Student Minor Research Project

AUDIO COMPRESSION USING WAVELETS IN MATLAB

Under RUSA 2.0 Scheme

(Through Ch.S.D.St.Theresa's College for Women (Autonomous), Eluru, AP)

Submitted by

Ms K Jyothsna , III B.Sc. MPE (Reg.No.11704002)

Ms A J R Satya, III B.Sc. MECs (Reg.No.11705010)

Mr G D N Raju, III B.Sc. MECs (Reg.No.11705041)

Under the guidance of

Dr K Venkateswarlu

HOD of Electronics & Project Advisor

Department Of Electronics

SRI Y N COLLEGE

(AUTONOMOUS)

Thrice Accredited by NAAC at 'A' Grade

Recognized by UGC as "College with Potential for Excellence"

Narsapur-534275, AP, India

December-2019

Department Of Electronics

SRI Y N COLLEGE

(AUTONOMOUS)

Thrice Accredited by NAAC at 'A' Grade

Recognized by UGC as "College with Potential for Excellence"

Narsapur-534275, AP, India

CERTIFICATE

*This is to certify that the project work entitled "***Audio Compression using Wavelets in MATLAB***" is bonafied work carried out by Ms K Jyothsna (Reg.No: 11704002), Ms A J R Satya (Reg.No: 11705010), Mr G D N Raju (Reg.No: 11705041), submitted in Third Year of the degree B.Sc. in Electronics during the year 2019-20 is an authentic work under my supervision and guidance.*

To the best of my knowledge, the matter embodied in the project work has not been submitted to any other College/Institute.

Date: 29-12-2019

Dr K Venkateswarlu
Project Advisor
Department of Electronics

ACKNOWLEDGEMENT

We place on record and warmly acknowledge the continuous encouragement, invaluable supervision, timely suggestions and inspired guidance offered by our Project advisor, ***Dr K Venkateswarlu,*** *Head, Department of Electronics,* ***Sri Y N College (Autonomous),*** ***Narsapur*** *in bringing this report to a successful completion.*

We are grateful to ***Mr K Vinaya Phaneendhra****, Lecturer, Department of Electronics for permitting us to make use of the facilities available in the department to carry out the project successfully. Last but not the least we express our sincere thanks to all of our friends who have patiently extended all sorts of help for accomplishing this undertaking.*

Finally we extend our gratefulness to one and all who are directly or indirectly involved in the successful completion of this project work.

Ms. K Jyothsna	**Ms. A J R Satya**	**Mr. G D N Raju**
III.B.Sc.MPE	III.B.Sc.MECs	III.B.Sc.MECs
Reg. No 11704002	Reg. No.11705010	Reg.No.11705041

DECLARATION

We, the undersigned, declare that the project entitled "**Audio Compression using Wavelets in MATLAB**", being submitted in Third Year of Bachelor of Science in Electronics, Sri Y N College (Autonomous), is the work carried out by us.

Ms. K Jyothsna	**Ms. A J R Satya**	**Mr. G D N Raju**
III.B.Sc.MPE	III.B.Sc.MECs	III.B.Sc.MECs
Reg. No 11704002	Reg. No.11705010	Reg.No.11705041

Contents **Page No.**

List of Figures & Tables Page No.

1. ABSTRACT

Audio frequencies range from 20Hz to 20kHz but these frequencies are not heard in the same way. Frequencies below 20Hz and above 20kHz are very difficult to hear, while those not much more than 20Hz, or not much less than 20kHz, cannot be heard by most people. We often need to process these audio signals for various applications. MATLAB is one of the best signal analysis and signal processing tools.

The main objective of this project is to study the audio compressing techniques that use wavelets. To simulate using MATLAB and obtain the compressed audio signal. Audio compression is a very good example of speech and signal processing. We use the Internet for various purposes including entertainment. Audio is common in all entertainment applications. If an audio file size is large, it takes more space to store.

Audio/video compression frees up space substantially, which can then be utilised for other purposes.

.

2. INTRODUCTION

Compression is process of converting an input data stream into another data stream that has smaller size. Compression provides the reduction in redundancy also used to reduce storage requirements overall program execution time may be reduced. This is because reduction in storage will result in reduction of disc access attempts. The compression algorithmhelp to reduce the bandwidth requirements and also provide a level of security for the data being transmitted. The wavelets consist of banks of low pass filters, high pass filters and down sampling units. Half of the filter convolution results are discarded because of the down sampling at each wavelet decomposition stage. Only the approximation part of the Daubechie wavelet results is kept so that the number of samples is reduced by half

Non linear frequency response of the hear:

Humans are able to hear frequencies in the range approximately from 20 Hz to 20 kHz. However, this does not mean that all frequencies are heard in the same way. One could make the assumption that a human would hear frequencies that make up speech better than others, and that is in fact a good guess. Furthermore, one could also hypothesize that hearing a tone becomes more difficult close to the extremes frequencies (i.e. close to 20 Hz and 20kHz).
After many cochlear studies, scientists have found that the frequency range from 20 Hz to 20 kHz can be broken up into critical bandwidths, which are non-uniform, non-linear, and dependent on the level of the incoming sound. Signals within one critical bandwidth are hard to separate for a human observer. A detailed description of this behavior is described in the Bark scale and Fletcher curves.

Masking property of the auditory system:

Auditory masking is a perceptual property of the human auditory system that occurs whenever the presence of a strong audio signal makes a temporal or spectral neighborhood of weaker audio signal imperceptible. This means that the masking effect can be observed in time and frequency domain. Normally they are studied separately and known as simultaneous masking and temporal masking.

If two sounds occur simultaneously and one is masked by the other, this is referred to as simultaneous masking. A sound close in frequency to a louder sound is more easily masked than if it is far apart in frequency. For this reason, simultaneous masking is also

sometimes called frequency masking. It is important to differentiate between tone and noise maskers, because tonality of a sound also determines its ability to mask other sounds. A sinusoidal masker, for example, requires a higher intensity to mask a noise- like masker than a loud noise-like masker does to mask a sinusoid. Similarly, a weak sound emitted soon after the end of a louder sound is masked by the louder sound. In fact, even a weak sound just before a louder sound can be masked by the louder sound. These two effects are called forward and backward temporal masking, respectively. Temporal masking effectiveness attenuates exponentially from the onset and offset of the masker, with the onset attenuation lasting approximately 10 ms and the offset attenuation lasting approximately 50 ms.

It is of special interest for perceptual audio coding to have a precise description of all masking phenomena to compute a masking threshold that can be used to compress a digital signal. Using this, it is possible to reduce the SNR and therefore the number of bits. A complete masking threshold should be calculated using the principles of simultaneous masking and temporal masking and the frequency response of the ear. In the perceptual audio coding schemes, these masking models are often called psychoacoustic models.

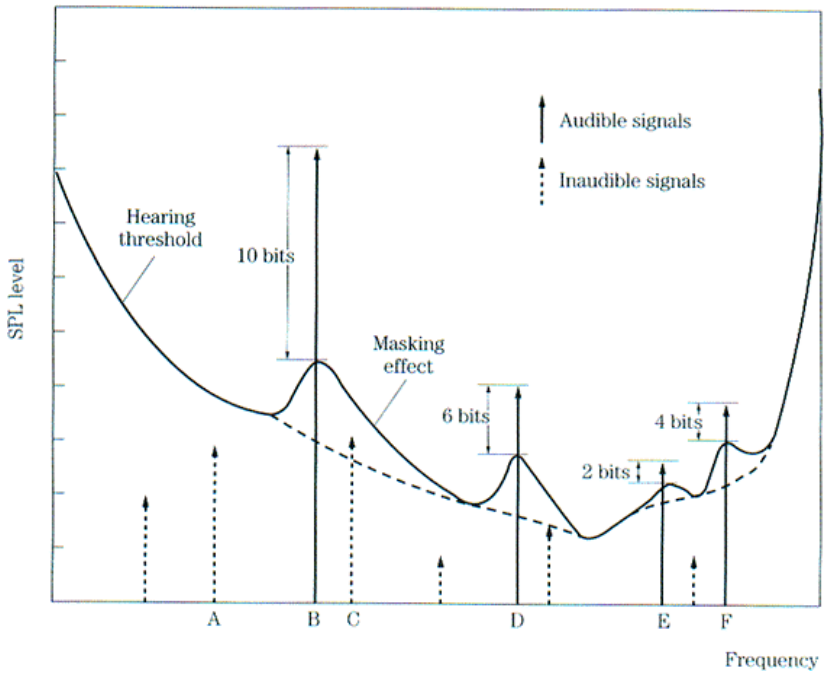


Fig (2) an example that shows how the auditory properties can be used to compress an digital audio signal.

Audio compression:

The idea of audio compression is to encode audio data to take up less storage space and less bandwidth for transmission. To meet this goal different methods for compression have been designed. Just like every other digital data compression, it is possible to classify them into two categories: lossless compression and lossy compression.

