Palestine Polytechnic University

Department of Electronics and Communication Engineering

De-noising of Speech Signal Using Wavelet

Project Submitted in Partial Fulfillment
of the Requirements for the Degree of

Bachelor Degree
In
Electronics and Communication Engineering

by

Bara' Al frookh

Islam Sharawneh

Mohammad Abu raida

Supervisor :

Dr. Ghandi Manasra

Hebron - Palestine

May 2015

Palestine Polytechnic University

Collage of Engineering

Electrical Engineering Department

Hebron – Palestine

De-noising of Speech Signal Using Wavelet

Project Team

Bara' Al frookh

Islam Sharawneh

Mohammad Abu raida

Submitted to the Collage of Engineering

In partial fulfillment of the requirements for the degree of

Bachelor degree in Electronics and Communication Engineering.

Supervisor Signature

……………………………………….

Testing Committee Signature

………………………………………………………………….…..

Chair of the Department Signature

……………………………………….

June 2015

الاهداء

الحمد الذي علم بالقلم، علم الإنسان ما لم يعلم، والحمد لله الذي هدانا بهدايته ووفقنا بتوفيقه ، اللهم اجعلنا ممن قلت فيهم "وَهُدُوا إِلَى الطَّيِّبِ مِنْ الْقَوْلِ وَهُدُوا إِلَى صِرَاطِ الْحَمِيدِ" .

نقدم هذا العمل المتواضع لوجه الله تعالى راجين منه يضعه في ميزان حسناتنا، وأن يجعل فيه البركة والفائدة لكل قارئ له.

نهدي هذا العمل الى أهلنا وأصدقائنا .

شكر

لم يكن هذا العمل لينجز لولا جهود كثير من الاشخاص، ومن هنا **نتقدم بالشكر الى الدكتور** غاندي مناصرة والى الهيئة التدريسية في **دائرة** الهندسة الكهربائية في جامعة بوليتكنك فلسطين.

Abstract

In this project the wavelet de-noising method is used to remove the additive white Gaussian noise from noisy speech signals. The idea of wavelet de-noising is to remove the noise by discarding small coefficients of the discrete wavelet transform for the noisy speech signal. These coefficients can be removed by applying some kind of thresholding function which removes any coefficient below a specific threshold value and keep any coefficient above it. Then, the signal reconstructed by applying inverse discrete wavelet transform. To evaluate the performance of such algorithm, some kind of performance measure such as signal to noise ratio (SNR) can be applied.

Several methods for speech de-noising using wavelets were tested to evaluate their performance. Universal thresholding method is used to threshold the wavelet coefficients. This method uses a fixed threshold for all coefficients, and the threshold selection depends on the statistical variance measurement. Interval dependent thresholding method is also tested to find its performance, here the signal is divided into different interval depends on variance change in it. Then, the threshold value is calculated for each subinterval depends on the noise variance of each interval. Setting all details coefficients in the first scale to zero by assuming that most of the noise power in the first level is tested to evaluate the performance such assumption.

Different comparisons are tested such as comparing the performance with different threshold selection rules, comparing the performance with different wavelet families, comparing with other filtering technique. The wiener filtering is compared with wavelet de-noising method.

Content

List of Figures

Chapter 1
Introduction and Motivation

Introduction and Motivation

1.1Introduction

Removing of the noise from signals is a key problem in a Digital Signal Processing field (DSP).

In the mid – 1960s, Dolby noise reduction system was developed for use in analog magnetic tape recording. Until the beginning of the 1990s, microelectronic and low cost computer with computation and algorithm design allowed a fast and vast expansion in the field of digital signal processing researches.

One of the most fundamental problem in the field of speech processing is how the noise can be removed from the noisy speech signals.

Speech de-noising is the field of studying methods used to recover an original speech signal from noisy signals corrupted by different types of noise (e.g. white noise, band-limited white noise, narrow band noise, coloured noise, impulsive noise, transient noise pulses).These methods can be used in many computers based speech and speaker recognition, coding and mobile communications, hearing aid. More reduction in noise increases the quality of such application.

The field of speech de-noising includes a lot of researches to improve the speeches overall quality and increase the speech intelligibility. There are different techniques for de-noising the speech signal. Generally speaking the approaches can be classified into two major categories of single microphone and multi microphone methods [1].

1.2 Related works

A lot of algorithms proposed to tackle the problem of noise in speech signals, such as Spectral Subtraction [2], Wieiner Filtering [3], Ephraim Malah filtering [4], hidden Markov modeling [5], signal subspace [6].

Gabor [7] introduced a new time – frequency signal analysis. In the field of mathematic, the papers of mathematicians Mallat [8,9] and Daubechies [10] are a big contribution not only in a mathematical side , but also in an engineering applications. These contributions build what so called "multi-rate filter banks basing on wavelet transform".

Mallat and Hwang [11] introduced an algorithm to remove white noises based on singularity information analysis, Donoho [12] introduced a non linear wavelet methods, Donoho and Johnstone proposed a well known universal wavelet thresholding to remove White Gaussian Noise (WGN) [Donoho12,13] ,[Donoho and johnstone 14], Johnstone and Silverman [15] proposed level dependant thresholding enhancement method.

1.3 Speech Production

In order to apply DSP techniques to speech processing problems, it is important to understand the fundamentals of the speech production process, [16].

Speech is the acoustic product of voluntary and well-controlled movement of a vocal mechanism of a human (see fig.1.1). During the generation of speech, air is inhaled into the human lungs by expanding the rib cage and drawing it in via the nasal cavity, velum and trachea it is then expelled back into the air by contracting the rib cage and increasing the lung pressure. During the expulsion of air, the air travels from the lungs and passes through vocal cords which are the two symmetric pieces of ligaments and muscles located in the larynx on the trachea. Speech is produced by the vibration of the vocal cords. Before the expulsion of air, the larynx is initially closed. When the pressure produced by the expelled air is sufficient, the vocal cords are pushed apart, allowing air to pass through. The vocal cords close upon the decrease in air flow. This relaxation cycle is repeated with generation frequencies in the range of 80Hz – 300Hz. The generation of this frequency depends on the speaker‘s age, sex, stress and emotions. This succession of the glottis openings and closure generates quasi-periodic pulses of air after the vocal cords. The speech signal is a time varying signal whose signal characteristics represent the different speech sounds produced. There are three ways of labelling events in speech. First is the silence state in which no speech is produced. Second state is the unvoiced state in which the vocal cords are not vibrating, thus the output speech waveform is a periodic and random in nature. The last state is the voiced state in which the vocal cords are vibrating periodically when air is expelled from the lungs. This results in the output speech being quasi-periodic- shows a speech waveform with unvoiced and voiced state. Speech is produced as a sequence of sounds. The type of sound produced depends on shape of the vocal tract. The vocal tract starts from the opening of the vocal cords to the end of the lips. Its cross sectional area depends on the position of the tongue, lips, jaw and velum. Therefore the tongue, lips, jaw and velum play an important part in the production of speech.[17]

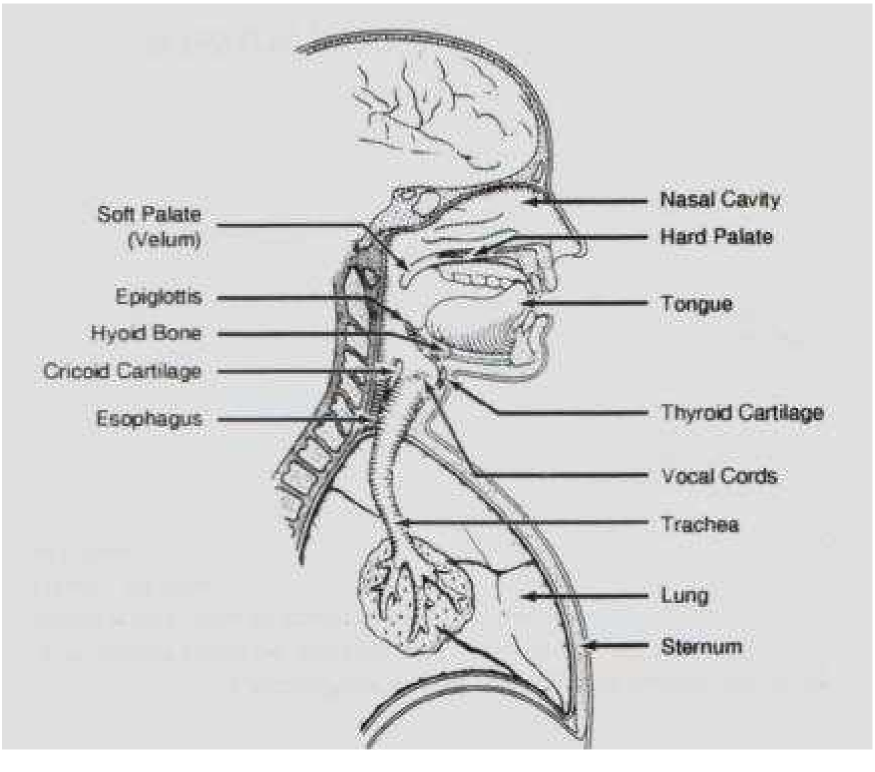


Fig.1.1:Speech -acoustic product of voluntary and well controlled movement of a vocal mechanism of a human

Audible sounds are transmitted to the human ears through the vibration of the particles in the air. Human ears consist of three parts, the outer ear, the middle ear and the inner ear. The function of the outer ear is to direct speech pressure variations toward the eardrum where the middle ear converts the pressure variations into mechanical motion. The mechanical motion is then transmitted to the inner ear, which transforms these motion into electrical potentials that passes through the auditory nerve, cortex and then to the brain . Figure (fig.1.2) below shows the schematic diagram of the human ear.[17]

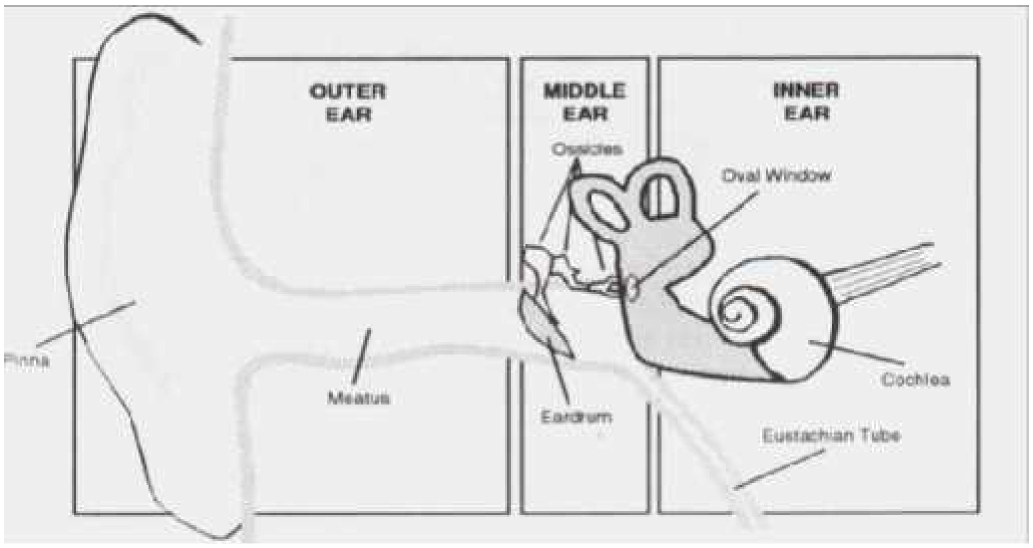


Fig.1.2: The schematic diagram of the human ear

1.4 Motivation

Speech is a native way for human communication and it considered one of the most important signals in multimedia system. Noise is presented in a speech signal due to communication channel. Removing the noise to improve the quality of speech is needed. One of the most important kind of noise is the white noise which is random and its power spectral density is constant. Specifically, Gaussian noise is normally distributed and generated by almost all natural phenomena.

Speech signal is a non-stationary signal. The wavelet transform is considered as appropriate choice to analyze local variations in signals. The multi-resolution properties of wavelet analysis reflect the frequency resolution of the human ear system. Most of data that represent the speech signal are not totally random, there is a certain correlation structure. The harmonic signals content is closely correlated, and this means that large coefficients represent the speech signal and the small values represent the uncorrelated noise. Thus, the noise can be removed by discarding the small coefficients.

1.5 Project Outline

The structure of this project is as follows, in chapter 2 some of background about wavelets, filter banks and multi-resolution theory. Wavelet de-noising model and algorithm design are presented in chapter 3. The speech quality evaluation and performance of algorithm are presented in chapter 4,conclusion is shown in chapter 5.

Chapter 2

Wavelet transform and multiresolution analysis

Wavelet transform and multiresolution analysis

In this chapter, we will briefly introduce the background behind the wavelet transform and multiresoltion analysis. This introduction will be as short as possible. There are several papers and articles talking about wavelets. For more details one can refer to [18 - 27].

2.1 What are wavelets ?

Wavelets are oscillatory waveforms of finite duration and zero average value. These waveforms must be localized. There are many mathematical conditions must be satisfied to ensure that an oscillatory function is admissible as a wavelet basis function. There are many kinds of wavelets whose characteristics vary according to many criteria. One can choose between smooth wavelets, compactly supported wavelets, orthogonal wavelets, symmetrical wavelets, wavelets with simple mathematical expressions, wavelets with simple associated fitters, etc. The simplest and the most important wavelet is the Haar wavelet, and we discuss it as an introductory example in the next section.

2.2 Haar wavelet

The following table shows the main information about haar wavelet.

General characteristic	compactly supported wavelet , the oldest and simplest wavelet
Scaling function phi "φ"	$\varphi(t)$ = 1 on [0,1] and zero other wise
Wavelet function psi "ψ"	$\psi(t)$ = 1 on [0,0.5[, = -1 on [0.5,1] and zero other wise
Family	Haar
Short name	Haar
Example	haar is the same as db1
Orthogonal	Yes
Biorthogonal	Yes
Compact support	Yes
Discrete Wavelet Transform (DWT)	Possible
Continuous Wavelet Transform (CWT)	Possible
Support width	1
Filter length	2
Regularity	haar is not continuous
Symmetry	Yes
Number of vanishing moment for psi	1

2.3 Main idea of wavelet and haar as example

The main idea of wavelets is represent the signal with two part the first is the slow varying part(average) and the second is the fast varying part(difference).

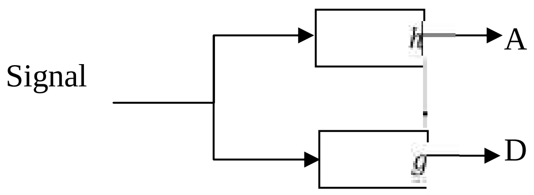


Fig.2.1 : Two band filter to extract average and detail of the input signal

Assume the input signal S = [. . . . , s(0) , s(1) , s(2) , ...]. If h is two-point data averaging and g is two-point data differencing, then we get the simplest wavelet "HAAR WAVELET".

The output of first filter will be A = [$\ldots(s_0+s_{-1}/2),(s_1+s_0/2),(s_2+s_1/2)\ \ldots$] and the output of second is D = [$\ldots(s_0-s_{-1}/2),(s_1-s_0/2),(s_2-s_1/2)\ldots$]. To recover the original signal S from the average values A and detail values D, we can apply reverse operation which is the same as forward operation. In this example they are addition and subtraction.

Average coefficients: $\ldots,\ a_0 = (s_0+s_{-1}/2)$, $a_1 = (s_1+s_0/2)$, $a_2 = (s_2+s_1/2)$, ….
Details coefficients : $\ldots,\ d_0 = (s_{-1}-s_0/2)$, $d_1 = (s_0-s_1/2)$, $d_2 = (s_1-s_2/2)$, ….
Original signal can be recovered using reverse operation (+,-) as following
$(a_0 + d_0) = s_{-1}$, $(a_1 + d_1) = s_0$, $(a_2 + d_2) = s_1$,…..
$(a_0 - d_0) = s_0$,$(a_1 - d_1) = s_1$, $(a_2 - d_2) = s_2$,…..

The Haar wavelet coefficient are h = {1/2,1/2} for averaging and g ={1/2,-1/2} for differencing (fig.2.2). Another point is that the output of *h* are details and it is less important than average values. In many application these values represent the noise and can be removed by applying a non-linear thresholding.

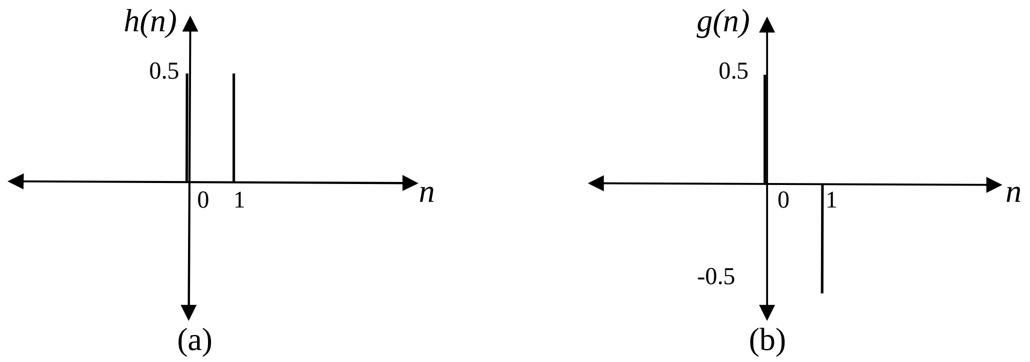


Fig.2.2 : Average and Difference filters
(a) Low pass average filter. (b) High pass difference filter.

The continuous version of Haar is shown below in (fig.2.3), where $\varphi\ t$ is called scaling function and $\psi(t)$ is a wavelet function.

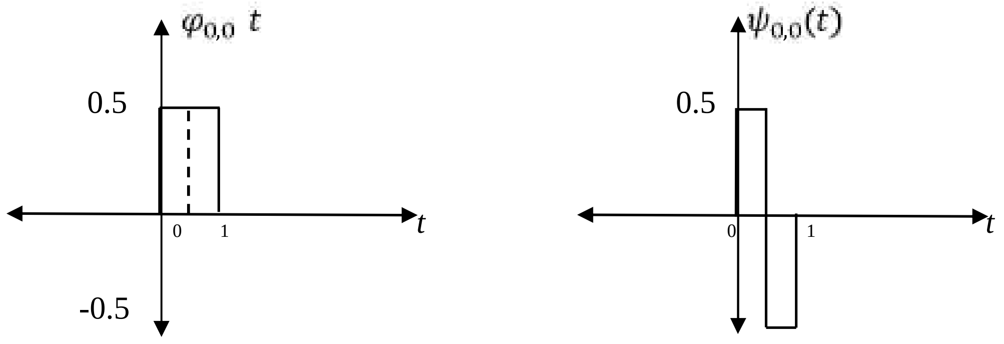


Fig.2.3 :Haar scaling and wavelet functions

$\varphi_{j-1,k}\ t\ =\ 0.5\varphi_{j,2k}\ t\ + 0.5\varphi_{j,2k+1}(t)$(2.1)

$\psi_{j-1,k}\ t\ =\ 0.5\varphi_{j,2k}\ t\ -\ 0.5\varphi_{j,2k+1}(t)$(2.2)

2.4 wavelet and Fourier Transform: comparison

The basis functions of fourier analysis are sine and cosine with infinite duration. These functions are easy to generate, easy to analyze. The back draw of these functions is that they are not local, all of time information lost in frequency domain and all of frequency information lost in time domain. These losses of time and frequency information can be avoided by using wavelet analysis. Wavelet basis functions are local not global with finite duration which mean most of energy concentrate with small duration. Wavelet basis functions are derived using single one function called mother wavelet by time compression and translation. In contrast, the fourier basis which derived by varying the frequency of a sinusoid.

In a summary. The fourier transform can provide frequency information only. The wavelet transform can give us time and frequency information simultaneously.

2.5 Wavelets and Multiresolution Analysis

As mentioned above, the wavelet basis achieved by time compression and translation of mother wavelet.

$$\psi_{jk}\ t := 2^{\frac{j}{2}}\psi\ 2^{j}t-k \qquad j,k\in\mathbb{Z} \tag{2.3}$$

where*j* is the scale factor, *k* is the translation factor.

The wavelet series is shown below with combination of scale and wavelet function.

$$s\ t\ =\ \sum_{k=-\infty}^{+\infty}a_{k}\,\varphi\ t-k\ +\sum_{k=-\infty}^{+\infty}\sum_{j=0}^{\infty}b_{jk}\psi_{jk}(t) \tag{2.4}$$

What we can note from above expression is that the signal $s\ \ t$ is decomposed by two part, the first part gives the approximation and the second gives the details. There are infinite choices to use φ and ψ as basis functions and one can choose the best one depend on application. Another thing, the small coefficients in $\{a_k\}$ and $\{b_{jk}\}$ can be discarded by applying thresholding technique as we will see in the next chapter.

In section 2.3 we have shown how haar scale and wavelet are expressed as a sum of $\{\varphi\ \ 2.-k\ \}_{k\in Z}$.

$\varphi\ t\ = 0.5\varphi\ 2t\ +0.5\varphi(2t-1)$

$\psi\ t\ = 0.5\varphi\ 2t\ - 0.5\varphi(2t-1)$

In general form

$\varphi\ t\ = 2\sum_{k=0}^{N} h_0\ \ k\ \ \varphi\ 2t-k \qquad k\in\mathbb{Z}, h_0\in\ell^2\ \mathbb{Z}$ (2.5)

$\psi\ t\ = 2\sum_{k=0}^{M} g_0\ \ k\ \ \varphi(2t-k) \qquad k\in\mathbb{Z}, g_0\in\ell^2\ \mathbb{Z}$ (2.6)

These equations are called dilation equation, $\{h_0\ (k)\}_{k\in Z}$ and $\{g_0\ (k)\}_{k\in Z}$ are scaling sequence (N-coefficients of low-pass filter) and wavelet sequence (M-coefficients of high-pass filter), respectively.

The relation in equation (2.5) and (2.6) is two-scale relation. The scaling and wavelet function are a combination of rescaled scaling function $\{\varphi(2t-k\}_{k\in Z}$ and this introduce us to what so called multiresolution analysis.

Definition : A Multiresolution Analysis is a sequence of nested, closed subspaces$\{V_j\ \}_{j\in Z}$ if the following statements are satisfied :

1 .$V_j \subset V_{j+1} \quad \forall j\in\mathbb{Z}$

2 .$x\ t\ \in V_j \Leftrightarrow x\ 2t\ \in V_{j+1} \quad \forall j\in\mathbb{Z}$

3 . $x\ t\ \in V_0 \Leftrightarrow x\ t-k\ \in V_0 \quad \forall k\in\mathbb{Z}$

4 .$\overline{\cap_{j\in Z} V_j} = 0$

5 .$\overline{\cup_{j\in Z} V_j} = L^2(\mathbb{R})$

6 .$\exists\ Orthogonal\ basis\ so\ V_0 = \overline{span}\ \ \varphi\ t-k \quad \forall k\in\mathbb{Z}$

The complement of V_j is called details space W_j . Hence, we can decompose V_{j+1} into

$V_{j+1}\ = V_j\quad W_j$ (2.7)

As an example, from equation (2.5) we see that $\varphi(t) \in V_0 \subset V_1$, and from equation (2.6) we see that $\psi(t) \in W_0 \subset V_1$, and this imply that

$$V_1 = V_0 \quad W_0 \tag{2.8}$$

The building block of this decomposition in the discrete time domain can be seen as two channel filter bank as shown in (fig.2.4).

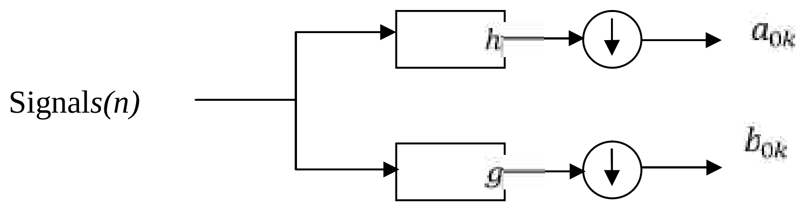


Fig 2.4 :One level two channel analysis filter bank

Fig.2.5 shows how the signal is divided by a two channel filter bank into two signals, the first one is the approximated signal with low frequency and the second is the detailed signal with high frequency.

