|  |
| --- |
| EECE26 |
| Communication Project Super-Heterodyne Receiver |
| Cairo University Faculty of Engineering |

|  |
| --- |
| Youssef Samir Zidan  9220988  Sec 4 |

Contents

[1. Discussion 0](#_Toc186122156)

[a. Discuss how you designed each of the blocks of the system in Figure 1. 0](#_Toc186122157)

[b. Answer the following: 3](#_Toc186122158)

[1) In two or three sentences, discuss the role of the RF, the IF and the baseband detector. Indicate why we need the IF stage? 3](#_Toc186122159)

[2) Suppose you want to demodulate the first station (i.e. at 𝜔0), plot the spectrum of the outputs of the RF, the IF and the baseband stages. Hint: use the ‘fft’ command in MATLAB. You may also find the ‘fftshift’ command useful. 3](#_Toc186122160)

[3) Use the command ‘sound’ on the demodulated signal and check whether you can successfully listen to the radio station. Please comment about this step in your report 4](#_Toc186122161)

[4) Add “noise” to your signal and then play “sound” the signal. What is the effect of the noise? 4](#_Toc186122162)

[5) Repeat parts 2 and 3 but after removing the RF BPF. That is, the RF stage does not exist, what would happen if you try to demodulate the station at 𝜔0? 4](#_Toc186122163)

[6) What happens (in terms of the spectrum and the sound quality) if the receiver oscillator has frequency offset by 0.2 KHz and 1.2 KHz? 6](#_Toc186122164)

[1.1 Filter order 7](#_Toc186122165)

[2 CODE appendix 8](#_Toc186122166)

[2.1 Samples (First to run) 8](#_Toc186122167)

[2.2 Modulation (second to run) 9](#_Toc186122168)

[2.3 Demodulation 10](#_Toc186122169)

[2.4 Whole code 12](#_Toc186122170)

[Fig. ‎1‑1 TIme and Freq domain of all sent signals (modulated) 0](#_Toc186122133)

[Fig. ‎1‑2 RF and IF Stages time and freq domain signal 3](#_Toc186122134)

[Fig. ‎1‑3 baseband signal time and freq 4](#_Toc186122135)

[Fig. ‎1‑4 after adding noise to received signal 5](#_Toc186122136)

[Fig. ‎1‑5 baseband after adding noise 5](#_Toc186122137)

[Fig. ‎1‑6 RF And IF Stages after noise added 6](#_Toc186122138)

[Fig. ‎1‑7 effect of offset on baseband signal (0.2kHz) 6](#_Toc186122139)

[Fig. ‎1‑8 effect of offset on baseband signal (1.2kHz) 7](#_Toc186122140)

# Discussion

## Discuss how you designed each of the blocks of the system in Figure 1.



AM

Modulator



Baseband Detection

IF Stage

𝜔𝐼𝐹

Oscillator

𝜔𝐶 + 𝜔𝐼𝐹

Mixer

RF Stage

𝜔𝑐

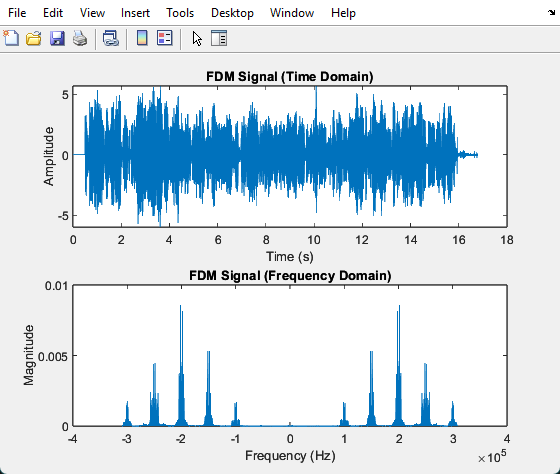
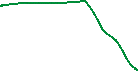
* The first stage is that I extracted the data from the signals getting important parameters like sampling frequency and BWs
* Then it came time for modulating which was as straight forward as just multiplying the signal by .\*cos(2pi\*Fc\*t) in an element wise multiplication
* 

Fig. ‑ TIme and Freq domain of all sent signals (modulated)

* Then the another function takes the whole receiver part
* At RF stage the user selects the required signal then a BPF is placed at the intended frequency to minimize the image interference
* Then a mixer simply multiplies with. \*Cos(2pi\*(Fc+IF) \*t) to get the band in intermediate frequency which could be an input directly to an ADC (but more BW overhead for ADC)
* Then we put another BPF filter in IF stage to get rid of any interference and other signals
* For the last stage we modulate again with IF only to get the signal to the baseband, then finally we put a LPF to get rid of all noise and interference and hear the signal clearly



## Answer the following:

### In two or three sentences, discuss the role of the RF, the IF and the baseband detector. Indicate why we need the IF stage?

* RF detector selects the desired station from multiple signals when input signal is at Wc

IF detector converts any signal to a fixed intermediate frequency regardless of the original carrier frequency. which simplifies the hardware and improves filtering performance

* Baseband detector is used to demodulate the signal from IF to the original Baseband

### Suppose you want to demodulate the first station (i.e. at 𝜔0), plot the spectrum of the outputs of the RF, the IF and the baseband stages. Hint: use the ‘fft’ command in MATLAB. You may also find the ‘fftshift’ command useful.