Lossless compression:

Lossless compression in audio is usually performed by waveform coding techniques. These coders attempt to copy the actual shape of the analog signal, quantizing each sample using different types of quantization. These techniques attempt to approximate the waveform, and, if a large enough bit rate is available they get arbitrary close to it. A popular waveform coding technique, that is considered uncompressed audio format, is the pulse code modulation (PCM), which is used by the Compact Disc Digital Audio (or simply CD). The quality of CD audio signals is referred to as a standard for hi-fidelity. CD audio signals are sampled at 44.1 kHz and quantized using 16 bits/sample Pulse Code Modulation (PCM) resulting in a very high bit rate of 705 kbps.

As mentioned before, human perception of sound is affected by SNR, because adding noise to a signal is not as noticeable if the signal energy is large enough. When digitalize an audio signal, ideally SNR could to be constant for al quantization levels, which requires a step size proportional to the signal value. This kind of quantization can be done using a logarithmic compander (compressor-expander). Using this technique it is possible to reduce the dynamic range of the signal, thus increasing the coding efficiency, by using fewer bits. The two most common standards are the μ-law and the A-law, widely used in telephony.

Other lossless techniques have been used to compress audio signals, mainly by finding redundancy and removing it or by optimizing the quantization process. Among those techniques it is possible to find Adaptative PCM and Differential quantization. Other lossless techniques such as Huffman coding and LZW have been directly applied to audio compression without obtaining significant compression ratio.

Lossy compression:

Opposed to lossless compression, lossy compression reduces perceptual redundancy; i.e. sounds which are considered perceptually irrelevant are coded with decreased accuracy or not coded at all. In order to do this, it is better to have scalar frequency domains coders, because the perceptual effects of masking can be more easily implemented in frequency domain by using subband coding.

Using the properties of the auditory system we can eliminate frequencies that cannot be perceived by the human ear, i.e. frequencies that are too low or too high are eliminated, as well as soft sounds that are drowned out by loud sounds. In order to determine what information in an audio signal is perceptual irrelevant, most lossy compression algorithms use transforms such as the Modified Discrete Cosine Transform (MDCT) to convert time domain sampled waveforms into a frequency domain. Once transformed into the frequency domain, frequencies component can be digitally allocated according to how audible they are (i.e. the number of bits can be determined by the SNR). Audibility of spectral components is determined by first calculating a masking threshold, below which it is estimated that sounds will be beyond the limits of human perception (see 2.1 on this report).

Briefly, the modified discrete cosine transform (MDCT) is a Fourier-related transform with the additional property of being lapped. It is designed to be performed on consecutive blocks of a larger data set, where subsequent blocks are overlapped so that the last half of one block coincides with the first half of the next block. This overlapping, in addition to the energy-compaction qualities of the DCT, makes the MDCT especially attractive for signal compression applications, since it helps to avoid artifacts stemming from the block boundaries.

MPEG Audio coding standards:

Moving Pictures Experts Group (MPEG) is an ISO/IEC group charged with the development of video and audio encoding standards. MPEG audio standards include an elaborate description of perceptual coding, psychoacoustic modeling and implementation issues. It is interesting for our report to mention some brief comments

on these audio coders, because some of the features of the wavelet-based audio coders are based in those models.

(a) MP1 (MPEG audio layer-1): Simplest coder/decoder. It identifies local tonal components based on local peaks of the audio spectrum.

(b) MP2 (MPEG audio layer-2): It has an intermediate complexity. It uses data from the previous two windows to predict, via linear interpolation, the component of the current window. This is based on the fact that tonal components, being more predictable, have higher tonality indices.

(c) MP3 (MPEG audio layer-3). Higher level of complexity. Not only includes masking in time domain but also a more elaborated psychoacoustic model, MDCT decomposition, dynamic allocation and Huffman coding.

All three layers of MPEG-1 use a polyphase fiterbank for signal decomposition into 32 equal width subbands. This is a computational simple solution and provides reasonable time-frequency resolution. However it is known that this approach has three notable deficiencies:

- Equal subbands do not reflect the critical bands of noise masking, and then the quantization error cannot be tuned properly.
- Those filter banks and their inverses do not yield perfect reconstruction, introducing error even in the absence of quantization error.
- Adjacent filter banks overlap, then a single tone can affect two filter banks.

These problems have been fixed by a new format which is considered the successor of the MP3 format: AAC (Advanced Audio Coding) defined in MPEG-4 Part 3 (with an extension .m4a or namely MP4 audio).

(d) M4A: AAC (MPEG-4 Audio): Similar to MP3 but it increases the number of subbands up to 48 and fix some issues in the previous perceptual model. It has higher coding efficiency for stationary and transient signals, providing a better and more stable quality than MP3 at equivalent or slightly lower bitrates.

Speech compression:

Speech signals has unique properties that differ from a general audio/music signals. First, speech is a signal that is more structured and band-limited around 4kHz. These two facts can be exploited through different models and approaches and at the end, make it easier to compress. Many speech compression techniques have been efficiently applied. Today, applications of speech compression (and coding) involve real time processing in mobile satellite communications, cellular telephony, internet telephony, audio for videophones or video teleconferencing systems, among others. Other applications include also storage and synthesis systems used, for example, in voice mail systems, voice memo wristwatches, voice logging recorders and interactive PC software.

Basically speech coders can be classified into two categories: waveform coders and analysis by synthesis vocoders. The first was explained before and are not very used for speech compression, because they do not provide considerable low bit rates. They are mostly focused to broadband audio signals. On the other hand, vocoders use an entirely different approach to speech coding, known as parametric coding, or analysis by synthesis coding where no attempt is made at reproducing the exact speech waveform at the receiver, but to create perceptually equivalent to the signal. These systems provide much lower data rates by using a functional model of the human speaking mechanism at the receiver. Among those, perhaps one of the most popular techniques is called Linear Predictive Coding (LPC) vocoder. Some higher quality vocoders include RELP (Residual Excited Linear Prediction) and CELP (Code Excited Linear Prediction). There are also lower quality vocoders that give very low bit rate such as Mixed Excitation vocoder, Harmonic coding vocoder and Waveform interpolation coders.

Evaluating compressed audio:

When evaluating the quality of compressed audio it is also convenient to differentiate between speech signals and general audio/music signals. Even though speech signals have more detailed methods to evaluate the quality of a compressed signal (like intelligibility tests), both audio/music and speech share one of the most common methods: acceptability tests. These tests are the most general way to evaluate the quality of an audio/speech signal, and they are mainly determined by asking users their preferences for different utterances. Among those tests, Mean Opinion Score (MOS) test is the most used one. It is a subjective measurement that is derived entirely by people listening to the signals and scoring the results

from 1 to 5, with a 5 meaning that speech quality is perfect or “transparent”. The test procedure requires carefully prepared and controlled test conditions. The term “transparent quality” means that most of the test samples are indistinguishable from the original for most of the listeners. The term was defined by the European Broadcasting Union (EBU) in 1991 and statistically implemented in formal listening tests since then.

Finally, it is necessary to emphasize that the fact that measures of quality of audio signal does not have an objective measure that we can extract directly from the signal (such mean square error), make it more difficult to evaluate it. This is because subjective evaluations require a large number of test samples and special conditions during the evaluation.

Different Compression Techniques

Different types of compression techniques are available they are lossless compression and lossy compression techniques. Redundancy information present in audio signal will removed in loss less compression. It is disadvantages such as it doesn’t give the constant output data rate and very small compression ratio, and advantage isit can be applied

To any data stream. In loss compression the information is irrelevant in that the receiver will not able to recognize the missing.

Wavelet Transforms:

The wavelet theory allows a very general and flexible description to transform signal from time domain to time-frequency domain, so called time-scale domain. Wavelet transform uses short window for high. Wavelet Transform uses short window for high frequencies, leading to a good time resolution and larger windows for low frequencies leading to a good frequency resolution.

3. WAVELET REPRESENTATION FOR AUDIO SIGNALS

Wavelet compression is a form of data compression well suited for audio compression, video compression,image compression. Wavelet compression methods are adequate for representing transients. Such as percussion sounds in audio, high-frequency components in 2-D images. This means transient elements of data signal can be represented by a smaller amount of information that would be the case of some, Reverse Biorthogonal Filters, Discrete Meyer, Harr.

Wavelet Representation For Audio Signal:

A wavelet transform can be defined as a "small wave" that has its energy concentrated in time, and it provides a tool for the analysis of transient, non-stationary or time varying phenomenon. It has oscillating wave like property. Wavelet is a waveform of limited duration having an average value zero. They are localized in space. Wavelet transform provides a time-frequency representation of the signal. In wavelet transform, the signal is decomposed into set of basic functions also known as WAVELETS. Wavelets with large number of vanishing moments are useful for this audio compression method, because if a wavelet with a large number of vanishing moments is used, a precise specification of the pass bands of each sub band in the wavelet
decomposition is possible. Thus, we can approximate the critical band division given by the auditory system with this structure and quantization noise power could be integrated.

