A Project Report for EE545: Digital Signal Processing-II, Fall-2009

A Comparison between Power Spectral Subtraction Method and Multi-Band Spectral Subtraction Method for Enhancing Speech Corrupted by Colored Noise

By

Sravan Kumar Buggaveeti — **Javier Perez Ramirez**

Srinivasu Rao Pudi — **Yousef Jaradat**

Professor

Dr. Charles Cruesere

12/04/09

Abstract

In this project we investigate the "Multi-Band Spectral Subtraction (MBSS) Method for Enhancing Speech Corrupted by Colored Noise" technique proposed by Sunil D. Kamath and Philipos C. Loizou from the University of Texas at Dallas [1]. This technique is used to filter the colored noise that affects the speech spectrum non-uniformly over different frequencies and to enhance the speech spectrum**.** In this project we will learn to analyze speech spectrum, study the effect of colored noise to speech signal, and implement the MBSS method for enhancing speech corrupted by colored noise and in addition to the above mentioned methods. In order to evaluate the results we will implement the MBSS technique using MATLAB, and compare the results with the authors' results in one side, and on the other side we will implement the other technique namely Power Spectral Subtraction (PSS) and compare the results with the MBSS.

Introduction

Speech conveys useful information to the listeners. The presence of background noise in speech significantly reduces the quality of the speech and distorts the information content in it. Though the speech can be perceptible with the noise corrupted to an extent, the quality of noise creates good interest in the listener. Distortion in the quality of speech brings down the interest to the listener. The speech quality is very much important for the mobile communication technologies, hearing aids, radio broadcasting and many more applications. In the past researchers have come with a variety of theoretical and relatively effective techniques to reduce the noise as much as possible. Those techniques were not effective enough to produce clean speech signal. Various non-linear approaches to the spectral subtraction have been described in literature [2][3][4][5]. Those non-linear approaches apply when the noise spectrum is flat which is not in the case of colored noise. In this project the objective is to filter real-world noise for which the spectrum is not flat from the speech signal. Therefore a modified spectral subtraction called MBSS is used.

Removing various types of noise is difficult due to the random nature of the noise and the inherent complexities of speech. Noise reduction techniques usually have a tradeoff between the amount of noise removal and speech distortions introduced due the processing of the speech signal. Complexity and ease of implementation of the noise reduction algorithms is also of concern in applications especially those related to portable devices such as mobile

communications and digital hearing aids. The randomness in the noise has made the noise removal really difficult for the researchers. They have come up with techniques based on spectral subtraction which is based on estimating the noise present in the speech signal and then subtracting the estimated from the original speech signal. One of the techniques is PSS proposed by Boll [6] and improved further by Berouti [7].

Power Spectral Subtraction (PSS)

The power spectral subtraction as in [6] estimates the magnitude of noise in the period of initial silence or in non-speech activity. That noise is subtracted from the speech spectrum and the clean speech is obtained. In [7], the PSS is improvised using the over-subtraction factor considering the Segmental Sound to Noise Ratio (SNR). The estimate of the clean spectrum is implemented using the following equation:

$$|\hat{S}(k)|^2 = |Y(k)|^2 - \alpha|\widehat{D}(k)|^2 \quad (1)$$

where α is an over-subtraction factor which is a function of the segmental SNR, S(k) is the magnitude spectra of the clean speech, D(k) is the magnitude spectra of the noise and $\widehat{D}(k)$ is an estimate calculated during periods of silence.

Although PSS offers good results, an assumption that noise affects the speech spectrum uniformly is considered. This assumption fails with real world noise where the speech spectrum is not uniformly affected by the noise. Though the PSS is advantageous as it is a simple technique to implement, it almost fails when noise is estimated as negative which is required to be assumed as floored to a specified level [8]. To handle the effect of this non-uniformity and spectral flooring, the technique MBSS as described in [1] was developed.

Multi-Band Spectral Subtraction

MBSS divides the noise spectrum obtained into multiple bands as per the frequency ranges. The spectral subtraction is implemented within each of these bands. In the MBSS, the noise is estimated at initial silence for each band. This will ensure that correct amount of noise is subtracted from the speech signal.

Implementation

Noise suppression is implemented using two methods namely PSS and MBSS and the results are evaluated using the PESQ score and from the statistics of subjective tests. Noise suppression is usually done by subtracting the estimated noise from the speech signal. PSS was proposed in [6] to obtain the clean speech from noise corrupted speech. The estimate of the clean spectrum is implemented using the following equation:

$$|\hat{S}(k)|^2 = |Y(k)|^2 - \alpha|\widehat{D}(k)|^2 \tag{1}$$

where α is an over-subtraction factor which is a function of the segmental SNR, $\hat{S}(k)$ is the magnitude spectra of the clean speech, D(k) is the magnitude spectra of the noise and $\widehat{D}(k)$ is an estimate calculated during periods of silence.

The subtraction of noise using the PSS method falls short because of the inaccurate estimation of the noise spectrum and leads to annoying distortions in the clean speech. The estimation of the noise spectrum was uniform here which is not as in the case of colored noise.

To overcome this problem, the MBSS was proposed as in [1] which takes into account of the non-uniform noise spectrum. Here the corrupted speech signal is divided into multiple bands and the spectral subtraction is carried out as per following equation

$$|\widehat{Si}(k)|^2 = |Y_i(k)|^2 - \alpha_i\,\delta_i|\widehat{D}_i(k)|^2 \qquad b_i \le k \le e_i \tag{2}$$

where b_i and e_i are the beginning and ending frequency bands of the i_{th} frequency band, α is the over-subtraction factor of the i_{th} band and δ_i is a tweaking factor. α_i is a function of the segmental SNR_i of the i_{th} frequency band calculated as:

$$SNR_i(dB) = 10log_{10}\left(\frac{\sum_{k=b_i}^{e_i}|Y_i(k)|^2}{\sum_{k=b_i}^{ei}|\widehat{D}_i(k)|^2}\right) \tag{3}$$

δ_i is individually set for each frequency band to customize the noise removal properties.

$$\alpha_i = \begin{cases} 4.75, & SNR_i < -5 \\ 4 - \frac{3}{20}(\ SNR_i), & -5 \le SNR_i \le 20 \\ 1, & SNR_i > 20 \end{cases} \tag{4}$$

$$\delta_i = \begin{cases} 1, & f_i < 1\ kHz \\ 2.5, & 1\ kHz < f_i \leq \frac{F_s}{2} - 2\ kHz \\ 1.5, & f_i > \frac{F_s}{2} - 2\ kHz \end{cases} \quad (5)$$

The signal is divided into i bands using the linear spacing approach where each band is divided equally.

