SIGNALS AND SYSTEMS

(ECE 1004)

AUDIO COMPRESSION TECHNIQUES USING MATLAB

SUBMITTED BY:
R. VENGATESH(19BEC0398), T.GEETH KULDEEP (19BEC0421), ADLA VISHWAJEET REDDY (19BED0521), SREE LALITHA P.V.S (19BEC0530), ANKITA MANDAL (19BEC0543)

SUBMITTED TO: PROF. YEPUGANTI KARUNA

CONTENTS

ABSTRACT

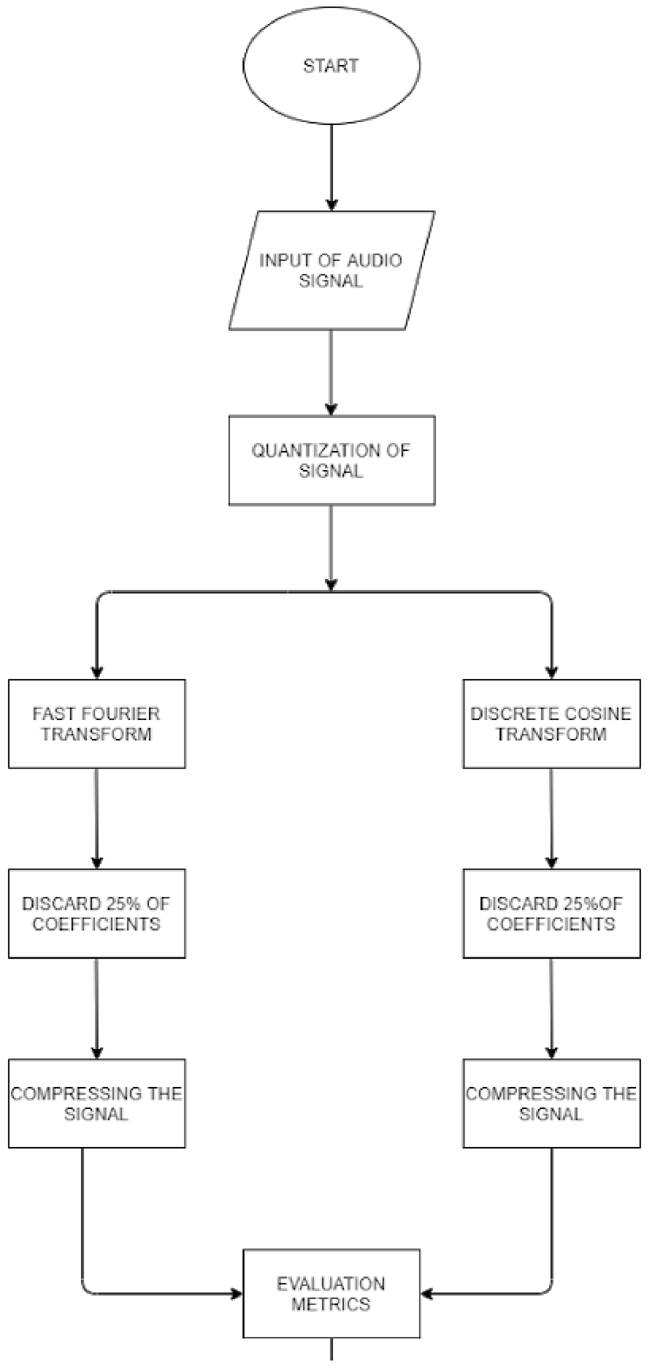
Compression of audio signals is of important significance in today's world due to limited bandwidth and transmission or storage capacity. Signal compression is a process of compressing a speech signal into efficient encoded representations that can be decoded to produce a signal that is as close to the original one.

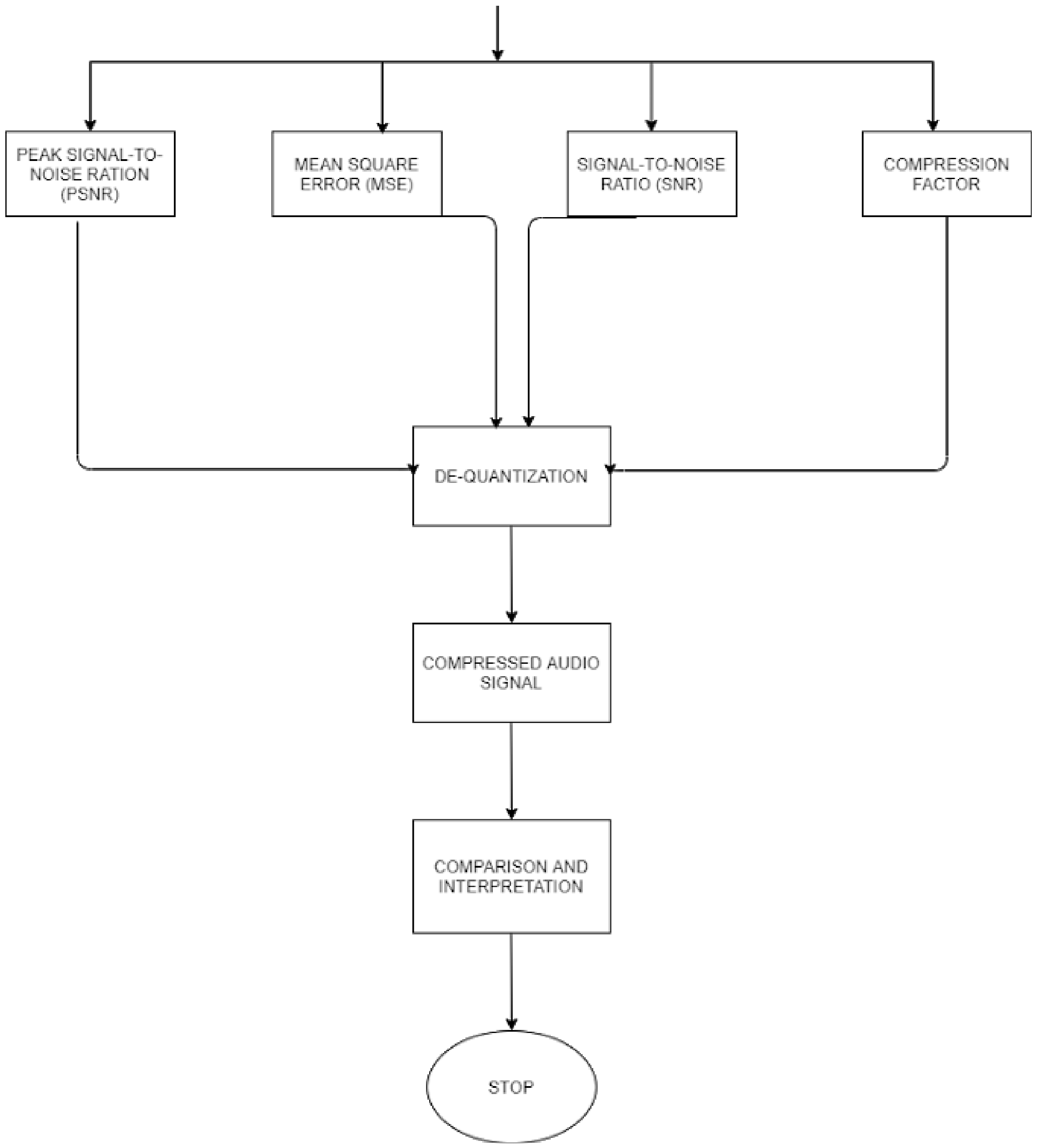
MATLAB is one of the most used software to analyse and process a signal, This project is based on the Matlab code to compress and analyse an audio signal using fourier transform and discrete cosine transform.

