Analog Communications Matlab Assignment

Team Members

Nourhan Waleed	6609
Nouran Hisham	6532
Rawan Hindawy	6491
Kareem Sabra	6594
Seif Mohamed	6624

EXPERIMENT ONE: DOUBLE SIDEBAND MODULATION:

Double Sideband modulation is the easiest and most direct type of analog modulation. In this scheme, the modulated signal is obtained using a direct multiplication of the modulating signal (the message) by a cosine carrier. This multiplication results in shifting the entire spectrum of the message to a center frequency defined by the carrier frequency. The modulation is said to be double sideband transmitted carrier (DSB-TC) when the carrier is transmitted along the modulation term. If the carrier term is omitted, the modulation is termed double sideband suppressed carrier (DSB-SC). DSB-TC has a significant advantage in the receiver design (the envelop detector). Also transmitting the carrier independently enables us to extract useful information such as the carrier frequency which can be helpful for carrier synchronization. However, the DSB-TC loses to the other variant (the SC) in terms of power efficiency.

The code (copied from editor):

```
clc;
clear;

%Reading file
[y,fs]=audioread('eric.wav');
len = length(y);
Y=fftshift(fft(y));

%Plot the spectrum & time domain:
t= linspace(0,length(y)/fs,length(y));
Fvec=linspace(-fs/2,fs/2,len);
subplot(2,1,1); plot(t,y);
title('Original Signal in Time Domain');
xlabel('Time');
ylabel('Amplitude');
subplot(2,1,2);plot(Fvec,abs(Y));
title('Spectrum of Original Signal(Frequency domain)');
xlabel('Frequency');
ylabel('Amplitude');

%filtering
fc = 4000;
Y(abs(Fvec)>fc) = 0;
figure;
subplot(2,1,2);plot(Fvec,abs(Y));
title('Spectrum after filter');
xlabel('Frequency');
ylabel('Amplitude');
xlim([-4200 4200])
filtered_signal=real(ifft(ifftshift(Y)));
subplot(2,1,1);plot(t,filtered_signal);
title('Signal in time domain after filter');
xlabel('Time');
ylabel('Amplitude');
%sound(filtered_signal,fs);
pause(3);

%modulation
fc=100000;
mod=0.5;
carrierFS=5*fc;
```

```
resampled_signal=resample(filtered_signal,carrierFS,fs);
time = linspace(0,(length(resampled_signal)/carrierFS), length(resampled_signal));
time=time';
carrier=cos(2*pi*fc*time);
DSBSC= carrier.*resampled_signal;
resamplen=resampled_signal/max(abs(resampled_signal));
DC = 2 * max(abs(resampled_signal));
DSBTC= (DC+resampled_signal).*carrier;
%DSBTC= DC*(1 + mod*resamplen).*carrier   the 2,0.5,max cancel eachother out

%plotting time domain
figure;
subplot(2,1,1);
plot(time,DSBSC)
title ('DSB-SC: modulated  resampled filtered signal')
xlabel('Time');
ylabel('Amplitude');
subplot(2,1,2);
plot(time,DSBTC);
title ('DSB-TC modulated resampled filtered signal time domain');
xlabel('Time');
ylabel('Amplitude');

%plotting frequency domain
freq_DSBSC=(fftshift(fft(DSBSC))/carrierFS);
Fvec=linspace(-carrierFS/2,carrierFS/2,length(freq_DSBSC));
figure;
subplot(2,1,1);
plot(Fvec,abs(freq_DSBSC))
title ('DSB-SC modulated resampled filtered signal freq. domain');
xlabel('Frequency');
ylabel('Amplitude');

freq_DSBTC = (fftshift(fft(DSBTC))/carrierFS);
Fvec=linspace(-carrierFS/2,carrierFS/2,length(freq_DSBTC));
subplot(2,1,2);
plot(Fvec,abs(freq_DSBTC))
title ('DSB-TC modulated resampled filtered signal freq. domain');

DSBSC_envelope = abs(hilbert(DSBSC));
DSBTC_envelope = abs(hilbert(DSBTC));

figure;
subplot(2,1,1);
plot(time,DSBSC_envelope);
title('DSB-SC envelope detector');
xlabel('Time');
ylabel('Amplitude');
subplot(2,1,2);
plot(time,DSBTC_envelope);
title('DSB-TC envelope detector');
xlabel('Time');
ylabel('Amplitude');
resampledMessageDSBTC = resample(DSBTC_envelope, fs, carrierFS);
resampledMessageDSBSC = resample(DSBSC_envelope, fs, carrierFS);
%sound(resampledMessageDSBTC,fs);
pause(.5);
%sound(resampledMessageDSBSC,fs);
pause(.5);
```

```
%observation: DSBTCmodulated signal is clearer, envelope can be used with DSB_TC
%demodulation and coherent detector dsbtc
fs = 1000000;

SNR1 = 0;
SNR2 = 10;
SNR3 = 30;
ytc1 = awgn(DSBTC,SNR1);
ytc2 = awgn(DSBTC,SNR2);
ytc3 = awgn(DSBTC,SNR3);
envelopeSNR1 = abs(hilbert(ytc1));
envelopeSNR2 = abs(hilbert(ytc2));
envelopeSNR3 = abs(hilbert(ytc3));
figure;

%plotting time domain
xtc1 = ytc1.*carrier;
subplot(3,1,1);
plot(time,[xtc1,DSBTC_envelope]);
title('DSBTC - SNR = 0 db in time domain');
xlabel('Time');
ylabel('Amplitude');
xtc2 = ytc2.*carrier;
subplot(3,1,2);
plot(time,[xtc2,DSBTC_envelope]);
title('DSBTC - SNR = 10 db in time domain');
xlabel('Time');
ylabel('Amplitude');
xtc3 = ytc3.*carrier;
subplot(3,1,3);
plot(time,[xtc3,DSBTC_envelope]);
title('DSBTC - SNR = 30 db in time domain');
xlabel('Time');
ylabel('Amplitude');

%plotting frequency domain
figure;
subplot(3,1,1);
freq_xtc1=fftshift(fft(xtc1));
plot(Fvec,[abs(freq_DSBTC),abs(freq_xtc1)]);
title('DSBTC - SNR = 0 db in frequency domain');
xlabel('Frequency');
ylabel('Amplitude');
subplot(3,1,2);
freq_xtc2=fftshift(fft(xtc2));
plot(Fvec,[abs(freq_DSBTC),abs(freq_xtc2)]);
title('DSBTC - SNR = 10 db in frequency domain');
xlabel('Frequency');
ylabel('Amplitude');
subplot(3,1,3);
freq_xtc3=fftshift(fft(xtc3));
plot(Fvec,[abs(freq_DSBTC),abs(freq_xtc3)]);
title('DSBTC - SNR = 30 db in frequency domain');
xlabel('Frequency');
ylabel('Amplitude');

%soundsc(envelopeSNR1);
pause(.5);
clear sound;
%soundsc(envelopeSNR2);
```

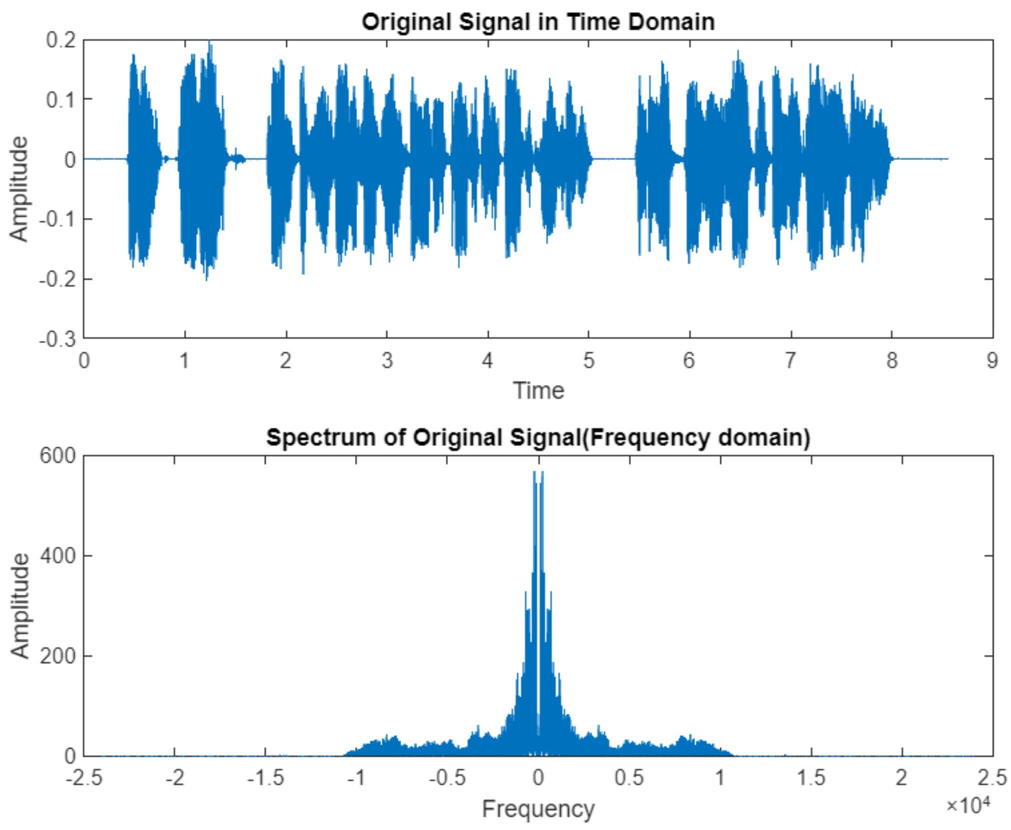
```matlab
pause(.7);
clear sound;
%soundsc(envelopeSNR3);
pause(.9);
clear sound;

% demodulation and coherent detector dsbsc
fs = 1000000;
y1 = awgn(DSBSC,SNR1);
y2 = awgn(DSBSC,SNR2);
y3 = awgn(DSBSC,SNR3);

x1 = y1.*carrier;
[b,a] = butter(5,fc/(fs/2));
x1 = filtfilt(b,a,x1);
subplot(3,1,1);
plot(time,[x1,DSBSC_envelope]);
title('DSBSC - SNR = 0 db in time domain');
xlabel('Time');
ylabel('Amplitude');

x2 = y2.*carrier;
[b,a] = butter(5,fc/(fs/2));
x2 = filtfilt(b,a,x2);
subplot(3,1,2);
plot(time,[x2,DSBSC_envelope]);
title('DSBSC - SNR = 10 db in time domain');
xlabel('Time');
ylabel('Amplitude');

x3 = y3.*carrier;
[b,a] = butter(5,fc/(fs/2));
x3 = filtfilt(b,a,x3);
subplot(3,1,3);
plot(time,[x3,DSBSC_envelope]);
title('DSBSC - SNR = 30 db in time domain');
xlabel('Time');
ylabel('Amplitude');

%soundsc(real(double(x1)));
pause(.5);
clear sound;
%soundsc(real(double(x2)));
pause(.7);
clear sound;
%soundsc(real(double(x3)));
pause(.9);
clear sound;

figure;
subplot(3,1,1);
freq_x1=fftshift(fft(x1));
plot(Fvec,[abs(freq_DSBSC),abs(freq_x1)]);
title('DSBSC - SNR = 0 db in frequency domain');
xlabel('Frequency');
ylabel('Amplitude');
subplot(3,1,2);
freq_x2=fftshift(fft(x2));
plot(Fvec,[abs(freq_DSBSC),abs(freq_x2)]);
```

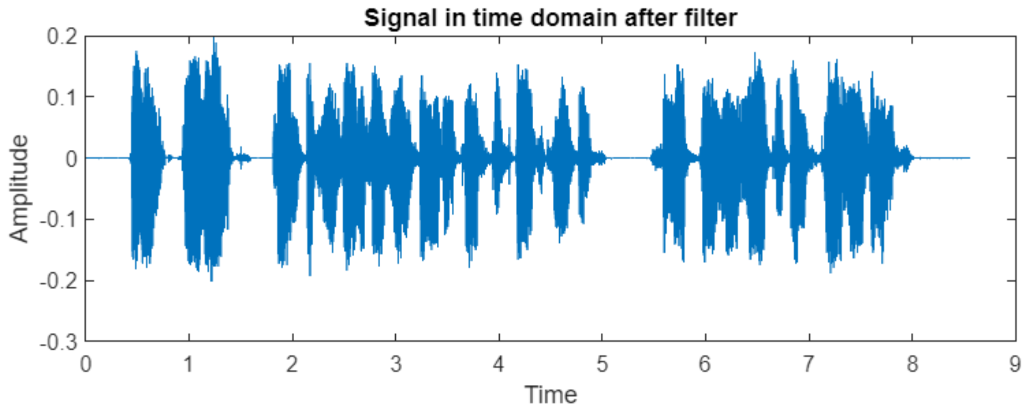
```
title('DSBSC - SNR = 10 db in frequency domain');
xlabel('Frequency');
ylabel('Amplitude');
subplot(3,1,3);
freq_x3=fftshift(fft(x3));
plot(Fvec,[abs(freq_DSBSC),abs(freq_x3)]);
title('DSBSC - SNR = 30 db in frequency domain');
xlabel('Frequency');
ylabel('Amplitude');

%frequency error
fc=100100;
a1 = y1.*cos(2*pi*fc*time);
a2 = y2.*cos(2*pi*fc*time);
a3 = y3.*cos(2*pi*fc*time);
figure;
[b,a] = butter(5,fc/(fs/2));
a1 = filtfilt(b,a,a1);
F_error_0dB=immse(x1,a1);
%frequency error plotting in time domain
subplot(3,1,1);
plot(time,[x1,a1]);
title('SNR = 0 db in time domain with frequency error');
xlabel('Time');
ylabel('Amplitude');

[b,a] = butter(5,fc/(fs/2));
a2 = filtfilt(b,a,a2);
F_error_10dB=immse(x2,a2);
subplot(3,1,2);
plot(time,[x2,a2]);
title('SNR = 10 db in time domain with frequency error');
xlabel('Time');
ylabel('Amplitude');

subplot(3,1,3);
[b,a] = butter(5,fc/(fs/2));
a3 = filtfilt(b,a,a3);
 F_error_30dB=immse(x3,a3);
plot(time,[x3,a3]);
title('SNR = 30 db in time domain with frequency error');
xlabel('Time');
ylabel('Amplitude');

%sound
%soundsc(real(double(a1)));
pause(.5);
clear sound;
%soundsc(real(double(a2)));
pause(.7);
clear sound;
%soundsc(real(double(a3)));
pause(.9);
clear sound;

figure;
subplot(3,1,1);
freq_a1=fftshift(fft(a1));
plot(Fvec,[abs(freq_x1),abs(freq_a1)]);
```

