EE 779: Advanced Techniques in Signal Processing

Project Report

Topic: Speech Watermarking

Team Members:
Shruti Hiray: 14D070016
Shubham Yadav: 140070028

1. Introduction:

Digital watermarking is technique of embedding additional data in host signal which includes image, video, audio, speech, text etc. This technique can be used for authentication and protection purposes. We plan to use the digital watermarking method for speech signals using the Spread spectrum signaling, Autoregressive (AR) model & Psychoacoustic model.

Digital Watermarking techniques must satisfy :

1. Perceptual transparency
2. Robustness
3. Watermark must be resistant to unauthorized detection and difficult to forge

2. Motivation

While there has been a considerable amount of attention devoted to the techniques of spread-spectrum signaling for use in image and audio watermarking applications, there has only been a limited study for embedding data signals in speech. A lot of information is in the form of speech signals. Thus watermarking in speech is important so that:

- To prove ownership of signal
- Detection of tampering of original signal

Speech is an uncharacteristically narrow band signal given the perceptual capabilities of the human hearing system. Speech differs from music in their acoustic characteristics and watermarking requirements. Speech is an acoustically rich signal that it uses only a small portion of the human perceptual range (10 Hz to 8 kHz) making it a narrow bandwidth signal. Thus the distinct characteristics of speech motivates for a novel method for watermarking

3. Implementation:

3.1 Watermark embedding using Direct spread spectrum and autoregressive modelling

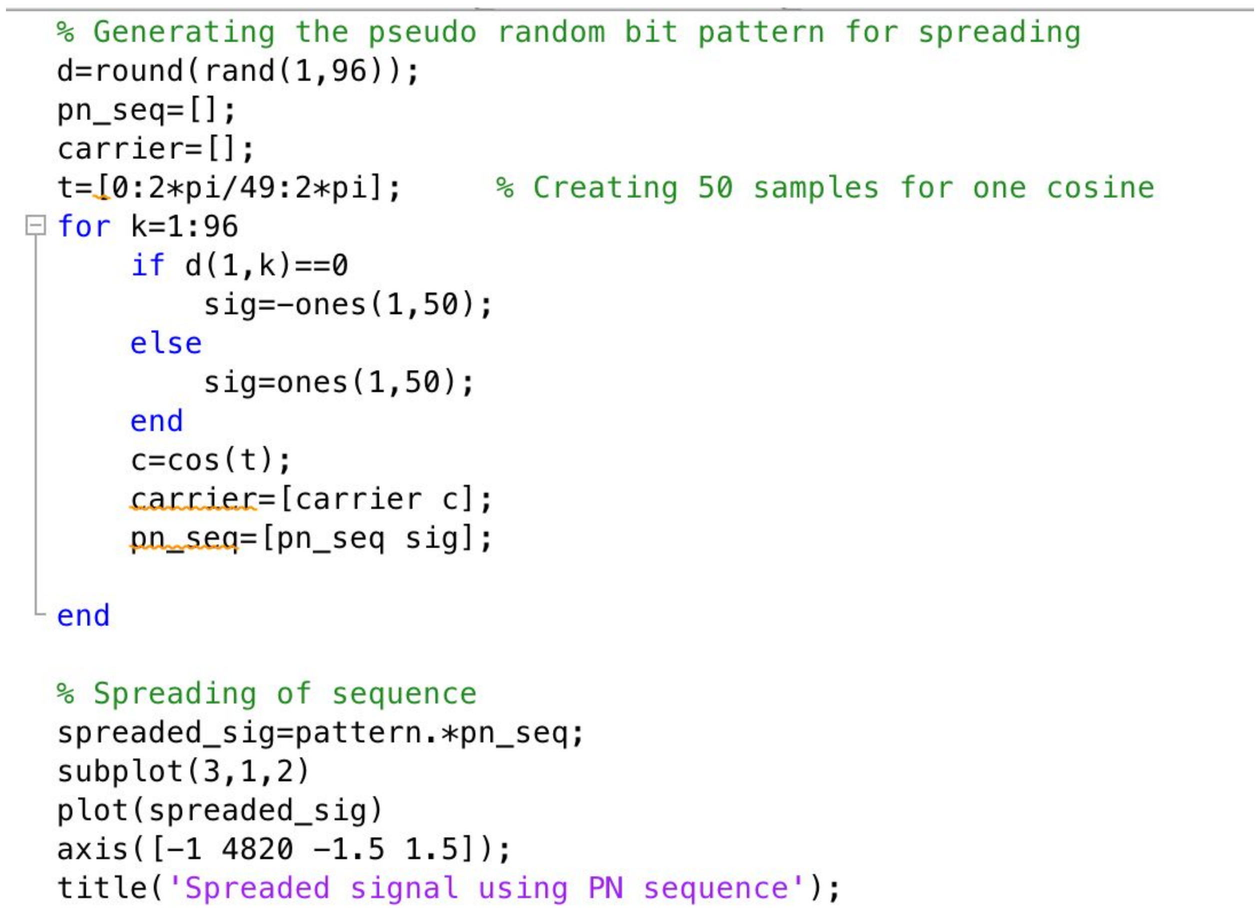
3.1.1 Spread spectrum signaling:

The speech signal is a considerably narrower bandwidth signal. The long time-averaged power spectral density of speech indicates that the signal is confined to a range of approximately 10 Hz to 8 kHz. In order that the watermark survives typical transformation of speech signals, it is important that the watermark be limited to the perceptually relevant portions of the spectra. However, the watermark should remain imperceptible to Human Auditory System (HAS). Therefore, a spread-spectrum signal with an uncharacteristically narrow bandwidth will be used.

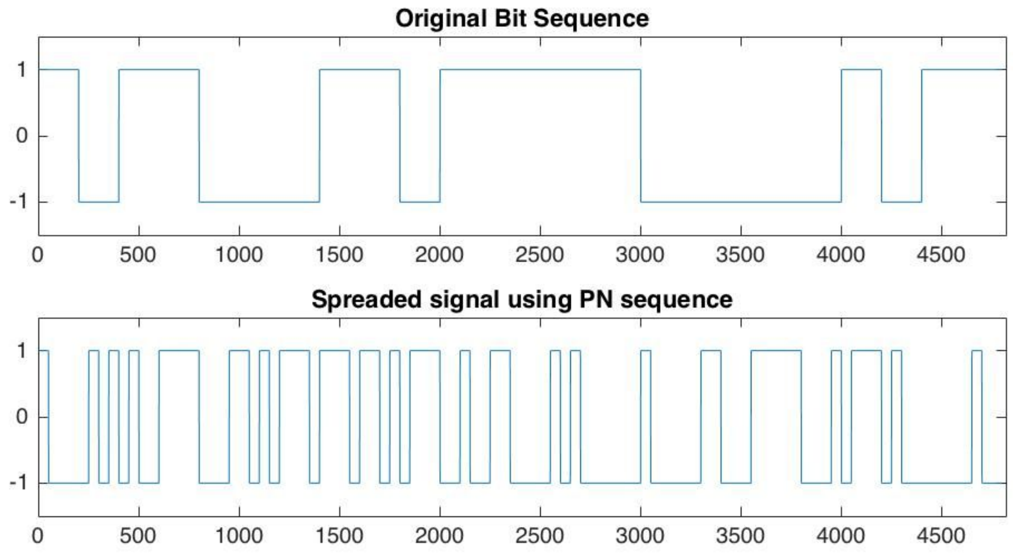
PN-sequences form the basis of our watermarking scheme because of their noise-like characteristics, resistance to interference, and good autocorrelation properties. Spread spectrum communication systems use pseudo-noise (PN) sequences to modulate transmitted data into noise-like wideband signals so they blend into the background. Spread spectrum signals are resistant to interference such as unintentional interference, channel noise, multiple users, multi-path interference, or intentional jammers and thus provide robustness to the watermarked signal.[3]

In Direct Sequence Spread Spectrum Coding (DSSS), the signature, a binary codeword or a watermark signal, is modulated using a PN-sequence. It is then added to the original signal as additive random noise.

Matlab code for modelling DSSS:



```
% Generating the pseudo random bit pattern for spreading
d=round(rand(1,96));
pn_seq=[];
carrier=[];
t=[0:2*pi/49:2*pi];      % Creating 50 samples for one cosine
for k=1:96
    if d(1,k)==0
        sig=-ones(1,50);
    else
        sig=ones(1,50);
    end
    c=cos(t);
    carrier=[carrier c];
    pn_seq=[pn_seq sig];

end

% Spreading of sequence
spreaded_sig=pattern.*pn_seq;
subplot(3,1,2)
plot(spreaded_sig)
axis([-1 4820 -1.5 1.5]);
title('Spreaded signal using PN sequence');
```



The simplest implementation of a speech watermark system would involve adding this signal, which sounds primarily like radio static, to the speech signal at the appropriate gain. However, taking advantage of our knowledge of the speech signal itself, we are able to embed a significantly higher gain signal using techniques

3.1.2 Autoregressive modelling:

To add as much watermark as much possible while still satisfying the constraint that the added signal not be perceivable when listened to. Using the *production model* for speech, the AR modelling technique is highly efficient for speech signal. In addition, human speech perception is related to production characteristics, thus production model enables hiding characteristics for watermarks.
The source-filter model of speech production, (production model) suggests that speech is comprised of voiced (periodic pulses) and unvoiced component (which can be represented by white Gaussian excitation signal) .

Before filtering the watermark signal using the all-pole filter, a bandwidth expansion operation is performed. This moves all of the poles closer to the center of the unit circle, increasing the bandwidth of their respective resonances. The vocal tract filter often tends to have quite narrow spectral peaks. Due to masking phenomena, sounds near these peaks are unlikely to be perceived by the listener.

Therefore, by increasing the bandwidth of formant responses, larger overall watermark signal gains should be tolerable. The bandwidth parameter γ is used to adjust the all-pole filter coefficients. [2]

$$a(i)' = a(i) * \gamma^{i}$$

where γ is chosen between 0 & 1.

Autoregressive modelling involves computing the coefficients of the all-pole filter model using Levinson - Durbin algorithm.

