

# Audio Processing and Compression

## 1. Abstract

A MATLAB-based implementation of an audio editor compressor is investigated. By varying the input parameters of the audio features, results were obtained that demonstrated the typical behaviour and benefits of compressor use. The parameters correspond to those found on specific hardware or software compressors. Adjustments to the MATLAB code were made to make the controls more understandable to audio practitioners.

## 2. Introduction

With the advent of new audio editing technologies, editing has become more accurate and easier over the years. Software and hardware programs are designed to help editors piece music or audio pieces together. These programs are generally referred to as digital audio workstations (DAWs). The idea behind audio editing is usually to take a piece of music and slice and dice it so that it is free of errors and consistent to listen to.

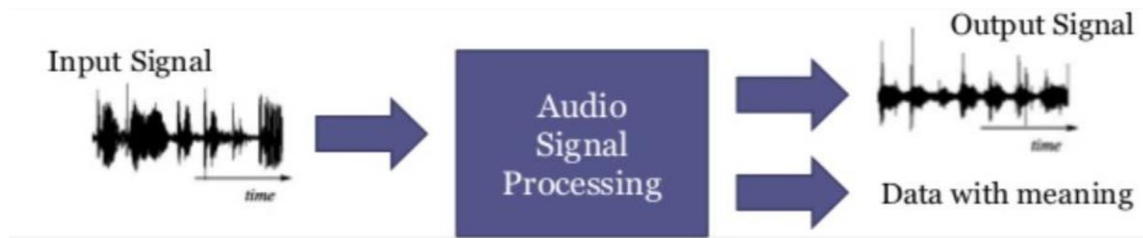


Fig. 1 Flowchart for Audio Signal Processing

Audio processing is an engineering field that focuses on the computational methods for intentionally altering auditory signals or sounds. Audio signal processing is a subfield of signal processing that is concerned with the electronic manipulation of audio signals. Audio signals are electronic representations of sound waves—longitudinal waves which travel through air, consisting of compressions and rarefactions.

Original file size	30% compressed	60% compressed	90% compressed
1.73 MB	1.2 MB	851 KB	443 KB
2.56 MB	1.9 MB	1.3 MB	300 KB
10.5 MB	7.1 MB	4 MB	1.2 MB
753 KB	511 KB	470 KB	405 KB

4.5 MB	3 MB	1.7 MB	700 KB
--------	------	--------	--------

Fig. 2 Comparison of file sizes after compression

### 3. Literature Survey

#### **1. Digital Audio Editing AES E-LIBRARY by McNally, Guy W.; Gaskell, Philip S.; Stirling, A. J.**

New techniques are needed for reliable, fast, and accurate digital audio editing. This paper reports on the use of a random-access disc to assist both in cut-tape editing with state-of-the-art audio platforms, and also as a self-contained editing system. Topics discussed include software engineering, the man-machine interface, and the role of the auxiliary 'labels' data channel.

#### **2. Hack Audio - An Introduction to Computer Programming and Digital Signal Processing in MATLAB By Eric Tarr**

This paper offers an overview of features related, among others, to timbre, tonality, rhythm, or form that can be extracted with the MIR toolbox. The toolbox also includes functions for statistical analysis, segmentation, and clustering. Particular attention has been paid to designing a syntax that offers both simplicities of use and transparent adaptiveness to a multiplicity of possible input types.

#### **3. A MATLAB TOOLBOX FOR MUSICAL FEATURE EXTRACTION FROM AUDIO by Olivier Lartillot, Petri Toivainen**

This paper serves as a standalone introduction to audio analysis, providing a theoretical background to many state-of-the-art techniques. It covers the essential theory necessary to develop audio engineering applications and uses programming techniques, notably MATLAB®, to take a more applied approach to the topic. Basic theory and reproducible experiments are combined to demonstrate theoretical concepts from a practical point of view and provide a solid foundation in audio analysis.

