Mohammad Elham Amin

21701543

MATLAB Assignment 2

EEE 391

1230/2021

1. DTMF Signal and Transceiver

1.1. Write DTMFTRA

```
function [output] = DTMFTRA(numbers)
    % Low and High group frequencies
    fr_low = [697 770 852 941];
    fr_high = [1209 1336 1477];
    % Sampling period, and frequency
    fs = 16384;
    % Output to hold the tones for each number
    x = [];

    duration_per_tone = 0.5;
    t = linspace(0, duration_per_tone, duration_per_tone * fs);
    % Find the index in the low and high group frequency lists
    % for each of the numbers
    for N = 1: length(numbers)
        number = numbers(N);
        switch number
            case 0, fl_idx = 4; fh_idx = 1;
            case '#', fl_idx = 4; fh_idx = 2;
            case '*', fl_idx = 4; fh_idx = 3;
            otherwise
                fh_idx = mod(number -1, 3) + 1; fl_idx = (number - fh_idx) / 3 + 1;
        end
        % Sum the high and low group freq for the number and play it
        % t = 0:1 / fs:0.5;
        y1 = cos(2 * pi * fr_low(fl_idx) * t);
        y2 = cos(2 * pi * fr_high(fh_idx) * t);
        d = (y1 + y2) / 2;
        x = [x d];
    end
    output = x;
end
```

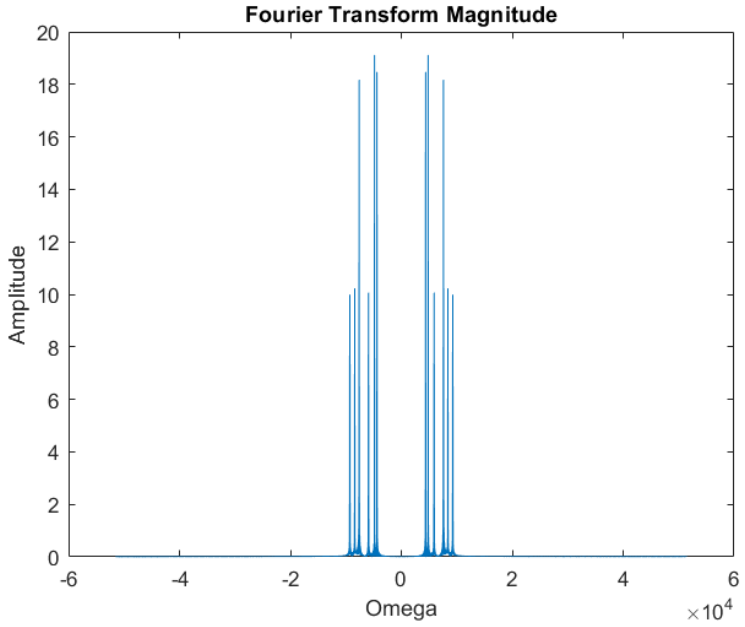
1.2. Test ID number

```
% ID 21701543
x = DTMFTRA([0 1 5 4 3]);
soundsc(x, 16384);
```

1.3. Compute Fourier Transform Magnitude

```
% ID 21701543
x = DTMFTRA([0 1 5 4 3]);
soundsc(x, 16384);

X = FT(x);
% Changed the linespace args
% to make both amtrices the same length
omega = linspace(-16384 * pi, 16384 * pi, 16384
 * 2.5 + 1);
omega = omega(1:end - 1);
figure
plot(omega, abs(X));
xlabel('Omega');
ylabel('Amplitude');
title('Fourier Transform Magnitude')
```

The resulted plot:



From the plot above we can see the magnitude of each number signal. If we click on one of the peaks, we notice that the frequency (x axis) is the same as corresponding number frequency given in the assignment.