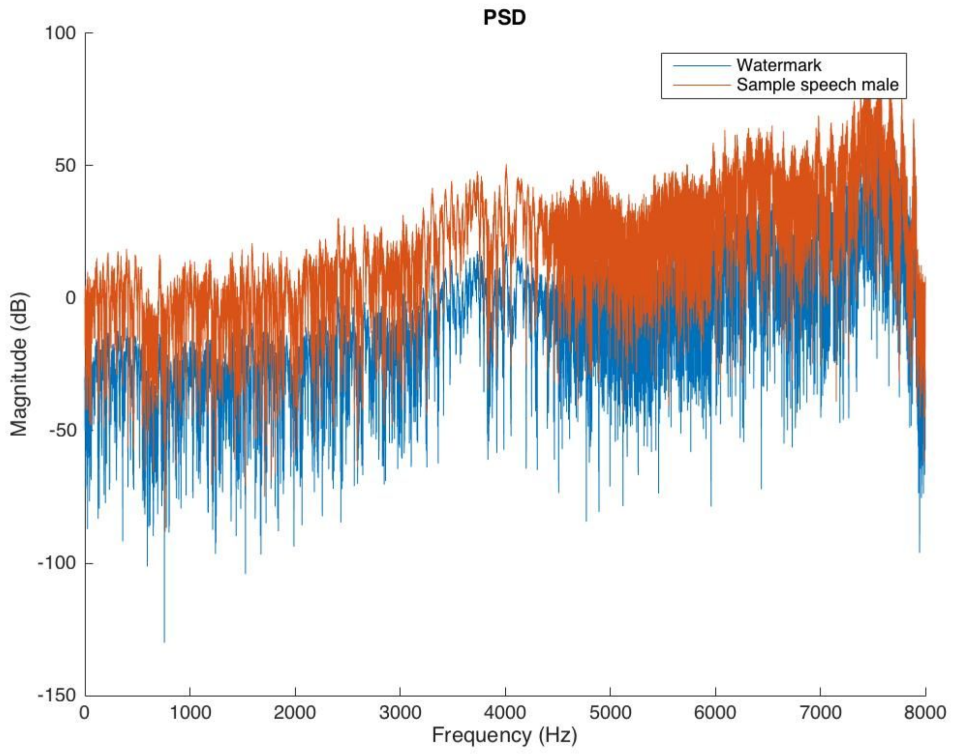
All-pole filter of order p is given by:

$$A(z) = 1/ (a0 + a1z^{-1} + a2z^{2} + a3z^{3} + ... + apz^{-p})$$

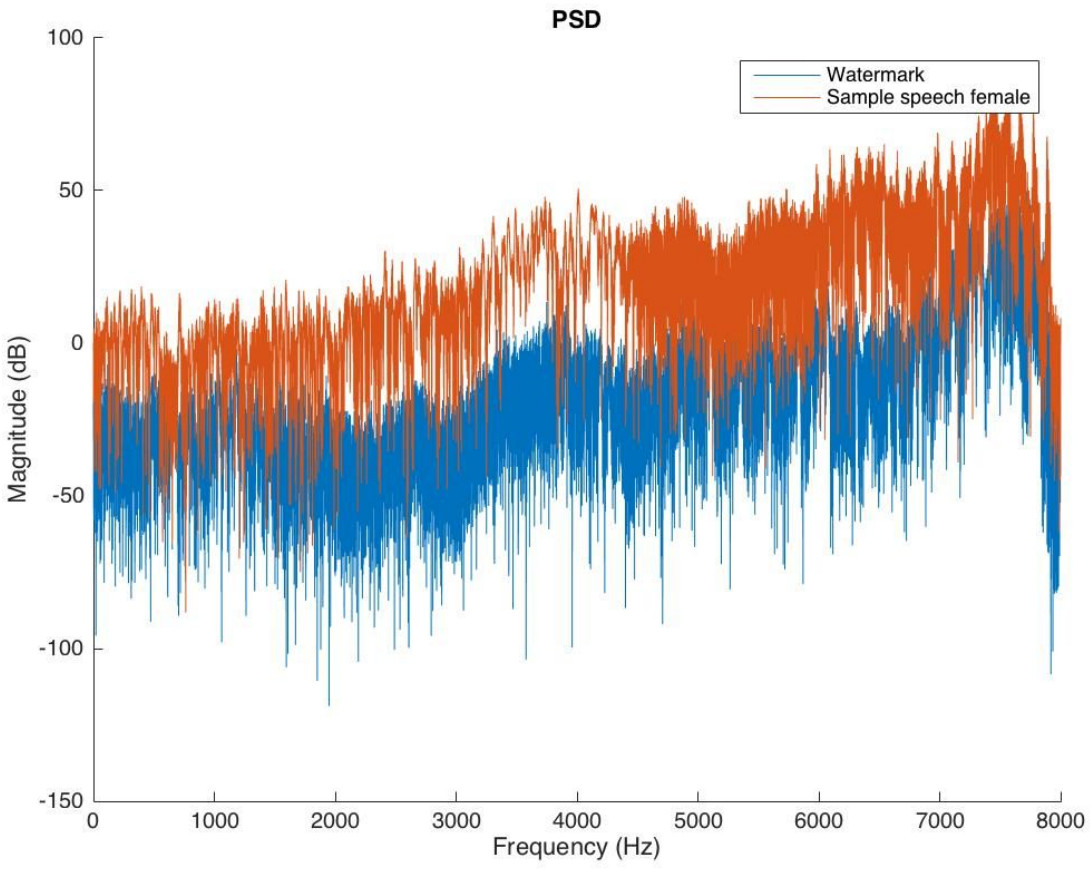
In the watermark embedding algorithm, the watermark signal is filtered using the all pole filter to match the overall spectral shape of the speech signal. The speech signal is divided into frames comprising approximately of a phoneme. The frame is windowed using Hamming window. The coefficients are obtained for these windowed frames.

The following are the plots of Power Spectral Density after the watermark is filtered using the sample speech:

1) Sample speech of male speaker



2) Sample speech of female speaker



We see that the watermark signal approximately follows the spectral shape of the sample speech signal.

3.1.3 Watermark signal gain:

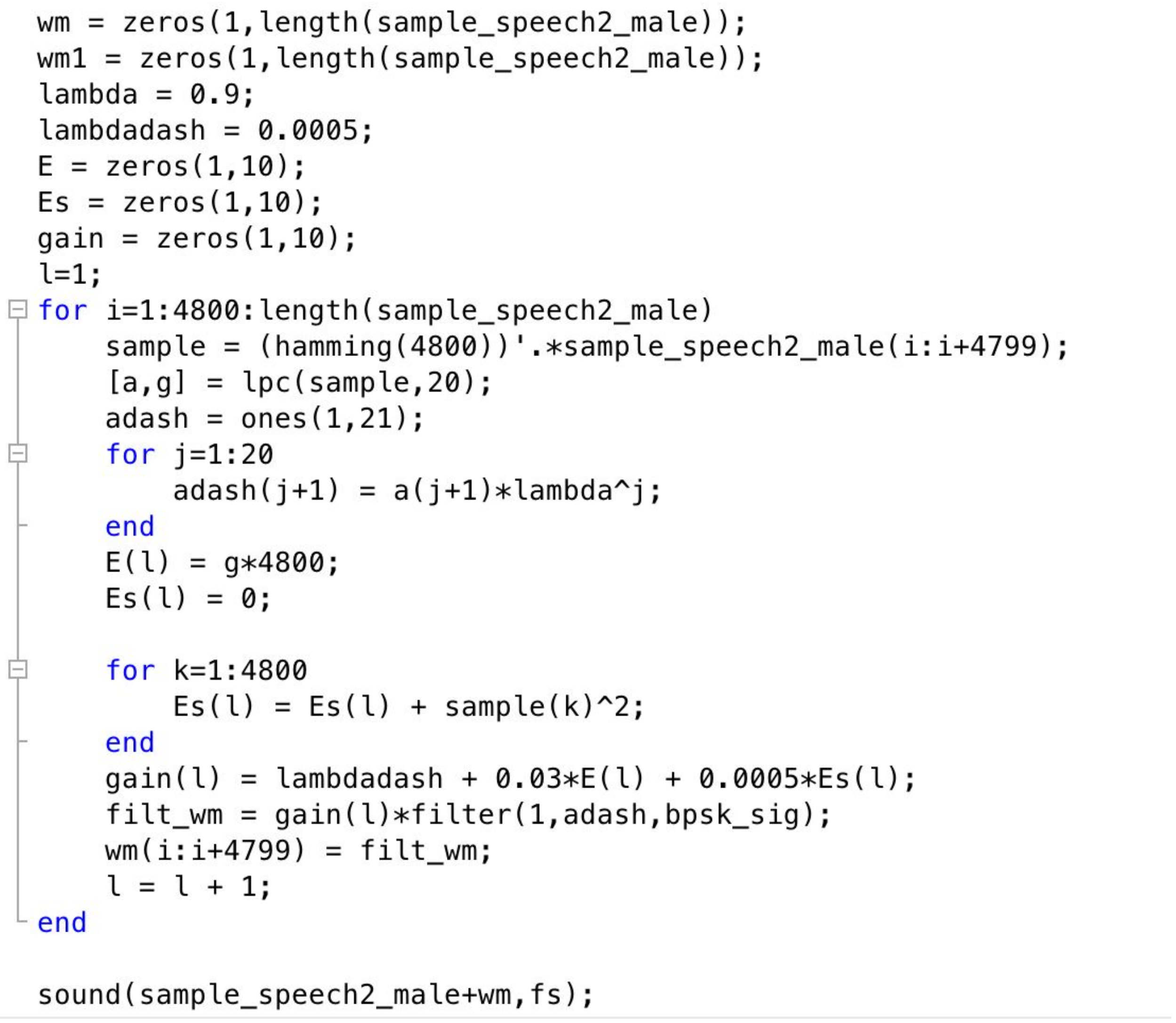
The instantaneous watermark gain is dynamically determined to match the characteristics of the speech signal. In the simplest case, when little speech energy is present (i.e. during silence) the watermark is added using a fixed gain threshold. This is selected so that the watermark becomes the effective noise floor of the recording. Perceptually, a small amount of noise is always expected in a recording and the watermark signal is not atypical of such recording noise. [2]

The watermark gain in each frame can be determined by the linear combination of the gains for silence, normalized per sample residual energy (E) , and normalized per sample speech energy (E_s).

$$g(t) = \lambda_0 + \lambda_0 E + \lambda_0 E_s$$

which maximizes the strength of the watermark signal without incurring perceptual degradations. If the gain is increased further, there is hoarseness in the watermarked speech and the difference with original speech is indeed perceptible.

Matlab Code for AR modelling & applying gain to watermark:



```
wm = zeros(1,length(sample_speech2_male));
wm1 = zeros(1,length(sample_speech2_male));
lambda = 0.9;
lambdadash = 0.0005;
E = zeros(1,10);
Es = zeros(1,10);
gain = zeros(1,10);
l=1;
for i=1:4800:length(sample_speech2_male)
    sample = (hamming(4800))'.*sample_speech2_male(i:i+4799);
    [a,g] = lpc(sample,20);
    adash = ones(1,21);
    for j=1:20
        adash(j+1) = a(j+1)*lambda^j;
    end
    E(l) = g*4800;
    Es(l) = 0;

    for k=1:4800
        Es(l) = Es(l) + sample(k)^2;
    end
    gain(l) = lambdadash + 0.03*E(l) + 0.0005*Es(l);
    filt_wm = gain(l)*filter(1,adash,bpsk_sig);
    wm(i:i+4799) = filt_wm;
    l = l + 1;
end

sound(sample_speech2_male+wm,fs);
```

3.2 Watermark Embedding using Psychoacoustic :

3.2.1 Psychoacoustic Modelling :

The Second Method of applying Watermarking is based on the taking the benefit of hearing system properties of Human ears.This watermarking system uses an algorithm that relies on the principles mentioned in earlier section, to generate a digital watermark and embed it into the original audio file (which is in .wav format) by spectrally shaping the watermark signal.

