MODELING FREQUENCY DIVISION MULTIPLEXING AND DE-MULTIPLEXING USING MATLAB

LAB # 09

Fall 2023

CSE-402L

Digital Signal Processing Lab

Submitted by: **AIMAL KHAN**

Registration No.: **21PWCSE1996**

Class Section: **A**

"On my honor, as student of University of Engineering and Technology, I have neither given nor received unauthorized assistance on this academic work."

Student Signature: [illegible]

Submitted to:

Dr. Yasir Saleem Afridi.

Sunday, January 7, 2024

Department of Computer Systems Engineering

University of Engineering and Technology, Peshawar

CSE 402L: Digital Signal Processing

Demonstration of Concepts	**Poor (Does not meet expectation (1))** The student failed to demonstrate a clear understanding of the assignment concepts	**Fair (Meet Expectation (2-3))** The student demonstrated a clear understanding of some of the assignment concepts	**Good (Exceeds Expectation (4-5)** The student demonstrated a clear understanding of the assignment concepts	**Score** **30%**
Accuracy	The student completed (<50%) tasks and provided MATLAB code and/or Simulink models with errors. Outputs shown are not correct in form of graphs (no labels) and/or tables along with incorrect analysis or remarks.	The student completed partial tasks (50% - <90%) with accurate MATLAB code and/or Simulink models. Correct outputs are shown in form of graphs (without labels) and/or tables along with correct analysis or remarks.	The student completed all required tasks (90%-100%) with accurate MATLAB code and/or Simulink models. Correct outputs are shown in form of labeled graphs and/or tables along with correct analysis or remarks.	**30%**
Following Directions	The student clearly failed to follow the verbal and written instructions to successfully complete the lab	The student failed to follow the some of the verbal and written instructions to successfully complete all requirements of the lab	The student followed the verbal and written instructions to successfully complete requirements of the lab	**20%**
Time Utilization	The student failed to complete even part of the lab in the allotted amount of time	The student failed to complete the entire lab in the allotted amount of time	The student completed the lab in its entirety in the allotted amount of time	**20%**

Dr. Yasir Saleem Afridi

Modeling Frequency Division Multiplexing and De-Multiplexing Using MATLAB

Objectives:

Implement the following steps in Matlab to Multiplex three input voice signals at the transmitter end and Demultiplex and play them back at the Receiver end. Add random noise to the signal while propagating via the channel.

Tasks:

Task: Full code of the mentioned task…

Code:

```
clc;
clear;
close all;

% --- Transmitter Section ---

%% Import voice files
[maleVoice, Fs] = audioread('funniest-thing.wav');
[femaleVoice, ~] = audioread('self-destruct-sequence.wav');
[randomVoice, ~] = audioread('monkey-scream.wav');

%% Play each voice for a fixed duration (5 seconds)
playDuration = 5; % seconds
numSamples = Fs * playDuration; % Number of samples to play

% Play original voice segments
sound(maleVoice(1:numSamples), Fs);
pause(playDuration + 1);

sound(femaleVoice(1:numSamples), Fs);
pause(playDuration + 1);

sound(randomVoice(1:numSamples), Fs);
pause(playDuration + 1);

%% Spectrum Analysis
% Perform FFT and plot the spectrum of each voice
fftMaleVoice = fft(maleVoice);
fftFemaleVoice = fft(femaleVoice);
fftRandomVoice = fft(randomVoice);
```

```
% Calculate magnitude spectrum
magSpectrumMale = abs(fftMaleVoice/length(maleVoice));
magSpectrumFemale = abs(fftFemaleVoice/length(femaleVoice));
magSpectrumRandom = abs(fftRandomVoice/length(randomVoice));

% Only take the first half of frequencies (symmetry of FFT)
magSpectrumMale = magSpectrumMale(1:length(maleVoice)/2+1);
magSpectrumFemale = magSpectrumFemale(1:length(femaleVoice)/2+1);
magSpectrumRandom = magSpectrumRandom(1:length(randomVoice)/2+1);

% Double non-DC components
magSpectrumMale(2:end-1) = 2 * magSpectrumMale(2:end-1);
magSpectrumFemale(2:end-1) = 2 * magSpectrumFemale(2:end-1);
magSpectrumRandom(2:end-1) = 2 * magSpectrumRandom(2:end-1);

% Frequency vector for plotting
f = Fs*(0:(length(maleVoice)/2))/length(maleVoice);

% Plotting the spectrum and time-domain signal
figure;
subplot(3,2,1);
plot(maleVoice);
title('Male Voice (Time Domain)');
xlabel('Samples');
ylabel('Amplitude');

subplot(3,2,2);
plot(magSpectrumMale);
title('Spectrum of Male Voice');
xlabel('Frequency (Hz)');
ylabel('Magnitude');

subplot(3,2,3);
plot(femaleVoice);
title('Female Voice (Time Domain)');
xlabel('Samples');
ylabel('Amplitude');

subplot(3,2,4);
plot(magSpectrumFemale);
title('Spectrum of Female Voice');
xlabel('Frequency (Hz)');
ylabel('Magnitude');

subplot(3,2,5);
plot(randomVoice);
title('Random Voice (Time Domain)');
xlabel('Samples');
ylabel('Amplitude');
```

```
subplot(3,2,6);
plot(magSpectrumRandom);
title('Spectrum of Random Voice');
xlabel('Frequency (Hz)');
ylabel('Magnitude');

%% Low-Pass Filtering (LPF)
% Design LPF (Butterworth filter as example)
% Design LPF (Butterworth filter as example)
cutoffFrequency = 3000; % Hz
filterOrder = 6;
[b, a] = butter(filterOrder, cutoffFrequency/(Fs/2), 'low');

% Apply LPF to each voice
lpfMaleVoice = filter(b, a, maleVoice);
lpfFemaleVoice = filter(b, a, femaleVoice);
lpfRandomVoice = filter(b, a, randomVoice);

% Perform FFT on the filtered voices
fftLpfMaleVoice = fft(lpfMaleVoice);
fftLpfFemaleVoice = fft(lpfFemaleVoice);
fftLpfRandomVoice = fft(lpfRandomVoice);

% Calculate magnitude spectrum for each filtered voice
magSpectrumLpfMale = abs(fftLpfMaleVoice/length(lpfMaleVoice));
magSpectrumLpfFemale =
abs(fftLpfFemaleVoice/length(lpfFemaleVoice));
magSpectrumLpfRandom =
abs(fftLpfRandomVoice/length(lpfRandomVoice));

% Only take the first half of frequencies (due to FFT symmetry)
magSpectrumLpfMale = magSpectrumLpfMale(1:length(lpfMaleVoice)/2+1);
magSpectrumLpfFemale =
magSpectrumLpfFemale(1:length(lpfFemaleVoice)/2+1);
magSpectrumLpfRandom =
magSpectrumLpfRandom(1:length(lpfRandomVoice)/2+1);

% Double non-DC components for proper magnitude
magSpectrumLpfMale(2:end-1) = 2 * magSpectrumLpfMale(2:end-1);
magSpectrumLpfFemale(2:end-1) = 2 * magSpectrumLpfFemale(2:end-1);
magSpectrumLpfRandom(2:end-1) = 2 * magSpectrumLpfRandom(2:end-1);

% Frequency vector for plotting
fLpf = Fs*(0:(length(lpfMaleVoice)/2))/length(lpfMaleVoice);

% Plot the time-domain and frequency-domain signals
figure;
subplot(3,2,1);
plot(lpfMaleVoice);
```

```
title('Low-Pass Filtered Male Voice (Time Domain)');
xlabel('Samples');
ylabel('Amplitude');

subplot(3,2,2);
plot(magSpectrumLpfMale);
title('Spectrum of LPF Male Voice');
xlabel('Frequency (Hz)');
ylabel('Magnitude');

subplot(3,2,3);
plot(lpfFemaleVoice);
title('Low-Pass Filtered Female Voice (Time Domain)');
xlabel('Samples');
ylabel('Amplitude');

subplot(3,2,4);
plot(magSpectrumLpfFemale);
title('Spectrum of LPF Female Voice');
xlabel('Frequency (Hz)');
ylabel('Magnitude');

subplot(3,2,5);
plot(lpfRandomVoice);
title('Low-Pass Filtered Random Voice (Time Domain)');
xlabel('Samples');
ylabel('Amplitude');

subplot(3,2,6);
plot(magSpectrumLpfRandom);
title('Spectrum of LPF Random Voice');
xlabel('Frequency (Hz)');
ylabel('Magnitude');

%% Play the filtered voices for 5 seconds each
sound(lpfMaleVoice(1:numSamples), Fs);
pause(playDuration + 1);

sound(lpfFemaleVoice(1:numSamples), Fs);
pause(playDuration + 1);

sound(lpfRandomVoice(1:numSamples), Fs);
pause(playDuration + 1);

%%  Modulation Section
% Define carrier frequencies (fraction of sampling rate)
carrierFreqMale = Fs/4;
carrierFreqFemale = Fs/5;
carrierFreqRandom = Fs/6;
```

```
% Amplitude Modulation (AM)
amMaleVoice = ammod(lpfMaleVoice, carrierFreqMale, Fs);
amFemaleVoice = ammod(lpfFemaleVoice, carrierFreqFemale, Fs);
amRandomVoice = ammod(lpfRandomVoice, carrierFreqRandom, Fs);

% Perform FFT on the AM modulated voices
fftAmMaleVoice = fft(amMaleVoice);
fftAmFemaleVoice = fft(amFemaleVoice);
fftAmRandomVoice = fft(amRandomVoice);

% Calculate magnitude spectrum for each AM voice
magSpectrumAmMale = abs(fftAmMaleVoice/length(amMaleVoice));
magSpectrumAmFemale = abs(fftAmFemaleVoice/length(amFemaleVoice));
