.3.4 Analysis in noisy environment

In the previous tests the signal was not corrupted by noise – however this is not representative of real life scenarios. In this scenario we investigate what happens when the received signal is[3] $y[n] = \sin(2\pi f_1 n) + \sin(2\pi f_2 n) + \sigma_n n[n]$ and we will determine what the minimum workable SNR could be.

In Figure 16 the SNR is 30 and the traces of the noise can be seen lightly but overall the tone frequencies are easily distinguishable .

Figure 17, with an SNR of 11dB, the noise while very visible does not interfere with visual identification of the sounds still possible.

Figure 18, with a negative SNR of -3dB the signal can still be identified however the noise starts to get visibly worse. The time domain plot (just above the spectrogram) starts showing the noise to be able to be confused with a tone period.

From Figure 19 which has an SNR of -18dB the signal is nearly indistinguishable from the noise with only faint traces of the signal remaining. By looking at the time domain plot it is very difficult to distinguish a pause from a tone.

Overall it is possible to determine that identifying the tones by spectral analysis is a modestly robust method due to the amount of noise coped with. This is due to the dual tone nature of this system where most power is concentrated exclusively at 2 frequencies.

[3] $n[n]$ is zero mean Gaussian

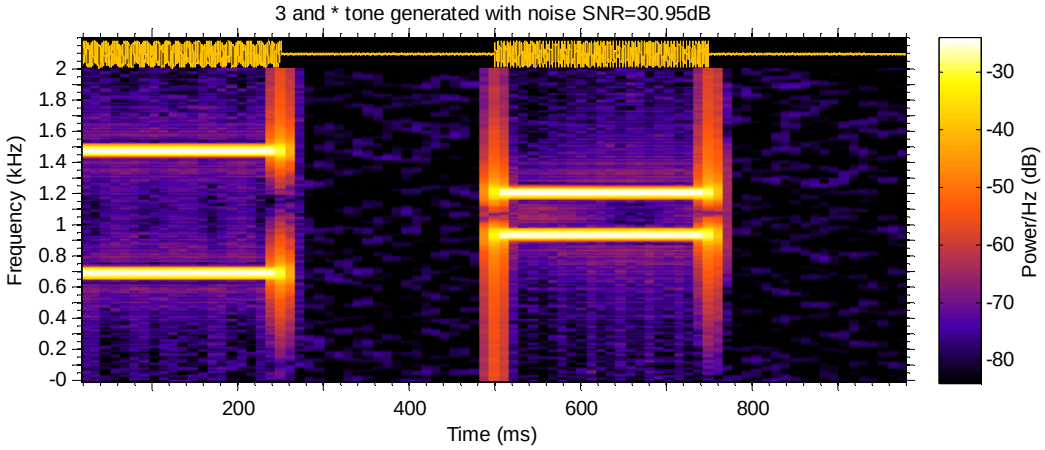

Figure 16– 3 and * tones generated with AWGN noise of $\sigma_n = 0.01$. Above is a small time domain version of the signal.