The psychoacoustic auditory model is an algorithm to imitate the Human Auditory System (HAS). HAS is sensitive to a very wide dynamic range of amplitude of one billion to one and of frequency of one thousand to one. It is also acutely sensitive to the additive random noise. These properties of HAS makes it all the more challenging to tamper with any kind of audio signal.
However, there are a few “holes” in the auditory system. While the HAS has a very large dynamic range, it has a very small differential range. This is called the masking effect of HAS. There are two types of masking observed in the HAS – frequency masking and temporal masking.

We use the Frequency Masking, in which simultaneous frequency masking is closely studied to be used for watermark shaping purposes. The auditory model processes the audio information to produce information about the final masking threshold. The final masking threshold information is used to shape the generated audio watermark. This shaped watermark is ideally imperceptible for the average listener.

3.2.2 Steps :

To overcome the potential problem of the audio signal being too long to be processed all at the same time, and also extract quasi-periodic sections of the waveform, the signal is segmented in short overlapping segments, processed and added back together. Each one of these segments is called a “frame.” The steps involved in creating a Psychoacoustic Auditory Model include:

(a) Fast Fourier Transform

The cochlea can be considered as a mechanical to electrical transducer, and its function is to make a time to frequency transformation of the audio signal. To be more specific, the audio information, in time, is translated in first instance into a frequency-spatial representation inside the basilar membrane. This spatial representation is perceived by the nervous system and translated into a frequency-electrical representation
This can be easily modeled by taking FFT of frames (STFT).

(b) Power spectra

Power spectrum or PSD is then calculated. And the frequencies are masked to Bark scale , using formulas :

$$z = 13\tan^{-1}\left(\frac{0.76 * f}{1000}\right) + 3.5\tan^{-1}\left(\left(\frac{f}{7500}\right)^2\right)$$

where f is the frequency in Hertz and z is the mapped frequency in Barks.

The Power spectrum is simply :

$$\begin{aligned} Sp(j\omega) &= \mathrm{Re}\{Sw(j\omega)\}^2 + \mathrm{Im}\{Sw(j\omega)\}^2 \\ &= |Sw(j\omega)|^2 \end{aligned}$$

(c) Energy per critical band (bark scale)

The energy per critical band is sum of the PSD in each band :

$$Spz(z) = \sum_{\omega=LBZ}^{HBZ} Sp(j\omega)$$

LBZ, HBZ represents lower and higher frequency limits

(d) Spread Masking using basilar function

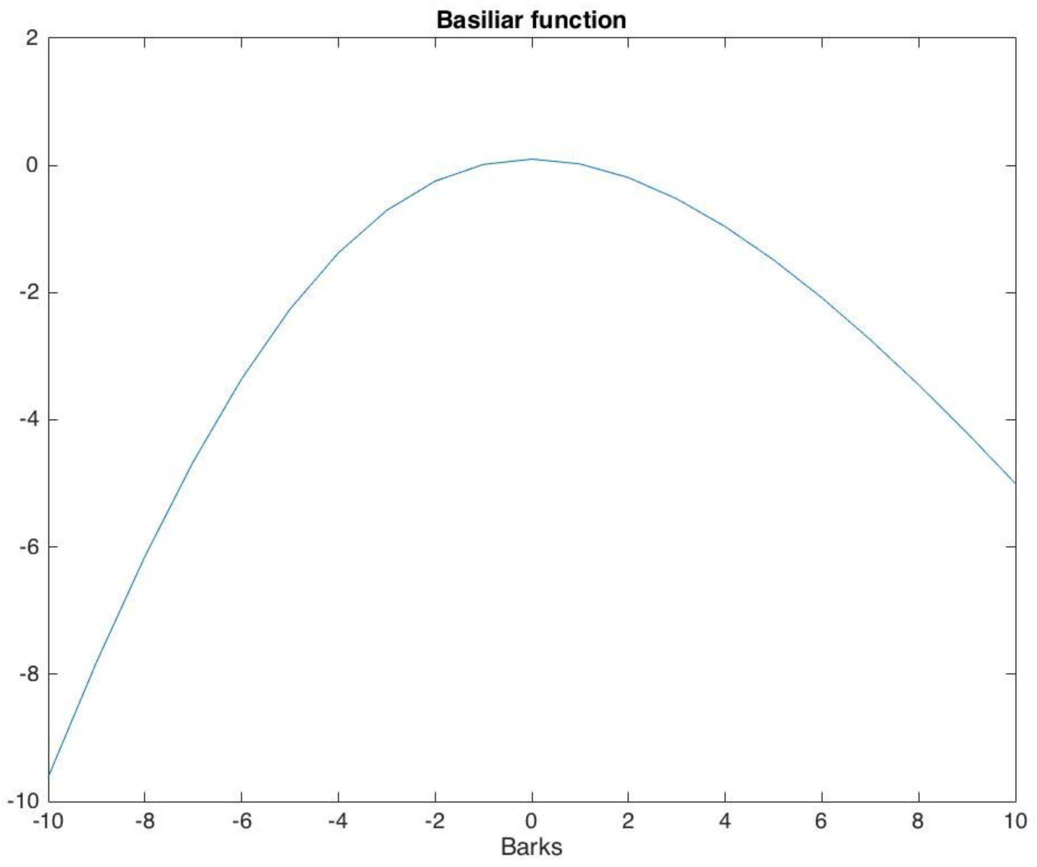
The model to approximate the Basilar membrane spreading is :

$$B(z) = 15.91 + 7.5(z + 0.474) - 17.5\sqrt{1 + (z + 0.474)^2}$$

The spread across critical bands Sm(z) is given by convolution :

$$Sm(z) = Spz(z) * B(z)$$

This represents the energy per critical bands after taking into account the masking occasioned by neighbouring bands.



(e) Masking threshold estimation

For Threshold estimation, we need to estimate first the Threshold in quiet and then shape the sound according to that :

(i) SFM (Spectral Flatness Measure) and Tonality Factor (alpha) :

The SFM is used to determine if actual frame is noise like or tone like and then to select appropriate masking index formula. The SFM is defined as the ratio of GM to AM of Spz(z) , expressed in dB.

$$SFM_{dB} = 10\log 10\left\{\frac{\prod_{z=1}^{Zt} Spz(z)}{\frac{1}{Zt}\sum_{z=1}^{Zt} Spz(z)}\right\}^{\frac{1}{Zt}}$$

with Zt = total number of critical bands on the signal

The Tonality Factor is further defined as minimum of 1 and the ratio of SFM to SFM-max which is taken as -60dB. This tonality factor will help to select the right masking index for actual frame.

So, Frame is noise-like → alpha is close to 0

And if Frame is tone-like → alpha is close to 1

Which further decides the Energy Offset O(z).

$$O(z) = \alpha(14.5 + z) + (1 - \alpha)5.5$$

The offset O(z)is subtracted from the spread masking threshold to estimate the raw masking threshold Traw (z).

$$Traw(z) = 10^{\left(\log_{10}(Sm(z)) - \frac{O(z)}{10}\right)}$$

(ii) Threshold Normalization :

The use of the spreading function B (z) increases the energy level in each one of the critical bands of the spectrum Sm (z) . This effect has to be undone using a normalization technique, to return Traw (z) to the desired level. The energy per critical band calculated is also affected by the number of components in each critical band.

$$Tnorm(z) = \frac{Traw(z)}{P_z}$$

P_z = number of points in each band z

(iii) Final Masking Threshold :

After normalization, the last step is to take into account the absolute auditory threshold or "hearing threshold." That is defined as a sinusoidal tone of 4000 Hz with one bit of dynamic range

$$TH = \max(|Pp(j\omega)|)$$

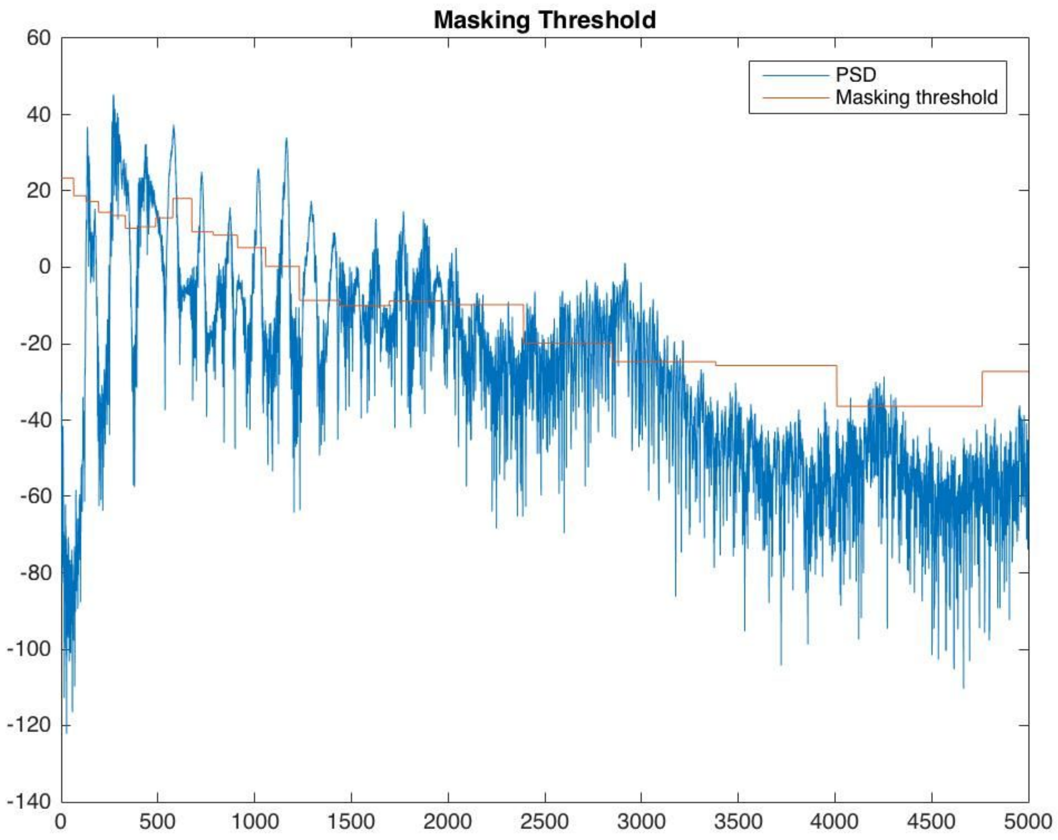
$Pp(j\omega)$ = power spectrum of the probe signal $p(t)$

$$p(t) = \sin(2\pi\, 4000t)$$

The final threshold $T(z)$ is:

