Cairo University - Faculty of Engineering
Computer Engineering Department
Communication Engineering (ELC3252A6) – Fall 2020

Final Project

Team 25

Name	Section	BN
Muhammad Ahmad Hesham Mahmoud	2	15
Muhammad Alaa Abdelkhaleq Ahmad	2	22

January 2021

Q1)

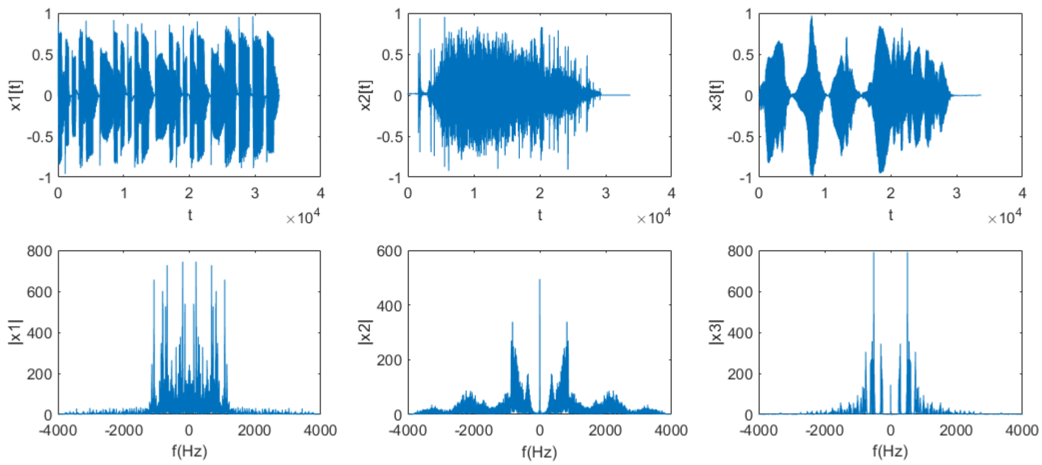

Figure 1 original signals in Time domain and magnitude spectrum

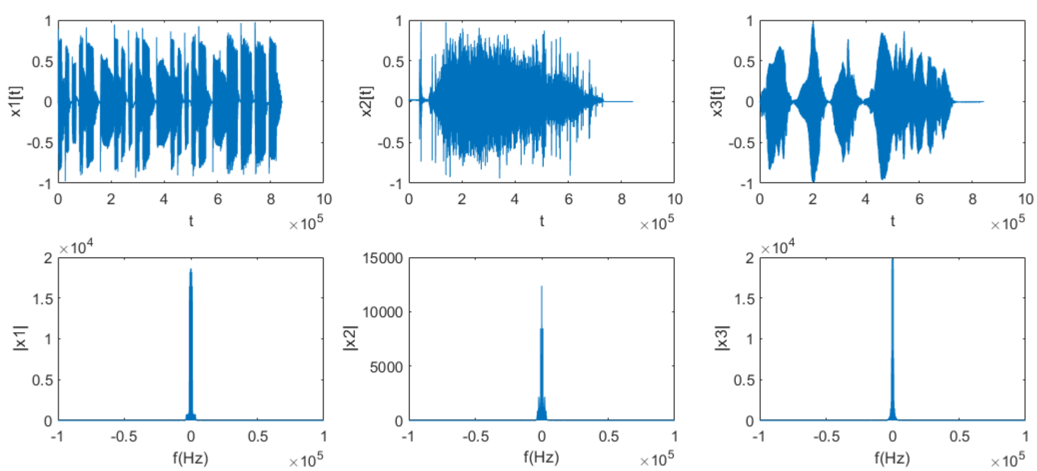

Figure 2 up sampled signals to have more room in frequency domain for manipulation