INTRODUCTION

Audio is the most important medium to be transmitted in a conference-like application. In order to be able to successfully transmit audio through a low bandwidth network, however, one needs to compress it, so that its required bandwidth is manageable. Compression basically reduces the dynamic range of the audio signal. Different methods for compression have been designed to meet this. A lot of research has been done in DCT and FFT which are used extensively. Fast Fourier Transform(FFT) is an algorithm of FFT. It reduces the number of needed computations for N points from $2N^2$ to $2N\log_2 N$.
One dimensional discrete cosine transform(DCT-1) and two dimensional discrete cosine transform(DCT-2) are used for data compression. Due to the high correlation between adjacent coefficients in the DCT method, signals are reconstructed accurately even with only few coefficients.

FLOW OF THE PROCESS




PEAK SIGNAL-TO-NOISE RATION (PSNR)
MEAN SQUARE ERROR (MSE)
SIGNAL-TO-NOISE RATIO (SNR)
COMPRESSION FACTOR
DE-QUANTIZATION
COMPRESSED AUDIO SIGNAL
COMPARISON AND INTERPRETATION
STOP

PARAMETERS OF COMPRESSION

INPUT GAIN: It controls the level of the signal going into the audio compressor.

COMPRESSION RATIO: controls how much the signal will be compressed once it's over the threshold level. The higher the ratio, the more compression there is.

- If the ratio is 1:1, there is no compression at all.
- If the ratio is set at 2:1, for every 2 dB of sound that goes over the threshold, you get 1 dB of output above the threshold. So if the signal goes over the threshold by 10 dB, the compressor reduces this signal so it's now 5 dB over the threshold.
- If the ratio goes up to 8:1, for every 8 dB of sound over the threshold you would get 1dB of output above the threshold. So if the signal goes over the threshold by 16 dB, the compressor reduces this so only 2 dB goes over the threshold.

KNEE: **Soft-knee** compression is gentler on the sound as it goes through the audio compressor – the change from uncompressed to compressed sound is smoother. **Hard-knee** compression is a more immediate and obvious effect.

THRESHOLD: The threshold sets the level at which the compressor kicks-in and starts changing the dynamics of the recording. So for example, if you set your threshold at -20 dB, everything below this level will not be affected by the compressor. But everything louder than this level (-20 dB) will be compressed.

After calculating the transform coefficients of the speech signal, compression involves truncating the transform coefficients below a threshold

- GLOBAL THRESHOLD : A global thresholding technique is one which makes use of a single threshold value for the whole signal.
- LEVEL DEPENDENT THRESHOLD : This method of thresholding selects the highest absolute valued coefficients at each level.The truncation of insignificant coefficients can be optimized when such a level dependent thresholding is used.

<u>**METHODS OF COMPRESSION -**</u>

FOURIER TRANSFORM METHOD

All data compression algorithms consist of at least a model and a coder (with optional pre-processing transforms). The coder assigns shorter codes to the more likely symbols. There are efficient and optimal solutions to the coding problem. However, optimal modelling has been proven not computable. Modelling (or equivalently, prediction) is both an artificial intelligence (AI) problem and art. Lossy compression consists of a transform to separate important from unimportant data, followed by lossless compression of the important part and discarding the rest. The audio compression software crops off the inaudible frequencies, reducing the bit rate of the less sensitive sound signals etc. Thus in this project, our aim is to use Fourier Transform to differentiate a frequency from another.

The FFT is the most important discrete transform, used to perform Fourier analysis in many practical applications. In digital signal processing, the function is any quantity or signal that varies over time, such as the pressure of a sound wave, a radio signal, or daily temperature readings, sampled over a finite time interval (often defined by a window function). In image processing, the samples can be the values of pixels along a row or column of a raster image. The FFT is also used to efficiently solve partial differential equations, and to perform other operations such as convolutions or multiplying large integers.

MATHEMATICAL FORMULA USED

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{-kN} \quad 0 \leq \mathrm{n} \leq \mathrm{N}\text{-}1$$

where $W_N = e^{-J2\pi}/_N$

similarly the DFT becomes

$$x(n) = \frac{1}{N} \sum_{K=0}^{N-1} X(k) W_N^{-kn} \quad 0 \leq \mathrm{n} \leq \mathrm{N}-1$$

We observe that for N numbers of points, direct computation requires N^2 complex multiplications whereas FFT algorithms require only N/2 $\log_2 N$ complex multiplications.

DISCRETE COSINE TRANSFORM TECHNIQUE

A discrete cosine transform (DCT) expresses a finite sequence of data points in terms of a sum of cosine functions oscillating at different frequencies. It is used in most digital media, including digital images , digital video , digital audio , digital television , digital radio , and speech coding. DCTs are also important to numerous other applications in science and engineering, such as digital signal processing, telecommunication devices, reducing network bandwidth usage, and spectral methods for the numerical solution of partial differential equations.

DCTs are equivalent to FFTs of roughly twice the length, operating on real data with even symmetry (since the Fourier transform of a real and even function is real and even), whereas in some variants the input and/or output data are shifted by half a sample.

MATHEMATICAL FORMULA USED

$$X(k) = \sqrt{\frac{2}{N}}\, C_k \sum_{n=0}^{N-1} x(n) \cos\left(\frac{\pi k(2n+1)}{2N}\right) \quad 0 \leq k \leq \text{N-1}$$

IDCT is defined as

$$x(n) = \sqrt{\frac{2}{N}} \sum_{k=0}^{N-1} C_k\, X(k) \cos\left(\frac{\pi k(2n+1)}{2N}\right) \quad 0 \leq \text{n} \leq \text{N-1}$$

where weighting function $C_m = \sqrt{\frac{1}{2}} \quad ; \; k = 0$

$1 \quad ; \; 1 \leq \text{k} \leq \text{N-1}$

PERFORMANCE EVALUATION PARAMETERS

Peak Signal to Noise Ratio (PSNR)

Peak signal-to-noise ratio, often abbreviated PSNR, is a term for the ratio between the maximum possible power of a signal and the power of corrupting noise that affects the fidelity of its representation

What is a good PSNR value?

