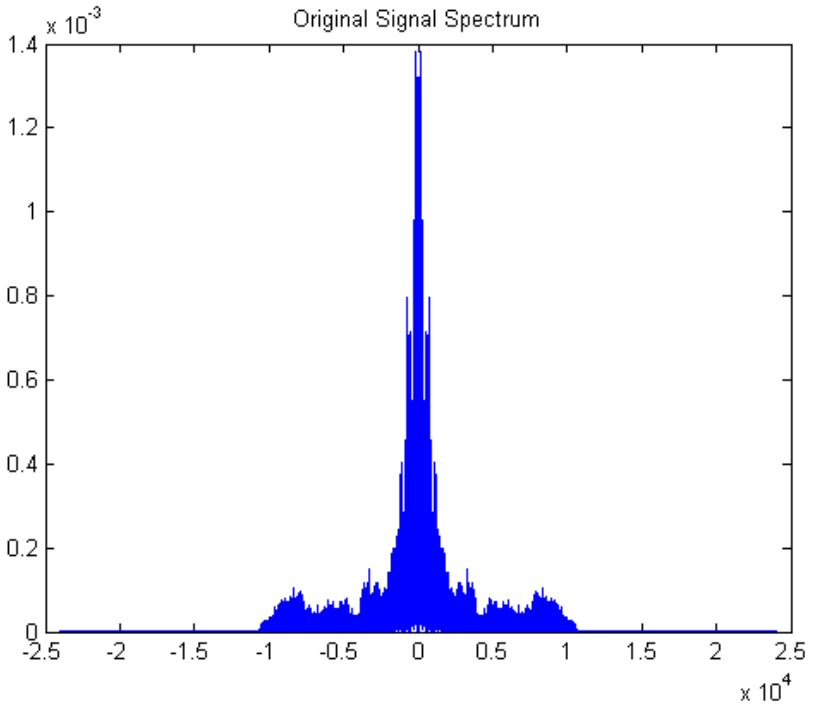
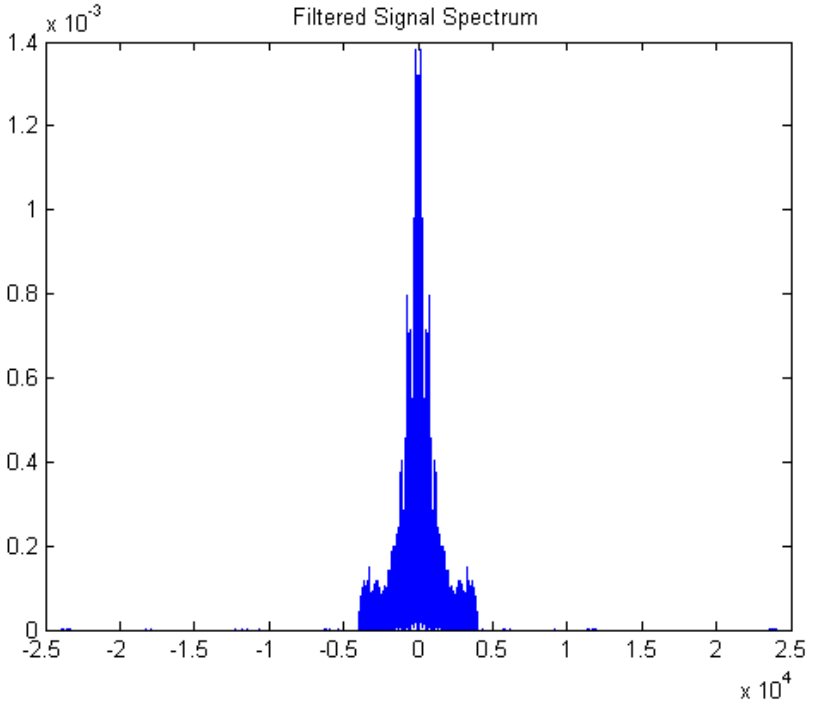
```
clear;
close all;

% Exp 2

[S, Fs] = audioread('eric.wav'); % read the attached audio file

% Fourier transform
L = length(S);
F = fftshift(fft(S));
f = Fs/2*linspace(-1,1,L);

% Plot
figure; plot(f, abs(F)/L); title('Original Signal Spectrum');

% Filter as 4kHz
W = 4000;
F(f>=W | f<=-W) = 0;
y =ifft(ifftshift(F));

% Plot after filter
L = length(y);
F = fftshift(fft(y));
f = Fs/2*linspace(-1,1,L);
figure; plot(f, abs(F)/L); title('Filtered Signal Spectrum');

%calculate time vector
tstart = 0;
tend = tstart + length(y) / Fs;
t1 = linspace(tstart, tend, length(y));
t1 = t1';

figure; plot(t1, y); title('Filtered Signal Time Domain');

% Hear the signal
sound(abs(y),Fs);

% Calculate constants
fm = W;
fc = 100000;
Am = max(y);
Ac = 2*Am;

% resample at 5Fc
y = resample(y,5*fc,Fs);
Fs = 5*fc;

% SSBSC

% calculate time vector
tstart = 0;
tend = tstart + length(y) / Fs;
t1 = linspace(tstart, tend, length(y));
t1 = t1';

% DSBSC generation
carrier_signal = cos(2*pi*fc*t1);
DSBSC=y.*carrier_signal;

% SSBLSB generation by filtering DSBSC
SSBLSB = DSBSC;
L = length(SSBLSB);
f = Fs/2*linspace(-1,1,L);
F = fftshift(fft(SSBLSB));  %fourier transform
F(f>=fc | f<=-fc) = 0;  %filter
figure; plot(f, abs(F) / L); title('SSBLSB Frequency Domain');
```