Wavelet Based Compression Techniques:

Wavelets concentrate speech signals into a few neighbouring coefficients. By taking the wavelet transform of a signal, many of its" coefficients will either be zero or have negligible magnitudes. Data compression can then be done by treating the small valued coefficients as insignificant data and discarding them. Compressing a speech signal using wavelets involves the following stages.

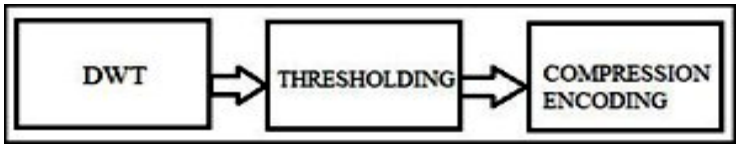


Fig (3): Compression System Design

a) Thresholding:

After the coefficients are received from different transforms, thresholding is done. Very few DCT coefficients represent 99% of signal energy; hence thresholding is calculated and applied to the coefficients. Coefficients having values less than threshold values are removed.

b) Quantization:

It is a process of mapping a set of continuous valued data to a set of discrete valued data. The aim of quantization is to reduce the information found in threshold coefficients. This process makes sure that it produces minimum errors. We basically perform uniform quantization process.

c) Encoding:

We use different encoding techniques like decomposition using N equal frames Encoding method is used to remove data that are repetitively occurring. In encoding we can also reduce the number of coefficients by removing the redundant data. Encoding can use any of the two compression techniques, lossless or lossy. This helps in reducing the bandwidth of the signal hence compression can be achieved.The compressed speech signal can be reconstructed to form the original signal by DECODING followed by DE-QUANTIZATION. This would reproduce the original signal.

4. WAVELET PACKET APPROACH

The main goal of this new algorithm is to compress high quality audio maintaining transparent quality at low bit rates. In order to do this, the authors explored the usage of an adaptative wavelet packet decomposition. Several key issues are considered as follows:

- Design a subband structure for wavelet representation of audio signals. This design also determines the computational complexity of the algorithm for each frame;
- Design a scheme for efficient bit allocation, which depends on the temporal resolution of the decomposition

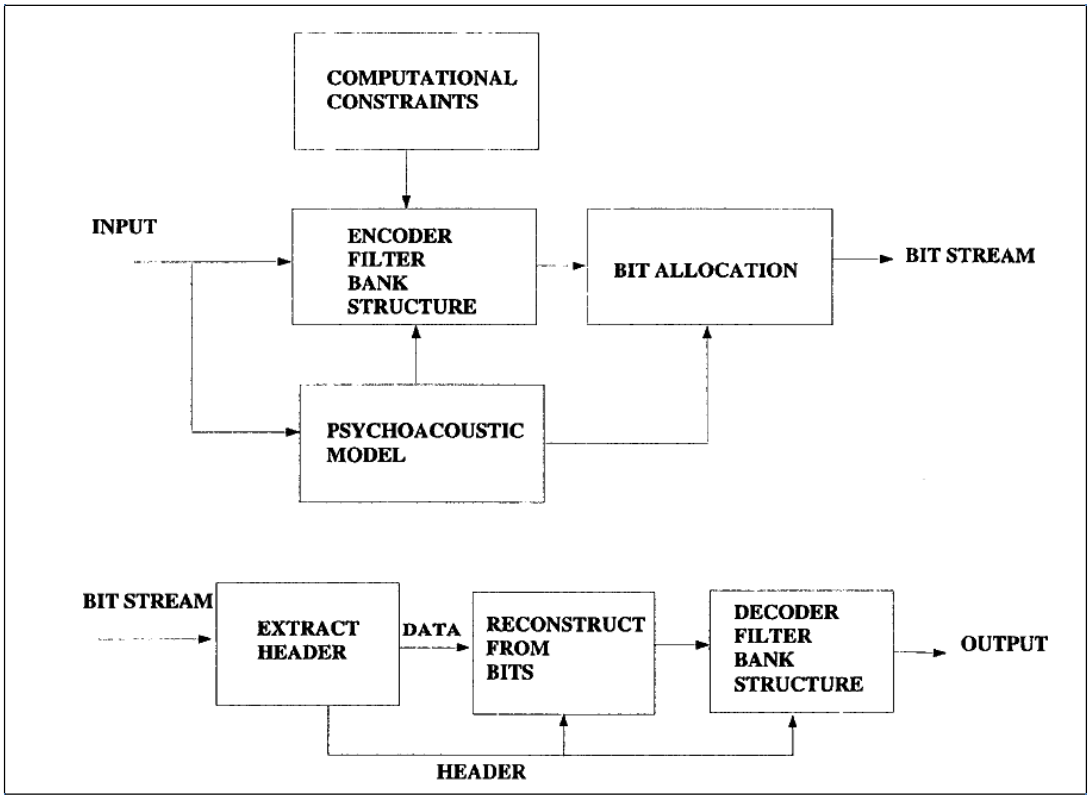


Fig (4) Block diagram of the described encoder/decoder

Wavelet packet representation:

Given a wavelet packet structure, a complete tree structured filter bank is considered. Once we find the “best basis” for this application, a fast implementation exists for determining the coefficients with respect to the basis. However, in the “best basis” approach, they do not subdivide every subband until the last level. The decision of whether to subdivide is made based on a reasonable criterion according to the application (further decomposition implies less temporal resolution).

The cost function, which determines the basis selection algorithm, will be a constrained minimization problem. The idea is to minimize the cost due to the bit rate given the filter bank structure, using as a variable the estimated computational complexity at a particular step of the algorithm, limited by the maximum computations permitted. At every stage, a decision is made whether to decompose the subband further based on this cost function.

Another factor that influences this decomposition is the tradeoff in resolution. If it is decomposed further down, it will sacrifice temporal resolution for frequency resolution. The last level of decomposition has minimum temporal resolution and has the best frequency resolution. The decision on whether to decompose is carried out top-down instead of bottom-up. Following that way, it is possible to evaluate the signal at a better temporal resolution before the decision to decompose. It is proved in this paper that the proposed algorithm yields the “best basis” (minimum cost) for the given computational complexity and range of temporal resolution.

5. INTRODUCTION TO MATLAB

What Is MATLAB?

MATLAB is a high-performance language for technical computing. It integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation. Typical uses include:

- Math and computation
- Algorithm development
- Modeling, simulation, and prototyping
- Data analysis, exploration, and visualization
- Scientific and engineering graphics
- Application development, including Graphical User Interface building

MATLAB is an interactive system whose basic data element is an array that does not require dimensioning. This allows you to solve many technical computing problems, especially those with matrix and vector formulations, in a fraction of the time it would take to write a program in a scalar noninteractive language such as C or Fortran.

The name MATLAB stands for matrix laboratory. MATLAB was originally written to provide easy access to matrix software developed by the LINPACK and EISPACK projects, which together represent the state-of-the-art in software for matrix computation.

MATLAB has evolved over a period of years with input from many users. In university environments, it is the standard instructional tool for introductory and advanced courses in mathematics, engineering, and science. In industry, MATLAB is the tool of choice for high-productivity research, development, and analysis.

MATLAB features a family of application-specific solutions called toolboxes. Very important to most users of MATLAB, toolboxes allow you to *learn* and *apply* specialized technology. Toolboxes are comprehensive collections of MATLAB functions (M-files) that extend the MATLAB environment to solve particular classes of problems. Areas in which toolboxes are available include signal processing, control systems, neural networks, fuzzy logic, wavelets, simulation, and many others.

The MATLAB System

The MATLAB system consists of five main parts:

The MATLAB language.

This is a high-level matrix/array language with control flow statements, functions, data structures, input/output, and object-oriented programming features. It allows both "programming in the small" to rapidly create quick and dirty throw-away programs, and "programming in the large" to create complete large and complex application programs.

The MATLAB working environment.

This is the set of tools and facilities that you work with as the MATLAB user or programmer. It includes facilities for managing the variables in your workspace and importing and exporting data. It also includes tools for developing, managing, debugging, and profiling M-files, MATLAB's applications.

Handle Graphics.

This is the MATLAB graphics system. It includes high-level commands for two-dimensional and three-dimensional data visualization, image processing, animation, and presentation graphics. It also includes low-level commands that allow you to fully customize the appearance of graphics as well as to build complete Graphical User Interfaces on your MATLAB applications.

The MATLAB mathematical function library.

This is a vast collection of computational algorithms ranging from elementary functions like sum, sine, cosine, and complex arithmetic, to more sophisticated functions like matrix inverse, matrix eigenvalues, Bessel functions, and fast Fourier transforms.

The MATLAB Application Program Interface (API).

This is a library that allows you to write C and Fortran programs that interact with MATLAB. It include facilities for calling routines from MATLAB (dynamic linking), calling MATLAB as a computational engine, and for reading and writing MAT-files.

