**Analog final lab assignment**

|  |  |
| --- | --- |
| **ID** | **NAME** |
| **6338** | Ahmed Mohamed ahmed ali |
| **6071** | Ahmed Wael Mohamed |
| **6297** | Adel Ashraf Mohamed |
| **6160** | Mazen Medhat Farid |
| **6646** | Ahmed Tarek Ahmed |
| **4629** | Mahmoud Osama |

**Experiment 1:**

**Code:**

clc

clear

close all

**Read the audio file and get its sampling frequency**

[y, F] = audioread('eric.wav');

ty = length(y)/F;

t = linspace(0, ty, length(y));

**Input signal conversion to frequency domain (to apply ideal LPF)**

z = fftshift(fft(y));

f = linspace(-F/2, F/2, length(y));

figure

plot(f, abs(z),'g');

title('Original Signal in Frequency Domain');

**generate ideal filter**

% Searching for indices where f = -4000 and f = 4000

for i = 1:length(f)

if (abs((f(i)+4.0000e+3)) < 0.01)

index1 = i;

end

if (abs((f(i)-4.0000e+3)) < 0.01)

index2 = i;

break;

end

end

% Generating a rect from index1 to index2

range = index2-index1;

step = [zeros(1, index1) ones(1, range) zeros(1, length(f)-index2)];

step = step.';

**Multiplying input signal by rect to eliminate frequencies other than 4k**

yfilteredFreq = step.\*z;

yfilteredFreqAbs = abs(step.\*z);

figure

plot(f, yfilteredFreqAbs,'g');

xlim([-15000 15000]);

title('Filtered Signal in Frequency Domain');

% Converting the filtered signal to time domain to be modulated

yfiltered = ifft(fftshift(yfilteredFreq));

figure

plot(t, yfiltered,'r');

title('Filtered Signal in Time Domain');

% Signal after low pass filter

sound(yfiltered, F);

pause(9);

**Filtered signal resampling**

maxAmplitude = max(yfiltered);

yfiltered = resample(yfiltered, 500000, F);

yfiltered = yfiltered.';

interval = length(yfiltered);

**DSB-SC Modulation and Demodulation**

**Modulation**

t = linspace(0, ty, interval);

dsbsc = 5\*cos(2\*pi\*100000\*t).\*yfiltered;

figure

plot(t, dsbsc,'r');

title('Modulted DSB-SC Signal in Time Domain');

zmsc = abs(fftshift(fft(dsbsc)));

fmsc = linspace(-500000/2, 500000/2, length(dsbsc));

figure

plot(fmsc, zmsc,'g');

xlim([-150000 150000]);

ylim([0 300]);

title('Modulated DSB-SC Signal in Frequency Domain');

**Demodulation**

dsbscEnvelope = abs(hilbert(dsbsc));

dsbscEnvelope = resample(dsbscEnvelope, F, 500000);

t = 0:1/F:length(dsbscEnvelope)/(F);

t = t(1:end-1);

figure

plot(t, dsbscEnvelope,'r');

title('Demodulted DSB-SC Signal in Time Domain');

zdsc = abs(fftshift(fft(dsbscEnvelope)));

fdsc = linspace(-F/2, F/2, length(dsbscEnvelope));

figure

plot(fdsc, zdsc,'g');

xlim([-8000 8000]);

ylim([0 1500]);

title('Demodulated DSB-SC Signal in Frequency Domain');

sound(dsbscEnvelope, F);

pause(9);

**Coherent detection**

dsbsc0 = awgn(dsbsc,0);

dsbsc10 = awgn(dsbsc,10);

dsbsc30 = awgn(dsbsc,30);

t = linspace(0, ty, length(dsbsc));

vpe = dsbsc.\*cos((2\*pi\*100000\*t) + deg2rad(20));

vfe = dsbsc.\*cos(2\*pi\*100100\*t);

v0 = dsbsc0.\*cos(2\*pi\*100000\*t);

v10 = dsbsc10.\*cos(2\*pi\*100000\*t);

v30 = dsbsc30.\*cos(2\*pi\*100000\*t);

v0 = resample(v0, F, 500000);

v10 = resample(v10, F, 500000);

v30 = resample(v30, F, 500000);

vpe = resample(vpe, F, 500000);

vfe = resample(vfe, F, 500000);

zsc0 = fftshift(fft(v0));

zsc10 = fftshift(fft(v10));

zsc30 = fftshift(fft(v30));

zscpe = fftshift(fft(vpe));

zscfe = fftshift(fft(vfe));

fsc = linspace(-F/2, F/2, length(v0));

for i = 1:length(fsc)

if (abs((fsc(i)+4.0000e+3)) < 0.01)

index1 = i;

end

if (abs((fsc(i)-4.0000e+3)) < 0.01)

index2 = i;

break;

end

end

range = index2-index1;

step = [zeros(1, index1) ones(1, range) zeros(1, length(fsc)-(index2))];

dsbscCoherent0 = step.\*zsc0;

dsbscCoherent10 = step.\*zsc10;

dsbscCoherent30 = step.\*zsc30;

dsbscCoherentpe = step.\*zscpe;

dsbscCoherentfe = step.\*zscfe;

figure

plot(fsc, abs(dsbscCoherent0),'g');

title('Demodulated DSB-SC Signal using Coherent Detection with SNR = 0 in Frequency Domain');