- **Pre-Processing:**

Initially the corrupted speech signal is pre-processed. It is first hamming windowed using 20 ms window and a 10 ms (50%) overlap between frames. These parameters can be varied based on the requirement. The weighted spectral average is taken for smoothen the variation in the noise as shown in fig.1 and is done using the following equation:

$$\bar{Y}_j(k) = \sum_{l=-M}^{M} W_l Y_{j-l}(k) \quad (6)$$

where j is Frame index, M the number of frames and W the filter weights. After the pre-processing, the spectral subtraction as discussed in (2) is done to obtain the enhanced speech sequence.

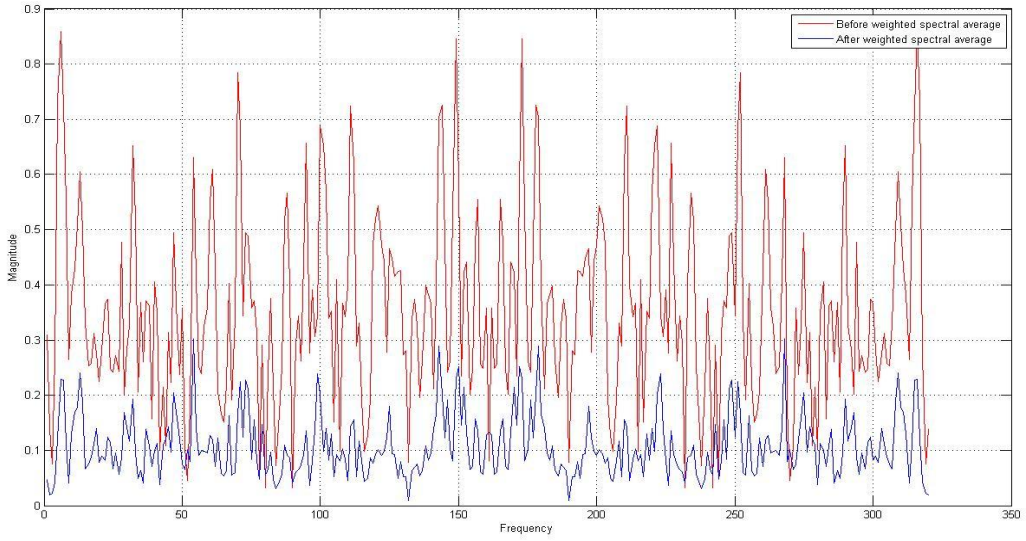


Fig. 1. Plot before and after weighted spectral average

The PSS and MBSS are implemented using MATLAB.

The MATLAB program takes in the corrupted noise signal and asks for the window length, the overlapping percentage and the number of bands for the MBSS method. Then the based upon the length of the audio vector, the hamming window size and the number of frames are obtained. After the multiplication of all the frames with the hamming window, the FFT is taken for each frame which takes us the analysis into frequency domain. The hamming window reduces the artifacts which occur at the edges of the window. FFT will allow us to separate the magnitude from the phase. The methods for noise subtraction are then applied on the magnitudes. The next step is the weighted spectral average implemented using the equation (6). The filter weights, W are chosen as [0.09 0.25 0.32 0.25 0.09] which are empirically determined and set as in [1]. The averaging is limited to 5 successive frames given by l in the equation (6), to prevent the spectral smearing, where $-2 \leq l \leq 2$. The obtained vectors are squared as the implementation works on the squares of magnitudes. Now, as the signal is preprocessed, the obtained spectrum is ready for the spectral subtraction.

- **MBSS:**

Here each of the averaged spectrums is divided into bands according to linear spacing technique where the bands obtained are divided equally. The number of bands can be entered by the user.

The δ_i as in equation (5) is calculated for each band of frequencies.

The magnitude of noise is estimated in the spectrum by assuming that the first frame corresponds to the initial silence of the speech. The threshold of the noise is estimated. Here, for the accurate estimation of the noise, we have averaged the noise when detected with the previously estimated noise. Before averaging the noise, the estimated noise is floored to take into account of the negative values in the speech spectrum which is not done in PSS. The flooring is implemented as per following equation:

$$|\widehat{S}_i(k)|^2 = \begin{cases} |\widehat{S}_i(k)|^2 & |\widehat{S}_i(k)|^2 > \beta|Y_i(k)|^2 \\ \beta|Y_i(k)|^2 & else \end{cases} \quad (7)$$

The over-subtraction factor α_i in equation (2) is determined using the equation (4) where $SNR_i(dB)$ is calculated from equation (3). The value of β is selected by trial and error on listening to the output speech signal. These factors are evaluated on each of the bands i and are used to estimate the subtraction factor to obtain the clean speech spectrum. After determining the parameters required for the estimation of subtraction factor, the MBSS as given in equation (2) is implemented to obtain the clean spectrum.

- **PSS:**

This technique is carried out in the similar manner as MBSS except that here the whole method is implemented with a single band. The pre-processing is done in the same way as described above for MBSS. The value of over-subtraction factor, α_i is calculated as per equation (4) where i corresponds to the frame index in PSS. The noise is estimated during the period of silence and then the spectral subtraction is applied as in equation (1). The spectral flooring is also carried out in the same way as described for MBSS in equation (7).

- **Signal Reconstruction:**

The speech signal obtained from the spectral subtraction is now to be reconstructed to get the complete speech signal and of the same sampling frequency. The speech spectrum in each frame is transformed into time domain using IFFT and then the complete speech signal is reconstructed using the overlap add approach.

Tests and Results

To evaluate the results of MBSS two type of tests namely objective and subjective tests are carried out.

- **Objective tests:**

  The implemented techniques (MBSS and PSS) are compared objectively using Perceptual Evaluation of Speech Quality (PESQ) test methodology. PESQ is a standard quality analysis technique that gives a quality of the speech signal as compared to other signal when two signals are given as input to it. The PESQ gives a quality factor of 4.5 when both the signals are exactly same. In our case, the original signal and the output

signals from PSS and MBSS are given as inputs and the results obtained for each case are compared.

- **PESQ scores for short length speech files (2 sec):**

The PESQ scores for babble noise of SNR 10dB and 5dB is considered and from fig. 2 we though it can be seen that PSS gives a better performance, the scale on y-axis is very small which shows that the scores are very close.