#### **4. Introduction to Audio Analysis by By Theodoros Giannakopoulos, Aggelos Pikrakis**

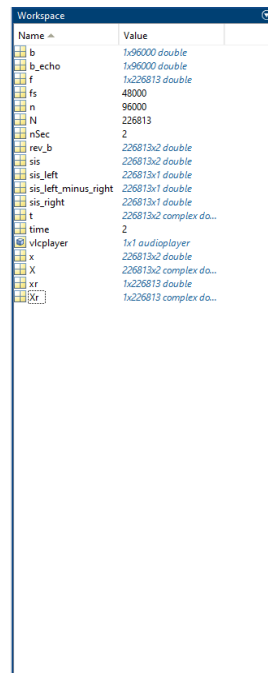
This paper provides a computer program to create audio effects using digital signal processing. It implements the following products: signal gain change, digital summing, tremolo, auto-pan, mid/side processing, stereo widening, distortion, echo, filtering, equalization, multi-band processing, vibrato, chorus, flanger, phaser, pitch shifter, auto-wah, convolution and algorithmic reverb, vocoder, transient designer, compressor, expander, and de-esser.

## 5. Audio Signal Processing and Coding

In-depth treatment of algorithms and standards for transparent coding of high-fidelity audio. Readers learn how algorithms for high-quality digital audio deliver fine signal reproduction with a minimum number of bits. The book's unique features include detailed coverage of topics such as filter banks, transform coding, sinusoidal analysis, linear prediction, hybrid algorithms, perceptual evaluation methods, scalable algorithms, Internet applications, MP3 and MP4 stereo systems, and current international and commercial audio standards.

## 4. Methodology

A compressor is an audio tool to compress some input source's dynamic range. A compressor works by reducing peaks and increasing the overall level. The dynamic range is reduced, yet the apparent loudness of a signal is increased. Essentially, the loud bits are softened, and the quiet bits are increased in level. An example may include; compressing a vocal, allowing it to sit on top of a mix, or squeezing a combination to obtain the maximum loudness without overloading a transmission channel. By varying a compressor's parameters, its behaviour can be changed to suit the application.



Name	Value
b	1x96000 double
b_echo	1x96000 double
f	1x226813 double
fs	48000
n	96000
N	226813
nSec	2
rev_b	226813x2 double
sis	226813x2 double
sis_left	226813x1 double
sis_left_minus_right	226813x1 double
sis_right	226813x1 double
t	226813x2 complex do...
time	2
vlcplayer	1x1 audioplayer
x	226813x2 double
X	226813x2 complex do...
xr	1x226813 double
Xr	1x226813 complex do...

Fig. 3. Program workspace on Matlab

The program also contains other variables that enable the following changes in its features:

- a. Changing the audio's speed

- b. Plotting the audio's waveform
- c. Reversing the audio
- d. Adding echo

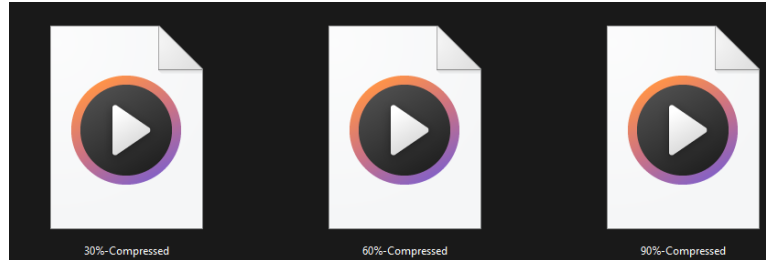


Fig. 4. Saved compressed audio files

- **Some important functions in the code:**

- 1. **fftshift**

- a. `fftshift(X)` rearranges a Fourier transform `X` by shifting the zero-frequency component to the center of the array.
- b. If `X` is a vector, then `fftshift` swaps the left and right halves of `X`.
- c. If `X` is a matrix, then `fftshift` swaps the first quadrant of `X` with the third, and the second quadrant with the fourth.
- d. If `X` is a multidimensional array, then `fftshift` swaps half-spaces of `X` along each dimension.