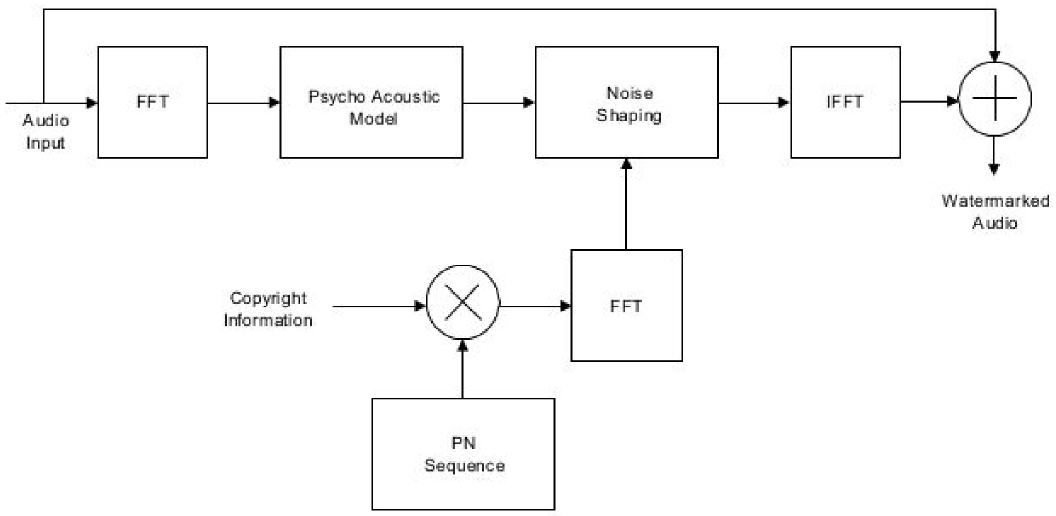
$$T(z) = \max(Tnorm(z), TH)$$

The final step is to use this Threshold to do the Noise shaping of Watermarking.
The output is the summation of both frames, which is after taking IFFT, is added with other frames to generate the **Embedded sound.**



3.2.3 Block diagram :

The per frame processing can be described in the block diagram [4] :



Where the PsychoAcoustic Model can further rbe expanded as [6]:

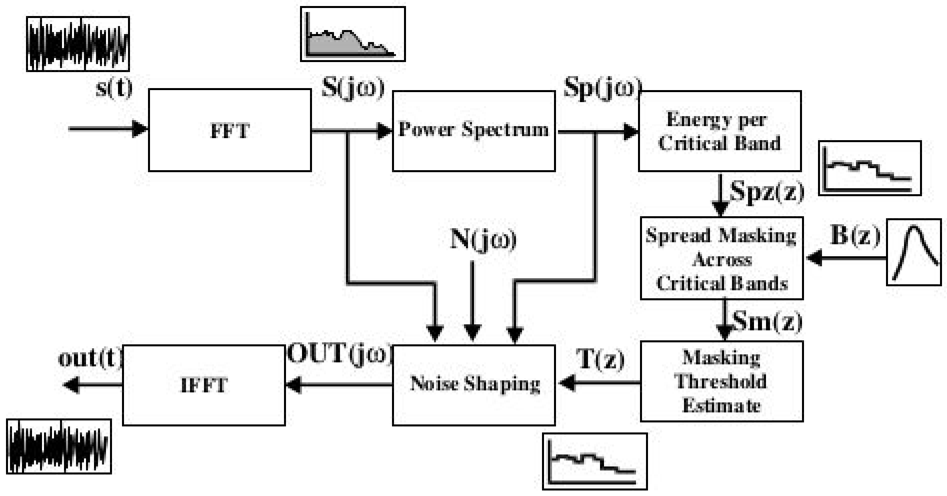
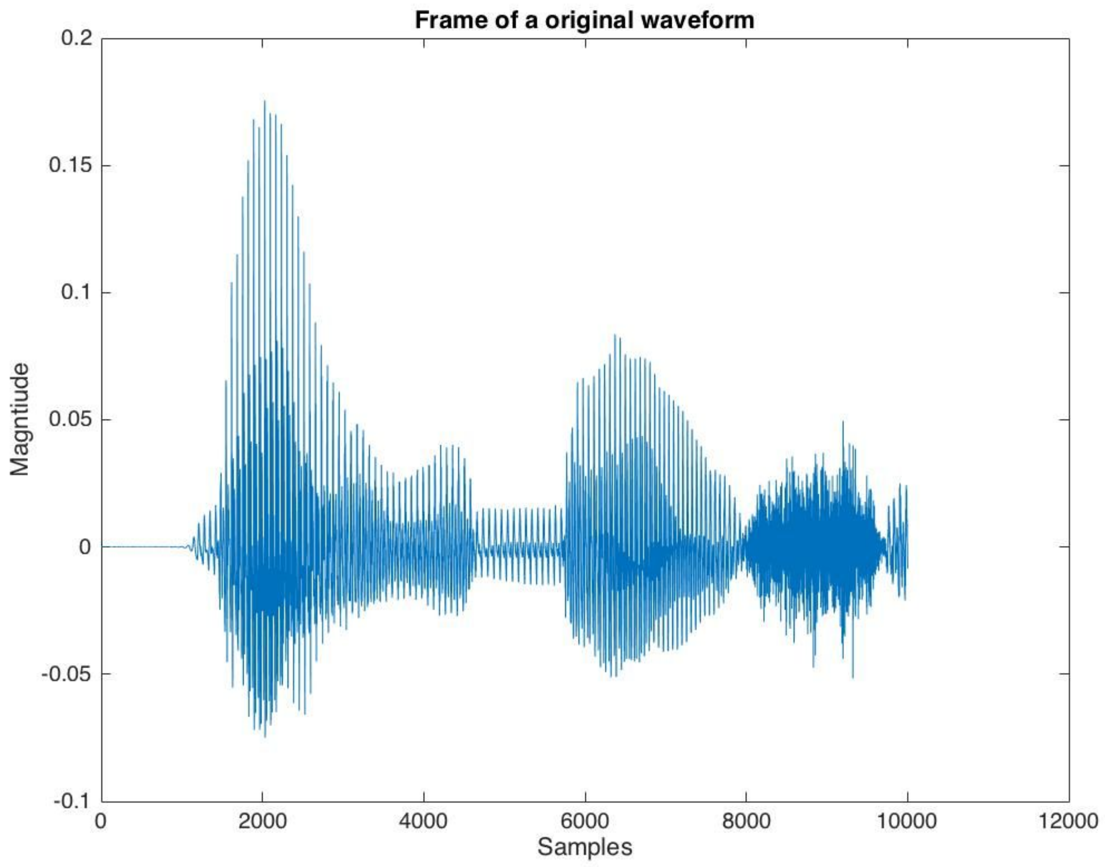
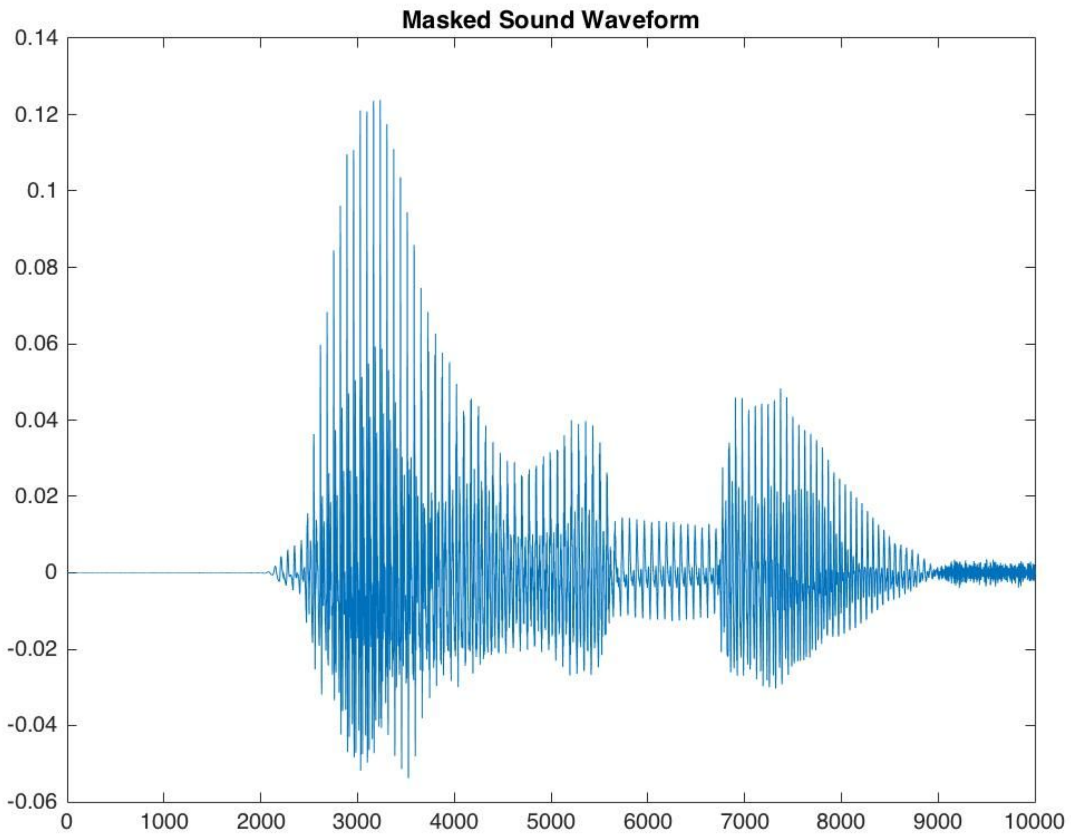


