Record your own Audio, play it back, and plot it using MATLAB

clear all;

close all;

clc;

% *Create an audiorecorder object with sample rate 8000 Hz, 8-bit depth, 1 channel (mono)*

a = audiorecorder(8000, 8, 1);

% Prompt user for the recording time

t = input('Enter the time for which you want to record: ');

% Start recording

disp('Start recording...');

recordblocking(a, t); % Record audio for 't' seconds

disp('End of recording.');

% Play the recorded audio

play(a);

% Get the audio data

y = getaudiodata(a);

% Plot the recorded audio data

figure;

plot(y);

title('Recorded Audio Signal');

xlabel('Sample Number');

ylabel('Amplitude');

grid on;

% Save the audio to a file

audiowrite('mysound.wav', y, 8000);

% Read the saved audio file

[y, fs] = audioread('mysound.wav');

% Play the saved audio

sound(y, fs);

% Time vector for plotting

t = (0:length(y)-1) / fs;

% Plot the audio data with time axis

figure;

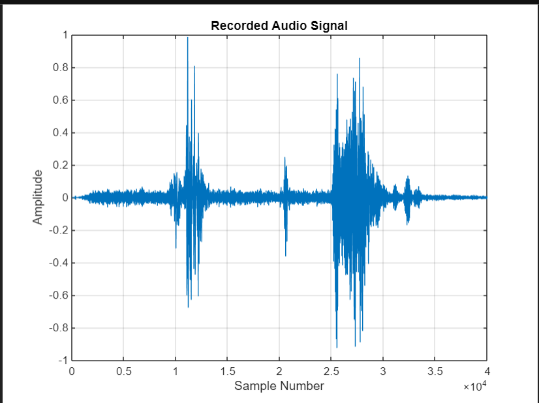
plot(t, y);

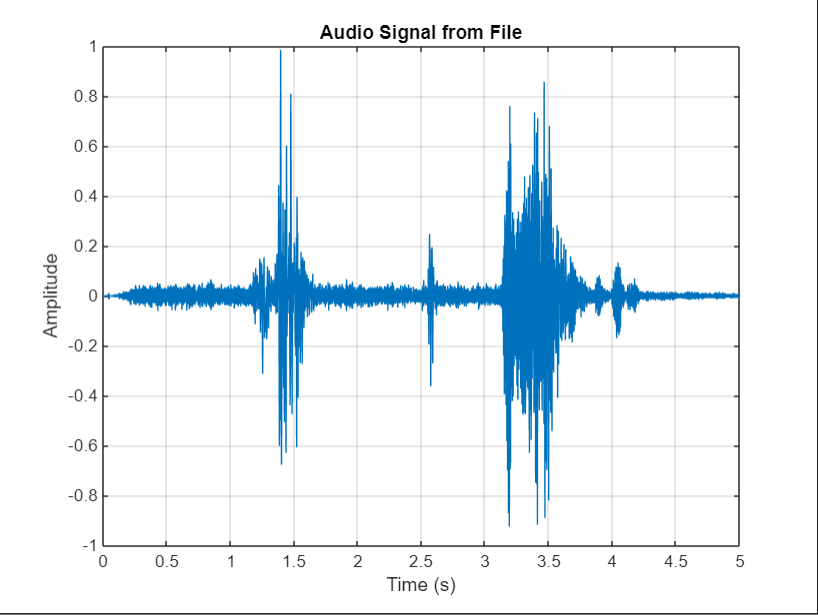
title('Audio Signal from File');

xlabel('Time (s)');

ylabel('Amplitude');

grid on;





Shifting and Scaling of Audio Signal Using MATLAB

clear all;

close all;

clc;

*% Create an audiorecorder object with sample rate 8000 Hz, 8-bit depth, 1 channel (mono)*

a = audiorecorder(8000, 8, 1);

% Prompt user for the recording time

t = input('Enter the time for which you want to record: ');

% Start recording

disp('Start recording...');

recordblocking(a, t); % Record audio for 't' seconds

disp('End of recording.');