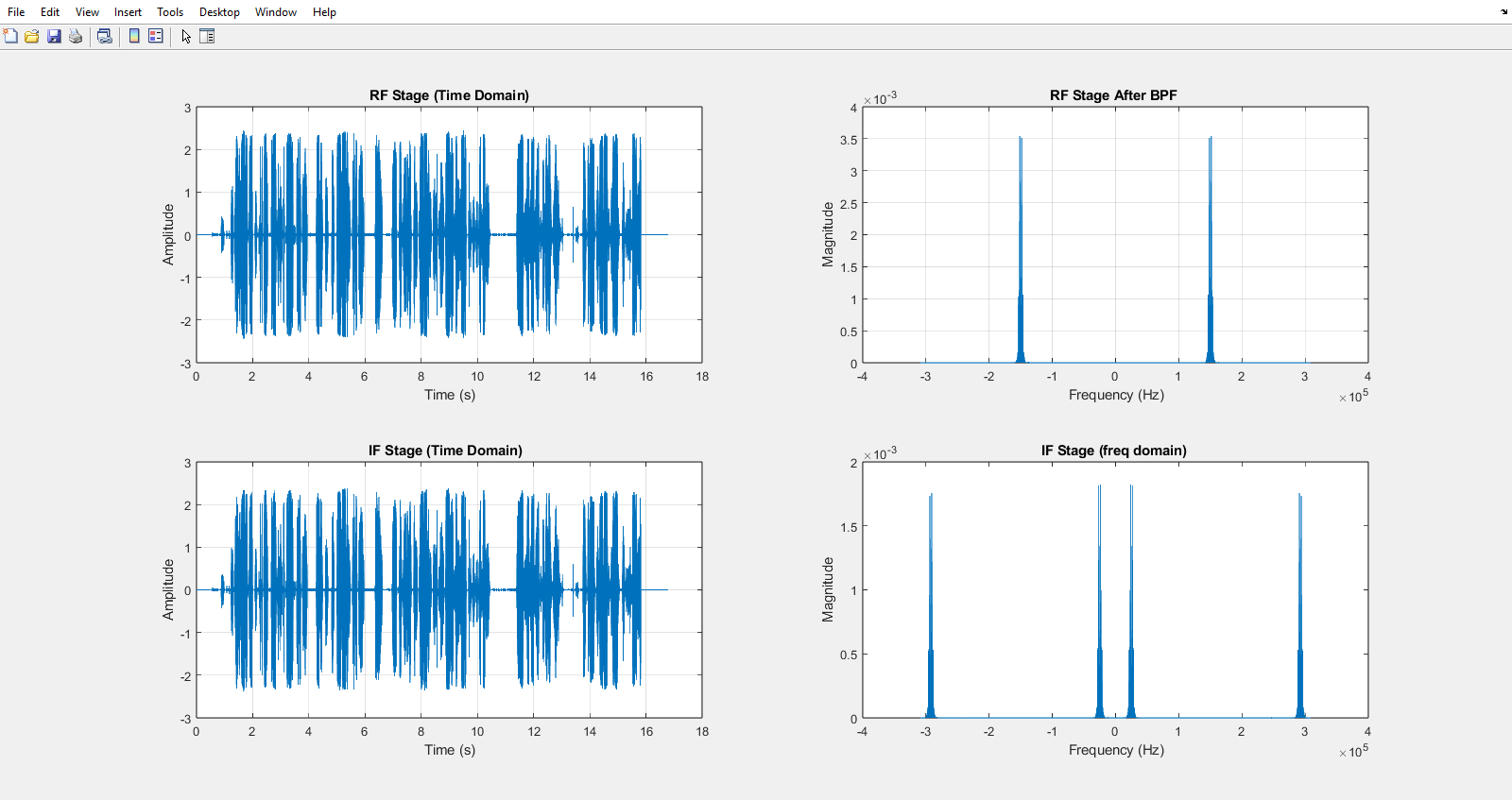


Fig. ‑ RF and IF Stages time and freq domain signal

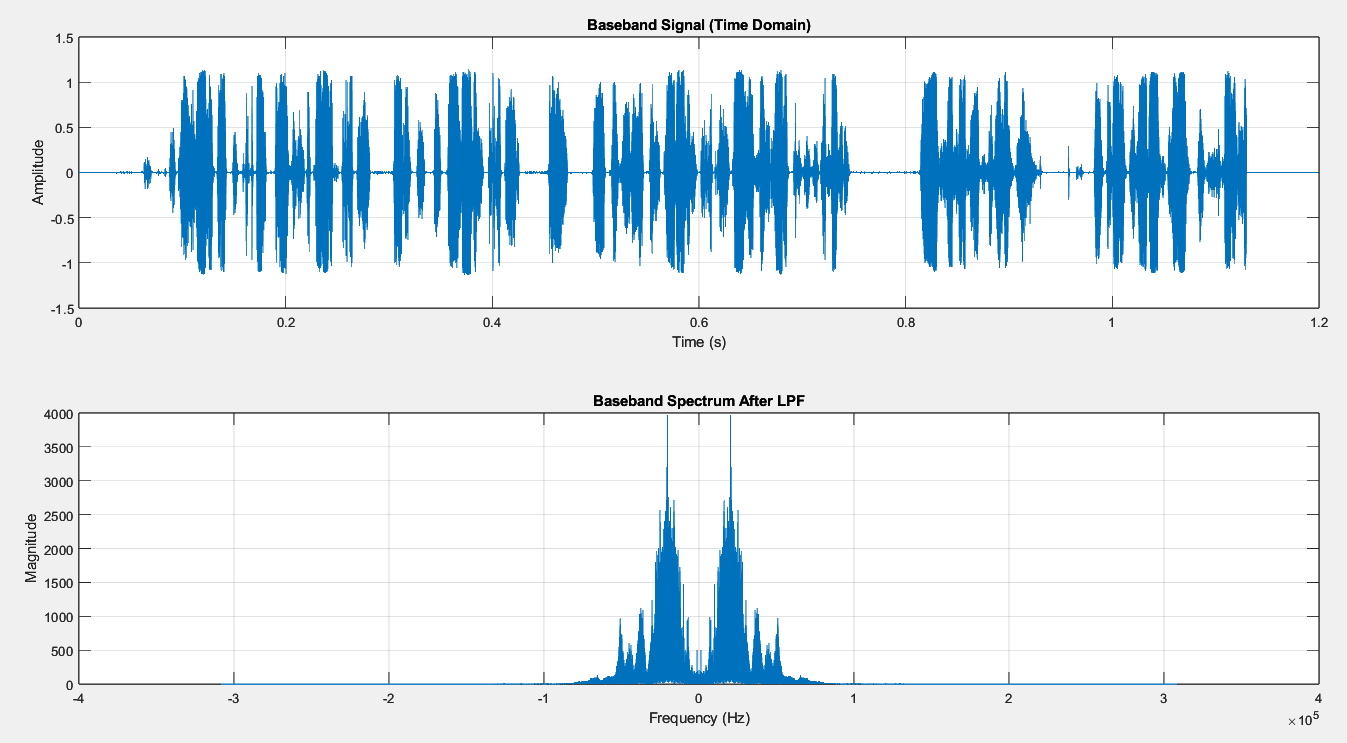


Fig. ‑ baseband signal time and freq

### Use the command ‘sound’ on the demodulated signal and check whether you can successfully listen to the radio station. Please comment about this step in your report

* Yes, I can hear it clearly and I had to down sample again to hear it

### Add “noise” to your signal and then play “sound” the signal. What is the effect of the noise?

* Using received Signal = awgn(sentsignal,10);
* I heard a white noise with the signal وشوشة

### Repeat parts 2 and 3 but after removing the RF BPF. That is, the RF stage does not exist, what would happen if you try to demodulate the station at 𝜔0?