xlim([-8000 8000]);

figure

plot(fsc, abs(dsbscCoherent10),'g');

title('Demodulated DSB-SC Signal using Coherent Detection with SNR = 10 in Frequency Domain');

xlim([-8000 8000]);

figure

plot(fsc, abs(dsbscCoherent30),'g');

title('Demodulated DSB-SC Signal using Coherent Detection with SNR = 30 in Frequency Domain');

xlim([-8000 8000]);

figure

plot(fsc, abs(dsbscCoherentpe),'g');

title('Demodulated DSB-SC Signal using Coherent Detection with Phase Error = 20 in Frequency Domain');

xlim([-8000 8000]);

figure

plot(fsc, abs(dsbscCoherentfe),'g');

title('Demodulated DSB-SC Signal using Coherent Detection with Frequency Error = 100 HZ in Frequency Domain');

xlim([-8000 8000]);

dsbscCoherentTime0 = ifft(fftshift(dsbscCoherent0));

dsbscCoherentTime10 = ifft(fftshift(dsbscCoherent10));

dsbscCoherentTime30 = ifft(fftshift(dsbscCoherent30));

dsbscCoherentTimepe = ifft(fftshift(dsbscCoherentpe));

dsbscCoherentTimefe = ifft(fftshift(dsbscCoherentfe));

t = linspace(0, ty, length(dsbscCoherentTime0));

figure

plot(t, abs(dsbscCoherentTime0),'r');

title('Demodulated DSB-SC Signal using Coherent Detection with SNR = 0 in Time Domain');

sound(abs(dsbscCoherentTime0), F);

pause(9);

figure

plot(t, abs(dsbscCoherentTime10),'r');

title('Demodulated DSB-SC Signal using Coherent Detection with SNR = 10 in Time Domain');

sound(abs(dsbscCoherentTime10), F);

pause(9);

figure

plot(t, abs(dsbscCoherentTime30),'r');

title('Demodulated DSB-SC Signal using Coherent Detection with SNR = 30 in Time Domain');

sound(abs(dsbscCoherentTime30), F);

pause(9);

figure

plot(t, abs(dsbscCoherentTimepe),'r');

title('Demodulated DSB-SC Signal using Coherent Detection with Phase Error = 20 in Time Domain');

sound(abs(dsbscCoherentTimepe), F);

pause(9);

figure

plot(t, abs(dsbscCoherentTimefe),'g');

title('Demodulated DSB-SC Signal using Coherent Detection with Frequency Error = 100 HZ in Time Domain');

sound(abs(dsbscCoherentTimefe), F);

pause(9);

**DSB-TC Modulation and Demodulation**

**Modulation**

t = linspace(0, ty, interval);

dsbtc = 2.\*maxAmplitude.\*(1+(0.5/maxAmplitude).\*yfiltered).\*cos(2.\*pi.\*100000.\*t);

figure

plot(t, dsbtc,'r');

title('Modulted DSB-TC Signal in Time Domain');

zmtc = abs(fftshift(fft(dsbtc)));

fmtc = linspace(-500000/2, 500000/2, length(dsbtc));

figure

plot(fmtc, zmtc,'g');

xlim([-150000 150000]);

ylim([0 1000]);

title('Modulated DSB-TC Signal in Frequency Domain');

**Demodulation**

dsbtcEnvelope = abs(hilbert(dsbtc));

dsbtcEnvelope = resample(dsbtcEnvelope, F, 500000);

dsbtcEnvelope=dsbtcEnvelope-mean(dsbtcEnvelope);

t = 0:1/F:length(dsbtcEnvelope)/(F);

t = t(1:end-1);

figure

plot(t, dsbtcEnvelope,'r');

title('Demodulted DSB-TC Signal in Time Domain');

zdtc = abs(fftshift(fft(dsbtcEnvelope)));

fdtc = linspace(-F/2, F/2, length(dsbtcEnvelope));

figure

plot(fdtc, zdtc,'g');

xlim([-8000 8000]);

ylim([0 1000]);

title('Demodulated DSB-TC Signal in Frequency Domain');

sound(dsbtcEnvelope, F);