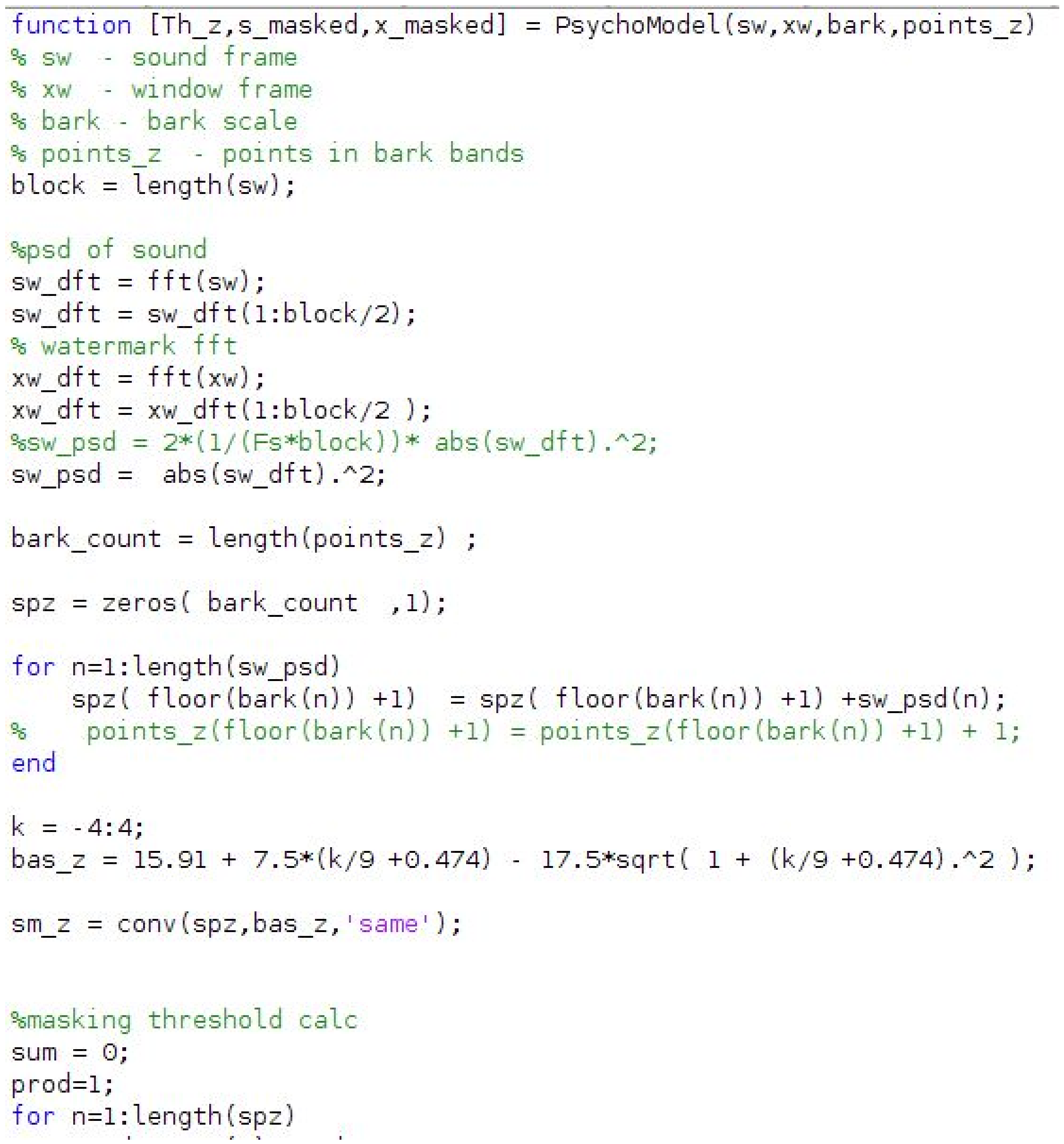
Figure 1. Psychoacoustic auditory model



Frame of a original waveform
Magntiude
Samples
0.2
0.15
0.1
0.05
0
-0.05
-0.1
0
2000
4000
6000
8000
10000
12000

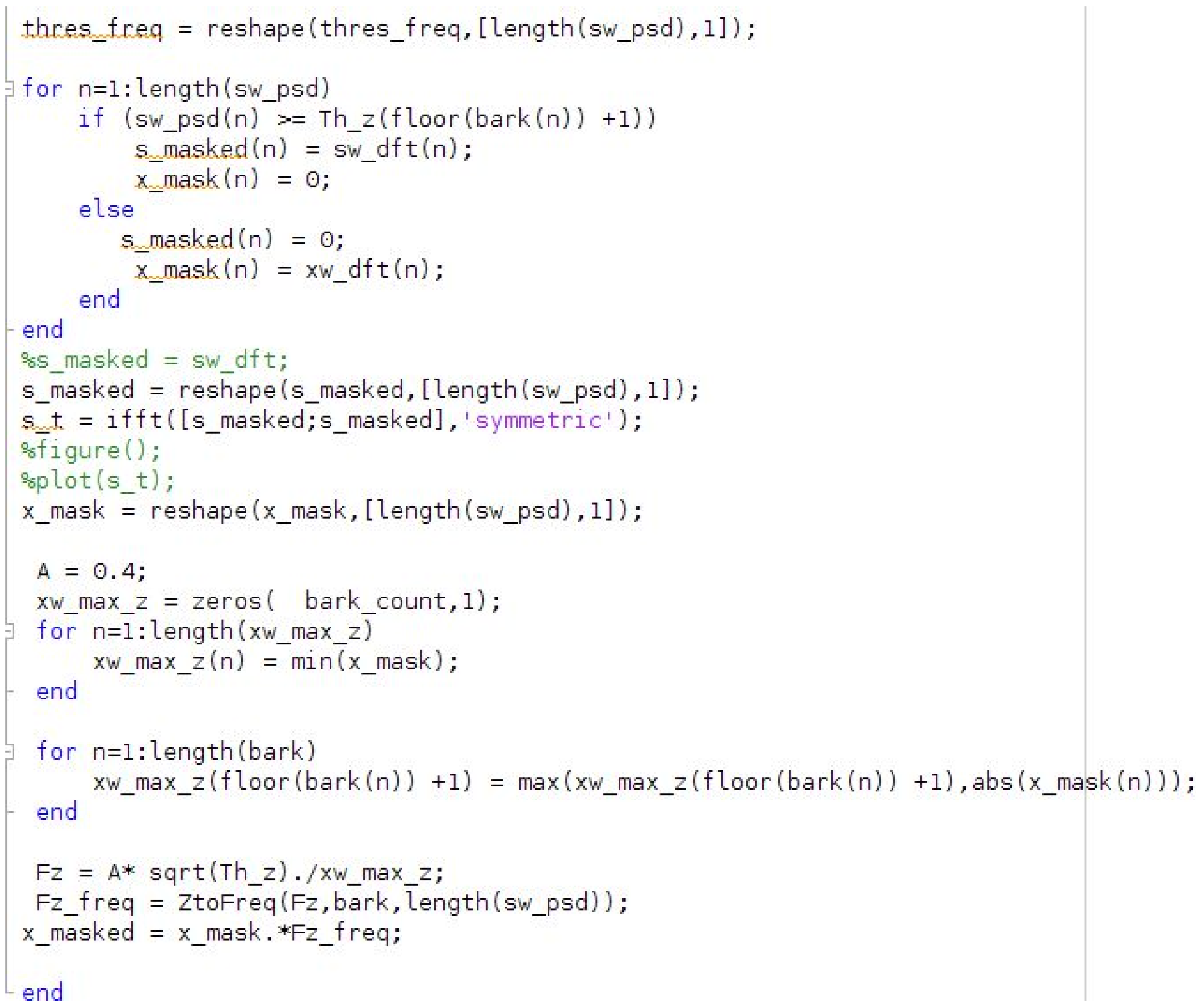

Masked Sound Waveform
0.14
0.12
0.1
0.08
0.06
0.04
0.02
0
-0.02
-0.04
-0.06
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000

Matlab Code for PsychoAcoustic Modelling



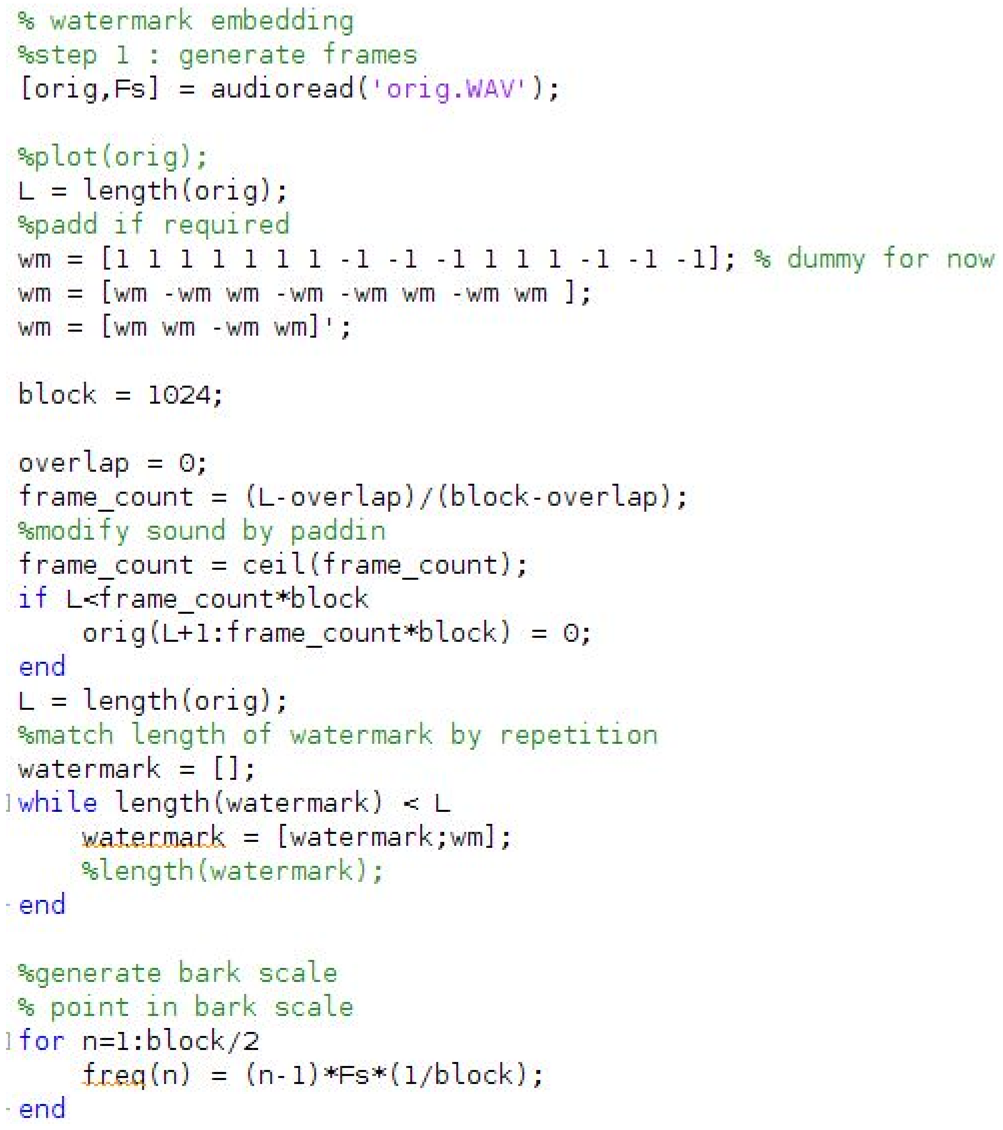
```
function [Th_z,s_masked,x_masked] = PsychoModel(sw,xw,bark,points_z)
% sw  - sound frame
% xw  - window frame
% bark - bark scale
% points_z  - points in bark bands
block = length(sw);

%psd of sound
sw_dft = fft(sw);
sw_dft = sw_dft(1:block/2);
% watermark fft
xw_dft = fft(xw);
xw_dft = xw_dft(1:block/2 );
%sw_psd = 2*(1/(Fs*block))* abs(sw_dft).^2;
sw_psd =  abs(sw_dft).^2;

bark_count = length(points_z) ;

spz = zeros( bark_count  ,1);

for n=1:length(sw_psd)
    spz( floor(bark(n)) +1)  = spz( floor(bark(n)) +1) +sw_psd(n);
%    points_z(floor(bark(n)) +1) = points_z(floor(bark(n)) +1) + 1;
end

k = -4:4;
bas_z = 15.91 + 7.5*(k/9 +0.474) - 17.5*sqrt( 1 + (k/9 +0.474).^2 );

sm_z = conv(spz,bas_z,'same');


%masking threshold calc
sum = 0;
prod=1;
for n=1:length(spz)
```

```
    sum = sum + spz(n);
end
prod = prod^( 1/length(spz));
sum = sum/length(spz);
sfm_db = 10*log10( prod/sum );
sfm_db_max = -60; %sign?
alpha = min(1, sfm_db/sfm_db_max); %tonality factor

for z=1:bark_count
offset_z(z) = alpha*(14.5+z) + (1-alpha)*5.5;
end
offset_z = reshape(offset_z,[bark_count,1]);

Traw_z = log10(sm_z) - offset_z/10;
Traw_z = abs(10.^Traw_z);
Tnorm_z = Traw_z./points_z;
%plot(Tnorm_z);
Tnorm_freq = ZtoFreq(Tnorm_z,bark,length(sw_psd));

%threshold in quiet
t=-block:block;
t=t';
p_t = sin(2*pi*4000*t);
p_fft = fft(p_t);
p_fft = p_fft(1:block);

th_q = max(abs(p_fft));

Th_z = max(Tnorm_z,th_q);
%figure();
for n=1:length(sw_psd)
    thres_freq(n )  = Th_z( floor(bark(n)) +1) ;
end
```