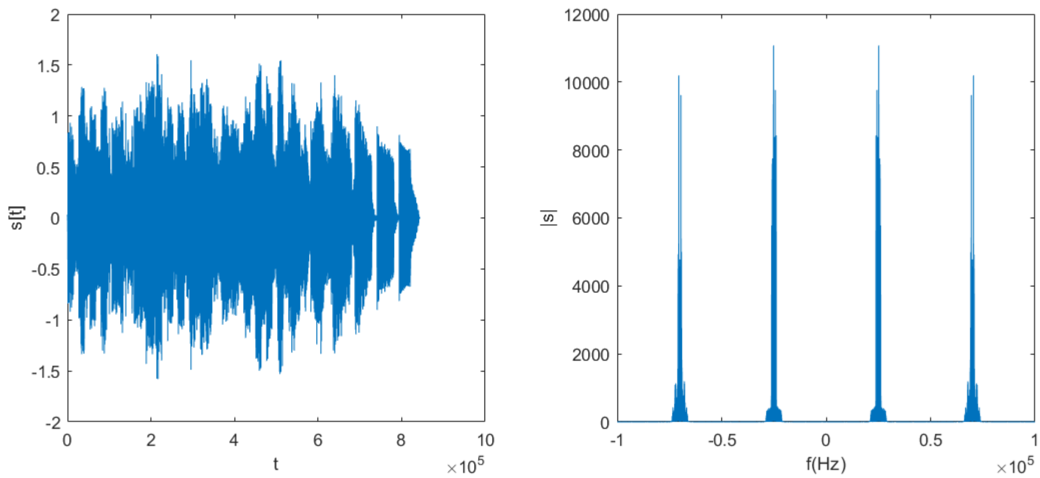

Figure 3 Modulated Signal in Time domain and magnitude spectrum

Q2)

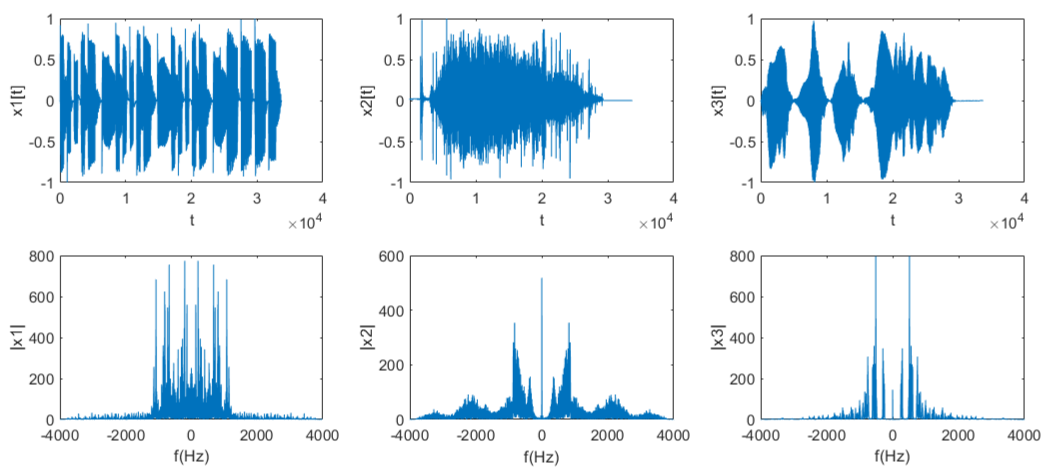

Figure 4 obtained signals after demodulation with phase shift 0

We notice that there is no interference at all between the three signals after demodulation with local carrier in phase with the carrier used in modulation due to wise choice of carrier frequency. As we can see in Figure 3 the first signal doesn't interfere with the other two signals and the other two signals don't interfere as they perpendicular to each other one has phase 0 while the other is phase 90.

Q3)

Due to producing phase shifts to the local carrier used in the demodulator for example the first signal x1 after demodulation will be $\frac{1}{2}X_1(t)cos(phaseShift)$ so if phaseShift changes to anything but zero this will cause attenuation to the recovered signal X_1.