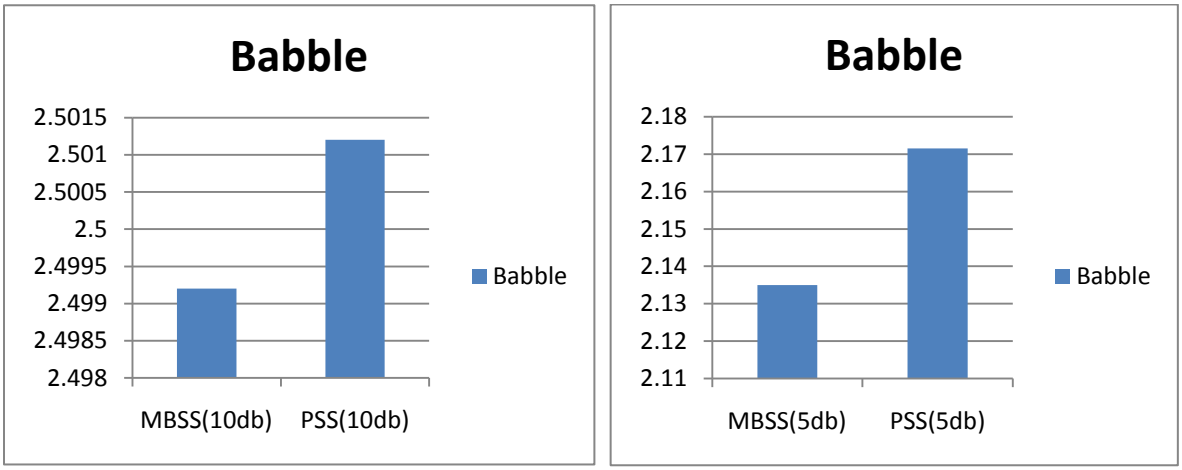


Fig. 2. PESQ scores of babble 10db and 5db for MBSS and PSS.

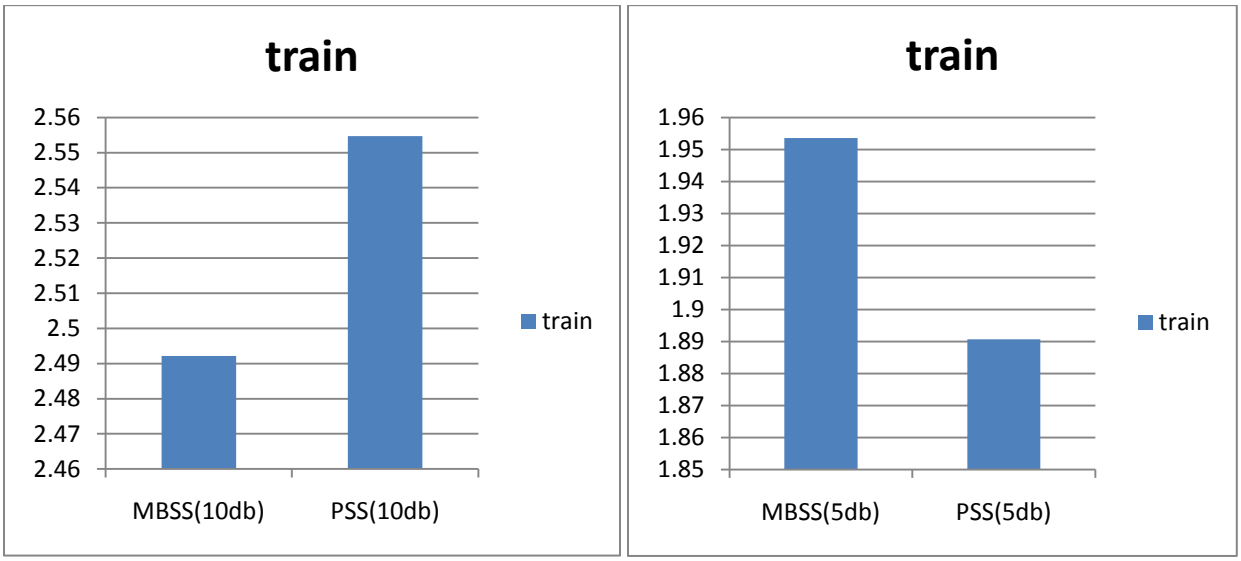


Fig. 3. PESQ scores of train 10db and 5db for MBSS and PSS.

Similar results were obtained for the train noise of 10dB and 5dB as shown in fig. 3.

- **PESQ score for long length speech file (10 sec):**

The speech file considered here is of 10 sec.

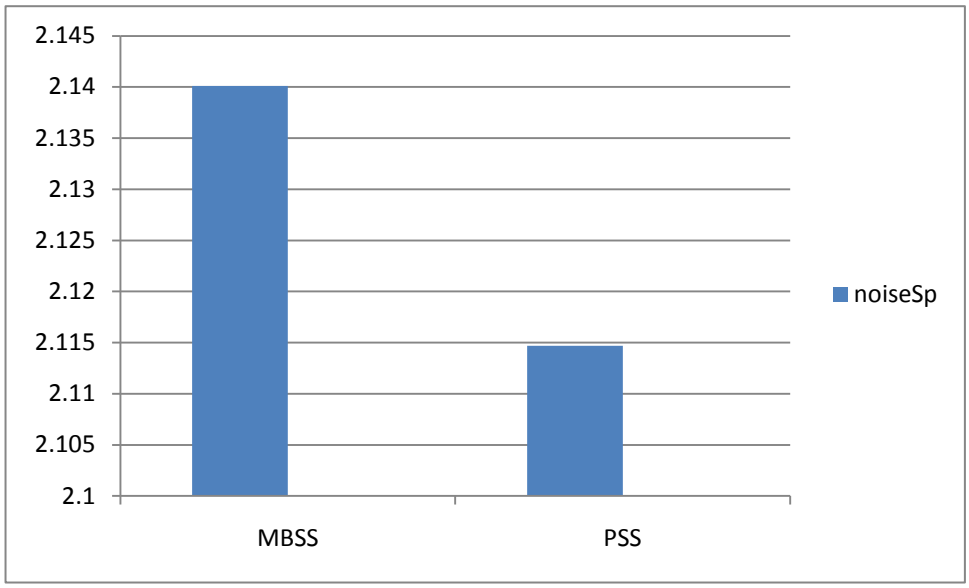


Fig. 4. PESQ score of long length speech file for MBSS and PSS

As in fig. 4, the MBSS gives a much better score than PSS for a long length speech file.