A minimum MATLABsession:

The goal of this *minimum* session (also called *starting* and *exiting* sessions) is to learn the first steps:

- How to log on
- Invoke MATLAB
- Do a few simple calculations
- How to quit MATLAB

Starting MATLAB:

After logging into your account, you can enter MATLAB by double-clicking on the MATLAB shortcut *icon* (MATLAB 7.0.4) on your Windows desktop. When you start MATLAB, a special window called the MATLAB desktop appears. The desktop is a window that contains *other* windows. The major tools within or accessible from the desktop are:

- The COMMAND WINDOW
- The COMMAND HISTORY
- The WORKSPACE
- The CURRENT DIRECTORY
- The HELP BROWSER
- The START button

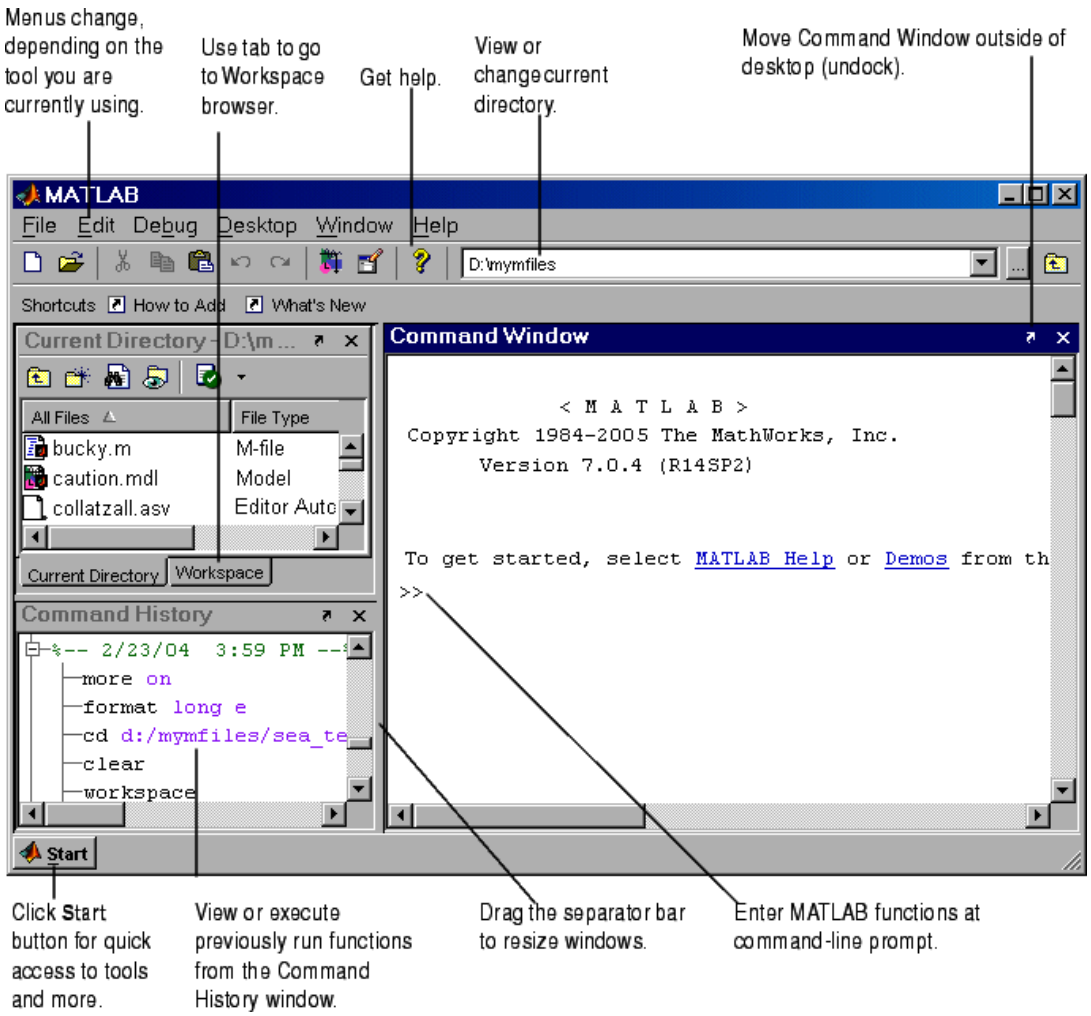


Fig (5)The graphical interface to the MATLAB workspace

When MATLAB is started for the first time, the screen looks like the one that shown in the Figure 1.1. This illustration also shows the default configuration of the MATLAB desktop. You can customize the arrangement of tools and documents to suit your needs.

Now, we are interested in doing some simple calculations. We will assume that you have sufficient understanding of your computer under which MATLAB is being run.

You are now faced with the MATLAB desktop on your computer, which contains the prompt (>>) in the Command Window. Usually, there are 2 types of prompt:

>> for full version

EDU> for educational version

NOTE: To simplify the notation, we will use this prompt, >>, as a standard prompt sign, though our MATLAB version is for educational purpose.

Quitting MATLAB

To end your MATLAB session, type **quit** in the Command Window, or select **File Exit MATLAB** in the desktop main menu.

6. IMPLEMENTATION METHODOLGY

An audio signal sample is taken and analysed using MATLAB for frequency and amplitude. Haar and Daubenches algorithms are applied on the speech signal and the audio is compressed. Audio sizes before and after compression are compared.

The following parameters are compared by the program: Peak signal-to-noise ratio (PSNR), normalised root-mean-square error (NRMSE) and compression ratios.

Haar wavelet algorithm performs the following functions:

1. Selects audio and finds actual signal size
2. Finds amplitude and frequency
3. Creates frame
4. Decomposes the signal spectrum into wavelet
5. Creates psychoacoustic model
6. Inspects the spectrum and finds tones maskers
7. Applies mu-law of compression
8. Finds and corrects offset
9. Rewrites wave
10. Finds the size of compressed signal

Daubenches wavelet transform performs the following functions:

1. Selects audio and finds the actual signal size
2. Finds amplitude and frequency
3. Chooses a block size
4. Changes compression percentages
5. Initialises compressed matrix
6. Does compression using inverse discrete cosine transform (IDCT)
7. Rewrites signal
8. Finds the size of compressed signal

MATLAB code file Audio Compression.m implements Haar wavelet and Audio Compression2.m file implements Daubenches wavelet. In this example, Windows XP Startup.wav is the sample audio file used for compression.

Comparison of performance metrics such as PSNR, MSE and compression ratio shows that Daubenches algorithm is best suited for lossless compression of speech signals. Advantages of audio compression are less storage space and associated cost, and faster data transfer.

There are several techniques for data compression. You should follow lossless compression technique as lossy audio compression results in data loss. Image and video can be compressed in a similar way.

Main features of the implementation

The Matlab implementation includes the following features:

(a) Signal division and processing using small frames

(b) Discrete wavelet decomposition of each frame

(c) Compression in the wavelet domain

(d) A psychoacoustic model

(e) Non linear quantization over the wavelet coefficient using the psychoacoustic model

(f) Signal reconstruction

(g) Main output: Audio files

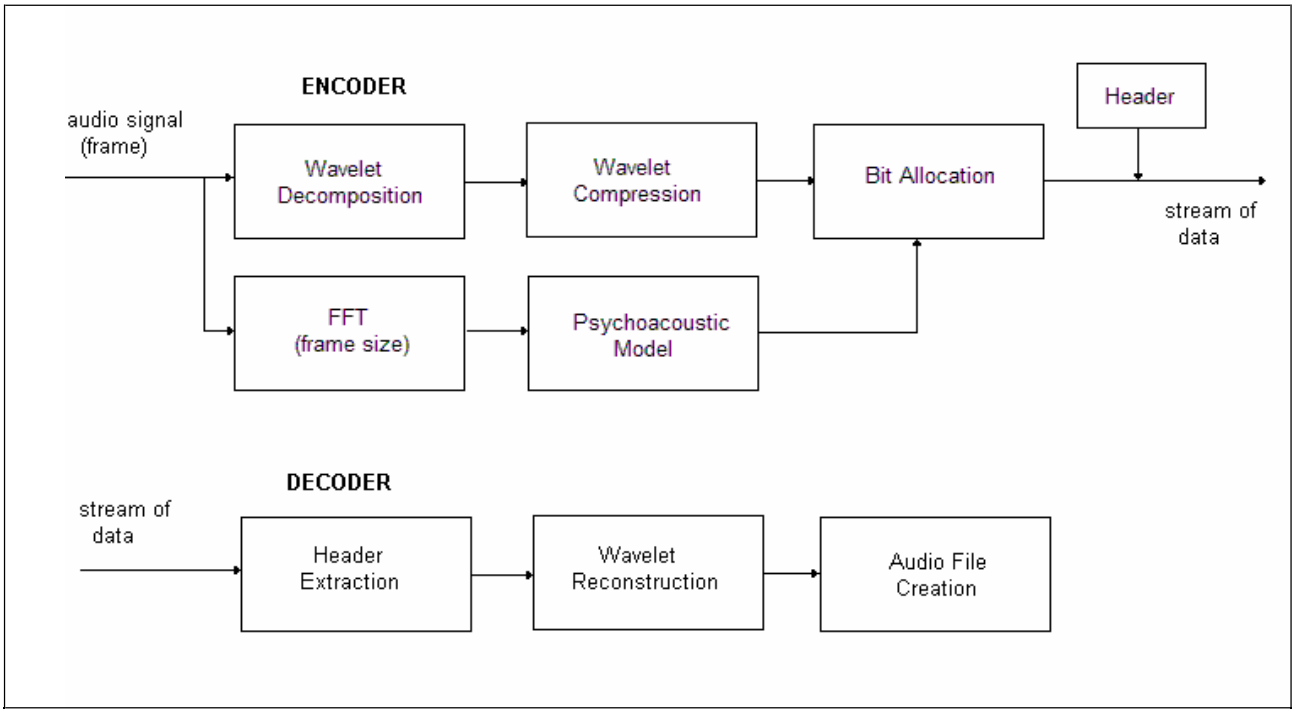


Fig 6(a).Block Diagram of the Matlab Implementation

Considerations:

Even though it is more convenient to implement the ideas, some of the suggested steps require a complicated implementation. Therefore, a few modifications and considerations have been included to the design of this MATLAB simulation:

(a) No search for optimal basis is performed:

Even though this is one of the key point, its implementation is requires a large programming design, and that is out of the scope of this demonstration. To compensate that, another compression technique has been used. This is based in the known discrete wavelet decomposition compression that uses an optimal global threshold. This technique has been successfully used in audio compression.the best results were observed when using a *Daubechies* wavelet with 10 vanishing moments (dB10 in matlab) and 5 levels of decomposition. These choices will overcome the lack of an optimal basis search.

(b) Non overlapping frames are included

This implementation does not have overlapping frames to avoid computational complexity. The frame size is given 2048 samples per frame.

(c) The psychoacoustic model is simplified

Due to the complexity associated with the construction of a psychoacoustic model, a simplified version was considered. This model can only detect masking tones in the signal, and gives a general threshold for all the frequencies

An example of this simplified psychoacoustic model is shown in the following figure. The main tonal components are detected. The power average of this components is used as masking threshold for every frequency.

(d) No new audio format was design

Even though this simplified matlab implementation performs compression over the audio signal, this is not reflected in the size of the new audio files. This is due to the fact that a new format design was not considered, so the *wavwrite* command was used to create the audio files (.wav). The compression ratio for each case is calculated using other variables of the simulation.

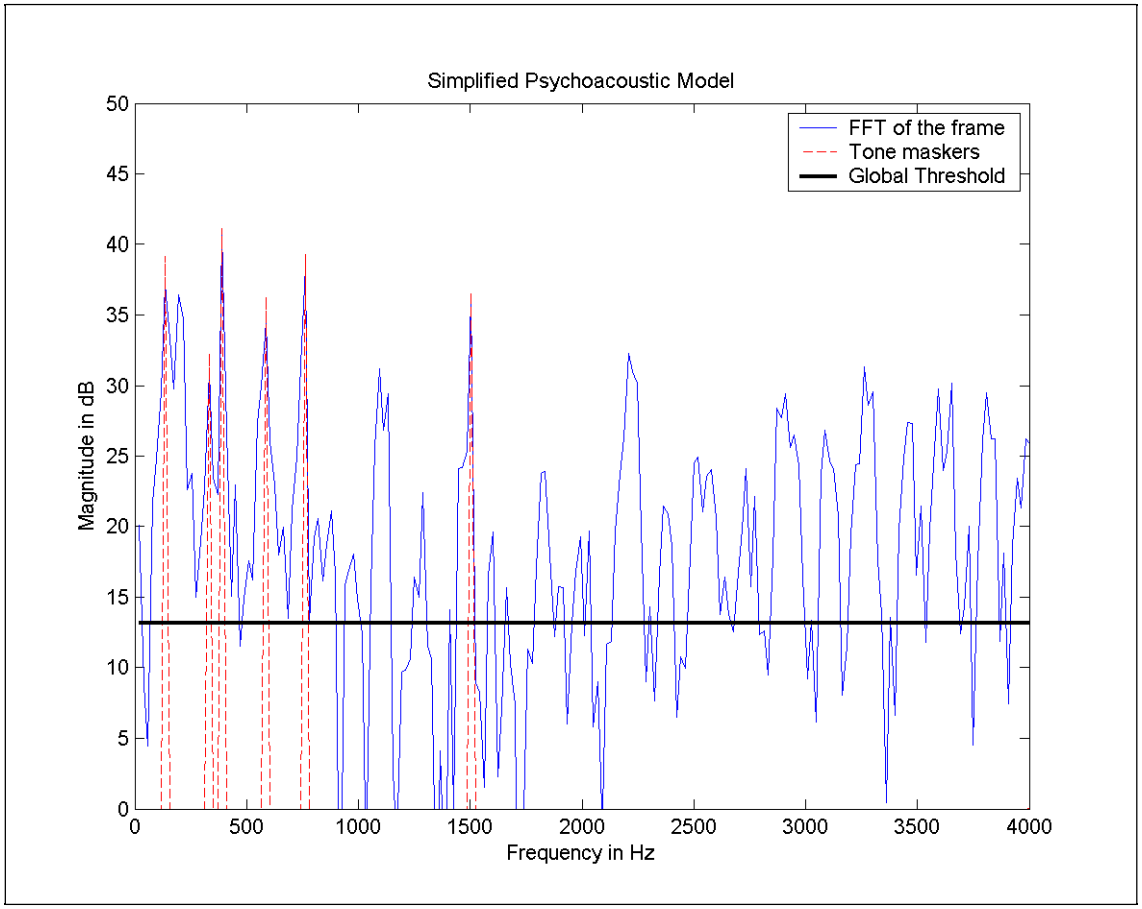


Fig 6(b) Tone masker detection in a frame - Matlab implementation.

(e) A lossless compression at the end was tested and suppressed:

Arithmetic compression (similar to Huffman coding) was tested in this simulation but was suppressed at the end because it made the simulation too slow. However its performance is considered in the results.

Equations:

Peak Signal to Noise Ratio PSNR = $10 \log_{10} \frac{NX^2}{\|x-r\|^2}$

N is the length of the reconstructed signal, X is the maximum absolute square value of the Signal x and $\|x - r\|^2$ is the energy of the difference between the original and reconstructed Signals.

2. Mean Square Error MSE = $\sqrt{\frac{\{x(n)-r(n)\}^2}{\{x(n)-\mu_A(n)\}^2}}$

x (n) is the speech signal, r(n) is the reconstructed signal, and μx(n) is the mean of the speech Signal.

3. Compression Ratio C = $\frac{Length(x(n))}{Length(cWc)}$

cWC is the length of the compressed wavelet transform vector.

4. HAAR Wavelet Algorithm

For a given signal x, the output of the μ-law compressor is,

$$y = \frac{V \log(1+ \mu |x| / V)}{\log(1 + \mu)} \text{sgn}(x)$$

Where V is the maximum value of signal xμ is the μ-law parameter.