magSpectrumAmRandom = abs(fftAmRandomVoice/length(amRandomVoice));

% Only take the first half of frequencies (due to FFT symmetry)
magSpectrumAmMale = magSpectrumAmMale(1:length(amMaleVoice)/2+1);
magSpectrumAmFemale =
magSpectrumAmFemale(1:length(amFemaleVoice)/2+1);
magSpectrumAmRandom =
magSpectrumAmRandom(1:length(amRandomVoice)/2+1);

% Double non-DC components for proper magnitude
magSpectrumAmMale(2:end-1) = 2 * magSpectrumAmMale(2:end-1);
magSpectrumAmFemale(2:end-1) = 2 * magSpectrumAmFemale(2:end-1);
magSpectrumAmRandom(2:end-1) = 2 * magSpectrumAmRandom(2:end-1);

% Frequency vector for plotting
fAm = Fs*(0:(length(amMaleVoice)/2))/length(amMaleVoice);

% Plot the time-domain and frequency-domain signals
figure;
subplot(3,2,1);
plot(amMaleVoice);
title('AM Modulated Male Voice (Time Domain)');
xlabel('Samples');
ylabel('Amplitude');

subplot(3,2,2);
plot(magSpectrumAmMale);
title('Spectrum of AM Male Voice');
xlabel('Frequency (Hz)');
ylabel('Magnitude');

subplot(3,2,3);
plot(amFemaleVoice);
title('AM Modulated Female Voice (Time Domain)');
xlabel('Samples');
ylabel('Amplitude');
```

```
subplot(3,2,4);
plot(magSpectrumAmFemale);
title('Spectrum of AM Female Voice');
xlabel('Frequency (Hz)');
ylabel('Magnitude');

subplot(3,2,5);
plot(amRandomVoice);
title('AM Modulated Random Voice (Time Domain)');
xlabel('Samples');
ylabel('Amplitude');

subplot(3,2,6);
plot(magSpectrumAmRandom);
title('Spectrum of AM Random Voice');
xlabel('Frequency (Hz)');
ylabel('Magnitude');

% --- Channel Section ---
%% Adding Noise
noiseLevel = 0.01; % Noise amplitude
noiseMale = noiseLevel * randn(size(amMaleVoice));
noiseFemale = noiseLevel * randn(size(amFemaleVoice));
noiseRandom = noiseLevel * randn(size(amRandomVoice));

% Noisy AM signals
noisyAmMaleVoice = amMaleVoice + noiseMale;
noisyAmFemaleVoice = amFemaleVoice + noiseFemale;
noisyAmRandomVoice = amRandomVoice + noiseRandom;

% Perform FFT on the noisy AM modulated voices
fftNoisyAmMaleVoice = fft(noisyAmMaleVoice);
fftNoisyAmFemaleVoice = fft(noisyAmFemaleVoice);
fftNoisyAmRandomVoice = fft(noisyAmRandomVoice);

% Calculate magnitude spectrum for each noisy AM voice
magSpectrumNoisyAmMale =
abs(fftNoisyAmMaleVoice/length(noisyAmMaleVoice));
magSpectrumNoisyAmFemale =
abs(fftNoisyAmFemaleVoice/length(noisyAmFemaleVoice));
magSpectrumNoisyAmRandom =
abs(fftNoisyAmRandomVoice/length(noisyAmRandomVoice));

% Only take the first half of frequencies (due to FFT symmetry)
magSpectrumNoisyAmMale =
magSpectrumNoisyAmMale(1:length(noisyAmMaleVoice)/2+1);
magSpectrumNoisyAmFemale =
magSpectrumNoisyAmFemale(1:length(noisyAmFemaleVoice)/2+1);
magSpectrumNoisyAmRandom =
magSpectrumNoisyAmRandom(1:length(noisyAmRandomVoice)/2+1);
```

```
% Double non-DC components for proper magnitude
magSpectrumNoisyAmMale(2:end-1) = 2 * magSpectrumNoisyAmMale(2:end-
1);
magSpectrumNoisyAmFemale(2:end-1) = 2 *
magSpectrumNoisyAmFemale(2:end-1);
magSpectrumNoisyAmRandom(2:end-1) = 2 *
magSpectrumNoisyAmRandom(2:end-1);

% Frequency vector for plotting
fNoisyAm =
Fs*(0:(length(noisyAmMaleVoice)/2))/length(noisyAmMaleVoice);

% Plot the time-domain and frequency-domain signals
figure;
subplot(3,2,1);
plot(noisyAmMaleVoice);
title('Noisy AM Male Voice (Time Domain)');
xlabel('Samples');
ylabel('Amplitude');

subplot(3,2,2);
plot(magSpectrumNoisyAmMale);
title('Spectrum of Noisy AM Male Voice');
xlabel('Frequency (Hz)');
ylabel('Magnitude');

subplot(3,2,3);
plot(noisyAmFemaleVoice);
title('Noisy AM Female Voice (Time Domain)');
xlabel('Samples');
ylabel('Amplitude');

subplot(3,2,4);
plot(magSpectrumNoisyAmFemale);
title('Spectrum of Noisy AM Female Voice');
xlabel('Frequency (Hz)');
ylabel('Magnitude');

subplot(3,2,5);
plot(noisyAmRandomVoice);
title('Noisy AM Random Voice (Time Domain)');
xlabel('Samples');
ylabel('Amplitude');

subplot(3,2,6);
plot(magSpectrumNoisyAmRandom);
title('Spectrum of Noisy AM Random Voice');
xlabel('Frequency (Hz)');
ylabel('Magnitude');
```

```
% --- Receiver Section ---
%% Demodulate the noisy signals
% Demodulate the noisy AM signals
demodMaleVoice = amdemod(noisyAmMaleVoice, carrierFreqMale, Fs);
demodFemaleVoice = amdemod(noisyAmFemaleVoice, carrierFreqFemale,
Fs);
demodRandomVoice = amdemod(noisyAmRandomVoice, carrierFreqRandom,
Fs);

% Perform FFT on the demodulated voices
fftDemodMaleVoice = fft(demodMaleVoice);
fftDemodFemaleVoice = fft(demodFemaleVoice);
fftDemodRandomVoice = fft(demodRandomVoice);

% Calculate magnitude spectrum for each demodulated voice
magSpectrumDemodMale =
abs(fftDemodMaleVoice/length(demodMaleVoice));
magSpectrumDemodFemale =
abs(fftDemodFemaleVoice/length(demodFemaleVoice));
magSpectrumDemodRandom =
abs(fftDemodRandomVoice/length(demodRandomVoice));

% Only take the first half of frequencies (due to FFT symmetry)
magSpectrumDemodMale =
magSpectrumDemodMale(1:length(demodMaleVoice)/2+1);
magSpectrumDemodFemale =
magSpectrumDemodFemale(1:length(demodFemaleVoice)/2+1);
magSpectrumDemodRandom =
magSpectrumDemodRandom(1:length(demodRandomVoice)/2+1);

% Double non-DC components for proper magnitude
magSpectrumDemodMale(2:end-1) = 2 * magSpectrumDemodMale(2:end-1);
magSpectrumDemodFemale(2:end-1) = 2 * magSpectrumDemodFemale(2:end-
1);
magSpectrumDemodRandom(2:end-1) = 2 * magSpectrumDemodRandom(2:end-
1);

% Frequency vector for plotting
fDemod = Fs*(0:(length(demodMaleVoice)/2))/length(demodMaleVoice);

% Plot the time-domain and frequency-domain signals
figure;
subplot(3,2,1);
plot(demodMaleVoice);
title('Demodulated Male Voice (Time Domain)');
xlabel('Samples');
ylabel('Amplitude');

subplot(3,2,2);
```