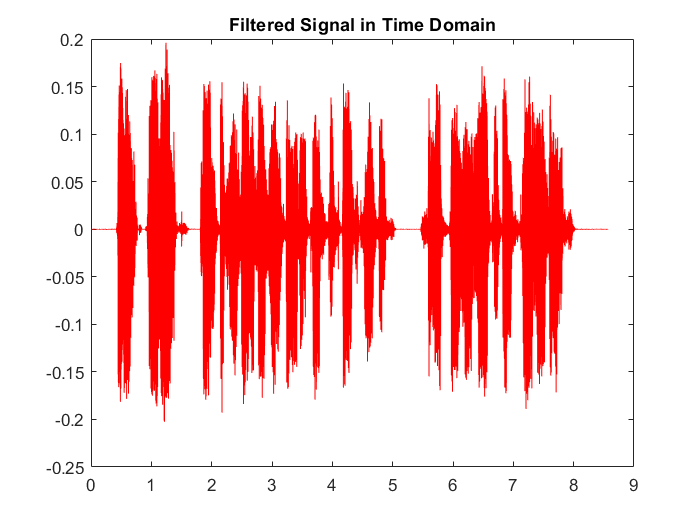
**output:**

Chart, histogram

Description automatically generated

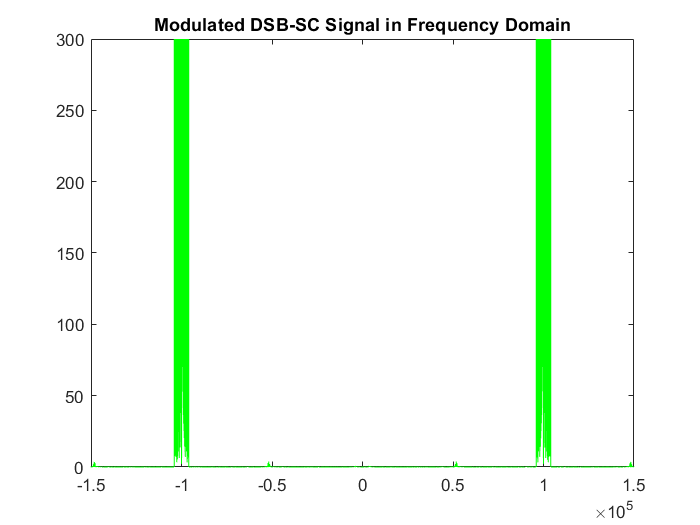
Chart, histogram

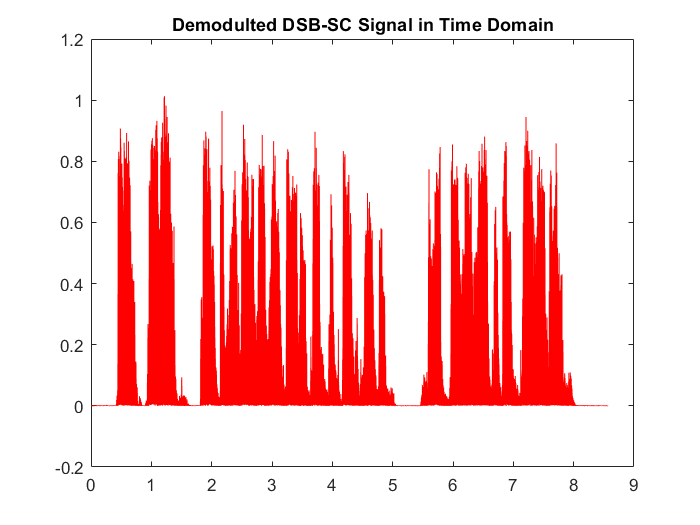
Description automatically generated



Chart, shape

Description automatically generated





Chart, histogram

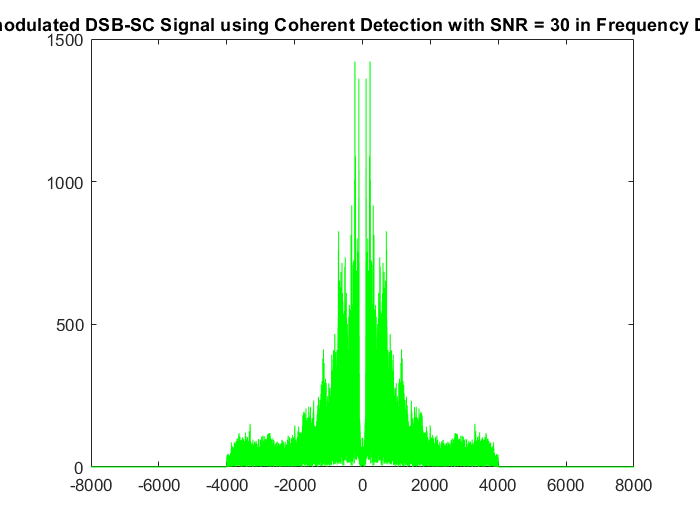
Description automatically generated

Chart, histogram

Description automatically generated

Chart, histogram

Description automatically generated

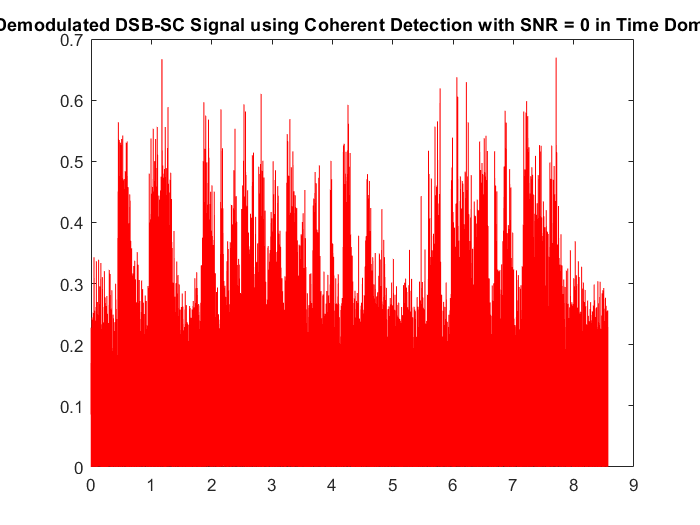


Chart, histogram

Description automatically generated

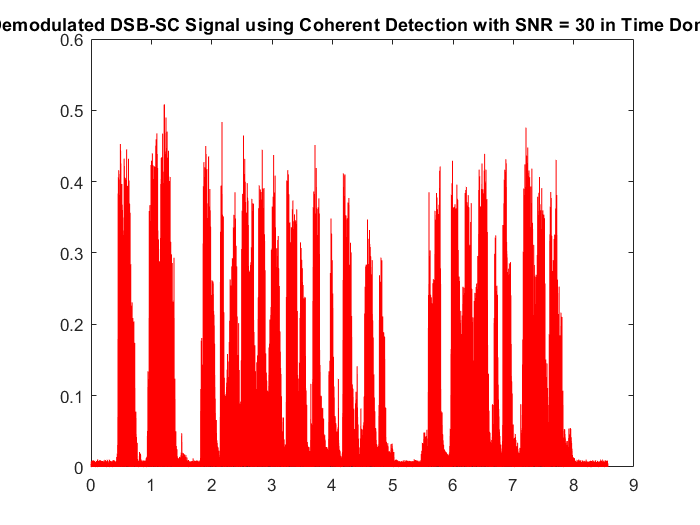