Typical values for the PSNR in lossy image and video compression are between 30 and 50 dB, provided the bit depth is 8 bits, where higher is better. For 16-bit data typical values for the PSNR are between 60 and 80 dB. Acceptable values for wireless transmission quality loss are considered to be about 20 dB to 25 dB.

Mean Square Error

The mean squared error or mean squared deviation of an estimator measures the average of the squares of the errors

Signal to Noise Ratio (SNR)

signal-to-noise ratio, often written S/N or SNR, is a measure of signal strength relative to background noise. The ratio is usually measured in decibels (dB) using a signal-to-noise ratio formula. If the incoming signal strength in microvolts is Vs, and the noise level, also in microvolts, is Vn, then the signal-to-noise ratio, S/N, in decibels is given by the formula:
$S/N = 20 \log10(Vs/Vn)$

What is a good SNR value?

Generally, a signal with an SNR value of 20 dB or more is recommended for data networks whereas an SNR value of 25 dB or more is recommended for networks that use voice applications.

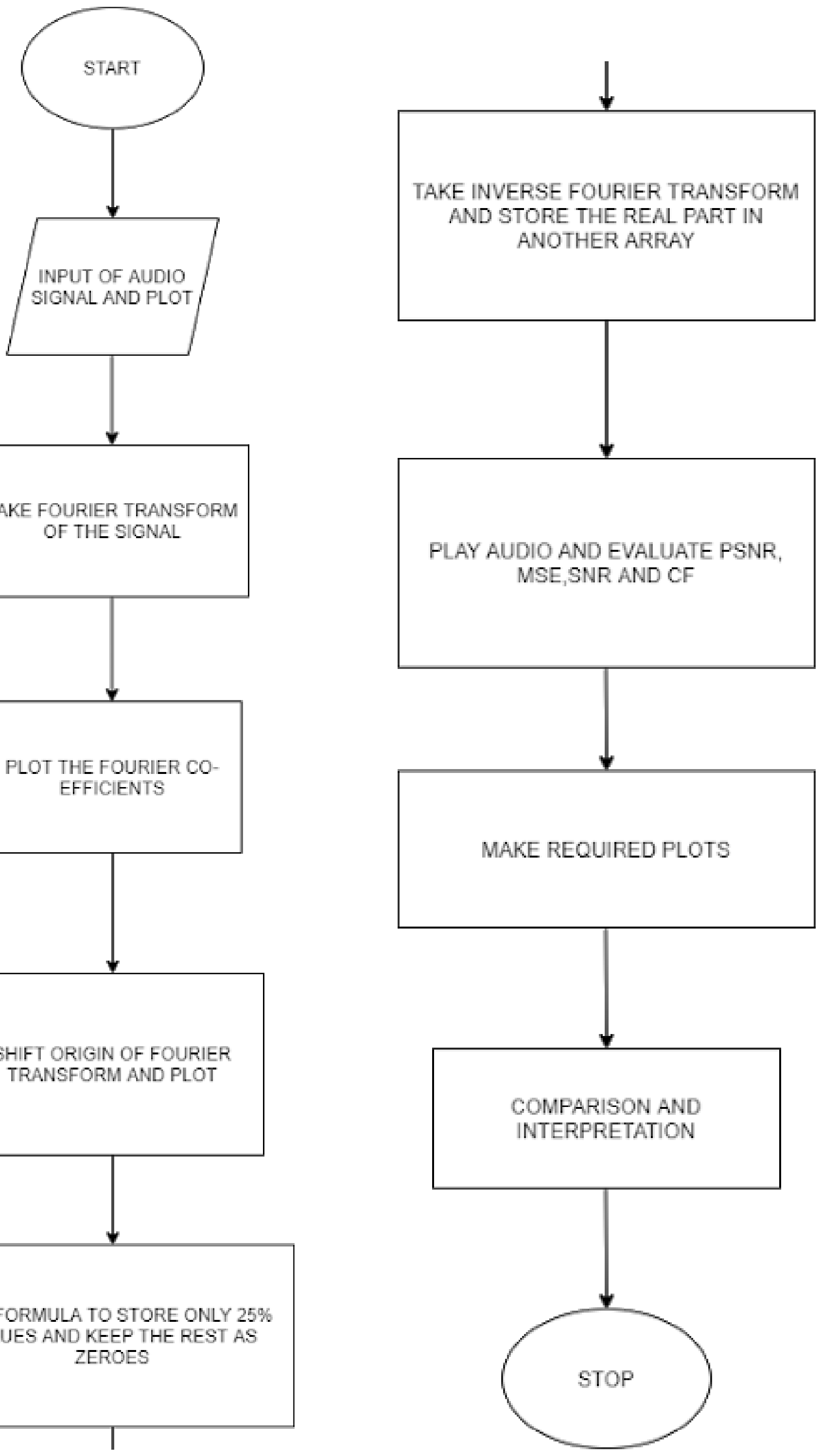
Compression Factor

It is the ratio of the original signal to the compressed signal. It is also known as compression power used to quantify the reduction in data-representation size produced by a data compression algorithm.

$$\mathrm{CF} = \frac{\text{length of original signal}}{\text{length of compressed signal}}$$

FAST FOURIER TRANSFORM

FLOWCHART OF OUR APPROACH


START
INPUT OF AUDIO SIGNAL AND PLOT
TAKE FOURIER TRANSFORM OF THE SIGNAL
PLOT THE FOURIER CO-EFFICIENTS
SHIFT ORIGIN OF FOURIER TRANSFORM AND PLOT
USE FORMULA TO STORE ONLY 25% VALUES AND KEEP THE REST AS ZEROES
TAKE INVERSE FOURIER TRANSFORM AND STORE THE REAL PART IN ANOTHER ARRAY
PLAY AUDIO AND EVALUATE PSNR, MSE,SNR AND CF
MAKE REQUIRED PLOTS
COMPARISON AND INTERPRETATION
STOP

ALGORITHM

STEP 1: Start

STEP 2: Take the input of the audio signal and visualize it.

STEP 3: Perform Fourier Transform on the input audio signal.

STEP 4: Plot the Fourier Coefficients.

STEP 5: Shift the origin of Fourier Transform and plot the shifted coefficients.