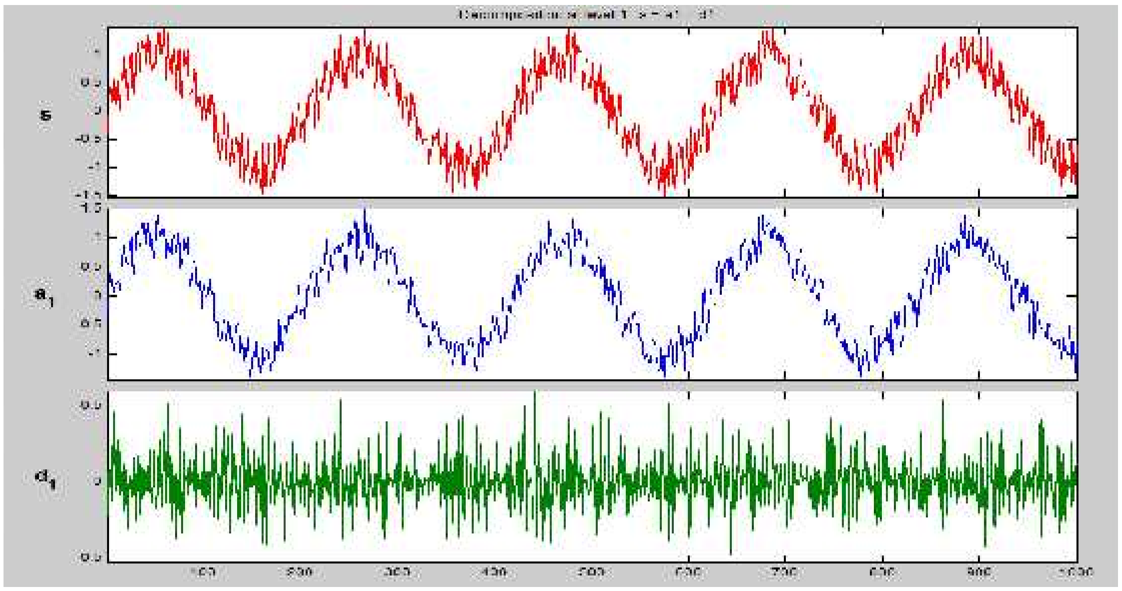


Fig2.5 : One level decomposition of (sinusoidal signal + white noise).

Decomposition of signal can take any level by iterating the filter bank at each output of the low pass filter. One can also iterate this filter bank at the output of high pass filter in addition to the iteration at the output of low pass filter, in this case it is called wavelet packet decomposition.

To reconstruct the signal, we can use inverse operation to the two channel analysis filter bank. The construction of synthesis filter bank is shown below (fig.2.6).

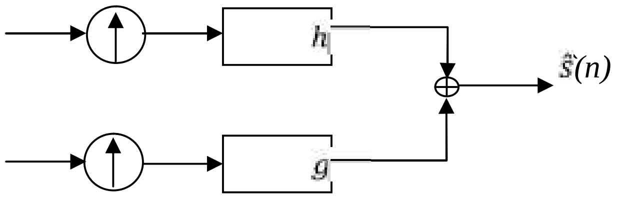


Fig 2.6 :One level two channel synthesis filter bank

The overall analysis and synthesis filters are shown below (fig.2.7). The filtering is linear, the thresholding is not. One can write the filtering and down/up-sampling in a matrix form.

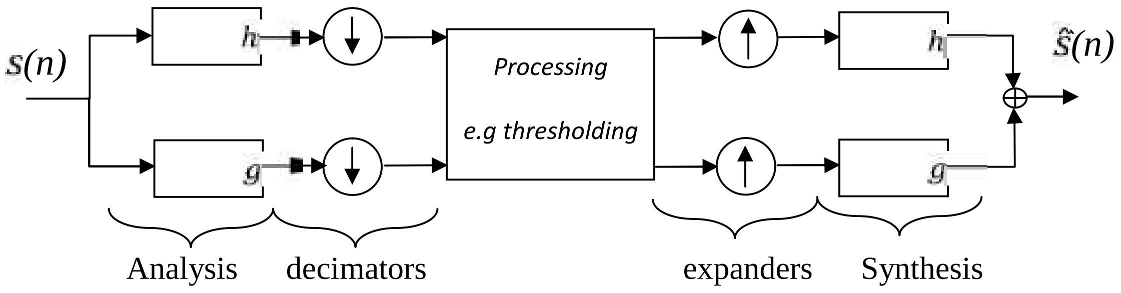


Fig 2.7 : Analysis and Synthesis two channel filter bank

Assume that the input signal $s(n)$ is the sampled signal of $s(t)$. The discrete signal $s(n)$ can be represented as N-points vector.

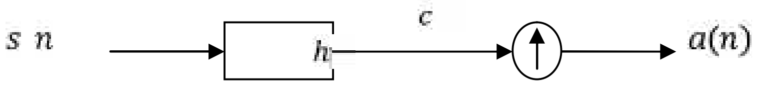


Fig.2.8 : First channel (low pass channel) in Analysis part

$$c(n) = s(n) * h_0(n) = \sum_k s(k)\, h_0(n-k) \tag{2.9}$$

The matrix representation of equation (2.9) is

$$\begin{bmatrix} \vdots \\ c(-1) \\ c(0) \\ c(1) \\ \vdots \end{bmatrix} = \begin{bmatrix} \ddots & & \ddots & & \ddots \\ & h_0(0) & h_0(-1) & h_0(-2) & \\ \ddots & h_0(1) & h_0(0) & h_0(-1) & \ddots \\ & h_0(2) & h_0(1) & h_0(0) & \\ \ddots & & \ddots & & \ddots \end{bmatrix} \begin{bmatrix} \vdots \\ s(-1) \\ s(0) \\ s(1) \\ \vdots \end{bmatrix} = H_0\ S$$

where H_0 is a low pass filter matrix , S is the input signal vector. For a causal filter $h_0(n) = 0$ for n<0.

After this matrix operation, the down sampler discards the odd rows so that the number of input samples equals to the output samples.

Chapter 3

Wavelet de-noising algorithm

Wavelet de-noising algorithm

3.1 Wavelet de-noising model

Wavelet de-noising is a non-parametric method which does not need parameter estimation of the speech enhancement model. Estimating the signal corrupted by Gaussian noise is considered as an important problem in many studies, and it will be our interest in this project. We will restrict our study only to Additive White Gaussian noise.

Let us consider a speech signal s_i, and an independently and identically additive white gaussian noise $n_i \sim N(0, \sigma^2)$, the noisy signal can be written as follows

$y_i = s_i + n_i \qquad {}_{i=0,1,\ldots\ldots,N-1}$(3.1)

The goal of wavelet de-noising is to find an approximation $\tilde{y}_i$ to the signal s_i, that minimize the mean squared error

$E \quad s - \tilde{s} \quad = \sum_{i=0}^{N-1} E[s_i - s_i]^2$ (3.2)

Where $s = [s_0 \quad s_1 \quad \cdots \quad s_{N-1}]^T$ and $\tilde{s} = [\tilde{s}_0 \quad \tilde{s}_1 \quad \cdots \quad \tilde{s}_{N-1}]^T$

Applying the wavelet transform matrix W, the equation (3.1) becomes as follows

$Wy_i = Ws_i + Wn_i$ (3.3)

$y^c_{j,k} = s^c_{j,k} + n^c_{j,k}$ (3.4)

where $(.)^c_{j,k}$ are the wavelet coefficients.

Because of using orthogonal transform W to express s_i in an orthogonal wavelet basis, the wavelet coefficients of the i.i.d Gaussian noise are also i.i.d Gaussian. This kind of transformation preserved the statistical independence of the noise and it is called a unitary transform.

By choosing a good matched wavelet for signal representation, the noise power will tend to concentrate in a small coefficients while the most of signal power will be in large coefficients. This idea of a sparse representation due to the wavelet transform allows us to remove the noise from the signal by discarding the small coefficients which represent the noise. To do that we need to apply a wavelet thresholding function $T(.)$ on a wavelet coefficients.

$$T(y^c_{j,k}) = T(s^c_{j,k}) + T(n^c_{j,k}) \qquad (3.5)$$

$\bar{y}^c_{j,k} = \bar{s}^c_{j,k} + \bar{n}^c_{j,k}$(3.6)

where $\{\ .\ {}^c_{j,k}\}$ are the wavelet coefficient after thresholding.

Now the inverse wavelet transform can be applied to get the estimate signal $\tilde{s_i}$

$$\tilde{s_i} = W^{-1}\ \bar{y}^{c}_{j,k} \quad (3.7)$$

$$\tilde{s_i} = W^{-1}(T\ \ W y_i\ ) (3.8)$$

From equation (3.8), the thresholding will introduce some effects on the signal's power. Thresholding is not linear and it is a lossy algorithm. Thus, it is impossible to filter out the noise without affecting the signal.

There are three basic steps (fig.3.1) for the de-noising algorithm as follows :

1. Decomposition: compute the discrete wavelet transform of a noisy signal.
2. Thresholding: remove the small coefficient based on the kind of threshoding function and threshold value.
3. Reconstruction: compute the discrete inverse wavelet transform.

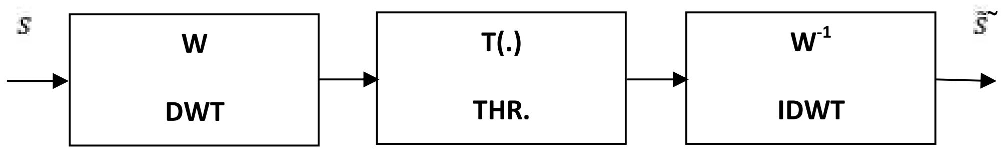


Fig.3.1: Procedure for reconstructing a noisy signal

The most common thresholding function (fig.3.2) or decision rule that used for coefficient thresholdingare

1. Hard thresholding function.
2. Soft thresholding function (also called the wavelet shrinkage functions).

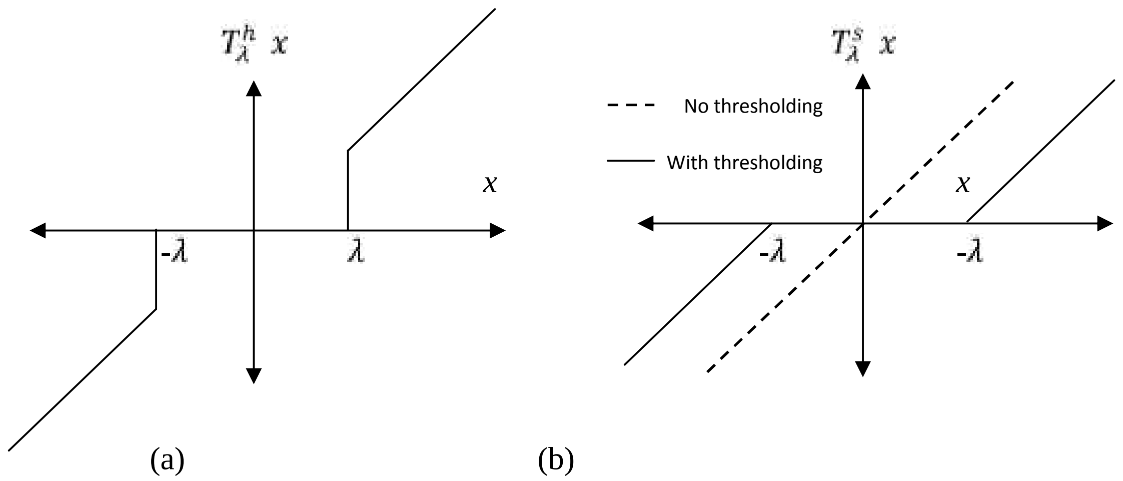
Hard thresholding keeps the wavelet coefficients above the specific threshold and set the rest of coefficients to zero. Soft thresholding removes the coefficients below the threshold value and shrinks the coefficient above it toward the zero. There is no discontinuity in the case of soft thresholding which is more suitable than hard thresholding. This means that hard thresholding is more sensitive to small change in the data. Hard thresholding tends to introduce a high variance because of the discontinuity while soft thresholding tends to introduce high bias due to the shifting of all the coefficient which are greater than the threshold λ with amount equal to the threshold value.

The mathematical description of these two thresholding functions are shown below

Hard thresholding : $$T_\lambda^h\ x\ = \begin{cases} x \quad, |x| \geq \lambda \\ 0, |x| \geq \lambda \end{cases} \qquad where\ \lambda \in [0, \infty[\qquad (3.9)$$

Soft thresholding : $$T_\lambda^s\ x\ = \begin{cases} sgn\ x \quad x - \lambda \qquad, |x| \geq \lambda \\ 0, |x| < \lambda \end{cases} \qquad (3.10)$$

$where\ \lambda \in [0, \infty[$



(a) (b)

Fig.3.2: Hard and Soft thresholding functions.

(a) Hard thresholding function. (b) Soft thresholding function

There are another variants of these threshold functions try to obtain smoother thresholding/shrinking functions, The idea is getting effective de-noising and preserving more useful information of the clean signal.

The threshold parameter could be fixed or changed. The selection of the threshold value is very important to get good result of de-noising. There are different standard methods of selecting a threshold and here we introduce the most common methods.

1. Universal method :

It is a fixed threshold de-noising method and the proper selection of the threshold for a discrete wavelet transform (DWT) is determined as follow

$$\lambda = \hat{\sigma}\ \ \overline{2log_e\ N} \qquad (3.11)$$

where N is the length (number of samples) in the noisy signal and $\hat{\sigma}$ is an estimate of the standard deviation of zero mean additive white gaussian noise calculated by the following median absolute deviation formula

$$\hat{\sigma} = \frac{median\ \ y^d_{1,k}}{0.6745} \qquad (3.12)$$

where $y^d_{1,k}$ is the details wavelet coefficient sequence of the noisy signal on first level.

For a wavelet packet transform (WPT), the threshold can be calculated by

$$\lambda = \hat{\sigma}\ \ \overline{2log_e\ \ Nlog_2(N)} \qquad (3.13)$$

where$N$ is the noisy signal length and $\hat{\sigma}$ is the standard deviation.

The universal threshold method uses global thresholds. This means, the computed threshold is used for all coefficients. This method of threshold selection depends on the statistical variance measurement of the noise and noisy signal length only.

2 .Minimaxmethod :

In this method, the threshold will be selected by minimizing the error between the wavelet coefficient of noisy signal and original signal. The noisy signal can be seen as unknown regression function, this kind of estimator can minimize the maximum mean square error for a given unknown regression function.

The threshold value can be calculated by

$$\lambda = \hat{\sigma}\lambda_n \qquad (3.14)$$

whereλ_n is calculated by a minimax rule such that the maximum error across the data is minimized.

The threshold selection in this method is independent of any signal information. Thus, it is good primarily choice for completely unknown signal information.

3 . SURE method :

SURE (Stein's unbiased risk estimator) is an adaptive thresholdingmethod that uses a threshold value λ_j at each resolution level j of the wavelet coefficients. In the level dependent universal threshold, the threshold at each scale j is selected as

$$\lambda_j = \hat{\sigma}_j \quad \overline{2log_e(N_j)} \tag{3.15}$$

whereN_j is the samples number in the scale *j* and $\hat{\sigma}_j$ is an estimate of the standard deviation in the scale *j*.

This method is a good choice for non-stationary noise, in this case the variance of the noise wavelet coefficients will differ for different scales in the wavelet decomposition.

Adaptive thresholding can be used to enhance the performance of de-noising algorithm, The threshold can be selected based on the data information in any generic domain. One choice of generic domain is the energy of the data. The threshold value depends not only on N but also on the energy of the data frame as follows

$$\lambda_i = T \quad E_i \tag{3.16}$$

whereE_i is the energy of the data i^{th} frame in a signal.

Since the speech and the noise are uncorrelatedand , from equation (3.1) and (3.4), we have the following relations

$$E_{y_i} = E_{s_i} + E_{n_i} \tag{3.17}$$

$$E_{(Wy_i)} = E_{(Ws_i)} + E_{(Wn_i)} \tag{3.18}$$

where E is the signal energy in the i^{th} frame

Equation (3.17) and (3.18) show that the energy of the noisy speech signal frame in the wavelet domain is equal to the energy of the noisy signal in a time domain. The energy transformation between time and wavelet domains is preserved.

In this project, we only concentrate our study about a single channel (single microphone) speech de-noising system which does not use multi-channel for noise reduction.

3.2 Algorithm for speech de-noising

The main steps of the de-noising procedure are shown below. Fig.3.3 shows the flow chart of algorithm for speech de-noising.

Summary of the algorithm :

1. Add a random additive white Gaussian noise to the clean signal.
2. Segment the noisy signal into frames.
3. Make the discrete wavelet transform for every input frame.
4. Calculate the energy of wavelet coefficient and zero crossing rate.
5. Based on the previous point, the feature of the frame is extracted to classify every noisy speech frame into one of three classes (voiced/unvoiced/silence).

6. The threshold value will change depend on the classifier output.
7. Make the inverse discrete wavelet transform.
8. Apply a performance measurement on the de-noised signal.

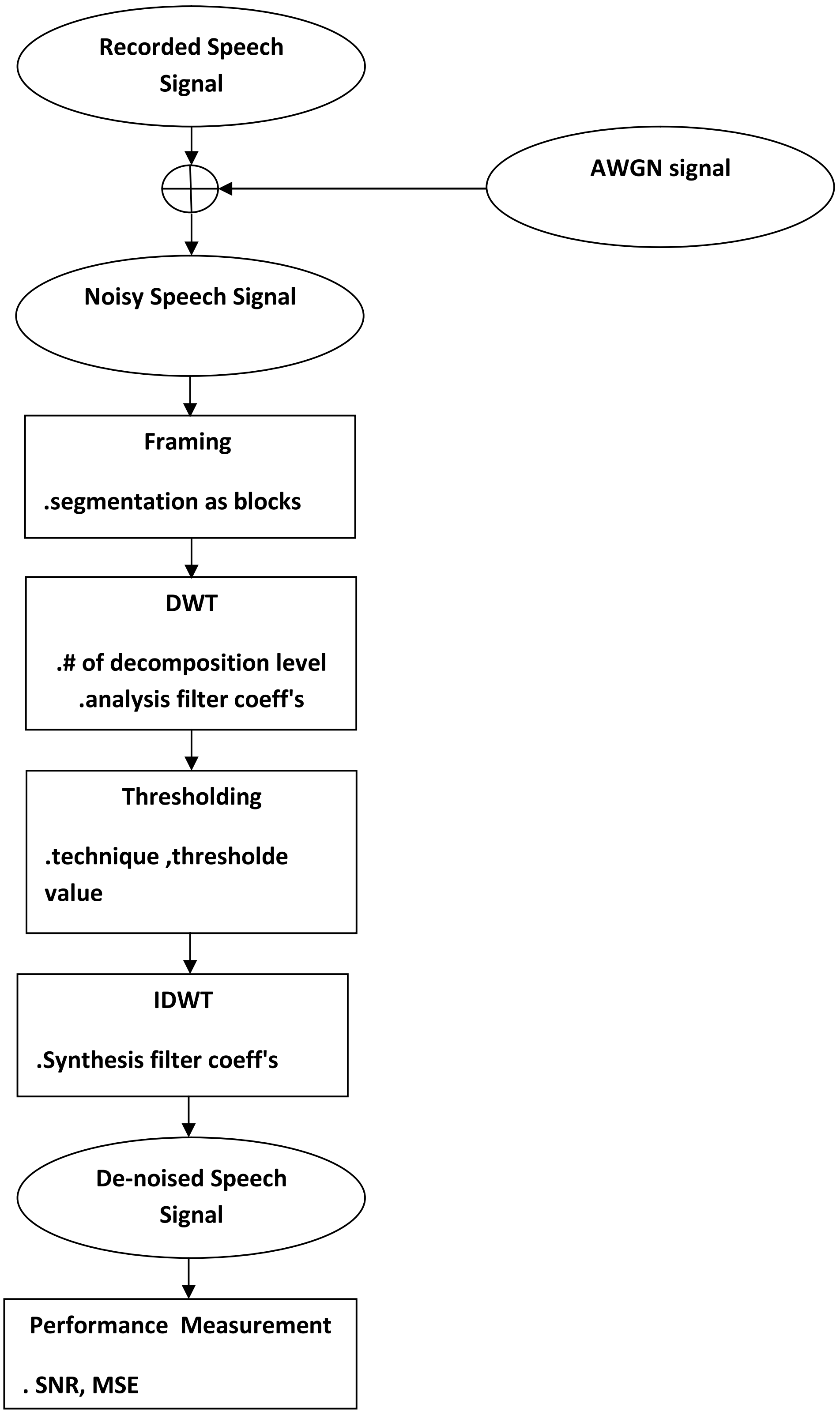


Fig.3.3 : Block diagram of the de-noising system

3.3 Challenges

Some challenges in applying the above algorithm :

1. Segmentation.
2. Voiced / unvoiced / silence.
3. Filter coefficients of analysis and synthesis.
4. Number of decomposition levels.
5. Thresholding type.
6. Threshold value is very important parameter.
7. Level of noise.

Now, let's proceed to investigate these challenges in more depth.

The first challenge is to segment the speech signal with proper frame duration, the frame with N samples can be conceived as N dimensional vector space, and when analyzing this vector of samples some features contained in the frame could be lost if we do not choose the proper framing mythology. The solution of this problem can be solved by introducing overlap frames. By choosing a proper widow for segmentation with some percent of overlapping we can minimize the losses of features in the frame.

Each frame will typically contain 100 sample if we assume the sampling frequency equal to 8 KHz. This imply that the frame duration will be 12.5 ms. We need to choose the number of sample in each frame as a power of 2 to avoid using signal extension(e.g.128samples).

The second challenge is that when applying the thresholding on the speech signal, the possibility of speech degradation is exist since some of frame is unvoiced which mean that most of energy of the frame is concentrated in the high frequency bands and eliminating of them will make a degradation in the quality of the de-noised signal.The solution of this problem is the most hardest part in this algorithm. However by choosing a proper decision rule for classification process we can avoid the speech degradation. Here we introduce two features and its equations

. Short – term average energy :

$$E_i = \frac{1}{N}\sum_{l=1}^{N}|y_i\ l\ |^2 \qquad (3.19)$$

where N is thei^{th} frame length and *l* is the data index.

. Zero crossing rate: calculate the number of sign changes of successive samples in the i^{th} frame.

$$ZCR_i = \sum_{l=1}^{N} sgn\ y_i\ l\ \ - sgn\ y_i\ l-1 \qquad (3.20)$$

wheresgn is the signum function.

These features are typically estimated for frames of speech with 10-20 ms duration.

The choice of widow type determines the nature of short-term average energy representation. If size of the widow is very long, then it is equivalent to a very narrowband low pass filter, that means the short term energy will reflect the amplitude variation in a speech signal.In contrast, if the window size is very short, the short-term energy will not provide a sufficient energy averaging.

Zero crossing rate reflects the frequency content in the frame. It is important to remove any offset in a signal to ensure a correct calculation in the case of zero crossing rate.

We can use short term energy and zero crossing rate to change the threshold value based on Voiced/Unvoiced/Silence classification.

. High energy and low zero crossing rate imply that the frame is voiced.

- Most of the power for the voiced frame is contained in the approximation part of wavelet decomposition.

. Low energy and high zero crossing rate imply that the frame is unvoiced.

- Most of the power for the unvoiced frame is contained in the details part of wavelet decomposition.

. Relatively equal power distribution imply that the frame is silence.

The third challenge is about the wavelet filter design. Choosing an appropriate filter coefficient is considered a critical part in all of this process of de-noising. There are several criteria that could be used to select the best wavelet filter. In this project we tended to use the most simple and the most important filter bank which is the Haar filter. This filter is considered as a good choice since it has a different property such as symmetry, orthogonality, biorthogonality, compactness and sparsity. We will investigate many other db wavelets with higher vanishing moments.

The fourth challenge is selecting the number of levels for wavelet decomposition. Generally speaking, the number of needed level for decomposition will increase as the power noise increases, however, increasing the levels of decomposition increase the computational complexity in the wavelet de-noising algorithm. Practically, increasing the number of the level more than five will not introduce a very significant change in the output signal to noise ratio. The selection of number of levels will depend on the kind of the signal or on some criteria as entropy.

The fifth challenge is choosing an appropriate threshold function, in this project we intend to use soft thresholding function since it is more stable than hard thresholding.

The sixth challenge is about how we can choose the threshold value. As discussed in the previous section the threshold should be adapted to avoid the speech degradation, the choice of threshold will be chosen such that the small coefficients are best threshold with high threshold values, whereas the small coefficients needed to be threshold with small threshold values.

The seventh challenge is about the level power of the additive noise. Practically, if the SNRs of the noisy signal is very low, such this method will fail since the noisy coefficient will becomes significantly large so that it is difficult to distinguish between the clean and noisy coefficient. In this project the signal to noise ratio level will be about 0 dB to 20 dB.

Actually, speech is a complex noise process and these are not the only challenges nor the only typical solution. There are a lot of optimization and adaptation process to get more optimum de-noising algorithm that could be used with a diverse conditions.

For performance measurement, objective and subjective quality can be used to provide a measure how much improvement occurred before the processing. The goal is to increase the output signal to noise ratio (SNR) in each frame such that the average SNR isincreased.

Objectively, there are two common measure as follows

. Signal to noise ratio SNR :

$$SNR_{i,out} = \frac{\sum_{n=1}^{N} s_i^2(n)}{\sum_{n=1}^{N} (s_i(n) - \tilde{s_i}(n))^2} \quad (3.23)$$

where $SNR_{i,out}$ is the segmental output signal to noise ratio of the i^{th} frame , $s_i(n)$ is the i^{th}input frame of the clean speech signal and $\tilde{s_i}(n)$ is the i^{th} output enhanced frame of the speech signal.

. Mean Square Error MSE :

$$MSE_i = \frac{1}{N}\sum_{n=1}^{N} (s_i(n) - \tilde{s_i}(n))^2 \quad (3.22)$$

whereMSE_i is a mean squared error in the i^{th} frame.