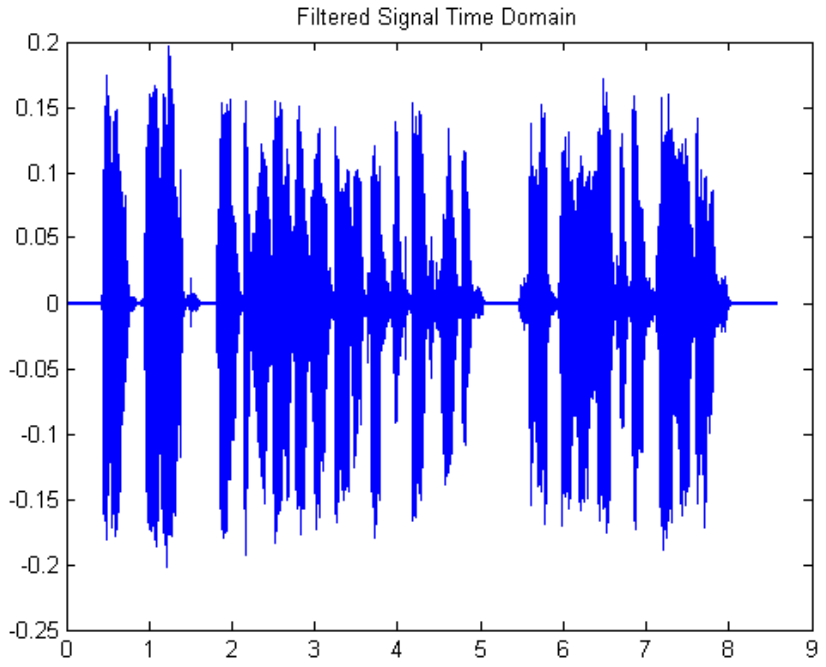
```
% Coherent Detection SSBSC using ideal filter
demodulated = SSBLSB .* cos(2*pi*fc*t1);

% fourier transform
demodulated_FFT = fftshift(fft(demodulated));

% LPF at Fm
demodulated_FFT(f>=W | f<=-W) = 0;

% inverse fourier transform to get demodulated signal in time domain
demodulated = ifft(ifftshift(demodulated_FFT));

% plot demodulated signal in time domain
figure; plot(t1, demodulated); title('Demodulated Signal using ideal filter in Time Domain');

% fourier transform
L = length(demodulated);
F = fftshift(fft(demodulated));
f = Fs/2*linspace(-1,1,L);

% plot demodulated signal in frequency domain
figure; plot(f, abs(F) / L); title('Demodulated Signal using ideal filter in Frequency Domain');

% resample to sound the demodulated signal
demodulated = resample(abs(demodulated), Fs/5, Fs);
sound(abs(demodulated), Fs/5);

% Coherent Detection SSBSC using butter order = 4
demodulated = SSBLSB.*cos(2*pi*fc*t1);
[b,a] = butter(4,W*2/Fs);
demodulated = filtfilt(b,a,demodulated);

% plot demodulated signal in time domain
figure; plot(t1, demodulated); title('Demodulated Signal using butter in Time Domain');

%fourier transform
L = length(demodulated);
F = fftshift(fft(demodulated));
f = Fs/2*linspace(-1,1,L);

% plot demodulated signal in frequency domain
figure; plot(f, abs(F) / L); title('Demodulated Signal using butter in Frequency Domain');

% resample to sound the demodulated signal
demodulated = resample(abs(demodulated), Fs/5, Fs);
sound(abs(demodulated), Fs/5);

% 0 SNR
% generate signal+noise
noisy_SSBLSB_0dB = awgn(SSBLSB, 0);

% Coherent Detection SSBSC using ideal filter
demodulated = noisy_SSBLSB_0dB.*cos(2*pi*fc*t1);
demodulated_FFT = fftshift(fft(demodulated));
demodulated_FFT(f>=W | f<=-W) = 0;
demodulated = ifft(ifftshift(demodulated_FFT));

% plot demodulated signal in time domain
figure; plot(t1, demodulated); title('0 SNR demodulated signal in time domain');

% fourier transform
L = length(demodulated);
F = fftshift(fft(demodulated));
f = Fs/2*linspace(-1,1,L);

% plot demodulated signal in frequency domain
```