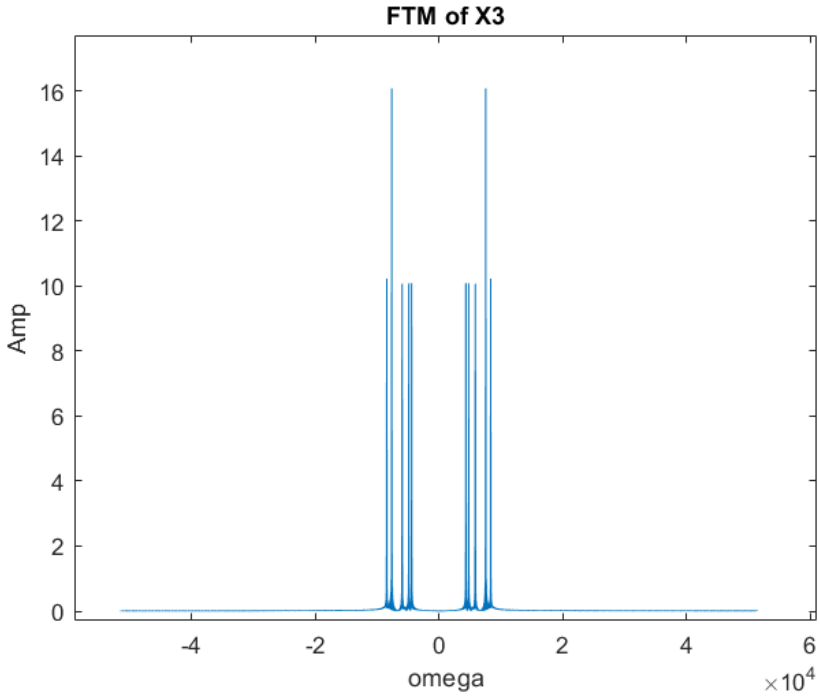
1.4. Define new signal

The new signal is defined in the following function, which is used to plot the frequency as well.

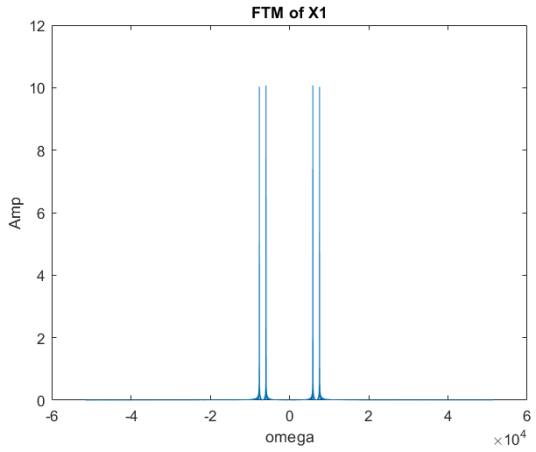
```
function [out] = freqPlotter(x, idx)
    % Define frequency and omega
    freq = 16384;
    omega = linspace(-16384 * pi, 16384 * pi,
16384 * 2.5 + 1);
    omega = omega(1:end - 1);

% Build rectangular signal for the given portio
n of signal

% i.e. idx = 1 ⇒ a rect signal between 0 ≤ t
 ≤ 0.5
    recX1 = [ones(1, (idx * 0.5) * freq) zeros(
1, (2.5 - idx * 0.5) * freq)]
    plot(recX1)
    % Multiply matrices and find FT
    X1w = x .* recX1;
    X1w = FT(X1w);
    out = X1w;
    figure
    plot(omega,abs(X1w));
    xlabel('omega');
    ylabel('Amp');
    title(sprintf('FTM of X%d', idx))
end
```

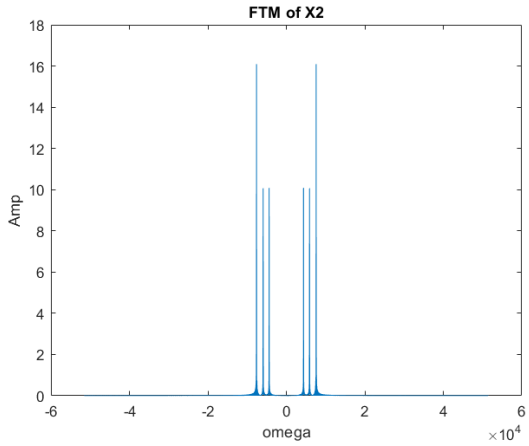
1.5. Compute X_3 (w)

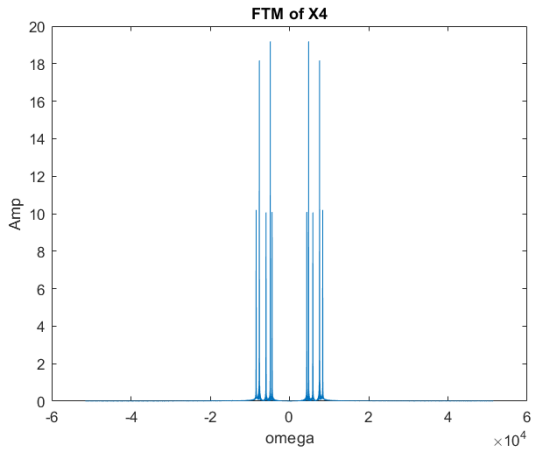
Calling freqPlotter (x, 3), which will compute the portion for number 5, we get the following plot. The freqPlotter function is used to build the plots in the next part for the remaining digits as well.