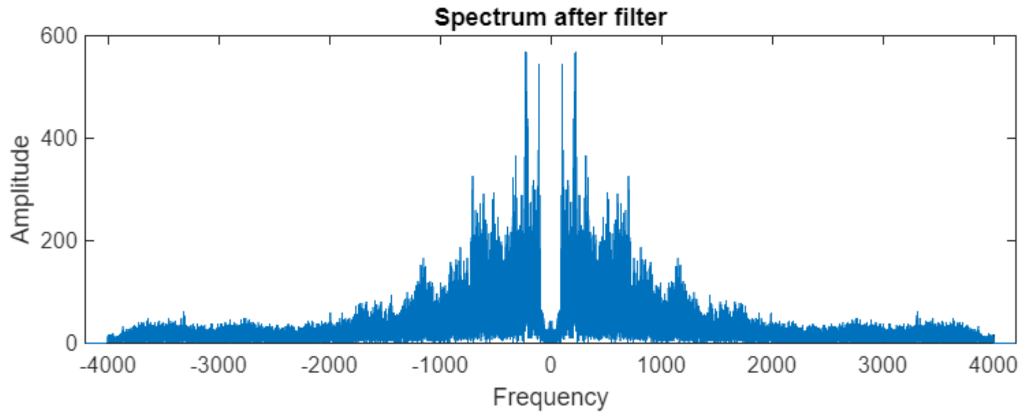
```
title('SNR = 0 db in frequency domain with frequency error');
xlabel('Frequency');
ylabel('Amplitude');
subplot(3,1,2);
freq_a2=fftshift(fft(x2));
plot(Fvec,[abs(freq_x2),abs(freq_a2)]);
title('SNR = 10 db in frequency domain with frequency error');
xlabel('Frequency');
ylabel('Amplitude');
subplot(3,1,3);
freq_a3=fftshift(fft(a3));
plot(Fvec,[abs(freq_x3),abs(freq_a3)]);
title('SNR = 30 db in frequency domain with frequency error');
xlabel('Frequency');
ylabel('Amplitude');

%phase error
fc=100000;
figure;
a1 = y1.*cos(2*pi*fc*time+20);
[b,a] = butter(5,fc/(fs/2));
a1 = filtfilt(b,a,a1);
F_error_0dB=immse(x1,a1);
subplot(3,1,1);
plot(time,[x1,a1]);
title('SNR = 0 db in time domain with phase error');
xlabel('Time');
ylabel('Amplitude');

a2 = y2.*cos(2*pi*fc*time+20);
[b,a] = butter(5,fc/(fs/2));
a2 = filtfilt(b,a,a2);
 F_error_10dB=immse(x2,a2);
subplot(3,1,2);
plot(time,[x2,a2]);
title('SNR = 10 db in time domain with phase error');
xlabel('Time');
ylabel('Amplitude');

a3 = y3.*cos(2*pi*fc*time+20);
[b,a] = butter(5,fc/(fs/2));
a3 = filtfilt(b,a,a3);
F_error_30dB=immse(x3,a3);
subplot(3,1,3);
plot(time,[x3,a3]);
title('SNR = 30 db in time domain with phase error');
xlabel('Time');
ylabel('Amplitude');

%soundsc(real(double(a1)));
pause(.5);
clear sound;
%soundsc(real(double(a2)));
pause(.7);
clear sound;
%soundsc(real(double(a3)));
pause(.9);
clear sound;

figure;
subplot(3,1,1);
```

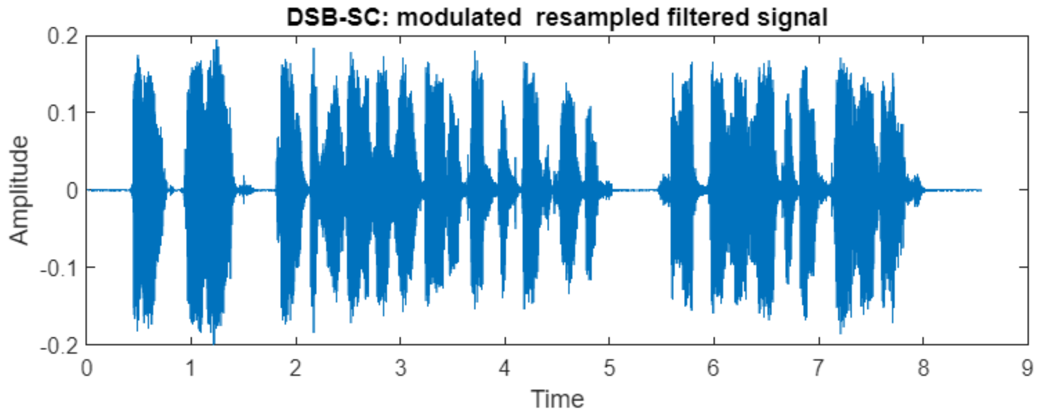
```
freq_a1=fftshift(fft(a1));
plot(Fvec,[abs(freq_x1),abs(freq_a1)]);
title('SNR = 0 db in frequency domain with phase error');
xlabel('Frequency');
ylabel('Amplitude');
subplot(3,1,2);
freq_a2=fftshift(fft(x2));
plot(Fvec,[abs(freq_x2),abs(freq_a2)]);
title('SNR = 10 db in frequency domain with phase error');
xlabel('Frequency');
ylabel('Amplitude');
subplot(3,1,3);
freq_a3=fftshift(fft(a3));
plot(Fvec,[abs(freq_x3),abs(freq_a3)]);
title('SNR = 30 db in frequency domain with phase error');
xlabel('Frequency');
ylabel('Amplitude');
```

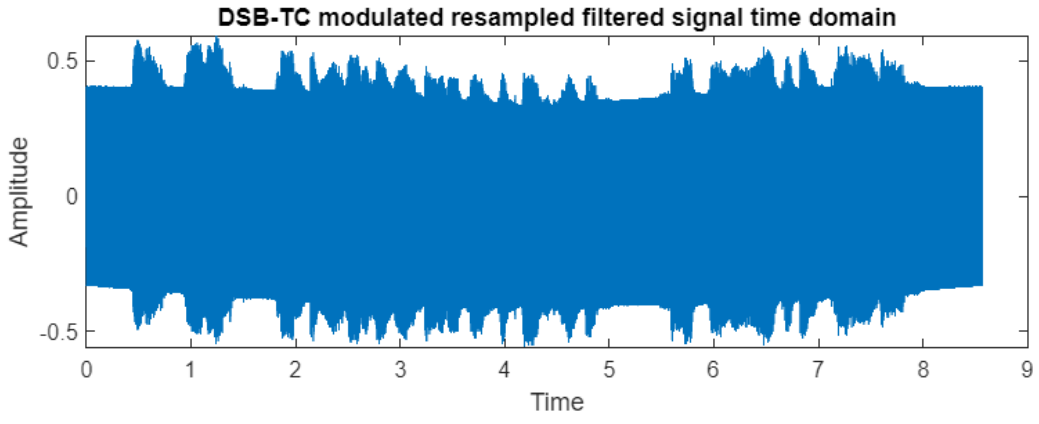
Outputs:

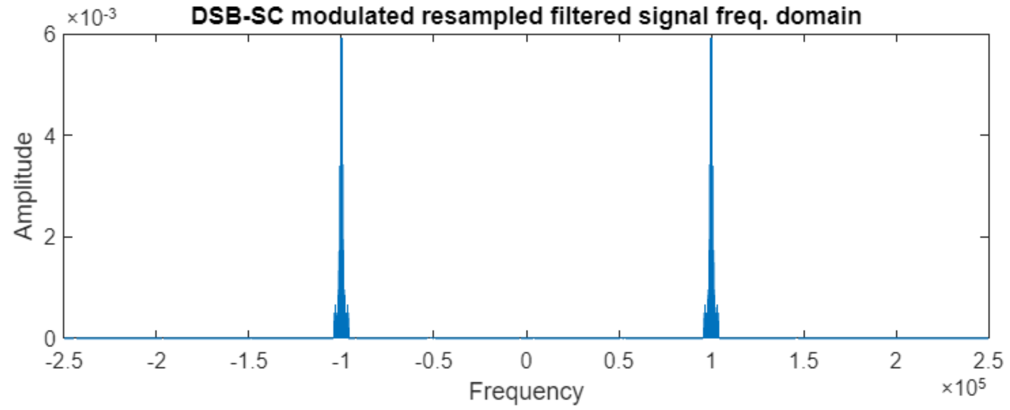


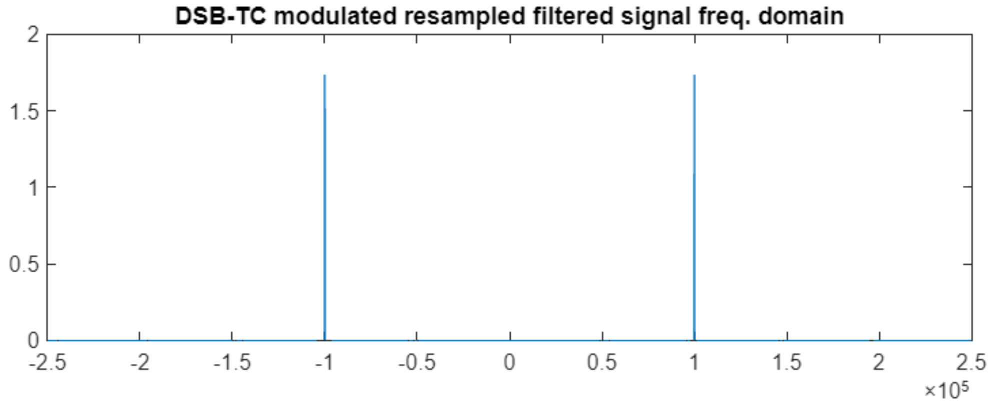

Signal in time domain after filter
Amplitude
Time
0.2
0.1
0
-0.1
-0.2
-0.3
0
1
2
3
4
5
6
7
8
9

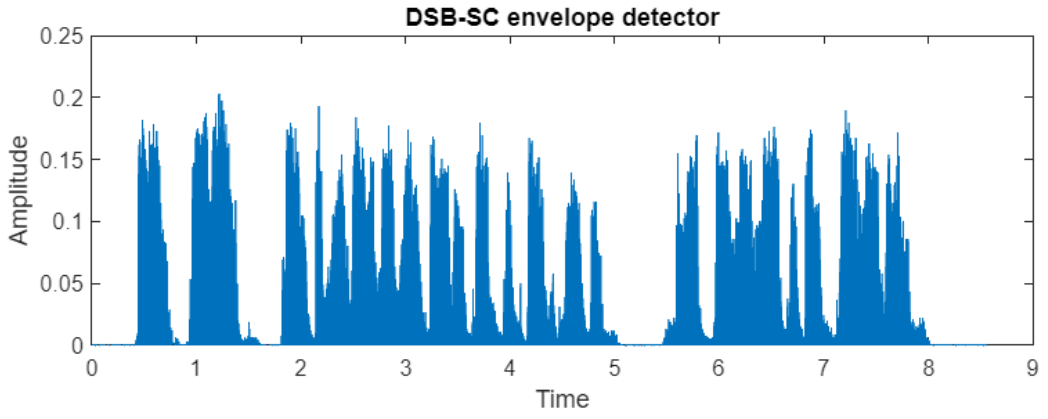

Spectrum after filter
Amplitude
Frequency
600
400
200
0
-4000
-3000
-2000
-1000
0
1000
2000
3000
4000

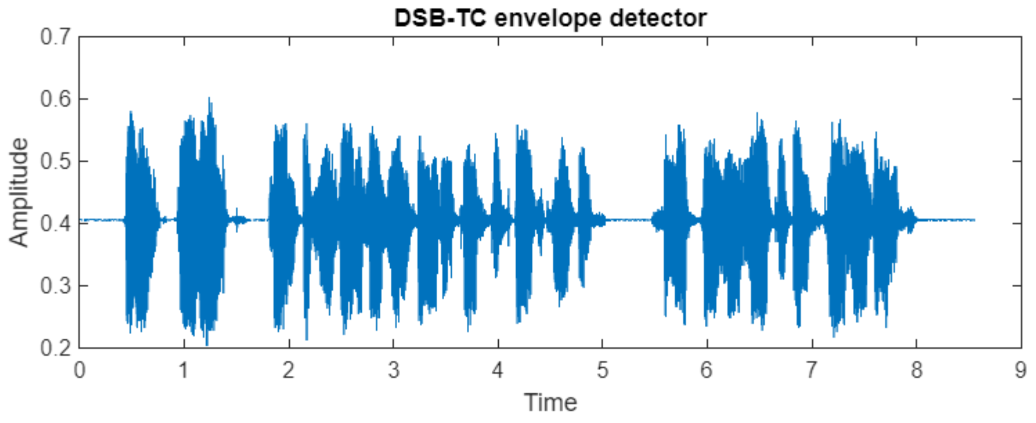

DSB-SC: modulated resampled filtered signal
Amplitude
Time
0.2
0.1
0
-0.1
-0.2
0
1
2
3
4
5
6
7
8
9

