```
% 19ucc023
% Mohit Akhouri
% Experiment 7 - Observation 4

% In this code , we take the input of audio file and divide the whole
% audio file into blocks of size 1x256 . Now we apply DCT on each
% individual block and Apply the COMPRESSION ALGORITHM

% COMPRESSION ALGORITHM : In this we decide the threshold ( minimum
% threshold and maximum threshold ) , we reject the coefficient values
% which are between the threshold values and accept the rest. This is
 how
% we achieve compression.

% We repeat the above compression algorithm for different threshold
 values
% and plot the graph between Mean square error (MSE) and Compression
 ratio

clc;
clear all;
close all;

[x,fs] = audioread('input.wav'); % Reading of audio file 'input.wav'

N = size(x,1); % Row size of the input audio signal

index = ceil(N/256); % calculating the index for division of blocks
x_orig = [x ; zeros(256*index-N,1)]; % storing the "padded with zeros"
 original signal in new variable
N = size(x_orig,1); % Re-calculation of size after adjustment of audio
 signal x(t)

% Plot of the original signal input.wav
figure;
plot(x_orig);
xlabel('time(t) ->');
ylabel('x(t) ->');
title('19ucc023 - Mohit Akhouri','Plot of Original Sound Wave
 input.wav ');
grid on;

% sound(x_orig,fs);

% Case 1 = removing coefficients for values between -0.09 and 0.09

x_recon = zeros(N,1); % Initializing variable to store reconstructed
 Audio signal

threshold_1a = 0.09; % Maximum threshold
threshold_1b = -0.09; % Minimum threshold
cnt = 0; % To hold the count of discarded coefficients
```

```matlab
% Main loop algorithm for the compression of audio signal starts here
for i=1:256:N
    x_block = x_orig(i:i+255); % Dividing input audio signal into
 1x256 blocks
    x_block_dct = myCompression(x_block); % Taking DCT of 1x256 block

    % Loop for the checking of coefficients
    % Those coefficients are rejected , which are between the
 threshold
    % values , rest are accepted
    for j=1:256
        if (x_block_dct(j) >= threshold_1b && x_block_dct(j) <=
 threshold_1a)
            x_block_dct(j) = 0;
            cnt = cnt + 1;
        end
    end

    x_block_idct = myDeCompression(x_block_dct);  % Inverse DCT of the
 Compressed Audio block
    x_recon(i:i+255) = x_block_idct; % Storing the IDCT into
 reconstructed wave variable x_recon
end

% Plot of the reconstructed Compressed wave after applying threshold
 values
% for compression
figure;
plot(x_recon);
xlabel('time(t) ->');
ylabel('x_{reconstructed}(t) ->');
title('19ucc023 - Mohit Akhouri','Plot of Reconstructed Sound Wave for
 Case 1 - removing coefficients between -0.09 and 0.09');
grid on;

sound(x_recon,fs); % To hear the compressed audio signal from speakers
audiowrite('Obs4_outputcompression_case1.wav',x_recon,fs); % Writing
 Compressed audio signal to a file

mse_case1 = mse(x_orig,x_recon); % calculation of MSE between Original
 and Reconstructed audio signal for case 1
p_case1 = (N-cnt)/N; % Calculation of Compression ratio for case 1


% Case 2 = removing coefficients for values between -0.5 and 0.5

x_recon = zeros(N,1); % Initializing variable to store reconstructed
 Audio signal

threshold_2a = 0.5; % Maximum threshold
threshold_2b = -0.5; % Minimum threshold
cnt = 0; % To hold the count of discarded coefficients
```

```
% Main loop algorithm for the compression of audio signal starts here
for i=1:256:N
    x_block = x_orig(i:i+255); % Dividing input audio signal into
 1x256 blocks
    x_block_dct = myCompression(x_block); % Taking DCT of 1x256 block

    % Loop for the checking of coefficients