STEP 6: Declare an array full of zeroes.

STEP 7: Use the formula to store only 25% of the values and keep the rest of the values as Zeros.

STEP 8: Take the Inverse Fourier Transform and store the real part in another array.

STEP 9: Make necessary plots.

STEP 10: Evaluate the performance parameters.

STEP 11: Stop.

MATLAB CODE

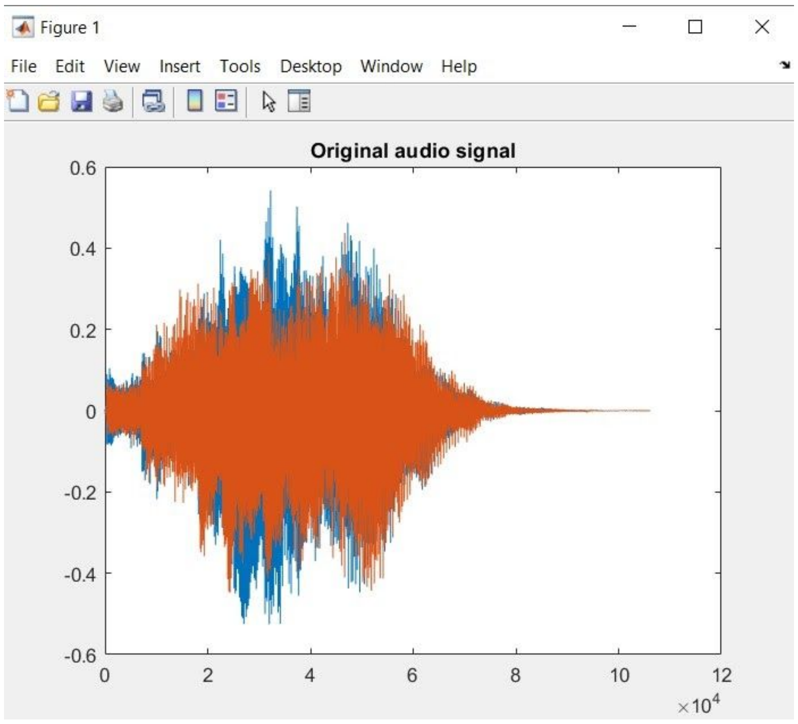
```
clear all

close all

Clc

[x,fs]=audioread('inputaudio.wav');

N=length(x);

x1=x(:,1);

x2=x(:,2);y
```

```
figure, plot(x); title('Original audio signal');

vlc player=audio player(x1,fs);

vlcplayer.play

%vlcplayer=audioplayer(x2,fs);

%vlcplayer.play

t=fft(x,N);

t1=fft(x1,N);

t2=fft(x2,N);

figure, plot(t); title('FFT of input audio signal');

X=fftshift(t);

X1=fftshift(t1);

X2=fftshift(t2);

figure, plot(X);title('Origin Shifted input audio signal');

figure;

plot(abs(X1)); hold on;

plot(abs(X2));title('Absolute FFT coefficients');

%output for 25% audio compression and its wave

Xr=zeros(1,N);

Xr1=zeros(1,N);

Xr2=zeros(1,N);

Xr1((N*((60/100)/2))+1:N*(1-(60/100)/2))=X1((N*((60/100)/2))+1:N*(1-(60/100)/2));
```

```
Xr2((N*((60/100)/2))+1:N*(1-(60/100)/2))=X2((N*((60/100)/2))+1:N*(1-(60/100)/2));

figure; plot(abs(Xr1));hold on;

plot(abs(Xr2));title('Compressed Audio Signal');

xr1=real(ifft(fftshift(Xr1)));%reconstruction

xr2=real(ifft(fftshift(Xr2)));%reconstruction

y(:,1)=xr1';

y(:,2)=xr2';

xr(:,1)=xr1;xr(:,2)=xr2;

figure, plot(xr); title('After Inverse FFT, Real part of the compressed signal');

audiowrite('25%compressed.wav',y,fs);

title('25% compressed audio')

xlabel('Freq(Hz)');ylabel('Magnitude');

%vlcplayer=audioplayer(y,fs);

%vlcplayer.play

%Evaluation

PSNR=psnr(y,x)

MSError=mse(y,x)

SNR=snr(y,x)

%%changing ratio to 60,70,80,90,95%,just change line 17 t0:

%Xr((N/4)+1:(3*N/4))= X((N/4)+1:(3*N/4));

%Xr((N*((70/100)/2))+1:N*(1-(70/100)/2))=X((N*((70/100)/2))+1:N*(1-(70/100)/2));
```

```
%Xr((N*((80/100)/2))+1:N*(1-(80/100)/2))=X((N*((80/100)/2))+1:N*(1-(80/100)/2));

%Xr((N*((90/100)/2))+1:N*(1-(90/100)/2))=X((N*((90/100)/2))+1:N*(1-(90/100)/2));

%Xr((N*((95/100)/2))+1:N*(1-(95/100)/2))=X((N*((95/100)/2))+1:N*(1-(95/100)/2));

%%changing each one to a diff wave to be heard on desktop:

%audiowrite('50%compressed.wav',xr,fs);

%audiowrite('60%compressed.wav',xr,fs);

%audiowrite('70%compressed.wav',xr,fs);

%audiowrite('80%compressed.wav',xr,fs);

%audiowrite('90%compressed.wav',xr,fs);

%audiowrite('95%compressed.wav',xr,fs);
```

GRAPH



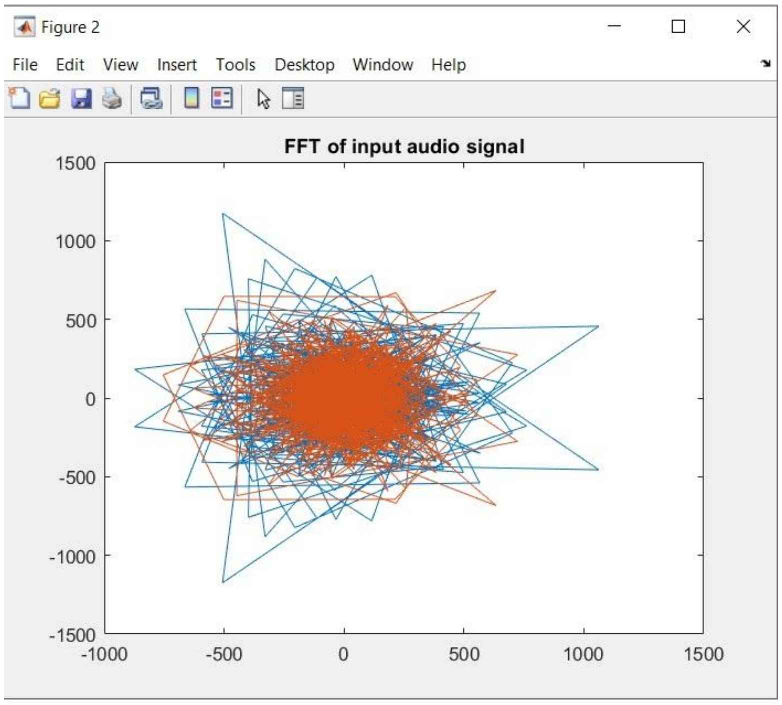


Figure 2
File Edit View Insert Tools Desktop Window Help
FFT of input audio signal
1500
1000
500
0
-500
-1000
-1500
-1000
-500
0
500
1000
1500