Chart, histogram

Description automatically generated



Chart, histogram

Description automatically generated



Chart, bar chart, histogram

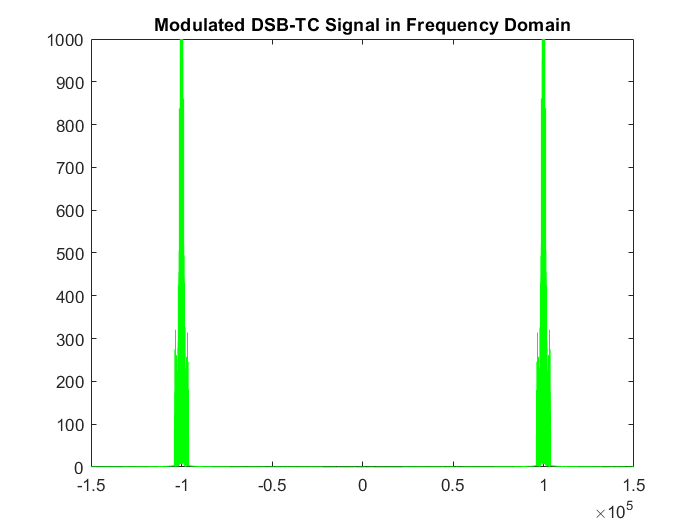
Description automatically generated

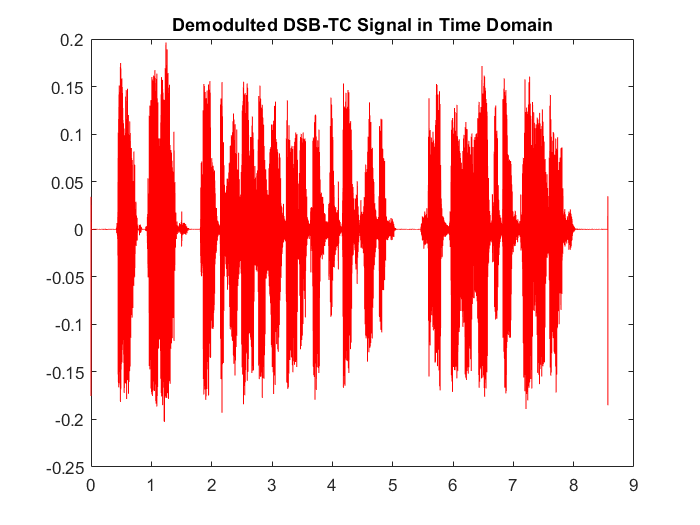
Chart, histogram

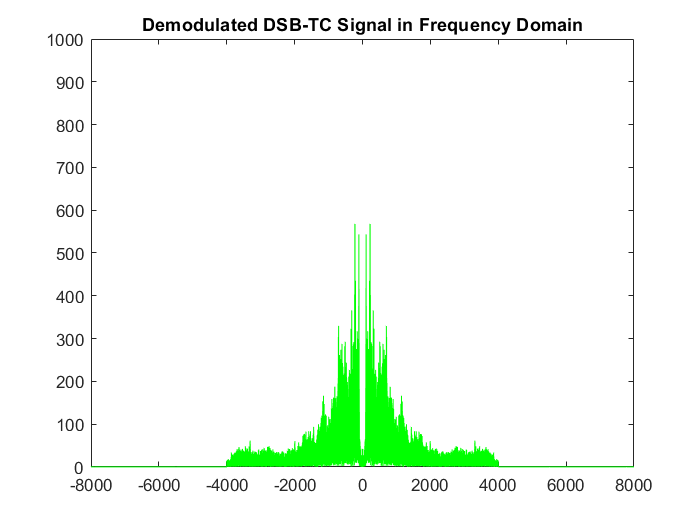
Description automatically generated

Chart, histogram

Description automatically generated







**Experiment Conclusions**

🡪 By using Envelope Detector, we observe that there is more

distortion in case of DSB-SC. DSB-SC should be demodulated by

coherent detection. Envelope Detector can be used to

demodulate DSB-TC.