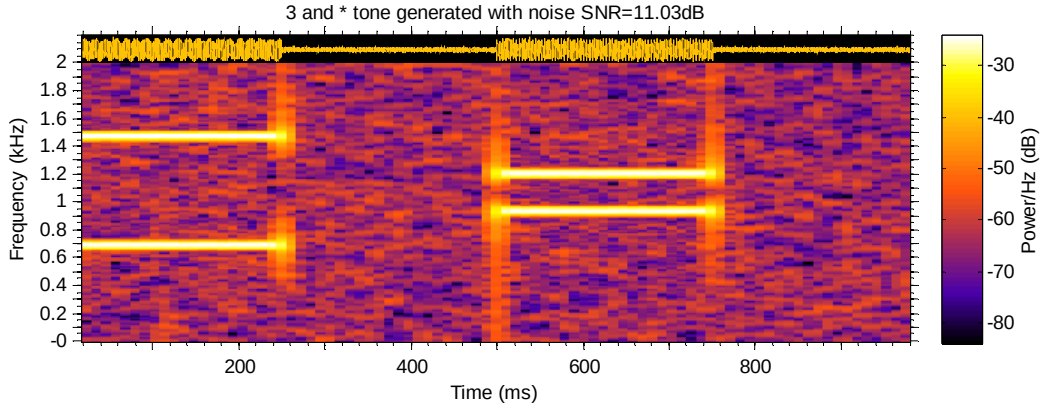

Figure 17– 3 and * tones generated with AWGN noise of $\sigma_n = 0.1$

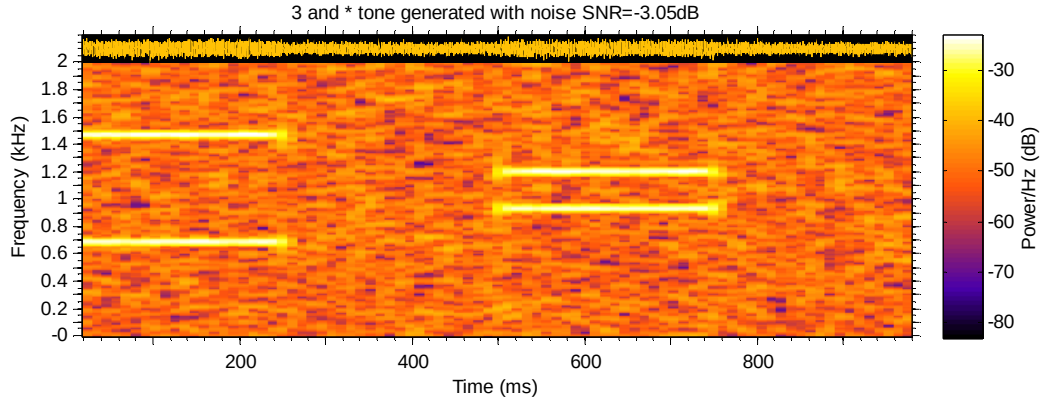

Figure 18– 3 and * tones generated with AWGN noise of $\sigma_n = 0.5$

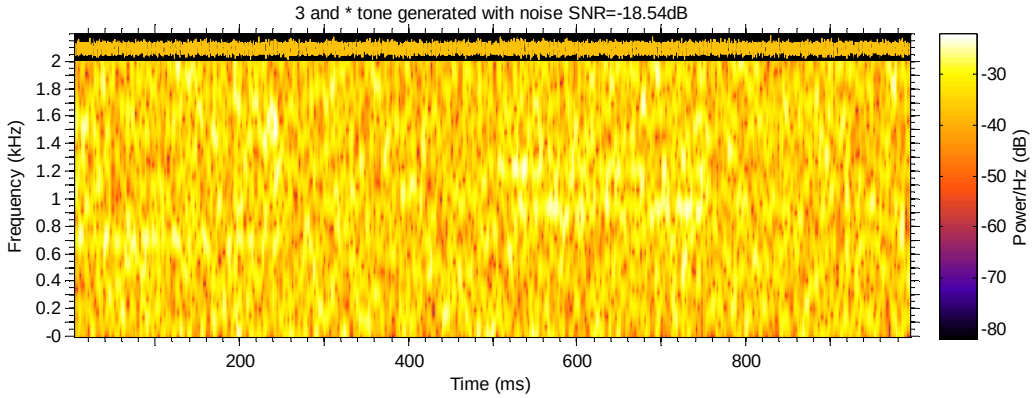

Figure 19 – 3 and * tones generated with AWGN noise of $\sigma_n = 3$

3.4 Spectrum of dialled signals using AR modelling

As seen in section 3.2 the PSD of a signal can be estimated by trying to model it to an AR process. This allows us to model the spectrum by the function

$$P_y(f) = \frac{{\sigma_x}^2}{\left|1 - \sum_{k=1}^{p} a[k]e^{-\mathrm{j}2\pi\mathrm{f}k}\right|^2}$$

Which in matlab gives us:

```
[h,w] = freqz(σ_x^2,a_k,512,Fs)
plot(w/(2*pi),abs(h).^2)
```

This essentially plots the estimate of the AR process. This is possible due to the way an AR process can be modelled. Consider the following AR2 process[4]:

$P_y(f) = \frac{{\sigma_x}^2}{\left|1-a_1e^{-j2\pi f}-a_2e^{-j4\pi f}\right|^2}$ with poles respectively at $p_1, p_2 = \frac{1}{2}\left(a_1 \pm \sqrt{a_1^2 + 4a_2}\right)$

When the poles form a complex conjugate i.e. $0 < a_1^2 + 4a_2$ then we get $\cos(2\pi f) = \frac{(1-a_2^2)(4a_2+a_1^2)}{4a_2}$, which when rearranged gives $f = \frac{1}{2\pi}cos^{-1}\left(\frac{(1-a_2^2)(4a_2+a_1^2)}{4a_2}\right)$.