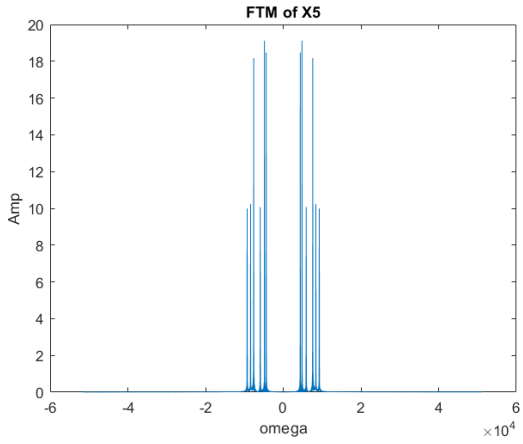


1.6. Repeat for other X_i. I = 0, 1, 4, 3









1.7. Comments

It seems that the first method is more practical than the latter one. That is because, it is relatively easy to use the Fourier Transform and get the desired result. However, for the other method, we need to find rectangular signals of different chunks of the signal and plot them, which requires more work and is prone to mistakes.

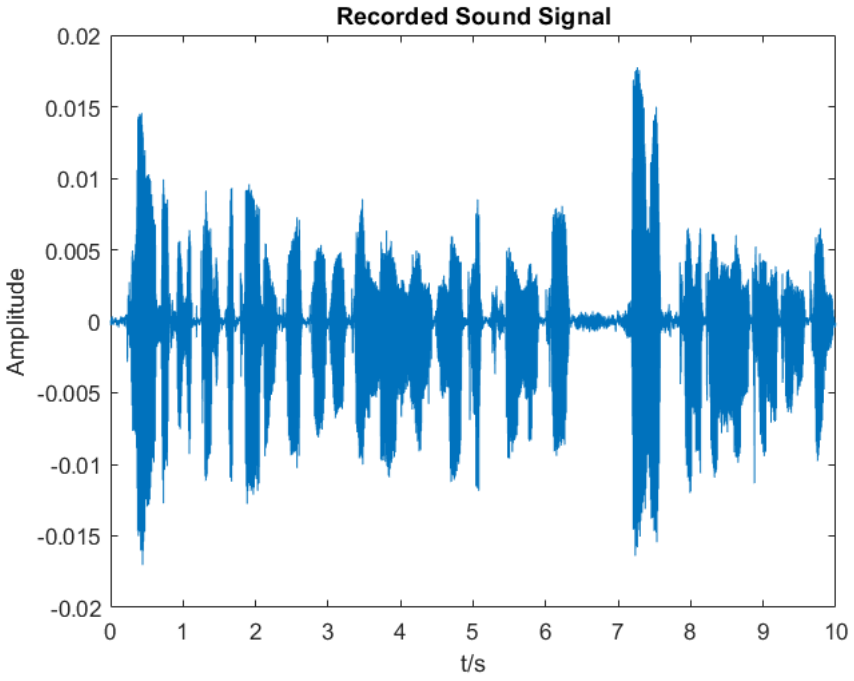
2. Part 2 (Echo Cancellation)

2.1. MATLAB code for recording the sound

```
%%% We use built-in audiorecorder function to record the voice %%%%

% Define the sampling frequency, bits per sample and number of channels
% The following code is partially taken from the Mathworks
% documentation page for Matlab audiorecorder.
fs = 8192;
bitsPerSample = 16;
numOfChannels = 2;
recordedSound = audiorecorder(fs, bitsPerSample, 1);
% Record a 10 second voice.
disp('Start speaking.');
recordblocking(recordedSound, 10);
disp('End of Recording.');
soundData = getaudiodata(recordedSound);
t = 0:1/fs:10-1/fs;
sound(soundData, fs);
figure
plot(t, soundData);
xlabel('t/s');
ylabel('Amplitude');
title('Recorded Sound Signal')

```

2.2. Plot the recorded sound

Sentence: Hello, this is a test recording for EE391 MATLAB assignment, fall of 2021-2022.