While recovered X_2 will be $\frac{1}{2}[X_2(t)cos(phaseShift) - X_3sin(phaseShift)]$ and X_3 will be $\frac{1}{2}[X_3(t)cos(phaseShift) - X_2sin(phaseShift)]$

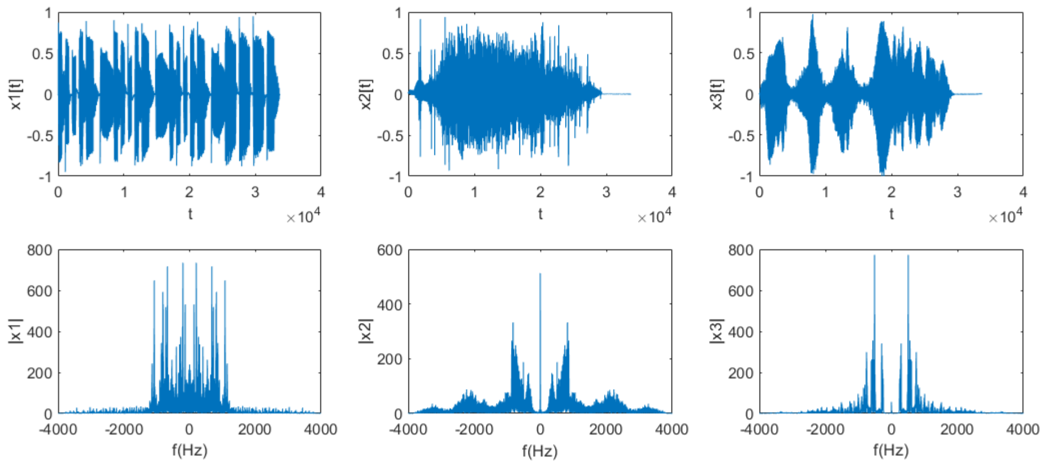

Figure 5 demodulated signals with phase shift 10

We begin to notice in Figure 5 that the signals X_2 and X_3 begin to interfere (cochannel interference). Also, attenuation starts to happen to all signals.

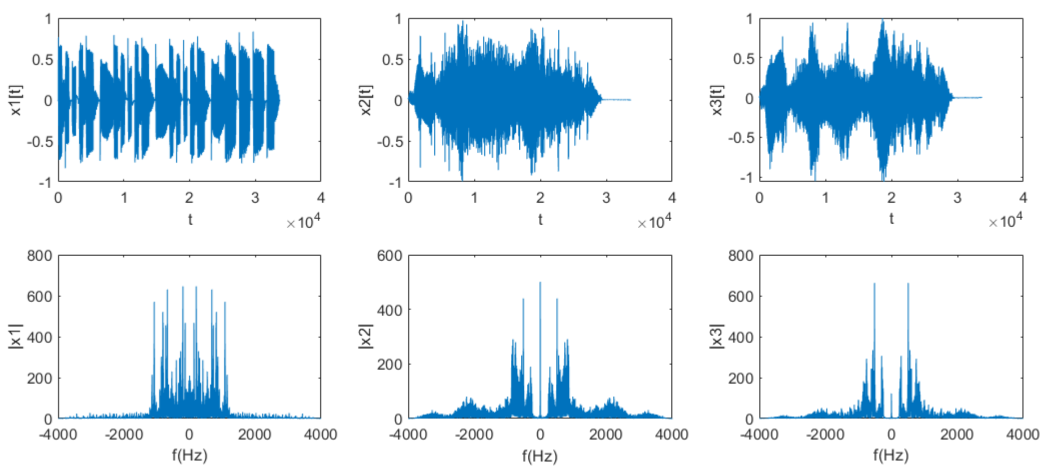

Figure 6 demodulated signals with phase shift 30

We begin to see in Figure 6 more attenuation in all signals and more cochannel interference between X_2 and X_3.