7. MATLAB CODE

AudioCompression.m

```
function varargout = AudioCompression1(varargin)
% AUDIOCOMPRESSION MATLAB code for AudioCompression.fig
%      AUDIOCOMPRESSION, by itself, creates a new AUDIOCOMPRESSION or
raises the existing
%      singleton*.
%
%      H = AUDIOCOMPRESSION returns the handle to a new AUDIOCOMPRESSION
or the handle to
%      the existing singleton*.
%
%      AUDIOCOMPRESSION('CALLBACK',hObject,eventData,handles,...) calls
the local
%      function named CALLBACK in AUDIOCOMPRESSION.M with the given input
arguments.
%
%      AUDIOCOMPRESSION('Property','Value',...) creates a new
AUDIOCOMPRESSION or raises the
%      existing singleton*.  Starting from the left, property value pairs
are
%      applied to the GUI before AudioCompression_OpeningFcn gets called.
An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to AudioCompression_OpeningFcn via
varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only
one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help AudioCompression

% Last Modified by GUIDE v2.5 21-Nov-2014 17:35:02

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @AudioCompression_OpeningFcn, ...
                   'gui_OutputFcn',  @AudioCompression_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
```

```
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before AudioCompression is made visible.
function AudioCompression_OpeningFcn(hObject, eventdata, handles,
varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to AudioCompression (see VARARGIN)

% Choose default command line output for AudioCompression
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes AudioCompression wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = AudioCompression_OutputFcn(hObject, eventdata,
handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global file_name;
%guidata(hObject,handles)
file_name=uigetfile({'*.wav'},'Select an Audio File');
fileinfo = dir(file_name);
SIZE = fileinfo.bytes;
Size = SIZE/1024;
[x,Fs,bits] = wavread(file_name);
xlen=length(x);
t=0:1/Fs:(length(x)-1)/Fs;
set(handles.text2,'string',Size);
%plot(t,x);
```

```
axes(handles.axes3) % Select the proper axes
plot(t,x)
set(handles.axes3,'XMinorTick','on')
grid on

% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global file_name;
if(~ischar(file_name))
   errordlg('Please select Audio first');
else
[x,Fs,bits] = wavread(file_name);
xlen=length(x);
t=0:1/Fs:(length(x)-1)/Fs;
wavelet='haar';
level=5;
frame_size=2048;
psychoacoustic='on '; %if it is off it uses 8 bits/frame as default
wavelet_compression = 'on ';
heavy_compression='off';
compander='on ';
quantization ='on ';

% ENCODER

step=frame_size;
N=ceil(xlen/step);
%computational variables
Cchunks=0;
Lchunks=0;
Csize=0;
PERF0mean=0;
PERFL2mean=0;
n_avg=0;
n_max=0;
n_0=0;
n_vector=[];
for i=1:1:N
if (i==N);
frame=x([(step*(i-1)+1):length(x)]);
else
frame=x([(step*(i-1)+1):step*i]);
end
%wavelet decomposition of the frame
[C,L] = wavedec(frame,level,wavelet);
%wavelet compression scheme
if wavelet_compression=='on '
[thr,sorh,keepapp] = ddencmp('cmp','wv',frame);
if heavy_compression == 'on '
thr=thr*10^6;
```

```
end
[XC,CXC,LXC,PERF0,PERFL2] = wdencmp('gbl',C, L,
wavelet,level,thr,sorh,keepapp);
C=CXC;
L=LXC;
PERF0mean=PERF0mean + PERF0;
PERFL2mean=PERFL2mean+PERFL2;
end
%Psychoacoustic model
if psychoacoustic=='on '
P=10.*log10((abs(fft(frame,length(frame)))).^2);
Ptm=zeros(1,length(P));
%Inspect spectrum and find tones maskers
for k=1:1:length(P)
if ((k<=1) | (k>=250))
bool = 0;
elseif ((P(k)<P(k-1)) | (P(k)<P(k+1))),
bool = 0;
elseif ((k>2) & (k<63)),
bool = ((P(k)>(P(k-2)+7)) & (P(k)>(P(k+2)+7)));
elseif ((k>=63) & (k<127)),
bool = ((P(k)>(P(k-2)+7)) & (P(k)>(P(k+2)+7)) & (P(k)>(P(k-3)+7)) &
(P(k)>(P(k+3)+7)));
elseif ((k>=127) & (k<=256)),
bool = ((P(k)>(P(k-2)+7)) & (P(k)>(P(k+2)+7)) & (P(k)>(P(k-3)+7)) &
(P(k)>(P(k+3)+7)) & (P(k)>(P(k-4)+7)) & (P(k)>(P(k+4)+7)) &(P(k)>(P(k-
5)+7)) & (P(k)>(P(k+5)+7)) & (P(k)>(P(k-6)+7)) &(P(k)>(P(k+6)+7)));
else
bool = 0;
end
if bool==1
Ptm(k)=10*log10(10.^(0.1.*(P(k-1)))+10.^(0.1.*(P(k)))+10.^(0.1.*P(k+1)));
end
end
sum_energy=0;
for k=1:1:length(Ptm)
sum_energy=10.^(0.1.*(Ptm(k)))+sum_energy;
end
E=10*log10(sum_energy/(length(Ptm)));
SNR=max(P)-E;
n=ceil(SNR/6.02);
if n<=3
n=4;
n_0=n_0+1;
end
if n>=n_max
n_max=n;
end
n_avg=n+n_avg;
n_vector=[n_vector n];
end
%Compander(compressor)
if compander=='on '
```

```
Mu=255;
C = compand(C,Mu,max(C),'mu/compressor');
end
%Quantization
if quantization=='on '
if psychoacoustic=='off'
n=8;
end
partition = [min(C):((max(C)-min(C))/2^n):max(C)];
codebook = [1 min(C):((max(C)-min(C))/2^n):max(C)];
[index,quant,distor] = quantiz(C,partition,codebook);
%find and correct offset
offset=0;
for j=1:1:N
if C(j)==0
offset=-quant(j);
break;
end
end
quant=quant+offset;
C=quant;
end
%Put together all the chunks
Cchunks=[Cchunks C];
Lchunks=[Lchunks L];
Csize=[Csize length(C)];
Encoder = round((i/N)*100); %indicator of progess
end
Cchunks=Cchunks(2:length(Cchunks));
%wavwrite(Cchunks,Fs,bits,'output1.wav')
Csize=[Csize(2) Csize(N+1)];
Lsize=length(L);
Lchunks=[Lchunks(2:Lsize+1) Lchunks((N-1)*Lsize+1:length(Lchunks))];
PERF0mean=PERF0mean/N; %indicator
PERFL2mean=PERFL2mean/N;%indicator
n_avg=n_avg/N;%indicator
n_max;%indicator
end_of_encoder='done';
xdchunks=0;
for i=1:1:N;
if i==N;
Cframe=Cchunks([((Csize(1)*(i-1))+1):Csize(2)+(Csize(1)*(i-1))]);
%Compander (expander)
if compander=='on '
if max(Cframe)==0
else
Cframe = compand(Cframe,Mu,max(Cframe),'mu/expander');
end
end
xd = waverec(Cframe,Lchunks(Lsize+2:length(Lchunks)),wavelet);
else
Cframe=Cchunks([((Csize(1)*(i-1))+1):Csize(1)*i]);
%Compander (expander)
```

```
if compander=='on '
if max(Cframe)==0
else
Cframe = compand(Cframe,Mu,max(Cframe),'mu/expander');
end
end
xd = waverec(Cframe,Lchunks(1:Lsize),wavelet);
end
xdchunks=[xdchunks xd];
Decoder = round((i/N)*100); %indicator of progess
end
xdchunks=xdchunks(2:length(xdchunks));
%distorsion = sum((xdchunks-x').^2)/length(x)
end_of_decoder='done';
%creating audio files with compressed schemes
wavwrite(xdchunks,Fs,bits,'output1.wav');
end_of_writing_file='done';%indicator of progess;
[x,Fs,bits] = wavread('output1.wav');
fileinfo = dir('output1.wav');
SIZE = fileinfo.bytes;
Size = SIZE/1024;
set(handles.text3,'string',Size)
xlen=length(x);
t=0:1/Fs:(length(x)-1)/Fs;
axes(handles.axes4) % Select the proper axes
plot(t,xdchunks)
set(handles.axes4,'XMinorTick','on')
grid on

[y1,fs1, nbits1,opts1]=wavread(file_name);
[y2,fs2, nbits2,opts2]=wavread('output1.wav');
[c1x,c1y]=size(y1);
[c2x,c2y]=size(y1);
if c1x ~= c2x
    disp('dimeonsions do not agree');
 else
 R=c1x;
 C=c1y;
  err = (sum(y1(2)-y2).^2)/(R*C);
 MSE=sqrt(err);
 MAXVAL=255;
  PSNR = 20*log10(MAXVAL/MSE);
  MSE= num2str(MSE);
  if(MSE > 0)
  PSNR= num2str(PSNR);
  else
PSNR = 99;
end
fileinfo = dir(file_name);
SIZE = fileinfo.bytes;
Size = SIZE/1024;
fileinfo1 = dir('output1.wav');
SIZE1 = fileinfo1.bytes;
```

```
Size1 = SIZE1/1024;

CompressionRatio = Size/Size1;

  set(handles.text14,'string',PSNR)
  set(handles.text16,'string',MSE)
  set(handles.text17,'string',CompressionRatio)

end


end


function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2 as a
double


% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1 as a
double


% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

AudioCompression2.m

```
function varargout = AudioCompression2(varargin)
% AUDIOCOMPRESSION2 MATLAB code for AudioCompression2.fig
%      AUDIOCOMPRESSION2, by itself, creates a new AUDIOCOMPRESSION2 or
raises the existing
%      singleton*.
%
%      H = AUDIOCOMPRESSION2 returns the handle to a new AUDIOCOMPRESSION2
or the handle to
%      the existing singleton*.
%
%      AUDIOCOMPRESSION2('CALLBACK',hObject,eventData,handles,...) calls
the local
%      function named CALLBACK in AUDIOCOMPRESSION2.M with the given input
arguments.
%
%      AUDIOCOMPRESSION2('Property','Value',...) creates a new
AUDIOCOMPRESSION2 or raises the
%      existing singleton*.  Starting from the left, property value pairs
are
%      applied to the GUI before AudioCompression2_OpeningFcn gets called.
An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to AudioCompression2_OpeningFcn via
varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only
one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help AudioCompression2

% Last Modified by GUIDE v2.5 21-Nov-2014 18:23:56

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @AudioCompression2_OpeningFcn, ...
                   'gui_OutputFcn',  @AudioCompression2_OutputFcn, ...
```

```
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before AudioCompression2 is made visible.
function AudioCompression2_OpeningFcn(hObject, eventdata, handles,
varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to AudioCompression2 (see VARARGIN)