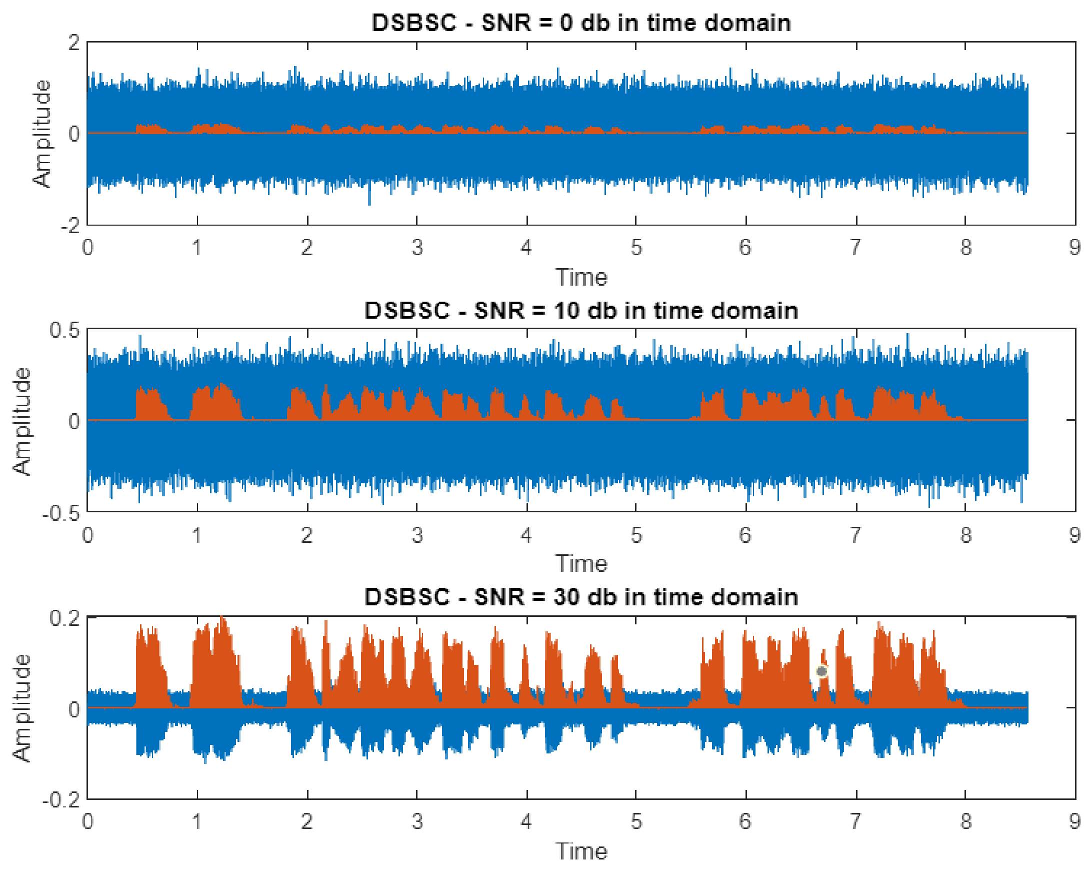

DSB-TC modulated resampled filtered signal time domain
Amplitude
Time
0.5
0
-0.5
0
1
2
3
4
5
6
7
8
9

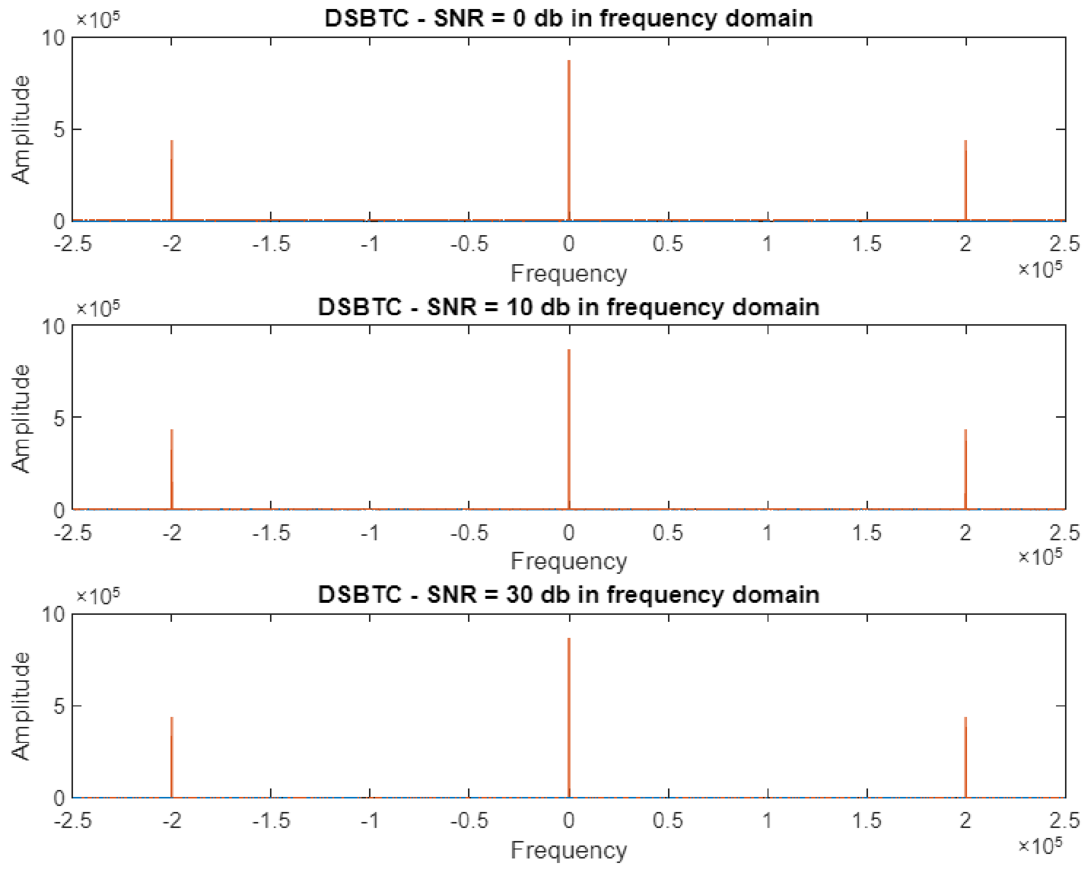

DSB-SC modulated resampled filtered signal freq. domain
×10-3
Amplitude
Frequency
×10^5
-2.5 -2 -1.5 -1 -0.5 0 0.5 1 1.5 2 2.5
0 2 4 6

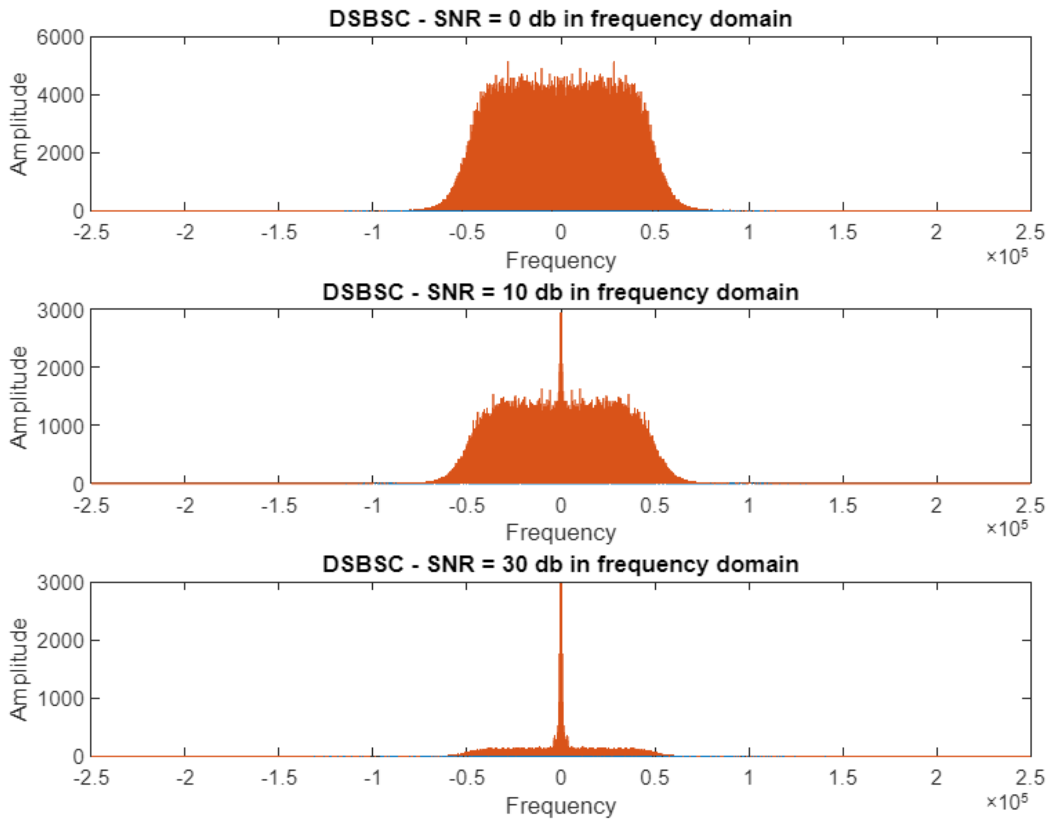

DSB-TC modulated resampled filtered signal freq. domain
×10^5
-2.5 -2 -1.5 -1 -0.5 0 0.5 1 1.5 2 2.5
0 0.5 1 1.5 2

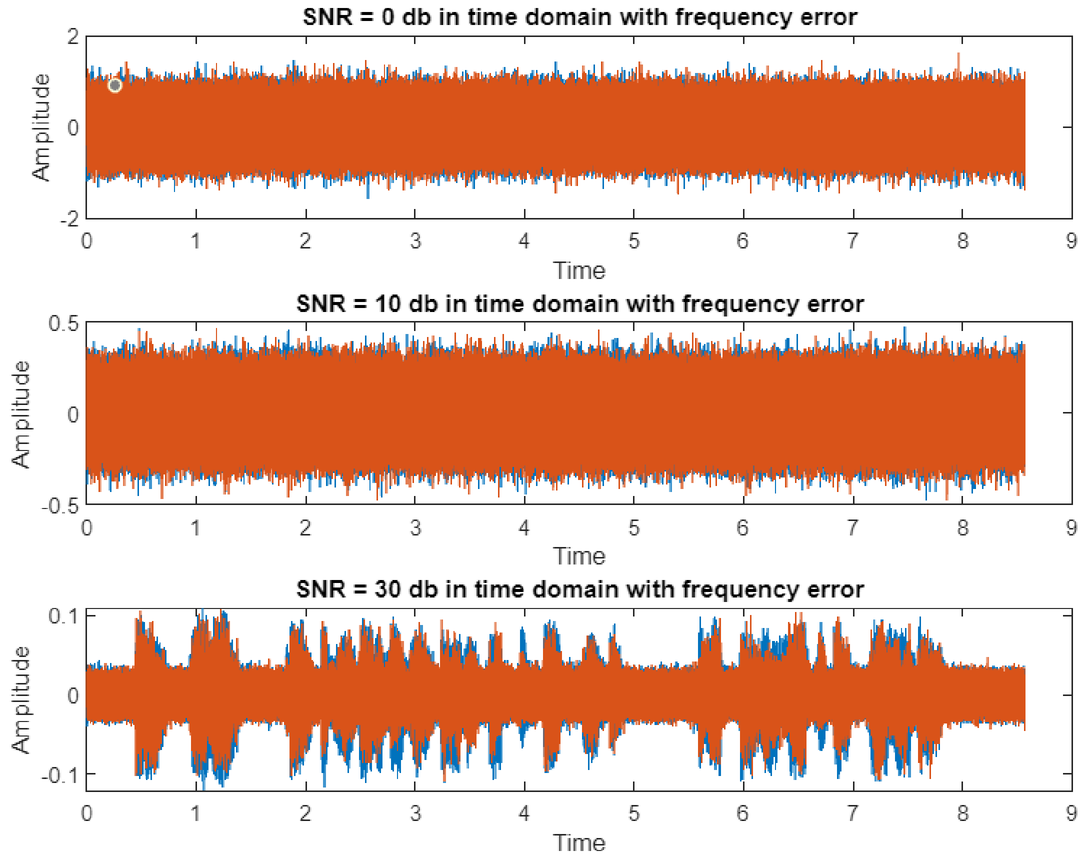

DSB-SC envelope detector
Amplitude
Time
0 1 2 3 4 5 6 7 8 9
0 0.05 0.1 0.15 0.2 0.25

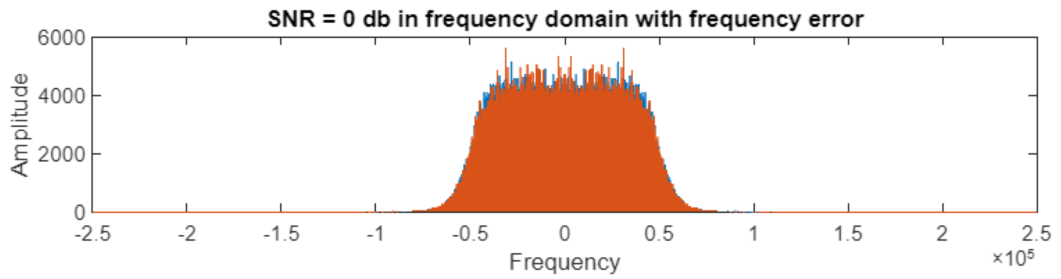

DSB-TC envelope detector
Amplitude
Time
0 1 2 3 4 5 6 7 8 9
0.2 0.3 0.4 0.5 0.6 0.7

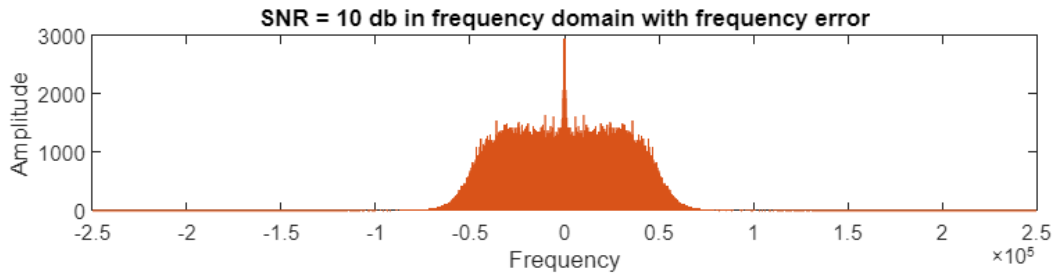

DSBSC - SNR = 0 db in time domain
Amplitude
Time
DSBSC - SNR = 10 db in time domain
Amplitude
Time
DSBSC - SNR = 30 db in time domain
Amplitude
Time