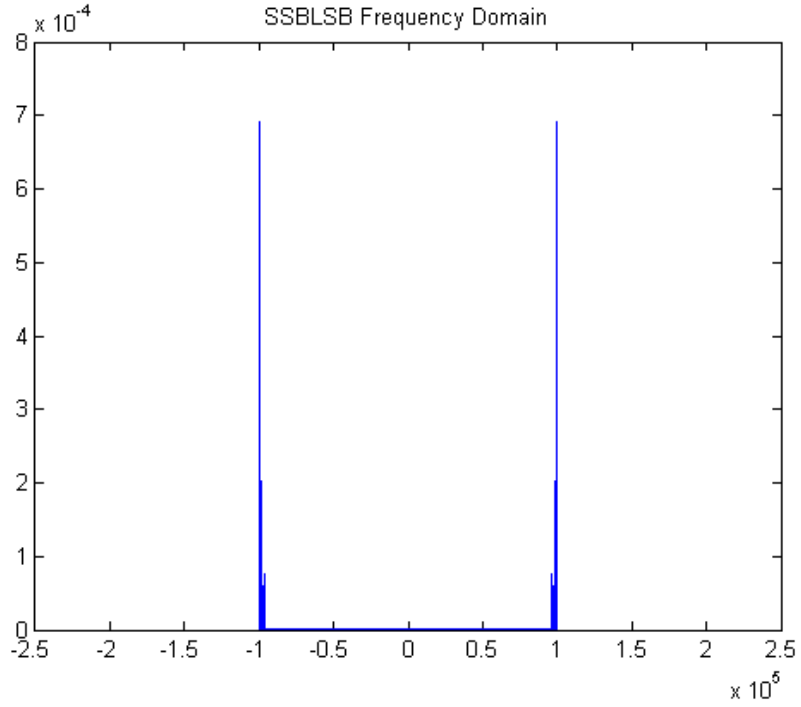
```
figure; plot(f, abs(F) / L); title('0 SNR demodulated signal in frequency domain');

% resample to sound the demodulated signal
demodulated = resample(abs(demodulated), Fs/5, Fs);
sound(abs(demodulated), Fs/5);

% 10 SNR
% generate signal+noise
noisy_SSBLSB_10dB = awgn(SSBLSB, 10);

% Coherent Detection SSBSC using ideal filter
demodulated = noisy_SSBLSB_10dB.*cos(2*pi*fc*t1);
demodulated_FFT = fftshift(fft(demodulated));
demodulated_FFT(f>=W | f<=-W) = 0;
demodulated = ifft(ifftshift(demodulated_FFT));

% plot demodulated signal in time domain
figure; plot(t1, demodulated); title('10 SNR demodulated signal in time domain');

% fourier transform
L = length(demodulated);
F = fftshift(fft(demodulated));
f = Fs/2*linspace(-1,1,L);

% plot demodulated signal in frequency domain
figure; plot(f, abs(F) / L); title('10 SNR demodulated signal in frequency domain');

% resample to sound the demodulated signal
demodulated = resample(abs(demodulated), Fs/5, Fs);
sound(abs(demodulated), Fs/5);

% 30 SNR
% generate signal+noise
noisy_SSBLSB_30dB = awgn(SSBLSB, 30);

% Coherent Detection SSBSC using ideal filter
demodulated = noisy_SSBLSB_30dB.*cos(2*pi*fc*t1);
demodulated_FFT = fftshift(fft(demodulated));
demodulated_FFT(f>=W | f<=-W) = 0;
demodulated = ifft(ifftshift(demodulated_FFT));

% plot demodulated signal in time domain
figure; plot(t1, demodulated); title('30 SNR demodulated signal in time domain');

% fourier transform
L = length(demodulated);
F = fftshift(fft(demodulated));
f = Fs/2*linspace(-1,1,L);

% plot demodulated signal in frequency domain
figure; plot(f, abs(F) / L); title('30 SNR demodulated signal in frequency domain');

% resample to sound the demodulated signal
demodulated = resample(abs(demodulated), Fs/5, Fs);
sound(abs(demodulated), Fs/5);

% SSBTC
SSBTC = Ac .* carrier_signal + SSBLSB;

% fourier transform
L = length(SSBTC);
F = fftshift(fft(SSBTC));
f = Fs/2*linspace(-1,1,L);

% Plot SSBTC frequency  domain
figure; plot(f, abs(F) / L); title('SSBTC in Frequency Domain');
```