% Play the recorded audio

play(a);

% Get the audio data

q = getaudiodata(a);

l = length(q);

% Plot the recorded audio data

figure;

plot(q);

title('Recorded Audio Signal');

xlabel('Sample Number');

ylabel('Amplitude');

grid on;

% User prompt for choices

prompt = 'Enter your choice:\n1. Listen from beginning\n2. Listen from end\n';

x = input(prompt);

while x > 0

if x == 1

% Amplification or attenuation factor

f = input('Enter attenuation factor (<1) or amplification factor (>1). Enter 1 for no change: ');

% Delay in seconds

g = input('Enter delay in seconds. Enter 0 for no change: ');

% Speed control

i = input('1. Slow down\n2. Speed Up\nEnter your choice: ');

e = input('Enter slow down factor or speed up factor. Enter 1 for no change: ');

% Speed up case

if i == 2

% Adjust for delay and speed-up factor

delay = zeros(g \* 8000, 1); % Delay in seconds

r = [delay; q(1:e:end)]; % Speed up by taking every 'e' sample

sound(r \* f, 8000); % Apply amplification/attenuation and play

figure;

plot(r \* f);

title('Modified Audio Signal (Speed Up)');

xlabel('Sample Number');

ylabel('Amplitude');

end

% Slow down case

if i == 1

d = zeros(e \* l, 1); % Slow down by creating space between samples

d(1:e:end) = q; % Put original samples at intervals

delay = zeros(g \* 8000, 1); % Delay in seconds

r = [delay; d]; % Concatenate delay and slowed down signal

sound(r \* f, 8000); % Apply amplification/attenuation and play

figure;

plot(r \* f);

title('Modified Audio Signal (Slow Down)');

xlabel('Sample Number');

ylabel('Amplitude');

end

end

if x == 2

% Flip the audio signal (listen from the end)

q = flipud(q);

% Amplification or attenuation factor

f = input('Enter attenuation factor (<1) or amplification factor (>1). Enter 1 for no change: ');

% Delay in seconds

g = input('Enter delay in seconds. Enter 0 for no change: ');

% Speed control

i = input('1. Slow down\n2. Speed Up\nEnter your choice: ');

e = input('Enter slow down factor or speed up factor. Enter 1 for no change: ');

% Speed up case

if i == 2

delay = zeros(g \* 8000, 1); % Delay in seconds

r = [delay; q(1:e:end)]; % Speed up by taking every 'e' sample

sound(r \* f, 8000); % Apply amplification/attenuation and play

figure;

plot(r \* f);

title('Modified Audio Signal (Speed Up - From End)');

xlabel('Sample Number');

ylabel('Amplitude');

end

% Slow down case

if i == 1

d = zeros(e \* l, 1); % Slow down by creating space between samples

d(1:e:end) = q; % Put original samples at intervals

delay = zeros(g \* 8000, 1); % Delay in seconds

r = [delay; d]; % Concatenate delay and slowed down signal

sound(r \* f, 8000); % Apply amplification/attenuation and play

figure;

plot(r \* f);

title('Modified Audio Signal (Slow Down - From End)');

xlabel('Sample Number');

ylabel('Amplitude');