## 5. Code

```
% Code I
% 1
[sis,fs]= audioread('C:\Users\Dell\Desktop\SP\Audio.wav');
% 2
sound(sis,fs);    % % Normal (1X)
sound(sis,fs*2);  % % Fast (2X)
sound(sis,fs/2);  % % Slow (0.5X)
% 3-Original plot
time=(1/fs)*length(sis);
t=linspace(0,time,length(sis));
plot(t,sis,'k');
% 4-Reverse
rev_b=flipud(sis);
sound(rev_b,fs);
soundsc(rev_b,fs);
b=sis(1:fs*10);
rev_b=flipud(b);
time=(1/fs)*length(b);
t=linspace(0, time, length(b));
% 5-Echo
nSec=2;
```

```

b=sis(1: fs*nSec);
b_echo=b;
N=fs/2;
for n=N+1:length(b)
% Adding N-off phase sound to the original input .
b_echo(n) = b(n)+b(n-N);
end
time=(1/fs)*length(b);
t=linspace(0,time,length(b));
plot(t,b,'k',t,b_echo,'r');
xlabel('Time (sec)');
ylabel('Signal Strength');
title('Signal Processing');
% Code II
% 7-Compression
% a
[x,fs]=audioread('C:\Users\Dell\Desktop\SP\Audio.wav');
N=length(x);
vlcplayer=audioplayer(x,fs);
vlcplayer.play
% b
t=fft(x,N);
X=fftshift(t);
f=-fs/2:fs/N:(fs/2-fs/N);
figure(1)
plot(f,abs(X))
title('Original-Signal')
% c-30% compression
Xr = zeros(1,N);
Xr((N*((30/100)/2))+1:N*(1-(30/100)/2))=X((N*((30/100)/2))+1:N*(1-(30/100)/2));
figure(2)
plot(f,abs((Xr)));
xr=real(ifft(fftshift(Xr)));
audiowrite('30%-Compressed.wav',xr,fs);
title('30%Compressed')
xlabel('Freq (Hz)'); ylabel('Magnitude');
% d-60% compression
Xr = zeros(1,N);
Xr((N*((60/100)/2))+1:N*(1-(60/100)/2))=X((N*((60/100)/2))+1:N*(1-(60/100)/2));
figure(2)
plot(f,abs((Xr)));
xr=real(ifft(fftshift(Xr)));
audiowrite('60%-Compressed.wav',xr,fs);
title('60%Compressed')
xlabel('Freq (Hz)'); ylabel('Magnitude');
% e-90% compression
Xr = zeros(1,N);
Xr((N*((90/100)/2))+1:N*(1-(90/100)/2))=X((N*((90/100)/2))+1:N*(1-(90/100)/2));
figure(2)
plot(f,abs((Xr)));
xr=real(ifft(fftshift(Xr)));
audiowrite('90%-Compressed.wav',xr,fs);
title('90%Compressed')
xlabel('Freq (Hz)'); ylabel('Magnitude');

```

Variables such as **sis**, **vlcplayer**, and **reverb** were used to perform the functions necessitated by the editing commands. The compressed audio files were saved using the **vlcplayer** process in the specified directory.