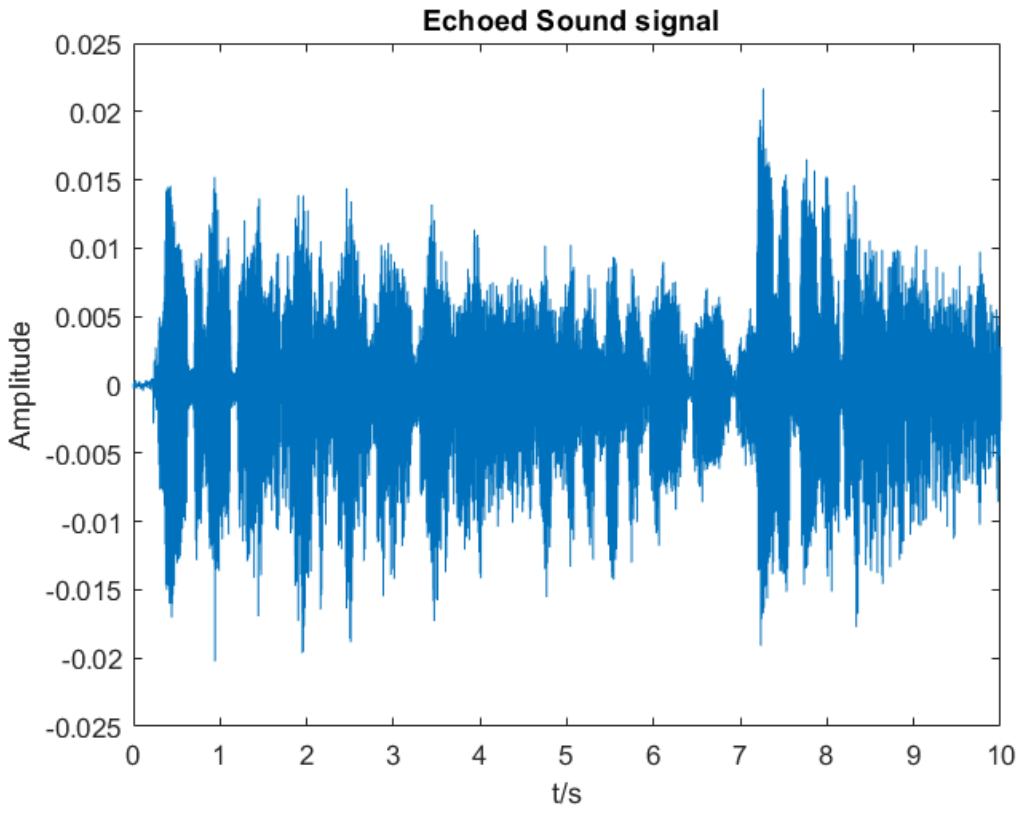


2.3. Echo Generation MATLAB code

```
function [out] = generateEcho(soundData, fs)
    x = transpose(soundData);
    len = length(x);
    Ai = [0.8, 0.6, 0.4, 0.2, 0.05];
    ti = [0.5, 1, 1.5, 2, 3];
    out = zeros(1, len);

    for i=1:length(Ai)
        xi = [zeros(1, ti(i) * fs) Ai(i) * x(1, 1:len-ti(i) * fs)];
        out = out + xi;
    end
    out = x + out;
end
```

2.4. Plot echoed sound

```
%%%% Generate Echoed Sound %%%%%%%%
fs = 8192;
T = 10;
t=0:1/fs:T-1/fs;
% Generate y from x
M = 5;
soundData = transpose(soundData);
y = generateEcho(soundData, fs);

% Plot y
figure
plot(t,y);
xlabel('t/s');
ylabel('Amplitude');
title(' Echoed Sound signal');
sound(y, fs);