end

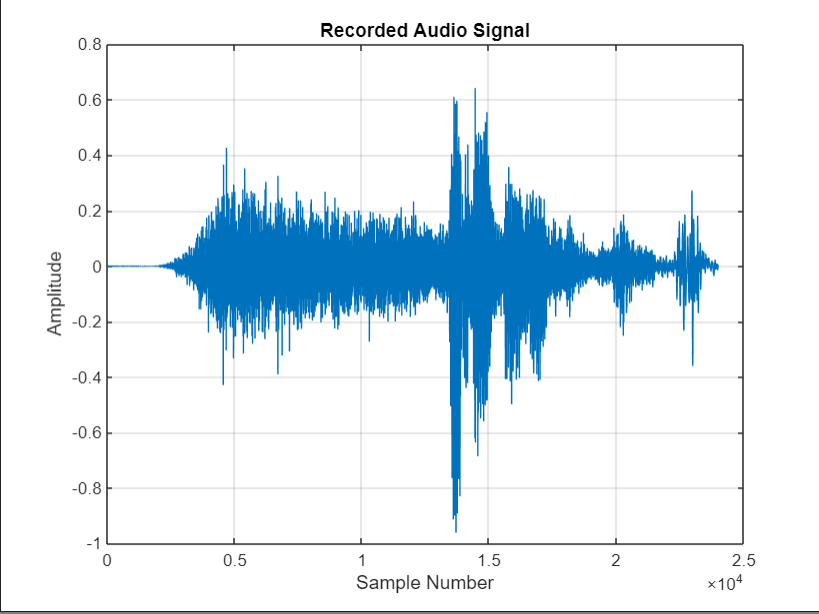
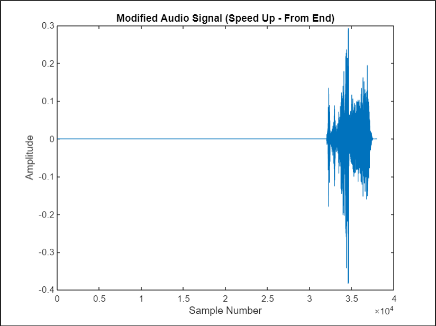
end

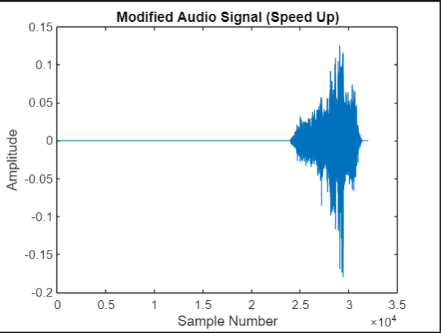
% Ask again for input or exit

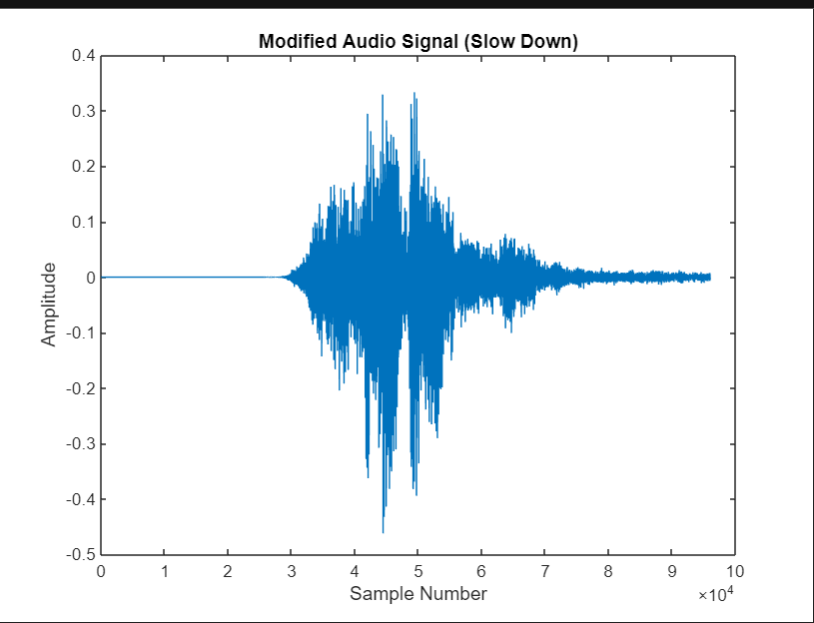
prompt = 'Enter your choice (0 to exit):\n1. Listen from beginning\n2. Listen from end\n';