The reason we need an AR4 process at minimum for the 3 and * dial tones draws from the reasoning above. Indeed to create an oscillation or a peak in frequency we need a complex pair (conjugate). In the case of 3 and * which are made from a dual-tone system there is a need to create 2 oscillations (or equivalently 2 frequency peaks), meaning that for 2 conjugate pairs to exist and AR process of 4 is needed at minimum.

In this section we take the 3 and * tones and look at the various estimated PSD under various noise conditions.

[4] http://www.ece.umd.edu/class/enee630.F2012/slides/part-3_sec2_slidesAll.pdf

In Figure 20,Figure 21 and Figure 22 we notice that the higher the SNR the more prevalent the peak whereas the lower the SNR gets the more spread out the PSD is, showing the effect of noise corruption. We however notice than an AR4 process is not enough to model the two expected peaks as only one peak appears in all the tested processes.

This is essentially because the AR4 process takes an estimate which leads to outputting a "blurred" PSD of a model of the sinusoids with the noise (after all the AR process is simply estimating what is given to it which also means the noise).

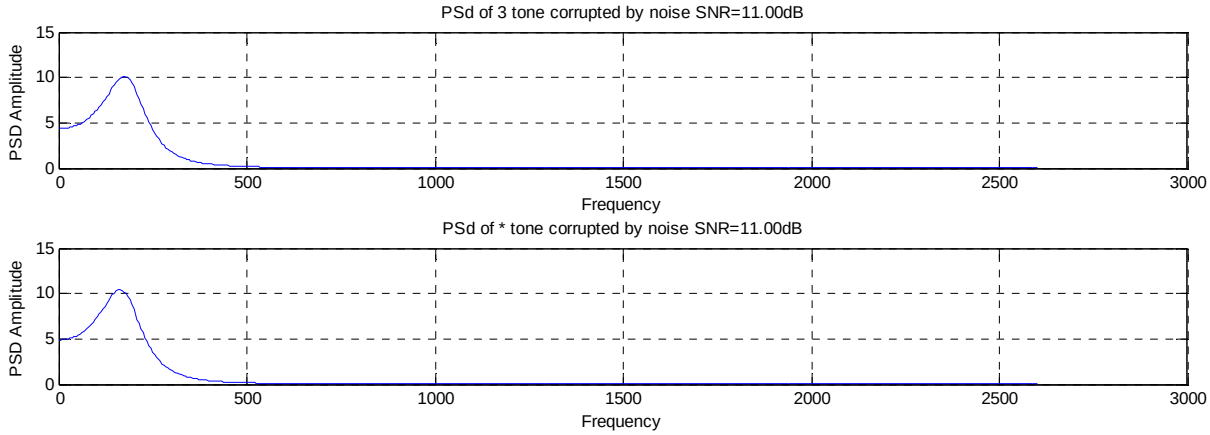

Figure 20 – AR4 PSD estimate model of both signals with a SNR of 11

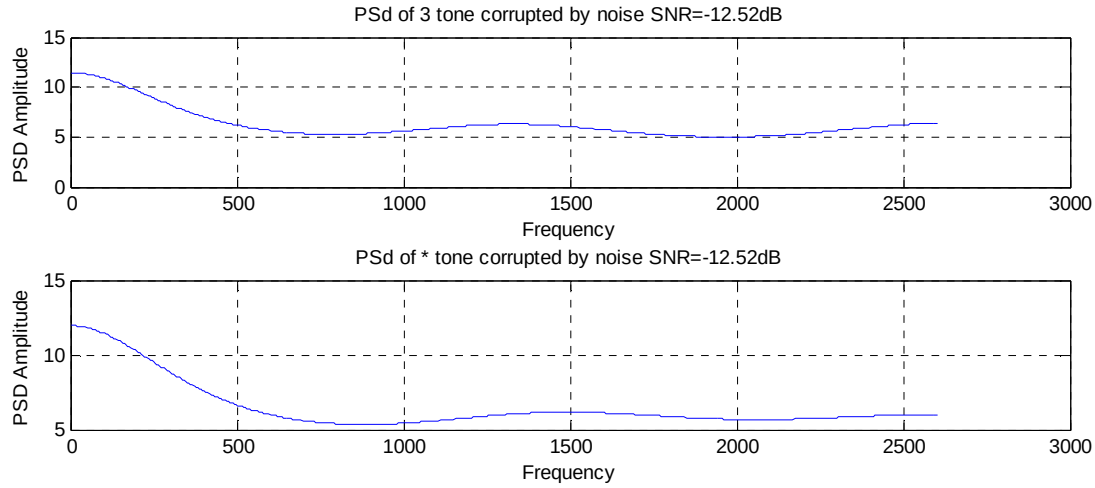

Figure 21 - AR4 PSD estimate model of both signals with a SNR of -11

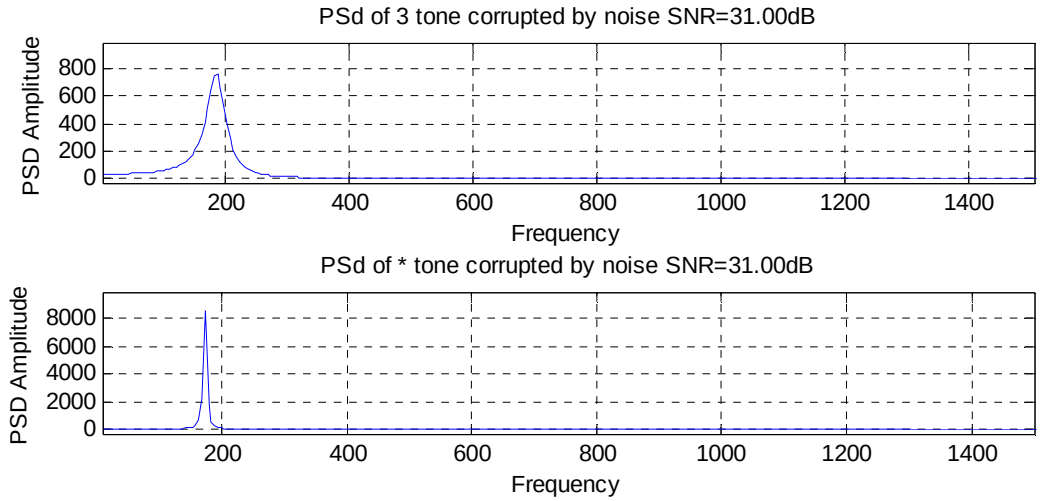

Figure 22 - AR4 PSD estimate model of both signals with a SNR of 31

Seeing how an AR4 seems to not show 2 peaks it was tried on a clean input as seen in Figure 23 and yet again AR4 is not enough. Thus it was chosen to increase the order to see if an order capturing 2 peaks existed. Figure 24 and Figure 25 show that for various SNR values a higher order does indeed give rise to the 2 expected peaks which are so useful in frequency identification. Indeed for a relatively noisy signal with -3dB an order of 50 was able to identify the 2 peaks and an order of 25 was needed to identify peaks with an SNR of 90dB.