```

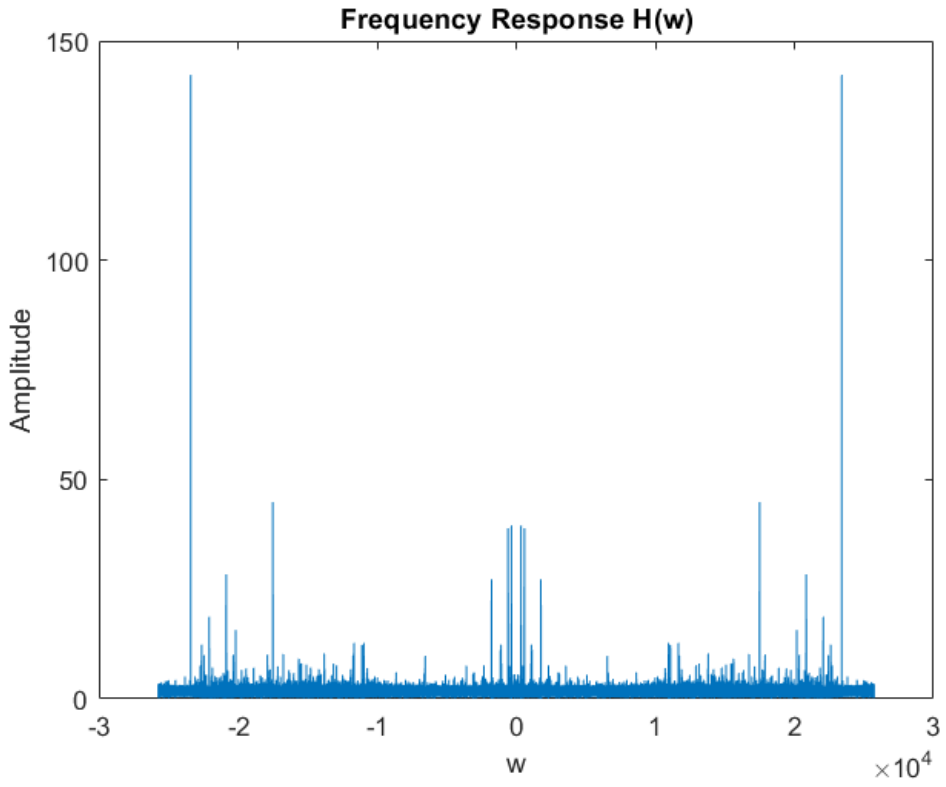


2.5. Listen to the echoed sound and describe

The echoed sound generated in the previous part is a sound that arrives later than the original sound due to the added delay. Therefore, we can see that the sound is being repeated.

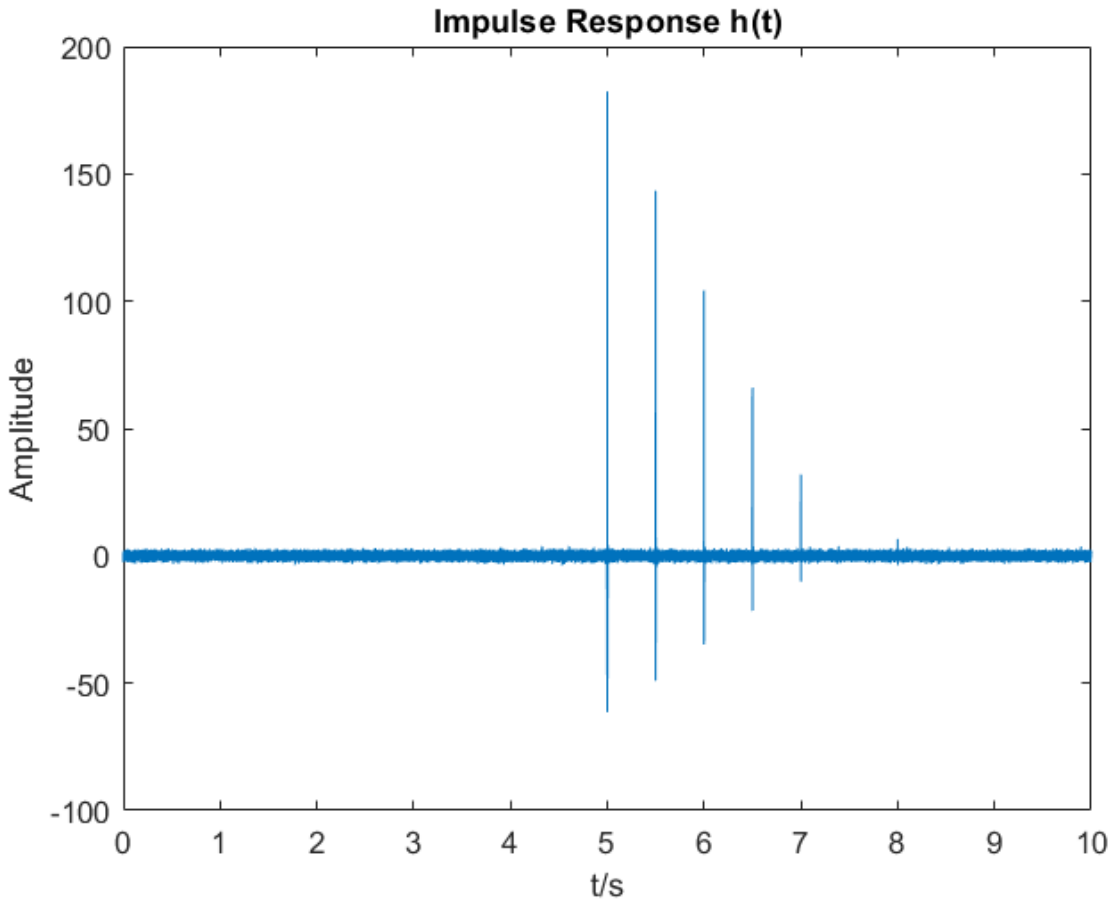
2.6. Compute frequency response ($H_{(w)}$)

```
%%%%% Compute frequency response %%%%%
omega=linspace(-fs * pi, fs * pi, fs * 10 + 1);
omega=omega(1:end-1);
Y = FT(y);
X = FT(transpose(soundData));

H = Y./X;
figure
plot(omega,abs(H));
xlabel('w');
ylabel('Amplitude');
title('Frequency Response H(w)');
```