* I only heard the image signal at 150khz completely overlapping and shadowing the intended signal
* And it’s visible in graph 2 IF stage that most of the interfering signals are still there
* I also would like to mention the I didn’t hear any image noise from signal no.5 as there’s no signal at “2\*IF” from it

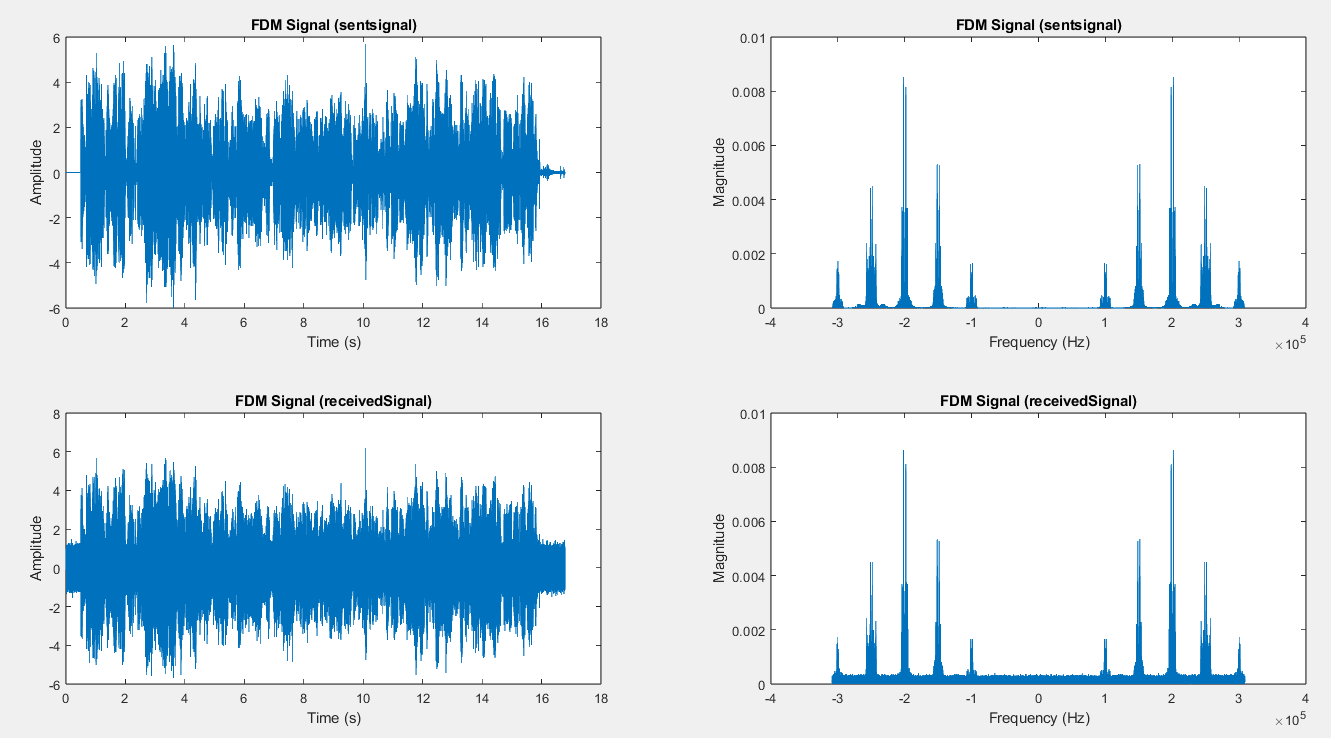


Fig. ‑ after adding noise to received signal

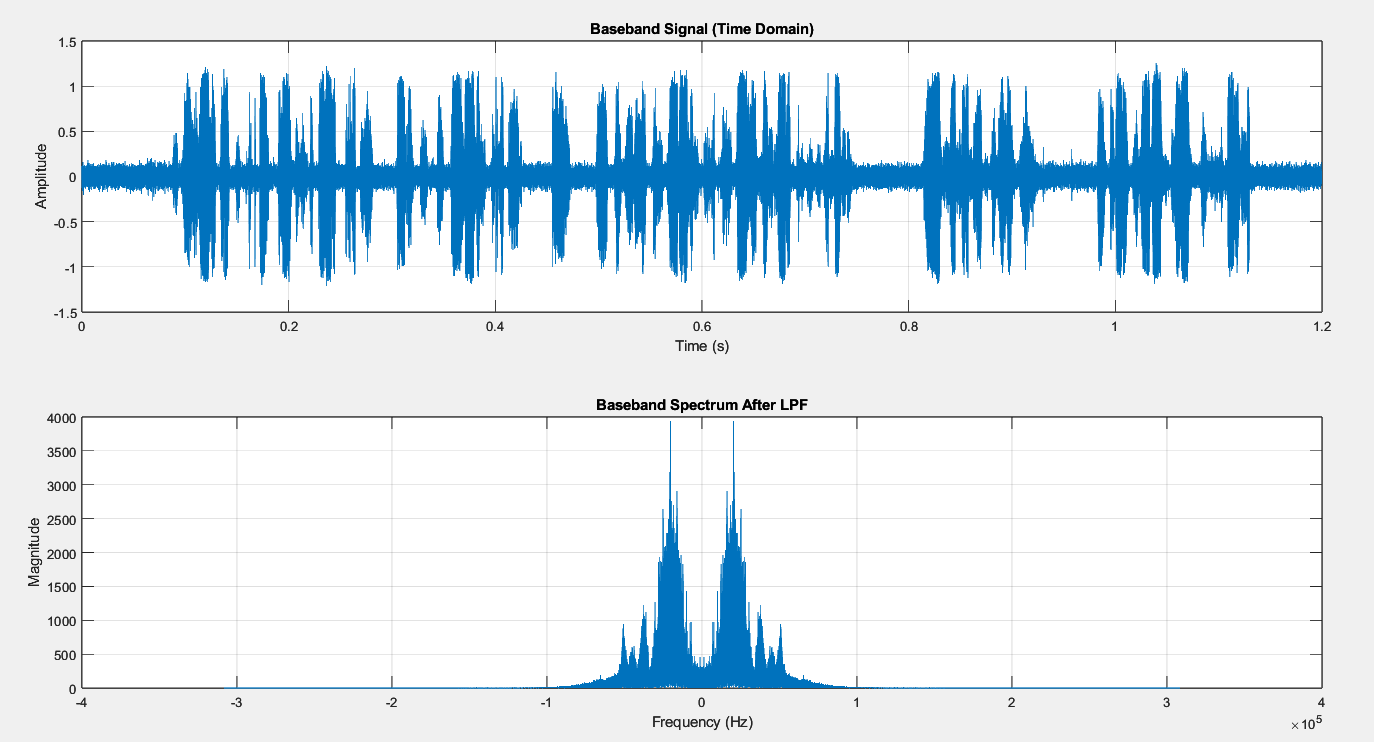


Fig. ‑ baseband after adding noise