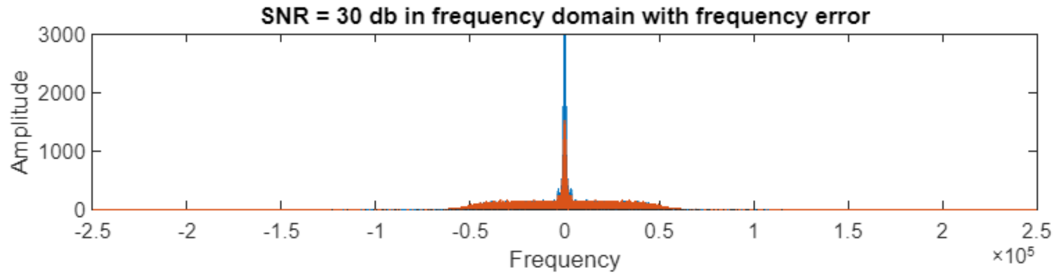

DSBTC - SNR = 0 db in frequency domain
×10⁵
Amplitude
Frequency
×10⁵
DSBTC - SNR = 10 db in frequency domain
Amplitude
Frequency
DSBTC - SNR = 30 db in frequency domain
Amplitude
Frequency

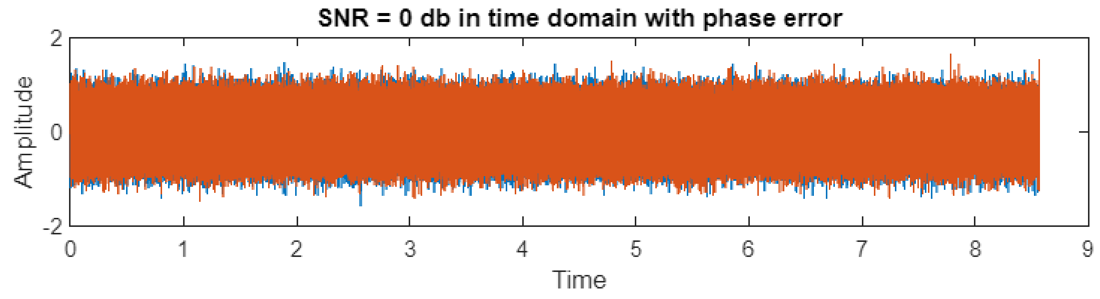

DSBSC - SNR = 0 db in frequency domain
DSBSC - SNR = 10 db in frequency domain
DSBSC - SNR = 30 db in frequency domain
Amplitude
Frequency
×10^5

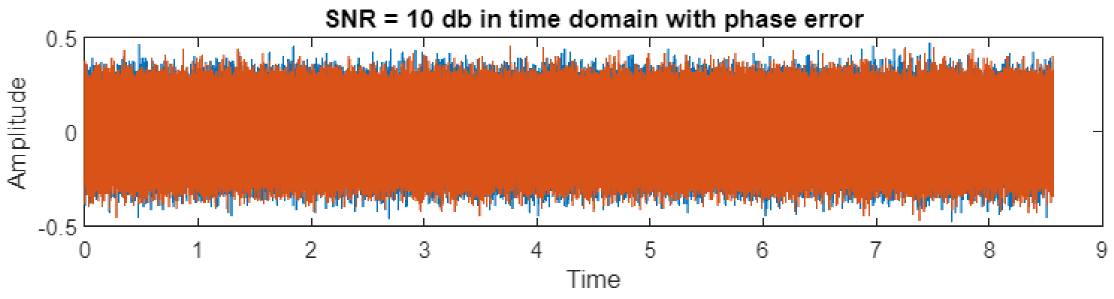

SNR = 0 db in time domain with frequency error
SNR = 10 db in time domain with frequency error
SNR = 30 db in time domain with frequency error
Amplitude
Time

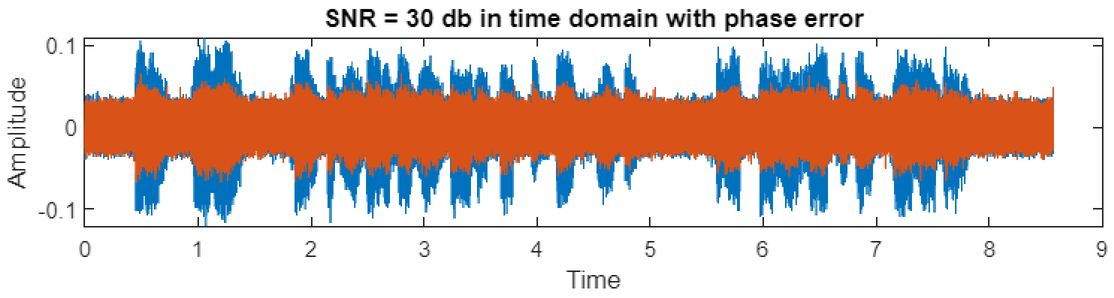

SNR = 0 db in frequency domain with frequency error
Amplitude
Frequency
×10^5

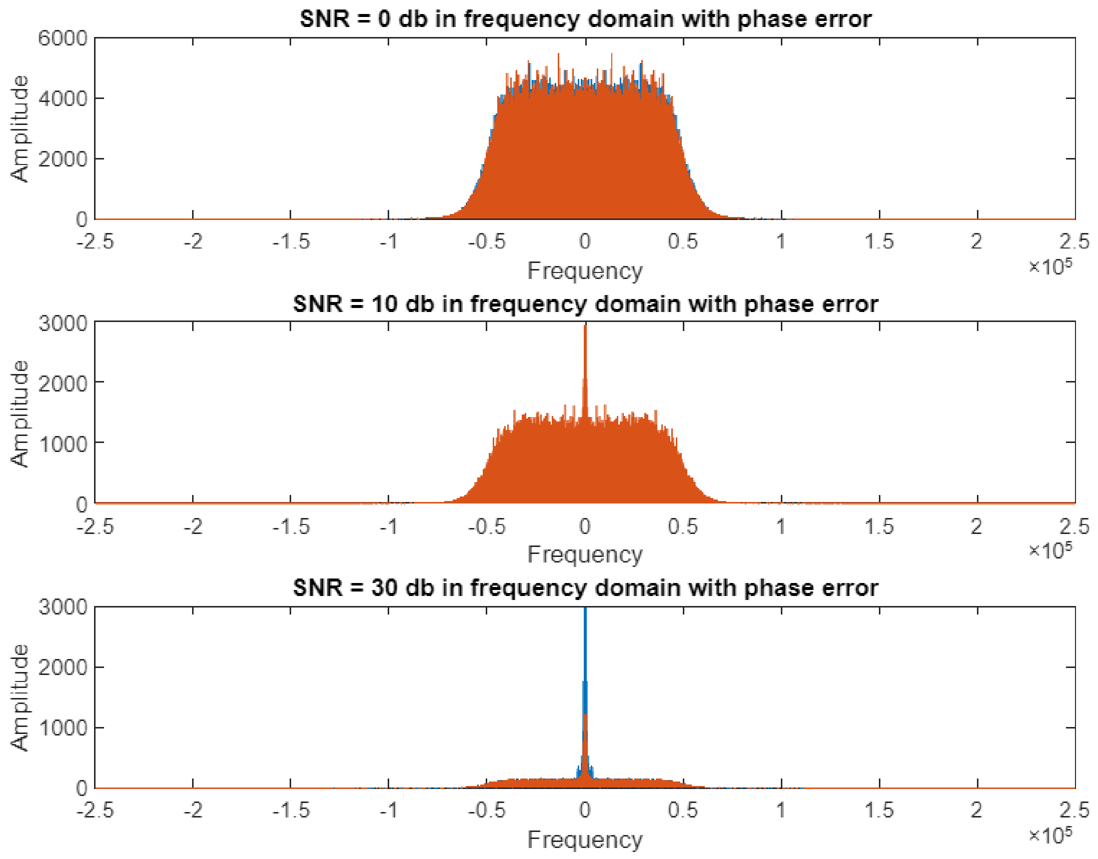

SNR = 10 db in frequency domain with frequency error
Amplitude
Frequency
×10^5


SNR = 30 db in frequency domain with frequency error
Amplitude
Frequency
×10^5


SNR = 0 db in time domain with phase error
Amplitude
Time


SNR = 10 db in time domain with phase error
Amplitude
Time


SNR = 30 db in time domain with phase error
Amplitude
Time



Answering questions:

9) Do you have a name for this phenomenon?

- The name of this phenomenon is Doppler effect.

EXPERIMENT TWO: SINGLE SIDEBAND MODULATION:

The bandwidth inefficiency stemming from the DSB transmission was the main reason why the single sideband (SSB) was developed. In SSB modulation, the bandwidth required for band pass transmission is equal to the bandwidth of that of the baseband. In other words, the band requirement for SSB is halved with respect to that of DSB which requires twice the baseband bandwidth. This reduction in transmission bandwidth is possible since the sideband are replicated twice over the positive and negative frequencies. However, the bandwidth reduction doesn’t come entirely free. In fact, SSB suffers from several disadvantages that we hope to cover in the following simulation.

The code (copied from editor):

```
clc;
clear;

[audio, Fs] = audioread('eric.wav');
s = length(audio) / Fs;
t = linspace(0, s, s*Fs + 1);
fs = linspace(-Fs/2, Fs/2, s*Fs + 1);
```

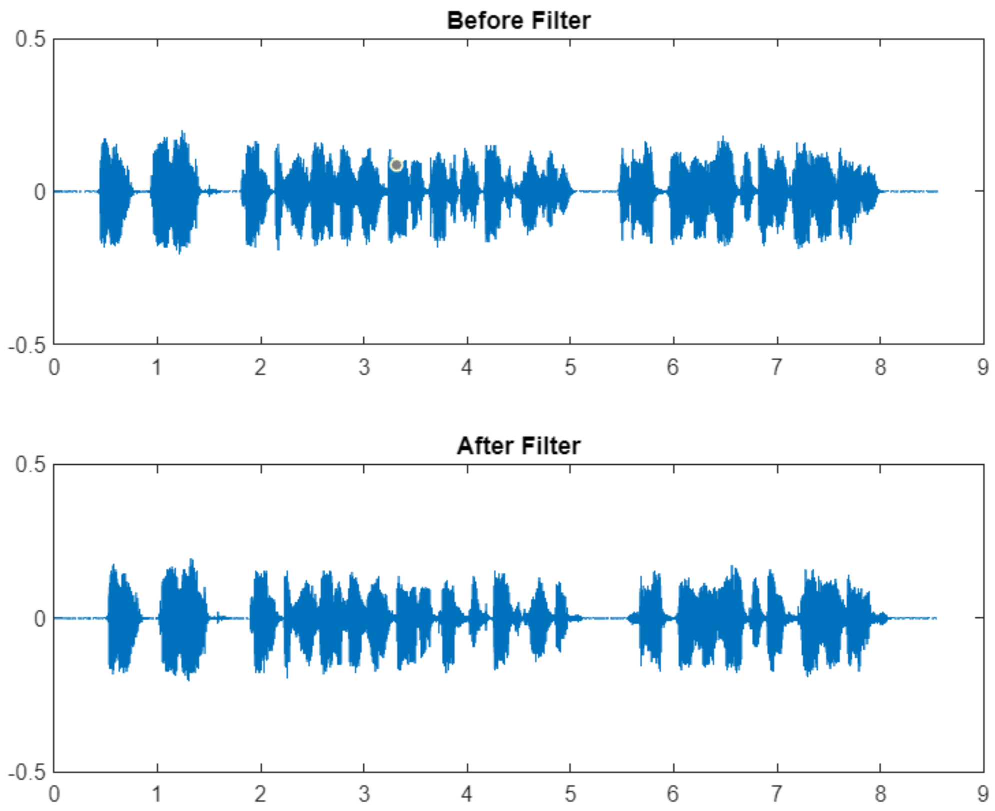
```
% Filtering
d = designfilt('lowpassfir', 'FilterOrder', 8000, 'CutoffFrequency', 4e3, 'SampleRate',
Fs);
filteredSig = filter(d, audio);

%sound(filteredSig, Fs);

figure;
subplot(2, 1, 1)
plot(t, audio);
ylim([-0.5, 0.5]);
title('Before Filter');
subplot(2, 1, 2);
plot(t, filteredSig);
ylim([-0.5, 0.5]);
title('After Filter');

figure;
subplot(2, 1, 1);
plot(fs, real(fftshift(fft(audio))));
xlim([-1e4, 1e4]);
ylim([-500, 500]);
title('Before Filter');
subplot(2, 1, 2);
plot(fs, real(fftshift(fft(filteredSig))));
xlim([-0.5e4, 0.5e4]);
ylim([-500, 500]);
title('After Filter');

Fc = 100e3;
message = resample(filteredSig, 5 * Fc, Fs); % Upsampling signal to 5 * Fc
Fs = 5*Fc;
s = length(message)/Fs;
t = linspace(0, s, s*Fs);
fs = linspace(-Fs/2, Fs/2, s*Fs);

%sound(message, Fs);

% Single Sideband Suppressed Carrier Modulation

carrier = cos(2*pi*Fc*t);
modulatedSCSig = message .* transpose(carrier);

figure;
subplot(2,1,1);
plot(t, modulatedSCSig);
title('DSB-SC Time Domain')
subplot(2,1,2);
plot(fs, real(fftshift(fft(modulatedSCSig))));
xlim([-1.2e5, 1.2e5]);
ylim([-2.5e3, 2.5e3])
title('DSB-SC Frequency Domain')

d = designfilt('lowpassfir', 'FilterOrder', 8000, 'CutoffFrequency', Fc, 'SampleRate',
Fs);
modulatedSCSig = filter(d, modulatedSCSig);

figure;
```