Chapter 4
Speech enhancement evaluation

4.1 Matlab code

4.2 Performance evaluation

Speech enhancement evaluation

In this chapter we are going to construct a matlab code for the wavelet de-noising algorithm and tackle the different challenges which discussed in previous chapter. After that, the discussion about the results is introduced in the context of the performance evaluation.

As mentioned before, the main three steps in the de-noising algorithm using wavelet thresholding are decoposition, thresholding and reconstruction. Every step in this algorithm is implemented using matlab programming language. The Wavelet Toolbox in Matlab contains various functions that can be called to build the de-noising algorithm. This kind of programming is called a procedural programming which is a programming paradigm, derived fromstructured programming. The abstraction nature of the function in Matlab is an input-output relation as shown below

[output arguments] = *functionName*(input arguments)

The above statement uses to call the functions built in Matlab. To get an information about how to use a given function, Matlab provides an help documentation about using the functions e.g. (doc *functionName* , help *functionName*). To get the details about the code of any function, the command (edit *functionName*) can be used.

Appendix-A contains the various functions in Wavelet Toolbox which used to write the code of de-noising algorithm. Here, we introduce some of these functions

Function name	Input arguments	Output arguments	Description
wavread	('filename.wav')	[s , Fs , nbits]	Reading audio file
randn	(length(s),1)	n	Random noise (µ ,σ) ~(1,0)
wavedec	(y , N , 'wname')	[Cad , L]	Multilevel 1-D wavelet decomposition
wthcoef	('t' , Cad , L , N , T , s_or_h)	NC	Wavelet coefficient thresholding 1-D
waverec	(NC , L , 'wname')	den_s	Multilevel 1-D wavelet reconstruction

Table 4.1 : Some predefined functions

From above table, the *wavread* function is used to read an audio file, returning the sampled data in s. It also return the sample rate (Fs) in Hertz used to encode the data in the file, and it returns the number of bits per sample (nbits). The *randn* function generates a normally distributed pseudorandom numbers in vector (y). The *wavedec* function performs a multilevel one-dimensional wavelet analysis using a specific wavelet (*'wname'*), returns the wavelet decomposition of the signal (y) at level (N).The *wthcoef* thresholds wavelet coefficients for the denoising of a 1-Dsignal, returns coefficients obtained from the wavelet decomposition structure [Cad , L] by soft (if s_or_h ='s') or hard (if s_or_h ='h') thresholding defined in vectors (N) and (T). Vector (N) contains the detail levels to be thresholded and vector (T) is the corresponding thresholds. (N) and (T) must be of the same length. The *waverec* function performs a multilevel one-dimensional wavelet reconstruction using a specific wavelet (*'wname'*), reconstructs the signal (den_s) based on the multilevel wavelet decomposition structure [NC , L]. For more information about many different functions for wavelet analysis, Reference [ ] provide a lot of details about these functions.

4.1 Matlabcode

Appendix-B shows the Matlab code for de-noising the speech signals. It includes many options that can be used to provide illustrative steps of wavelet de-noising method. The first subsection of this section introduces the different options of this Matlabprogram, the next subsection shows an illustrative example of using the program.

4.1.1 Program options

- ❖ Reading Speech signal and adding noise
  - Reading an audio file stored in computer.
    - o Ability to choose (.wav or .mat) extension.
    - o Ability to take any segment from the signal.
    - o Ability to decide the sample frequency and number of bits per sample.
    - o Ability to decide whether the chosen file is noisy speech or clear speech, in the second case the noise with specific SNR can be added to the clear signal.
  - Online recording speech using microphone
    - o Ability to record a speech signal with specific duration time and sample rate.
- ❖ De-noising using discrete wavelet transform DWT or DWP
  - o Discrete wavelet transform DWT
    - o Ability to decide the number of decomposition levels and wavelet function.
    - o Ability to decide the type of thresholding function (soft or hard).
    - o Ability to choose the global threshold value (the default value is calculated for a given decomposition using universal threshold selection rule).
    - o Ability to segment the speech signal for frame by frame de-noising usinga specific window with percent of overlap between these segments.
    - o Ability to choose the type of thresholding.
      - o Global thresholding
      - o Level dependent thresholding
        - ▪ Manual setting
        - ▪ Based on threshold selection rule
          - ▪ rigrsure , heursure , sqtwolog , minimaxi
      - o Thresholding the details for a given set of levels
        - ▪ Forcing all coefficients at a given levels to zero
        - ▪ Using soft or hard at a given levels
      - o Interval dependent
        - ▪ Manual setting
        - ▪ Based on variance change

- Discrete wavelet packet DWP
    - Ability to decide the number of decomposition levels and wavelet function.
    - Ability to decide the type of thresholding function (soft or hard).
    - Ability to choose the global threshold value (the default value is calculated for a given decomposition using a penalization method).

- Illustration plots
    - Case of DWT
        - Clear and noisy speech signals
        - Scaling and wavelet functions
        - Decomposition and reconstruction filters
        - FFT of filters
        - Decomposition coefficients for each level
        - Reconstructed coefficients for each level
        - Energy of coefficients and variance of detailsfor each level
        - Thresholding functions illustration
        - Noisy, de-noised and residual signals
        - Clear, noisy, de-noised signals
        - Correlation between clear signal and noisy signal before denoising, and correlation between clear signal and de-noised signal after de-noising
        - Power distribution of clear, noisy and de-noised signals
        - Spectrograms of clear, noisy and de-noised signals
        - Absolute coefficients of DWT for clear, noisy and de-noised signals
        - Histogram and cumulative histogram of clear, noisy and de-noised signals
        - Some statistics about residual signal
    - Case of DWP
        - Clear and noisy speech signals
        - Wavelet packets functions at third scale
        - Decomposition and reconstruction filters
        - FFT of filters
        - Thresholding functions illustration
        - Noisy, de-noised and residual signals
        - Clear, noisy, de-noised signals
        - Correlation between clear signal and noisy signal before denoising, and correlation between clear signal and de-noised signal after de-noising
        - Power distribution of clear, noisy and de-noised signals
        - Spectrograms of clear, noisy and de-noised signals
        - Wavelet packet spectrum
        - Histogram and cumulative histogram of clear, noisy and de-noised signals
        - Some statistics about residual signal

- Performance measurements
    - Signal to noise ratio SNR
    - Mean squared error MSE

4.1.2 Illustrative example

In this example, the clear speech signal with duration time equal to four seconds and sample rate equal to 8000 sample/second, every sample is encoded using 16 bit/sample. Normally and identically additive white Gaussian noise with zero mean and variance equal to one tenth of average power of clear signal which implies that the input signal to noise ratio equal to 10 db. The signal is segmented using hamming window of 160 samples and 50% overlapping. Figure 4.1 shows both the clear speech and noisy speech signals.

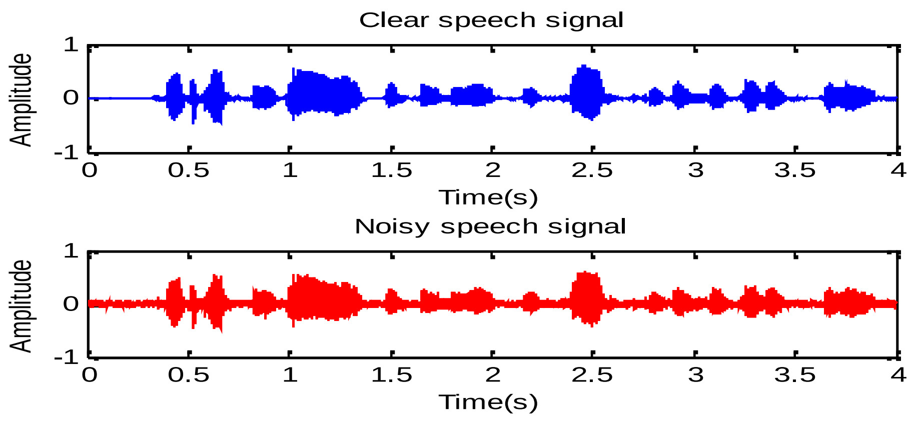


Fig.4.1 : Clear and noisy speech signals

Applying FWT on the noisy speech signal by using three levels of decomposition and db4 as a wavelet function. Figure 4.2 shows the scaling and wavelet function, also it shows the wavelet filers and its FFT. Wavelet function has more oscillation than scaling function so that the integration of wavelet function equal to zero and integration of scaling function equal to one. Using db wavelet with four vanishing moment, the length of each filter will be equal to eight. These filters have a quadrature mirror image property. It is clear from below figure that the analysis and synthesis low pass filters have the same magnitude of FFT, however, they differ in phase, the analysis and synthesis high pass filters also differ in phase.

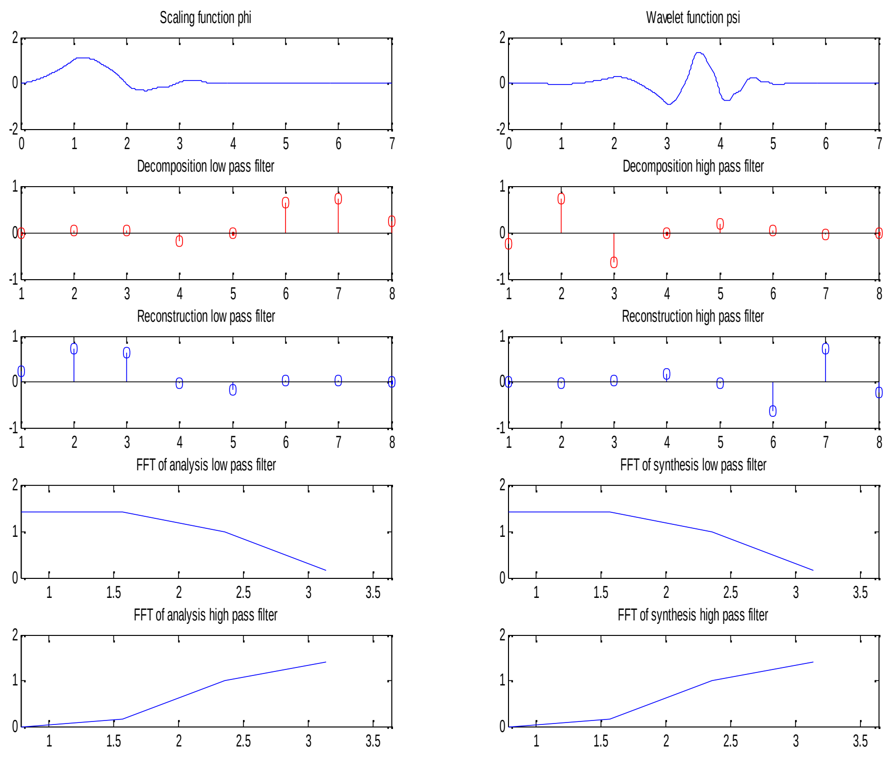


Fig 4.2 : scaling and wavelet functions, decomposition and reconstruction filters, and FFT of decomposition and reconstruction filters

Fig 4.3 shows the wavelet coefficients at each level from the finest scale (third level) to the coarser scale (first level). Figure 4.4 shows the reconstructed signal at each level, the sum of these signals will give the original noisy speech signal. Fig 4.5 shows the energy of coefficients at every scale, and the variance of details at every scale. It is clear from the figure that the largest percent of power is in the third level (approximation coefficients) and small power is concentrated in the first level.

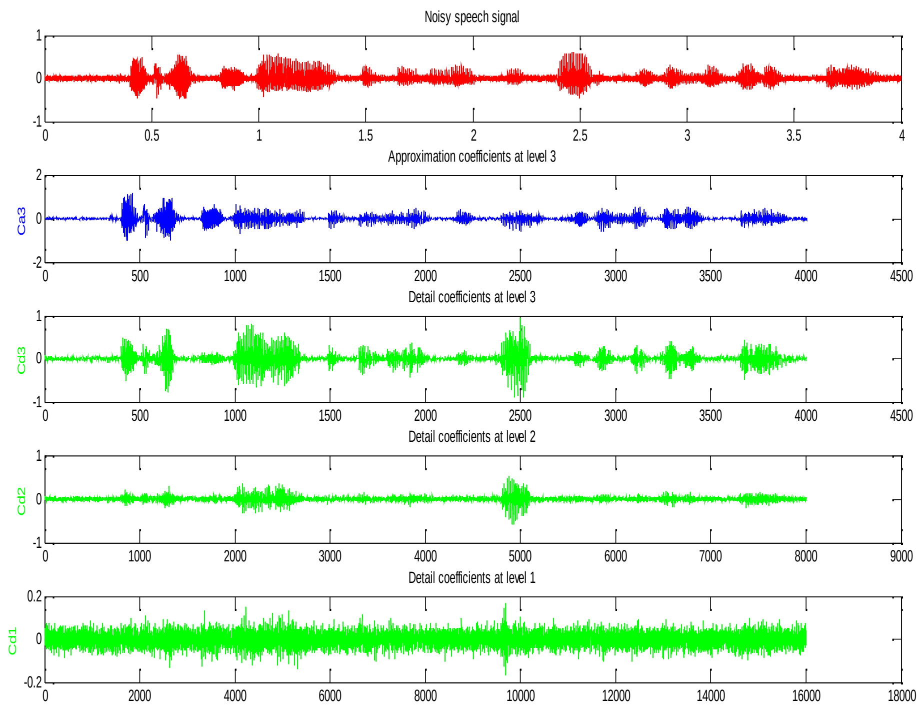


Fig 4.3 : Wavelet coefficients for each level

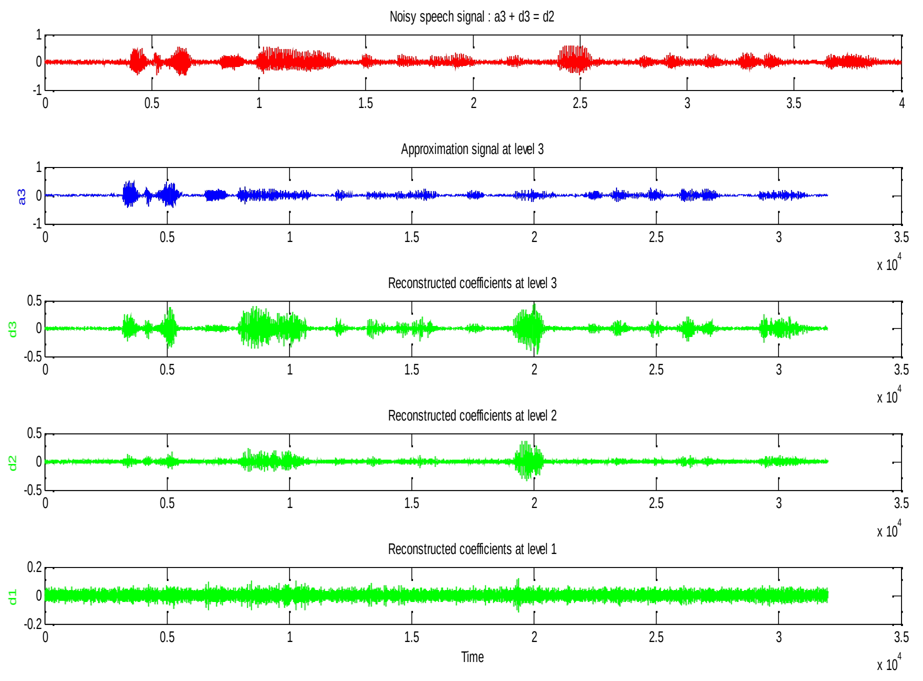


Fig 4.5 Reconstructed signals for each level

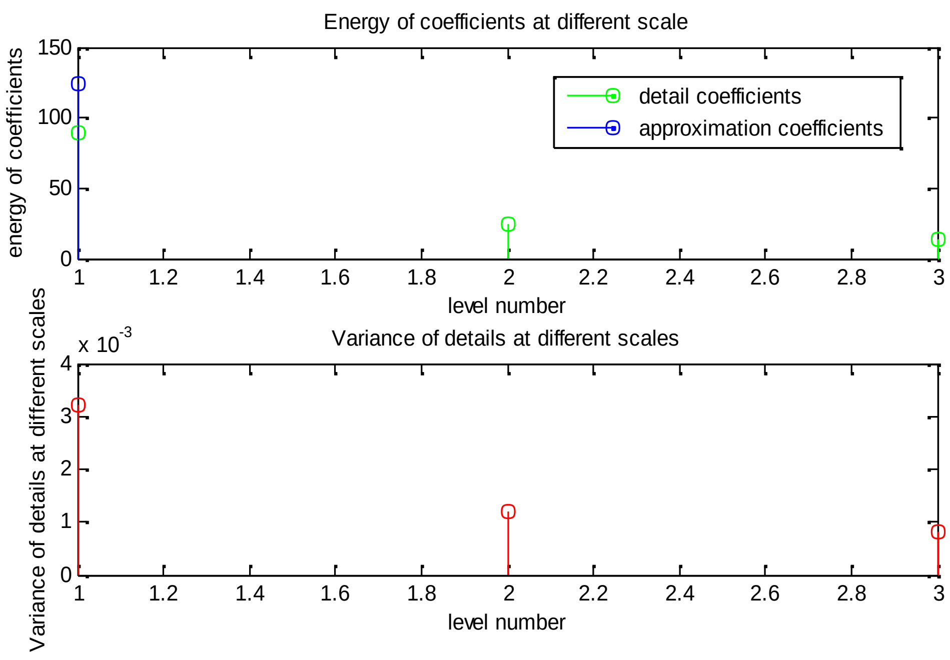


Fig 4.5 : Energy of coefficients and variance of details at different scales

For de-noising process, the soft thresholding function is used and the universal threshod selection rule is applied to fine the global threshold value. The thresholding was not applied on the approximation coefficients. Fig 4.6 shows an illustration about both of thresholding function (soft and hard), the value of global threshold for 32000 samples with no framing is equal to 0.13116

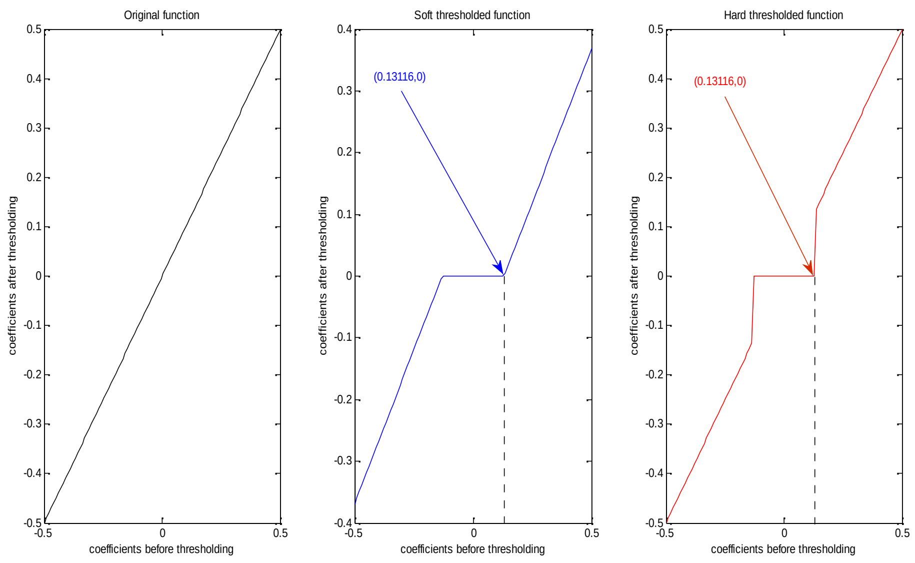


Fig 4.6 : thresholding functions (soft and hard)

Figure 4.7 shows the noisy speech before de-noising and de-noised speech after applying the wavelet de-noising. It is clear from the Fig 4.7 that the noise is reduced, however, there is some residual noise. The bottom of the figur is the residual signal which taken as the difference between the clear signal and the de-noised signal.

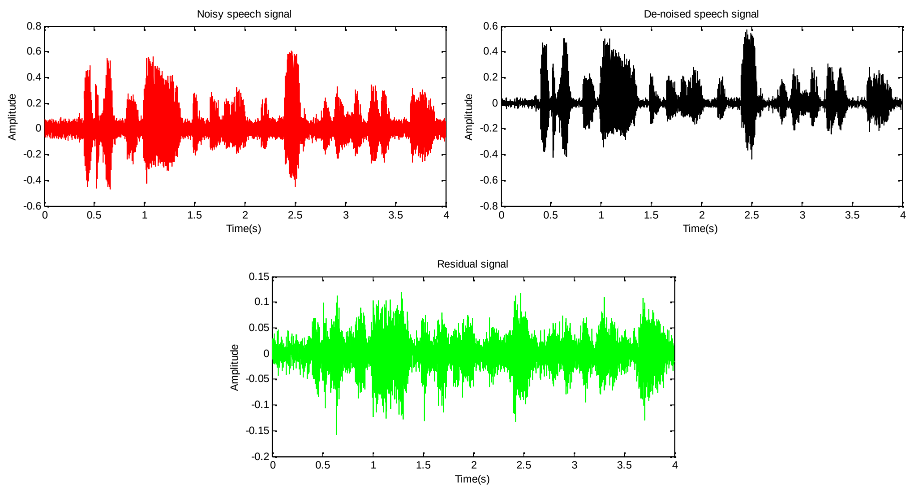


Fig 4.7 : Noisy, de-noised and residual signals

Figure 4.8 shows a comparison between the clear speech signal , noisy speech signal and de-noised speech signal.

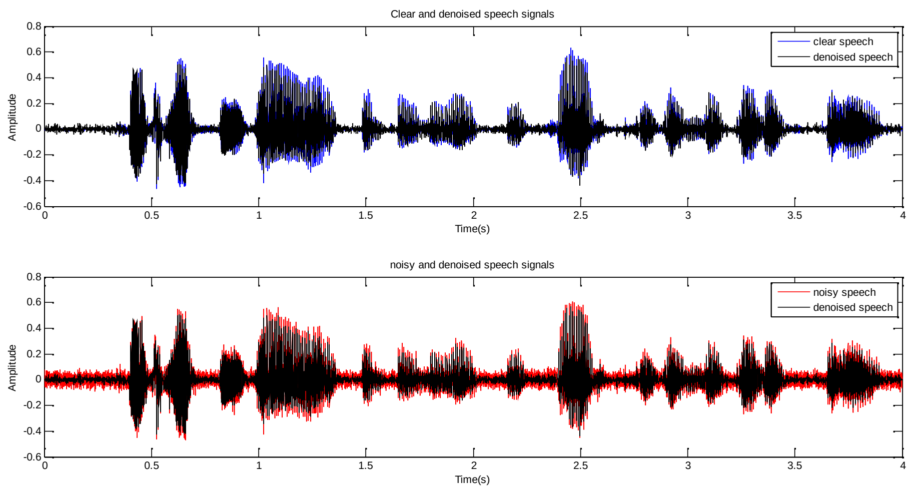


Fig 4.8 : Comparison between clear signal, noisy signal and de-noised signal

The top of Fig 4.9 shows the correlation relation between the clear speech signal and the noisy speech signal, the correlation is equal to 0.9539 at zero lag. In the bottom of the figure, the correlation between the clear speech signal and de-noised speech signal, the correlation is equal to 0.966 which is greater than 0.9539. This indicates that the de-noised signal is tended to become more correlated with the original clear speech signal.

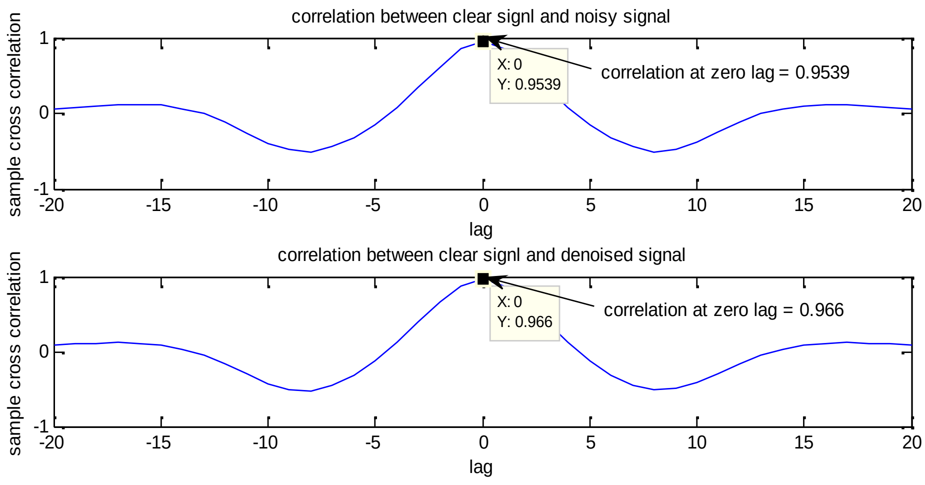


Fig 4.9 : Correlation between clear speech signal and noisy speech signal, correlation between clear speech signal and de-noised speech signal

Fig 4.10 shows the power distribution of the clear, noisy and de-noised speech signals. The power of the additive white Gaussian noise is spreaded over all the frequency band of the speech signal and its power density is constant. As it shown below figure, the most power of the signal is between 0 Hz and 2000 Hz. The power distribution of noisy speech signal indicates that the detail coefficients of low value in the first scale can be discarded to remove the noise in this high frequency band. The thresholding can be applied to the rest of bands in the wavelet decomposition to remove any small coefficient.