- **Subjective tests:**

  The test has been conducted on two different lengths of speech files - short length speech files and long length speech file. A recommendation from ITU P.835 was followed to perform the subjective tests. According to the recommendation, the speech quality is rated on the scale of 1 to 5 (1 is worst quality and 5 is best quality) from the listeners by having them to listen the speech files. Analysis of Variance (ANOVA) was used to analyze the data gathered. Variance was taken on the ratings of the subjects tested.

The subjective test was conducted on 8 listeners. In the test the instructions were given to the subjects and were trained to correctly perform the tests. They were asked to listen to the three types of test speech files: noisy speech, MBSS enhanced speech, and PSS enhanced speech for each type of speech files (i.e., short and long speech files). For each type of speech, each listener gave three corresponding ratings for the foreground, the background, and the overall ratings, as per the recommendation. The ANOVA was calculated on the ratings for each case. The results are as follows:

- **When test conducted on short length speech file:**

The speech file of duration 2 sec is considered. Babble and train noise speech files are tested. The noise speech files considered were of SNR 5db and 10 db.

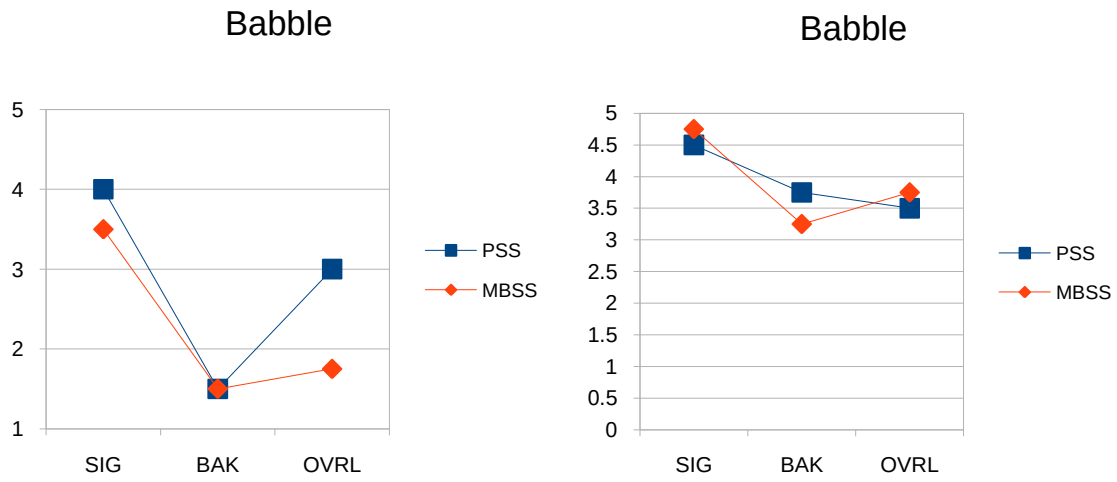


Fig.5. ANOVA of subjects for Babble noise (5dB &10db) as per ITU-P.835

From the fig. 5 the PSS method gets higher rating than MBSS for the 5dB babble noise and MBSS gets higher rating for 10dB babble noise .

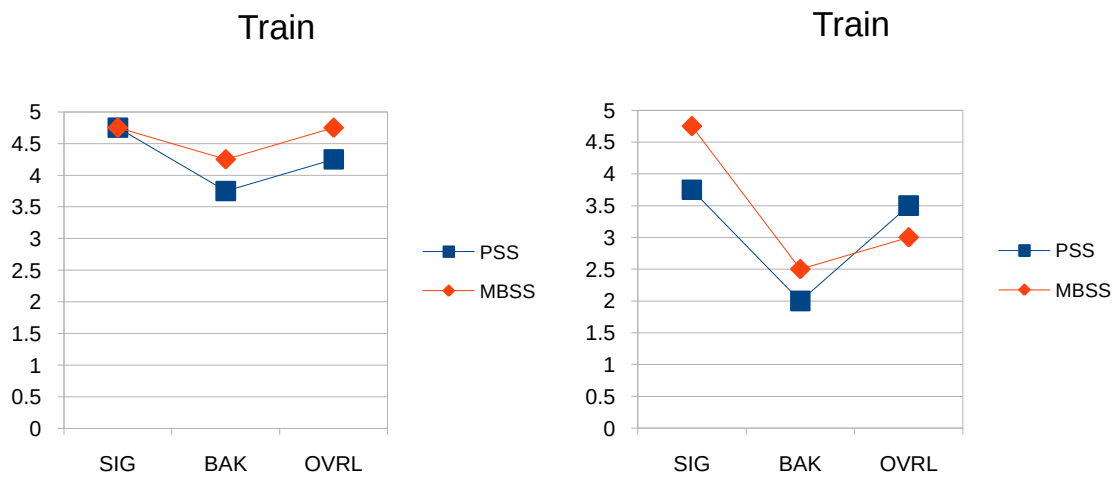


Fig.6. ANOVA of subjects for train noise (5dB &10db) as per ITU-P.835

When tested with the train noise the MBSS performs better than the PSS for both 5dB and 10dB SNR speech files as shown in fig.6. The overall quality for 10dB train noise is higher for PSS

even though the rating for the signal and background noise is higher for MBSS. This is because of less number of subjects considered.

- **When test conducted on long length speech file:**

The speech file of about 10 sec was considered. Fig. 7 indicates that the MBSS gives better performance than PSS.

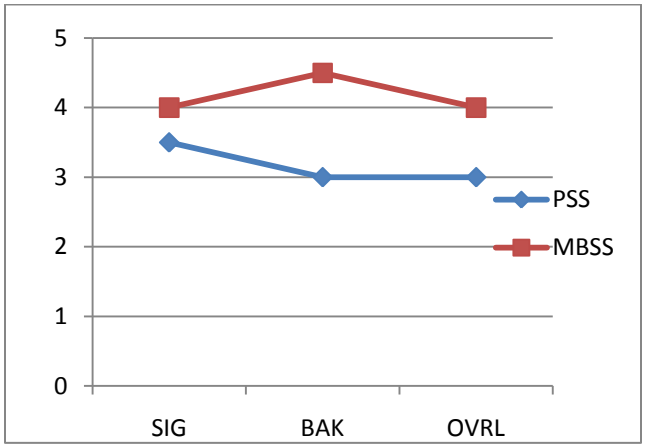


Fig.7. ANOVA of 8 subjects for long length speech file *as per ITU-P.835.*

The results from the speech files of different lengths tested indicates that for a short length speech file the MBSS method is unable to estimate accurately the noise present in the speech. For a long speech file the new noise detected gets averaged with the already estimated noise. As the time goes the MBSS gets better. From the subjective tests we also noticed some musical noise with both the methods. The musical noise is reduced by the MBSS with the duration which does not happen with the PSS method.