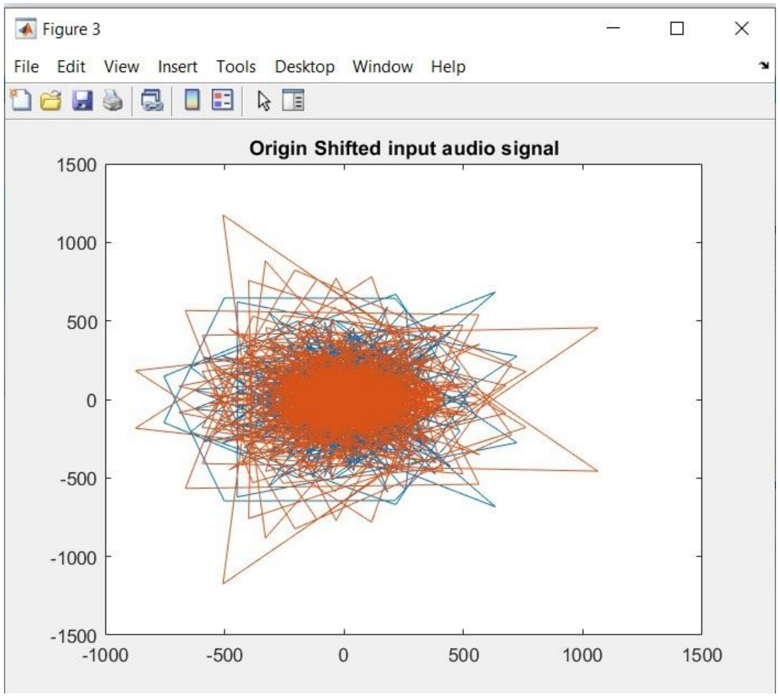


Figure 3
File Edit View Insert Tools Desktop Window Help
Origin Shifted input audio signal
1500
1000
500
0
-500
-1000
-1500
-1000
-500
0
500
1000
1500

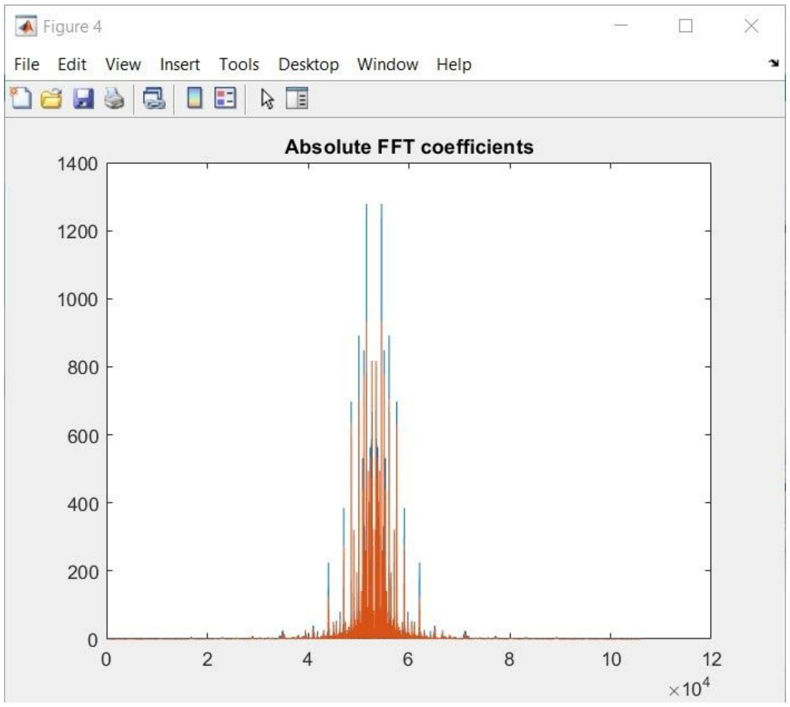


Figure 4
File Edit View Insert Tools Desktop Window Help
Absolute FFT coefficients
1400
1200
1000
800
600
400
200
0
0
2
4
6
8
10
12
×10^4

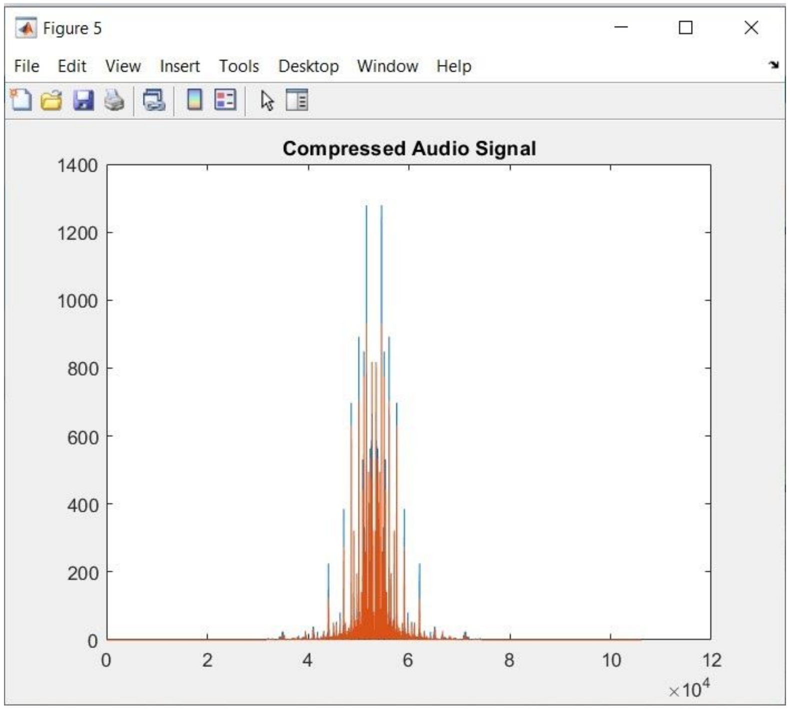


Figure 5
File Edit View Insert Tools Desktop Window Help
Compressed Audio Signal
1400
1200
1000
800
600
400
200
0
0
2
4
6
8
10
12
×10^4