## 6. Output

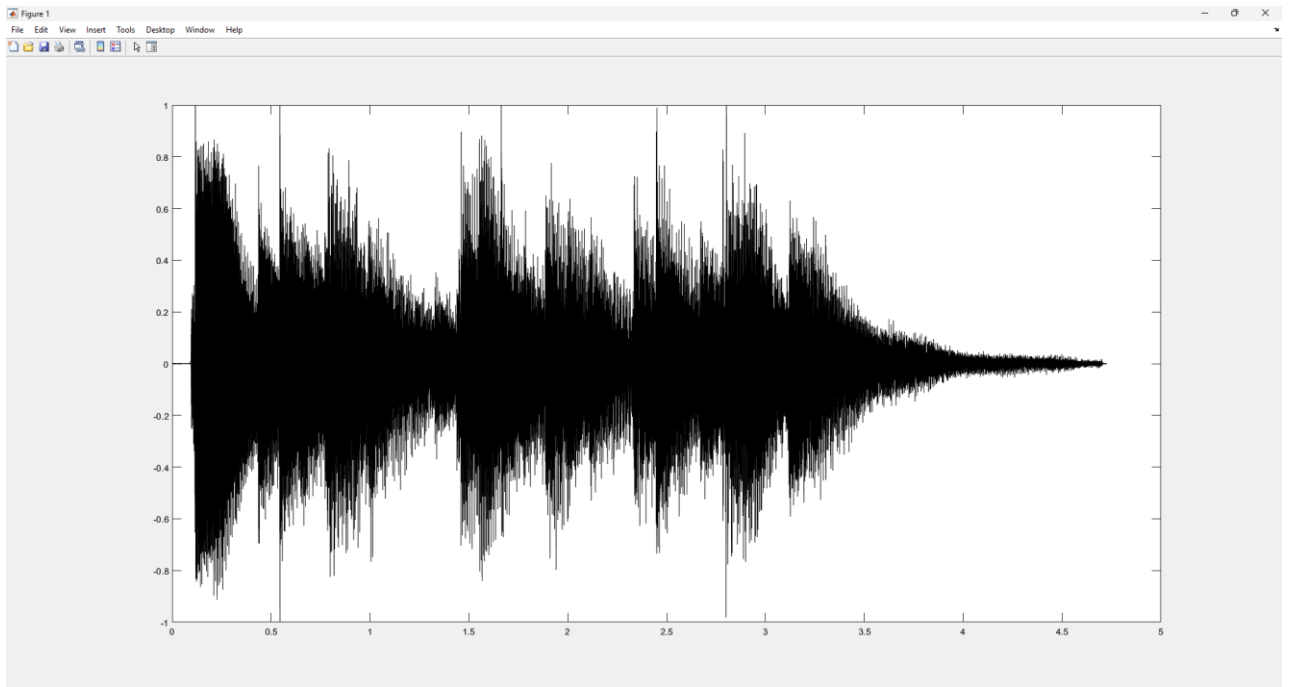


Fig. 5 Audio Waveform

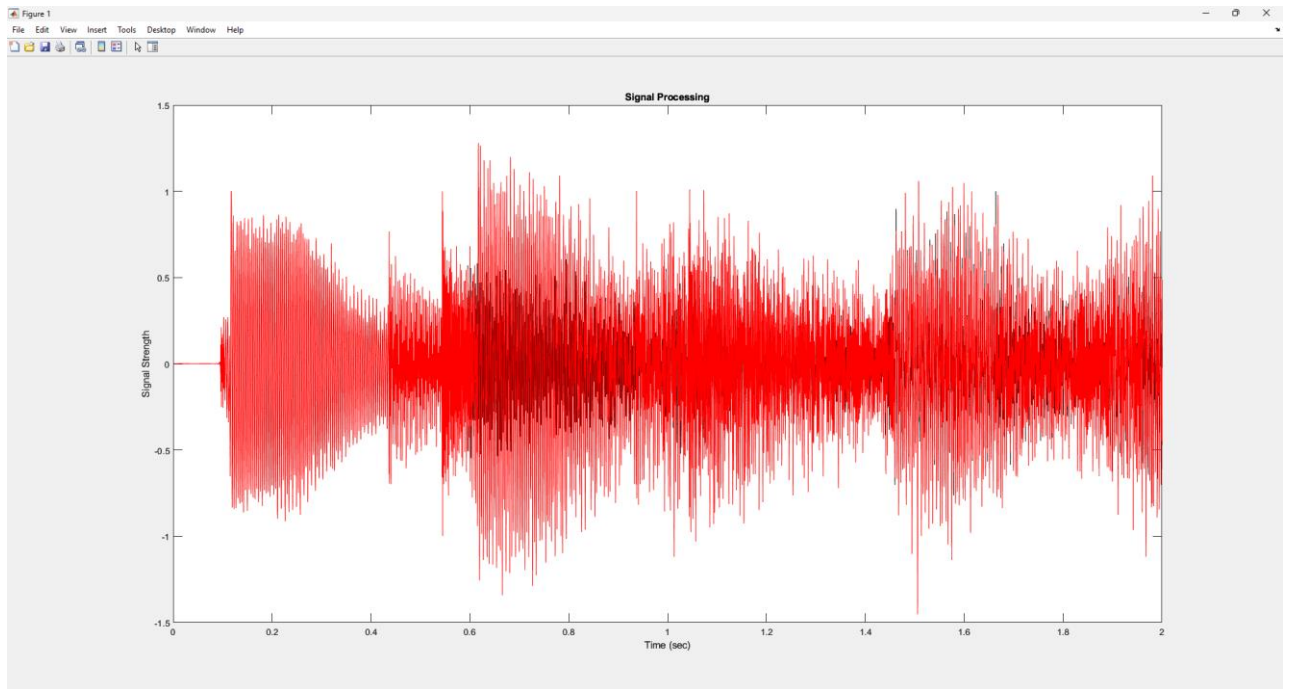


Fig. 6 Signal Strength

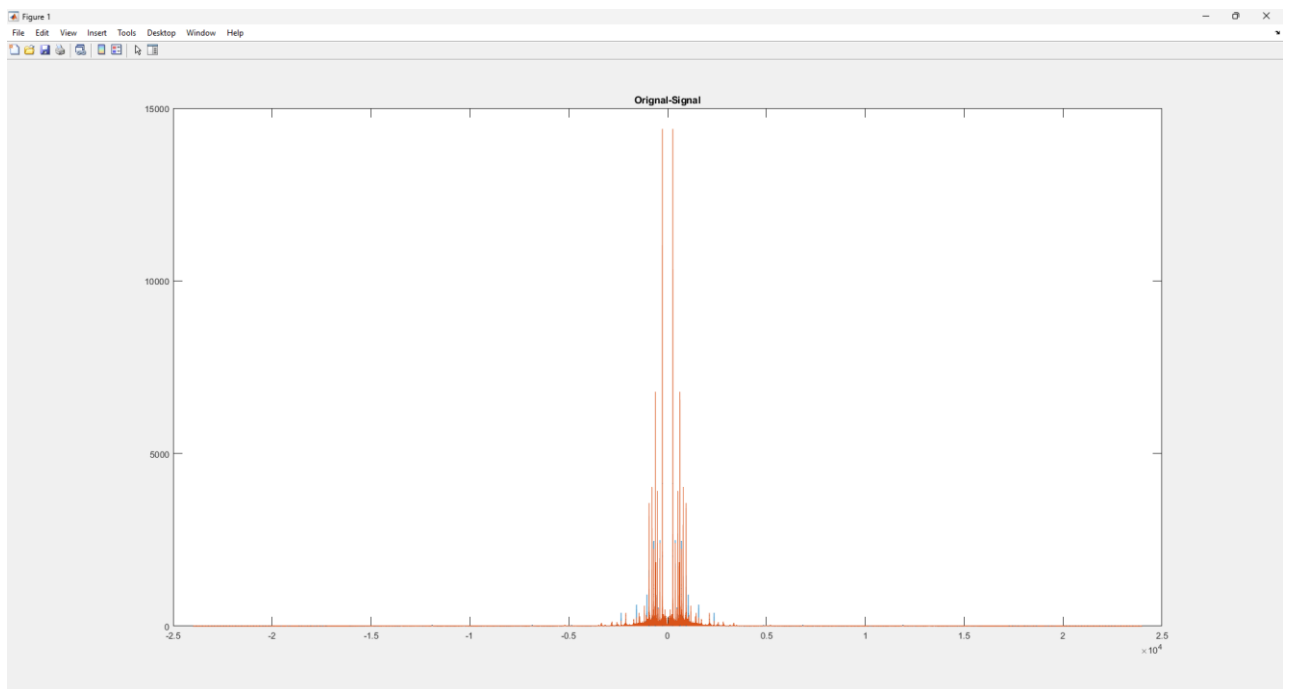


Fig. 7 Magnitude of the original signal

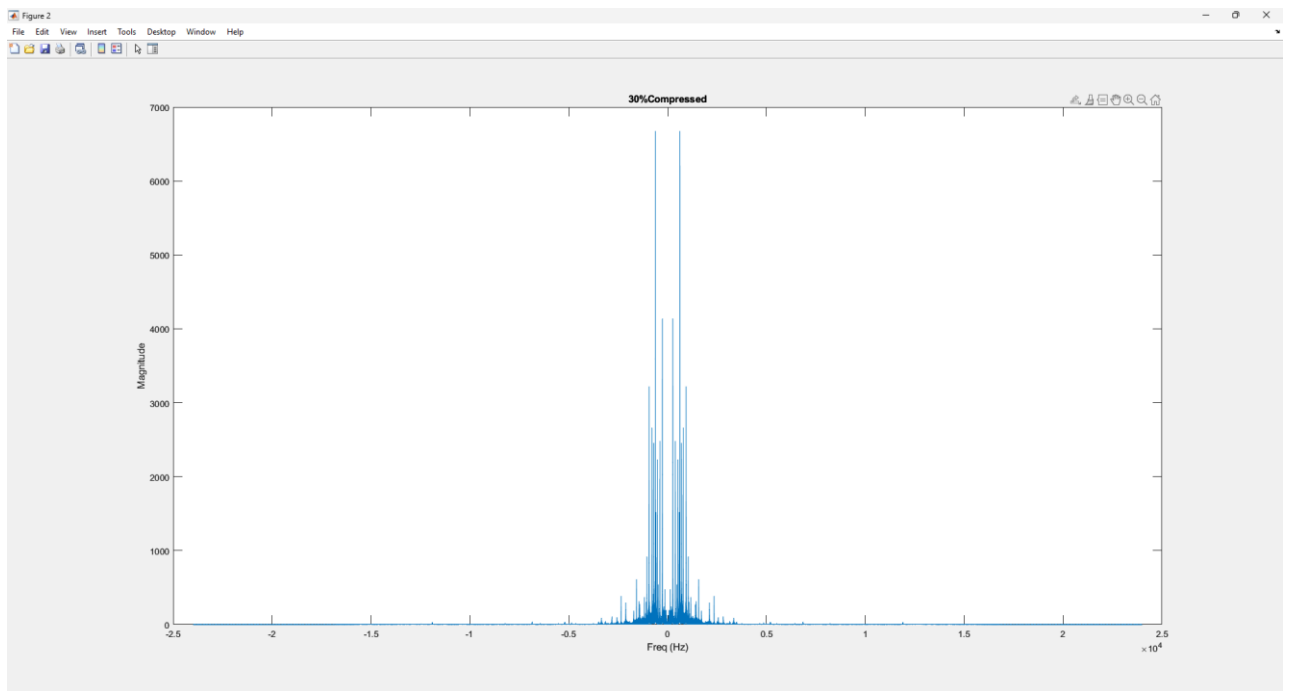


Fig. 8 Magnitude of the 30% compressed signal

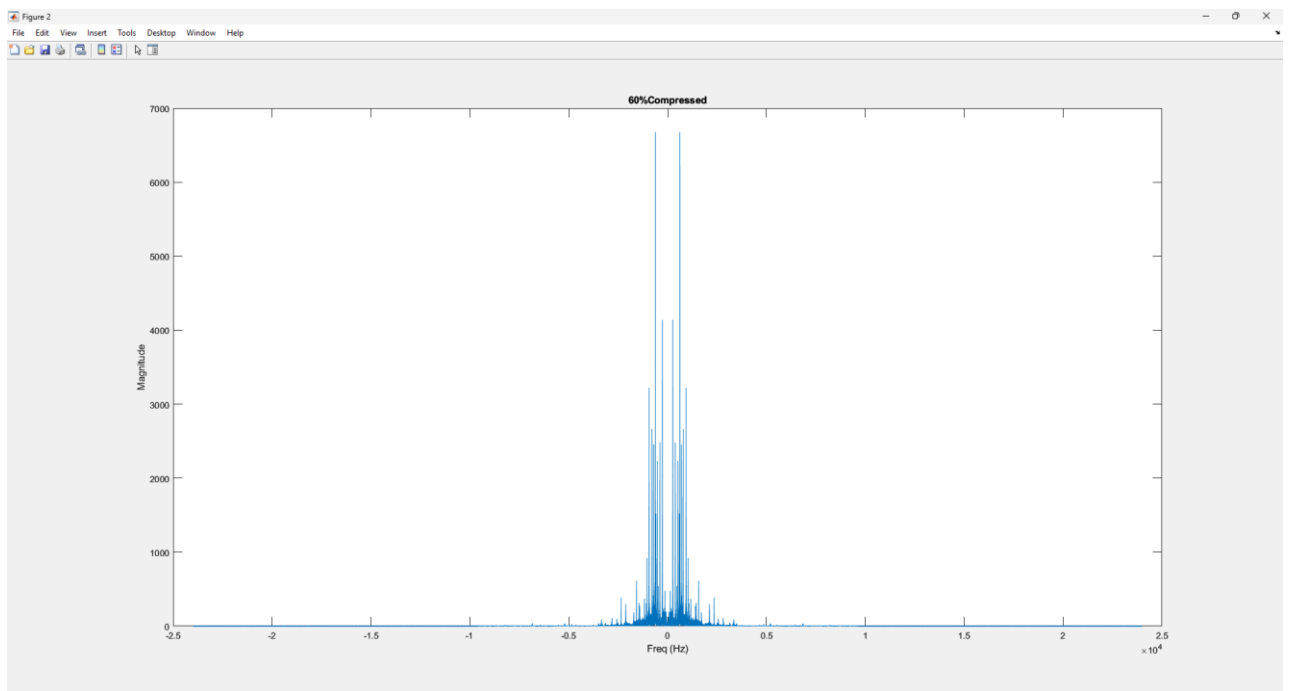


Fig. 9 Magnitude of the 60% compressed signal



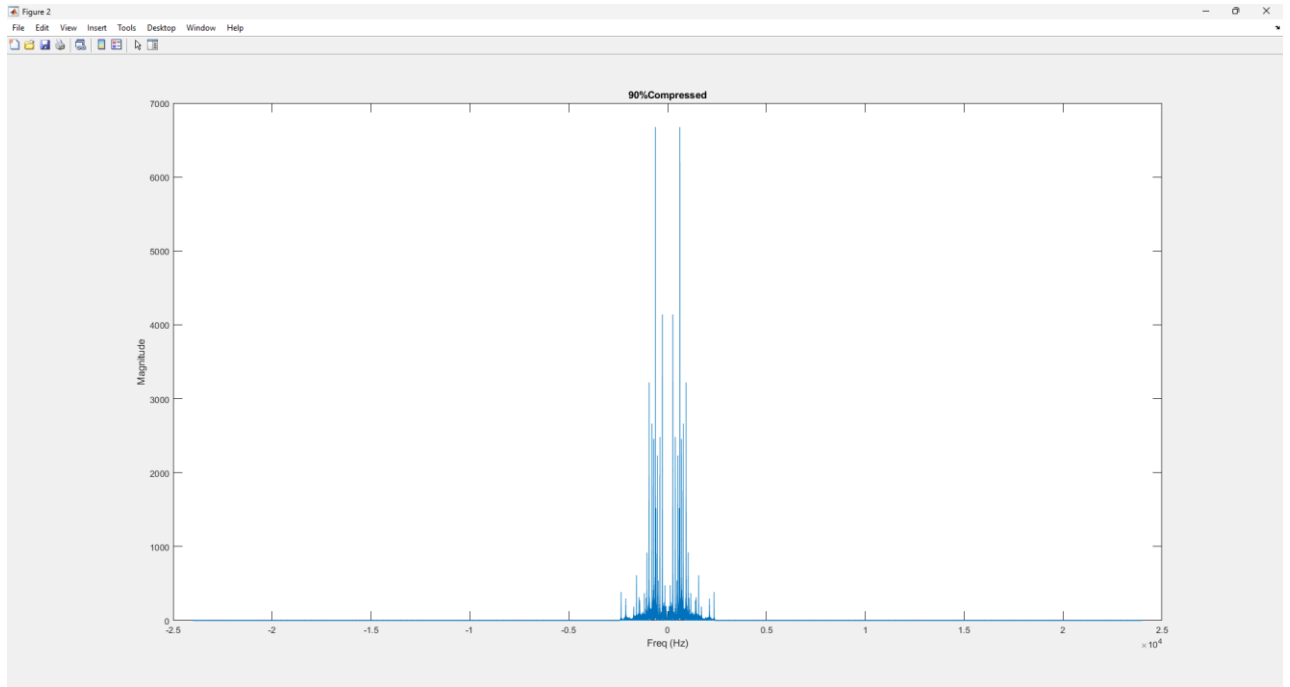


Fig. 10 Magnitude of the 90% compressed signal