🡪It is called beat effect

**Experiment 2:**

**Code:**

clc

clear

close all

**Read the audio file and get its sampling frequency**

[y, F] = audioread('eric.wav');

ty = length(y)/F;

t = linspace(0, ty, length(y));

T=linspace(0, ty, length(y));

plot(t,y,'r');

title('Original Signal in Time Domain');

**Input signal conversion to frequency domain**

z = fftshift(fft(y));

zf=abs(z);

f = linspace(-F/2, F/2, length(y));

plot(f,zf,'g');

title('Original Signal in Frequency Domain');

**apply ideal LPF ,Searching for indices where f = -4000 and f = 4000**

for i = 1:length(f)

if (abs((f(i)+4.0000e+3)) < 0.01)

index1 = i;

end

if (abs((f(i)-4.0000e+3)) < 0.01)

index2 = i;

break;

end

end

%Generating a rect from index1 to index2

range = index2-index1;

step = [zeros(1, index1) ones(1, range) zeros(1, length(f)-index2)];

step = step.';

**Multiplying input signal by rect to eliminate frequencies other than 4k**

yfilteredFreq = step.\*z;

yfilteredFreqAbs = abs(step.\*z);

figure

plot(f, yfilteredFreqAbs,'g');

xlim([-4.5 4.5].\*10^3)

title('Filtered Signal in Frequency Domain at 4KHZ (ideal filter)');

**Converting the filtered signal to time domain to be modulated DSB-SC**

yfiltered= ifft(ifftshift(yfilteredFreq));

yfiltered=yfiltered.';

figure

plot(T,yfiltered,'r');

title('Filtered Signal in Time Domain at 4KHZ (ideal filter)');

sound(yfiltered, F);

pause(9);

%Filtered signal resampling

maxAmplitude = max(yfiltered);

yfiltered = resample(yfiltered, 500000, F);

interval = length(yfiltered);

**DSB-SC Modulation**

t = linspace(0, ty, interval);

carrier=cos(2\*pi\*100000\*t);

dsbsc =carrier.\*yfiltered;

figure

plot(t, dsbsc,'r');

title('Modulted DSB-SC Signal in Time Domain');

zm\_sc=fftshift(fft(dsbsc));

zmsc = abs(fftshift(fft(dsbsc)));

fmsc = linspace(-500000/2, 500000/2, length(dsbsc));

figure

plot(fmsc, zmsc,'g');

xlim([-150000 150000]);

ylim([0 1000]);

title('Modulated DSB-SC Signal in Frequency Domain');

xlim([-12 12].\*10^4)

**remove USB to get SSB using ideal filter**

for i1 = 1:length(dsbsc)

if (abs((fmsc(i1)+1.0000e+5)) < 0.1)

idx1 = i1;

end

if (abs((fmsc(i1)-1.0000e+5)) < 0.1)

idx2 = i1;

break;

end

end

%Generating a rect from idx1 to idx2

range1 = idx2-idx1;

step1 = [zeros(1, idx1) ones(1, range1) zeros(1, length(fmsc)-idx2)];

LSB\_Frequency = step1.\*zm\_sc;

LSB\_FAbs = abs(step1.\*zm\_sc);

figure

plot(fmsc, LSB\_FAbs,'g');

xlim([-150000 150000]);

ylim([0 1000]);

title('Obtain LSB in Frequency Domain using from DSB-SC (ideal filter)');

xlim([-12 12].\*10^4)

LSB\_time = ifft(ifftshift(LSB\_Frequency));

**demodulation of SSB using coherent detector ideal filter**

SSB\_SC = LSB\_time.\*carrier;

SSB\_SC = resample(SSB\_SC,F,500000);

SSB\_SC\_ff= fftshift(fft(SSB\_SC)) ;

%ideal lpf to get signal in frequency domain

SSB\_SC\_ff=SSB\_SC\_ff(1:end-1);

SSB\_SC\_ff =SSB\_SC\_ff.\*step';

SSB\_SC\_time =ifft(ifftshift(SSB\_SC\_ff)) ;

signal\_frequency\_domain= fftshift(fft(SSB\_SC\_time));

figure

plot(T,SSB\_SC\_time,'r');

title('recieved LSB in Time Domain (ideal filter)');

figure;

plot(f,abs(signal\_frequency\_domain),'g');

title('recieved LSB in Freqency Domain (ideal filter)');

xlim([-4.5 4.5].\*10^3)

sound(SSB\_SC\_time,F);

pause(9);

**remove USB to get SSB using butter filter**

**Butterworth filter BPF to get LSB**

fnorm = 500000/2;

BW\_fitler=(500000\*4000)/48000;

[numerator, denomenator] = butter(4,[100000 100000+BW\_fitler]/fnorm,'bandpass');

Filter\_DSB = filter(numerator, denomenator, dsbsc);

LSB\_Butter = Filter\_DSB.\*carrier;

plot(fmsc,abs(fftshift(fft(Filter\_DSB))),'g');

title('Obtain LSB in Freqency Domain (Butter filter)');

%down sample butter filter

LSB\_down =resample(LSB\_Butter,F,500000);

LSB\_down =LSB\_down(1:end-1);

lsb\_freq=abs(fftshift(fft(LSB\_down)));

**Butterworth filter LPF to get LSB after demodulation**

[numerator, denomenator] = butter(4,4000/(F/2));

LSB\_LPF\_Time = filter(numerator, denomenator, LSB\_down);