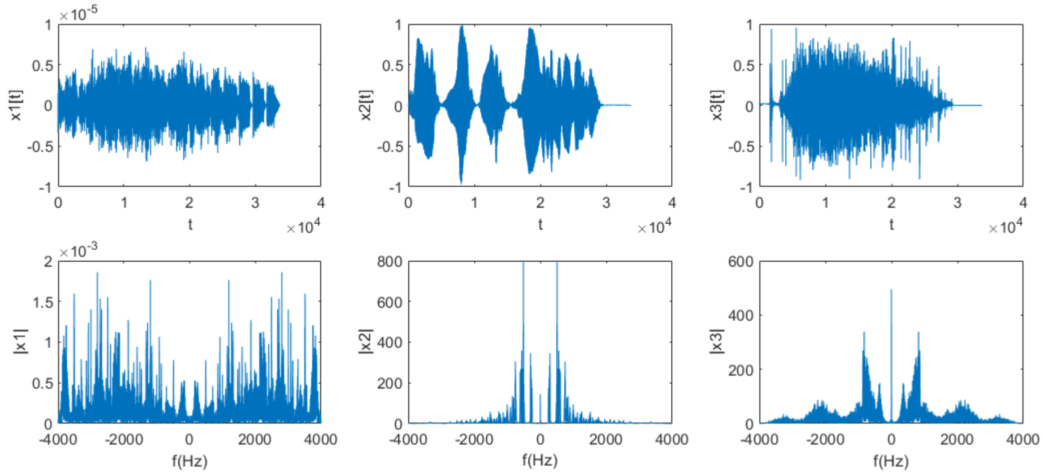

Figure 7 demodulated signals with phase shift 90

We notice in Figure 7 that the signal X_1 is now attenuated and has a very low amplitude (approx. zero) due to imperfection but it should be absolutely zero, we also notice that the signals X_2 and X_3 reached maximum interference as when we wanted to retrieve signal X_2 we got X_3 and vice versa.