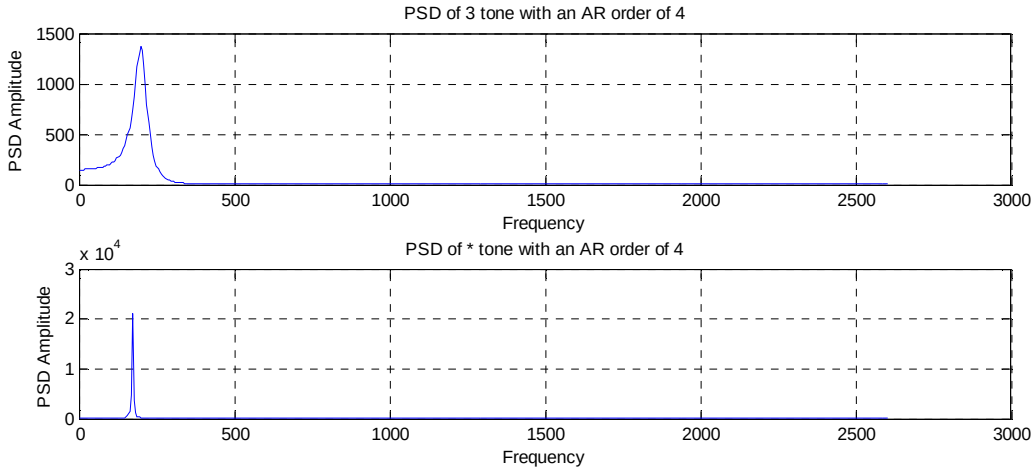

Figure 23 – AR4 PSD estimate of a clean signal

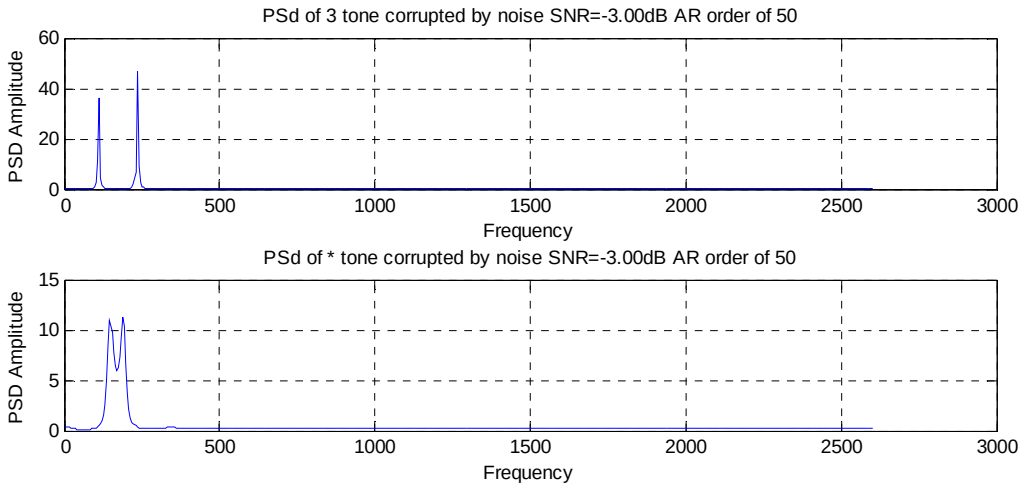

Figure 24 – AR50 PSD estimate of a corrupted signal at -3dB

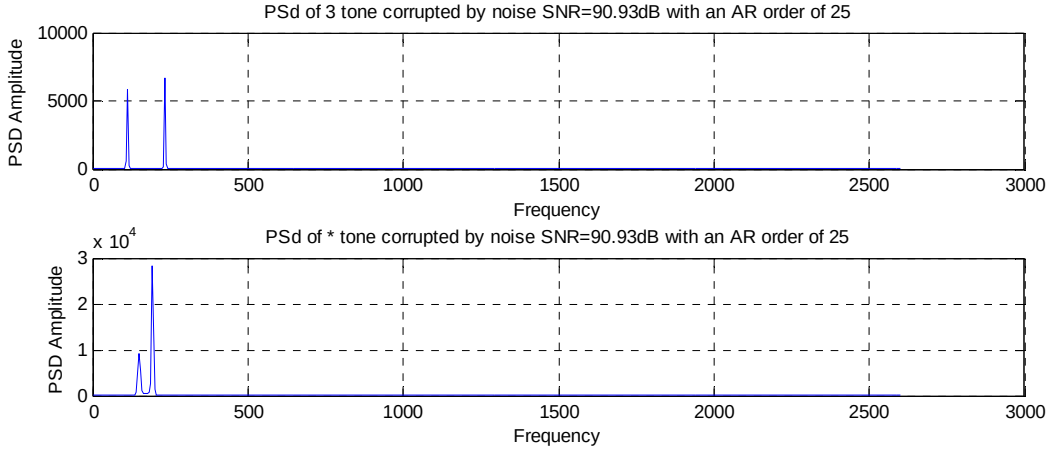

Figure 25– AR25 PSD estimate of a corrupted signal at 90dB

Appendix

Table of Figures

Matlab code

Periodogram function

```
function [P, k] = pgm(data)
%returns PSD estimate in P with the normalized frequencies in k
%input is a set of input valuesn which form the signal in time domain
N = length(data);
k = (0:1/N:(N-1)/N);
P=zeros(N,1);
for n = 1:N
    e = exp((-1i*2*pi*(n-1)).*(0:1:N-1)'./N)';
    P(n) = (abs(sum((data.*e)))).^2./N;
end
end
```

Part 3.1

```
clc;
clear all;
close all;
x=randn(1024,1);
c=zeros(128,8);
for i=1:8
    [c(:,i),~] = pgm(x((i-1)*128+1:128*i)');
end
c = mean(c');
%%
[a,b]=pgm(c);
```

```
figure(1);
stem(c,'.')
axis tight;
str = sprintf('Periodogram of AWGN average of 8 sample sets');
title(str);
figure(2);

%filtfilt(0.2*[1 1 1 1 1],1,a); %used for 3.1.1
for i=1:8
    subplot(8,1,i)
[a,b]=pgm(x((i-1)*128+1:128*i)');
stem(a,'.')
axis tight;
str = sprintf('Periodogram of AWGN for N=%d to %d',(i-1)*128+1,128*i);
title(str);
end
xlabel('Time/Sample');
ylabel('Amplitude')
```

Part 3.2

```
clc;
clear all;

figure(1)
x=randn(1064,1);
y=filter([1],[1 0.9],x);