LSB\_LPF\_Freq = fftshift(fft(LSB\_LPF\_Time));

figure

plot(LSB\_LPF\_Time,'r');

title('recieved LSB in Time Domain (butter filter)');

figure

plot(f,abs(LSB\_LPF\_Freq),'g');

title('recieved LSB in Freqency Domain (butter filter)');

xlim([-4.5 4.5].\*10^3)

%NOISE is added to signal

noised\_signal\_0 = awgn(LSB\_time,0, 'measured');

noised\_signal\_10 = awgn(LSB\_time,10,'measured');

noised\_signal\_30 = awgn(LSB\_time,30,'measured');

noised\_signal\_0=noised\_signal\_0';

noised\_signal\_10=noised\_signal\_10';

noised\_signal\_30=noised\_signal\_30';

carrier=carrier.';

LSB0 = noised\_signal\_0.\*carrier;

coherent0 = resample(LSB0,F,500000);

coherent0 = coherent0(1:end-1);

coherentFreq0 = fftshift(fft(coherent0));

coherentFilter0 = step.\*coherentFreq0;

coherentTime0 = ifft(ifftshift(coherentFilter0)) ;

figure;

plot(T,coherentTime0,'r');

title('recieved LSB in Time Domain (ideal filter) with noise SNR=0');

figure;

plot(f,abs(coherentFilter0),'g');

title('recieved LSB in Freq Domain (ideal filter) with noise SNR=0');

xlim([-4.5 4.5].\*10^3)

sound(real(coherentTime0),F);

pause(9);

LSB10 = noised\_signal\_10.\*carrier;

coherent10 = resample(LSB10,F,500000);

coherent10 = coherent10(1:end-1);

coherentFreq10 = fftshift(fft(coherent10));

coherentFilter10 = step.\*coherentFreq10;

coherentTime10 = ifft(ifftshift(coherentFilter10)) ;

figure;

plot(T,coherentTime10,'r');

title('recieved LSB in Time Domain (ideal filter) with noise SNR=10');

figure;

plot(f,abs(coherentFilter10),'g');

title('recieved LSB in Freq Domain (ideal filter) with noise SNR=10');

xlim([-4.5 4.5].\*10^3)

sound(real(coherentTime10),F);

pause(9);

LSB30 = noised\_signal\_30.\*carrier;

coherent30 = resample(LSB30,F,500000);

coherent30 = coherent30(1:end-1);

coherentFreq30 = fftshift(fft(coherent30));

coherentFilter30 = step.\*coherentFreq30;

coherentTime30 = ifft(ifftshift(coherentFilter30)) ;

figure;

plot(T,coherentTime30,'r');

title('recieved LSB in Time Domain (ideal filter) with noise SNR=30');

figure;

plot(f,abs(coherentFilter30),'g');

title('recieved LSB in Freq Domain (ideal filter) with noise SNR=30');

xlim([-4.5 4.5].\*10^3)

sound(real(coherentTime30),F);

pause(9);

**OBTAIN SSB-TC using ideal filter**

A=2\*max(LSB\_time);

carrier=carrier.';

SSB\_TC=(A+LSB\_time).\*carrier;

SSB\_TC\_freq = fftshift(fft(SSB\_TC));

step=step.';

step=resample(step,500000,F);

SSB\_SCinFreqIdeal = SSB\_TC\_freq.\*step;

SSB\_TC\_time = ifft(ifftshift(SSB\_SCinFreqIdeal));

envelopeSSBTC=abs(hilbert(real(SSB\_TC\_time)));

%down sample envelop detector

envelopeSSBTC=resample(envelopeSSBTC,F,500000);

envelopeSSBTC=envelopeSSBTC-mean(envelopeSSBTC);

figure;

plot(T,envelopeSSBTC(1:end-1),'r');

title('Demodulated SSB-TC Signal in Time Domain');

zdtc = abs(fftshift(fft(envelopeSSBTC)));

fdtc = linspace(-F/2, F/2, length(envelopeSSBTC));

figure

plot(fdtc, zdtc,’g’);

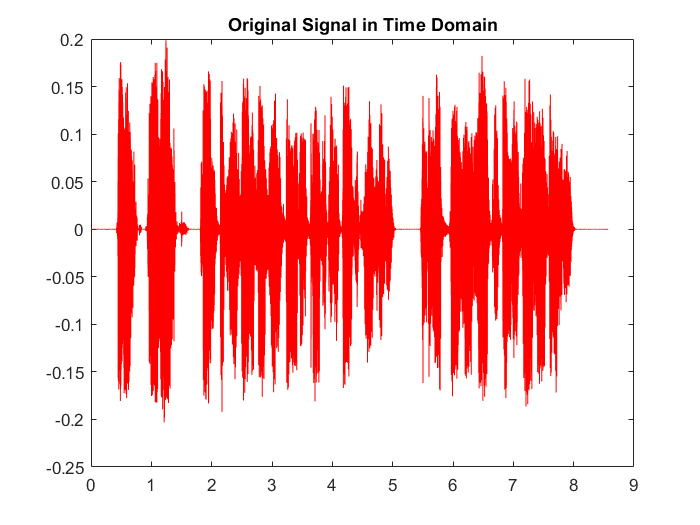
xlim([-5000 5000]);

ylim([0 300]);

title('Demodulated SSB-TC Signal in Frequency Domain');

sound(real(envelopeSSBTC),F);