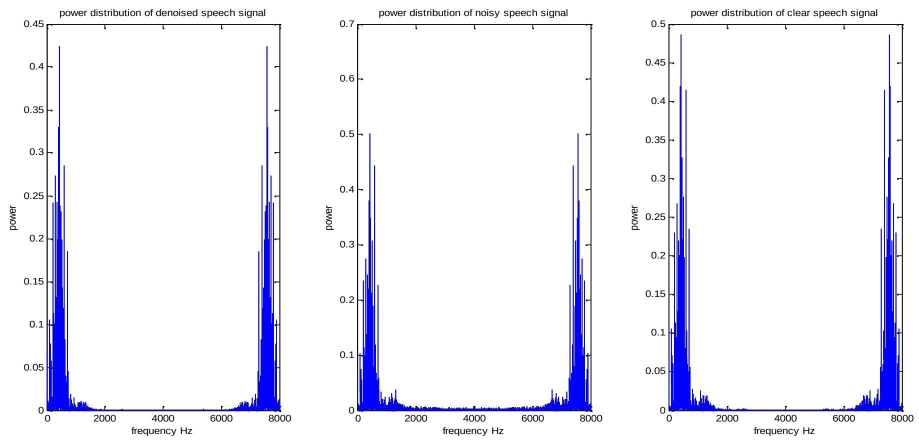


Fig 4.10 : Power distribution of clear, noisy and de-noised speech signals

Fig 4.11 shows the spectrograms of clear, noisy and de-noised speech signals. The spectrogram uses to clarify the time and frequency contents of the speech signal. It is clear from below figure that the spectrogram of de-noised speech signal tends to be more similar to the original clear speech signal. Fig 4.11 and Fig 4.12 show that the power of STFT coefficients of noisy speech signal in the high frequency band is reduced after de-noising process.

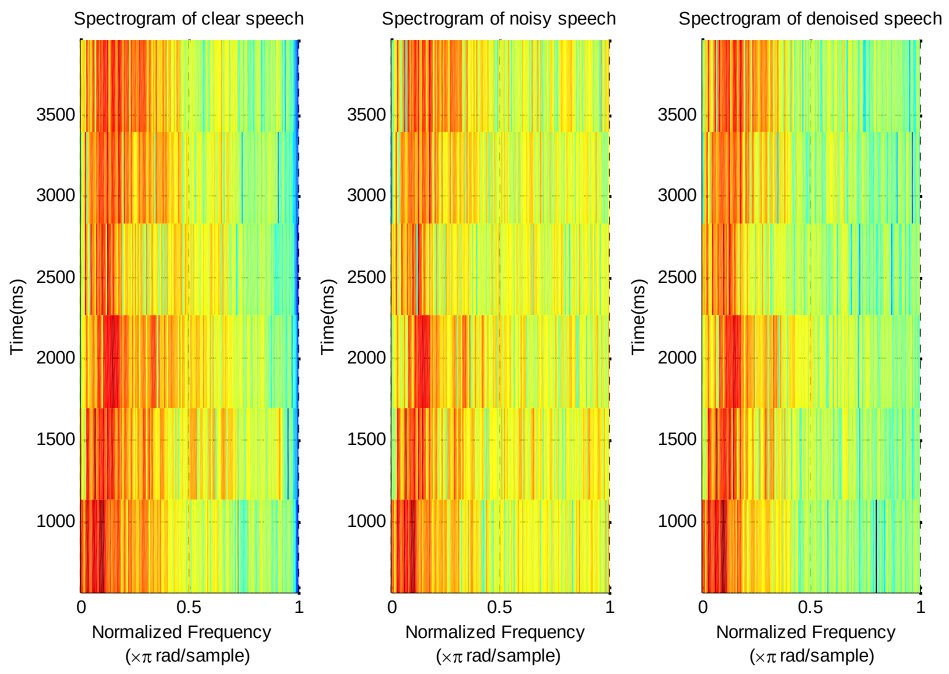


Fig 4.11 : Spectrograms of clear, noisy and de-noised speech signals

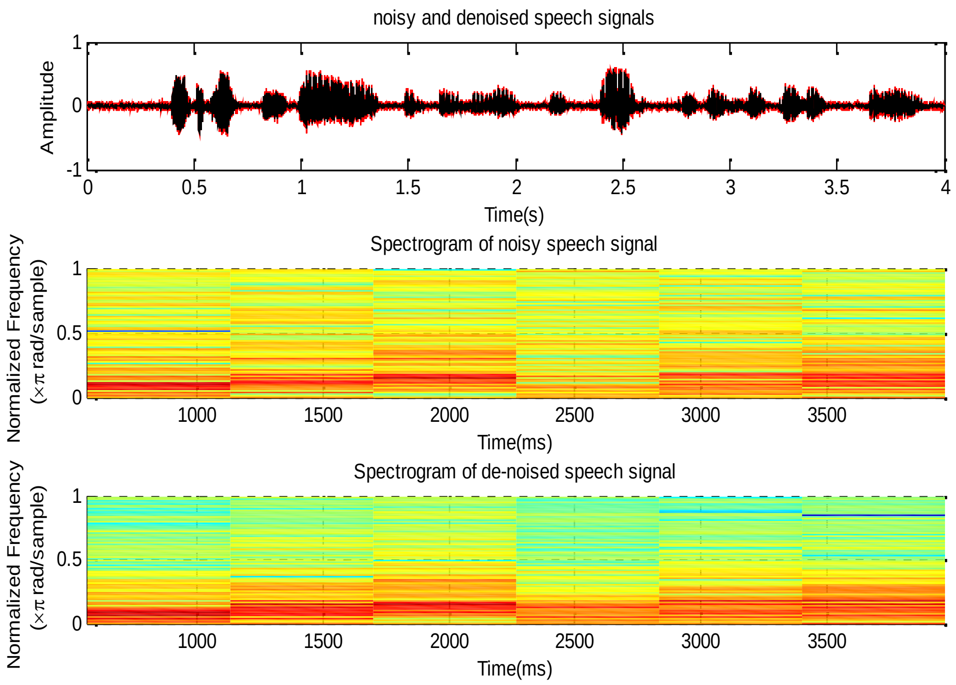


Fig4.12 : Comparison between the spectrograms of noisy and de-noised speech signals.

Fig 4.13 shows the absolute coefficients of DWT for clear, noisy and de-noised speech signals. The percent of noise power in each level is reduced so that most of the power of original speech signal is preserved.

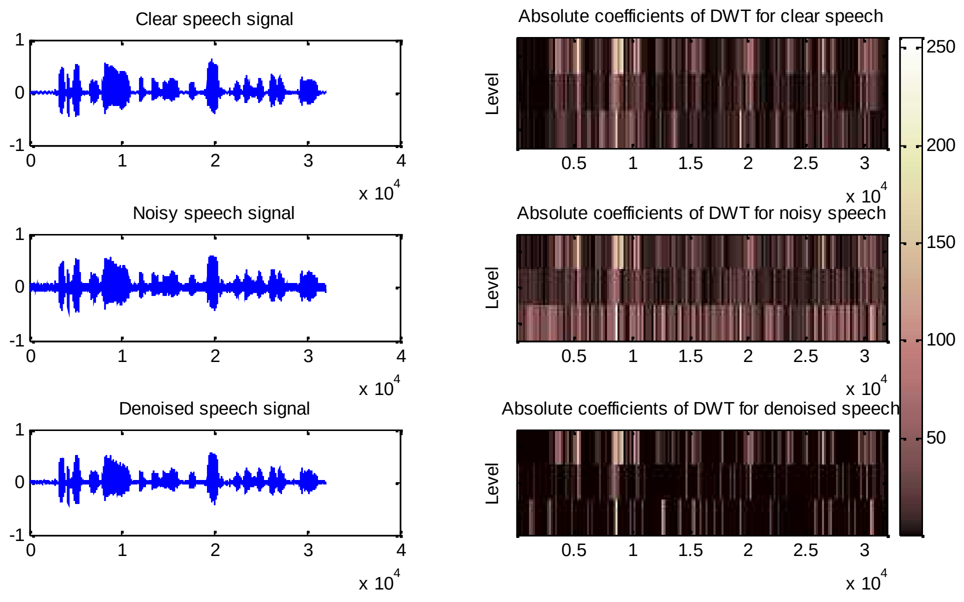


Fig 4.13 : Absolute coefficients of DWT for clear, noisy and de-noised speech signals

Fig 4.14 shows some of statistic measurements about clear, noisy, de-noised. The histograms and cumulative histograms of the clear, noisy and de-noised speech signals indicate that the estimated probability distribution of these three signals are approximately normal distribution. Specifically, Gaussian distribution with zero mean. Since most of the noise power is reduced, the variance of de-noised speech signal is less than the variance of noisy speech signal.

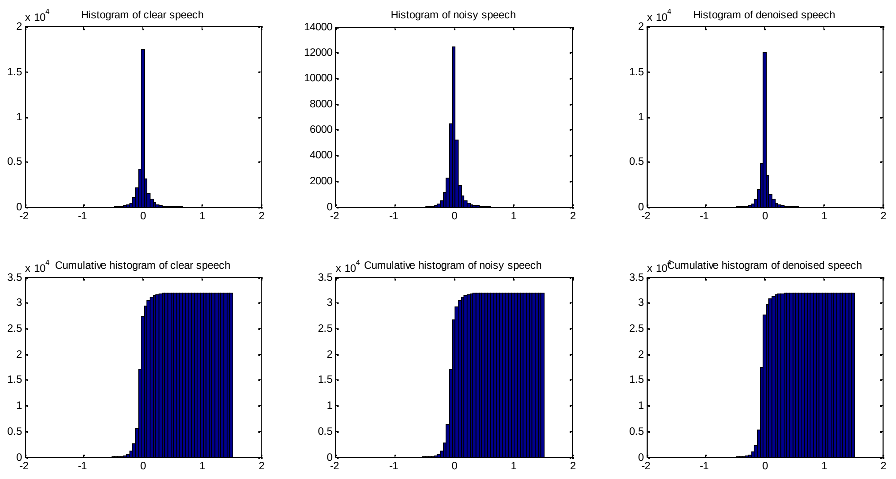
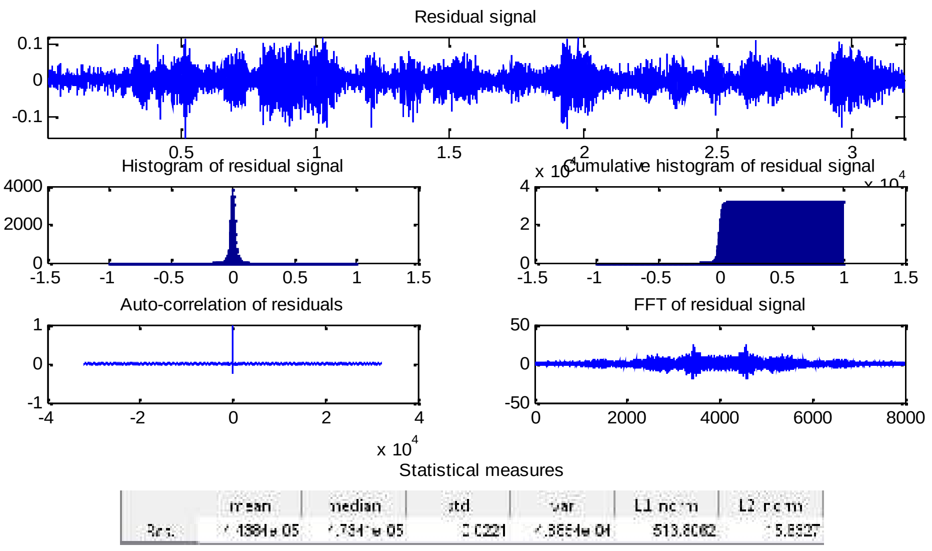


Fig 4.14:Histograms and cumulative histograms of clear, noisy and de-noised speech signals.

Fig 4.15 shows the statistics of residual signal. Residual signal indicate that the noise was not removed totally. Some of statistical measure of this signal such as means, median, standard deviation, variance, L1-norm and L2-norm are shown in below figure, the mean is approximately zero, the variance is very low which is an indication of existing a high frequency components.

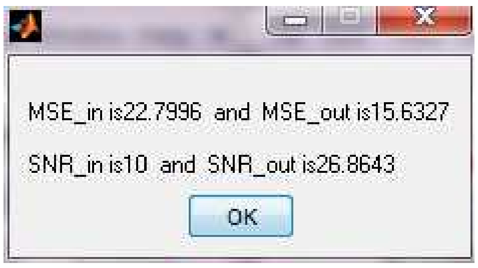
The autocorrelation between the residual samples is equal to zero, the sample is only correlated with itself. The FFT of the residual signal shows that the low frequency band contains some of noise power.



	mean	median	std	var	L1 norm	L2 norm
Res.	[illegible].1381e-05	[illegible].7341e-05	0.0221	4.8854e-04	513.8082	[illegible]5.8927

Fig 4.15 : Statistical measures of the residual signal

Finally, Fig 4.16 is a dialogue which display the output mean squared error and the output signal to noise ratio. The MSE_in is the mean squared error between the clear and noisy speech signals while the MSE_out is the mean squared error between the clear and de-noised speech signals. It is clear that the MSE_out is less than the MSE_in which implies a reduction of noise. The SNR_in is the ratio between the mean squared power of clear signal and the mean squared error between clear and noisy signals. The SNR_out is the ratio between the mean squared power of clear signal and the mean squared error between clear and de-noised signals.



MSE_in is22.7996 and MSE_out is15.6327

SNR_in is10 and SNR_out is26.8643

Fig 4.16 : Output MSE and output SNR after de-noising

4.2 Performance evaluation

We tested several methods for speech de-noising using wavelets. The speech signal with duration equal to four seconds that used is sampled at 8 Khz. Different parameters wereused, some of them are fixed and other was changed to get information about the performance of these methods. The performance measure that we used is the output signal to noise ratio so that it is considered as a dependent variable for all tests.In the following subsections we show the results from these tests .

4.2.1 Global thresholding method

Fig 4.17a shows the relation between output SNR and number of decomposition levels by using global thresholding with different types of db family. The thresholding function that used is soft and the input SNR is equal to 10 db. The speech signal was framed using hamming window with 50% percent of overlapping. The frame length is 160 samples, 80 samples of overlapping with any previous frame.

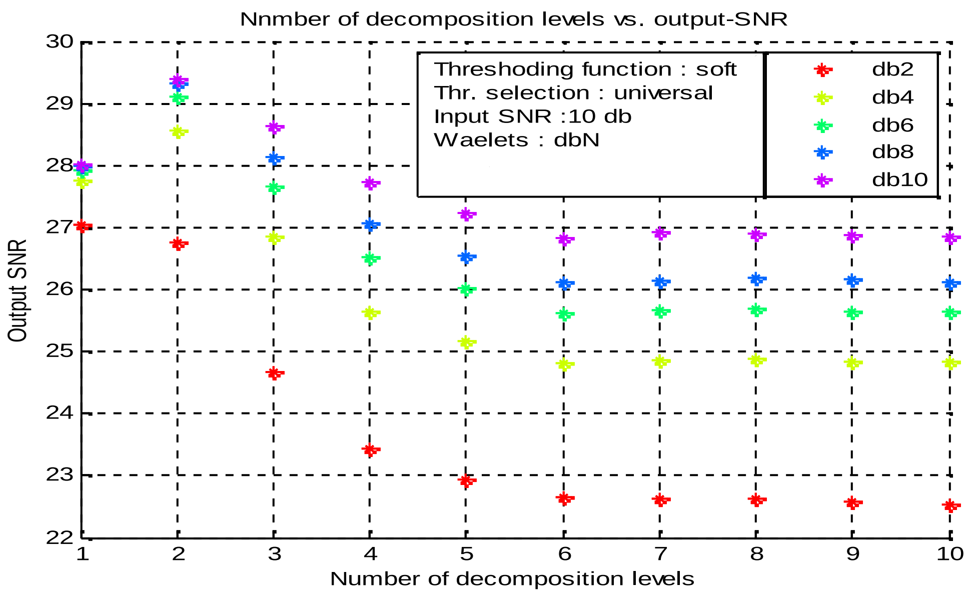


Fig 4.17a :Number of decomposition levels vs. Output SNR using soft thresholding function and universal selection rule

From Fig4.17a, the output SNRs are increased with different levels. Increasing the number of vanishing moments of db wavelet increases the output SNR. Increasing the number of decomposition levels greater than five levels will not introduce a large change for output SNR.

Fig 4.17b shows the relation between the input SNR and the output SNR by using global thresholding with different types of db family. The thresholding function that used is soft and the level of decomposition is equal to six levels. The clear signal is corrupted by additive white Gaussian noise at different level of signal to noise ratio (0 db, 5db, 10db, 15 db and 20 db). . The speech signal was framed using hamming window with 50% percent of overlapping. The frame length is 160 samples, 80 samples of overlapping with any previous frame.

.

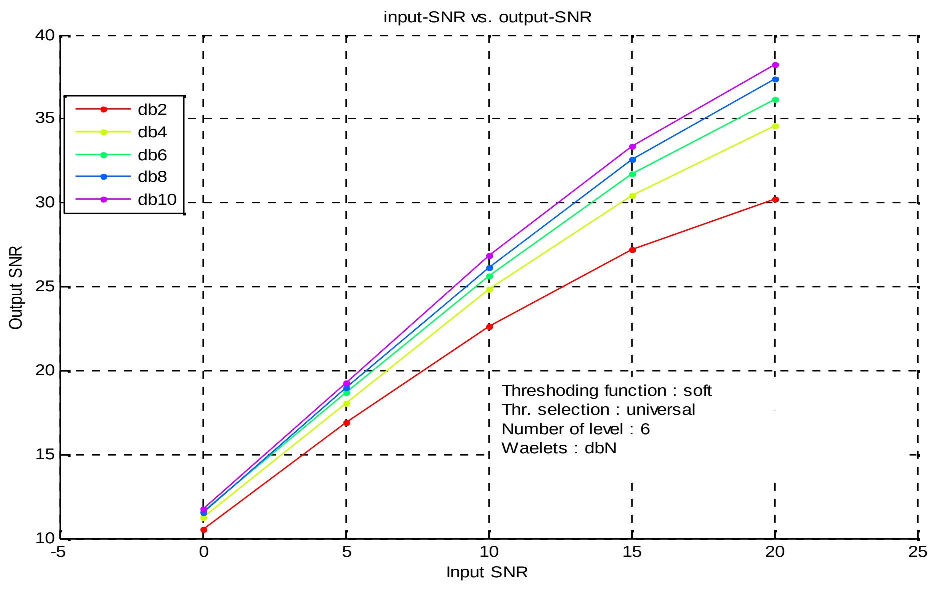


Fig 4.17b: Input SNR vs. Output SNR using soft thresholding function and universal selection rule

From Fig 4.17b,there is an enhancement in the output SNR. Increasing the number of vanishing moments of db wavelet increases the output SNR, but as the power of noise increases then, the rate of increasing of output SNR slows down.

Fig 4.18a shows the relation between output SNR and number of decomposition levels by using global thresholding with different types of db family. The thresholding function that used is hard and the input SNR is equal to 10 db. The speech signal was framed using hamming window with 50% percent of overlapping. The frame length is 160 samples, 80 samples of overlapping with any previous frame.

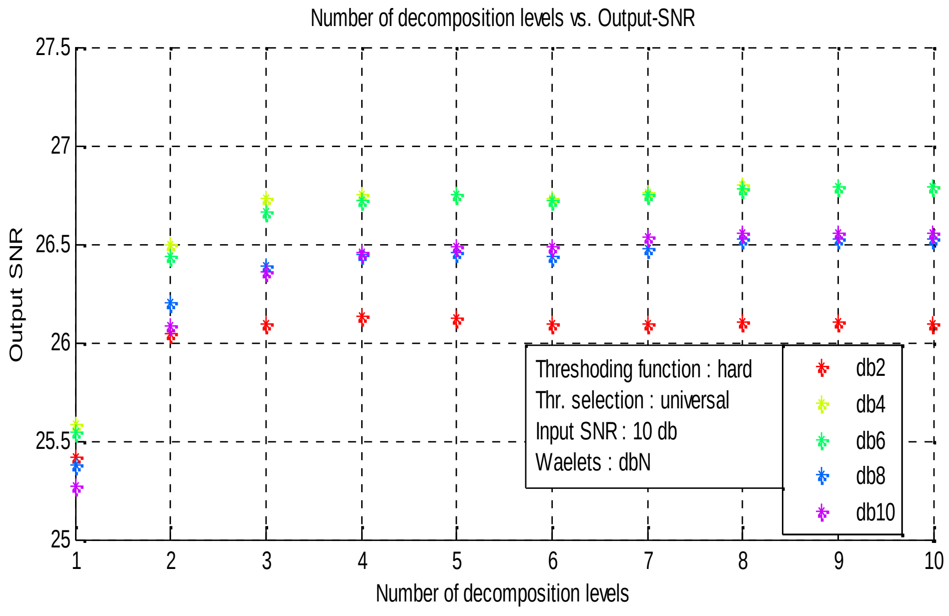


Fig 4.18a :Number of decomposition levels vs. Output SNR using hard thresholding function and universal selection rule

From Fig4.18a, the output SNRs are increased with different levels. Increasing the number of vanishing moments of db wavelet not implies increasing the output SNR for any specific level. Increasing the number of decomposition levels greater than five levels will not introduce a large change for output SNR as in the case of soft thresholding.

Fig 4.18b shows the relation between the input SNR and the output SNR by using global thresholding with different types of db family. The thresholding function that used is soft and the number of decomposition levels is equal to six levels. The clear signal is corrupted by additive white Gaussian noise at different level of signal to noise ratio (0 db, 5db, 10db, 15 db and 20 db. The speech signal was framed using hamming window with 50% percent of overlapping. The frame length is 160 samples, 80 samples of overlapping with any previous frame.

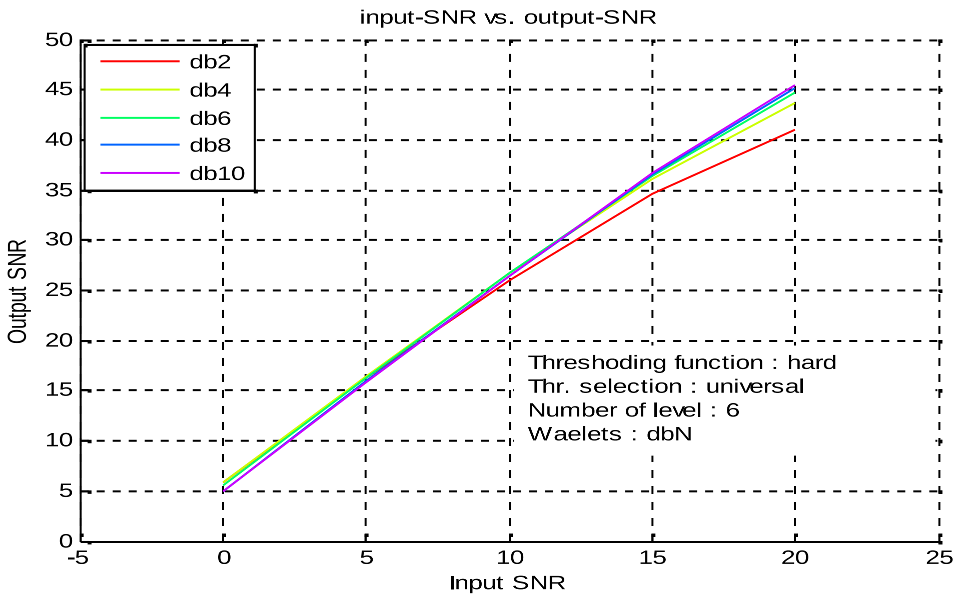


Fig 4.18b :Input SNR vs. Output SNR using hardthresholding function and universal selection rule

From Fig 4.18b,there is an enhancement in the output SNR. Increasing the number of vanishing moments of db wavelet increases the output SNR in the case of high SNR, but as the power of noise increases then, there is no significant change about the output SNR at a specific signal to noise ratio.

Fig 4.19a shows the relation between different threshold values and the output SNR with different levels of decomposition. The speech signal was framed using hamming window with 50% percent of overlapping. The frame length is 160 samples, 80 samples of overlapping with any previous frame. The input SNR is 10 db, the wavelet that used is db8 and the thresholding function is of type soft thresholding.