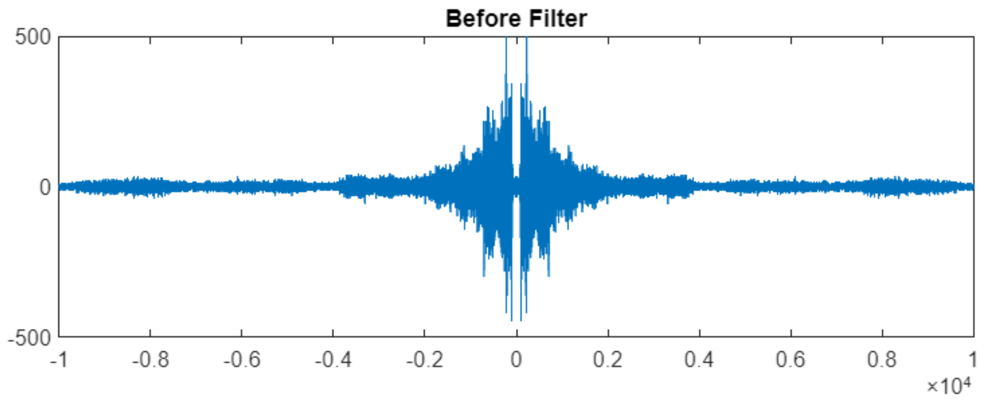
```
subplot(2,1,1);
plot(t, modulatedSCSig);
title('SSB-SC Time Domain')
subplot(2,1,2);
plot(fs, real(fftshift(fft(modulatedSCSig))));
xlim([-1.2e5, 1.2e5]);
ylim([-2.5e3, 2.5e3])
title('SSB-SC Frequency Domain')

% Coherent Detection

carrier = cos(2*pi*Fc*t);
demodSignal = modulatedSCSig .* transpose(carrier);
d = designfilt('lowpassfir', 'FilterOrder', 8000, 'CutoffFrequency', 4e3, 'SampleRate', 5
* Fc);
coherentSC  = filter(d, demodSignal);

%sound(coherentSC, Fs);

figure;
subplot(2, 1, 1);
plot(t, coherentSC);
ylim([-0.05, 0.05]);
title('Suppressed Carrier Coherent');
subplot(2, 1, 2);
plot(fs, real(fftshift(fft(coherentSC))));
xlim([-0.5e4, 0.5e4]);
ylim([-1e3, 1e3]);
title('Suppressed Carrier Coherent Spectrum');

% Butterworth Coherent Detection

carrier = cos(2*pi*Fc*t);
demodSignal = modulatedSCSig .* transpose(carrier);
[b, a] = butter(3, Fc/(Fc * 5 / 2));
butteredSig = filter(b, a, demodSignal);
butteredSigmod = filter(b, a, modulatedSCSig);

figure;
subplot(2, 1, 1);
plot(t, butteredSigmod);
title('Suppressed Carrier modulated signal (Butterworth)');
subplot(2, 1, 2);
plot(fs, real(fftshift(fft(butteredSigmod))));
xlim([-1.2e5, 1.2e5]);
ylim([-2.5e3, 2.5e3])
title('Suppressed Carrier modulated signal Spectrum (Butterworth)');

figure;
subplot(2, 1, 1);
plot(t, butteredSig);
title('Suppressed Carrier Coherent (Butterworth)');
subplot(2, 1, 2);
plot(fs, real(fftshift(fft(butteredSig))));
xlim([-0.5e4, 0.5e4]);
ylim([-1e3, 1e3]);
title('Suppressed Carrier Coherent Spectrum (Butterworth)');
```

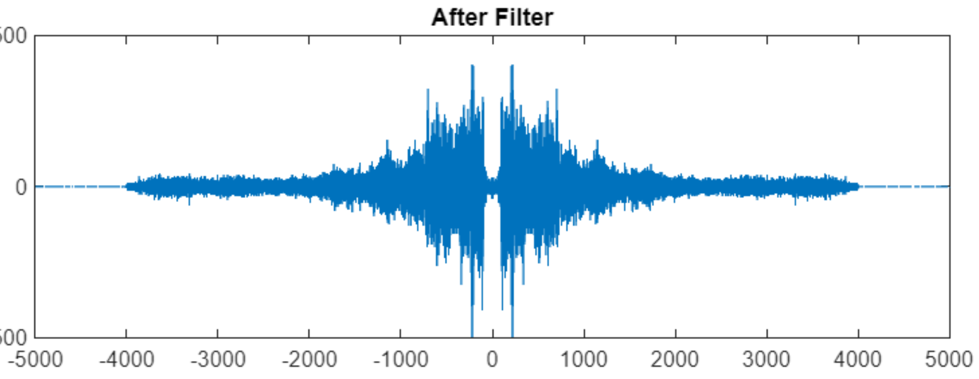
```matlab
% Bad SNR

coherentSC0SNR = awgn(coherentSC, 0);
coherentSC10SNR = awgn(coherentSC, 10);
coherentSC30SNR = awgn(coherentSC, 30);

%sound(coherentSC0SNR, Fs);
%sound(coherentSC10SNR, Fs);
%sound(coherentSC30SNR, Fs);

figure;
subplot(2, 1, 1);
plot(t, coherentSC0SNR);
title('Suppressed Carrier Coherent 0 SNR');
subplot(2, 1, 2);
plot(fs, real(fftshift(fft(coherentSC0SNR))));
xlim([-0.5e4, 0.5e4]);
title('Suppressed Carrier Coherent Spectrum 0 SNR');

figure;
subplot(2, 1, 1);
plot(t, coherentSC10SNR);
title('Suppressed Carrier Coherent 10 SNR');
subplot(2, 1, 2);
plot(fs, real(fftshift(fft(coherentSC10SNR))));
xlim([-0.5e4, 0.5e4]);
title('Suppressed Carrier Coherent Spectrum 10 SNR');

figure;
subplot(2, 1, 1);
plot(t, coherentSC30SNR);
title('Suppressed Carrier Coherent 30 SNR');
subplot(2, 1, 2);
plot(fs, real(fftshift(fft(coherentSC30SNR))));
xlim([-0.5e4, 0.5e4]);
title('Suppressed Carrier Coherent Spectrum 30 SNR');

% Transmitted Carrier Modulation

carrier = cos(2*pi*Fc*t);
message = message + (max(message) * 2);
modulatedTCSig = message .* transpose(carrier);

d = designfilt('lowpassfir', 'FilterOrder', 8000, 'CutoffFrequency', Fc, 'SampleRate',
Fs);
modulatedTCSig = filter(d, modulatedTCSig);

figure;
subplot(2, 1, 1);
plot(t, modulatedTCSig);
ylim([-0.3, 0.3]);
title('DSB-TC Time Domain');
subplot(2, 1, 2);
plot(fs, real(fftshift(fft(modulatedTCSig))));
title('DSB-TC Frequency Domain');
```

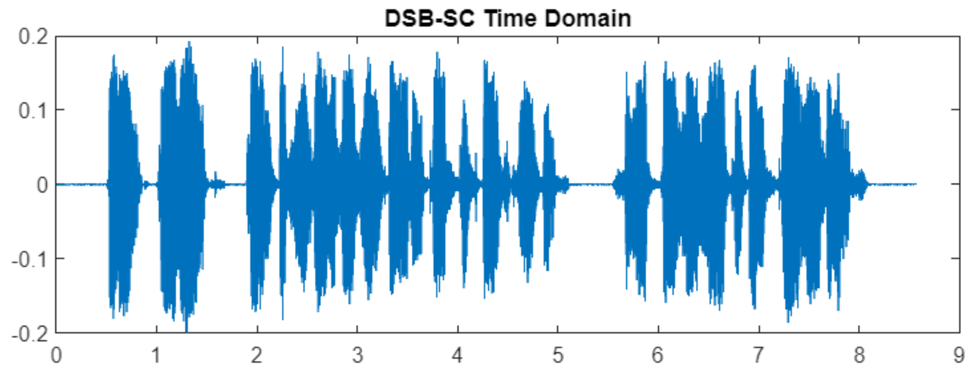
```
% Envelope Detection
envelopeTC = abs(hilbert(modulatedTCSig)); % DSB-TC envelope detection
%sound(envelopeTC, Fs);

figure;
subplot(2, 1, 1);
plot(t, envelopeTC);
xlim([0, 8.5]);
ylim([0.05, 0.3]);
title('Transmitted Carrier Envelope');
subplot(2, 1, 2);
plot(fs, real(fftshift(fft(envelopeTC))));
title('Transmitted Carrier Envelope Spectrum');
```

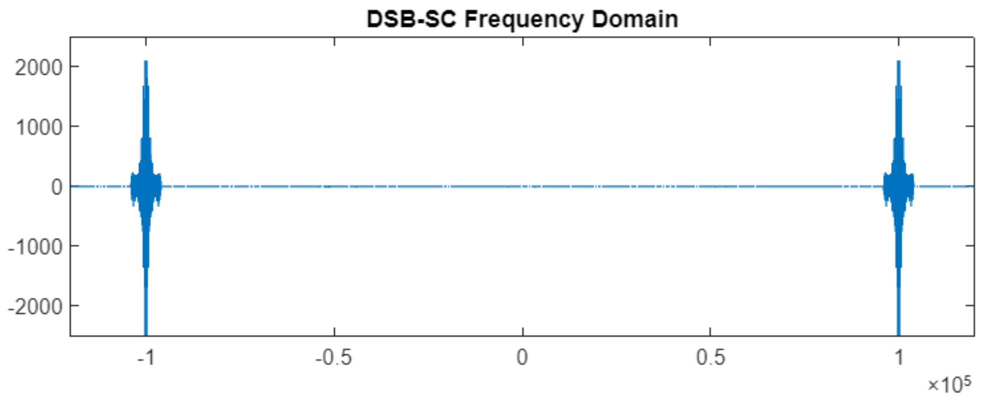
Outputs:

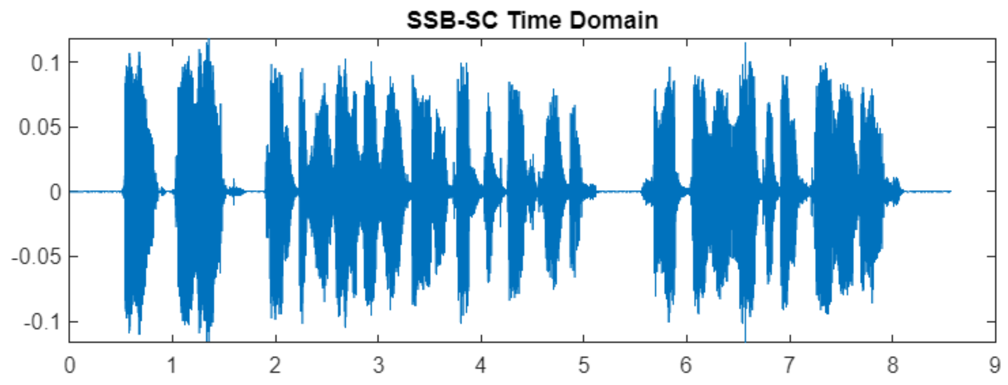


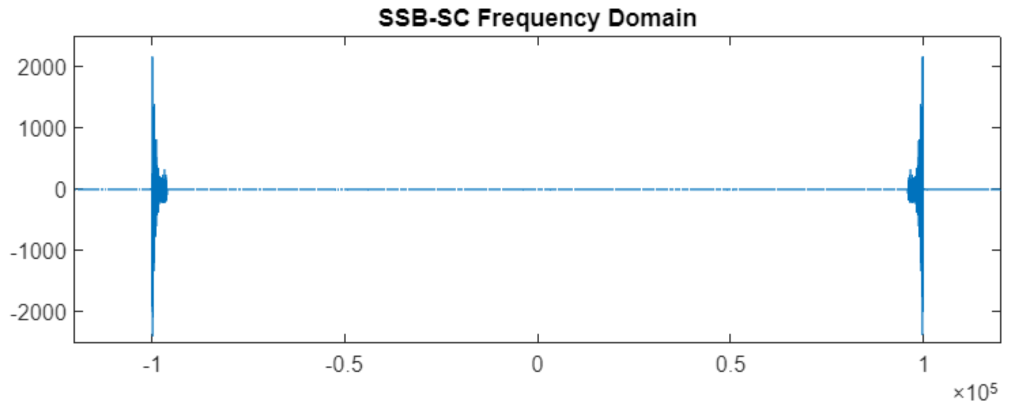

Before Filter
500
0
-500
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1
×10^4