Appendix (codes were developed using MATLAB R2018a)

```
% assumptions made on the signals:
%     1- they have same sampling rate
%     2- they are less than 10 seconds
%     3- they are mono not stereo

speechSignalFileName = 'Team25_speech signal';



[message1,samplingFrequency1]=audioread([speechSignalFileName '_1.wav']);
[message2,samplingFrequency2]=audioread([speechSignalFileName '_2.wav']);
[message3,samplingFrequency3]=audioread([speechSignalFileName '_3.wav']);

% make lengths of all signals equal
padding1 = 0;
padding2 = 0;
padding3 = 0;
maxSamples = max(max(length(message2), length(message3)), length(message1));
if length(message1) ~= maxSamples
    padding1 = maxSamples - length(message1);
    message1 = padarray(message1, maxSamples - length(message1), 0, "post");
end
if length(message2) ~= maxSamples
    padding2 = maxSamples - length(message2);
    message2 = padarray(message2, maxSamples - length(message2), 0, "post");
end
if length(message3) ~= maxSamples
    padding3 = maxSamples - length(message3);
    message3 = padarray(message3, maxSamples - length(message3), 0, "post");
end

displaySignals(message1, message2, message3, samplingFrequency1,'After extending to same length');

% upsampling to make manipulation easier
upSamplingRate = 25;
message1 = resample(message1, upSamplingRate, 1);
message2 = resample(message2, upSamplingRate, 1);
message3 = resample(message3, upSamplingRate, 1);
samplingFrequency1 = samplingFrequency1 * upSamplingRate;
samplingFrequency2 = samplingFrequency2 * upSamplingRate;
samplingFrequency3 = samplingFrequency3 * upSamplingRate;

displaySignals(message1, message2, message3, samplingFrequency1,'After upsampling');


% modulating signals
duration = length(message1) ./ samplingFrequency1;
t=-(duration-1/samplingFrequency1) / 2:1/samplingFrequency1:(duration-1/samplingFrequency1) / 2 ;

% current Fs is 200kHz assume worst case that our sound signals take a
% bandwidth of 20Khz
fcarrier1 = 25000;
fcarrier2 = 70000;

carrier1 = cos(2 * pi * fcarrier1 * t);
carrier2 = cos(2 * pi * fcarrier2 * t);
carrier3 = sin(2 * pi * fcarrier2 * t);

s1 = message1' .* carrier1;
```

```matlab
s2 = message2' .* carrier2;
s3 = message3' .* carrier3;

displaySignals(s1, s2, s3, samplingFrequency1, "signals after modulating the carriers");

s = s1 + s2 + s3;


% plot s in time and frequency domain
figure('name', 'modulated Signal');
set(gcf,'position',[100 100 1000 400]);
subplot(1,2,1);plot(s);ylabel("s[t]");xlabel("t");
subplot(1,2,2);[x, y] = audioMagnitudeSpectrum(s, samplingFrequency1);plot(x,
y);ylabel("|s|");xlabel("f(Hz)");

% demodulate the signal s

demodulator(s, 0, samplingFrequency1, upSamplingRate, fcarrier1, fcarrier2, speechSignalFileName);
demodulator(s, 10, samplingFrequency1, upSamplingRate, fcarrier1, fcarrier2, speechSignalFileName);
demodulator(s, 30, samplingFrequency1, upSamplingRate, fcarrier1, fcarrier2, speechSignalFileName);
demodulator(s, 90, samplingFrequency1, upSamplingRate, fcarrier1, fcarrier2, speechSignalFileName);



function [x, y] = audioMagnitudeSpectrum(signal, Fs)
    x = (-Fs/2:Fs/length(signal):Fs/2-Fs/length(signal));
    y = abs(fftshift(fft(signal)));
end

function a = displaySignals(signal1, signal2, signal3, Fs, figureTitle)
    a = 0;
    figure('name', figureTitle)
    set(gcf,'position',[100 100 1000 400])

    subplot(2,3,1);
    plot(signal1);
    ylabel("x1[t]");xlabel("t");

    subplot(2,3,2);
    plot(signal2);
    ylabel("x2[t]");xlabel("t");

    subplot(2,3,3);
    plot(signal3);
    ylabel("x3[t]");xlabel("t");

    subplot(2,3,4);
    [x, y] = audioMagnitudeSpectrum(signal1, Fs);
    plot(x, y);
    ylabel("|x1|");xlabel("f(Hz)");

    subplot(2,3,5);
    [x, y] = audioMagnitudeSpectrum(signal2, Fs);
    plot(x, y);
    ylabel("|x2|");xlabel("f(Hz)");

    subplot(2,3,6);
    [x, y] = audioMagnitudeSpectrum(signal3, Fs);
    plot(x, y);
    ylabel("|x3|");xlabel("f(Hz)");
end
```

```matlab
function c = demodulator(signal, phaseShiftDegrees, Fs, upSamplingRate, fcarrier1, fcarrier2,
speechSignalFileName)
    c=0;
    duration = length(signal) ./ Fs;
    t=-(duration-1/Fs) / 2:1/Fs:(duration-1/Fs) / 2 ;
    carrier1 = cos(2 * pi * fcarrier1 * t + (phaseShiftDegrees * pi / 180));
    carrier2 = cos(2 * pi * fcarrier2 * t + (phaseShiftDegrees * pi / 180));
    carrier3 = sin(2 * pi * fcarrier2 * t + (phaseShiftDegrees * pi / 180));

    % demodulate
    output1 = 2*(carrier1 .* signal);
    output2 = 2*(carrier2 .* signal);
    output3 = 2*(carrier3 .* signal);

    % applying low pass filter
    output1 = lowpass(output1, 20000, Fs);
    output2 = lowpass(output2, 20000, Fs);
    output3 = lowpass(output3, 20000, Fs);

    % downsmapling the signals to the original sampling rate
    output1 = resample(output1, 1,upSamplingRate);
    output2 = resample(output2, 1,upSamplingRate);
    output3 = resample(output3, 1,upSamplingRate);
    Fs = Fs/upSamplingRate;


    displaySignals(output1, output2, output3, Fs, ['demodulation output phase'
int2str(phaseShiftDegrees)]);

    % save audio files
    audiowrite([speechSignalFileName '_1_' int2str(phaseShiftDegrees) '.wav'], output1, Fs);
    audiowrite([speechSignalFileName '_2_' int2str(phaseShiftDegrees) '.wav'], output2, Fs);
    audiowrite([speechSignalFileName '_3_' int2str(phaseShiftDegrees) '.wav'], output3, Fs);
end
```