Variation of β in equation (7) was analyzed with speech of long duration keeping other parameters constant and the PESQ score was plotted as in fig. 8. It can be observed that MBSS have higher score than PSS for all values of β. The size of the Hamming window was varied and the PESQ score was computed. Fig. 9 shows that for larger Hamming window sizes the PSS gives better result.

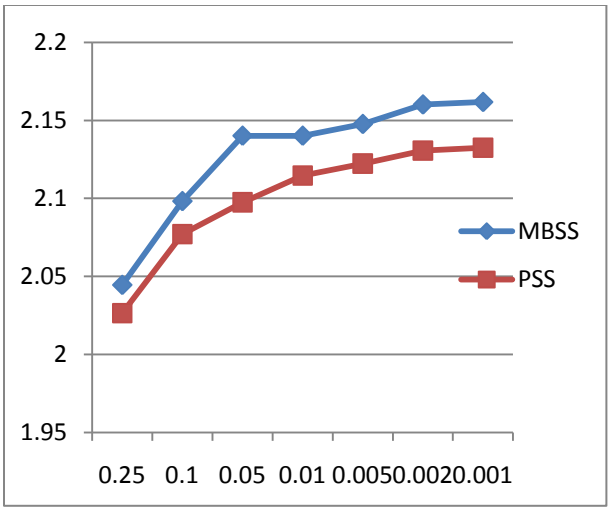


Fig.8. PESQ scores with variation in β for long speech

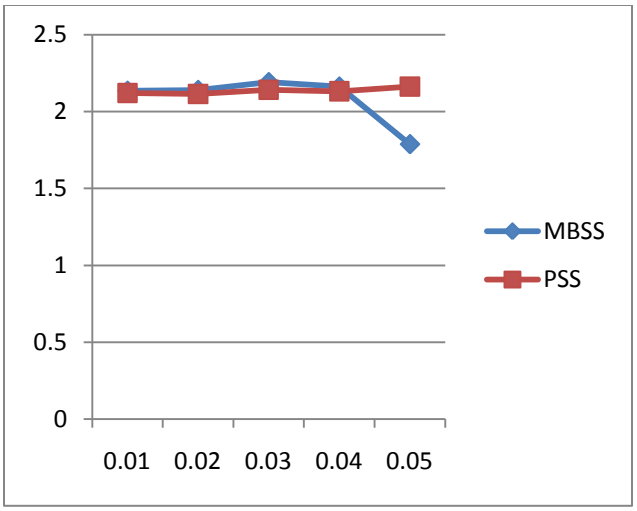
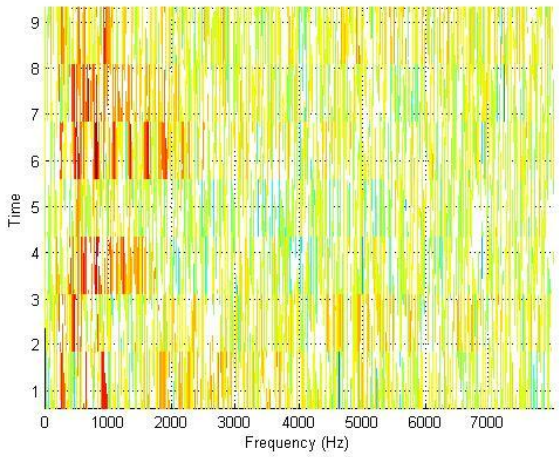


Fig.9. PESQ scores with variation in hamming size.

- **Power Spectograms:**

The spectrograms were plotted for noisy speech, enhanced speech from PSS method and enhanced speech from MBSS method to see the distribution of power in a speech signal of 10 sec. The spectrogram as shown in fig. 10(a) was plotted for the corrupted speech, spectrogram in fig. 10(b) is for PSS enhanced speech and fig. 10(c) spectrogram is for MBSS enhanced speech. The spectrogram for enhanced speech of PSS indicates shows that the power of the speech signal is spread in the whole spectrum. It can be observed that there is more power in the band between 300 Hz and 3 kHz than in the higher frequencies for MBSS. Besides, we can see lack of musical noise in the enhanced speech.



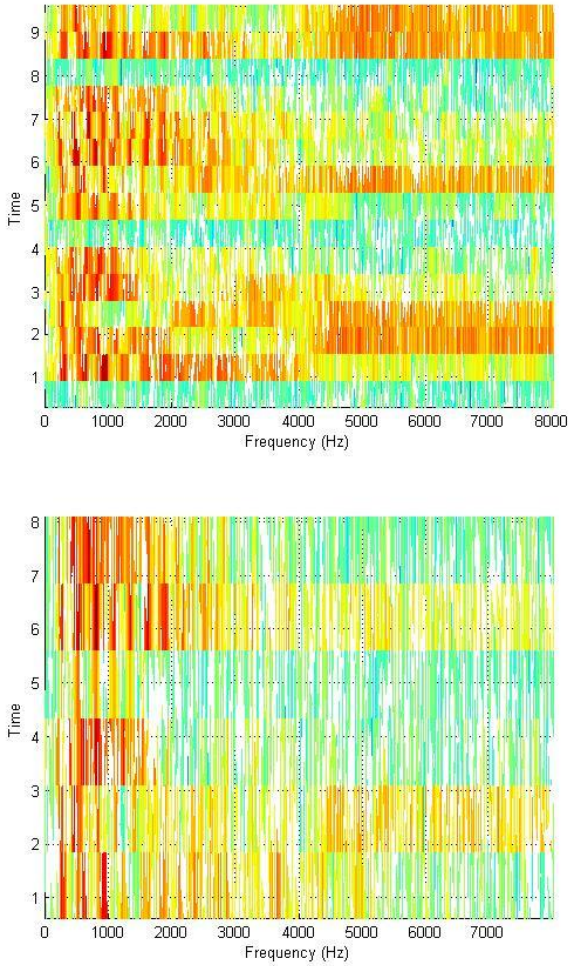


Fig. 10. Power spectrograms from top to bottom : (a)noise speech , (b)enhanced speech from PSS and (c)enhanced speech from MBSS.