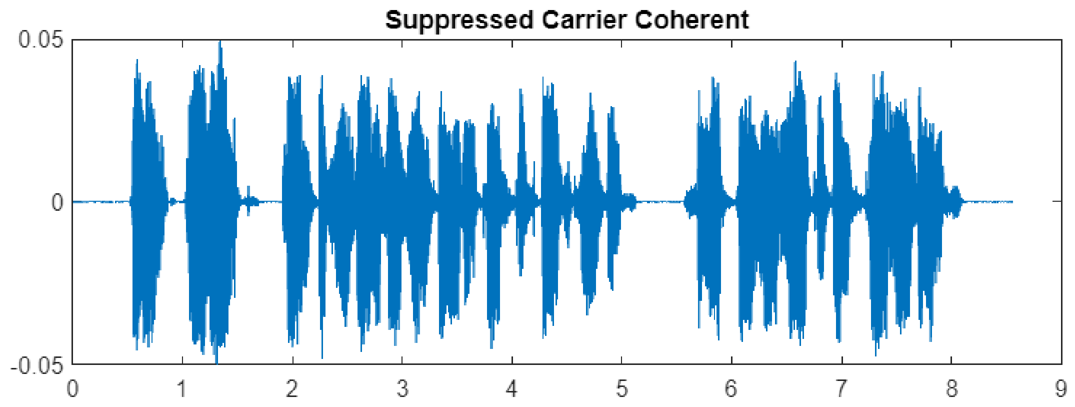

After Filter
500
0
-500
-5000
-4000
-3000
-2000
-1000
0
1000
2000
3000
4000
5000

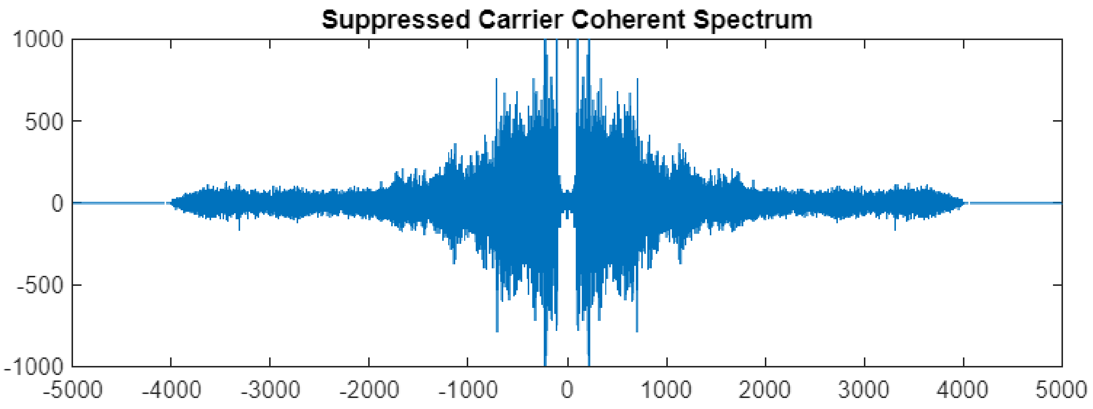

DSB-SC Time Domain
0.2
0.1
0
-0.1
-0.2
0
1
2
3
4
5
6
7
8
9

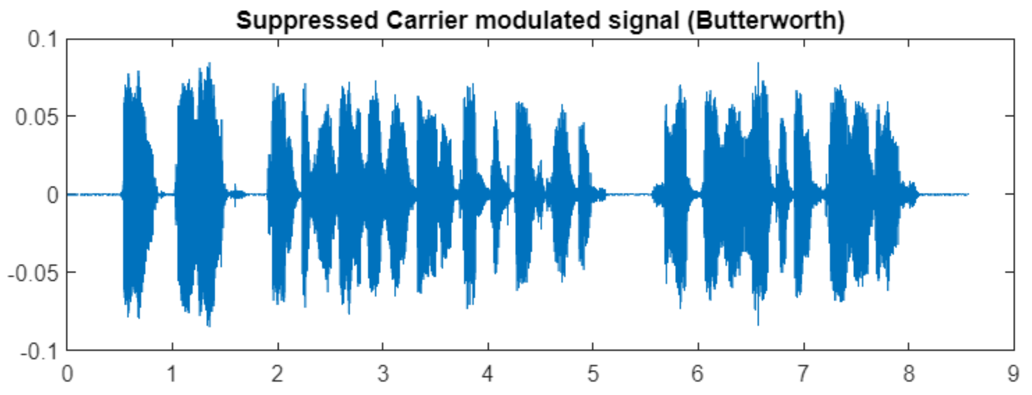

DSB-SC Frequency Domain
2000
1000
0
-1000
-2000
-1
-0.5
0
0.5
1
×10^5

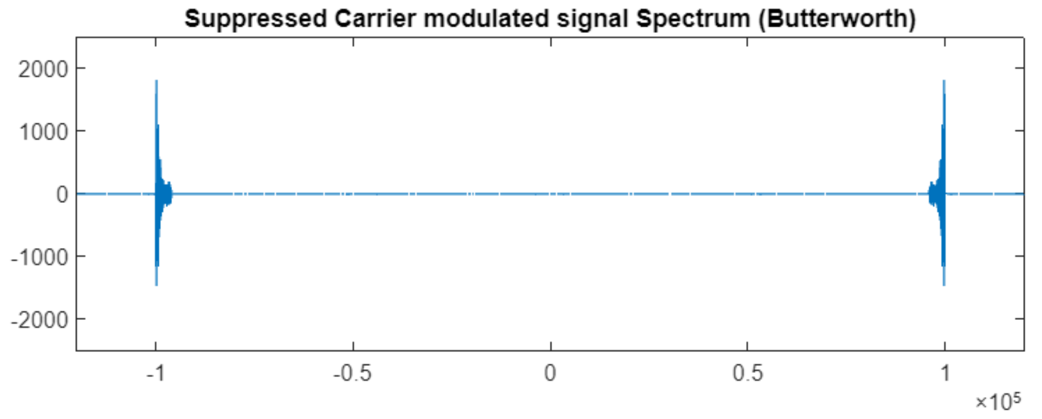

SSB-SC Time Domain
0.1
0.05
0
-0.05
-0.1
0
1
2
3
4
5
6
7
8
9

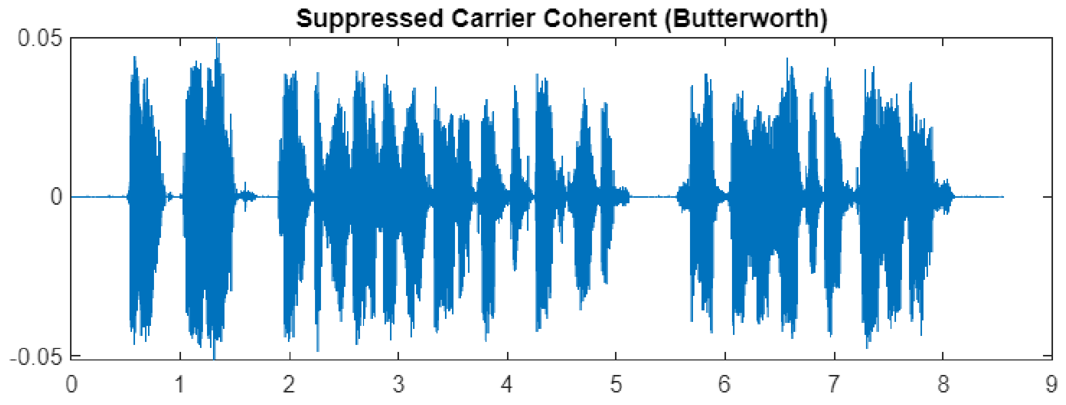

SSB-SC Frequency Domain
2000
1000
0
-1000
-2000
-1
-0.5
0
0.5
1
×10^5

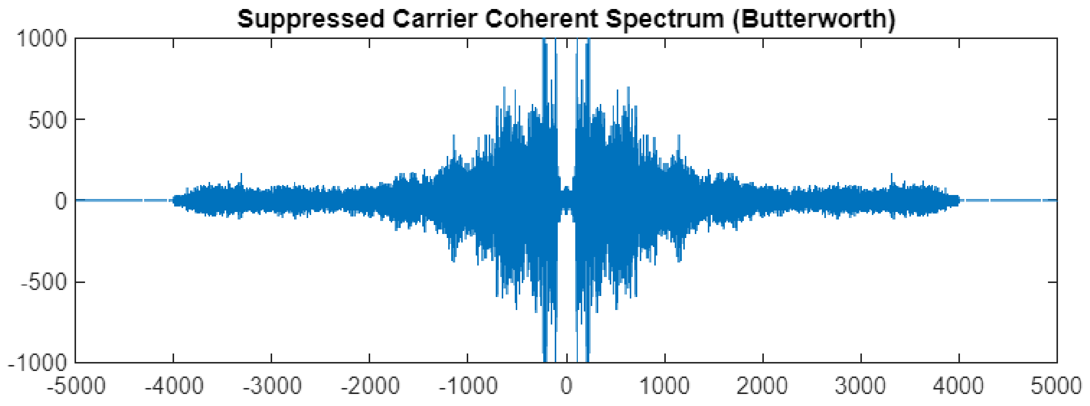

Suppressed Carrier Coherent
0.05
0
-0.05
0
1
2
3
4
5
6
7
8
9

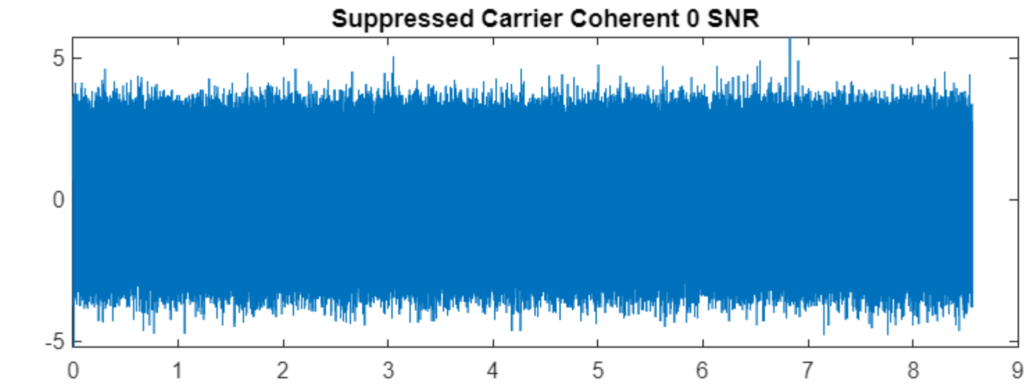

Suppressed Carrier Coherent Spectrum
1000
500
0
-500
-1000
-5000
-4000
-3000
-2000
-1000
0
1000
2000
3000
4000
5000

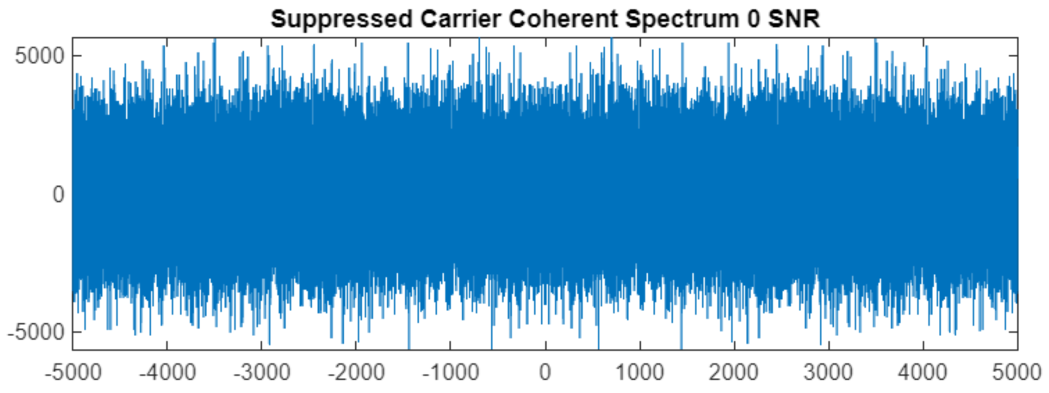

Suppressed Carrier modulated signal (Butterworth)
0.1
0.05
0
-0.05
-0.1
0
1
2
3
4
5
6
7
8
9

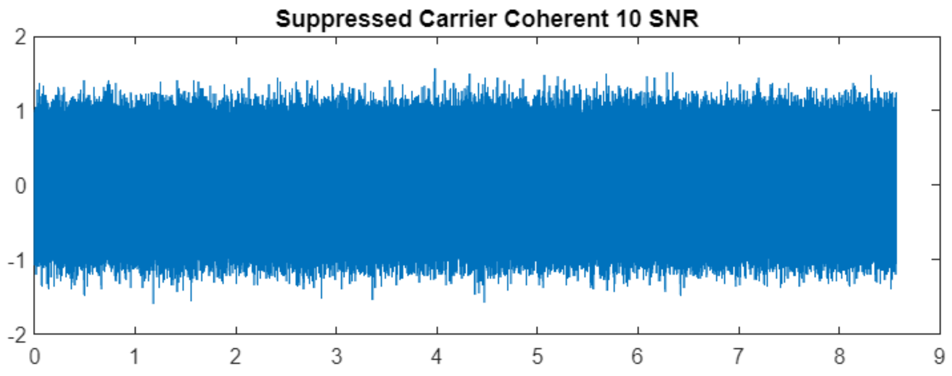

Suppressed Carrier modulated signal Spectrum (Butterworth)
2000
1000
0
-1000
-2000
-1
-0.5
0
0.5
1
×10^5

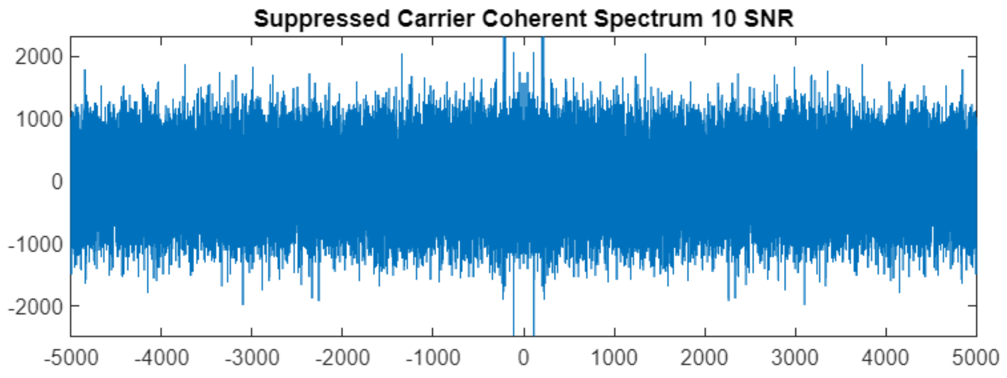

Suppressed Carrier Coherent (Butterworth)
0.05
0
-0.05
0
1
2
3
4
5
6
7
8
9

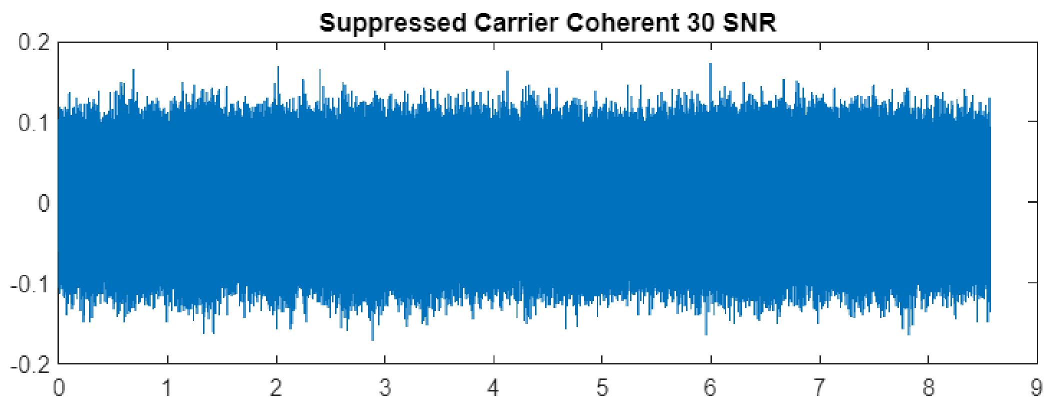

Suppressed Carrier Coherent Spectrum (Butterworth)
1000
500
0
-500
-1000
-5000
-4000
-3000
-2000
-1000
0
1000
2000
3000
4000
5000