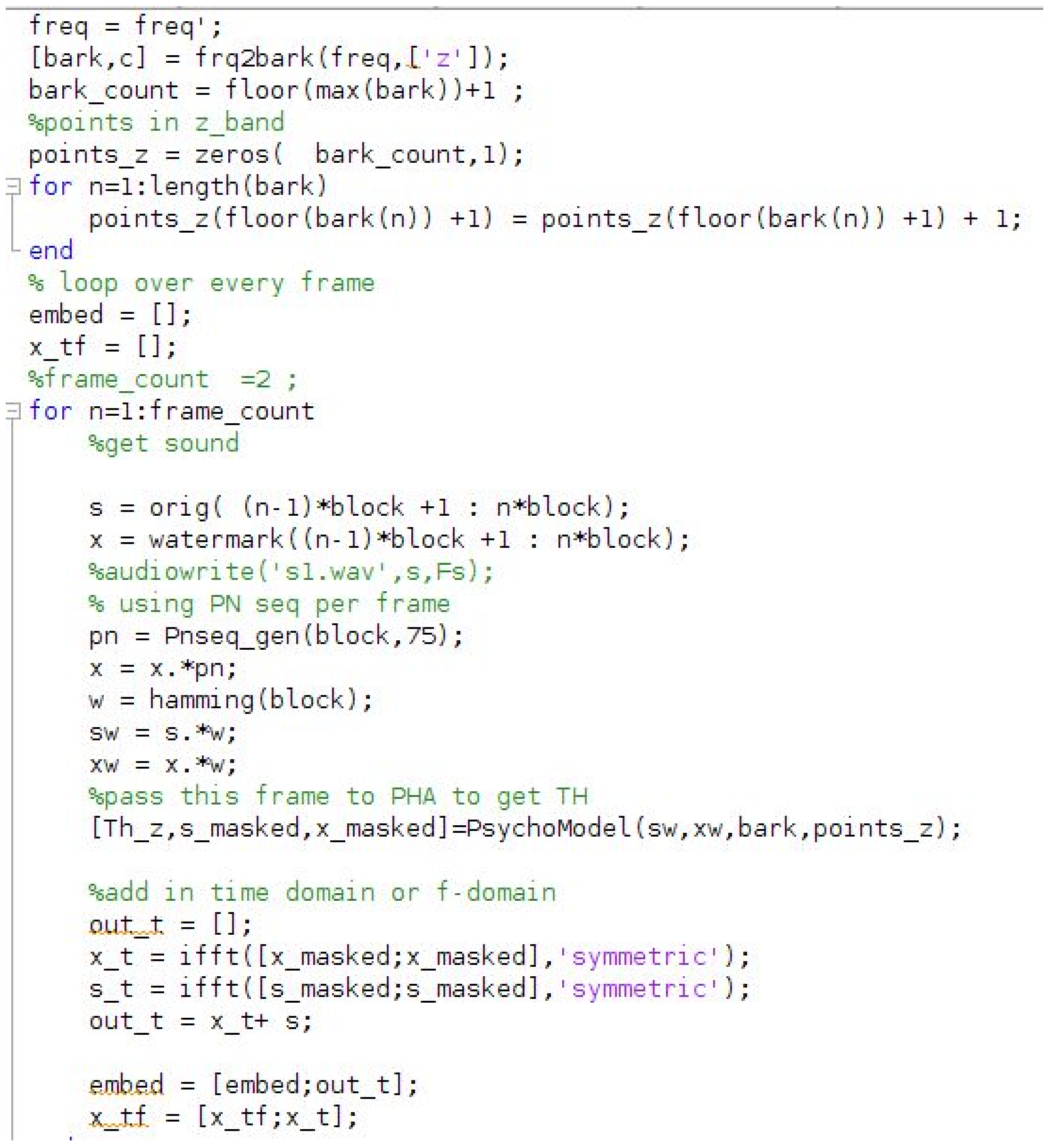


```
thres_freq = reshape(thres_freq,[length(sw_psd),1]);

for n=1:length(sw_psd)
    if (sw_psd(n) >= Th_z(floor(bark(n)) +1))
        s_masked(n) = sw_dft(n);
        x_mask(n) = 0;
    else
       s_masked(n) = 0;
        x_mask(n) = xw_dft(n);
    end
end
%s_masked = sw_dft;
s_masked = reshape(s_masked,[length(sw_psd),1]);
s_t = ifft([s_masked;s_masked],'symmetric');
%figure();
%plot(s_t);
x_mask = reshape(x_mask,[length(sw_psd),1]);

 A = 0.4;
 xw_max_z = zeros(  bark_count,1);
 for n=1:length(xw_max_z)
     xw_max_z(n) = min(x_mask);
 end

 for n=1:length(bark)
     xw_max_z(floor(bark(n)) +1) = max(xw_max_z(floor(bark(n)) +1),abs(x_mask(n)));
 end

 Fz = A* sqrt(Th_z)./xw_max_z;
 Fz_freq = ZtoFreq(Fz,bark,length(sw_psd));
x_masked = x_mask.*Fz_freq;

end
```

Matlab Code for watermark embedding



```
% watermark embedding
%step 1 : generate frames
[orig,Fs] = audioread('orig.WAV');

%plot(orig);
L = length(orig);
%padd if required
wm = [1 1 1 1 1 1 1 -1 -1 -1 1 1 1 -1 -1 -1]; % dummy for now
wm = [wm -wm wm -wm -wm wm -wm wm ];
wm = [wm wm -wm wm]';

block = 1024;

overlap = 0;
frame_count = (L-overlap)/(block-overlap);
%modify sound by paddin
frame_count = ceil(frame_count);
if L<frame_count*block
    orig(L+1:frame_count*block) = 0;
end
L = length(orig);
%match length of watermark by repetition
watermark = [];
while length(watermark) < L
    watermark = [watermark;wm];
    %length(watermark);
end

%generate bark scale
% point in bark scale
for n=1:block/2
    freq(n) = (n-1)*Fs*(1/block);
end
```



```
freq = freq';
[bark,c] = frq2bark(freq,['z']);
bark_count = floor(max(bark))+1 ;
%points in z_band
points_z = zeros(  bark_count,1);
for n=1:length(bark)
    points_z(floor(bark(n)) +1) = points_z(floor(bark(n)) +1) + 1;
end
% loop over every frame
embed = [];
x_tf = [];
%frame_count  =2 ;
for n=1:frame_count
    %get sound

    s = orig( (n-1)*block +1 : n*block);
    x = watermark((n-1)*block +1 : n*block);
    %audiowrite('s1.wav',s,Fs);
    % using PN seq per frame
    pn = Pnseq_gen(block,75);
    x = x.*pn;
    w = hamming(block);
    sw = s.*w;
    xw = x.*w;
    %pass this frame to PHA to get TH
    [Th_z,s_masked,x_masked]=PsychoModel(sw,xw,bark,points_z);

    %add in time domain or f-domain
    out_t = [];
    x_t = ifft([x_masked;x_masked],'symmetric');
    s_t = ifft([s_masked;s_masked],'symmetric');
    out_t = x_t+ s;

    embed = [embed;out_t];
    x_tf = [x_tf;x_t];
```

3.3 Watermark Detection

The received signal $r_o(t)$ is given by,

$$ro(t) = w(t) + s(t) + I(t)$$

where w(t) is the AR-shaped watermark signal, s(t) is the speech signal & I(t) is attacks on the signal.

Applying AR model on the received signal, the all-pole filter model coefficients are obtained. Inverse filtering of $r_o(t)$ gives $r(t)$.

After inverse LPC filtering, voiced speech becomes periodic pulses, and unvoiced speech becomes whitened noise. As is typical for speech processing, we model the inverse filtered as White Gaussian Noise (WGN). Inverse LPC filtering decorrelates the speech samples as well as equalizes the watermark signal w(t).