**output:**



Chart, histogram

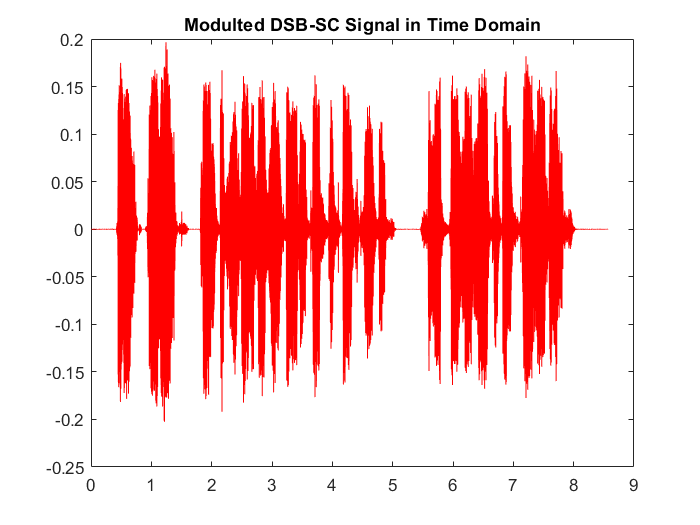
Description automatically generated

Chart, histogram

Description automatically generated

Chart, shape

Description automatically generated



Chart, histogram

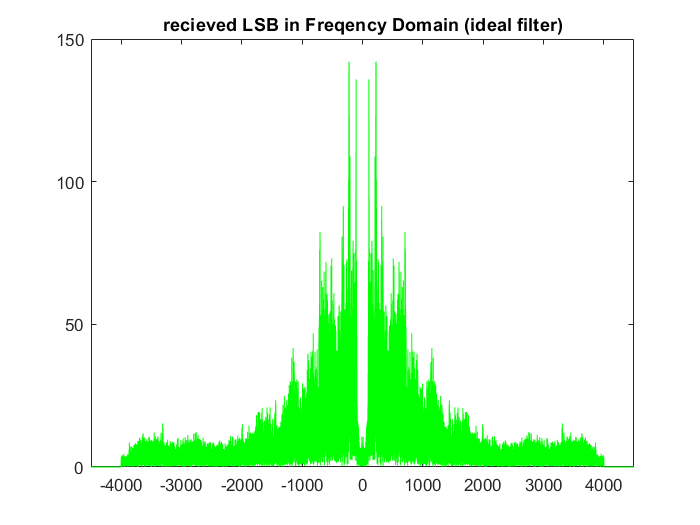
Description automatically generated

Chart, histogram

Description automatically generated

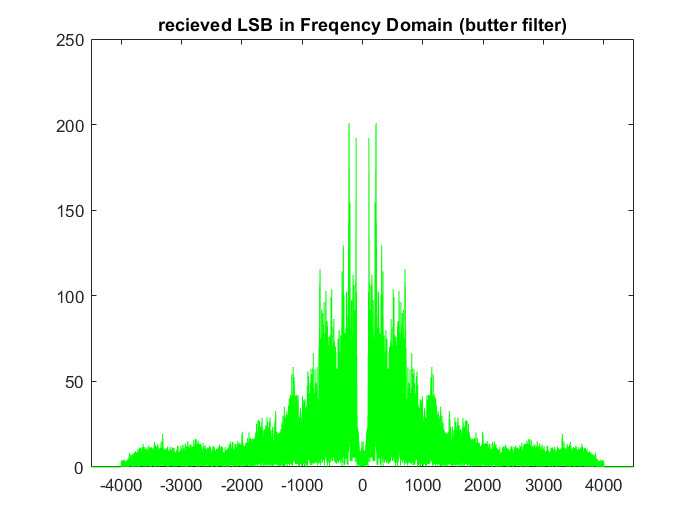
Chart

Description automatically generated



Chart, shape

Description automatically generated



Chart

Description automatically generated

Chart, histogram

Description automatically generated

Chart, histogram

Description automatically generated

Chart

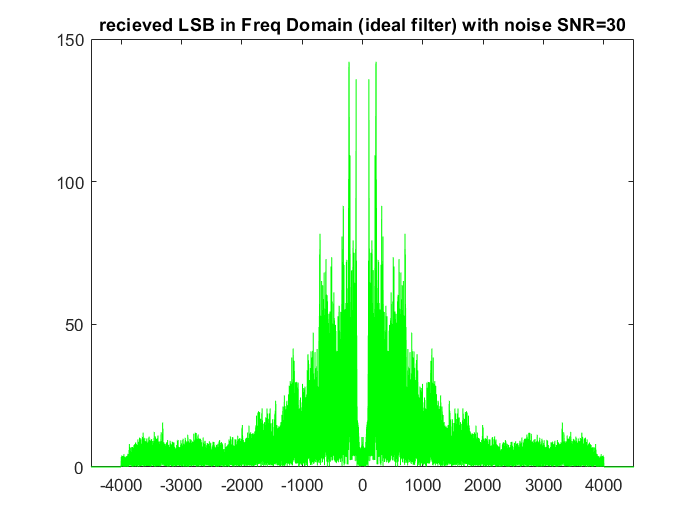
Description automatically generated

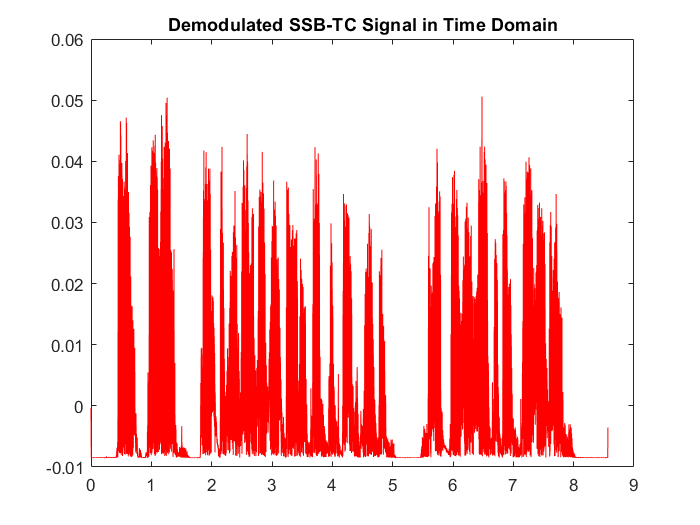
Chart, histogram

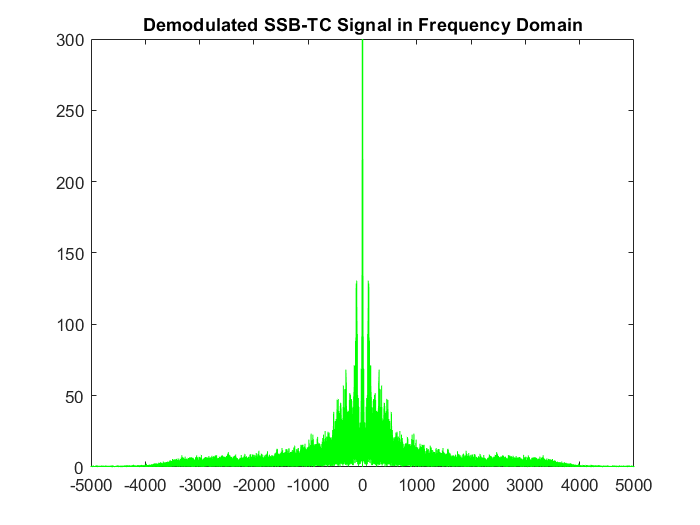
Description automatically generated

Chart, shape

Description automatically generated







**Experiment 3:**

**Code:**

clc

clear

close all

**Read the audio file and get its sampling frequency**

[y, F] = audioread('eric.wav');

ty = length(y)/F;

t = linspace(0, ty, length(y));

**Input signal conversion to frequency domain (to apply ideal LPF)**

z = fftshift(fft(y));

f = linspace(-F/2, F/2, length(y));

figure

plot(f, abs(z),'g');

title('Original Signal in Frequency Domain');

**Searching for indices where f = -4000 and f = 4000**

for i = 1:length(f)

if (abs((f(i)+4.0000e+3)) < 0.01)

index1 = i;

end

if (abs((f(i)-4.0000e+3)) < 0.01)

index2 = i;

break;

end

end

% Generating a rect from index1 to index2

range = index2-index1;

step = [zeros(1, index1) ones(1, range) zeros(1, length(f)-index2)];

step = step.';

**Multiplying input signal by rect to eliminate frequencies other than 4k**

yfilteredFreq = step.\*z;