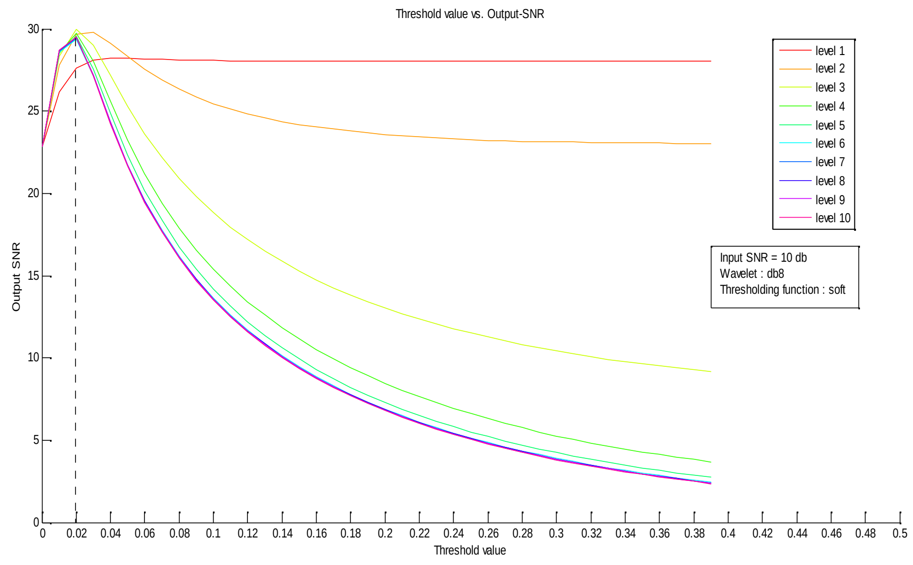


Fig 4.19a : The threshold value vs. output SNR with different levels of decomposition

From Fig 4.19a the output SNR starts to increase as the threshold value increases until reaching a specific threshold value (less than 0.05), after that, the output SNR decreases as the threshold value increases. For example, the maximum output SNR for the sixth level is with threshod value equal to 0.02 which approximately equal to the calculated value using universal thresholding method.

Fig 4.19b shows the same relation as in Fig 4.19a but with different input SNRs. The speech signal was framed using hamming window with 50% percent of overlapping. The frame length is 160 samples, 80 samples of overlapping with any previous frame.The number of decomposition levels is equal to six levels, the wavelet that used is db8 and the thresholding function is of type soft thresholding.

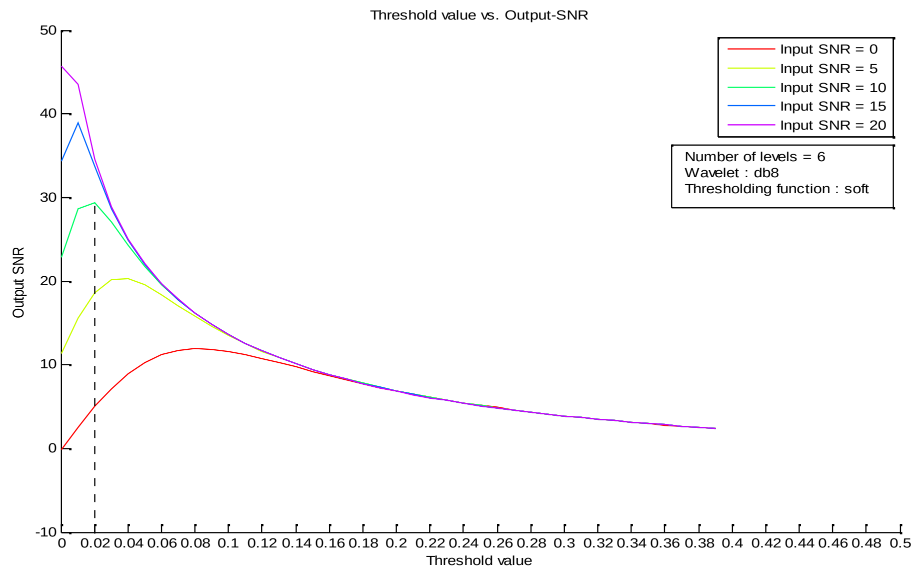


Fig 4.19b : The threshold value vs. output SNR with different input SNRs

4.2.2 Interval-Dependent thresholding method

Fig 4.20a shows the relation between input SNR and output SNR by using interval-dependent thresholding with different number of intervals. The thresholding function that used is soft, the wavelet is db8 and the number of decomposition levels is equal to six levels. The speech signal was framed using hamming window with 50% percent of overlapping. The frame length is 160 samples, 80 samples of overlapping with any previous frame.Every frame is divided into several intervals based on variance changes,then the coefficients of each interval are thresholded using universal thresholding method.

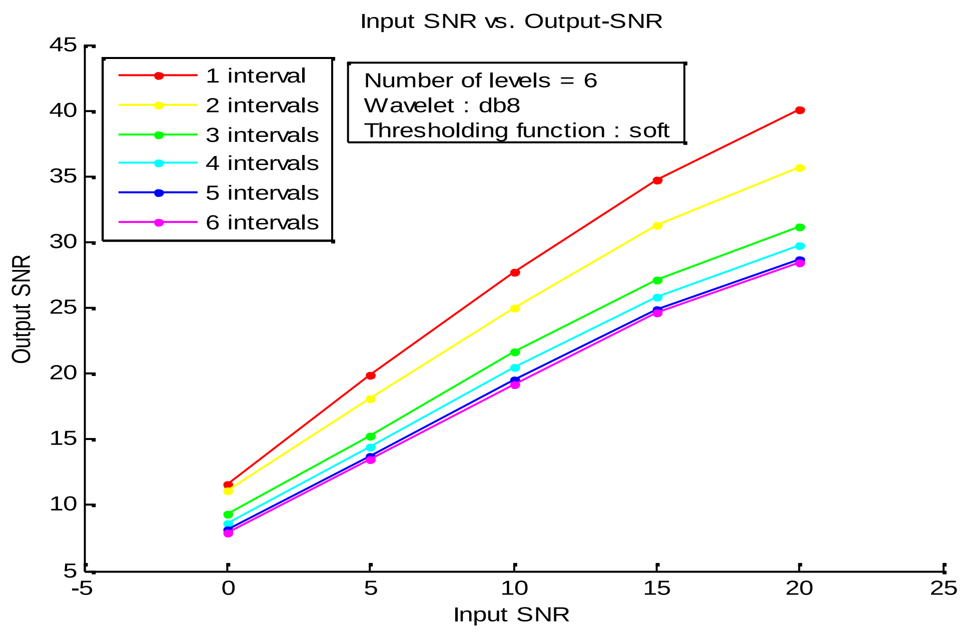


Fig 4.20a : Input SNR vs. Output SNR for interval-dependent thresholding method

From Fig 4.20a, increasing the number of intervals at a specific input SNR will reduce the output SNR. The best number of interval is one, this is because the white noise that we added have a constant variance that does not change with time.

Fig 4.20b shows the relation between the number of decomposition levels and output SNR by using interval-dependent thresholding with different number of intervals. The thresholding function that used is soft, the wavelet is db8 and input SNR is equal to 10 db. The speech signal was framed using hamming window with 50% percent of overlapping. The frame length is 160 samples, 80 samples of overlapping with any previous frame. Every frame is divided into several intervals based on variance changes,then the coefficients of each interval are thresholded using universal thresholding method.

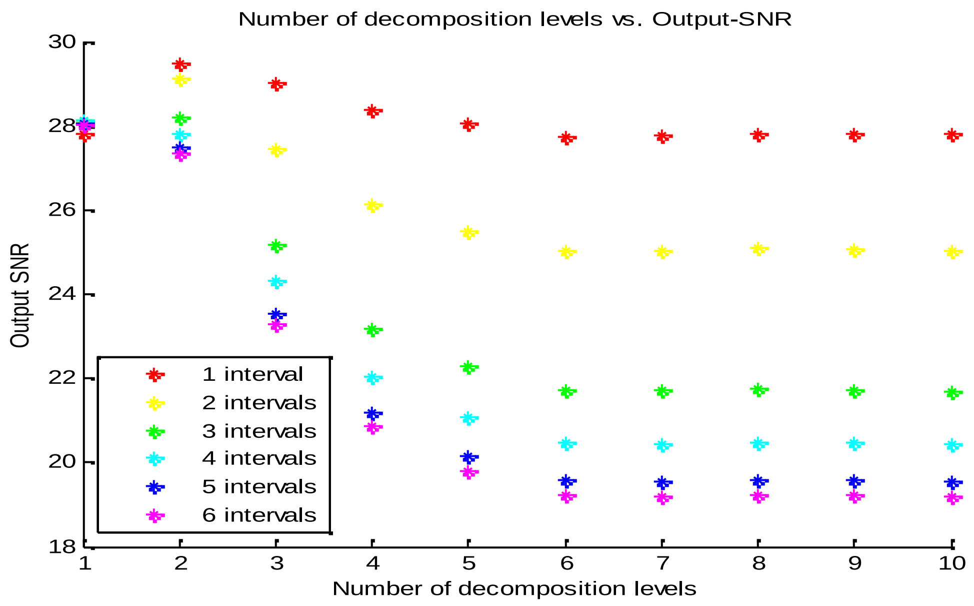


Fig 4.20b : Input SNR vs. Output SNR for interval-dependent thresholding method

From Fig 4.20b, increasing the number of intervals using a specific number of decomposition levels will reduce the output SNR. The best number of interval is one, this is also because the white noise that we added have a constant variance that does not change with time.

4.2.3 Setting all details coefficients in the first scale to zero

In this test only the detail coefficients of the first scale are setted to zero by assuming that most of the noise power is in the first level. Fig 4.21a shows the relation between the input SNR and output SNR with different db wavelets.

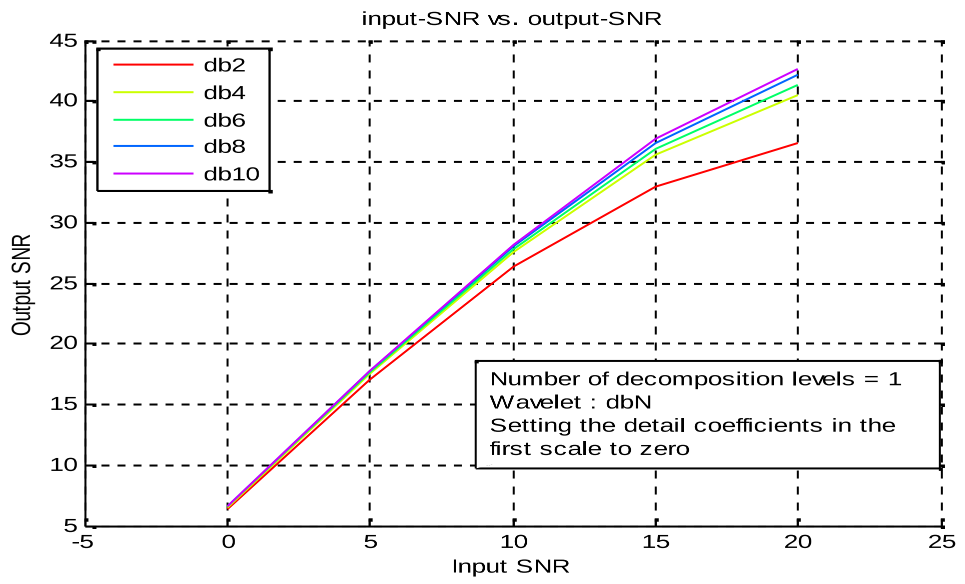


Fig 4.21a : Input SNR vs. Output SNR (Setting all coefficients in the first scale to zero)

Fig 4.21b shows the same relation as in Fig 4.21a but instead of setting the detail coefficients of the first scale to zero, soft thresholding function with universal threshold selection rule is used to threshold the details of first scale.

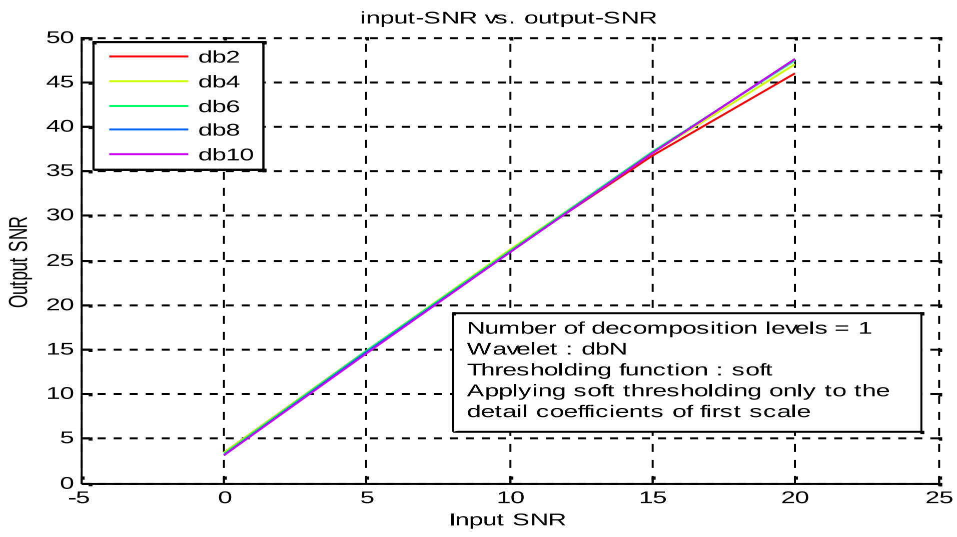


Fig 4.21b : Input SNR vs. Output SNR (Applying soft thresholding on details of first scale only)

From Fig 4.21a and Fig 4.21b, Applying soft thresholding on details of first scale only at a specific input SNR, the output SNR did not changed significantly as the number of vanishing changed. However, in the case of setting the details of first scale to zero, the output SNR is changed significantly as the number of vanishing changed, especially at high input SNR.

4.3 : Comparing the performance with different threshold selection rules

Fig 4.22 shows the relation between the input SNR and the output SNR with different threshold selection criteria (Fixed threshold, SURE, Mix of fixed threshold and SURE, minimaxi). The number of decomposition levels that used is equal to six, the wavelet is db8 and the type of thresholding function is soft.

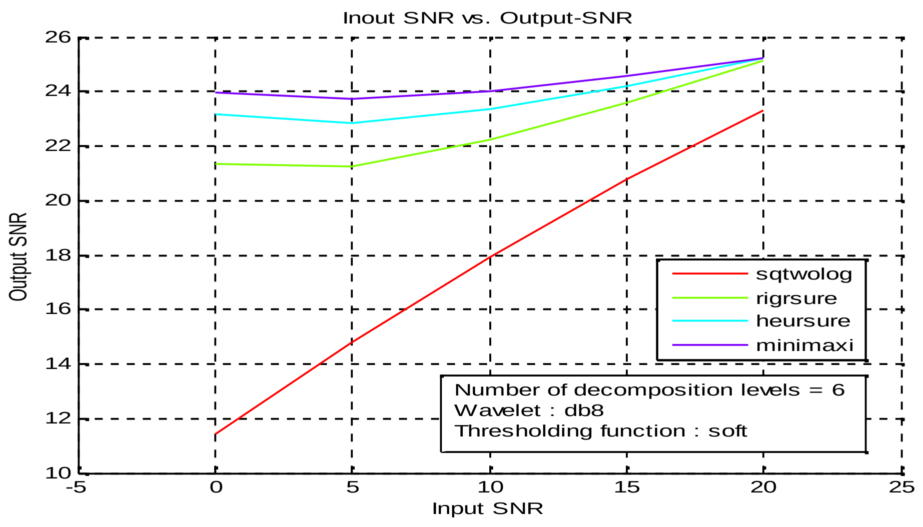


Fig 4.22 : Input SNR vs. Output SNR with different threshold selection criteria

4.4 Comparing the performance with different wavelet families

Fig 4.23a shows the relation between the input SNR and the output SNR with different types of wavelet families. The threshold selection rule is the universal method, the number of levels is six and the thresholding function is soft.

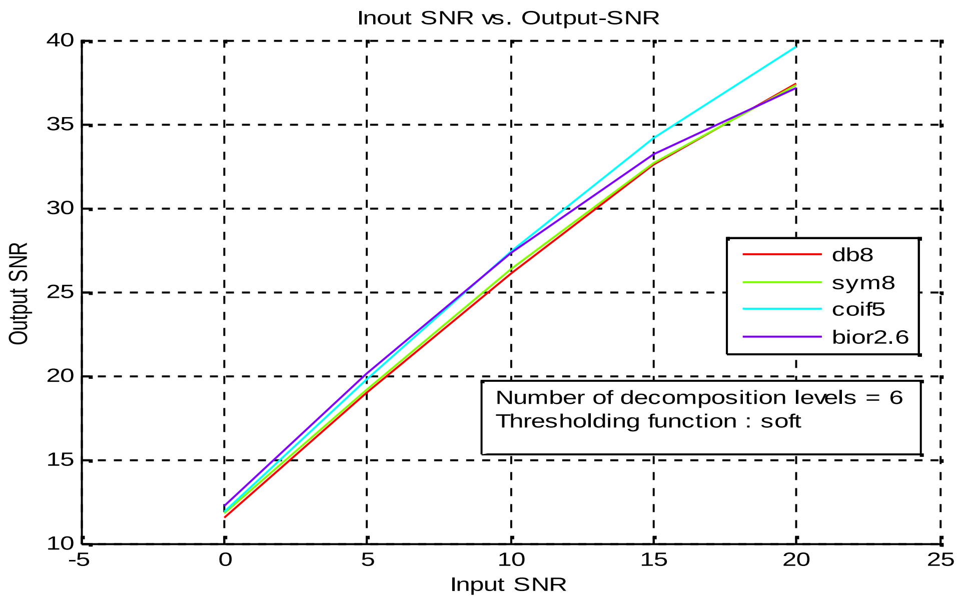


Fig 4.23a : Input SNR vs. Output SNR with different type of wavelet families

From Fig 4.23a, there is approximately only 1 db change with output SNR between coif5 and db8 .This comparison is with using six levels of decomposition. Fig 4.23b shows the relation between the number of decomposition levels and output SNR. The input SNR is 10db and the thresholding function that used is soft.

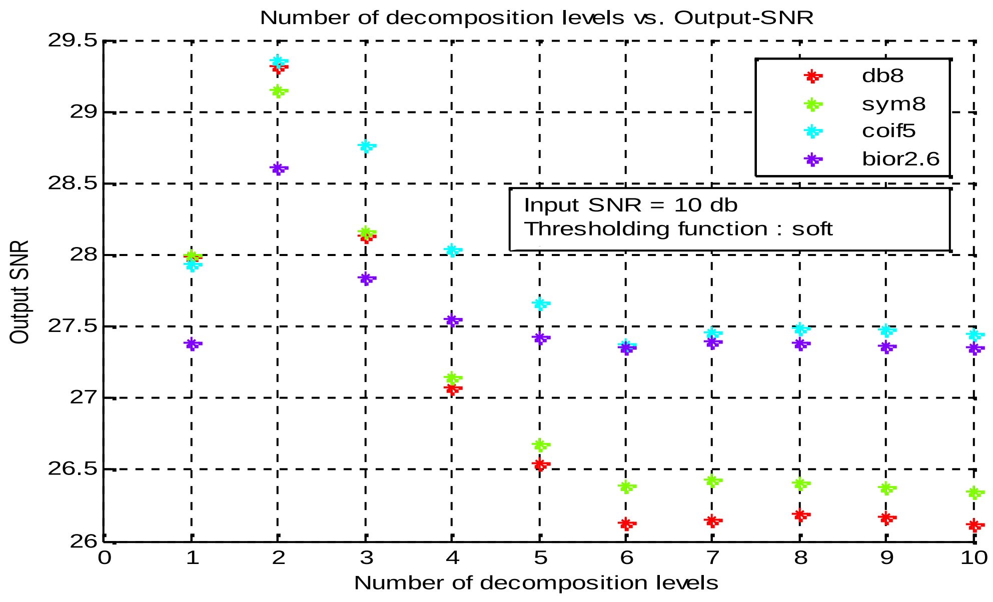


Fig 4.23b : The number of decomposition levels vs. output SNR with different wavelet families

From Fig 4.23b, using coif5 introduced the largest output SNR for all levels. Using two levels of decomposition maximize the global output SNR. Also, increasing the number of decomposition levels greater than five levels will not introduce a significant change in output SNR.

Fig 4.24 shows the relation between the input SNR and the output SNR using two methods of de-noising. The first method is by using DWT and the second is by using Wiener filtering. The wiener filtering is based on noise estimation using wavelet decomposition, so that the variance of the noise is estimated by using median approximation of the detail coefficients in the first scale. Appendix C gives brief discussion about the wiener filtering.

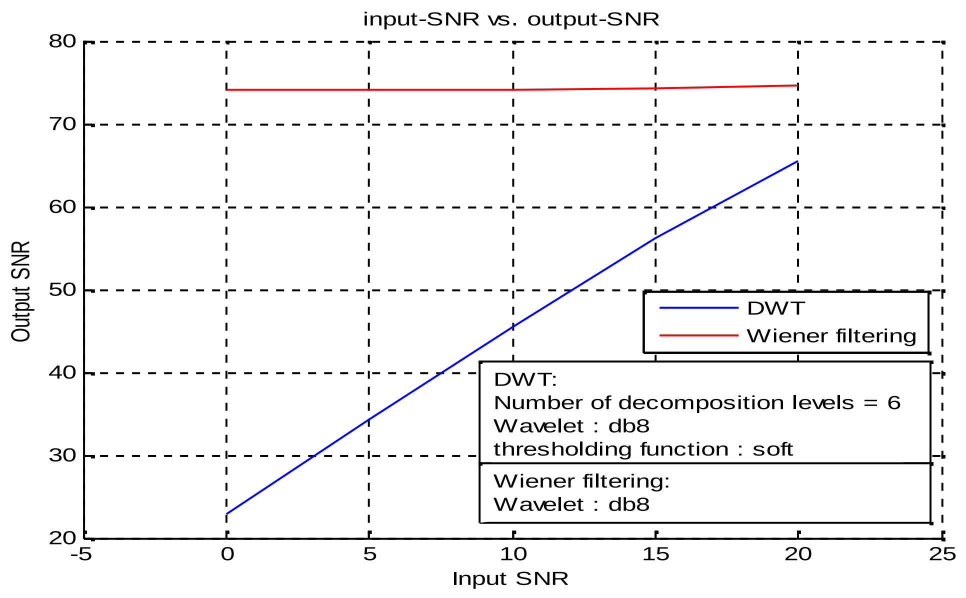


Fig 4.24 : Input SNR vs. Output SNR(comparison between DWT and Wiener Filtering)

From Fig 4.24, the wiener filtering is better than de-noising by DWT. Wiener filtering gives a constant output SNR approximately. However, wiener filtering needs to know the spectral properties of the original signal and the noise, for this purpose the variance of the noise is estimated from details of the first wavelet decomposition level.

Chapter 5
Conclusion and Future works

5.1 Conclusion

5.2 Future works

5.1 Conclusion

Speech de-noising algorithm using discrete wavelet transform is implemented to eliminate a white noise. As shown in this project the selection of threshold value is an important parameter for speech enhancement. Using universal thesholding by fixed threshold applied to threshold the wavelet coefficients introduce an efficient way to remove the additive white Gaussian noise. Interval dependent method is also used to adapt the threshold value, however since the Gaussian noise is stationary and its variance did not change with time this method is more appropriate to non-white noise. Different parameters were changed to get more optimal choice of them. This project concentrates on db wavelets and shows that this kind of wavelet tends to be an appropriate choice for speech enhancement, Specially, under the assumption that the noise is Additive white Gaussian noise. Soft thresholding function is more appropriate for speech de-noising.

As a comparison with other method of de-noising, Wiener filtering based on the wavelet decomposition for noise estimation. In this method the noise is estimated from the first scale of wavelet decomposition and this estimation used to apply wiener filtering. The experiment shows that wiener filtering introduce more enhancement in output SNR compared with de-noising using global thresholding in the discrete wavelet domain.

5.2 Future works

The project concentrated on the additive white Gaussian noise, and this work can be extended to a non-white noise. There are many wavelet families that could be tested for speech enhancement. There are many other variations about thresholding function that could also be tested. There are many other techniques to adapt the threshold value which could be tested. Using other filtering techniques with wavelet de-noising method to get more optimal filtering.

Wavelet Toolbox Functions

Note : any shaded function is used in the Matlab program of this project.