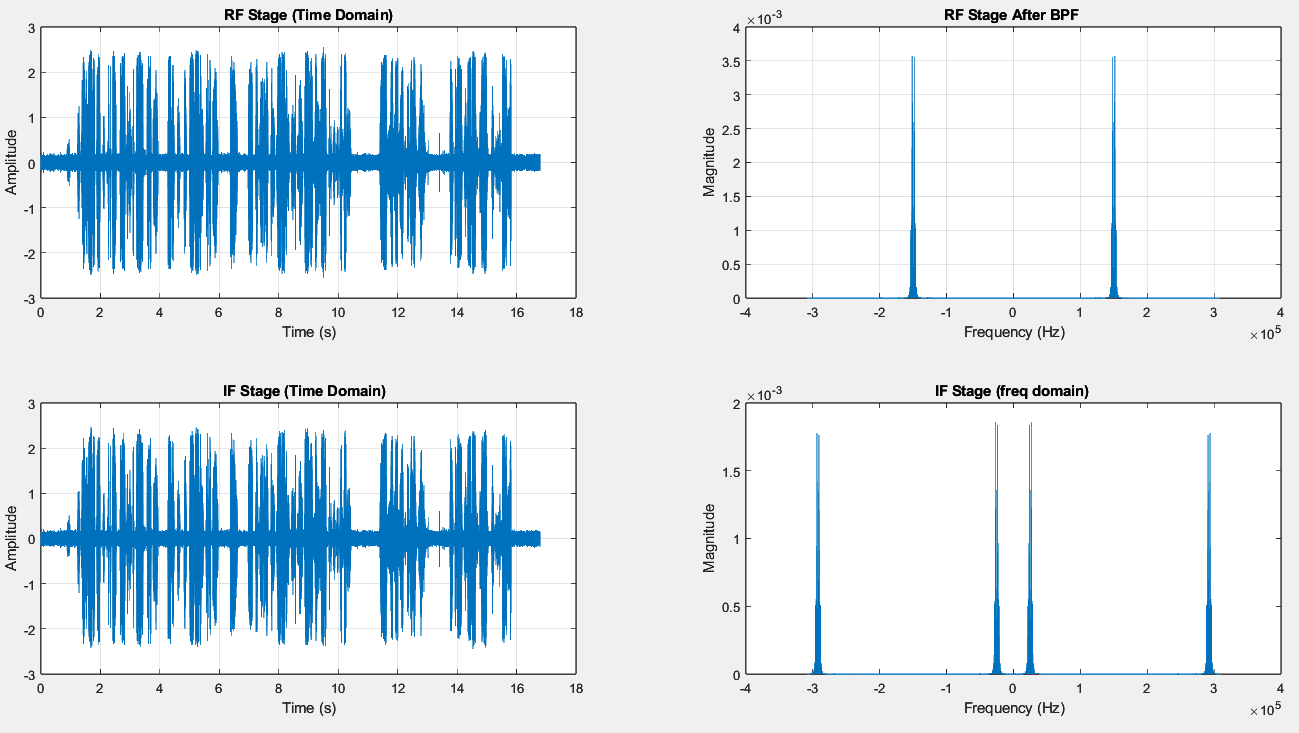


Fig. ‑ RF And IF Stages after noise added

### What happens (in terms of the spectrum and the sound quality) if the receiver oscillator has frequency offset by 0.2 KHz and 1.2 KHz?

* carrier\_LO = cos (2 \* pi \* (fc\_mod + IF + offset) \* t); % Local oscillator signal
* at 0.2khz I could barely hear the signal with very annoying sound effect

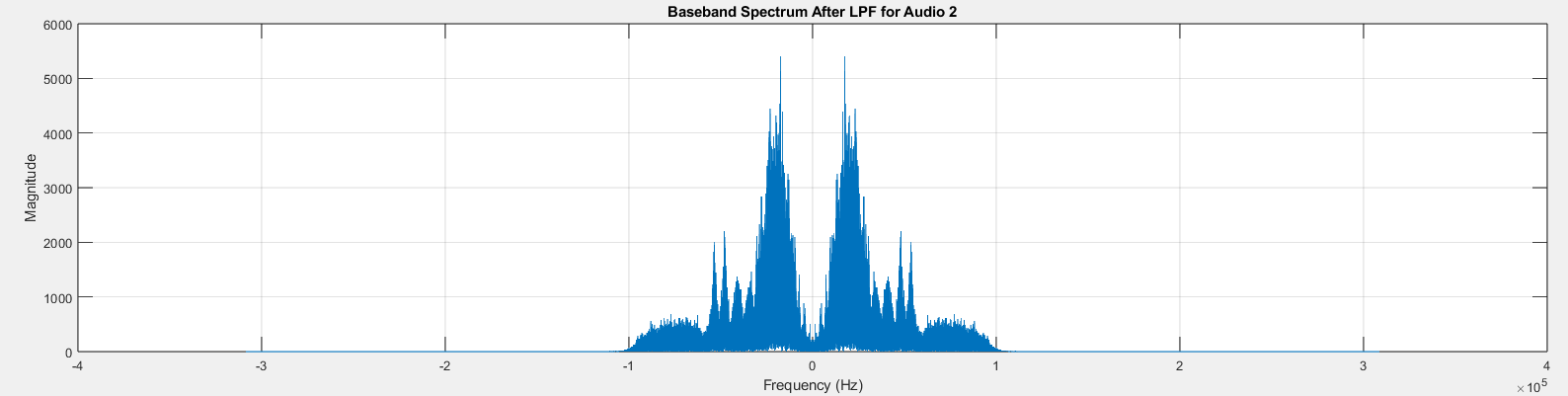


Fig. ‑ effect of offset on baseband signal (0.2kHz)