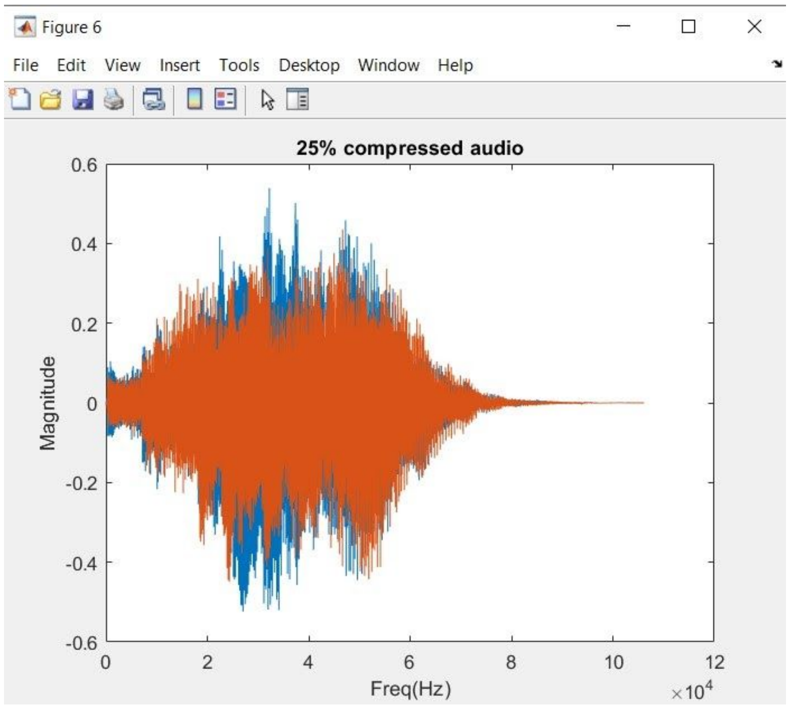
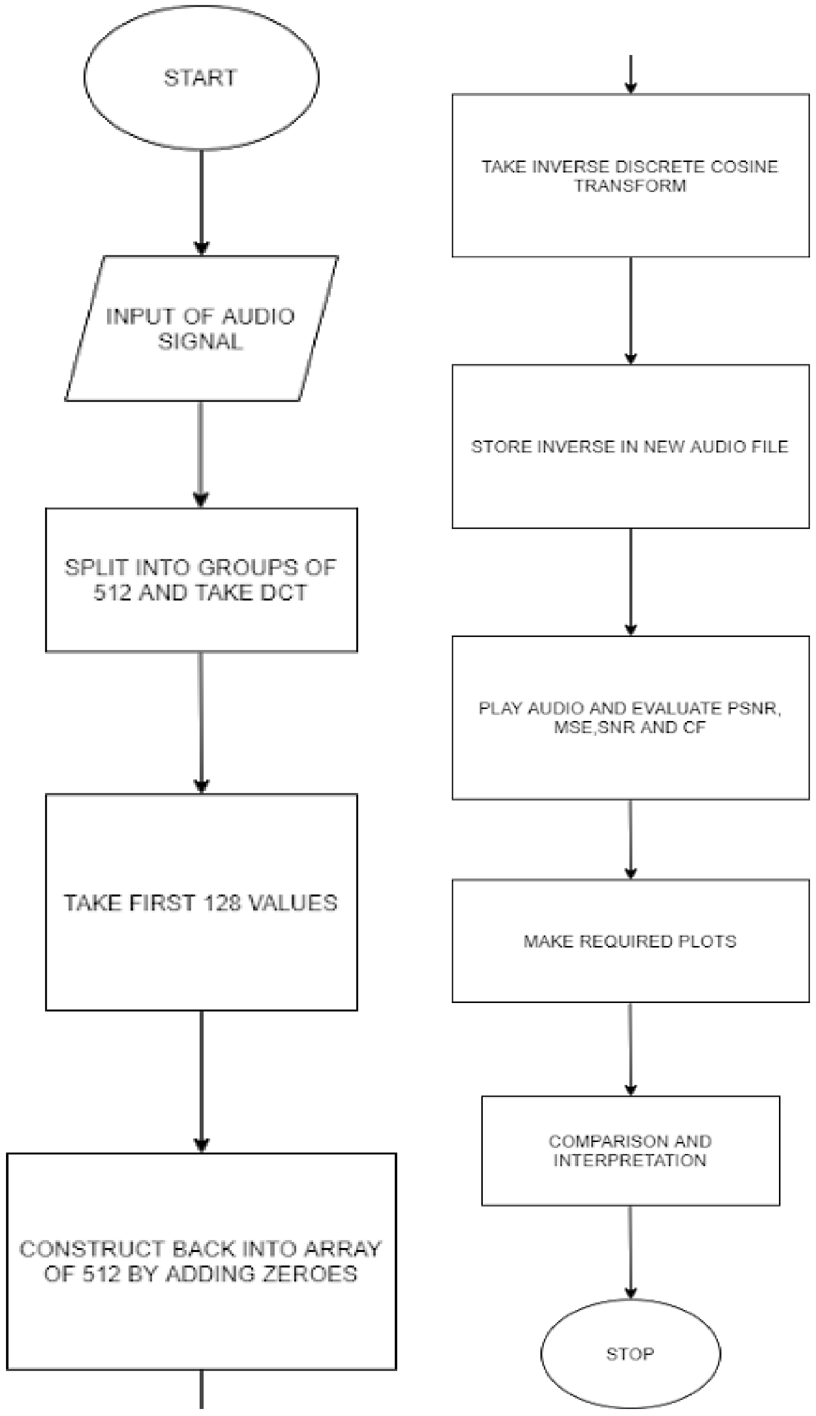


Figure 6
File Edit View Insert Tools Desktop Window Help
25% compressed audio
Magnitude
Freq(Hz)
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
0
2
4
6
8
10
12
$\times 10^4$

DISCRETE COSINE TRANSFORM

FLOWCHART OF OUR APPROACH


START
INPUT OF AUDIO SIGNAL
SPLIT INTO GROUPS OF 512 AND TAKE DCT
TAKE FIRST 128 VALUES
CONSTRUCT BACK INTO ARRAY OF 512 BY ADDING ZEROES
TAKE INVERSE DISCRETE COSINE TRANSFORM
STORE INVERSE IN NEW AUDIO FILE
PLAY AUDIO AND EVALUATE PSNR, MSE,SNR AND CF
MAKE REQUIRED PLOTS
COMPARISON AND INTERPRETATION
STOP

ALGORITHM

STEP 1: Start

STEP 2: Take the input of the audio signal and visualize it.