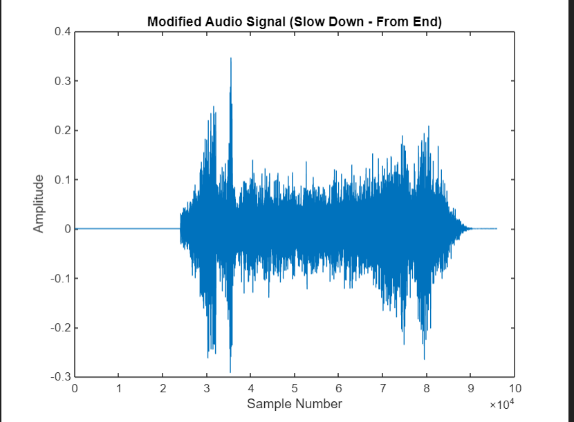
x = input(prompt);

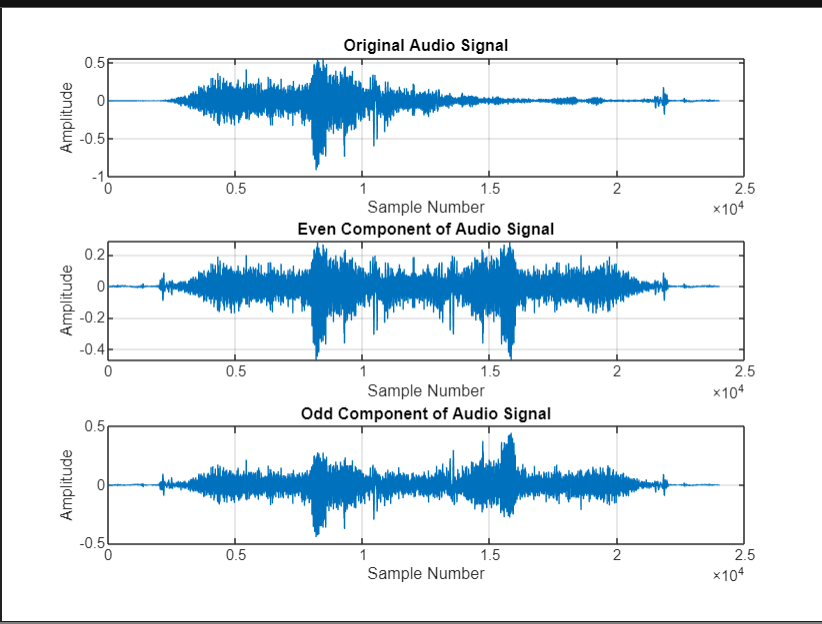
end











Even and Odd Parts of Audio Signals Using MATLAB

clear all;

close all;

clc;

*% Create an audiorecorder object with sample rate 8000 Hz, 8-bit depth, 1 channel (mono)*

a = audiorecorder(8000, 8, 1);

% Prompt user for the recording time

t = input('Enter the time for which you want to record: ');

% Start recording

disp('Start recording...');

recordblocking(a, t); % Record audio for 't' seconds

disp('End of recording.');

% Play the recorded audio

play(a);

% Get the audio data

q = getaudiodata(a);

l = length(q);

% Time vector for plotting

n = 0:l-1;

% Even and odd components computation

q\_flipped = flipud(q); % Flipped version of the audio (x[-n])

q\_even = (q + q\_flipped) / 2; % Even component

q\_odd = (q - q\_flipped) / 2; % Odd component

% Play the even component of the sound

disp('Playing even component...');

sound(q\_even, 8000);

pause(t + 1); % Wait until the audio finishes playing

% Play the odd component of the sound

disp('Playing odd component...');

sound(q\_odd, 8000);

pause(t + 1); % Wait until the audio finishes playing

% Plot original audio signal

figure;

subplot(3,1,1);

plot(n, q);

title('Original Audio Signal');

xlabel('Sample Number');

ylabel('Amplitude');

grid on;

% Plot even component

subplot(3,1,2);

plot(n, q\_even);

title('Even Component of Audio Signal');

xlabel('Sample Number');

ylabel('Amplitude');

grid on;

% Plot odd component

subplot(3,1,3);

plot(n, q\_odd);

title('Odd Component of Audio Signal');

xlabel('Sample Number');

ylabel('Amplitude');

grid on