* at 1.2khz I couldn’t hear the signal at all

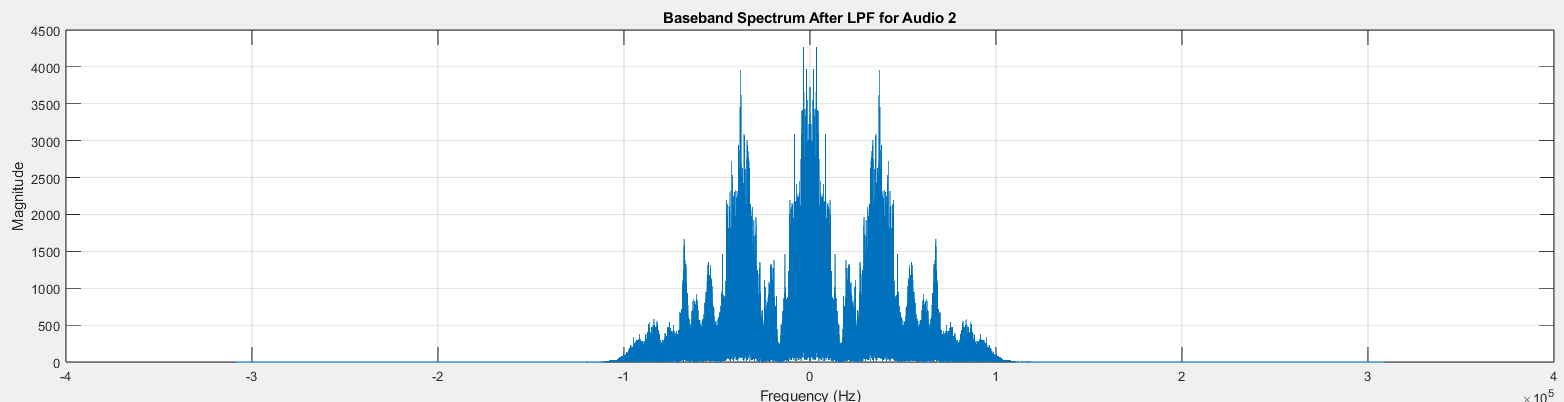


Fig. ‑ effect of offset on baseband signal (1.2kHz)

## Filter order

# CODE appendix

## Samples (First to run)

%% Section 1: Load Data

addpath 'Audio Signals'\

[c1, fs1] = audioread("Short\_BBCArabic2.wav");

[c2, fs2] = audioread("Short\_FM9090.wav");

[c3, fs3] = audioread("Short\_QuranPalestine.wav");

[c4, fs4] = audioread("Short\_RussianVoice.wav");

[c5, fs5] = audioread("Short\_SkyNewsArabia.wav");

audioSignals = {c1, c2, c3, c4, c5};

samplingRates = [fs1, fs2, fs3, fs4, fs5];

Left\_channel = cell(1, length(audioSignals));

Right\_channel = cell(1, length(audioSignals));

monoSignals = cell(1, 5);

paddedSignals = cell(1, 5);

maxLength = 0;

% Extract left, right, and mono channels; calculate max signal length

for i = 1:length(audioSignals)

Left\_channel{i} = audioSignals{i}(:, 1);

Right\_channel{i} = audioSignals{i}(:, 2);

monoSignals{i} = sum(audioSignals{i}, 2); % Combine left and right channels

maxLength = max(maxLength, length(monoSignals{i}));

end

% Pad signals to match the maximum length

for i = 1:length(monoSignals)

paddedSignals{i} = [monoSignals{i}; zeros(maxLength - length(monoSignals{i}), 1)];

end

%% Section 2: Plot and Extract Data

BWs = zeros(1, 5); % Initialize array for bandwidths

% Plot frequency spectra and calculate bandwidths

for i = 1:length(audioSignals)

x = audioSignals{i};

fs = samplingRates(i);

% FFT for frequency spectrum

N = length(x);

X = fft(x, N);

f = (-N/2:N/2-1) \* fs / N;

% Calculate bandwidth

BWs(i) = obw(monoSignals{i}, fs);

% Plot frequency spectrum

figure;

plot(f, abs(fftshift(X)) / N);

title(['Frequency Spectrum of Signal ', num2str(i)]);

xlabel('Frequency (Hz)');

ylabel('Magnitude');

grid on;

end

% Convert bandwidths to Hz and kHz

BWs\_Hz = BWs;

BWs\_kHz = BWs\_Hz / 1000;

clearvars -except BWs paddedSignals audioSignals

## Modulation (second to run)

% Constants

fc\_mod = zeros(1, 5); % Carrier frequencies

modulated\_signal = cell(1, 5); % Modulated signals

sentsignal = []; % Combined FDM signal

interp\_factor = 14; % Interpolation factor

fs = 44100; % Sampling frequency

fs\_interpolated = interp\_factor \* fs; % Interpolated sampling frequency

% Modulation process

for i = 1:length(audioSignals)

% Retrieve the padded mono signal

x = paddedSignals{i};

% Take the first portion of the signal

x = x(1:floor(length(x) / 1));

% Carrier frequency (in Hz)

fc\_mod(i) = 100e3 + (i - 1) \* 50e3;

% Interpolate the signal

x\_inter = interp(x, interp\_factor);

% Time vector for the interpolated signal

t = (0:1/fs\_interpolated:(length(x\_inter) - 1) / fs\_interpolated)';

% Generate the carrier signal and perform modulation

Mod\_carrier = cos(2 \* pi \* fc\_mod(i) \* t);

modulated\_signal{i} = x\_inter .\* Mod\_carrier;

end

% Combine all modulated signals into the FDM signal

for i = 1:length(modulated\_signal)

if isempty(sentsignal)

sentsignal = modulated\_signal{i};

else

sentsignal = sentsignal + modulated\_signal{i};

end

end

% Plot the combined FDM signal in the time domain

figure;

subplot(2, 1, 1);

t\_total = (0:length(sentsignal) - 1) / fs\_interpolated; % Time vector for FDM signal

plot(t\_total, sentsignal);

title('FDM Signal (Time Domain)');

xlabel('Time (s)');

ylabel('Amplitude');

% Plot the combined FDM signal in the frequency domain

subplot(2, 1, 2);

N = length(sentsignal);

f = (-N/2:N/2-1) \* fs\_interpolated / N; % Frequency axis

spectrum\_fdm = fftshift(fft(sentsignal, N)); % Compute and shift the FFT

plot(f, abs(spectrum\_fdm) / N);

title('FDM Signal (Frequency Domain)');

xlabel('Frequency (Hz)');

ylabel('Magnitude');

% Play the FDM signal

%sound(sentsignal, fs\_interpolated);

clearvars -except BWs sentsignal interp\_factor fs

## Demodulation

% Constants

offset = 0;