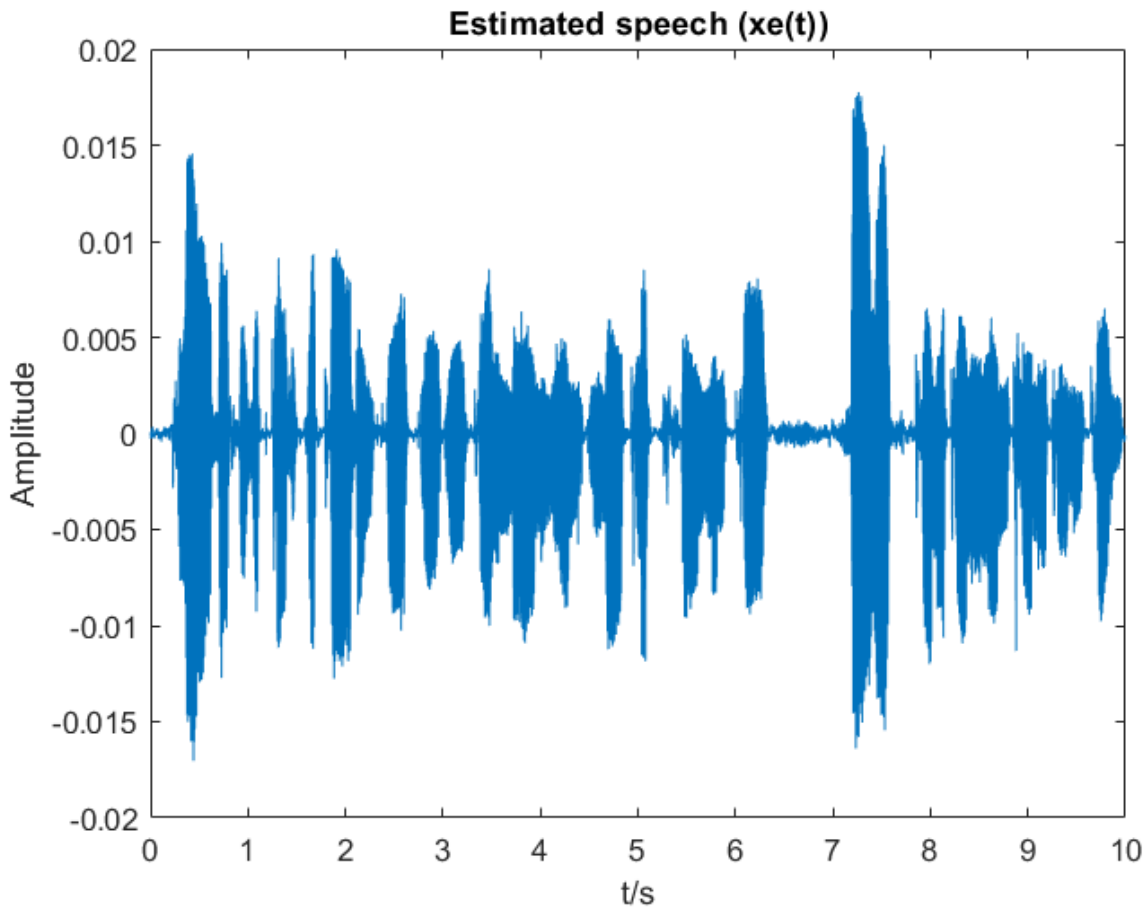
2.7. Compute impulse response $h_{(t)}$

```
%%%%%% Compute the impulse response %%%%%%
h = IFT(H)
figure
plot(t, h)
xlabel('t/s');
ylabel('Amplitude');
title('Impulse Response h(t)');
```