yfilteredFreqAbs = abs(step.\*z);

figure

plot(f, yfilteredFreqAbs,'g');

xlim([-15000 15000]);

title('Filtered Signal in Frequency Domain');

**Converting the filtered signal to time domain to be modulated**

yfiltered = ifft(fftshift(yfilteredFreq));

figure

plot(t, yfiltered,'r');

title('Filtered Signal in Time Domain');

**Signal after low pass filter**

sound(yfiltered, F);

pause(9);

**Filtered signal resampling**

maxAmplitude = max(yfiltered);

yfiltered = resample(yfiltered, 500000, F);

yfiltered = yfiltered.';

interval = length(yfiltered);

**Frequency Modulation**

**NBFM Signal**

kf = 1e+1\*pi;

t = linspace(0, ty, interval);

deltaF = abs(kf\*max(yfiltered));

m\_int = kf.\*cumsum(yfiltered);

nbfm = 2\*maxAmplitude.\*cos(2.\*pi.\*100000.\*t + m\_int);

figure

plot(t, nbfm,'r');

title('Modulted NBFM Signal in Time Domain');

zmnbfm = abs(fftshift(fft(nbfm)));

fmnbfm = linspace(-500000/2, 500000/2, length(nbfm));

figure

plot(fmnbfm, zmnbfm,'g');

title('Modulated NBFM Signal in Frequency Domain');

**Frequency Demodulation**

nbfm = diff(nbfm);

nbfmEnvelope = abs(hilbert(nbfm));

nbfmEnvelope = resample(nbfmEnvelope, F, 500000);

sound(nbfmEnvelope , F);

**output:**

Chart, histogram

Description automatically generated

Chart, histogram

Description automatically generated

Chart, shape

Description automatically generated

Shape, square

Description automatically generated

Chart, histogram

Description automatically generated

**Experiment 3 Conclusions**

🡪When the instantaneous frequency increases the value of the

phase deviation increases.

🡪To achieve NBFM modulation index, frequency deviation, phase

deviation and frequency deviation constant are very small.