STEP 3: Store the two signals in array A1 and A2.

STEP 4: Split the signals into groups of 512 and apply Discrete Cosine Transform.

STEP 5: Take the first 128 values from each group of 512 and concatenate it to another array in order to compress to 25%.

STEP 6: Construct an array of 512 again by adding 384 zeroes after every 128 values.

STEP 7: Take the Inverse Cosine Transform and store it in a new array. Write it as a new audio file.

STEP 8: Play audio and evaluate the performance parameters.

STEP 9: Make necessary plots of the original signal, DCT coefficients, reconstructed signal and the difference signal.

STEP 10: Stop.

MATLAB CODE

```
clc

clear

close all

[A,fs]=audioread('inputaudio.wav');

sound(A,fs);

%% Sender

C1=[ ];

C2=[ ];
```

```
A=A';

A1=A(1,:);

A2=A(2,:);

for i=512:512:size(A,2)

   B1=dct(A1(i-511:i));

   C1=[C1, B1(1:128)];

   B2=dct(A2(i-511:i));

   C2=[C2, B2(1:128)];

end

%% Receiver

D1=[ ];

D2=[ ];

for i=128:128:numel(C1)

   S1=[C1(i-127:i),zeros(1,384)];

   S1=idct(S1);

   D1=[D1,S1];

   S2=[C2(i-127:i),zeros(1,384)];

   S2=idct(S2);

   D2=[D2,S2];

end

D(:,1)=D1;

D(:,2)=D2;

audiowrite('receivedaudio.wav',D,fs);

[D,fs]=audioread('receivedaudio.wav');

sound(D,fs);pause(0.2);
```

```
%% Evaluation
dis=numel(A')-numel(D);
z1=[D(:,1)',zeros(1,dis/2)];
z2=[D(:,2)',zeros(1,dis/2)];
E(1,:)=z1;
E(2,:)=z2;
PSNR=psnr(E,A)
MSError=mse(E,A)
SNR=snr(E,A)
%% plots
figure;
plot(A(1,:),'b');
hold on
plot(A(2,:),'r');
title('Original Audio Signal');
xlabel('Samples');
ylabel('Amp.');
figure;
plot(C1,'b');hold on
plot(C2,'r');
title('DCT Coefficients');
figure;
plot(E(1,:),'b');
hold on
plot(E(2,:),'r')
```

```
title('Reconstructed Audio Signal');

xlabel('Samples');

ylabel('Amp.');

figure;

plot((E(1,:)-A(1,:)),'b')

hold on

plot((E(2,:)-A(2,:)),'r')

title('Difference Audio Signal');

xlabel('Samples');

ylabel('Amp.');

% legend('Original' , 'Received','Difference')

% title('Audio Compression By DCT');

% xlabel('Samples');

% ylabel('Amp.');

% grid();
```

GRAPH

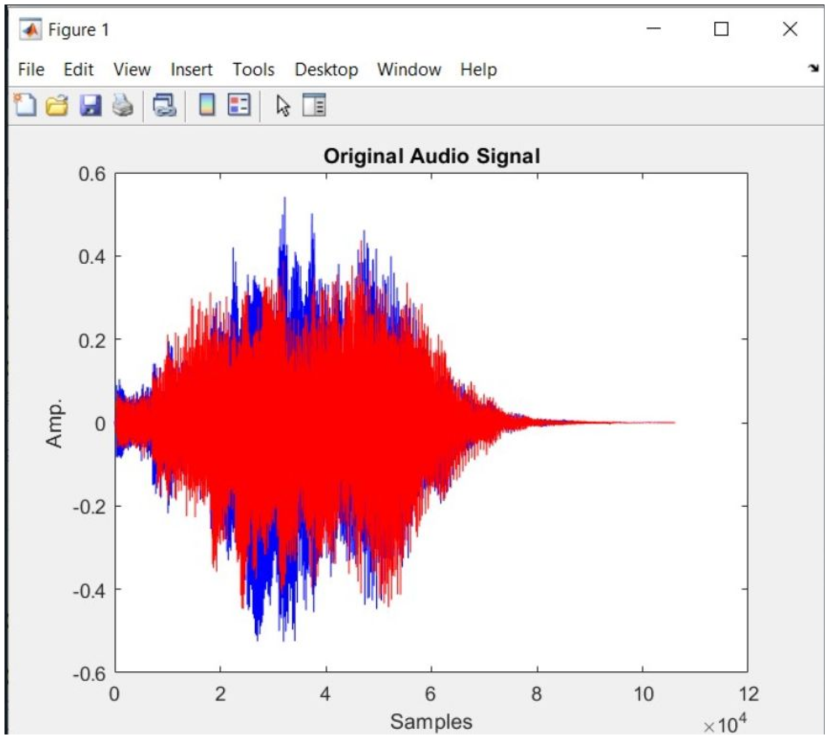


Figure 1
File Edit View Insert Tools Desktop Window Help
Original Audio Signal
Amp.
Samples
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
0
2
4
6
8
10
12
$\times 10^4$

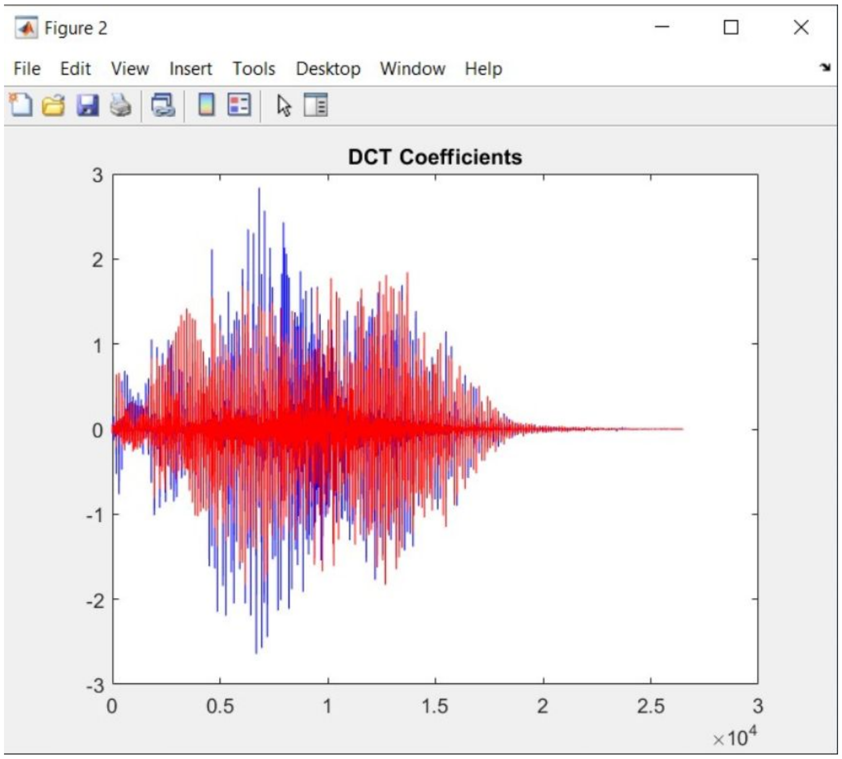


Figure 2
File Edit View Insert Tools Desktop Window Help
DCT Coefficients
3
2
1
0
-1
-2
-3
0
0.5
1
1.5
2
2.5
3
$\times 10^4$