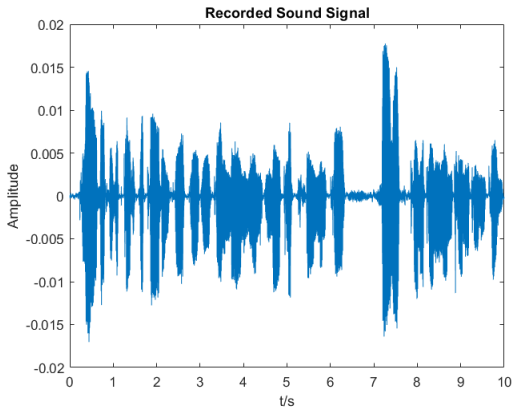
2.8. Compute estimated sound (Compute Xe and $x_e(t)$)

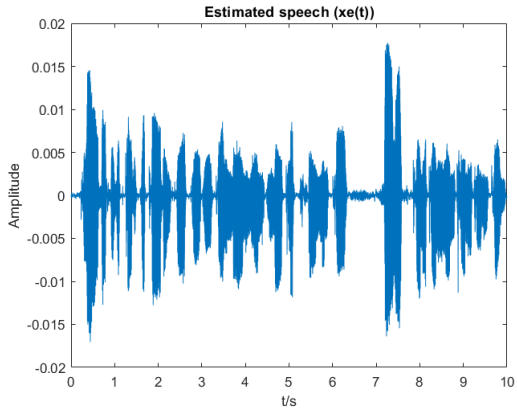
```
%%%%% Compute the estimated (echo cancelled) sound %%%%%%
Xe = Y./H
xe = IFT(Xe)
plot(t, xe);
xlabel('t/s');
ylabel('Amplitude');
title('Estimated speech (xe(t))');
sound(xe, fs);
```



2.9. Comments

The generated sound after echo cancellation is the original sound. This can be confirmed by listening to the sound or comparing the plots of both sounds. As it can be seen, both plots are the same without any difference.





Therefore, we can conclude that the extra sound (repeated sound) that we have added in the previous part is now removed from the echoed sound and we have achieved the original sound back. The following plots are the original sound and the echo cancelled sound.

PART 1

1.1 DTMFRA function

```
function [output] = DTMFTRA(numbers)
    % Low and High group frequencies
    fr_low = [697 770 852 941];
    fr_high = [1209 1336 1477];
    % Sampling period, and frequency
    fs = 16384;
    % Output to hold the tones for each number
    x = [];

    duration_per_tone = 0.5;
    t = linspace(0, duration_per_tone, duration_per_tone * fs);
    % Find the index in the low and high group frequency lists
    % for each of the numbers
    for N = 1: length(numbers)
        number = numbers(N);
        switch number
            case 0, fl_idx = 4; fh_idx = 1;
            case '#', fl_idx = 4; fh_idx = 2;
            case '*', fl_idx = 4; fh_idx = 3;
            otherwise
                fh_idx = mod(number -1, 3) + 1; fl_idx = (number - fh_idx) / 3 + 1;
        end
        % Sum the high and low group freq for the number and play it
        % t = 0:1 / fs:0.5;
        y1 = cos(2 * pi * fr_low(fl_idx) * t);
        y2 = cos(2 * pi * fr_high(fh_idx) * t);
        d = (y1 + y2) / 2;
        x = [x d];
    end
    output = x;
end
```

1.2 Compute Fourier Transform Magnitude

```
% ID 21701543
x = DTMFTRA([0 1 5 4 3]);
soundsc(x, 16384);

X = FT(x);
% Changed the linespace args
% to make both amtrices the same length
omega = linspace(-16384 * pi, 16384 * pi, 16384 * 2.5 + 1);
omega = omega(1:end - 1);
figure
plot(omega, abs(X));
xlabel('Omega');
ylabel('Amplitude');
title('Fourier Transform Magnitude')
```