Wavelets and Filter Banks

Real and Complex-Valued Wavelets

`bswfun`	Biorthogonal scaling and wavelet functions
`centfrq`	Wavelet center frequency
`cgauwavf`	Complex Gaussian wavelet
`cmorwavf`	Complex Morlet wavelet
`fbspwavf`	Complex frequency B-spline wavelet
`gauswavf`	Gaussian wavelet
`intwave`	Integrate wavelet function psi (ψ)
`mexihat`	Mexican hat wavelet
`meyer`	Meyer wavelet
`meyeraux`	Meyer wavelet auxiliary function
`morlet`	Morlet wavelet
`scal2frq`	Scale to frequency
`shanwavf`	Complex Shannon wavelet
`wavefun`	Wavelet and scaling functions
`wavefun2`	Wavelet and scaling functions 2-D
`wavsupport`	Wavelet support
`wavemenu`	Wavelet Toolbox GUI tools
`wavemngr`	Wavelet manager
`waveletfamilies`	Wavelet families and family members
`waveinfo`	Wavelets information

Orthogonal and Biorthogonal Filter Banks

`biorwavf`	Biorthogonal spline wavelet filter
`biorfilt`	Biorthogonal wavelet filter set
`coifwavf`	Coiflet wavelet filter
`dddtree`	Dual-tree and double-density 1-D wavelet transform
`dbaux`	Daubechies wavelet filter computation
`dbwavf`	Daubechies wavelet filter
`orthfilt`	Orthogonal wavelet filter set

rbiowavf	Reverse biorthogonal spline wavelet filters
qmf	Scaling and Wavelet Filter
symaux	Symlet wavelet filter computation
symwavf	Symlet wavelet filter
wavefun	Wavelet and scaling functions
wavefun2	Wavelet and scaling functions 2-D
wfilters	Wavelet filters
wrev	Flip vector
wavemenu	Wavelet Toolbox GUI tools
wavemngr	Wavelet manager
waveletfamilies	Wavelet families and family members
waveinfo	Wavelets information
wavenames	Wavelet names for LWT

Lifting

addlift	Add lifting steps to lifting scheme
displs	Display lifting scheme
filt2ls	Transform quadruplet of filters to lifting scheme
laurmat	Laurent matrices constructor
laurpoly	Laurent polynomials constructor
liftfilt	Apply elementary lifting steps on quadruplet of filters
liftwave	Lifting schemes
lsinfo	Lifting schemes information
ls2filt	Transform lifting scheme to quadruplet of filters
wave2lp	Laurent polynomials associated with wavelet
wavemngr	Wavelet manager
waveletfamilies	Wavelet families and family members
waveinfo	Wavelets information
wavenames	Wavelet names for LWT

Wavelet Design

pat2cwav	Build wavelet from pattern
wavemenu	Wavelet Toolbox GUI tools
wavemngr	Wavelet manager

addlift	Add lifting steps to lifting scheme
displs	Display lifting scheme
filt2ls	Transform quadruplet of filters to lifting scheme
laurmat	Laurent matrices constructor
laurpoly	Laurent polynomials constructor
liftfilt	Apply elementary lifting steps on quadruplet of filters
liftwave	Lifting schemes
lsinfo	Lifting schemes information
ls2filt	Transform lifting scheme to quadruplet of filters
wave2lp	Laurent polynomials associated with wavelet

Continuous Wavelet Analysis

conofinf	Cone of influence
cwt	Continuous 1-D wavelet transform
cwtext	Real or complex continuous 1-D wavelet coefficients using extensi
cwtft	Continuous wavelet transform using FFT algorithm
cwtftinfo	Valid analyzing wavelets for FFT-based CWT
cwtftinfo2	Supported 2-D CWT wavelets and Fourier transforms
cwtft2	2-D continuous wavelet transform
icwtft	Inverse CWT
icwtlin	Inverse continuous wavelet transform (CWT) for linearly spaced sc
localmax	Identify and chain local maxima
pat2cwav	Build wavelet from pattern
wcoher	Wavelet coherence
wscalogram	Scalogram for continuous wavelet transform
wavemenu	Wavelet Toolbox GUI tools
wavemngr	Wavelet manager

Discrete Wavelet Analysis

Signal Analysis

dyaddown	Dyadic downsampling
dyadup	Dyadic upsampling

`upcoef`	Direct reconstruction from 1-D wavelet coefficients
`appcoef`	1-D approximation coefficients
`detcoef`	1-D detail coefficients
`wrcoef`	Reconstruct single branch from 1-D wavelet coefficients
`dwt`	Single-level discrete 1-D wavelet transform
`dwtmode`	Discrete wavelet transform extension mode
`idwt`	Single-level inverse discrete 1-D wavelet transform
`waverec`	Multilevel 1-D wavelet reconstruction
`wavedec`	Multilevel 1-D wavelet decomposition
`upwlev`	Single-level reconstruction of 1-D wavelet decomposition
`lwt`	1-D lifting wavelet transform
`lwtcoef`	Extract or reconstruct 1-D LWT wavelet coefficients
`ilwt`	Inverse 1-D lifting wavelet transform
`swt`	Discrete stationary wavelet transform 1-D
`iswt`	Inverse discrete stationary wavelet transform 1-D
`dddtree`	Dual-tree and double-density 1-D wavelet transform
`dddtreecfs`	Extract dual-tree/double-density wavelet coefficients or projections
`dtfilters`	Analysis and synthesis filters for oversampled wavelet filter banks
`idddtree`	Inverse dual-tree and double-density 1-D wavelet transform
`plotdt`	Plot dual-tree or double-density wavelet transform
`wenergy`	Energy for 1-D wavelet or wavelet packet decomposition
`wvarchg`	Find variance change points
`wmaxlev`	Maximum wavelet decomposition level
`wfbm`	Fractional Brownian motion synthesis
`wfbmesti`	Parameter estimation of fractional Brownian motion
`measerr`	Approximation quality metrics
`wrev`	Flip vector
`wextend`	Extend vector or matrix
`wkeep`	Keep part of vector or matrix
`wavemenu`	Wavelet Toolbox GUI tools
`wavemngr`	Wavelet manager

Image Analysis

`dyaddown`	Dyadic downsampling
`dyadup`	Dyadic upsampling
`upcoef2`	Direct reconstruction from 2-D wavelet coefficients
`appcoef2`	2-D approximation coefficients
`detcoef2`	2-D detail coefficients
`dwt2`	Single-level discrete 2-D wavelet transform
`dwtmode`	Discrete wavelet transform extension mode
`idwt2`	Single-level inverse discrete 2-D wavelet transform
`wavedec2`	Multilevel 2-D wavelet decomposition
`waverec2`	Multilevel 2-D wavelet reconstruction
`wrcoef2`	Reconstruct single branch from 2-D wavelet coefficients
`upwlev2`	Single-level reconstruction of 2-D wavelet decomposition
`wenergy2`	Energy for 2-D wavelet decomposition
`ilwt2`	Inverse 2-D lifting wavelet transform
`iswt2`	Inverse discrete stationary wavelet transform 2-D
`lwt2`	2-D lifting wavelet transform
`lwtcoef2`	Extract or reconstruct 2-D LWT wavelet coefficients
`swt2`	Discrete stationary wavelet transform 2-D
`iswt2`	Inverse discrete stationary wavelet transform 2-D
`dddtreecfs`	Extract dual-tree/double-density wavelet coefficients or projections
`dddtree2`	Dual-tree and double-density 2-D wavelet transform
`dtfilters`	Analysis and synthesis filters for oversampled wavelet filter banks
`idddtree2`	Inverse dual-tree and double-density 2-D wavelet transform
`plotdt`	Plot dual-tree or double-density wavelet transform
`wcodemat`	Extended pseudocolor matrix scaling
`wfusimg`	Fusion of two images
`wfusmat`	Fusion of two matrices or arrayz
`measerr`	Approximation quality metrics
`wextend`	Extend vector or matrix
`wkeep`	Keep part of vector or matrix
`wavemenu`	Wavelet Toolbox GUI tools
`wavemngr`	Wavelet manager

3-D Analysis

dwt3	Single-level discrete 3-D wavelet transform
dwtmode	Discrete wavelet transform extension mode
idwt3	Single-level inverse discrete 3-D wavelet transform
wavedec3	Multilevel 3-D wavelet decomposition
waverec3	Multilevel 3-D wavelet reconstruction
wavemenu	Wavelet Toolbox GUI tools
wavemngr	Wavelet manager

Multisignal Analysis

chgwdeccfs	Change multisignal 1-D decomposition coefficients
dwtmode	Discrete wavelet transform extension mode
mdwtcluster	Multisignals 1-D clustering
mdwtdec	Multisignal 1-D wavelet decomposition
mdwtrec	Multisignal 1-D wavelet reconstruction
mswcmp	Multisignal 1-D compression using wavelets
mswcmpscr	Multisignal 1-D wavelet compression scores
mswcmptp	Multisignal 1-D compression thresholds and performances
mswden	Multisignal 1-D denoising using wavelets
mswthresh	Perform multisignal 1-D thresholding
wavemenu	Wavelet Toolbox GUI tools
wdecenergy	Multisignal 1-D decomposition energy distribution
wmspca	Multiscale Principal Component Analysis
wextend	Extend vector or matrix
wkeep	Keep part of vector or matrix
wavemenu	Wavelet Toolbox GUI tools

Wavelet Packet Analysis

dwtmode	Discrete wavelet transform extension mode
wpdec	Wavelet packet decomposition 1-D
wpdec2	Wavelet packet decomposition 2-D
wprec	Wavelet packet reconstruction 1-D
wprec2	Wavelet packet reconstruction 2-D

wpcoef	Wavelet packet coefficients
wprcoef	Reconstruct wavelet packet coefficients
bestlevt	Best level tree wavelet packet analysis
besttree	Best tree wavelet packet analysis
entrupd	Entropy update (wavelet packet)
wentropy	Entropy (wavelet packet)
plot	Plot tree GUI
wpviewcf	Plot wavelet packets colored coefficients
wavemenu	Wavelet Toolbox GUI tools
wpfun	Wavelet packet functions
wpspectrum	Wavelet packet spectrum
cfs2wpt	Wavelet packet tree construction from coefficients
depo2ind	Node depth-position to node index
wp2wtree	Extract wavelet tree from wavelet packet tree
wpcutree	Cut wavelet packet tree
wpsplt	Split (decompose) wavelet packet
wpjoin	Recompose wavelet packet
ind2depo	Node index to node depth-position
otnodes	Order terminal nodes of binary wavelet packet tree
write	Write values in WPTREE fields
read	Read values of WPTREE
readtree	Read wavelet packet decomposition tree from figure
set	WPTREE field contents
tnodes	Determine terminal nodes
wptree	WPTREE constructor
disp	WPTREE information
drawtree	Draw wavelet packet decomposition tree (GUI)
dtree	DTREE constructor
allnodes	Tree nodes
get	WPTREE contents
isnode	Existing node test
istnode	Terminal nodes indices test

leaves	Determine terminal nodes
nodeasc	Node ascendants
nodedesc	Node descendants
nodejoin	Recompose node
nodepar	Node parent
nodesplt	Split (decompose) node
noleaves	Determine nonterminal nodes
ntnode	Number of terminal nodes
ntree	NTREE constructor
treedpth	Tree depth
treeord	Tree order
wtbo	WTBO constructor
wtreemgr	NTREE manager

Denoising

cmddenoise	Interval-dependent denoising
ddencmp	Default values for denoising or compression
thselect	Threshold selection for de-noising
wbmpen	Penalized threshold for wavelet 1-D or 2-D de-noising
wdcbm	Thresholds for wavelet 1-D using Birgé-Massart strategy
wdcbm2	Thresholds for wavelet 2-D using Birgé-Massart strategy
wden	Automatic 1-D de-noising
wdencmp	De-noising or compression
wmulden	Wavelet multivariate de-noising
wnoise	Noisy wavelet test data
wnoisest	Estimate noise of 1-D wavelet coefficients
wpbmpen	Penalized threshold for wavelet packet de-noising
wpdencmp	De-noising or compression using wavelet packets
wpthcoef	Wavelet packet coefficients thresholding
wthcoef	1-D wavelet coefficient thresholding
wthcoef2	Wavelet coefficient thresholding 2-D
wthresh	Soft or hard thresholding

`wthrmngr`	Threshold settings manager
`wvarchg`	Find variance change points
`measerr`	Approximation quality metrics
`wavemenu`	Wavelet Toolbox GUI tools

Compression

`wcompress`	True compression of images using wavelets
`wpdencmp`	De-noising or compression using wavelet packets
`measerr`	Approximation quality metrics
`wmpalg`	Matching pursuit
`wmpdictionary`	Dictionary for matching pursuit
`wavemenu`	Wavelet Toolbox GUI tools

Reference :