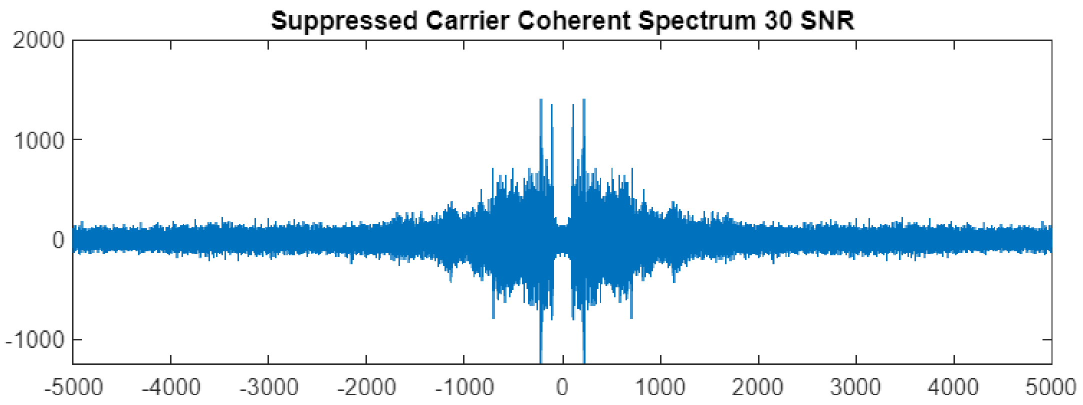

Suppressed Carrier Coherent 0 SNR
5
0
-5
0
1
2
3
4
5
6
7
8
9

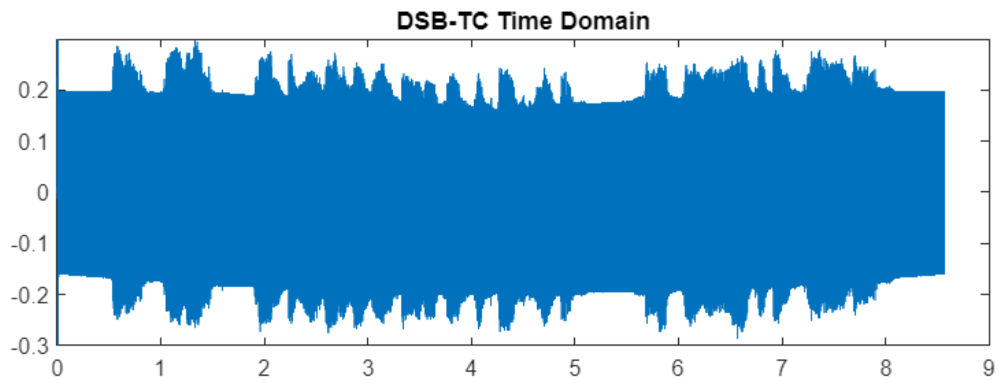

Suppressed Carrier Coherent Spectrum 0 SNR
5000
0
-5000
-5000
-4000
-3000
-2000
-1000
0
1000
2000
3000
4000
5000

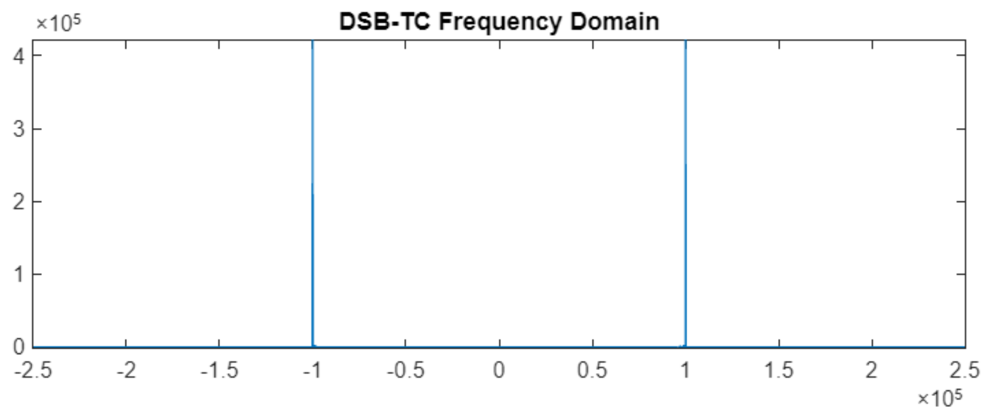

Suppressed Carrier Coherent 10 SNR
2
1
0
-1
-2
0
1
2
3
4
5
6
7
8
9

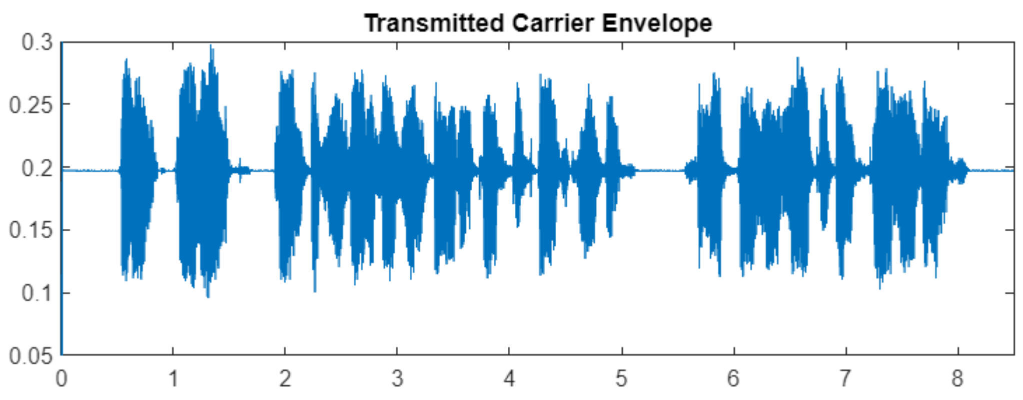

Suppressed Carrier Coherent Spectrum 10 SNR
2000
1000
0
-1000
-2000
-5000
-4000
-3000
-2000
-1000
0
1000
2000
3000
4000
5000

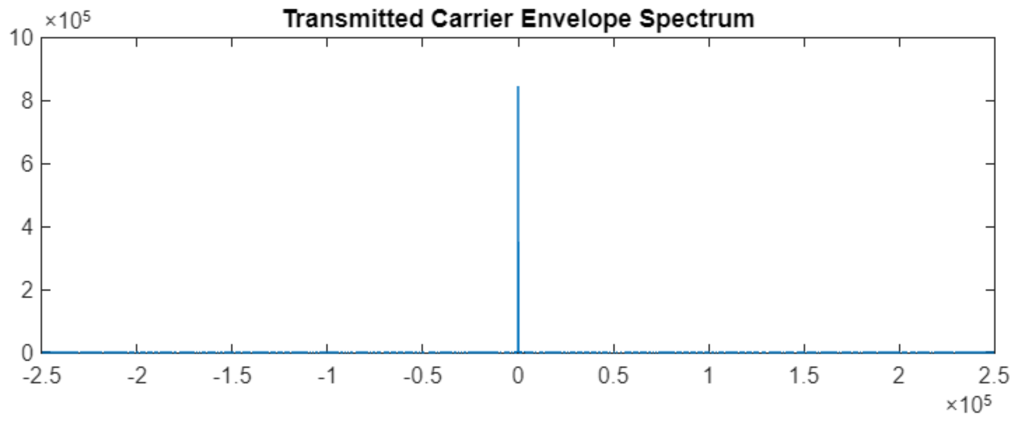

Suppressed Carrier Coherent 30 SNR
0.2
0.1
0
-0.1
-0.2
0
1
2
3
4
5
6
7
8
9


Suppressed Carrier Coherent Spectrum 30 SNR
2000
1000
0
-1000
-5000
-4000
-3000
-2000
-1000
0
1000
2000
3000
4000
5000


DSB-TC Time Domain
0.2
0.1
0
-0.1
-0.2
-0.3
0
1
2
3
4
5
6
7
8
9


DSB-TC Frequency Domain
×10^5
4
3
2
1
0
-2.5
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2
2.5
×10^5





EXPERIMENT THREE: FREQUENCY MODULATION:

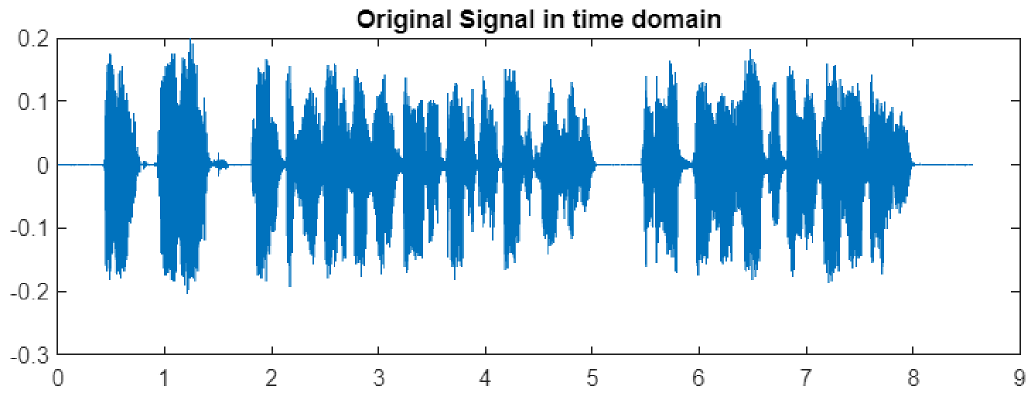
Frequency modulation (FM) is a modulation type in which the instantaneous frequency of the carrier is changed according to the message amplitude. The motive behind the frequency modulation was to develop a scheme with inherent ability to combat noise. The noise, being usually modelled as additive, has a negative effect on the amplitude by introducing unavoidable random variations which are superimposed on the desired signal. Unlike the amplitude, frequency has a latent immunity against noise. Since it resides “away” from the amplitude, any changes in the amplitude would be completely irrelevant to the frequency. In other words, there is no direct correlation between the variation in amplitude and frequency, thus making FM a better candidate over AM with respect to noise immunity. However, what FM gains in noise immunity lacks in bandwidth efficiency. Since FM usually occupies larger bandwidth, AM is considered more bandwidth wise.

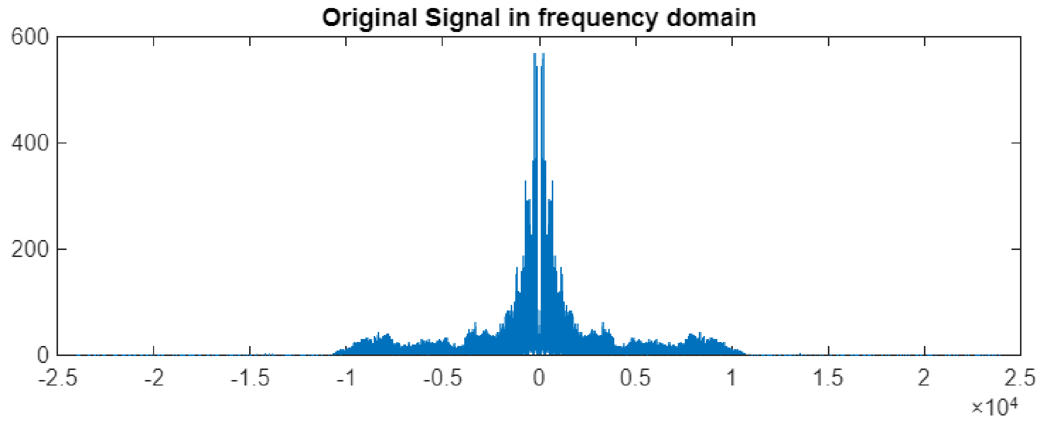
The code (copied from editor):

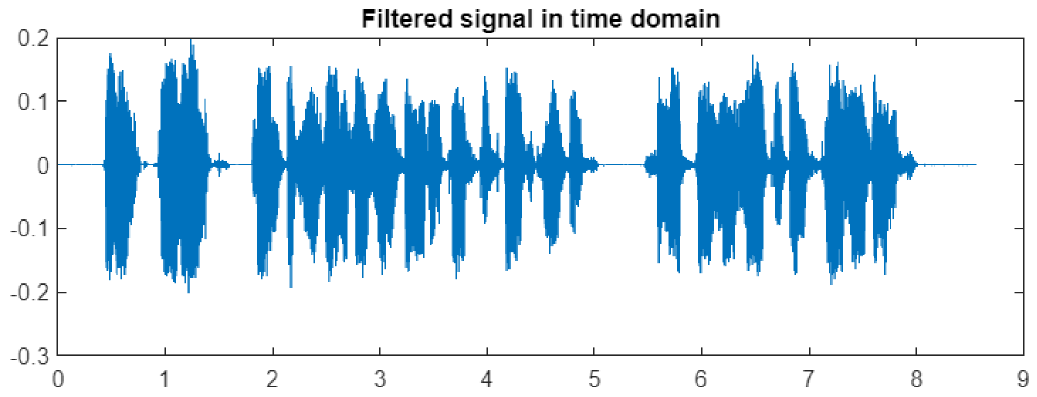
```
clc;
clear;

%Read the attached audio file , plot the spectrum of this signal.
[signal,Fs] = audioread('eric.wav');
sound(signal,Fs);
t = linspace(0,length(signal)/Fs,length(signal));
f = (-Fs/2 : Fs/length(signal) : Fs/2 - Fs/length(signal));
figure(1);
```