1.3 Define new signal (freqPlotter function)

```
function [out] = freqPlotter(x, idx)
    % Define frequency and omega
    freq = 16384;
    omega = linspace(-16384 * pi, 16384 * pi, 16384 * 2.5 + 1);
    omega = omega(1:end - 1);
    % Build rectangular signal for the given portion of signal
    % i.e. idx = 1 => a rect signal between 0 <= t <= 0.5
    recX1 = [ones(1, (idx * 0.5) * freq) zeros(1, (2.5 - idx * 0.5) * freq)]
    plot(recX1)
    % Multiply matrices and find FT
    X1w = x .* recX1;
    X1w = FT(X1w);
    out = X1w;
    figure
    plot(omega,abs(X1w));
    xlabel('omega');
    ylabel('Amp');
    title(sprintf('FTM of X%d', idx))
end
```

1.4 Content of the main script

```
% ID 21701543
x = DTMFTRA([0 1 5 4 3]);
soundsc(x, 16384);

X = FT(x);
% Changed the linespace args
% to make both amtrices the same length
omega = linspace(-16384 * pi, 16384 * pi, 16384 * 2.5 + 1);
omega = omega(1:end - 1);
figure
plot(omega, abs(X));
xlabel('Omega');
ylabel('Amplitude');
title('Fourier Transform Magnitude')


x = DTMFTRA([0 1 5 4 3]);
freq = 16384;

%%%%%%%%% Compute Xi(w) for i = 1, 2, 3, 4, 5 %%%%%%%%%%
x1w = freqPlotter(x, 3);
x2w = freqPlotter(x, 5);
x3w = freqPlotter(x, 2);
x4w = freqPlotter(x, 4);
x5w = freqPlotter(x, 5);
```

Part 2

2.1. MATLAB Code for Recording the Sound

```
% Define the sampling frequency, bits per sample and number of channels
% The following code is partially taken from the Mathworks
% documentation page for Matlab audiorecorder.
fs = 8192;
bitsPerSample = 16;
numOfChannels = 2;
recordedSound = audiorecorder(fs, bitsPerSample, 1);
% Record a 10 second voice.
disp('Start speaking.');
recordblocking(recordedSound, 10);
disp('End of Recording.');
soundData = getaudiodata(recordedSound);
t = 0:1/fs:10-1/fs;
sound(soundData, fs);
figure
plot(t, soundData);
xlabel('t/s');
ylabel('Amplitude');
title('Recorded Sound Signal')
```

2.2. Echo Generation code

```
function [out] = generateEcho(soundData, fs)
    x = transpose(soundData);
    len = length(x);
    Ai = [0.8, 0.6, 0.4, 0.2, 0.05];
    ti = [0.5, 1, 1.5, 2, 3];
    out = zeros(1, len);

    for i=1:length(Ai)
        xi = [zeros(1, ti(i) * fs) Ai(i) * x(1, 1:len-ti(i) * fs)];
        out = out + xi;
    end
    out = x + out;
end
```

2.3. Plot Echoed Sound

```
%%%%% Generate Echoed Sound %%%%%%%%%
fs = 8192;
T = 10;
t=0:1/fs:T-1/fs;
% Generate y from x
M = 5;
y = generateEcho(soundData, fs);

% Plot y
figure
plot(t,y);
```

```
xlabel('t/s');
ylabel('Amplitude');
title(' Echoed Sound signal');
sound(y, fs);
```

2.4. Compute Frequency Response

```
%%%%%% Compute frequency response %%%%%%
omega=linspace(-fs * pi, fs * pi, fs * 10 + 1);
omega=omega(1:end-1);
Y = FT(y);
X = FT(transpose(soundData));

H = Y./X;
% figure
plot(omega, abs(H));
xlabel('w');
ylabel('Amplitude');
title('Frequency Response H(w)');
```

2.5. Compute Impulse Response

```
%%%%%% Compute the impulse response %%%%%%
h = IFT(H)
figure
plot(t, h)
xlabel('t/s');
ylabel('Amplitude');
title('Impulse Response h(t)');
```

2.6. Compute Estimated Sound

```
%%%%% Compute the estimated (echo cancelled) sound %%%%%%
Xe = Y./H
xe = IFT(Xe)
plot(t, xe);
xlabel('t/s');
ylabel('Amplitude');
title('Estimated speech (xe(t))');
sound(xe, fs);
```