https://www.mathworks.com/help/wavelet/index.html

Appendix B
Matlab code

```
clc
clear all
close all
%% Stage 1:
%   1.1 - Reading speech signal.
%   1.2 - Adding Additive White Gaussian Noise.
%   1.3 - Plotting  clear signal and noisy signal.
%   1.4 - Playing clear signal and noisy signal.

%----------
%1.1 - Reading speech signal.
% option 1 : Get recorded file from PC (.wav OR .mat).
% option 2 : Online recording via microphone.
c = menu('speech choice options :','Get recorded file
from PC (clear signal or noisy signal)','Online recording
via microphone');
switch c
%----------
% option 1 : Get recorded file from PC
    case 1
d = dir;
strn = {d.name};
[s,v] = listdlg('PromptString' , 'Select a file : ' ,
'SelectionMode' , 'single' , 'ListString' , strn);
[ a , b] = strread(strn{s} , '%s %s' , 'delimiter' ,
'.');
file_str_name = strn{s};

if strcmp(b , 'mat')
    n_c_mat = menu('Is this signal noisy or
clear','Clear','Noisy');
switch n_c_mat
    case 1
        ref0 = 0;ref1= 0;
        signal_mat = load(file_str_name);
        FieldName = fieldnames(signal_mat);
        Field_speech_full =
getfield(signal_mat,FieldName{1});
        N_samples = length(Field_speech_full);
        N_str = num2str(N_samples);
        prompt = {strcat('Enter the number of first
N_samples of signal : N_samples <= ',N_str),'Enter the
frequency of sampling','Enter the number of bits per
sample','Enter the value of signal to noise ratio'};
        dlg_title = 'Audio File Selection';
        num_lin = 1;
        def_filename_snr = {'1,32000','8000','16','10'};
        file_name = inputdlg(prompt ,dlg_title ,num_lin
,def_filename_snr);
        [N1 , N2] = strread(file_name{1} , '%s %s' ,
'delimiter' , ',');
        N1 = str2num(N1{1});
        N2 = str2num(N2{1});
        len_seg = N2 - N1 + 1;
        Fs = str2num(file_name{2});
        nbits = str2num(file_name{3});
        snrval = str2num(file_name{4});

        clear_speech = Field_speech_full(N1:N2);%clear
signal
    case 2
        ref0 = 0;ref1= 1;
        signal_mat = load(file_str_name);
        FieldName = fieldnames(signal_mat);
        Field_speech_full =
getfield(signal_mat,FieldName{1});
        N_samples = length(Field_speech_full);
        N_str = num2str(N_samples);
        prompt = {strcat('Enter the number of first
N_samples of signal : N_samples <= ',N_str),'Enter the
frequency of sampling','Enter the number of bits per
sample'};
        dlg_title = 'Audio File Selection';
        num_lin = 1;
        def_filename_snr = {'1,32000','8000','16'};
        file_name = inputdlg(prompt ,dlg_title ,num_lin
,def_filename_snr);
        [N1 , N2] = strread(file_name{1} , '%s %s' ,
'delimiter' , ',');
        N1 = str2num(N1{1});
        N2 = str2num(N2{1});
        len_seg = N2 - N1 + 1;
        Fs = str2num(file_name{2});
        nbits = str2num(file_name{3});
        noisy_speech = Field_speech_full(N1:N2)';

        load h_orig.mat;% clear signal
        clear_speech = h_orig(N1:N2)';
        snrval = 10;
end

elseif strcmp(b , 'wav')
    n_c_wav = menu('Is this signal noisy or
clear','Clear','Noisy' );
    switch n_c_wav
        case 1
            ref0 = 1;ref1= 0
            [clear_speech_all , Fs , nbits] = 
wavread(file_str_name);
            N_samples = length(clear_speech_all);
            N_str = num2str(N_samples);
            prompt = {strcat('Enter two numbers N1 and N2 
separated by commas\n N1 < N2 <= ',N_str),'Enter the 
value of signal to noise ratio'};
            dlg_title = 'Signal Interval and SNR values';
            num_lin = 1;
            def_filename_snr = {'1,8000','10'};
            file_name = inputdlg(prompt ,dlg_title 
,num_lin ,def_filename_snr);
            snrval = str2num(file_name{2});
            firstNsamp = str2num(file_name{1});
            [N1 , N2] = strread(file_name{1} , '%s %s' , 
'delimiter' , ',');
            N1 = str2num(N1{1});
            N2 = str2num(N2{1});
            len_seg = N2 - N1 + 1;
            snrval = str2num(file_name{2});
            clear_speech = clear_speech_all(N1:N2);
        case 2
            ref0 = 1;ref1 = 1;
            [noisy_speech_all , Fs , nbits] = 
wavread(file_str_name);
            N_samples = length(noisy_speech_all);
            N_str = num2str(N_samples);
            prompt = {strcat('Enter two numbers N1 and N2 
separated by commas\n N1 < N2 <= ',N_str),};
            dlg_title = 'Signal Interval';
            num_lin = 1;
            def_filename_snr = {'1,32000'};
            file_name = inputdlg(prompt ,dlg_title 
,num_lin ,def_filename_snr);
            firstNsamp = str2num(file_name{1});
            [N1 , N2] = strread(file_name{1} , '%s %s' , 
'delimiter' , ',');
            N1 = str2num(N1{1});
            N2 = str2num(N2{1});
            len_seg = N2 - N1 + 1;
            noisy_speech = noisy_speech_all(N1:N2);
            load h_orig.mat;
            clear_speech = h_orig;%clear signal
            snrval = 5;
    end

else
    msg = msgbox('Extension of the file must be .wav or 
.mat');
end
```

```
if (size(clear_speech,1) == 1)
    clear_speech = clear_speech';
end

%----------
%1.2 - Adding Aditive White Gaussian Noise
if (ref0 == 0 && ref1 == 0) ||(ref0 == 1 && ref1 == 0)
noise_generator = menu('noise generator','Add white
gaussian noise to signal' , 'Add normally distributed
pseduorandom numbers to signal');
if ~(noise_generator-1)
noisy_speech = awgn(clear_speech,snrval,'measured');%Add
AWGN
else
snr_lin = 10^(snrval/10);
power_signal = mean(abs(clear_speech).^2);
var = power_signal/snr_lin;
noise = (randn(length(clear_speech),1).*sqrt(var));
noisy_speech = clear_speech + noise; % Add I.I.D AWGN
end
end
% Test the vector dimensions agreement.
        if(size(clear_speech,1)-size(noisy_speech,1))~=0
            if size(noisy_speech,1)==1
                noisy_speech = noisy_speech';
            else
                clear_speech = clear_speech';
            end
        end

save E:\Noisy_File\noisy_speech.mat
noisyfile = 'E:\Noisy_File\noisy_speech.wav';
wavwrite(noisy_speech, Fs ,nbits, noisyfile)% Write the
noisy signal into noisyfile


%----------
%1.3 - Plotting  clear signal and noisy signal

    %----------
    fig1 = figure('name' ,'clear and noisy speech
signals','Color','w');
    %----------

subplot(211); axis tight;
plot([1:length(clear_speech)]/Fs , clear_speech ,'b');
xlabel('Time(s)'); ylabel('Amplitude');
title('Clear speech signal');
subplot(212); axis tight;
plot([1:length(noisy_speech)]/Fs , noisy_speech,'r');
xlabel('Time(s)'); ylabel('Amplitude');
```

```
title('Noisy speech signal');
%----------
%1.4 - Play clean and noisy speech signal
Pause_t = (length(clear_speech)/Fs) + 2 ;
sound(clear_speech,Fs)
pause(Pause_t);
sound(noisy_speech , Fs)
%----------
% option 2 : Online recording via microphone
    case 2
        prompt = {'Enter the number of second to be
recorded','Enter the frequency sampling rate'};
        dlg_title = 'Audio Recording';
        num_lin = 2;
        def_t_f = {'5','8000'};
        record_par = inputdlg(prompt ,dlg_title ,num_lin
,def_t_f);
        record_t = str2num(record_par{1})
        Fs = str2num(record_par{2})
        record_speechObj = audiorecorder(Fs , 16 , 1);
        msg0 = msgbox('Start speaking');
        h = waitbar(0,'Start speaking')
        for step=1:1
        recordblocking(record_speechObj , record_t);
        waitbar(step)
        end
        msg1 = msgbox('End speaking');
        pause(2);
        close(h)
        noisy_speech = getaudiodata(record_speechObj ,
'double');

        %----------
        fig2 = figure('name' , 'noisy recorded
speech','Color','w');
        %----------
        plot([1:length(noisy_speech)]/Fs ,
noisy_speech);% Plotting the noisy recorded signal
        play(record_speechObj); % Play the noisy recorded
speech signal
end

%----------

%% Stage 2:
%   2.1 - Choose a)DWT or b)DPT
%   In the case of DWT :
%   2.2.a - Set the number of level decomposition and the
wavelet function name
%   2.3.a - Find the wavelet and scaling functions
%   2.4.a - Find the wavelet and scaling filters
%   2.5.a - Decompose the noisy signal at a given level
using the wavelet filters
%   2.6.a - Extract the approximation coefficients at
coarser scale(final level) from
%   the wavelet decomposition structure [Cad , L]
%   2.7.a - Extract the detail coefficients at the levels
(1 , 2 , ... , level)
%   from the wavelet decomposition structure [Cad , L]
%   2.8.a - Find the energy of the wavelet coefficients
%   2.9.a - Reconstruct the approximation signal at
coarser scale(final level)from wavelet decomposition
%   2.10.a - Reconstruct the detail signals at all levels
from the wavelet decomposition
%   structure [Cad , L]
%   2.11.a - Plotting illustration

%----------
%2.1 - Choose a)DWT or b)DPT
c1 = menu('Denoising using :' ,'Discrete Wavelet
Transform (DWT)','Discrete Wavelet packet (DWP)');
switch c1
    case 1% DWT
%----------
%2.2.a - Set the number of level decomposition and the
wavelet function name
lev_wname = inputdlg({'Enter the number of decomposition
levels','Enter the wave name : dbN'},'Number of level and
wavename');
level = str2num(lev_wname{1});
wavename = lev_wname{2};
%----------
%2.3.a - Find the wavelet and scaling functions
iteration = 15;
[phi , psi , xval_dbn] = wavefun(wavename , iteration);
%----------
%2.4.a - Find the wavelet and scaling filters
[Lo_d , Lo_r] = wfilters(wavename , 'l')
Hi_r = qmf(Lo_r)
Hi_d = wrev(Hi_r)

sum_Hi_d = sum(Hi_d)
sum_Lo_d = sum(Lo_d)

Nextpow2_Lo_d = nextpow2(length(Lo_d));
Nextpow2_Hi_d = nextpow2(length(Lo_d));
Nextpow2_Lo_r = nextpow2(length(Lo_r));
Nextpow2_Hi_r = nextpow2(length(Lo_r));

fftLo_d = fft(Lo_d,2^Nextpow2_Lo_d);
fftHi_d= fft(Hi_d,2^Nextpow2_Hi_d);
fftLo_r = fft(Lo_r,2^Nextpow2_Lo_r);
```

```
fftHi_r= fft(Hi_r,2^Nextpow2_Hi_r);

%----------
%2.5.a - Decompose the noisy signal at a given level
using the wavelet filters
[Cad , L] = wavedec(noisy_speech(:,1) ,level , wavename);
%----------
%2.6.a - Extract the approximation coefficients
%2.7.a - Extract the detail coefficients
%2.8.a - Find the energy of the wavelet coefficients
Capp = appcoef(Cad , L , Lo_d , Hi_d, level);%Extract the
approximation coefficients for level (level)
a2sq = sum(Capp.^2);%energy of the wavelet approximation
coefficients
Energy_coef = zeros(1,level+1);
Energy_coef(1) = a2sq;
SDEV = zeros(1,level);
SDEV_COEF = zeros(1,level);
for i= level : -1 : 1
    Cdet = detcoef(Cad , L , i);% Extract the detail
coefficients for level (i)
    d2sq = sum(Cdet.^2);
    Array_det_coef{level-i+1}  = Cdet;
    d2sq = sum(Array_det_coef{level-i+1}.^2);%energy of
the wavelet coefficients for every scale
    Energy_coef(level-i+2)= d2sq;
    SDEV(i) = wnoisest(Cad , L ,i);% standard deviation
appriximation for detail coefficients for every scale
    SDEV_COEF(i) = std(Cdet);
end
engcoef = sum(Energy_coef)% total energy of wavelet
coefficients(approximation & detail)
engsig = sum(noisy_speech(:,1).^2)% total energy of the
noisy signal
engerr = abs(engcoef - engsig)% energy preservation
percent_energy_app =
(Energy_coef(1)/sum(Energy_coef))*100
%----------
%2.9.a - Reconstruct the approximation signal at coarser
scale(final level)from wavelet decomposition
App_sig = wrcoef('a' , Cad , L , wavename , level);
ReconstructArray_sig{1} = App_sig;
%----------
%2.10.a - Reconstruct the detail signals at all levels
from the wavelet decomposition
SDEV_REC = zeros(1,level);
for j = level : -1 : 1
    Det_sig  = wrcoef('d' , Cad , L , wavename , j);
    ReconstructArray_sig{level-j+2} = Det_sig;
    SDEV_REC(j) = std(Det_sig);
end
```

```
%----------
%2.11.a - Plotting illustration

    %----------
    fig3 = figure('name' , 'phi_psi functions ,filters
and fft of filters','Color','w');
    %----------

subplot(521); axis tight ;
plot(xval_dbn, phi);title('Scaling function phi');
subplot(522); axis tight ;
plot(xval_dbn, psi);title('Wavelet function psi');

subplot(523); axis tight ;
stem(Lo_d,'r');title('Decomposition low pass filter');
subplot(524); axis tight ;
stem(Hi_d,'r');title('Decomposition high pass filter');
subplot(525); axis tight ;
stem(Lo_r,'b');title('Reconstruction low pass filter');
subplot(526); axis tight ;
stem(Hi_r,'b');title('Reconstruction high pass filter');

subplot(527); axis tight
Freq_Lo_d = (2*pi)/(2^Nextpow2_Lo_d)
:(2*pi)/(2^Nextpow2_Lo_d) : pi ;
fa_ld = abs(fftLo_d(1:(2^(Nextpow2_Lo_d)/2)));
plot(Freq_Lo_d,fa_ld);title('FFT of analysis low pass
filter');
xlim([(2*pi)/(2^Nextpow2_Lo_d) , pi+0.5]);

subplot(528); axis tight
Freq_Lo_r = (2*pi)/(2^Nextpow2_Lo_r)
:(2*pi)/(2^Nextpow2_Lo_r) : pi ;
fa_hr = abs(fftLo_r(1:(2^(Nextpow2_Lo_r)/2)));
plot(Freq_Lo_r,fa_hr);title('FFT of synthesis low pass
filter');
xlim([(2*pi)/(2^Nextpow2_Lo_r) , pi+0.5]);

subplot(529); axis tight
Freq_Hi_d = (2*pi)/(2^Nextpow2_Hi_d)
:(2*pi)/(2^Nextpow2_Hi_d) : pi ;
fa_ld = abs(fftHi_d(1:((2^Nextpow2_Hi_d)/2)));
plot(Freq_Hi_d,fa_ld);title('FFT of analysis high pass
filter');
xlim([(2*pi)/(2^Nextpow2_Hi_d) , pi+0.5]);

subplot(5,2,10); axis tight
Freq_Hi_r = (2*pi)/(2^Nextpow2_Hi_r)
:(2*pi)/(2^Nextpow2_Hi_r) : pi ;
fa_hr = abs(fftHi_r(1:((2^Nextpow2_Hi_r)/2)));
```

```
plot(Freq_Hi_r,fa_hr);title('FFT of synthesis high pass
filter');
xlim([(2*pi)/(2^Nextpow2_Hi_r) , pi+0.5]);


    %----------
    fig4 = figure('name' , 'decomposition
coefficients','Color','w');
    %----------

subplot(level+2,1,1); axis tight;
plot([1:length(noisy_speech)]/Fs , noisy_speech,'r');
title('Noisy speech signal');
subplot(level+2,1,2); axis tight;
plot(Capp , 'b')
title(['Approximation coefficients at level '
,num2str(level)]);
ylabel(['Ca' , num2str(level)], 'Color' , 'b');
for f = level : -1 : 1
    row = level+2;
    no_fig = level-f+3;
    s = level-f+1;
    lbl = num2str(f);
subplot(row,1,no_fig); axis tight;
plot(Array_det_coef{s} , 'g')
title(['Detail coefficients at level ' ,lbl]);
ylabel(['Cd',lbl],'Color' , 'g')
end
subplot(row,1,row)

    %----------
    fig5 = figure('name' , 'reconstructed
coefficients','Color','w') ;
    %----------

subplot(level+2,1,1); axis tight;
plot([1:length(noisy_speech)]/Fs , noisy_speech,'r');
title(['Noisy speech signal : a',num2str(level),' +
d',num2str(level),' = d',num2str(level-1)]);
subplot(level+2,1,2); axis tight;
plot( ReconstructArray_sig{1} , 'b')
title(['Approximation signal at level '
,num2str(level)]);
ylabel(['a' ,num2str(level)],'Color' , 'b')
for r = level : -1 : 1
    row = level+2;
    no_fig = level-r+3;
    s = level-r+1;
    lbl = num2str(r);
subplot(row,1,no_fig); axis tight;
plot(ReconstructArray_sig{level-r+2} , 'g')
```

```
title(['Reconstructed coefficients at level ' ,lbl]);
ylabel(['d' , lbl],'Color' , 'g')
end
subplot(row,1,row);
xlabel('Time');

    %----------
    fig6 = figure('name' , 'energy and standard
deviation','Color','w');
    %----------

subplot(211) , axis tight;
stem(Energy_coef(2:end),'g');
hold on
stem(Energy_coef(1),'b');
title('Energy of coefficients at different scale');
xlabel('level number'); ylabel('energy of coefficients');
legend('detail coefficients energy','approximation
coefficients energy')
subplot(212); axis tight;
stem(wrev(SDEV.^2),'r'); title('Variance of details at
different scales');
xlabel('level number'); ylabel('Variance of details at
different scales');

tilefigs
%----------

%% Stage 3:
%   3.1.a - Choose the type of thresholding function (
soft or hard)
%           s : soft thresholding  or  h : hard
thresholding
%   3.2.a - Decide whether you need to threshod the
approximation coefficient or not
%           KeepApp = 1 : withno threshold approx.
coeff's  or  KeepApp = 0 : with threshold approx. coeff's
%   3.3.a - Set the value of threshold (this value for
global thresholding)
%   The above three steps could also be choosen by
default option instead of manual option.
%   3.4.a - Choose the type of thresholding
%           3.4.a.1 - gbl : global thresholding
%           3.4.a.2 - lvd : level dependent thresholding
%               3.4.a.2.1 - Choose the noise model
%               3.4.a.2.2 - Chooose the threshold
selection rule
%               3.4.a.2.3 - Calculate the threshold
vector which contains
%                           the threshold values for
every level
```

```
%           3.4.a.3 selected level thresholding
%               3.4.a.3.1 - Choose selcted level
thresholding methology
%               ( set all coeff's of selected level to
zero or threshod the selected level by (s or h) )
%               3.4.a.3.2 - Decide the level numbers to
be thresholded
%           3.4.a.4 Interval dependent thresholding
%   4.4.a - Reconstruct the signal from thresholded
coefficients
%   4.5.a - Reconstruct the approximation and detail
signals from wavelet thresholded decomposition
%   4.6.a - Plotting illustration
%   4.7.a - Playing the denoised speech

%----------
%3.1.a - Choose the type of thresholding function ( soft
or hard)
%3.2.a - Decide whether you need to threshod the
approximation coefficient or not
%3.3.a - Set the value of threshold (this value for
global thresholding)
%The above three steps could also be choosen by default
option instead of manual option.
setting = menu('Set the values of (threshold value , soft
or hard thresholding function , KeepApp)','manual
setting','defult setting');
switch setting
    case 1
        std_glb =
median(abs(Array_det_coef{level}))/0.6745;
        thr = std_glb*sqrt(2*log(length(noisy_speech)));
        thr_globstr  = num2str(thr);
        def_thr_s_1 = {thr_globstr,'s','1'};
        thr_sorh_k = inputdlg({'Enter the value of
threshold','Enter the type of thresholding function soft
or hard s or h','Threshold the approximation? 1:no or
0:yes'},'Setting parameters',1,def_thr_s_1);
        thr = str2num(thr_sorh_k{1});
        s_or_h = thr_sorh_k{2};
        KeepApp = str2num(thr_sorh_k{3});
    case 2
        % the default value of the threshold is
calculated as thr = std*sqrt(2*log(length(noisy_speech)))
where std = median(abs(D))/0.6745
        % such that D is calculated form the single level
DWT using haar wavelet.[D , A] = dwt('db1' ,
noisy_speech];also, std(noise) = median(abs(D))/0.6745
        [thr , s_or_h , KeepApp] = ddencmp('den' , 'wv' ,
noisy_speech(:,1));
```

```
        default_dialog = msgbox({'thr = ',num2str(thr)
,'s_or_h = ' ,s_or_h, 'Keepapp =
',num2str(KeepApp)},'Default values');
end
%----------
%3.4.a - Choose the type of thresholding

gbl_or_lvd = menu('Type of thresholding','Global
thresholding','Level-dependent thresholding','1-D wavelet
coefficients thresholding','Interval-dependent
thresholding');
msg2 = msgbox('De_noising ...');

FrameSelection = menu('Do you want to segment the speech
or not?','Yes','No');

switch FrameSelection
    case 1
seg_step = Fs*0.01;
overlap = Fs*0.01;
seg_len = seg_step + overlap;
sp_len = length(noisy_speech);
Nseg = floor(sp_len/(seg_step))-1;
window = hamming(seg_len);
de = hanning(2*overlap - 1)';
dewindow =  [de(1:overlap) , ones(1,seg_len -2*overlap) ,
de(overlap:end)]'./window;
        switch gbl_or_lvd
    %----------
    %3.4.a.1 - gbl : global thresholding
    case 1

%----------
%4.4.a - Reconstruct the signal from thresholded
coefficients

denoised_speech = zeros(sp_len, 1);

for i = 1 : Nseg
    sp_Seg(:,i) = noisy_speech((i-1)*seg_step+1 :
i*seg_step+overlap);

    noisy_speechW(:,i) = window.*sp_Seg(:,i);
    [Cad_seg , L] =
wavedec(noisy_speechW(:,i),level,wavename);

    Cdet_seg = detcoef(Cad_seg , L , 1);
    sigma_seg = median(abs(Cdet_seg))/0.6745;
    thr_seg =
sigma_seg*sqrt(2*log(length(noisy_speechW(:,i))));
```

```
    [denoised_seg ,Cad_thr_seg , L_thr_seg ,
L2norm_recovery_seg , cmp_score_seg] = wdencmp('gbl' ,
Cad_seg , L , wavename , level ,thr_seg , s_or_h , 1);

    denoised_seg (:,i) = denoised_seg;
    noisy_speechDe(:,i) = denoised_seg (:,i).*dewindow;
    denoised_speech((i-1)*seg_step+1 :
i*seg_step+overlap) = noisy_speechDe(:,i) +
denoised_speech((i-1)*seg_step+1 : i*seg_step+overlap);
end
%----------

    %----------
    %3.4.a.2 - lvd : level-dependent thresholding
    case 2
        THR_setManual = menu('Choose the thresholds (only
details coefficients) for level-dependet
thresholding','Manual setting','Based on threshold
selection rules');

        if ~(THR_setManual-1)
        THR_dlg = inputdlg({'Enter a list of thresholds
seperated by commas'},'Thresholds setting for level-
dependent')
        THR = str2num(THR_dlg{1});

        else

        %----------
        %3.4.a.2.1 - Choose the noise model
        noise_mod_menu = menu('Noise model','Unscaled
white noise','Scaled white noise','Non-white noise');
        noise_model = {'one' , 'sln' , 'mln'};
        SCAL = noise_model(noise_mod_menu);
        f =char(SCAL{1})
        %----------
        %3.4.a.2.2 - Chooose the threshold selection rule
        thrrule = menu('Threshold selection
rule','rigrsure' ,'heursure' , 'sqtwolog' , 'minimaxi');
        menu_thrrule = {'rigrsure' ,'heursure' ,
'sqtwolog' , 'minimaxi'};
        ThrSelectRule = menu_thrrule(thrrule);
        tptr = ThrSelectRule{1}
        %----------
        denoised_speech = zeros(sp_len , 1);
for i = 1 : Nseg
    sp_Seg(:,i) = noisy_speech((i-1)*seg_step+1 :
i*seg_step+overlap);
    noisy_speechW(:,i) = window.*sp_Seg(:,i);
    denoised_seg =
wden(noisy_speechW(:,i),tptr,s_or_h,f,level,wavename);

    denoised_seg(:,i) = denoised_seg;
    noisy_speechDe(:,i) = denoised_seg (:,i).*dewindow;
    denoised_speech((i-1)*seg_step+1 :
i*seg_step+overlap) = noisy_speechDe(:,i) +
denoised_speech((i-1)*seg_step+1 : i*seg_step+overlap);
    THR = zeros(1,level);
end
        end
    %----------
%----------

    %----------
    %3.4.a.3 selected level thresholding (only details
coefficients)
    case 3
        %----------
        %3.4.a.3.1 - Choose selcted level thresholding
methology
        x = menu('detail coeficients
thresholding:','thresholding the details for a given set
of level by forceing all coefficients to be
zero','thresholding the details for a given set of level
by using soft or hard thresholding function');
        if ~(x-1) %set all coeff's of selected level to
zero
            selected_lev = inputdlg({'Enter a list of
numbers (number of levels for thresholding)separated by
spaces or commas'},'Wavelet coefficient thresholding');
            LEV = str2num(selected_lev{1});

            denoised_speech = zeros(sp_len,1);

for i = 1 : Nseg
    sp_Seg(:,i) = noisy_speech((i-1)*seg_step+1 :
i*seg_step+overlap);

    noisy_speechW(:,i) = window.*sp_Seg(:,i);

    [Cad_seg , L] =
wavedec(noisy_speechW(:,i),level,wavename);

    Cdet_seg = detcoef(Cad_seg , L , 1);
    Capp = appcoef(Cad_seg,L,wavename,1);
    sigma_seg = median(abs(Cdet_seg))/0.6745;
    thr_seg = sigma_seg*sqrt(2*log(length(sp_Seg(:,i))));

    Cad_thr_seg = wthcoef('d' , Cad_seg , L , 1);
    denoised_seg = waverec(Cad_thr_seg,L,wavename);
    denoised_seg(:,i) = denoised_seg;
    noisy_speechDe(:,i) = denoised_seg (:,i).*dewindow;
    denoised_speech((i-1)*seg_step+1 :
i*seg_step+overlap) = noisy_speechDe(:,i) +
denoised_speech((i-1)*seg_step+1 : i*seg_step+overlap);
end

        else %threshod the selected level by (s or h)

            selected_lev = inputdlg({'Enter a list of
numbers (number of levels for thresholding)separated by
spaces or commas','Enter the crresponding
thresholds'},'Wavelet coefficient thresholding');
            LEV = str2num(selected_lev{1});
            T = str2num(selected_lev{2});

        denoised_speech = zeros(sp_len,1);

for i = 1 : Nseg
    sp_Seg(:,i) = noisy_speech((i-1)*seg_step+1 :
i*seg_step+overlap);

    noisy_speechW(:,i) = window.*sp_Seg(:,i);

    [Cad_seg , L] =
wavedec(noisy_speechW(:,i),level,wavename);

    Cdet_seg = detcoef(Cad_seg , L , 1);
    Capp = appcoef(Cad_seg,L,wavename,1);
    sigma_seg = median(abs(Cdet_seg))/0.6745;
    thr_seg = sigma_seg*sqrt(2*log(length(sp_Seg(:,i))));

    Cad_thr_seg = wthcoef('t' , Cad_seg , L , LEV
,T,s_or_h);
    denoised_seg = waverec(Cad_thr_seg,L,wavename);

    denoised_seg(:,i) = denoised_seg;
    noisy_speechDe(:,i) = denoised_seg (:,i).*dewindow;
    denoised_speech((i-1)*seg_step+1 :
i*seg_step+overlap) = noisy_speechDe(:,i) +
denoised_speech((i-1)*seg_step+1 : i*seg_step+overlap);
end
        end
        %----------
    %----------
    %3.4.a.4 - Interval-dependent thresholding
    case 4
        x1 = menu('Interval-dependent
denoising:','Interval-dependent denoising based on
variance change');
```

```
            num_int = inputdlg({'Enter the number of
intervals'} ,'Number of intervals' , 1 ,{'1'});
            nb_int = str2num(num_int{1});
            denoised_speech = zeros(sp_len,1);

for i = 1 : Nseg
    sp_Seg(:,i) = noisy_speech((i-1)*seg_step+1 :
i*seg_step+overlap);

    noisy_speechW(:,i) = window.*sp_Seg(:,i);


[denoised_seg,Cad_thr_seg,thrParamsOut_seg,int_DepThr_Cel
l_seg,BestNbOfInt_seg] =
cmddenoise(noisy_speechW(:,i),wavename,level,s_or_h,nb_in
t);
     denoised_seg = denoised_seg';
     denoised_seg0(:,i) = denoised_seg;
    noisy_speechDe(:,i) = denoised_seg0(:,i).*dewindow;
    denoised_speech((i-1)*seg_step+1 :
i*seg_step+overlap) = noisy_speechDe(:,i) +
denoised_speech((i-1)*seg_step+1 : i*seg_step+overlap);
end

    %

end

   [Cad_thr , L] =
wavedec(denoised_speech,level,wavename);

    case 2

    switch gbl_or_lvd
    %----------
    %3.4.a.1 - gbl : global thresholding
    case 1

%----------
%4.4.a - Reconstruct the signal from thresholded
coefficients

        [denoised_speech ,Cad_thr , L_thr ,
L2norm_recovery , cmp_score] = wdencmp('gbl' , Cad , L ,
wavename , level,thr , s_or_h , KeepApp);
        L2norm_recovery ,cmp_score
%----------

    %----------
    %3.4.a.2 - lvd : level-dependent thresholding
    case 2
        THR_setManual = menu('Choose the thresholds (only
details coefficients) for level-dependet
thresholding','Manual setting','Based on threshold
selection rules');

        if ~(THR_setManual-1)
        THR_dlg = inputdlg({'Enter a list of thresholds
seperated by commas'},'Thresholds setting for level-
dependent')
        THR = str2num(THR_dlg{1});

        else

        %----------
        %3.4.a.2.1 - Choose the noise model
        noise_mod_menu = menu('Noise model','Unscaled
white noise','Scaled white noise','Non-white noise');
        noise_model = {'one' , 'sln' , 'mln'};
        SCAL = noise_model(noise_mod_menu);
        f =char(SCAL{1})
        %----------
        %3.4.a.2.2 - Chooose the threshold selection rule
        thrrule = menu('Threshold selection
rule','rigrsure' ,'heursure' , 'sqtwolog' , 'minimaxi');
        %----------
        %3.4.a.2.3 - Calculate the threshold vector
        switch thrrule
            case 1
        THR= wthrmngr('dw1ddenoLVL' ,'rigrsure',Cad , L
,f);
            case 2
        THR= wthrmngr('dw1ddenoLVL' ,'heursure',Cad , L
,f);
            case 3
        THR= wthrmngr('dw1ddenoLVL','sqtwolog',Cad , L
,f);
            case 4
        THR= wthrmngr('dw1ddenoLVL' ,'minimaxi',Cad , L
,f);
        end
        %----------
        end
    %----------

%----------
%4.4.a - Reconstruct the signal from the thresholded
coefficients
        [denoised_speech ,Cad_thr , L_thr ,
L2norm_recovery , cmp_score] = wdencmp('lvd' , Cad , L ,
wavename ,level , THR , s_or_h);
```

```
%----------

    %----------
    %3.4.a.3 selected level thresholding (only details
coefficients)
    case 3
        %----------
        %3.4.a.3.1 - Choose selcted level thresholding
methology
        x = menu('detail coeficients
thresholding:','thresholding the details for a given set
of level by forceing all coefficients to be
zero','thresholding the details for a given set of level
by using soft or hard thresholding function');
        if ~(x-1) %set all coeff's of selected level to
zero
            selected_lev = inputdlg({'Enter a list of
numbers (number of levels for thresholding)separated by
spaces or commas'},'Wavelet coefficient thresholding');
            LEV = str2num(selected_lev{1});
            Cad_thr = wthcoef('d' , Cad , L , LEV);%1<=
N(i)<=length(L)-2
        else %threshod the selected level by (s or h)

            selected_lev = inputdlg({'Enter a list of
numbers (number of levels for thresholding)separated by
spaces or commas','Enter the crresponding
thresholds'},'Wavelet coefficient thresholding');
            LEV = str2num(selected_lev{1});
            T = str2num(selected_lev{2});
        Cad_thr = wthcoef('t' , Cad , L , LEV ,T,s_or_h);
        end
        %----------
    %----------

%----------
%4.4.a - Reconstruct the signal from the thresholded
coefficients
        denoised_speech = waverec(Cad_thr,L,wavename);
%----------

    %----------
    %3.4.a.4 - Interval-dependent thresholding
    case 4
        x1 = menu('Interval-dependent
denoising:','Interval-dependent denoising based on
variance change','Manual setting for intervals and its
thresholds');
        if ~(x1-1)
            num_int = inputdlg({'Enter the number of
intervals'} ,'Number of intervals' , 1 ,{'1'});
```

```
            nb_int = str2num(num_int{1});

[denoised_speech,Cad_thr,thrParamsOut,int_DepThr_Cell,Bes
tNbOfInt] =
cmddenoise(noisy_speech,wavename,level,s_or_h,nb_int);
            denoised_speech = denoised_speech';

        else
            cel = cell(1,level);
            def = cell(1,level);
            for w0 = 1 :level
                cel{w0} = strcat('level',' '
,num2str(w0));
                def{w0} = strcat('start , end , thr ;
...');
            end
            options.Resize = 'on';
            THR_int_dep = inputdlg(cel,'Threshold setting
for interval-dependent thresholding',level,def,options);
           for w1 = 1:level
            f = str2num(THR_int_dep{w1});
            cel{w1} = f;
           end
           Cad_thr =
cadthrCompute(cel,Cad,L,level,s_or_h);

           denoised_speech =
cmddenoise(noisy_speech,wavename,level,s_or_h,NaN,cel);
           denoised_speech = denoised_speech';
        end
    %----------
%4.5.a - Reconstruct the approximation and detail signals
from wavelet thresholded decomposition
           App_den_sig = wrcoef('a', Cad_thr , L ,
wavename , level);
           ReconstructArray_densig{1} = App_den_sig;
           for k0 = level : -1 : 1
               Det_den_sig  = wrcoef('d' , Cad_thr , L ,
wavename , k0);
               ReconstructArray_densig{k0+1} =
Det_den_sig;
           end
end
end

 case 2 %DWP
   %----------
%2.2.a - Set the number of level decomposition and the
wavelet function name
```

```
lev_wname = inputdlg({'Enter the number of decomposition
levels','Enter the wave name : dbN'},'Number of level and
wavename');
level = str2num(lev_wname{1});
wavename = lev_wname{2};
%----------
%2.3.a - Find the wavelet and scaling functions
[W , xval_dbn] = wpfun(wavename ,7);
%----------
%2.4.a - Find the wavelet and scaling filters
[Lo_d , Lo_r] = wfilters(wavename , 'l')
Hi_r = qmf(Lo_r)
Hi_d = wrev(Hi_r)

sum_Hi_d = sum(Hi_d);
sum_Lo_d = sum(Lo_d);

Nextpow2_Lo_d = nextpow2(length(Lo_d));
Nextpow2_Hi_d = nextpow2(length(Lo_d));
Nextpow2_Lo_r = nextpow2(length(Lo_r));
Nextpow2_Hi_r = nextpow2(length(Lo_r));

fftLo_d = fft(Lo_d,2^Nextpow2_Lo_d);
fftHi_d= fft(Hi_d,2^Nextpow2_Hi_d);
fftLo_r = fft(Lo_r,2^Nextpow2_Lo_r);
fftHi_r= fft(Hi_r,2^Nextpow2_Hi_r);
%----------
%2.5.a - Decompose the noisy signal at a given level
using the wavelet filters
wpt = wpdec(noisy_speech ,level , wavename);
%----------
cfs_cell = cell(1,level);
rcfs_cell = cell(1,level);
for i = 1 :level+1
    for j = 0 : (2^level)-1
        node(1) = i-1;
        node(2) = j;
cfs_cell{i} = wpcoef(wpt,[node(1),node(2)]);
rcfs_cell{i} = wprcoef(wpt,[node(1),node(2)]);
    end
end
%----------
det_first_scl = wpcoef(wpt,[1 1]);
sigma_stdNoise = median(det_first_scl)/0.6745;
alpha = 2
thr = wpbmpen(wpt,sigma_stdNoise,alpha);
setting = menu('Set the values of (threshold value , soft
or hard thresholding function , KeepApp)','setting');
        thr_str  = num2str(thr);
        def_thr_s_1 = {thr_str,'s','1'};
        thr_sorh_k = inputdlg({'Enter the value of
threshold','Enter the type of thresholding function soft
or hard s or h','threshold the approximation 1 or
0'},'Setting parameters',1,def_thr_s_1);
        thr = str2num(thr_sorh_k{1});
        s_or_h = thr_sorh_k{2};
        KeepApp = str2num(thr_sorh_k{3});

        NT = wpthcoef(wpt,KeepApp,s_or_h,thr);
        denoised_speech = wprec(NT);
%denoised_speech =
wpdencmp(wpt,s_or_h,'nobest',thr,KeepApp);
%----------
    %----------
    fig7 = figure('name' , strcat('The ',wavename,'
Wavelet Packets'),'Color','w');
    %----------

    [ d_str , moment_str] = strread(wavename , '%s %s' ,
'delimiter' , 'b');
    moment = str2num(moment_str{1});
    for wfun=1:8
        subplot(2,4,wfun); axis tight;
        plot(xval_dbn,W(wfun,:));
        xlabel(strcat('W',num2str(wfun-1)));
        xlim([0,(2*moment)-1]);
    end
    title(strcat('The ',wavename,' Wavelet Packets'));

    %----------
    fig8 = figure('name' , 'filters and fft of
filters','Color','w');
    %----------

subplot(421); axis tight ;
stem(Lo_d,'r');title('Decomposition low pass filter');
subplot(422); axis tight ;
stem(Hi_d,'r');title('Decomposition high pass filter');
subplot(423); axis tight ;
stem(Lo_r,'b');title('Reconstruction low pass filter');
subplot(424); axis tight ;
stem(Hi_r,'b');title('Reconstruction high pass filter');

subplot(425); axis tight
Freq_Lo_d = (2*pi)/(2^Nextpow2_Lo_d)
:(2*pi)/(2^Nextpow2_Lo_d) : pi ;
fa_ld = abs(fftLo_d(1:(2^(Nextpow2_Lo_d)/2)));
plot(Freq_Lo_d,fa_ld);title('FFT of analysis low pass
filter');
xlim([(2*pi)/(2^Nextpow2_Lo_d) , pi+0.5]);
```

```
subplot(426); axis tight
Freq_Lo_r = (2*pi)/(2^Nextpow2_Lo_r)
:(2*pi)/(2^Nextpow2_Lo_r) : pi ;
fa_hr = abs(fftLo_r(1:(2^(Nextpow2_Lo_r)/2)));
plot(Freq_Lo_r,fa_hr);title('FFT of synthesis low pass
filter');
xlim([(2*pi)/(2^Nextpow2_Lo_r) , pi+0.5]);

subplot(427); axis tight
Freq_Hi_d = (2*pi)/(2^Nextpow2_Hi_d)
:(2*pi)/(2^Nextpow2_Hi_d) : pi ;
fa_ld = abs(fftHi_d(1:((2^Nextpow2_Hi_d)/2)));
plot(Freq_Hi_d,fa_ld);title('FFT of analysis high pass
filter');
xlim([(2*pi)/(2^Nextpow2_Hi_d) , pi+0.5]);

subplot(428); axis tight
Freq_Hi_r = (2*pi)/(2^Nextpow2_Hi_r)
:(2*pi)/(2^Nextpow2_Hi_r) : pi ;
fa_hr = abs(fftHi_r(1:((2^Nextpow2_Hi_r)/2)));
plot(Freq_Hi_r,fa_hr);title('FFT of synthesis high pass
filter');
xlim([(2*pi)/(2^Nextpow2_Hi_r) , pi+0.5]);
end

%----------
%4.6.a/b - Plotting illustration

    %----------
    fig9 = figure('name' , 'thresholding function
illustration','Color','w');
    %----------

lin_fun = linspace(-0.5 , 0.5 , 100);
lin_fun_s = wthresh(lin_fun , 's' , thr);
lin_fun_h = wthresh(lin_fun , 'h' , thr);
subplot(131); plot(lin_fun,lin_fun,'k'); title('Original
function');
subplot(132); plot(lin_fun,lin_fun_s,'b'); title('Soft
thresholded function');
text(thr , -0.05 , [ strcat('(',num2str(thr),',') ,
strcat(num2str(0),')')],'Color' ,
'b','HorizontalAlignment','center')
subplot(133); plot(lin_fun,lin_fun_h,'r'); title('Hard
thresholded function');
text(thr , -0.05 , [ strcat('(',num2str(thr),',') ,
strcat(num2str(0),')')],'Color' ,
'r','HorizontalAlignment','center')
for fig = 1 : 3
   subplot(1,3,fig)
   xlabel('coefficients before thresholding');
   ylabel('coefficients after thresholding');
end
shg

switch c
case 1

    %----------
    fig10 = figure('name' , 'clear , Noisy and Denoised
speech signals','Color','w');
    %----------

subplot(2,4,1:2); axis tight;
plot([1:length(noisy_speech)]/Fs , noisy_speech ,'r');
xlabel('Time(s)'); ylabel('Amplitude');
title('Noisy speech signal');

subplot(2,4,3:4); axis tight;
plot([1:length(denoised_speech)]/Fs ,
denoised_speech,'k');
xlabel('Time(s)'); ylabel('Amplitude');
title('De-noised speech signal');

subplot(2,4,6:7); axis tight;
residual = denoised_speech - clear_speech;
plot([1:length(denoised_speech)]/Fs , residual,'g');
xlabel('Time(s)'); ylabel('Amplitude');
title('Residual signal');

    %----------
    fig11 = figure('name' , 'clear , Noisy and Denoised
speech signals','Color','w');
    %----------

subplot(211); axis tight;
plot([1:length(clear_speech)]/Fs , clear_speech ,'b');
xlabel('Time(s)'); ylabel('Amplitude');
title('Clear and denoised speech signals');
hold on;
plot([1:length(denoised_speech)]/Fs ,
denoised_speech,'k');
legend('clear speech' , 'denoised speech');

subplot(212); axis tight;
plot([1:length(noisy_speech)]/Fs , noisy_speech ,'r');
xlabel('Time(s)'); ylabel('Amplitude');
title('noisy and denoised speech signals');
hold on;
plot([1:length(denoised_speech)]/Fs ,
denoised_speech,'k');
legend('noisy speech' , 'denoised speech');
```

```
    %----------
    fig12 = figure('name' , 'correlation','Color','w');
    %----------

subplot(211); axis tight;
[xcor_bef , lag_bef] = crosscorr(clear_speech ,
noisy_speech);
 xc_bef = xcor_bef(21);
plot(lag_bef , xcor_bef);
xlabel('lag'); ylabel('sample cross correlation');
title('correlation between clear signl and noisy
signal');

subplot(212); axis tight;
[xcor_aft,lag_aft] = crosscorr(clear_speech ,
denoised_speech);
xc_aft = xcor_aft(21);
plot(lag_aft , xcor_aft);
xlabel('lag'); ylabel('sample cross correlation');
title('correlation between clear signl and denoised
signal');

    %----------
    fig13 = figure('name' , 'power
distribution','Color','w');
    %----------

Npow2 = pow2(nextpow2(length(denoised_speech)));
denoised_speech_pad = fft(denoised_speech , Npow2);
noisy_speech_pad = fft(noisy_speech , Npow2);
clear_speech_pad = fft(clear_speech , Npow2);

freq_range = (0:(Npow2 - 1))*(Fs/Npow2);

power_denoised_speech =
denoised_speech_pad.*conj(denoised_speech_pad)/Npow2;
power_noisy_speech =
noisy_speech_pad.*conj(noisy_speech_pad)/Npow2;
power_clear_speech =
clear_speech_pad.*conj(clear_speech_pad)/Npow2;

subplot(131); axis tight;
plot(freq_range , power_denoised_speech);
xlabel('frequency Hz'); ylabel('power');
title('power distribution of denoised speech signal');

subplot(132); axis tight;
plot(freq_range , power_noisy_speech);
xlabel('frequency Hz'); ylabel('power');
title('power distribution of noisy speech signal');
```

```
subplot(133); axis tight;
plot(freq_range , power_clear_speech);
xlabel('frequency Hz'); ylabel('power');
title('power distribution of clear speech signal');

    %----------
    fig14 = figure('name' , 'Spectrograms','Color','w');
    %----------
subplot(131);
spectrogram(clear_speech);ylabel('Time(ms)');
title('Spectrogram of clear speech')
subplot(132);
spectrogram(noisy_speech);ylabel('Time(ms)');
title('Spectrogram of noisy speech')
subplot(133);
spectrogram(denoised_speech);ylabel('Time(ms)');
title('Spectrogram of denoised speech')
figx = figure('name' , 'Spectrograms of noisy and
denoised speech signals','Color','w');
subplot(311);
plot([1:length(noisy_speech)]/Fs , noisy_speech ,'r');
xlabel('Time(s)'); ylabel('Amplitude');
title('noisy and denoised speech signals');
hold on;
plot([1:length(noisy_speech)]/Fs , denoised_speech,'k');
subplot(312);
 spectrogram(noisy_speech,'yaxis');
 xlabel('Time(ms)')
 title('Spectrogram of noisy speech signal')
 subplot(313);
 spectrogram(denoised_speech,'yaxis');
 xlabel('Time(ms)')
 title('Spectrogram of de-noised speech signal')

if(c1-1)

    %----------
    fig15 = figure('name','Wavelet Packet
Spectrum','Color','w');
    %----------

subplot(131);
[spect_noisy_coef,Time0,Frequency0] =
wpspectrum(wpt,Fs,'plot');
title('Wavelet Packet Decomposition of Noisy Speech')
subplot(132);
[spect_denoised_coef,Time1,Frequency1] =
wpspectrum(NT,Fs,'plot');
title('Wavelet Packet Decomposition of Denoised Speech')
subplot(133);
```

```
wpt_c = wpdec(clear_speech ,level , wavename);
[spect_clear_coef,Time2,Frequency2] =
wpspectrum(wpt_c,Fs,'plot');
title('Wavelet Packet Decomposition of Clear Speech')
else
     %----------
     fig15 = figure('name' , 'Absolute coefficients of
DWT','Color','w');
     %----------

[Cad_clear , L] = wavedec(clear_speech,level,wavename);

len = length(clear_speech);
cfd1 = zeros(level,len);
cfd2 = zeros(level,len);
cfd3 = zeros(level,len);
 for k0 = 1 : level
     d_1 = detcoef(Cad_clear,L,k0);
     d_2 = detcoef(Cad,L,k0);
     d_3 = detcoef(Cad_thr,L,k0);
     d_1 = d_1(:)';
     d_2 = d_2(:)';
     d_3 = d_3(:)';
     d_1 = d_1(ones(1,2^k0),:);
     d_2 = d_2(ones(1,2^k0),:);
     d_3 = d_3(ones(1,2^k0),:);
     cfd1(k0,:) = wkeep1(d_1(:)',len);
     cfd2(k0,:) = wkeep1(d_2(:)',len);
     cfd3(k0,:) = wkeep1(d_3(:)',len);
 end
 cfd1 = cfd1(:);
 cfd2 = cfd2(:);
 cfd3 = cfd3(:);
 I1 =find(abs(cfd1)<sqrt(eps));
 I2 =find(abs(cfd2)<sqrt(eps));
 I3 =find(abs(cfd3)<sqrt(eps));
 cfd1(I1) = zeros(size(I1));
 cfd2(I2) = zeros(size(I2));
 cfd3(I3) = zeros(size(I3));
 cfd1 = reshape(cfd1,level,length(denoised_speech));
 cfd2 = reshape(cfd2,level,length(denoised_speech));
 cfd3 = reshape(cfd3,level,length(denoised_speech));
 %Plot abs. of DWT
 subplot(321);
 plot(clear_speech); title('Clear speech signal');
 subplot(322);
 image(flipud(wcodemat(cfd1,255,'row')));
 colormap(pink(255));
 set(gca,'yticklabel',[]);
 title('Absolute coefficients of DWT for clear speech');
ylabel('Level');
```

```
 subplot(323);
 plot(noisy_speech); title('Noisy speech signal');
 subplot(324);
 image(flipud(wcodemat(cfd2,255,'row')));
 colormap(pink(255));
 set(gca,'yticklabel',[]);
 title('Absolute coefficients of DWT for noisy speech');
ylabel('Level');

 subplot(325);
 plot(denoised_speech); title('Denoised speech signal');
 subplot(326);
 image(flipud(wcodemat(cfd3,255,'row')));
 colormap(pink(255));
 set(gca,'yticklabel',[]);
 title('Absolute coefficients of DWT for denoised
speech'); ylabel('Level');
end

    %----------
    fig16 = figure('name' , 'Histograms','Color','w');
    %----------

x_bar = -1.5:0.05:1.5;
subplot(231);rng(0,'twister');
hist(clear_speech,x_bar);
title('Histogram of clear speech')
subplot(234);
n_1 = histc(clear_speech,x_bar);
cum_1 = cumsum(n_1);
bar(x_bar,cum_1,'BarWidth',1);
title('Cumulative histogram of clear speech')
subplot(232);rng(0,'twister');
hist(noisy_speech,x_bar);
title('Histogram of noisy speech')
subplot(235);
n_2 = histc(noisy_speech,x_bar);
cum_2 = cumsum(n_2);
bar(x_bar,cum_2,'BarWidth',1);
title('Cumulative histogram of noisy speech')
subplot(233);rng(0,'twister');
hist(denoised_speech,x_bar);
title('Histogram of denoised speech')
subplot(236);
n_3 = histc(denoised_speech,x_bar);
cum_3 = cumsum(n_3);
bar(x_bar,cum_3,'BarWidth',1);
title('Cumulative histogram of denoised speech')

    %----------
    fig17 = figure('name' , 'Statistics of residual
signal','Color','w');
    %----------

%struct_statistic =
struct('mean',mean(residual),'median',median(residual),'s
td',std(residual),'var',var(residual),'L1_norm',sum(abs(r
esidual)),'L2_norm',sum(abs(residual).^2));
data_colm =
{'mean','median','std','var','L1_norm','L2_norm'};
data_statistic =
[mean(residual),median(residual),std(residual),var(residu
al),sum(abs(residual)),sum(abs(residual).^2)];
x_bar = -1:0.005:1;
subplot(4,2,1:2);
plot(residual); axis tight;
subplot(4,2,3);
hist(residual,x_bar);
title('Histogram of residual signal')
subplot(4,2,4);
n_4 = histc(residual,x_bar);
cum_4 = cumsum(n_4);
bar(x_bar,cum_4);
title('Cumulative histogram of residual signal')
subplot(4,2,5);
[auto_cor,lag_auto] = xcorr(residual,'coeff');
plot(lag_auto,auto_cor);
title('Auto-correlation of residuals')
subplot(4,2,6);
residual_pad = fftshift(fft(residual , Npow2));
plot(freq_range , residual_pad);
title('FFT of residual signal')
subplot(4,2,7:8);
axis off
tab =
uitable(fig17,'Data',data_statistic,'ColumnName',data_col
m,'RowName','Res.','Position',[40 40 650 70]);

case 2

    %----------
    fig18 = figure('name' ,'Noisy and Denoised recorded
speech signals','Color','w');
    %----------

subplot(211); axis tight;
plot([1:length(noisy_speech)]/Fs , noisy_speech,'r');
xlabel('Time(s)'); ylabel('Amplitude');
title('noisy recorded speech signals');
legend('noisy recorded speech');
```