## 7. Conclusion

A MATLAB function was used to simulate the behaviour of an audio compressor and editor. While the compressor lacked the control commonly found on traditional software or hardware compressors, it did demonstrate its main file-reduction capabilities. With experimentation, suitable input parameter values were found that best demonstrate the expected functionality. Other audio editing features were tested, like changing audio speed, plotting audio waveform, reversing the audio, adding echo to audio, and compressing audio files. Future work could be undertaken to improve the functionality of the compressor to enable its file reduction capability better while maintaining the audio quality and adding more editing features such as reverb, bass, treble changes, etc.

## 8. References

1. GU. W.. McNally, PH. S.. Gaskell, and A.. J.. Stirling, "Digital Audio Editing," Paper 2214, (1985 March.). doi:
2. Tarr, E. (2018). Hack Audio: An Introduction to Computer Programming and Digital Signal Processing in MATLAB (1st ed.). Routledge. <https://doi.org/10.4324/9781351018463>
3. Lartillot, Olivier, and Petri Toiviainen. "A Matlab toolbox for musical feature extraction from audio." In International conference on digital audio effects, vol. 237, p. 244. 2007.

4. Giannakopoulos, Theodoros., Pikrakis, Aggelos. Introduction to Audio Analysis: A MATLAB® Approach. Netherlands: Elsevier Science, 2014.
5. Atti, Venkatraman., Painter, Ted., Spanias, Andreas. Audio Signal Processing and Coding. Germany: Wiley, 2006.