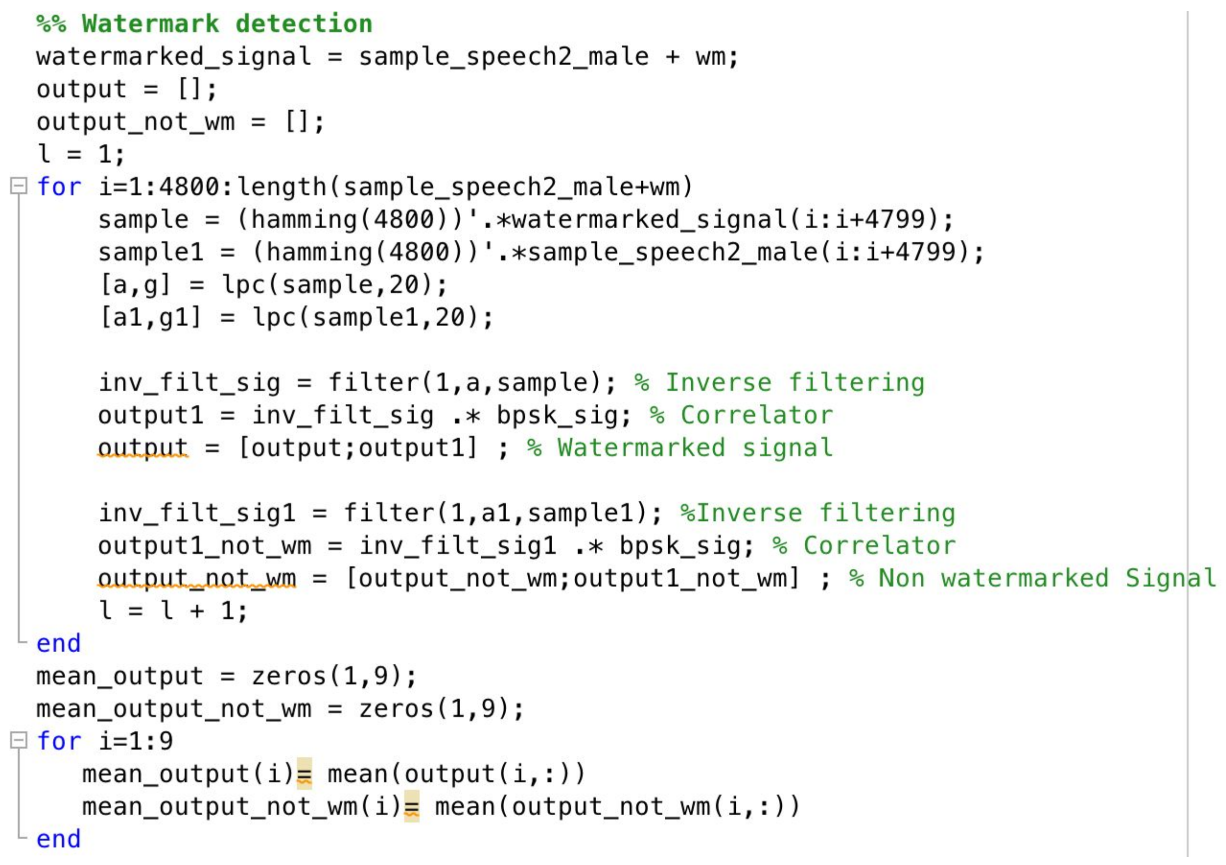
A correlation receiver,

$$d(t)r(t) \geq 0$$

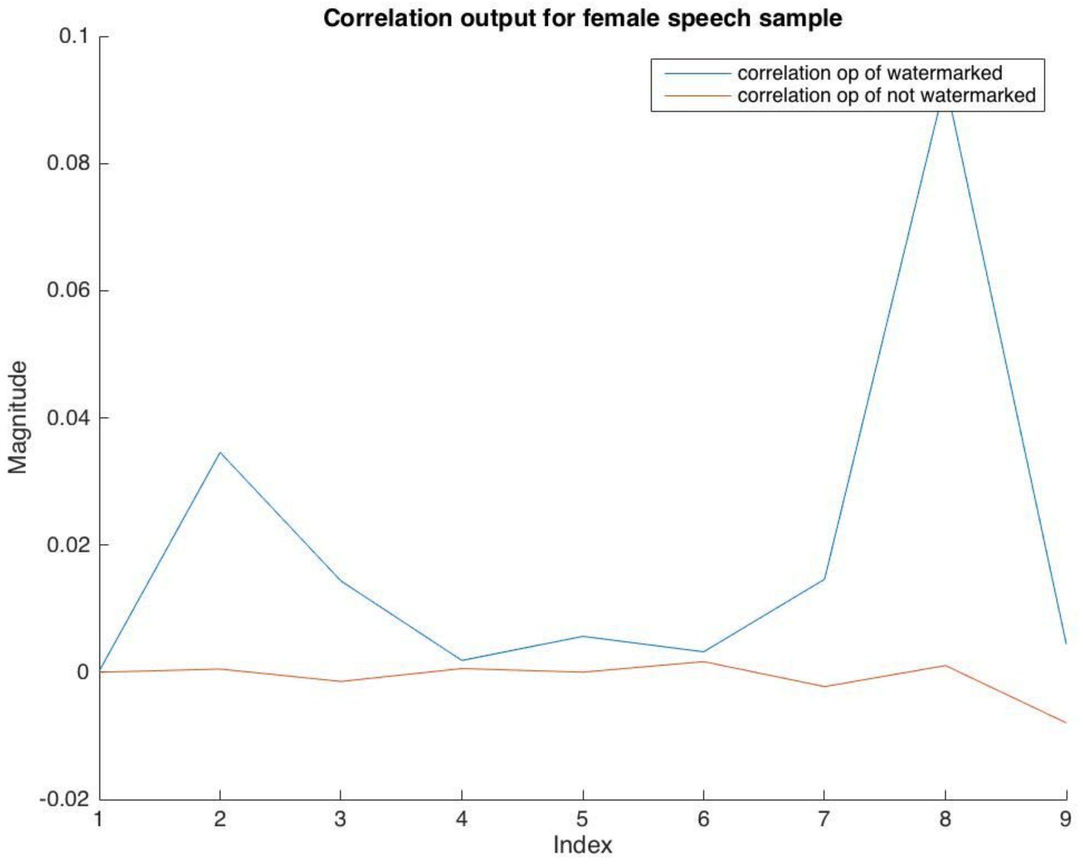
gives optimum detection performance in AWGN.

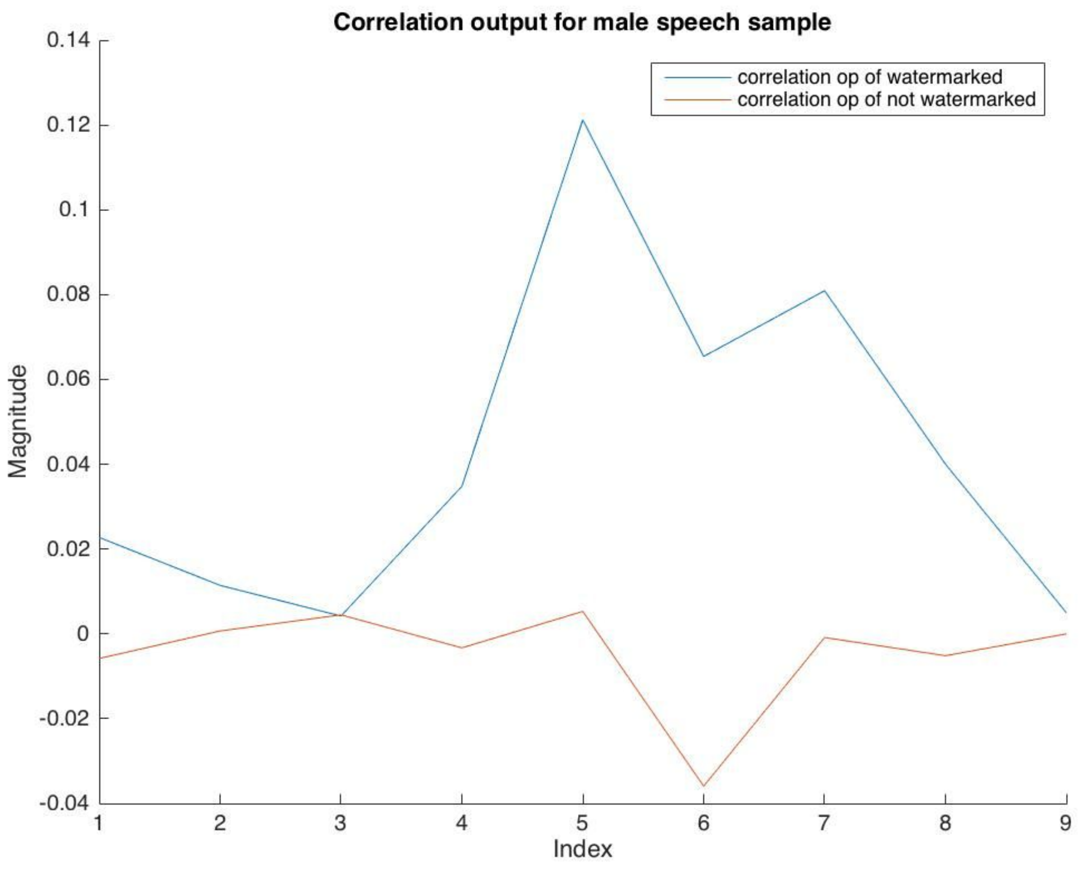
Where d(t) is despreading function and r(t) is received signal.
The correlation with d(t) can average out the interference.

Matlab code for watermark detection:



```
%% Watermark detection
watermarked_signal = sample_speech2_male + wm;
output = [];
output_not_wm = [];
l = 1;
for i=1:4800:length(sample_speech2_male+wm)
    sample = (hamming(4800))'.*watermarked_signal(i:i+4799);
    sample1 = (hamming(4800))'.*sample_speech2_male(i:i+4799);
    [a,g] = lpc(sample,20);
    [a1,g1] = lpc(sample1,20);

    inv_filt_sig = filter(1,a,sample); % Inverse filtering
    output1 = inv_filt_sig .* bpsk_sig; % Correlator
    output = [output;output1] ; % Watermarked signal

    inv_filt_sig1 = filter(1,a1,sample1); %Inverse filtering
    output1_not_wm = inv_filt_sig1 .* bpsk_sig; % Correlator
    output_not_wm = [output_not_wm;output1_not_wm] ; % Non watermarked Signal
    l = l + 1;
end
mean_output = zeros(1,9);
mean_output_not_wm = zeros(1,9);
for i=1:9
   mean_output(i)= mean(output(i,:))
   mean_output_not_wm(i)= mean(output_not_wm(i,:))
end
```