IF = 25e3; % Intermediate frequency (25 kHz)

fs\_interpolated = interp\_factor \* fs; % Interpolated sampling rate

receivedSignal = sentsignal; % Received FDM signal

Rf\_enable = 0; % Enable/Disable RF filter

% Relaxed Filter Constraints

A\_pass = 2;

A\_stop1 = 20;

A\_stop2 = 20;

fprintf("\nSignal Processing Initialized.\n");

while true

% User input for channel selection

fprintf("\nChoose one of these channels (Enter 0 to Exit):\n");

fprintf("1. Short\_BBCArabic2\n");

fprintf("2. Short\_FM9090\n");

fprintf("3. Short\_QuranPalestine\n");

fprintf("4. Short\_RussianVoice\n");

fprintf("5. Short\_SkyNewsArabia\n");

i = input("Choose: ");

% Exit condition

if i == 0

fprintf("Exiting program...\n");

break;

elseif i < 1 || i > 5

fprintf("Invalid input! Please choose a number between 1 and 5, or 0 to exit.\n");

continue;

end

% Carrier frequency and bandwidth for the selected channel

fc\_mod = 100e3 + (i - 1) \* 50e3; % Carrier frequency for the selected channel

bw = BWs(i); % Bandwidth (fixed for all channels)

% Define proper increasing frequencies and cap at Nyquist

F\_stop1 = fc\_mod - bw ; % Stopband start

F\_pass1 = fc\_mod - bw/2; % Passband start

F\_pass2 = fc\_mod + bw/2; % Passband end

F\_stop2 = fc\_mod + bw ; % Stopband end

% RF Bandpass Filter

if Rf\_enable == 1

if F\_stop2 > fs\_interpolated / 2

error("Filter frequencies exceed Nyquist limit. Increase fs\_interpolated.");

end

rf\_bandpass\_filter = designfilt('bandpassiir', ...

'StopbandFrequency1', F\_stop1, 'PassbandFrequency1', F\_pass1, ...

'PassbandFrequency2', F\_pass2, 'StopbandFrequency2', F\_stop2, ...

'StopbandAttenuation1', A\_stop1, 'PassbandRipple', A\_pass, ...

'StopbandAttenuation2', A\_stop2, 'SampleRate', fs\_interpolated);

fprintf("RF Filter Order: %d\n", filtord(rf\_bandpass\_filter));

filtered\_signal = filter(rf\_bandpass\_filter, receivedSignal);

else

filtered\_signal = receivedSignal;

end

% Plot RF Stage Spectrum

figure(1); clf;

rf\_spectrum = fftshift(fft(filtered\_signal, 2^nextpow2(length(filtered\_signal))));

N = length(rf\_spectrum);

f = (-N/2:N/2-1) \* fs\_interpolated / N;

subplot(2, 1, 1);

plot(f, abs(rf\_spectrum));

title("RF Stage After BPF for Audio " + num2str(i));

xlabel("Frequency (Hz)"); ylabel("Magnitude"); grid on;

% Step 2: Mix with Local Oscillator to Shift to IF

t = (1:length(filtered\_signal))' / fs\_interpolated; % Time vector

carrier\_LO = cos(2 \* pi \* (fc\_mod + IF) \* t); % Local oscillator signal

mixed\_signal\_IF = filtered\_signal .\* carrier\_LO; % Mixer output

% Plot IF Stage Spectrum

IF\_spectrum = fftshift(fft(mixed\_signal\_IF, 2^nextpow2(length(mixed\_signal\_IF))));

subplot(2, 1, 2);

plot(f, abs(IF\_spectrum));

title("IF Stage for Audio " + num2str(i));

xlabel("Frequency (Hz)"); ylabel("Magnitude"); grid on;

% Step 3: Bandpass Filter to Isolate IF

F\_stop1\_IF = max(0, IF - bw - 5e3); % Stopband start

F\_pass1\_IF = IF - bw; % Passband start

F\_pass2\_IF = IF + bw; % Passband end

F\_stop2\_IF = min(fs\_interpolated / 2 - 1, IF + bw + 5e3); % Stopband end

if\_bandpass\_filter = designfilt('bandpassiir', ...

'StopbandFrequency1', F\_stop1\_IF, 'PassbandFrequency1', F\_pass1\_IF, ...

'PassbandFrequency2', F\_pass2\_IF, 'StopbandFrequency2', F\_stop2\_IF, ...

'StopbandAttenuation1', A\_stop1, 'PassbandRipple', A\_pass, ...

'StopbandAttenuation2', A\_stop2, 'SampleRate', fs\_interpolated);

fprintf("IF Filter Order: %d\n", filtord(if\_bandpass\_filter));

filtered\_signal\_IF = filter(if\_bandpass\_filter, mixed\_signal\_IF);

% Step 4: Demodulate IF Signal

carrier\_IF = cos(2 \* pi \* IF \* t); % IF carrier signal

baseband\_signal = filtered\_signal\_IF .\* carrier\_IF; % Demodulation

% Step 5: Apply Low-Pass Filter to Recover Baseband Signal

F\_pass = bw; F\_stop = bw + 5e3;

lowpass\_filter = designfilt('lowpassiir', ...

'PassbandFrequency', F\_pass, 'StopbandFrequency', F\_stop, ...

'PassbandRipple', A\_pass, 'StopbandAttenuation', A\_stop1, ...

'SampleRate', fs\_interpolated);

Base\_Band\_received\_signal\_LPF = filter(lowpass\_filter, baseband\_signal);

% Resample and Playback the Demodulated Signal

Base\_Band\_received\_signal\_LPF = 4 \* resample(Base\_Band\_received\_signal\_LPF, 1, interp\_factor);

sound(Base\_Band\_received\_signal\_LPF, fs);

% Plot Baseband Spectrum After LPF

figure(2); clf;

baseband\_spectrum = fftshift(fft(Base\_Band\_received\_signal\_LPF, 2^nextpow2(length(Base\_Band\_received\_signal\_LPF))));

N = length(baseband\_spectrum);

f = (-N/2:N/2-1) \* fs\_interpolated / N;

plot(f, abs(baseband\_spectrum));

title("Baseband Spectrum After LPF for Audio " + num2str(i));

xlabel("Frequency (Hz)"); ylabel("Magnitude"); grid on;

fprintf("Audio %d processed successfully.\n", i);

end

fprintf("Signal Processing Completed.\n");

## Whole code

%% Section 1: Load Data

addpath 'Audio Signals'\

[c1, fs1] = audioread("Short\_BBCArabic2.wav");