y=y(41:1064);
x=x(41:1064);
subplot(2,1,1)
plot(x)
xlabel('Sample #')
ylabel('Amplitude')
axis tight;
str = sprintf('Original Unfiltered Signal');
title(str);
subplot(2,1,2)
plot(y)
xlabel('Sample #')
ylabel('Amplitude')
axis tight;
str = sprintf('Filtered Signal');
title(str);
[h,w]=freqz([1],[1 0.9],512);
figure(2)
py = pgm(y');
plot(w/(2*pi),abs(h).^2); hold on;
plot(w/(2*pi),py(1:512),'r');hold off;
legend('Ideal PSD','Periodogram of Y (i.e. estimated PSD)')
xlabel('Normalized Frequency (rad)')
ylabel('Amplitude')
axis tight;
grid on;
str = sprintf('PSD of filter');
title(str);
%% plot estimate
corry = xcorr(y,'unbiased');
%calculate a and sigma
a1 = -corry(2)/corry(1);
sigma_x = corry(1)+a1*corry(2);
```

```
%sigma_x = var(x); %alternative to line above
figure(3);
%get the data
[h,w]=freqz(sigma_x,[1 a1],512);
%plot estimate with model
plot(w/(2*pi),abs(h).^2); hold on;
plot(w/(2*pi),py(1:512),'r');hold off;
legend('Estimated','Periodogram of Y (i.e. estimated PSD)')
xlabel('Normalized Frequency (rad)')
ylabel('Amplitude')
axis tight;
grid on;
str = sprintf('Estimated PSD of AR(1)\n \\sigma_x = %f a_1=%f',abs(sigma_x),a1);
title(str);
% [a,b]=pgm(x((i-1)*128+1:128*i)');
% %filtfilt(0.2*[1 1 1 1 1],1,a);
% plot(a)
% axis tight;
% str = sprintf('Periodogram of AWGN for N=%d to %d',(i-1)*128+1,128*i);
% title(str);
```