Conclusions

- The proposed project on enhancement of speech by multi-band spectral subtraction is compared with the power spectral subtraction method. The speech quality of the enhanced speech signals obtained from both the methods were compared using the objective and subjective tests. The results obtained from objective tests show that both the methods are almost the same for the short speech and MBSS is better for long speech signals. The subjective test results show that for short speech signals MBSS performs better for 10dB SNR whereas PSS performs better for 5dB SNR. MBSS gave higher ratings for longer speech files.

- With the MBSS method we can reduce the residual noise and improve the speech quality.
- The band subtraction factor δ_i gives a more accurate estimation of noise in MBSS.

References

[1] S. D, Kamath, P. C. Loizou, "A multi-band spectral substraction method for enhancing speech corrupted by colored noise" *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process,* v 4, p. IV/4164 ,May 2002.

[2] P. Lockwood and J. Boudy, "Experiments with a nonlinear spectral subtractor (NSS), hidden markov models and the projection, for robust speech recognition in cars," *Speech Communication,* Vol. 11, Nos. 2-3, pp. 215-228, 1992.

[3] I. Soon, S. Koh and C. Yeo, "Selective magnitude subtraction for speech enhancement," *Proceedings. The Fourth International Conference/Exhibition on High Performance Computing in the Asia-Pacific Region*, vol.2, pp. 692-695, 2000.

[4] K. Wu and P. Chen, "Efficient speech enhancement using spectral subtraction for car hands-free application," *International Conference on Consumer Electronics*, vol. 2, pp. 220 221, 2001.

[5] C. He and G. Zweig, "Adaptive two-band spectral subtraction with multi-window spectral estimation," *ICASSP*, vol.2, pp. 793-796, 1999.

[6] S.F. Boll, "Suppression of acoustic noise in speech using spectral subtraction," *IEEE Trans. Acoust., Speech, Signal Process*, vol.27, pp. 113-120, Apr. 1979.

[7] M. Berouti, R. Schwartz and J. Makhoul, "Enhancement of speech corrupted by acoustic noise," *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process,* pp. 208-211, Apr. 1979.

[8] D. V. Compernolle and K. V. Leuvan, Available:
http://cslu.cse.ogi.edu/HLTsurvey/ch10node5.html

User's guide

For MBSS:

1. The main program is named as **mbss.m** in MBSS folder.

2. The functions included are weightedSpectralAVG.m, deltaFunction.m, thresholdLevel.m, speechDetection.m, floorEnhaced.m, alphaFunction.m and overlapAddFunction.m
3. Run the program and two input parameters are asked.
   Input the corrupted speech file name with path within ‘’ and then enter the percentage overlap for windowing process. The results in this report are simulated by taking 50% of overlap.
4. The other parameters beta and threshold values are set to 0.01 and 10 respectively in the code. These can be varied as required.
5. The original corrupted sound is played first and then the enhanced speech file is played.
6. The output file is saved as testMBSS.wav in the same folder.

For PSS:

1. The main program is named as **pss.m** in PSS folder.
2. The functions included are weightedSpectralAVG.m, thresholdLevel.m, speechDetection.m, floorEnhaced.m, alphaFunction.m and overlapAddFunction.m
3. Run the program and two input parameters are asked.
   Input the corrupted speech file name with path within ‘’ and then enter the percentage overlap for windowing process. The results in this report are simulated by taking 50% of overlap.
4. The other parameters beta and threshold values are set to 0.01 and 10 respectively in the code. These can be varied as required.
5. The original corrupted sound is played first and then the enhanced speech file is played.
6. The output file is saved as testPSS.wav in the same folder.

Group Contribution and predefined tools used

We have used MATLAB to program the PSS and MBSS methods.

Predefined commands used in MATLAB are: Hamming from signal processing toolbox, FFT, IFFT, wavread, floor and repmat are from basic MATLAB toolbox.

The code for PESQ score and the recommendation for subjective tests ITU-P.835 were downloaded from the web.