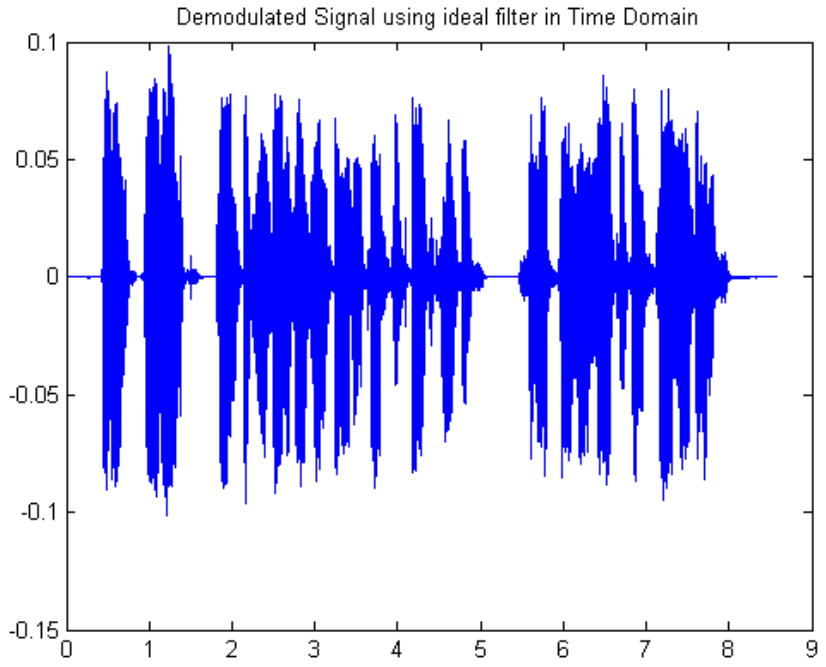
```
% envelope detector SSBTC
envelopeSSBTC = abs(hilbert(SSBTC));

figure; plot(t1, SSBTC);
hold on;
plot(t1,-envelopeSSBTC,'r-',t1,envelopeSSBTC,'-r','Linewidth',1.5);
title('SSBTC time (blue) domain with envelope detector (red)');
hold off;
ylim([-5 5])
xlim([3 3.5])

% resample to sound the signal
envelopeSSBTC = resample(envelopeSSBTC, Fs/5, Fs);
sound(abs(envelopeSSBTC), Fs/5)
```

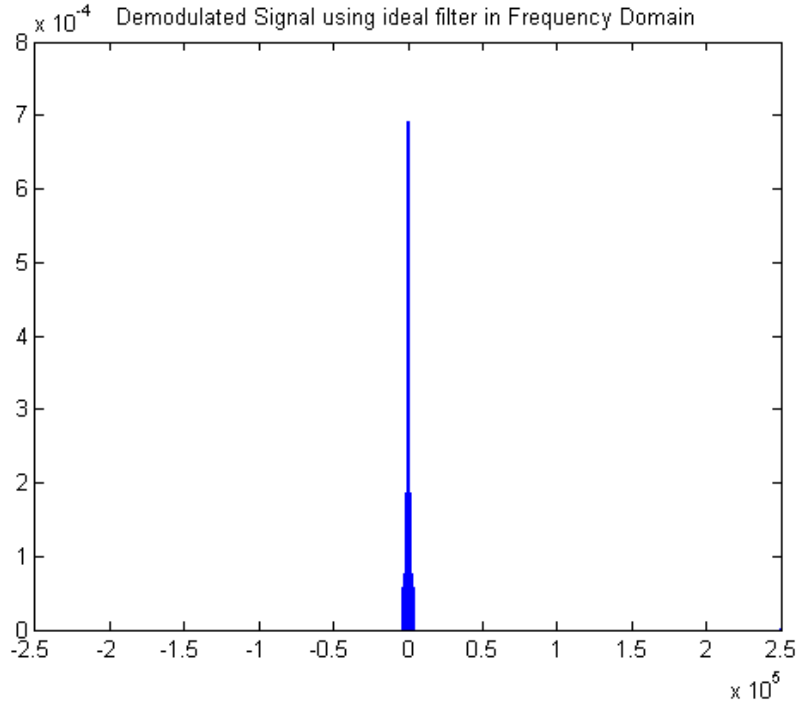

Original Signal Spectrum
x 10^-3
1.4
1.2
1
0.8
0.6
0.4
0.2
0
-2.5
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2
2.5
x 10^4