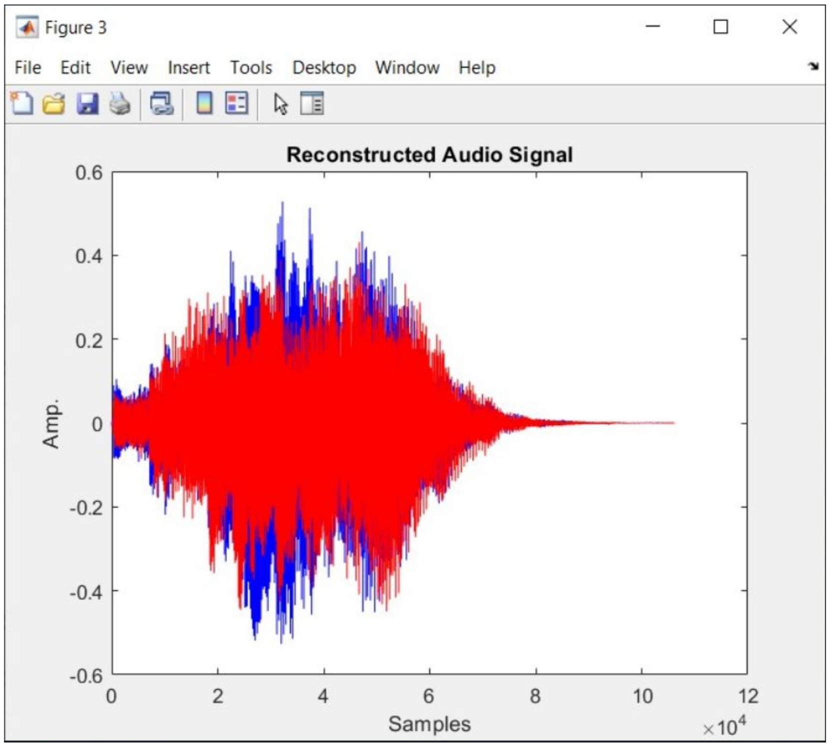


Figure 3
File Edit View Insert Tools Desktop Window Help
Reconstructed Audio Signal
Amp.
Samples
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
0
2
4
6
8
10
12
×10^4

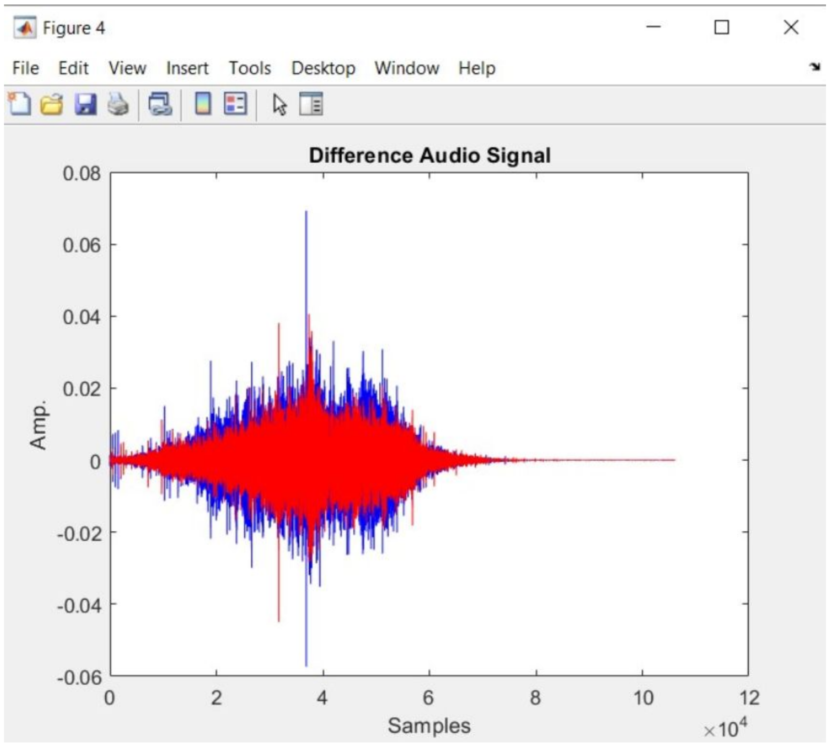


Figure 4
File Edit View Insert Tools Desktop Window Help
Difference Audio Signal
Amp.
Samples
0.08
0.06
0.04
0.02
0
-0.02
-0.04
-0.06
0
2
4
6
8
10
12
×10^4

RESULT

Compression Technique	PSNR	Mean Square Error	SNR	CF
FT	55.7568	2.6566e-06	-0.0014	0.25
DCT	48.0827	1.5550e-05	-0.0083	0.25

CONCLUSION

The Fourier Transform (FT) and Discrete Cosine Transform (DCT) perform similar functions: they both decompose a finite-length discrete-time vector into a sum of scaled-and-shifted basis functions. The difference between the two is the type of basis function used by each transform; the FT uses a set of harmonically-related complex exponential functions, while the DCT uses only (real-valued) cosine functions.

The DISCRETE TRANSFORM method gives a PSNR value between 30-50 which is considered to retain maximum part of the signal and reduce noise. However, the FOURIER TRANSFORM gives a PSNR value greater than 50 and hence the DISCRETE COSINE TRANSFORM method is the better method of compression.

REFERENCES

- James, J., & Thomas, V. J. (2014). A Comparative Study of Speech Compression using Different Transform Techniques. *International Journal of Computer Applications*, *975*, 8887.
- http://www.signal.uu.se/Courses/CourseDirs/SignSyst/rapporter/AudioHBV.pdf
- https://link.springer.com/referenceworkentry/10.1007%2F0-387-30038-4_6
- https://www.uaudio.com/blog/audio-compression-basics/
- https://www.sciencedirect.com/topics/computer-science/audio-compression