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.';

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

coherent = resample(LSB,F,500000);

coherent = coherent(1:end-1);

coherentFreq = fftshift(fft(coherent));

coherentFilter = step.\*coherentFreq;

coherentTime = ifft(ifftshift(coherentFilter)) ;

sound(real(coherentTime),F);

pause(9);

figure;

plot(T,coherentTime,'r');

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

figure;

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

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

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

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));

envlopeSSBTC=abs(hilbert(SSB\_TC\_time));

%down sample envelop detector

envlopeSSBTC=resample(envlopeSSBTC,F,500000);

figure;

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

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

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

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

figure

plot(fdtc, zdtc,'g');

xlim([-5000 5000]);

ylim([0 300]);

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

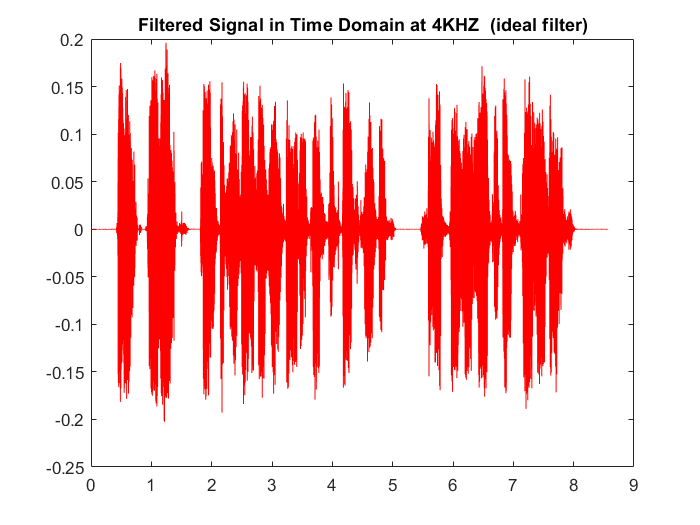
sound(real(envlopeSSBTC),F);

Chart

Description automatically generated

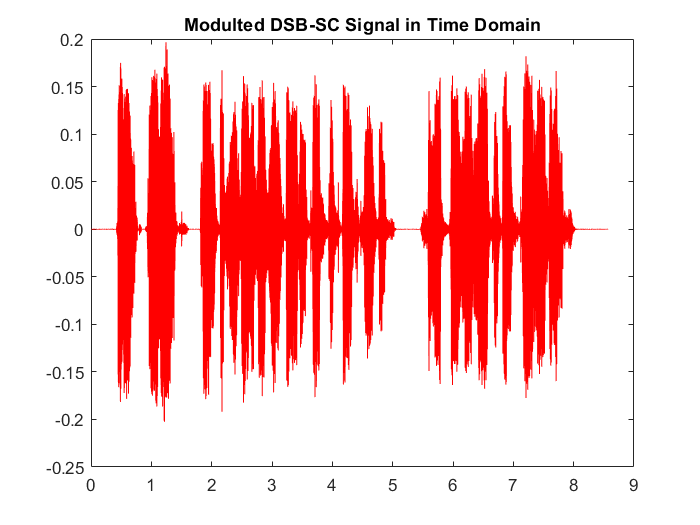
Chart, histogram

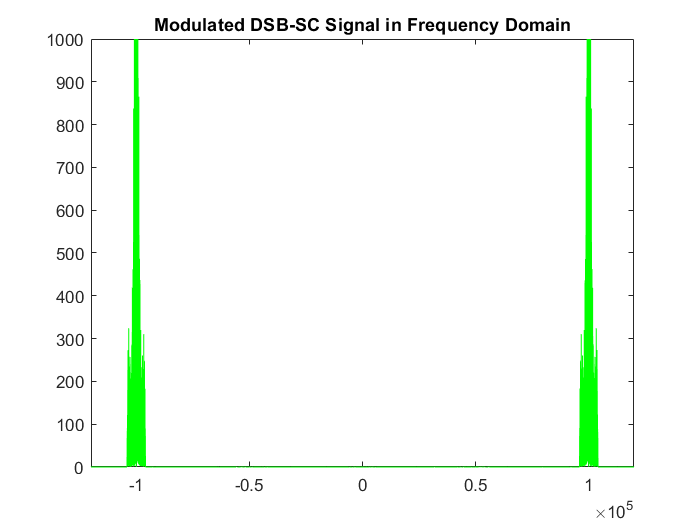
Description automatically generated



Chart, histogram

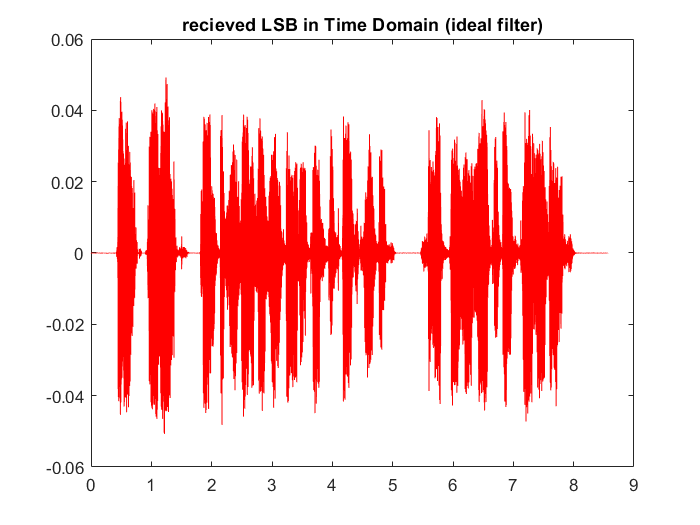
Description automatically generated





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, histogram

Description automatically generated

Chart, histogram

Description automatically generated

Chart, histogram

Description automatically generated