Appendix

```
% filename: mbss.m

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%
%%%  CODE FOR MULTI BAND SPECTRAL SUBTRACTION METHOD
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%

close all;
clear all;
clc;

% % Input Data %%%%%%%%%%

% audioName = input('Introduce the audio file name (between ''): ');
% [audioVector, samplingFreq] = wavread(audioName);
% overlapPercentage = input('Introduce the overlap percentage: ')/100;
% hammingSize = input('Introduce the Hamming Window Size (in seconds): ');
% threshold = input('Introduce threshold (on dB): ');
% noiseAverage = input('Introduce the number of noise frames considered: ');
% beta = input('Introduce beta parameter (0 ~ 0.01): ');
% nBands = input('Introduce the number of bands (linearly spaced): ');
% sound(audioVector,samplingFreq)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Here we read in the original test speech signals.
audioName = 'noisecut';  %%
[audioVector, samplingFreq] = wavread(audioName);

%%%%%%%Input parameters.
overlapPercentage = 50/100;
hammingSize = 0.02;
noiseAverage = 10;
beta = 0.01;
nBands = 8;

  sound(audioVector,samplingFreq);  %%Plays Original input Sound
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Generate Hamming Window for the required number of samples calculated as
%sampling frequency * hamming size.
hammingSize = floor(samplingFreq*hammingSize);
hammVector = hamming(hammingSize);
%%%%%%%%%%%%%%%%%

% Segmentation of the audio signal + Hamming Window %%%%
```

```
sizeAudio = length(audioVector);
overlappingNumber = floor(overlapPercentage*hammingSize);
% Calculation of number of sample vectors
numberOfSegments = floor((sizeAudio-hammingSize)/overlappingNumber) + 1;

matrixIndex = repmat((1:hammingSize)',1,numberOfSegments);
matrixIndex1 = repmat((0:overlappingNumber:(numberOfSegments-1)*overlappingNumber),hammingSize,1);
matrixIndex = matrixIndex + matrixIndex1;
hammingMatrix = repmat(hammVector,1,numberOfSegments);
segmMatrix = audioVector(matrixIndex).*hammingMatrix;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
figure
plot(segmMatrix(:,100))

% FFT %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
fftSegmMatrix = fft(segmMatrix,hammingSize);
fftSegmMatrixPhase = angle(fftSegmMatrix);
fftMSize = size(fftSegmMatrix);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

figure
plot(abs(fftSegmMatrix(:,40)),'R')
hold on

% Delta Vector Calculation %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[deltaVector] = deltaFunction(hammingSize,samplingFreq,nBands);
% deltaVector = ones(1,fftMSize(1));
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Weighted Spectral Average %%%%%%%%%%%%%%%%%%%%%%%%%%%
[avgMatrixSquared] = weightedSpectralAVG(fftSegmMatrix);
outputAvgMatrixSquared = avgMatrixSquared;
plot(outputAvgMatrixSquared(:,40))
grid on
[threshold] = thresholdLevel(avgMatrixSquared)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


% Enhancement %%%%%%%%%%%%%%%%%%%%%%%%%%%%
noiseSquaredVector = avgMatrixSquared(:,1)';
counter = 0;
for n=2:fftMSize(2)
    [speechFlag,snrValue,newNoiseSquaredVector] =
speechDetection(avgMatrixSquared(:,n),noiseSquaredVector,threshold);
    if speechFlag == 1
        [alphaValue] = alphaFunction(snrValue);
        outputAvgMatrixSquared(:,n) =  avgMatrixSquared(:,n) -
0.95.*(alphaValue.*deltaVector.*noiseSquaredVector)';
        outputAvgMatrixSquared(:,n) =
floorEnhaced(outputAvgMatrixSquared(:,n),noiseSquaredVector,beta);
    else
        [alphaValue] = alphaFunction(snrValue);
        noiseSquaredVector = ((noiseAverage-
1)/noiseAverage).*noiseSquaredVector +
(1/noiseAverage).*newNoiseSquaredVector;
```

```
        outputAvgMatrixSquared(:,n) =  avgMatrixSquared(:,n) -
0.95.*(alphaValue.*deltaVector.*noiseSquaredVector)';
        outputAvgMatrixSquared(:,n) =
floorEnhaced(outputAvgMatrixSquared(:,n),noiseSquaredVector,beta);
        counter = counter + 1
%         plot(noiseSquaredVector);
%         hold on;
    end
end
%%%

% Signal Reconstrunction %
[recoveredSpeechSignal] =
overlapAddFunction(sqrt(outputAvgMatrixSquared),hammingSize,overlappingNumber
,fftSegmMatrixPhase);
 wavwrite(recoveredSpeechSignal,samplingFreq,'testMBSS.wav');
sound(recoveredSpeechSignal,samplingFreq)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Smoothed Original Spectrum &  Smoothed Enhanced Spectrum %
sizeBands = floor(sizeAudio/nBands)
%  plot(abs(fftSegmMatrix(:,floor(fftMSize(2)/2))),'-r');
% hold on;
% plot(sqrt(outputAvgMatrixSquared(:,floor(fftMSize(2)/2))));
 figure;
 spectrogram(audioVector,sizeBands,0,sizeBands,samplingFreq);
 figure;
 spectrogram(recoveredSpeechSignal,sizeBands,0,sizeBands,samplingFreq);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% filename:weightedSpectralAVG.m
%%%%%%%%%%%Function to implement Weighted spectral average for 5 frames.

function [avgMatrixSquared] = weightedSpectralAVG(fftSegmMatrix)

% avgVector = [0.09 0.25 0.32 0.25 0.09];
% Weights determined empirically
sizeFftSegmMatrix = size(fftSegmMatrix);
absFFTSegmMAtrix = abs(fftSegmMatrix);
avgMatrix = absFFTSegmMAtrix;

% spectral averaging
for n=1:sizeFftSegmMatrix(2)-4
   avgMatrix(:,n+2) = 0.09*absFFTSegmMAtrix(:,n) + 0.25*absFFTSegmMAtrix(:,n+1) +
0.32*absFFTSegmMAtrix(:,n+2) + 0.25*absFFTSegmMAtrix(:,n+3) + 0.09*absFFTSegmMAtrix(:,n+4);
end

avgMatrixSquared = avgMatrix.^2;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%filename: deltaFunction.m
% To determine the band subtraction factor beta%
function [deltaVector] = deltaFunction(fftVectorSize,samplingFreq,nBands)

upperBandFrequenciesVector = zeros(1,nBands);
```

```
upperBandFrequenciesVector(1) = samplingFreq*(1/nBands);
for n=2:nBands
   upperBandFrequenciesVector(1,n) = samplingFreq*(n/nBands);
end
sFreq = samplingFreq;
nSamplesBand = floor(fftVectorSize/(2*nBands));
deltaTemp1Vector = zeros(1,nSamplesBand*nBands);
deltaVector = 1.5.*ones(1,fftVectorSize);

deltaTempVector = upperBandFrequenciesVector<=1000;
deltaTempVector = deltaTempVector + 2.5.*((1000 <
upperBandFrequenciesVector).*(upperBandFrequenciesVector <= sFreq/2-2000));
deltaTempVector = deltaTempVector + 1.5*((sFreq/2-2000<upperBandFrequenciesVector));

for n=1:nBands
   for m = 1:nSamplesBand
      deltaTemp1Vector((n-1)*nSamplesBand+m) = deltaTempVector(n);
   end
end

deltaVector(1:nBands*nSamplesBand) = deltaTemp1Vector;
deltaVector(end-(nBands*nSamplesBand-1):end) = flipdim(deltaTemp1Vector,2);

%filename: thresholdLevel.m
%%% To determine the threshold level of noise

function [threshold] = thresholdLevel(avgMatrixSquared)

sizeFrame = size(avgMatrixSquared);
powerNoise = sum(avgMatrixSquared(:,1))/sizeFrame(1);