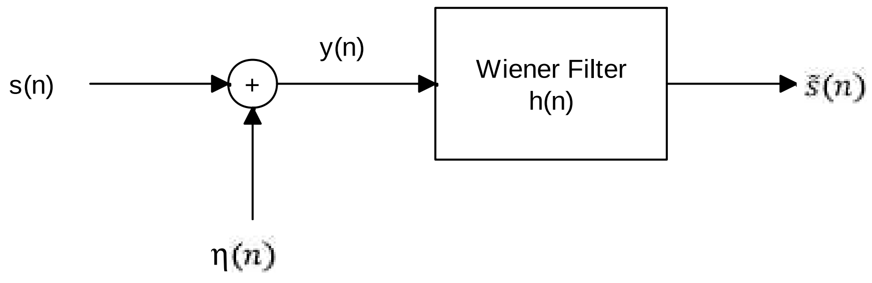
```
subplot(212); axis tight;
plot([1:length(noisy_speech)]/Fs , noisy_speech,'r');
xlabel('Time(s)'); ylabel('Amplitude');
title('Noisy and De-noised speech signal');
hold on;
plot([1:length(denoised_speech)]/Fs ,
denoised_speech,'k');
legend('noisy speech' , 'denoised speech');

    %----------
    fig19 = figure('name' , 'power
distribution','Color','w');
    %----------

Npow2 = pow2(nextpow2(length(denoised_speech)));
denoised_speech_pad = fft(denoised_speech , Npow2);
noisy_speech_pad = fft(noisy_speech , Npow2);

freq_range = (0:(Npow2 - 1))*(Fs/Npow2);

power_denoised_speech =
denoised_speech_pad.*conj(denoised_speech_pad)/Npow2;
power_noisy_speech =
noisy_speech_pad.*conj(noisy_speech_pad)/Npow2;

subplot(121); axis tight;
plot(freq_range , power_denoised_speech);
xlabel('frequency Hz'); ylabel('power');
title('power distribution of denoised speech signal');

subplot(122); axis tight;
plot(freq_range , power_noisy_speech);
xlabel('frequency Hz'); ylabel('power');
title('power distribution of noisy recorded speech
signal');
end

%----------
%4.7.a - Playing denoised speech
switch c
    case 1
sound(clear_speech , Fs); pause((length(clear_speech)/Fs)
+ 2 )
sound(noisy_speech , Fs); pause((length(clear_speech)/Fs)
+ 2 )
sound(denoised_speech , Fs);

    case 2
sound(noisy_speech , Fs); pause((length(noisy_speech)/Fs)
+ 2 )
sound(denoised_speech , Fs);
```

```
end

tilefigs
%----------
%% Stage 4:
%   5.1.a - Performance measurements
%       5.1.a.1 - Mean Square Error (MSE)
%       5.1.a.2 - Signal to Noise Ratio (SNR)
%   5.2.a - Plotting of curve measurements

%----------
%5.1.a - Performance measurements

%----------
%5.1.a.1 - Mean Square Error (MSE)
if ~(c-1)
MSE_in = mean(sum((clear_speech - noisy_speech).^2));
MSE_out = mean(sum((clear_speech - denoised_speech).^2));
%----------
%5.1.a.2 - Signal to Noise Ratio (SNR)
SNR_out =
10*log(mean(abs(clear_speech.^2))/mean(abs((clear_speech-
denoised_speech).^2)));
%----------
msg3 = msgbox({strcat('MSE_in is ',num2str(MSE_in),'  and
MSE_out is ',num2str(MSE_out)),'',strcat('SNR_in is
',num2str(snrval),'  and  SNR_out
is',num2str(SNR_out))});
end
```

Appendix C
Wiener Filtering

The goal of the Wiener filter is to compute a statistical estimate of an unknown signal using a related signal as an input and filtering that known signal to produce the estimate as an output. For example, the known signal might consist of an unknown signal of interest that has been corrupted by additive noise. The Wiener filter can be used to filter out the noise from the corrupted signal to provide an estimate of the underlying signal of interest.

The design of the Wiener filter takes a different approach. One is assumed to have knowledge of the spectral properties of the original signal and the noise, and one seeks the linear time-invariant filter whose output would come as close to the original signal as possible. The assumption is that the signal and additive noise are stationary linear stochastic processes with known spectral characteristics or known autocorrelation and cross-correlation. The requirement is that the filter must be physically realizable/causal (this requirement can be dropped, resulting in a non-causal solution). The performance criterion is the minimum mean-square error (MMSE).[29]



Block diagram of Wiener filtering

From the above block diagram, the equation that describe the wiener filtering as follows

$$y\ n\ = s\ n\ + \eta\ n \tag{1}$$

$$\tilde{s}\ n\ = h\ n\ * y\ n \tag{2}$$

$$e\ n\ = s\ n\ - \tilde{s}\ n \tag{3}$$

Where $s\ \ n$ is the clear speech signal, $\eta(n)$ is the additive noise, $y\ \ n$ is the noisy speech signal, $h\ \ n$ is the impulse response of wiener filter, $\tilde{s}\ n$ is the de-noised speech signal and $e\ \ n$ is the error signal.

Based on the assumption that the speech signal and the additive noise signal are uncorrelated stationary random process then the wiener filter frequency response is

$$H\ \ w\ = \frac{S^2(w)}{S^2(w) + N^2(w)} \qquad (4)$$

Where $H\ \ w$ is the transfer function of the wiener filter, $S^2(w)$ is the clear speech power spectrum and $N^2(w)$is the noise power spectrum.

For the additive white gaussian noisy signal with length equal to L and variance noise equal to σ^2 then

$$N^2\ \ w\ = L.\sigma^2 \qquad (5)$$

$$H\ \ w\ = \frac{S^2(w)}{S^2(w) + L.\sigma^2} \qquad (6)$$

To estimate the variance of the noise, the following estimation can be used based on wavelet transform

$$\hat{\sigma}^2 = \frac{median\ \ y_{1,k}^{d}}{0.6745}\ ^{2}$$

where $y_{1,k}^{d}$ is the details wavelet coefficient sequence of the noisy signal on first level.

Matlab code for wiener filtering:

```
[Cad,L] = wavedec(y,1,'db8');
sigma = median(abs(detcoef(Cad,L,1)))/0.6745;

FFT_s = fft(s);% FFT of clear speech signal
FFT_y = fft(y);% FFT of noisy speech signal

Power_s = abs(FFT_s.*FFT_s);% Power density of clear
speech signal

H = Power_s./(Power_s+ sp_len*sigma^2);% Wiener filter

FFT_den = FFT_y.*H;% FFT of de-noised speech signal

Power_err = abs((FFT_s - FFT_y).*(FFT_s - FFT_y));%
Estimate of noise power density

denoised_speech = real(ifft(FFT_den));% denoised speech
signal
```

References

[1] Pankaj Bactor, Anil Garg, ''Different Techniques for the Enhancement of the Intelligibility of a Speech Signal'', *International Journal of Engineering Research and Development, Volume 2, Issue 2 (July 2012), PP. 57-64.*

[2] Steven F.Boll, ''Suppression of Acoustic Noise in Speech Using Spectral Subtraction'', IEEE TRANSACTIONS ON ACOUSTICS, SPEECH, AND SIGNAL PROCESSING, VOL. ASSP-27, NO. 2, APRIL 1979.

[3] M. A. Abd El-Fattah, M. I. Dessouky, S. M. Diab and F. E. Abd El-samie , ''SPEECH ENHANCEMENT USING AN ADAPTIVE WIENER FILTERING APPROACH'', *Progress In Electromagnetics Research M, Vol. 4, 167–184, 2008.*

[4] Yariv Ephraim, Student Member,*IEEE,and David Malah, Senior Member, IEEE* ,''Speech Enhancement Using a- Minimum Mean-Square Error Short-Time Spectral Amplitude Estimator'', IEEE TRANSACTIONS ON ACOUSTICS, SPEECH, AND SIGNAL PROCESSING, VOL. ASSP-32, NO. *6*, DECEMBER 1984.

[5] Xiaoqiang Xiao, Peng Lee, and Robert M. Nickel, '' INVENTORY BASED SPEECH DENOISING WITH HIDDEN MARKOV MODELS'', 16th European Signal Processing Conference (EUSIPCO 2008), Lausanne, Switzerland, August 25-29, 2008, copyright by EURASIP.

[6] Mark Klein and Peter Kabal, ''SIGNAL SUBSPACE SPEECHENHANCEMENTWITH PERCEPTUAL POST-FILTERING'', *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing* (Orlando, FL), pp. I-537-I-540, May 2002.

[7] D.Gabor, ''THEOREY OF COMMUNICATION'', Britsh Thomson-Houston Co., Ltd, Research laboratory, received 25th November, 1944.

[8] Stephane G. Mallat,'' A Theory for Multiresolution Signal Decomposition: The Wavelet Representation'', IEEE TRANSACTIONSO N PATTERNA NALYSIS AND MACHINE INTELLIGENCE.V OL. II, NO. 7. JULY 1989.

[9] Stephane G. Mallat,'' Multifrequency Channel Decompositions of Images and Wavelet Models'', *IEEE TRANSACTIONS ON. Acoustics, Speech, and Signal Processing VOL-37.NO.12.december 1989.*

[10] Ingrid Daubechies, '' Orthonormal Bases of Compactly Supported Wavelets'', AT&T Bell laboratories, received December, 1987.

[11] Stephane G. Mallat and Wen Liang Hwang, ''SingularityDetection and Processing with Wavelets'',*IEEE TRANSACTIONS ON INFORMATION* THEORY, VOL.38,*NO.2,MARCH 1992.*

[12] David L.Donoho, ''NONLINEAR WAVELET METHODS FOR RECOVERY OF SIGNALS,DENSITIES,AND SPECTRA FROM INDIRECT AND NOISY DATA'', Proceeding of Symposia in Applied Margematics, VOL.00, 1993.

[13] David L. Donoho,’’De-Noising by Soft-Thresholding’’, TRANSACTIONS ON INFORMATION THEORY, VOL. 41, NO. 3, MAY 1995.

[14] David L. Donoho and Iain M. Johnstone,’’ Ideal Spatial Adaptation by Wavelet Shrinkage’’, Biometrika, 81, 425-455.

[15] Iain M.Johnstone and Bernard W.Silverman,’’Wavelet threshold estimators for data with correlated noise’’,revision august 20,1996.

[16] Lawrence Rabiner, Biig-Hwang Juang, FUNDAMENTALS OF SPEECH RECOGNITION, PTR Prentice-Hall, Inc, Englewood Cliffs, New Jersey, 1993.

[17] S.ChinaVenkateswarlu, Dr. K.Satya Prasad, Dr. A.SubbaRamiReddy,’’Improve Speech Enhancement Using Weiner Filtering’’, Global Journal of Computer Science and Technology, Volume 11 Issue 7 Version 1.0 May 2011.

[18] RobiPolikar, THE ENGINEER'S ULTIMATE GUIDE TO WAVELET ANALYSIS, 1999, available at

http://users.rowan.edu/~polikar/WAVELETS/WTtutorial.html

[19] R.J.E. Merry, Wavelet Theory and Applications A literature study, 2005.

[20] Brani Vidakovic and Peter Mueller, WAVELETS FOR KIDS A Tutoial Introduction, 1991.

[21] Andrew E. Yagle and Byung-Jae Kwak, THE WAVELET TRANSFORM: WHAT'S IN IT FOR YOU?,Presentation to Ford Motor Co., 1996.

[22] Bernhard Wieland, Speech Signal Noise Reduction with Wavelets, 2009.

[23] Ste'phane Mallat, AWavelet Tour of Signal Processing, Academic Press, USA,1998.

[24] Gilbert Strange and Truong Nguyen, Wavelets and Filter Banks, Wellesley-Cambridge Press, USA,1997.

[25]Gilbert Strange, Computational Science and Engineering, Wellesley-Cambridge Press, USA,2007.

[26] Ole Christensen, Functions, Spaces, and Expansions : Mathematical Tools in Physics and Engineering, Springer Science+Business Media LLC, USA, 2010.

[27] Michel Misiti, Yves Misiti, Georges Oppenheim and Jean-Michel Poggi, Wavelet Toolbox For Use With MATLAB, The MathWorks, Inc., 1996.

[28] www.mathworks.com

[29] wikipedia.org/wiki/Wiener_filter