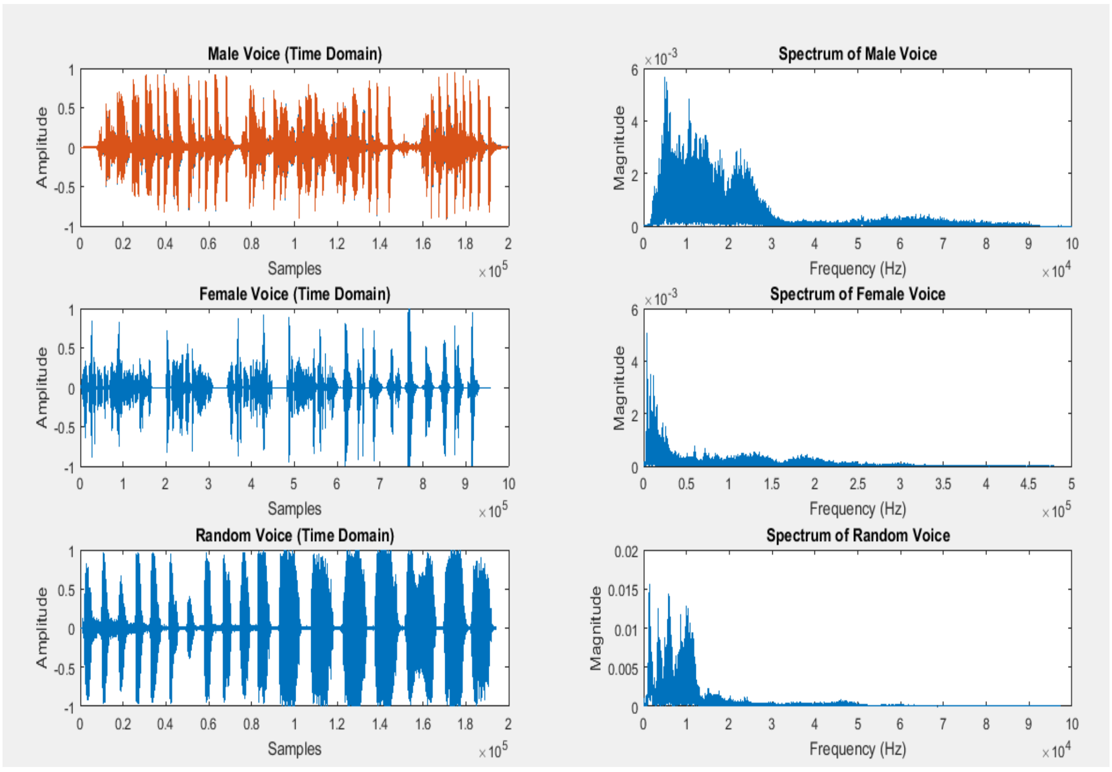
```
plot(magSpectrumDemodMale);
title('Spectrum of Demodulated Male Voice');
xlabel('Frequency (Hz)');
ylabel('Magnitude');

subplot(3,2,3);
plot(demodFemaleVoice);
title('Demodulated Female Voice (Time Domain)');
xlabel('Samples');
ylabel('Amplitude');

subplot(3,2,4);
plot(magSpectrumDemodFemale);
title('Spectrum of Demodulated Female Voice');
xlabel('Frequency (Hz)');
ylabel('Magnitude');

subplot(3,2,5);
plot(demodRandomVoice);
title('Demodulated Random Voice (Time Domain)');
xlabel('Samples');
ylabel('Amplitude');

subplot(3,2,6);
plot(magSpectrumDemodRandom);
title('Spectrum of Demodulated Random Voice');
xlabel('Frequency (Hz)');
ylabel('Magnitude');

%% Low-Pass Filtering on Demodulated Signals
% Second LPF (same parameters as before)
[b2, a2] = butter(filterOrder, cutoffFrequency/(Fs/2), 'low');

% Apply LPF to demodulated signals
lpfDemodMaleVoice = filter(b2, a2, demodMaleVoice);
lpfDemodFemaleVoice = filter(b2, a2, demodFemaleVoice);
lpfDemodRandomVoice = filter(b2, a2, demodRandomVoice);

% Perform FFT on the filtered, demodulated voices
fftLpfDemodMaleVoice = fft(lpfDemodMaleVoice);
fftLpfDemodFemaleVoice = fft(lpfDemodFemaleVoice);
fftLpfDemodRandomVoice = fft(lpfDemodRandomVoice);

% Calculate magnitude spectrum for each filtered, demodulated voice
magSpectrumLpfDemodMale =
abs(fftLpfDemodMaleVoice/length(lpfDemodMaleVoice));
magSpectrumLpfDemodFemale =
abs(fftLpfDemodFemaleVoice/length(lpfDemodFemaleVoice));
magSpectrumLpfDemodRandom =
abs(fftLpfDemodRandomVoice/length(lpfDemodRandomVoice));
```

```
% Only take the first half of frequencies (due to FFT symmetry)
magSpectrumLpfDemodMale =
magSpectrumLpfDemodMale(1:length(lpfDemodMaleVoice)/2+1);
magSpectrumLpfDemodFemale =
magSpectrumLpfDemodFemale(1:length(lpfDemodFemaleVoice)/2+1);
magSpectrumLpfDemodRandom =
magSpectrumLpfDemodRandom(1:length(lpfDemodRandomVoice)/2+1);

% Double non-DC components for proper magnitude
magSpectrumLpfDemodMale(2:end-1) = 2 *
magSpectrumLpfDemodMale(2:end-1);
magSpectrumLpfDemodFemale(2:end-1) = 2 *
magSpectrumLpfDemodFemale(2:end-1);
magSpectrumLpfDemodRandom(2:end-1) = 2 *
magSpectrumLpfDemodRandom(2:end-1);

% Frequency vector for plotting
fLpfDemod =
Fs*(0:(length(lpfDemodMaleVoice)/2))/length(lpfDemodMaleVoice);

% Plot the time-domain and frequency-domain signals
figure;
subplot(3,2,1);
plot(lpfDemodMaleVoice);
title('Filtered Demodulated Male Voice (Time Domain)');
xlabel('Samples');
ylabel('Amplitude');

subplot(3,2,2);
plot(magSpectrumLpfDemodMale);
title('Spectrum of Filtered Demodulated Male Voice');
xlabel('Frequency (Hz)');
ylabel('Magnitude');

subplot(3,2,3);
plot(lpfDemodFemaleVoice);
title('Filtered Demodulated Female Voice (Time Domain)');
xlabel('Samples');
ylabel('Amplitude');

subplot(3,2,4);
plot(magSpectrumLpfDemodFemale);
title('Spectrum of Filtered Demodulated Female Voice');
xlabel('Frequency (Hz)');
ylabel('Magnitude');

subplot(3,2,5);
plot(lpfDemodRandomVoice);
title('Filtered Demodulated Random Voice (Time Domain)');
```