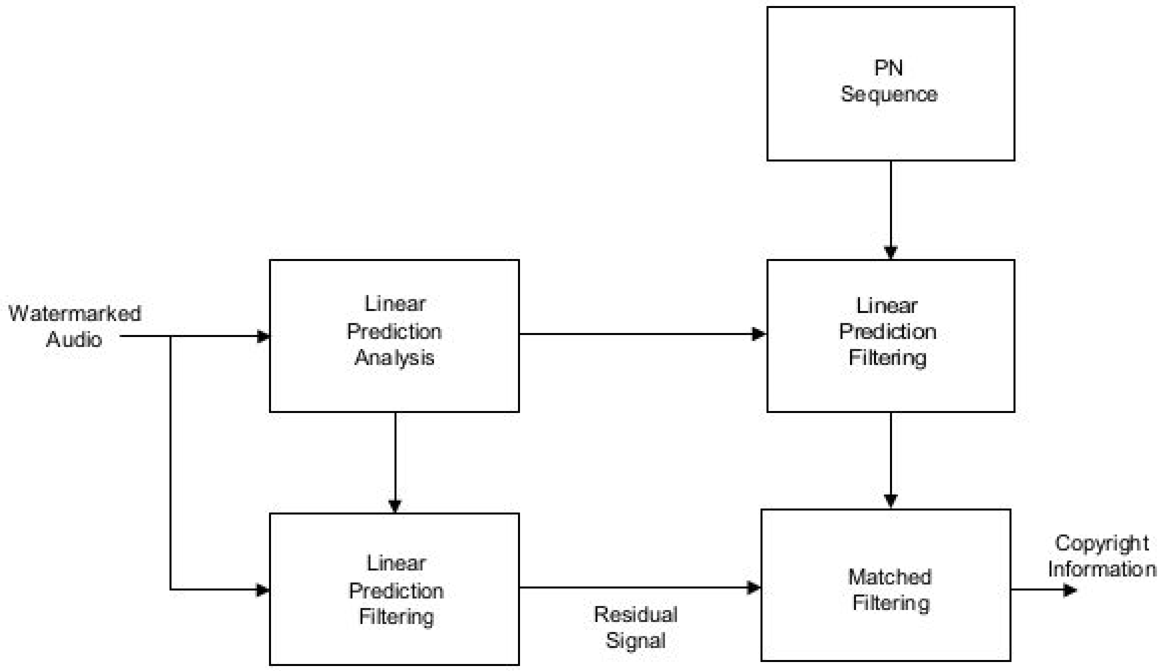

Correlation output for female speech sample
correlation op of watermarked
correlation op of not watermarked
Magnitude
Index
0.1
0.08
0.06
0.04
0.02
0
-0.02
1
2
3
4
5
6
7
8
9



By keeping a threshold, we see that watermarked signal gives an average output of above threshold and hence can be detected using this method.

3.4 Watermark extraction

The watermark can be extracted by using a matched filter.
The box diagram can be described as[4] :

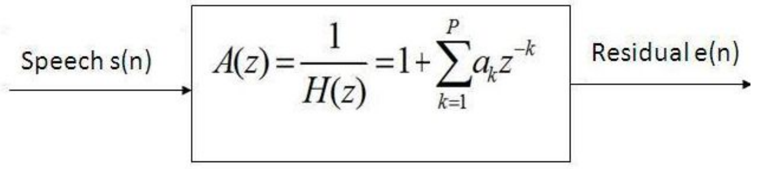


The steps followed for extraction are :

1. The received audio is categorised into small frames, which are used for further processing $s(n) = x(n).w(n)$, where x(n) is the watermarked audio and s(n) is the segment w(n) can be a rectangular window or as we used a hamming window
2. We use Linear Prediction coefficients to model the received watermarked signal , which is a p-order AR model

$$\hat{s}(n) = -\sum_{k=1}^{p} a_k.s(n-k)$$

3. Using these coefficients we designed a Linear prediction filter, where these coefficients are assumed constant over the analysis audio segment.
4. Applying the filter on the audio segment, we get an error term, called as residual signal

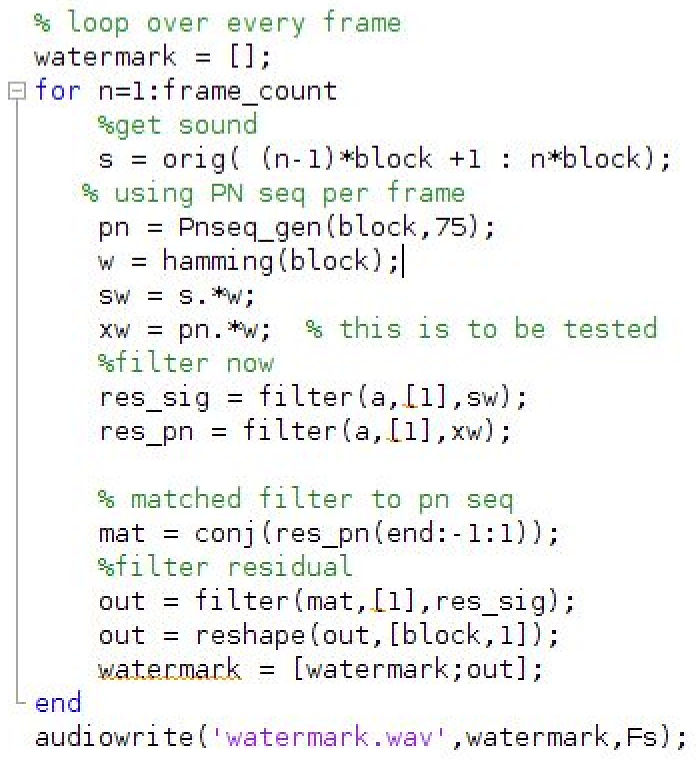


5. Generate a PN sequence as we did during the embedding process, on which we apply same filter to obtain a similar residual signal
6. In this process, the residual signal from the watermarked audio , $e(n) = s(n) - \hat{s}(n)$ contains characteristics of both the original audio and the watermark signal.
7. This method transforms the non-white watermarked audio signal to whitened signal by removing audio spectrum, which can be easily modeled using Gaussian pdf.
8. Using the residual signal and PN sequence as a template , we apply the matched filter to extract the embedded watermark information.

Thus by applying matched filtering on the inversely filtered received signal we get the signal which has a form like the original watermark.

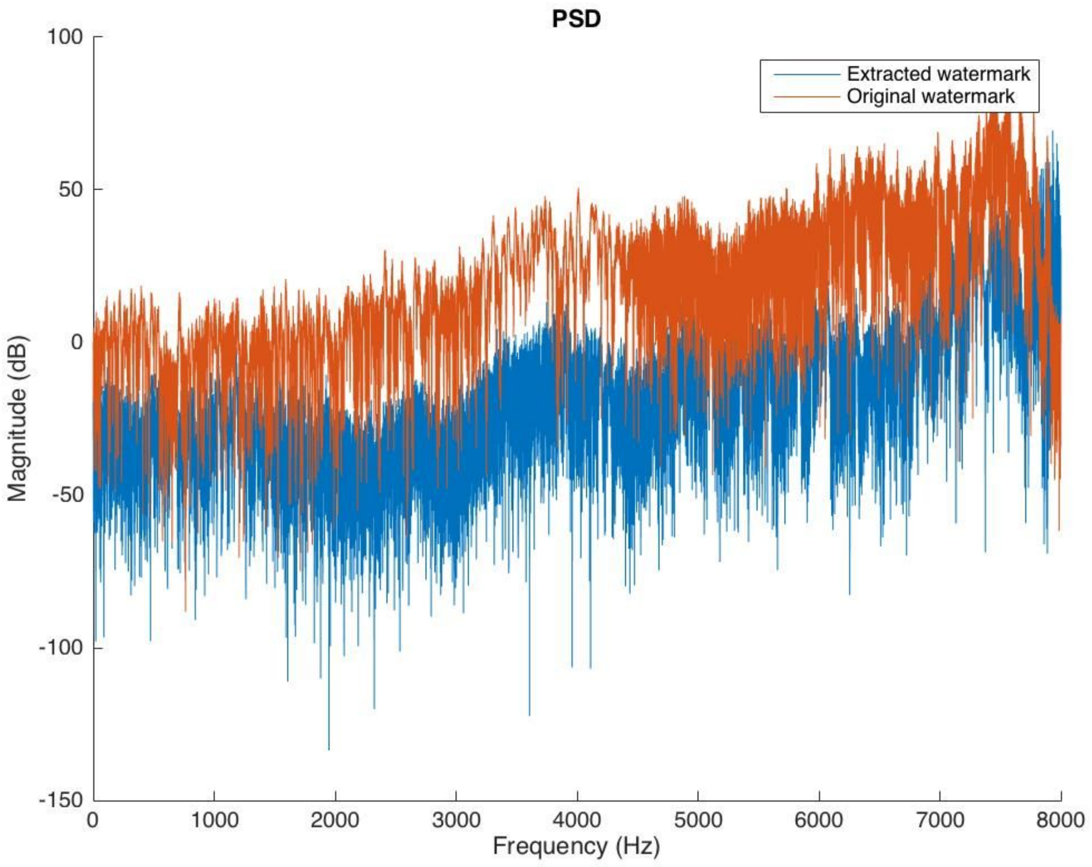
Matlab code for watermark extraction:

```
% watermark extraction
[rx_t,Fs] = audioread('orig.WAV');
order = 10;

%autocor_rx = xcorr(rx_t);
% get lpc coeff
%a = levinson(autocor_rx)';
% or apply LPC on seq direclty
[a,g] = lpc(rx_t,order);

% lpc on whole seq

residual = filter(a,[1],rx_t);
pnseq = Pnseq_gen(length(rx_t),75);
pn_res = filter(a,[1],pnseq);
%matched
matched = conj( pn_res(end:-1:1) );
%filter
water_mark = filter(mat,[1],residual);
audiowrite('watermark_full.wav',water_mark,Fs);

% apply frame wise lpc
% create filter
L = length(rx_t);
block = 1024;

overlap = 0;
frame_count = (L-overlap)/(block-overlap);
%modify sound by paddin
frame_count = ceil(frame_count);
if L<frame_count*block
    rx_t(L+1:frame_count*block) = 0;
end
L = length(rx_t);
```



```
% loop over every frame
watermark = [];
for n=1:frame_count
    %get sound
    s = orig( (n-1)*block +1 : n*block);
   % using PN seq per frame
    pn = Pnseq_gen(block,75);
    w = hamming(block);
    sw = s.*w;
    xw = pn.*w;  % this is to be tested
    %filter now
    res_sig = filter(a,[1],sw);
    res_pn = filter(a,[1],xw);

    % matched filter to pn seq
    mat = conj(res_pn(end:-1:1));
    %filter residual
    out = filter(mat,[1],res_sig);
    out = reshape(out,[block,1]);
    watermark = [watermark;out];
end
audiowrite('watermark.wav',watermark,Fs);
```

PSD of original watermark & extracted watermark:



We see that the extracted watermark resembles the original watermark.

3.5 References:

1) K. Hofbauer, “Speech watermarking and air traffic control,” Ph.D. dissertation, Graz Univ. of Technol., Graz, Austria, Mar. 2009. pp. 48

2) Q. Cheng and J. Sorensen, “Spread spectrum signaling for speech watermarking,”

3) Laurence Boney, Ahmed Tewfik, Khaled Handy, “Digital watermarking of audio signals”

4) Jongwon Seok, Jinwoo Hong, and Jinwoong Kim, “A Novel Audio Watermarking Algorithm for Copyright Protection of Digital Audio” ETRI Journal, Volume 24, Number 3, June 2002 pp 5

5) Mohammad Ali Nematollahi,· S.A.R. Al-Haddad “An overview of digital speech watermarking”

6) Ricardo A. Garcia “Digital Audio Watermarking using a Psychoacoustic Auditory Model and Spread Spectrum Theory”