Part 3.3

```
clear all;
clc;
close all;
addpath('voicebox');
N = 1;
Fs = 32768;
%generate phone numebrs
l_phone = floor(10*rand(N,11));
l_phone(:,1:3) = ones(N,1)*[0 2 0];
f1 = zeros(11,1);
f2 = zeros(11,1);
% assign frequencies
for i=1:11
    switch mod(l_phone(1,i),3)
        case 0
            f1(i) = 1477;
        case 1
            f1(i) = 1209;
        case 2
            f1(i) = 1366;
    end
    switch floor((l_phone(1,i)-1)/3)
        case 0
            f2(i) = 697;
        case 1
            f2(i) = 770;
        case 2
            f2(i) = 852;
    end
    %handle exception
    if l_phone(1,i) == 0
        f1(i) = 1336;
        f2(i) = 941;
    end
end
tone = zeros(1,180224);
%generate sound
```

```
for i=1:11

    for t=(1+Fs/2*(i-1)):(Fs/4*i + Fs*(i-1)/4)
        tone(t) = (sin(2*pi*f1(i)*(t)/Fs)+sin(2*pi*f2(i)*t/Fs))/2;
    end
    %silent section for 0.25 seconds
    for t=((1+Fs/4*i)+ Fs*(i-1)/4):Fs/2*i
        tone(t) = 0;
    end
end
figure(1);
i=1;
range = (1+Fs/2*(i-1)):(Fs/4*i + Fs*(i-1)/4);
plot(range,tone(range));hold all;
i=4;
range = (1+Fs/2*(i-1)):(Fs/4*i + Fs*(i-1)/4);
plot((1:Fs/4),tone(range),'r');hold off;
title('Tone generation 0 and 2')
xlabel('Time');
ylabel('Amplitude');
figure(2);
spgrambw(tone, Fs, 'Jcw',20,2000);
title('Tone generation for a random London Number')
% spectrogram(tone,hann(8192),0,512,Fs)
% view(-90,90)
% set(gca,'ydir','reverse')
% spectrogram(tone,Fs/4,Fs/4-40,256,Fs);%[S,F,T,P] =
%surf(T,F,10*log10(P),'edgecolor','none'); axis tight;

%xlabel('Time (Seconds)'); ylabel('Hz');
% soundsc(tone,Fs)
```

Part 3.4

```
clear all;
clc;
close all;
addpath('voicebox');
order = 25;
N = 1;
Fs = 32768;
f1 = [1477 1209];
f2 = [697 941];
tone = zeros(1,Fs);
awgn = 0.00001*randn(Fs,1);
for i=1:2
    fprintf('start:%i   end:%i\n',(1+Fs/2*(i-1)),(Fs/4*i + Fs*(i-1)/4));
    for t=(1+Fs/2*(i-1)):(Fs/4*i + Fs*(i-1)/4)
        tone(t) = (sin(2*pi*f1(i)*(t)/Fs)+sin(2*pi*f2(i)*t/Fs))/2;
    end
    for t=((1+Fs/4*i)+ Fs*(i-1)/4):Fs/2*i
        tone(t) = 0;
    end
%     tone(i) = sin(2*pi*f1*i/Fs);
end
ptone = 0;
pnoise = 0;

for i = 1:length(tone)-1
    ptone = ptone + tone(i).^2;
```

```
    pnoise = pnoise + awgn(i).^2;
    tone(i) = tone(i) + awgn(i);
end
SNR = 10*log10(ptone/pnoise);
% plot((1:Fs/2*11)/Fs,tone)
figure(1);
plot((1:Fs)/Fs,tone)
figure(2);
xlabel('Time');
ylabel('Amplitude');
spgrambw(tone, Fs, 'Jcw',40,2000,60, [0 1]);
str = sprintf('3 and * tone generated with noise SNR=%2.2fdB',SNR);
title(str)
% soundsc(tone,Fs)
ar_three = aryule(tone(1:8192),order);
ar_star = aryule(tone(16385:24576),order);
std_three = std(tone(1:8192));
std_star = std(tone(16385:24576));
figure;
[h,w] = freqz(std_three^2,ar_three,512,Fs);
subplot(2,1,1)
plot(w/(2*pi),abs(h).^2)
grid on
xlabel('Frequency');
ylabel('PSD Amplitude');
str = sprintf('PSd of 3 tone corrupted by noise SNR=%2.2fdB with an AR order of %d',SNR,order);
title(str)
subplot(2,1,2)
[h,w] = freqz(std_star^2,ar_star,512,Fs);
plot(w/(2*pi),abs(h).^2)
grid on
xlabel('Frequency');
ylabel('PSD Amplitude');
str = sprintf('PSd of * tone corrupted by noise SNR=%2.2fdB with an AR order of %d',SNR,order);
title(str)
```