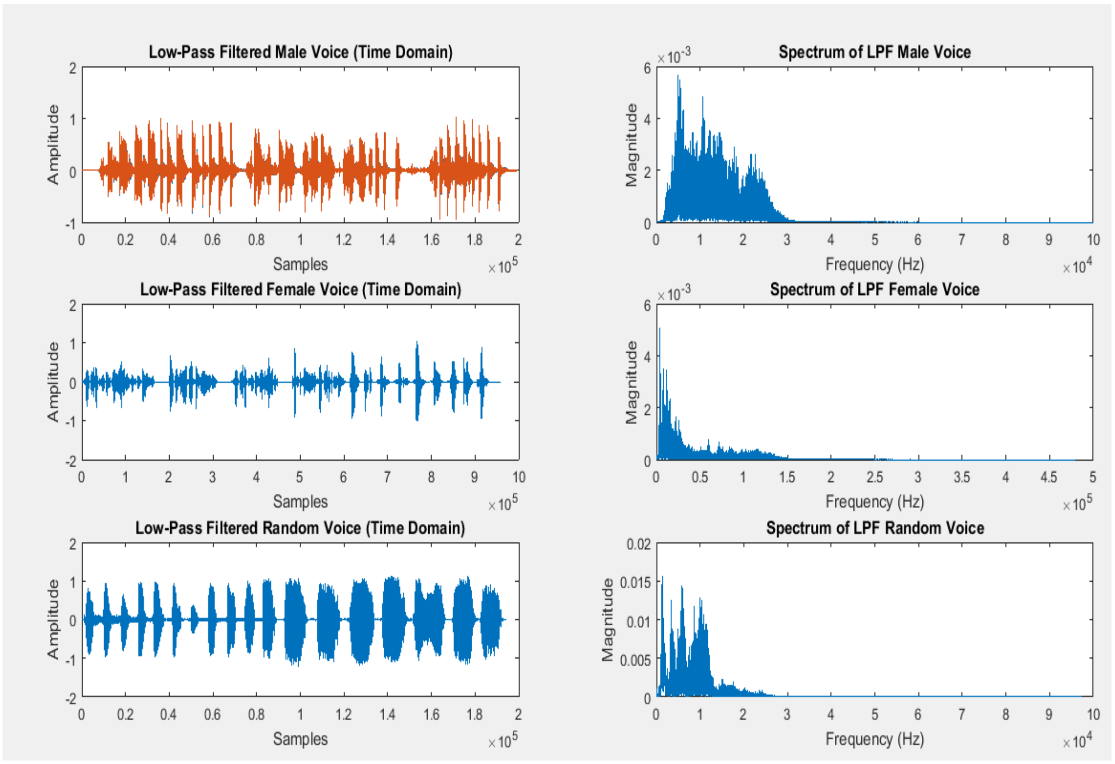
```
xlabel('Samples');
ylabel('Amplitude');

subplot(3,2,6);
plot(magSpectrumLpfDemodRandom);
title('Spectrum of Filtered Demodulated Random Voice');
xlabel('Frequency (Hz)');
ylabel('Magnitude');

%% Play the filtered, demodulated voices
sound(lpfDemodMaleVoice, Fs);
pause(length(lpfDemodMaleVoice)/Fs + 1);

sound(lpfDemodFemaleVoice, Fs);
pause(length(lpfDemodFemaleVoice)/Fs + 1);

sound(lpfDemodRandomVoice, Fs);
```