[c2, fs2] = audioread("Short\_FM9090.wav");

[c3, fs3] = audioread("Short\_QuranPalestine.wav");

[c4, fs4] = audioread("Short\_RussianVoice.wav");

[c5, fs5] = audioread("Short\_SkyNewsArabia.wav");

audioSignals = {c1, c2, c3, c4, c5};

samplingRates = [fs1, fs2, fs3, fs4, fs5];

Left\_channel = cell(1, length(audioSignals));

Right\_channel = cell(1, length(audioSignals));

monoSignals = cell(1, 5);

paddedSignals = cell(1, 5);

maxLength = 0;

% Extract left, right, and mono channels; calculate max signal length

for i = 1:length(audioSignals)

Left\_channel{i} = audioSignals{i}(:, 1);

Right\_channel{i} = audioSignals{i}(:, 2);

monoSignals{i} = sum(audioSignals{i}, 2); % Combine left and right channels

maxLength = max(maxLength, length(monoSignals{i}));

end

% Pad signals to match the maximum length

for i = 1:length(monoSignals)

paddedSignals{i} = [monoSignals{i}; zeros(maxLength - length(monoSignals{i}), 1)];

end

% Section 2: Plot and Extract Data

BWs = zeros(1, 5); % Initialize array for bandwidths

% Plot frequency spectra and calculate bandwidths

for i = 1:length(audioSignals)

x = audioSignals{i};

fs = samplingRates(i);

% FFT for frequency spectrum

N = length(x);

X = fft(x, N);

f = (-N/2:N/2-1) \* fs / N;

% Calculate bandwidth

BWs(i) = obw(monoSignals{i}, fs);

end

% Convert bandwidths to Hz and kHz

BWs\_Hz = BWs;

BWs\_kHz = BWs\_Hz / 1000;

clearvars -except BWs paddedSignals audioSignals samplingRates

%% section2: Modulation

% Constants

fc\_mod = zeros(1, 5); % Carrier frequencies

modulated\_signal = cell(1, 5); % Modulated signals

sentsignal = []; % Combined FDM signal

interp\_factor = 14; % Interpolation factor

fs = 44100; % Sampling frequency

fs\_interpolated = interp\_factor \* fs; % Interpolated sampling frequency

% Modulation process

for i = 1:length(audioSignals)

% Retrieve the padded mono signal

x = paddedSignals{i};

% Take the first portion of the signal

x = x(1:floor(length(x) / 1));

% Carrier frequency (in Hz)

fc\_mod(i) = 100e3 + (i - 1) \* 50e3;

% Interpolate the signal

x\_inter = interp(x, interp\_factor);

% Time vector for the interpolated signal

t = (0:1/fs\_interpolated:(length(x\_inter) - 1) / fs\_interpolated)';

% Generate the carrier signal and perform modulation

Mod\_carrier = cos(2 \* pi \* fc\_mod(i) \* t);

modulated\_signal{i} = x\_inter .\* Mod\_carrier;

end

% Combine all modulated signals into the FDM signal

for i = 1:length(modulated\_signal)

if isempty(sentsignal)

sentsignal = modulated\_signal{i};

else

sentsignal = sentsignal + modulated\_signal{i};

end

end

% Subplot 1: Time domain of sentsignal

subplot(2, 2, 1);

t\_total = (0:length(sentsignal) - 1) / fs\_interpolated;

plot(t\_total, sentsignal);

title('FDM Signal (sentsignal)');

xlabel('Time (s)');

ylabel('Amplitude');

% Subplot 2: Frequency domain of sentsignal

subplot(2, 2, 2);

N = length(sentsignal);

f = (-N/2:N/2-1) \* fs\_interpolated / N;

spectrum\_fdm = fftshift(fft(sentsignal, N));

plot(f, abs(spectrum\_fdm) / N);

title('FDM Signal (sentsignal)');

xlabel('Frequency (Hz)');

ylabel('Magnitude');

% Play the FDM signal

%sound(sentsignal, fs\_interpolated);

clearvars -except BWs sentsignal interp\_factor fs audioSignals samplingRates

%% Section3: Receiver

% Constants

offset = 0;

IF = 25e3; % Intermediate frequency (25 kHz)

fs\_interpolated = interp\_factor \* fs; % Interpolated sampling rate

receivedSignal = sentsignal; % Received FDM signal

%receivedSignal = awgn(sentsignal,10); % Received FDM signal

% Subplot 3: Time domain of receivedSignal

subplot(2, 2, 3);

t\_total = (0:length(receivedSignal) - 1) / fs\_interpolated;

plot(t\_total, receivedSignal);

title('FDM Signal (receivedSignal)');

xlabel('Time (s)');

ylabel('Amplitude');

% Subplot 4: Frequency domain of receivedSignal

subplot(2, 2, 4);

N = length(receivedSignal);

f = (-N/2:N/2-1) \* fs\_interpolated / N;

spectrum\_fdm = fftshift(fft(receivedSignal, N));

plot(f, abs(spectrum\_fdm) / N);

title('FDM Signal (receivedSignal)');

xlabel('Frequency (Hz)');

ylabel('Magnitude');

Rf\_enable = 1; % Enable/Disable RF filter

% Relaxed Filter Constraints

A\_pass = 2;

A\_stop1 = 20;

A\_stop2 = 20;

fprintf("\nSignal Processing Initialized.\n");

while true

% User input for channel selection

fprintf("\nChoose one of these channels (Enter 0 to Exit):\n");

fprintf("1. Short\_BBCArabic2\n");

fprintf("2. Short\_FM9090\n");

fprintf("3. Short\_QuranPalestine\n");

fprintf("4. Short\_RussianVoice\n");

fprintf("5. Short\_SkyNewsArabia\n");

i = input("Choose: ");

% Exit condition

if i == 0

fprintf("Exiting program...\n");

break;

elseif i < 1 || i > 5

fprintf("Invalid input! Please choose a number between 1 and 5, or 0 to exit.\n");

continue;

end

% Carrier frequency and bandwidth for the selected channel

fc\_mod = 100e3 + (i - 1) \* 50e3; % Carrier frequency for the selected channel

bw = BWs(i); % Bandwidth (fixed for all channels)

% Define proper increasing frequencies and cap at Nyquist

F\_stop1 = fc\_mod - bw ; % Stopband start

F\_pass1 = fc\_mod - bw/2; % Passband start

F\_pass2 = fc\_mod + bw/2; % Passband end

F\_stop2 = fc\_mod + bw ; % Stopband end

% RF Bandpass Filter

if Rf\_enable == 1

if F\_stop2 > fs\_interpolated / 2

error("Filter frequencies exceed Nyquist limit. Increase fs\_interpolated.");

end

rf\_bandpass\_filter = designfilt('bandpassiir', ...