% Choose default command line output for AudioCompression2
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes AudioCompression2 wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = AudioCompression2_OutputFcn(hObject, eventdata,
handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global file_name;
%guidata(hObject,handles)
file_name=uigetfile({'*.wav'},'Select an Audio File');
```

```
fileinfo = dir(file_name);
SIZE = fileinfo.bytes;
Size = SIZE/1024;
[x,Fs,bits] = wavread(file_name);
xlen=length(x);
t=0:1/Fs:(length(x)-1)/Fs;
set(handles.text12,'string',Size);
%plot(t,x);
axes(handles.axes1) % Select the proper axes
plot(t,x)
set(handles.axes1,'XMinorTick','on')
grid on

% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global file_name;
if(~ischar(file_name))
   errordlg('Please select Audio first');
else
[Data,Fs,bits] = wavread(file_name);
[Data, Fs, bits] = wavread('Windows XP Startup.wav');

%chosing a block size
windowSize = 8192;

%changing compression  percentages
samplesHalf = windowSize / 2;
samplesQuarter = windowSize / 4;
samplesEighth = windowSize / 8;

%initializing compressed matrice
DataCompressed2 = [];
DataCompressed4 = [];
DataCompressed8 = [];

%actual compression
for i=1:windowSize:length(Data)-windowSize
    windowDCT = dct(Data(i:i+windowSize-1));
    DataCompressed2(i:i+windowSize-1) = idct(windowDCT(1:samplesHalf),
windowSize);
    DataCompressed4(i:i+windowSize-1) = idct(windowDCT(1:samplesQuarter),
windowSize);
    DataCompressed8(i:i+windowSize-1) = idct(windowDCT(1:samplesEighth),
windowSize);
end

wavwrite(DataCompressed2,Fs,bits,'output3.wav')
[x,Fs,bits] = wavread('output3.wav');
fileinfo = dir('output3.wav');
SIZE = fileinfo.bytes;
```

```
Size = SIZE/1024;
xlen=length(x);
t=0:1/Fs:(length(x)-1)/Fs;
set(handles.text14,'string',Size);
%plot(t,x);
axes(handles.axes2) % Select the proper axes
plot(t,x)
set(handles.axes2,'XMinorTick','on')
grid on

wavwrite(DataCompressed4,Fs,bits,'output4.wav')
[x,Fs,bits] = wavread('output4.wav');
fileinfo = dir('output4.wav');
SIZE = fileinfo.bytes;
Size = SIZE/1024;
xlen=length(x);
t=0:1/Fs:(length(x)-1)/Fs;
set(handles.text16,'string',Size);
%plot(t,x);
axes(handles.axes3) % Select the proper axes
plot(t,x)
set(handles.axes3,'XMinorTick','on')
grid on

wavwrite(DataCompressed8,Fs,bits,'output5.wav')
[x,Fs,bits] = wavread('output5.wav');
fileinfo = dir('output5.wav');
SIZE = fileinfo.bytes;
Size = SIZE/1024;
xlen=length(x);
t=0:1/Fs:(length(x)-1)/Fs;
set(handles.text18,'string',Size);
%plot(t,x);
axes(handles.axes4) % Select the proper axes
plot(t,x)
set(handles.axes4,'XMinorTick','on')
grid on


[y1,fs1, nbits1,opts1]=wavread(file_name);
[y2,fs2, nbits2,opts2]=wavread('output3.wav');
[c1x,c1y]=size(y1);
[c2x,c2y]=size(y1);
if c1x ~= c2x
    disp('dimeonsions do not agree');
 else
 R=c1x;
 C=c1y;
  err = (sum(y1(2)-y2).^2)/(R*C);
 MSE=sqrt(err);
 MAXVAL=255;
   PSNR = 20*log10(MAXVAL/MSE);
  MSE= num2str(MSE);
```

```
if(MSE > 0)
  PSNR= num2str(PSNR);
   else
PSNR = 99;
end
fileinfo = dir(file_name);
SIZE = fileinfo.bytes;
Size = SIZE/1024;
fileinfo1 = dir('output3.wav');
SIZE1 = fileinfo1.bytes;
Size1 = SIZE1/1024;

CompressionRatio = Size/Size1;