    % Those coefficients are rejected , which are between the
 threshold
    % values , rest are accepted
    for j=1:256
        if (x_block_dct(j) >= threshold_2b && x_block_dct(j) <=
 threshold_2a)
            x_block_dct(j) = 0;
            cnt = cnt + 1;
        end
    end

    x_block_idct = myDeCompression(x_block_dct); % Inverse DCT of the
 Compressed Audio block
    x_recon(i:i+255) = x_block_idct; % Storing the IDCT into
 reconstructed wave variable x_recon
end

% Plot of the reconstructed Compressed wave after applying threshold
 values
% for compression
figure;
plot(x_recon);
xlabel('time(t) ->');
ylabel('x_{reconstructed}(t) ->');
title('19ucc023 - Mohit Akhouri','Plot of Reconstructed Sound Wave for
 Case 2 - removing coefficients between -0.5 and 0.5');
grid on;

sound(x_recon,fs); % To hear the compressed audio signal from speakers
audiowrite('Obs4_outputcompression_case2.wav',x_recon,fs); % Writing
 Compressed audio signal to a file

mse_case2 = mse(x_orig,x_recon); % calculation of MSE between Original
 and Reconstructed audio signal for case 2
p_case2 = (N-cnt)/N; % Calculation of Compression ratio for case 2


% Case 3 = removing coefficients for values between -0.01 and 0.01

x_recon = zeros(N,1); % Initializing variable to store reconstructed
 Audio signal

threshold_3a = 0.01; % Maximum threshold
threshold_3b = -0.01; % Minimum threshold
cnt = 0; % To hold the count of discarded coefficients

% Main loop algorithm for the compression of audio signal starts here
```

```
for i=1:256:N
    x_block = x_orig(i:i+255); % Dividing input audio signal into
 1x256 blocks
    x_block_dct = myCompression(x_block); % Taking DCT of 1x256 block

    % Loop for the checking of coefficients
    % Those coefficients are rejected , which are between the
 threshold
    % values , rest are accepted
    for j=1:256
        if (x_block_dct(j) >= threshold_3b && x_block_dct(j) <=
 threshold_3a)
            x_block_dct(j) = 0;
            cnt = cnt + 1;
        end
    end

    x_block_idct = myDeCompression(x_block_dct); % Inverse DCT of the
 Compressed Audio block
    x_recon(i:i+255) = x_block_idct; % Storing the IDCT into
 reconstructed wave variable x_recon
end

% Plot of the reconstructed Compressed wave after applying threshold
 values
% for compression
figure;
plot(x_recon);
xlabel('time(t) ->');
ylabel('x_{reconstructed}(t) ->');
title('19ucc023 - Mohit Akhouri','Plot of Reconstructed Sound Wave for
 case 3 - removing coefficients between -0.01 and 0.01');
grid on;

sound(x_recon,fs); % To hear the compressed audio signal from speakers
audiowrite('Obs4_outputcompression_case3.wav',x_recon,fs); % Writing
 Compressed audio signal to a file

mse_case3 = mse(x_orig,x_recon); % calculation of MSE between Original
 and Reconstructed audio signal for case 3
p_case3 = (N-cnt)/N; % Calculation of Compression ratio for case 3

% Plotting Graph between MSE and Compression Ratio for different cases

mse_array = zeros(1,3); % Initializing MSE Array
p_array = zeros(1,3); % Initializing Compression Ratio Array

% Storing MSE for different cases in array
mse_array(1) = mse_case1;
mse_array(2) = mse_case2;
mse_array(3) = mse_case3;

% Storing Compression Ratio for different cases in array
p_array(1) = p_case1;
```

```
p_array(2) = p_case2;
p_array(3) = p_case3;

% Plot of MSE vs. Compression Ratio
figure;
stem(mse_array,p_array,'Linewidth',1.5);
xlabel('Compression ratio (\rho) ->');
ylabel('Mean square error (\epsilon) ->');
title('19ucc023 - Mohit Akhouri','Plot of Mean square error (\epsilon)
 vs. Compression Ratio (\rho)');
grid on;
```

Published with MATLAB® R2020b