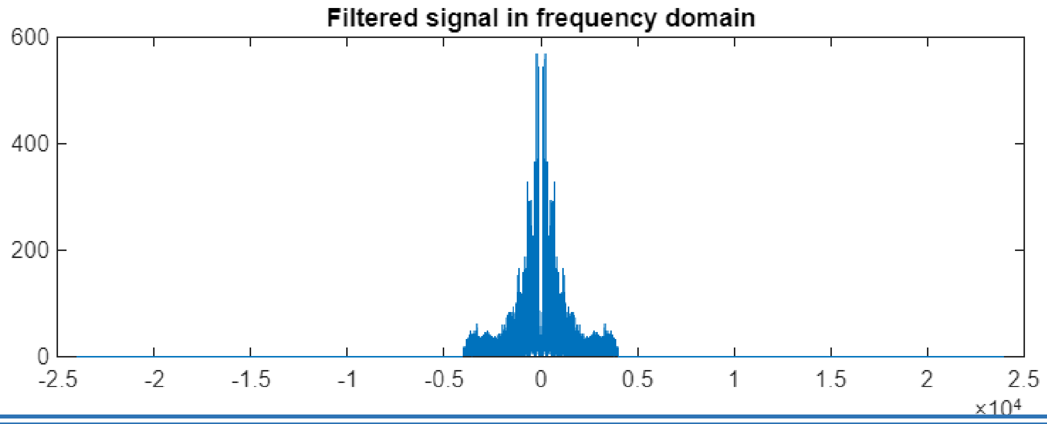
```
subplot(2,1,1);
plot(t,signal);
title ('Original Signal in time domain');
f_signal = fftshift(fft(signal));
subplot(2,1,2);
plot(f,abs(f_signal));
title('Original Signal in frequency domain');

%Use an ideal filter to remove all frequencies greater than 4KHZ
L = length(f_signal)*2*4000/Fs; % L = 68542
s = (length(f_signal)-68542)/2;
y1 = zeros(1,s);
y2 = ones(1,68542);
ideal_filter = [y1 y2 y1];
filtered_signal = f_signal.*ideal_filter';

%Obtain the filtered signal in both frequency and time domain
figure(2);
subplot(2,1,2);
plot(f,abs(filtered_signal));
title('Filtered signal in frequency domain');
t_filtered_signal=ifft(ifftshift(filtered_signal));
subplot(2,1,1);
plot(t,t_filtered_signal);
title('Filtered signal in time domain');

%Sounding the filtered audio signal
%sound(filteredTimeDom,Fs);

%Generating NBFM
kf=0.0001*pi;                                             %Setting Kf
fc=100000;                                                %100khz FC for the carrier
fsig_resam=resample(t_filtered_signal,Fs,5*fc);            %resampling with samples
increased
t=linspace(0,length(fsig_resam)/(5 * fc),length(fsig_resam));   %time of
modulation
phaseDiv= kf.*cumsum(fsig_resam)';                        %phase deviation of modulation
NBFM= cos(2*fc*pi*t)-(phaseDiv.*sin(2*fc*pi*t));          %NB equation : u(t)=cos2Pifct
- Ac.PhaseDev.sin2Pifct
xb=linspace((-5*fc/2)-fc,5*fc/2+fc,length(NBFM));         %x axis for ploting
figure(3); NBFM_FD=fftshift(fft(NBFM));                      %NBFM in frequency domain
using fourier transform.
plot(xb,abs(NBFM_FD));                                    %Plotting
title('NBFM in Frequency Domain');

%condition to generate NBFM that modulation index <<< (<1)

%Demodulation
NBFM_diff = diff(NBFM) ;                                  %Convert FM to AM by
differentiation
env =abs(hilbert(NBFM_diff));                             %Envelop detector
env = detrend(env);                                       %remove effect of AM
figure(4); plot(env);                                        %Plotting
title('NBFM after demodulatin');
```

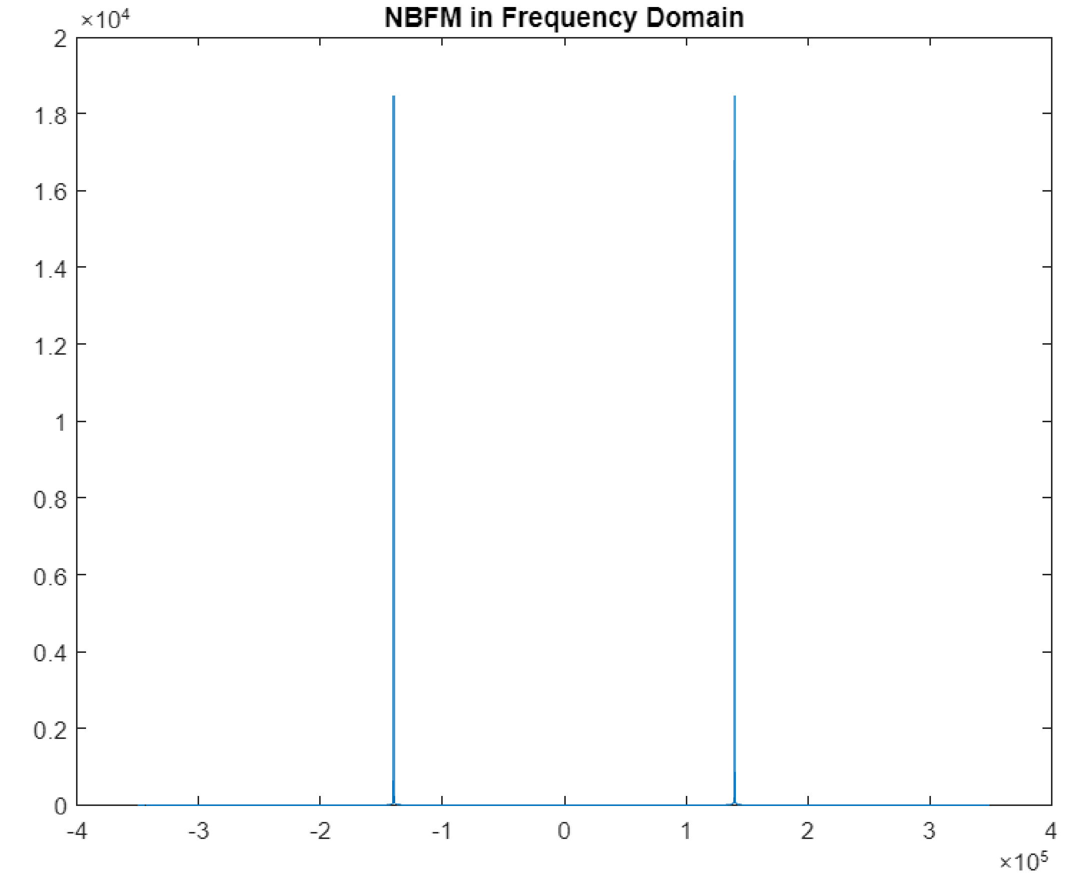
Outputs:

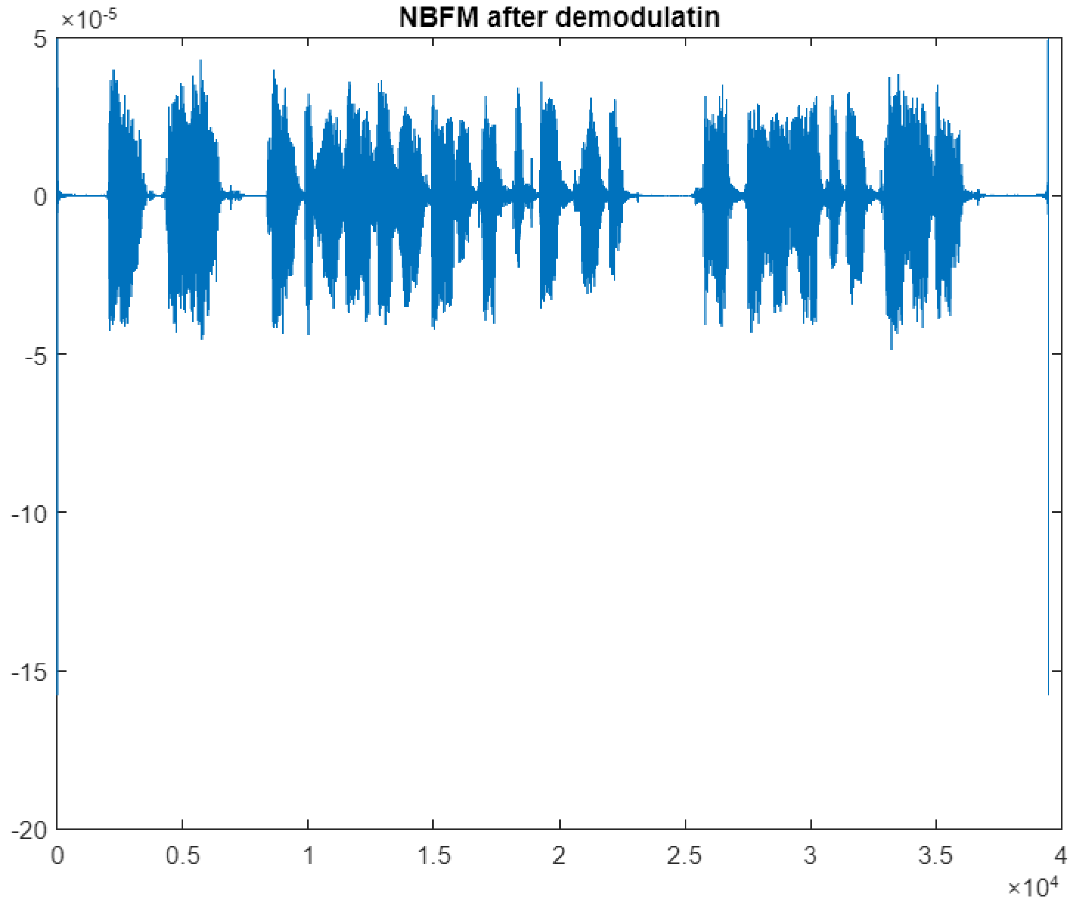

Original Signal in time domain
0.2
0.1
0
-0.1
-0.2
-0.3
0
1
2
3
4
5
6
7
8
9


Original Signal in frequency domain
600
400
200
0
-2.5
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2
2.5
×10^4


Filtered signal in time domain
0.2
0.1
0
-0.1
-0.2
-0.3
0
1
2
3
4
5
6
7
8
9


Filtered signal in frequency domain
600
400
200
0
-2.5
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2
2.5
×10^4


NBFM in Frequency Domain
×10^4
2
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
-4
-3
-2
-1
0
1
2
3
4
×10^5


NBFM after demodulatin
×10^-5
5
0
-5
-10
-15
-20
0
0.5
1
1.5
2
2.5
3
3.5
4
×10^4

Answering questions:

2) What can you make out of the resulting plot?

- Spectrum is the same as DSB-TC

3) What is the condition we needed to achieve NBFM?

- beta << 1

Useful matlab functions used and their usage:

1- Audioread:

[y,Fs] = audioread(filename) reads data from the file named filename, and returns sampled data, y, and a sample rate for that data, Fs.

2- fft:

Y = fft(X) computes the discrete Fourier transform (DFT) of X using a fast Fourier transform (FFT) algorithm.

3- fftshift:

Y = fftshift(X) rearranges a Fourier transform X by shifting the zero-frequency component to the center of the array.

4- Ifftshift:

X = ifftshift(Y) rearranges a zero-frequency-shifted Fourier transform Y back to the original transform output. In other words, ifftshift undoes the result of fftshift

5- Ifft:

X = ifft(Y) computes the inverse discrete Fourier

transform of Y using a fast Fourier transform algorithm. X is the same size as Y.

6- awgn:

out = awgn(in,snr) adds white Gaussian noise to the vector signal in

7- resample:

y = resample(x,p,q) resamples the input sequence, x, at p/q times the original sample rate

8- sound:

sound(y,Fs), sends the signal in vector y (with sample frequency Fs) to the speaker on PC

9- hilbert:

The toolbox function hilbert computes the Hilbert transform for a real input sequence x and returns a complex result of the same length, y = hilbert(x) , where the real part of y is the original real data and the imaginary part is the actual Hilbert transform.

10-abs:

Y = abs(X) returns the absolute value of each element in array X

11-max:

M = max(A) returns the maximum elements of an array

12-Mean:

M = mean(A) returns the mean of the elements of A along the array

13-Cumsum:

B = cumsum(A) returns the cumulative sum of A starting at the beginning of the array

14-diff:

Y = diff(X) calculates differences between adjacent elements of X along the array

Y = [X(2)-X(1) X(3)-X(2) ... X(m)-X(m-1)]

15-plot:

It is used to plot the graph of a function

16-butter:

[b,a] = butter(n , Wn) returns the transfer function coefficients of an n th-order lowpass digital Butterworth filter with normalized cutoff frequency Wn

17-designfilt:

d = designfilt(resp,Name,Value) to design a digital filter, d , with response type resp . Customize the filter further using Name,Value pairs.

18-filtfilt:

y = filtfilt(b , a , x) performs zero-phase digital filtering by processing the input data, x , in both the forward and reverse directions.

Conclusions:

- Frequency modulation’s SNR is better than SNR of the Amplitude modulation
- Envelope detector is a good method of detection for DSB-TC but not for DSB-SC
- When the SNR increases the quality of the sound increases.
- Coherent detection is suitable for any type of modulation.