  set(handles.text21,'string',PSNR)
  set(handles.text23,'string',MSE)
  set(handles.text24,'string',CompressionRatio)
end
end
```

8. RESULTS

- We RUN AudioCompresssion.m File. This is Method 1 using HAAR Wavelet.

- When we RUN in MATLAB, Following window will appear.(Fig 8(a))
- Select AUDIO file and then Press Compress Audio Button. Observe difference between Size of Audio. It is compressed then we will get output windows as shown in Fig 8(b). Compare audio file sizes before and after compression. The compressed audio file is generated as Output1.wav in the same path as the original source file (AudioCompresssion.m).

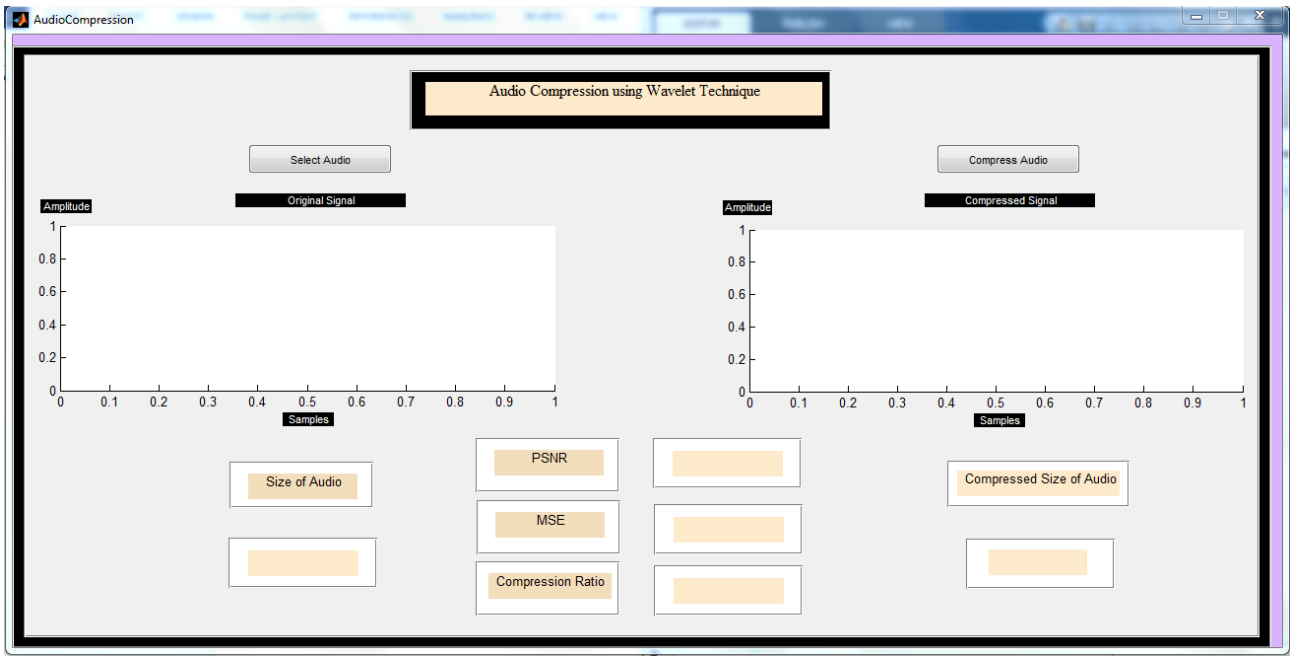


Fig 8(a): Program output (Haar wavelet)

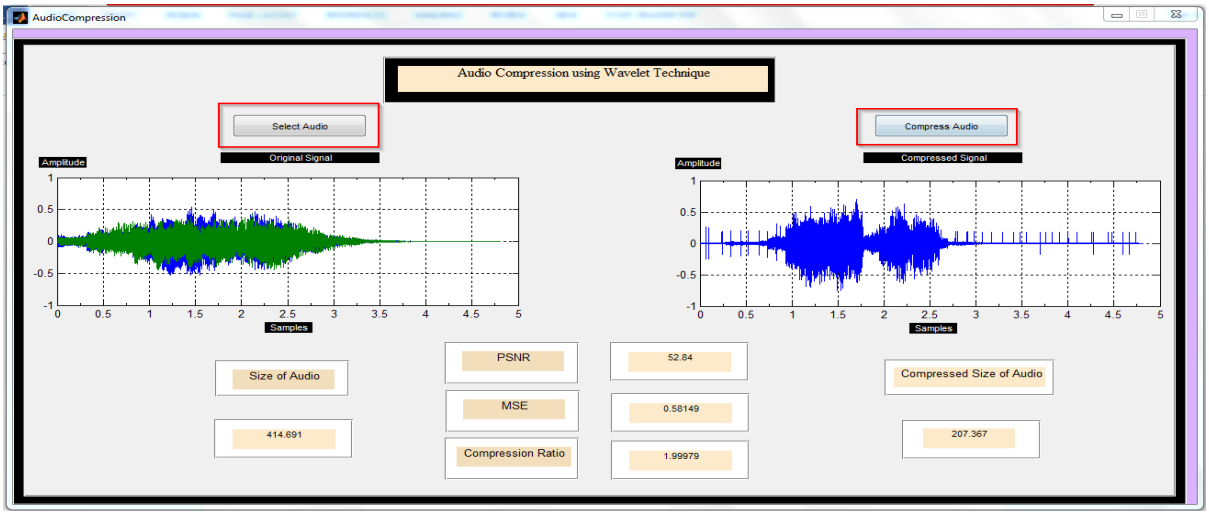
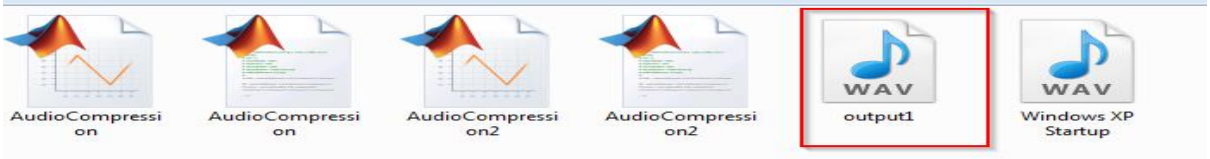


Fig 8(b) Graphical user interface for audio compression



- Now run AudioCompression2.m file. Select audio file and press Compress Audio button. We will get the program output window as shown in Fig 7(c). Observe the size of compressed audio. Here, three compressed audio files are generated and saved in the same path as the original source file. These three outputs correspond to different discrete cosine transform (DCT) window sizes of 2, 4 and 8. The percentage change in compressed files may differ depending on the quality of original file and size.

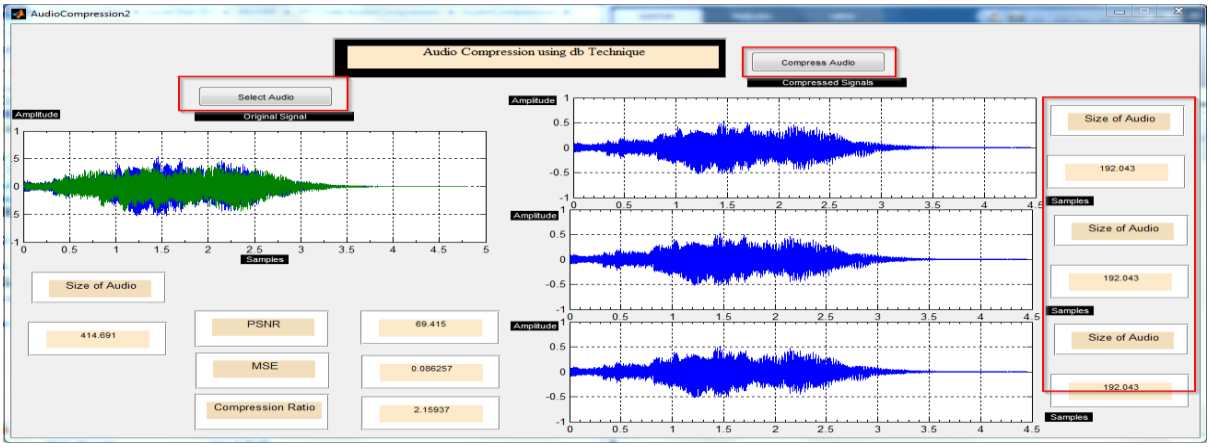


Fig.8(c) Program output (Daubenches wavelet)

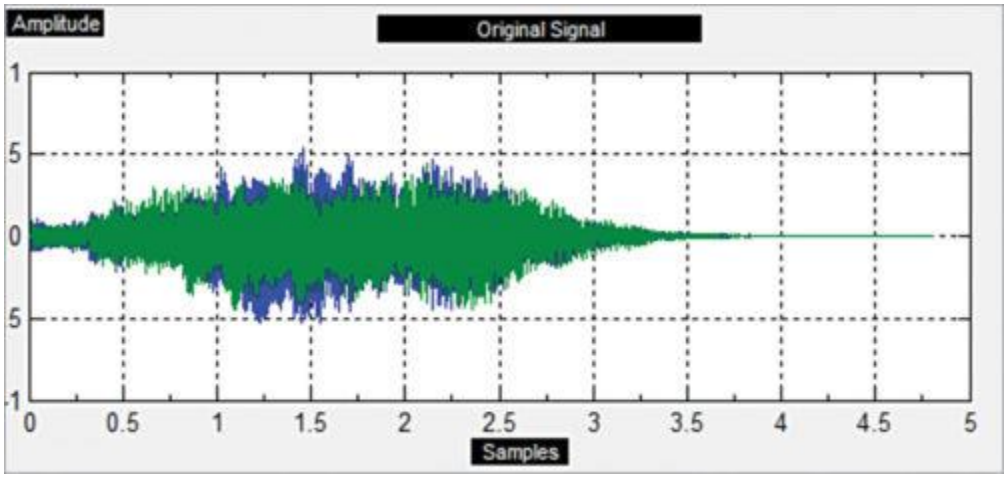


Fig.8(d) : Original audio signal (size: 414.691kB)

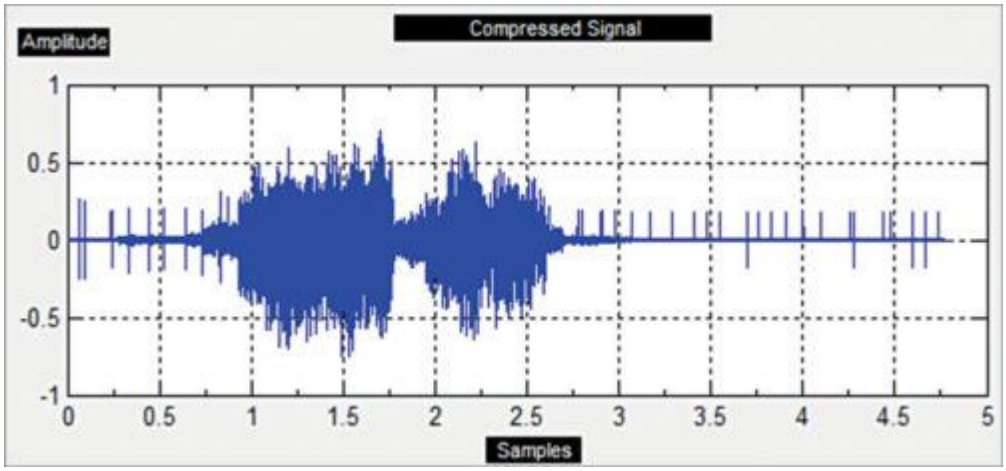


Fig.8(e) Haar-wavelet-decomposed audio signal (size: 207.367kB)

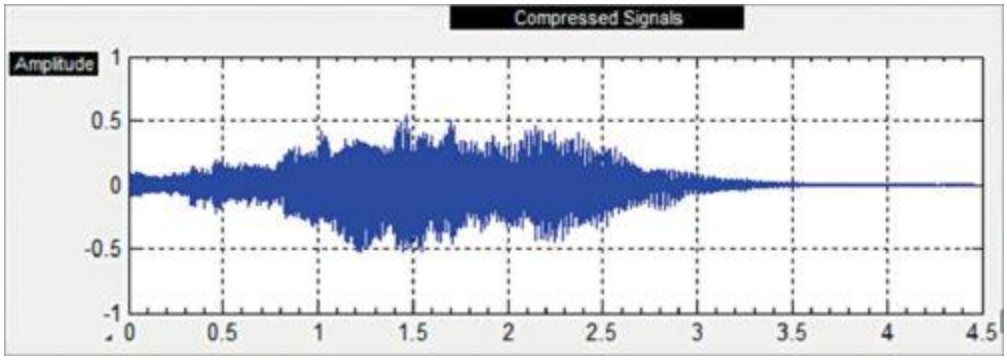


Fig. 8(f) Daubenches-wavelet-decomposed audio signal (size: 192.043kB)

9. CONCLUSION

We have implemented an audio compression procedure using wavelet transform technique. A MATLAB simulation of the project was successfully implemented, simplifying some of its features, but keeping its main structure and contributions. Our algorithm successfully compresses the audio which consists of speech signal. We have applied our algorithm on many audio files and found that it is successfully compressed.

10. BIBLIOGRAPHY

REFERENCES:

1. Understanding Matlab: A Textbook for Beginners by S.S. Alam S.N. Alam
2. MATLAB for Machine Learning by Giuseppe Ciaburro
3. “The ATMEGA Microcontroller and Embedded systems” by Mazidi.
4. MATLAB Deep Learning: With Machine Learning, Neural Networks and Artificial Intelligence by Phil Kim

WEBSITES

- www.mathworks.com
- www.wikipedia.org
- www.alldatasheets.com