powerSpeechSignal = sum(sum(avgMatrixSquared(:,2:11)))/(10*sizeFrame(1));
threshold = 10*log10(powerSpeechSignal) - 10*log10(powerNoise);  % gives the estimation of noise present in the
speech signal

%%% SpeechDetection.m%%%
%%% to detect wheather most of the vector contains speech or noise and
%%% returns a flag of 1 for speech and 0 for noise.
function [speechFlag,snrValue,newNoiseSquaredVector] =
speechDetection(avgVectorSquared,noiseSquaredVector,threshold)

snrValue = 10*log10(sum(avgVectorSquared)) - 10*log10(sum(noiseSquaredVector));  %%% Calculation of SNR
Value..Speech to Noise Ratio.
newNoiseSquaredVector = noiseSquaredVector;
speechFlag = 0;

if snrValue > threshold
   speechFlag = 1;
else
   newNoiseSquaredVector = avgVectorSquared';
end

%floorEnhaced.m
%%%To floor the samples if negative values of it are found in the signal.
function [newFFTEnhacedFrame] = floorEnhaced(FFTEnhacedFrame,noiseVector,beta)
```

```
substractVector = FFTEnhacedFrame - noiseVector'.*beta;
negativeVector = (substractVector < 0);    %%% check if negative
indexVector = find(negativeVector);
newFFTEnhacedFrame = FFTEnhacedFrame;
newFFTEnhacedFrame(indexVector) = noiseVector(indexVector).*beta;   %%% update the floored values with the
set Beta.

%alphaFunction.m
% to determine over subtraction factor alpha%%%%%%%%%%%%%%%%%%%%
function [alphaValue] = alphaFunction(snrInput)

snrCoef = snrInput;               % takes snr value as an input to caluculate over subtraction factor%
if snrInput<(-5)
   snrCoef = (-5);
else
   if snrInput>20
      snrCoef = 20;
   end
end

alphaValue = 4-snrCoef*(3/20);

%overlapAddFunction.m
%%%%To implement Overlap Add Function for signal Reconstruction %%%%%%

function [recoveredSpeechSignal] =
overlapAddFunction(FFTMatrix,windowSize,overlappingNumber,phaseMatrix)

% Original Phase addition (Enhanced FFT Matrix) %
newFFTMatrix = FFTMatrix.*exp(j*phaseMatrix);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Output initialization
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
sizeNewFFTMatrix = size(newFFTMatrix);
nFrames = sizeNewFFTMatrix(2);
recoveredSpeechSignal = zeros(1,(nFrames-1)*overlappingNumber + windowSize);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Overlap Add Method %
for n=1:nFrames
   index = (n-1)*overlappingNumber + 1;
   recoveredSpeechSignal(index:index + windowSize -1) = recoveredSpeechSignal(index:index + windowSize -1) +
real(ifft(newFFTMatrix(:,n),windowSize))';
end
%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%
%%%  CODE FOR Power SPECTRAL SUBTRACTION METHOD
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%
```

```
close all;
clear all;
clc;

% % Input Data
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 audioName = input('Introduce the audio file name (between "): ');
% [audioVector, samplingFreq] = wavread(audioName);
 overlapPercentage = input('Introduce the overlap percentage: ')/100;
% hammingSize = input('Introduce the Hamming Window Size (in seconds): ');
% threshold = input('Introduce threshold (on dB): ');
% noiseAverage = input('Introduce the number of noise frames considered: ');
% beta = input('Introduce beta parameter (0 ~ 0.01): ');
% sound(audioVector,samplingFreq)
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Input Data
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% audioName = 'noisyy_2.wav';
% audioName = 'noisecut.wav';
[audioVector, samplingFreq] = wavread(audioName);
% overlapPercentage = 40/100;
hammingSize = 0.01;
threshold = 0;
noiseAverage = 10;
beta = 0.002;
nBands = 16;
% sound(audioVector,samplingFreq)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Plot the used Hamming Window %%%%%%%%%%%%%%
hammingSize = floor(samplingFreq*hammingSize);
hammVector = hamming(hammingSize);
%%%%%%%%%%%%%%%%

% Segmentation of the audio signal + Hamming Window %%%%
sizeAudio = length(audioVector);
overlappingNumber = floor(overlapPercentage*hammingSize);
numberOfSegments = floor((sizeAudio-hammingSize)/overlappingNumber) + 1;
matrixIndex = repmat((1:hammingSize)',1,numberOfSegments);
matrixIndex1 = repmat((0:overlappingNumber:(numberOfSegments-1)*overlappingNumber),hammingSize,1);
matrixIndex = matrixIndex + matrixIndex1;
hammingMatrix = repmat(hammVector,1,numberOfSegments);
segmMatrix = audioVector(matrixIndex).*hammingMatrix;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% FFT %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
fftSegmMatrix = fft(segmMatrix,hammingSize);
fftSegmMatrixPhase = angle(fftSegmMatrix);
fftMSize = size(fftSegmMatrix);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Weighted Spectral Average %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[avgMatrixSquared] = weightedSpectralAVG(fftSegmMatrix);
```

```
% avgMatrixSquared = abs(fftSegmMatrix).^2;
outputAvgMatrixSquared = avgMatrixSquared;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Enhacement %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
noiseSquaredVector = avgMatrixSquared(:,1)';
counter = 0;
for n=2:fftMSize(2)
   [speechFlag,snrValue,newNoiseSquaredVector] =
speechDetection(avgMatrixSquared(:,n),noiseSquaredVector,threshold);
   if speechFlag == 1
      [alphaValue] = alphaFunction(snrValue);
      outputAvgMatrixSquared(:,n) =  avgMatrixSquared(:,n) - 0.95.*(alphaValue.*noiseSquaredVector)';
      outputAvgMatrixSquared(:,n) = floorEnhaced(outputAvgMatrixSquared(:,n),noiseSquaredVector,beta);
   else
      [alphaValue] = alphaFunction(snrValue);
      noiseSquaredVector = ((noiseAverage-1)/noiseAverage).*noiseSquaredVector +
(1/noiseAverage).*newNoiseSquaredVector;
      outputAvgMatrixSquared(:,n) =  avgMatrixSquared(:,n) - 0.95.*(alphaValue.*noiseSquaredVector)';
      outputAvgMatrixSquared(:,n) = floorEnhaced(outputAvgMatrixSquared(:,n),noiseSquaredVector,beta);
      counter = counter + 1
      plot(noiseSquaredVector);
      hold on;
   end
end
%%%

% Signal Reconstrunction %
[recoveredSpeechSignal] =
overlapAddFunction(sqrt(outputAvgMatrixSquared),hammingSize,overlappingNumber,fftSegmMatrixPhase);
  sound(recoveredSpeechSignal,samplingFreq)
wavwrite(recoveredSpeechSignal,samplingFreq,'testSNS.wav');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%

% Smoothed Original Spectrum &  Smoothed Enhaced Spectrum %
sizeBands = floor(sizeAudio/nBands)
plot(abs(fftSegmMatrix(:,floor(fftMSize(2)/2))),'-r');
hold on;
plot(sqrt(outputAvgMatrixSquared(:,floor(fftMSize(2)/2))));
figure;
spectrogram(audioVector,sizeBands,0,sizeBands,samplingFreq);
figure;
spectrogram(recoveredSpeechSignal,sizeBands,0,sizeBands,samplingFreq);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%
```