'StopbandFrequency1', F\_stop1, 'PassbandFrequency1', F\_pass1, ...

'PassbandFrequency2', F\_pass2, 'StopbandFrequency2', F\_stop2, ...

'StopbandAttenuation1', A\_stop1, 'PassbandRipple', A\_pass, ...

'StopbandAttenuation2', A\_stop2, 'SampleRate', fs\_interpolated);

fprintf("RF Filter Order: %d\n", filtord(rf\_bandpass\_filter));

filtered\_signal = filter(rf\_bandpass\_filter, receivedSignal);

else

filtered\_signal = receivedSignal;

end

% Plot RF Stage Spectrum and Time Domain in a New Figure

figure;

% Subplot 1: Time Domain

subplot(2, 2, 1);

t\_total = (0:length(filtered\_signal) - 1) / fs\_interpolated;

plot(t\_total, filtered\_signal);

title('RF Stage (Time Domain)');

xlabel('Time (s)');

ylabel('Amplitude');

grid on;

% Subplot 2: Frequency Domain of filtered\_signal (RF Spectrum)

rf\_spectrum = fftshift(fft(filtered\_signal, 2^nextpow2(length(filtered\_signal)))); % Shifted FFT

N = length(rf\_spectrum); % Length of the signal

f = (-N/2:N/2-1) \* fs\_interpolated / N; % Correct frequency axis scaling

subplot(2, 2, 2);

plot(f, abs(rf\_spectrum) / N); % Normalize by the length of the signal

title('RF Stage After BPF ');

xlabel('Frequency (Hz)');

ylabel('Magnitude');

grid on;

% Step 2: Mix with Local Oscillator to Shift to IF

t = (1:length(filtered\_signal))' / fs\_interpolated; % Time vector

carrier\_LO = cos(2 \* pi \* (fc\_mod + IF) \* t); % Local oscillator signal

mixed\_signal\_IF = filtered\_signal .\* carrier\_LO; % Mixer output

% Plot IF Stage Spectrum in a Different Figure with Its Own Subplot

% Subplot 1: Time Domain of mixed\_signal\_IF (Optional if you want to plot it)

subplot(2, 2, 3);

t\_total\_IF = (0:length(mixed\_signal\_IF) - 1) / fs\_interpolated;

plot(t\_total\_IF, mixed\_signal\_IF);

title('IF Stage (Time Domain) ');

xlabel('Time (s)');

ylabel('Amplitude');

grid on;

% Subplot 2: Frequency Domain of mixed\_signal\_IF (IF Spectrum)

IF\_spectrum = fftshift(fft(mixed\_signal\_IF, 2^nextpow2(length(mixed\_signal\_IF)))); % Shifted FFT

subplot(2, 2, 4);

plot(f, abs(IF\_spectrum) / N); % Normalize by the length of the signal

title("IF Stage (freq domain) " );

xlabel("Frequency (Hz)");

ylabel("Magnitude");

grid off;

% Step 3: Bandpass Filter to Isolate IF

F\_stop1\_IF = max(0, IF - bw - 5e3); % Stopband start

F\_pass1\_IF = IF - bw; % Passband start

F\_pass2\_IF = IF + bw; % Passband end

F\_stop2\_IF = min(fs\_interpolated / 2 - 1, IF + bw + 5e3); % Stopband end

if\_bandpass\_filter = designfilt('bandpassiir', ...

'StopbandFrequency1', F\_stop1\_IF, 'PassbandFrequency1', F\_pass1\_IF, ...

'PassbandFrequency2', F\_pass2\_IF, 'StopbandFrequency2', F\_stop2\_IF, ...

'StopbandAttenuation1', A\_stop1, 'PassbandRipple', A\_pass, ...

'StopbandAttenuation2', A\_stop2, 'SampleRate', fs\_interpolated);

fprintf("IF Filter Order: %d\n", filtord(if\_bandpass\_filter));

filtered\_signal\_IF = filter(if\_bandpass\_filter, mixed\_signal\_IF);

% Step 4: Demodulate IF Signal

carrier\_IF = cos(2 \* pi \* IF \* t); % IF carrier signal

baseband\_signal = filtered\_signal\_IF .\* carrier\_IF; % Demodulation

% Step 5: Apply Low-Pass Filter to Recover Baseband Signal

F\_pass = bw; F\_stop = bw + 5e3;

lowpass\_filter = designfilt('lowpassiir', ...

'PassbandFrequency', F\_pass, 'StopbandFrequency', F\_stop, ...

'PassbandRipple', A\_pass, 'StopbandAttenuation', A\_stop1, ...

'SampleRate', fs\_interpolated);

Base\_Band\_received\_signal\_LPF = filter(lowpass\_filter, baseband\_signal);

% Resample and Playback the Demodulated Signal

Base\_Band\_received\_signal\_LPF = 4 \* resample(Base\_Band\_received\_signal\_LPF, 1, interp\_factor);

sound(Base\_Band\_received\_signal\_LPF, fs);

% Create a new figure

figure;

% Subplot 1: Time Domain of Base\_Band\_received\_signal\_LPF

subplot(2, 1, 1); % First position in a 2x2 grid

t\_total = (0:length(Base\_Band\_received\_signal\_LPF) - 1) / fs\_interpolated;

plot(t\_total, Base\_Band\_received\_signal\_LPF);

title('Baseband Signal (Time Domain)');

xlabel('Time (s)');

ylabel('Amplitude');

grid on;

% Subplot 3: Frequency Domain of Base\_Band\_received\_signal\_LPF (Baseband Spectrum)

baseband\_spectrum = fftshift(fft(Base\_Band\_received\_signal\_LPF, 2^nextpow2(length(Base\_Band\_received\_signal\_LPF)))); % Shifted FFT

N = length(baseband\_spectrum); % Length of the signal

f = (-N/2:N/2-1) \* fs\_interpolated / N; % Frequency axis scaling

subplot(2, 1, 2); % Third position in a 2x2 grid

plot(f, abs(baseband\_spectrum));

title('Baseband Spectrum After LPF');

xlabel('Frequency (Hz)');

ylabel('Magnitude');

grid on;

fprintf("Audio %d processed successfully.\n", i);

end

fprintf("Signal Processing Completed.\n");