Output:

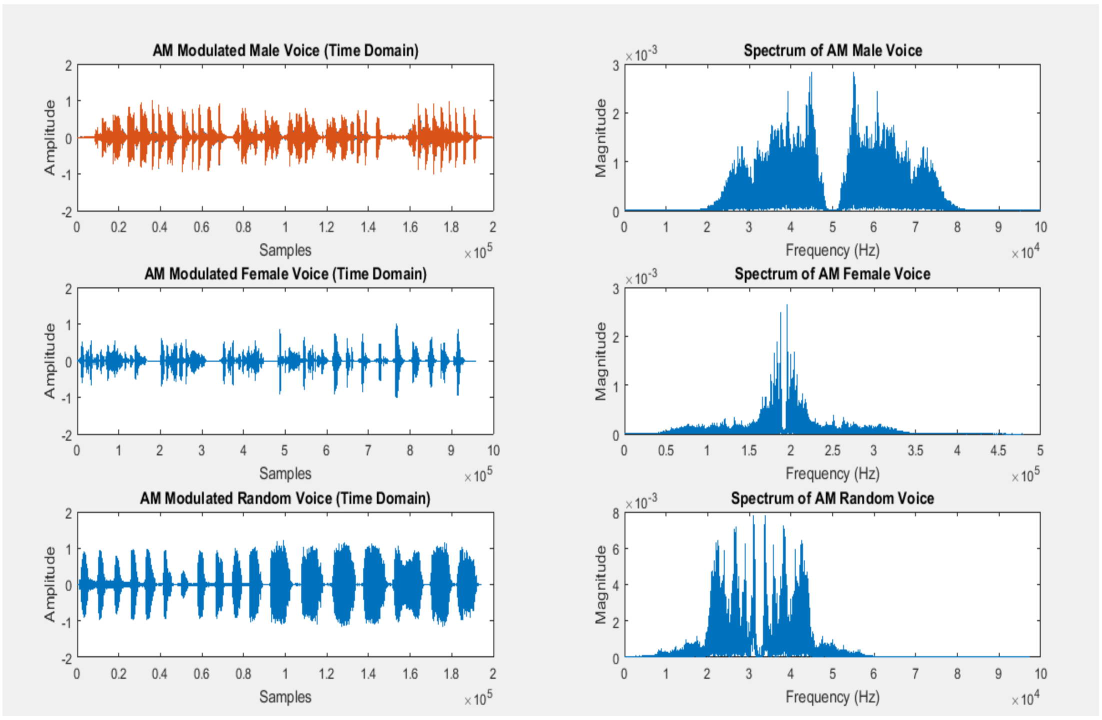
Original Voices:



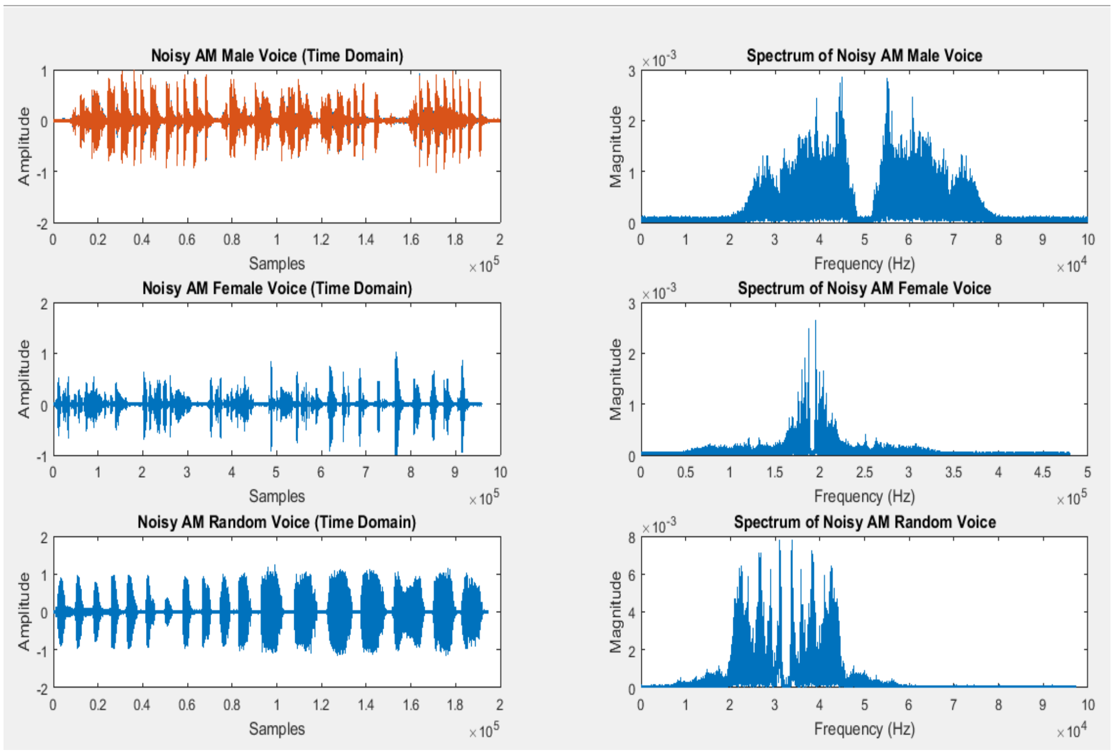
Passing original voices in low pass filter



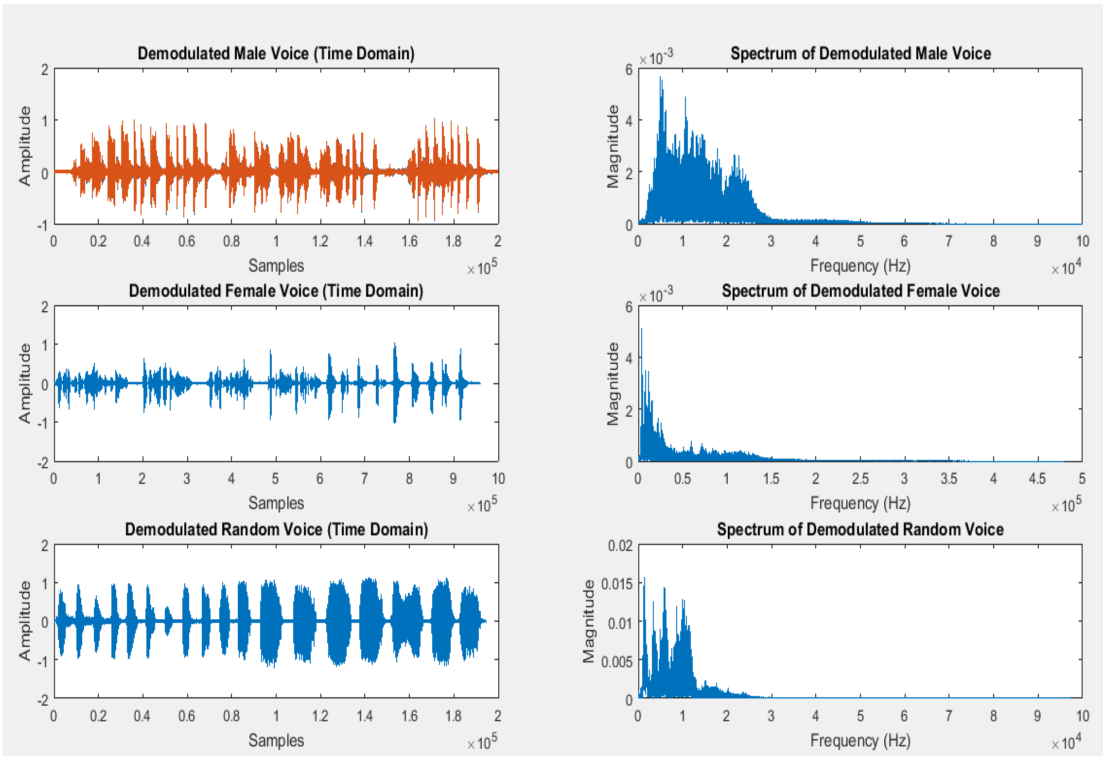
Modulating the voices using different carriers



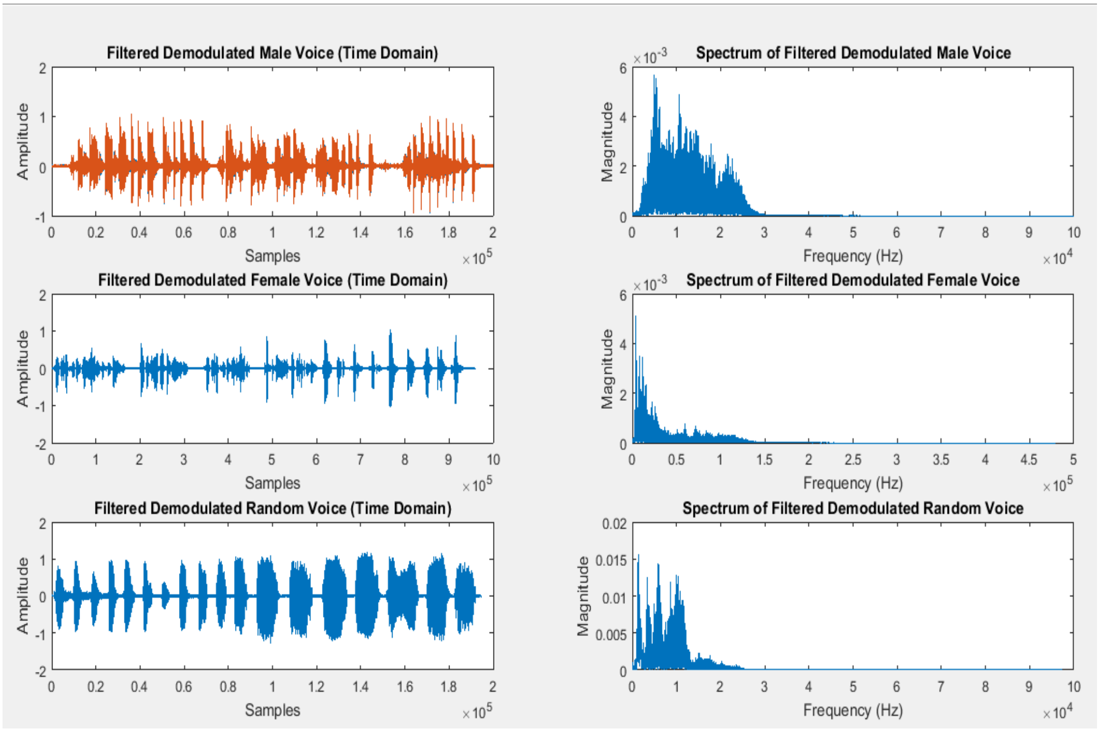
Adding noise from the channel in voices



Demodulating the noisy signal at receiver end



At last, passing the voices through low pass filter to recover the signal at receiver end.



Reference:

To view my codes, please refer to my GitHub account.

Conclusion:

In conclusion, I have created voice signals. And then modulated it with different carriers of different frequencies. Then I have added some noise to each signal which resembles of channel noise. After that I have demodulated the signal and recover the original signal after passing it through LPF.

The End.