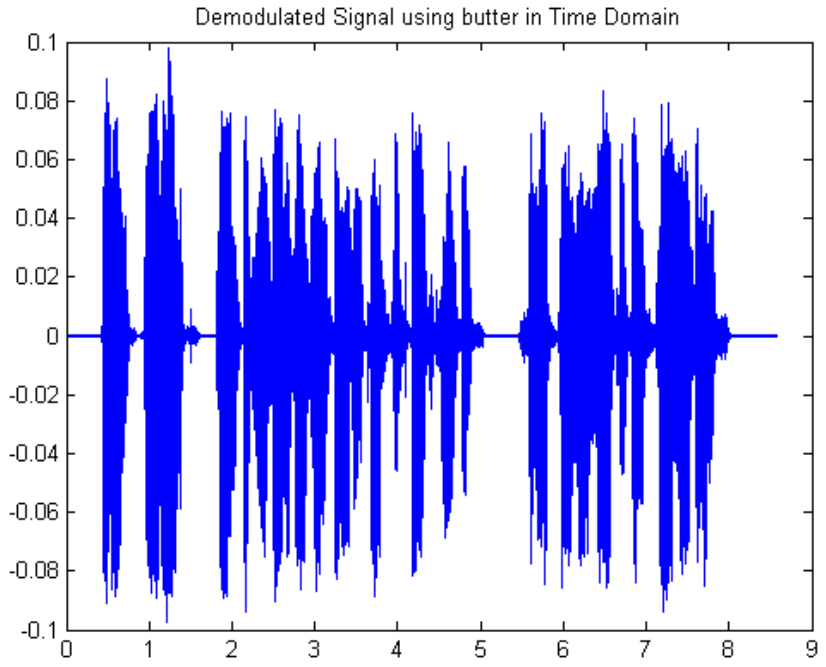

Filtered Signal Spectrum
x 10^-3
1.4
1.2
1
0.8
0.6
0.4
0.2
0
-2.5
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2
2.5
x 10^4

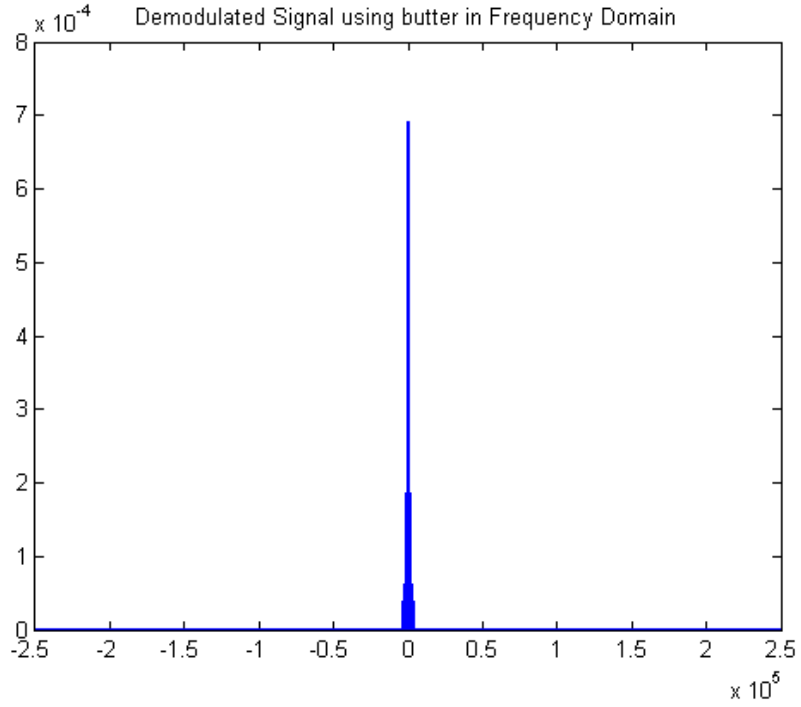

Filtered Signal Time Domain
0.2
0.15
0.1
0.05
0
-0.05
-0.1
-0.15
-0.2
-0.25
0
1
2
3
4
5
6
7
8
9

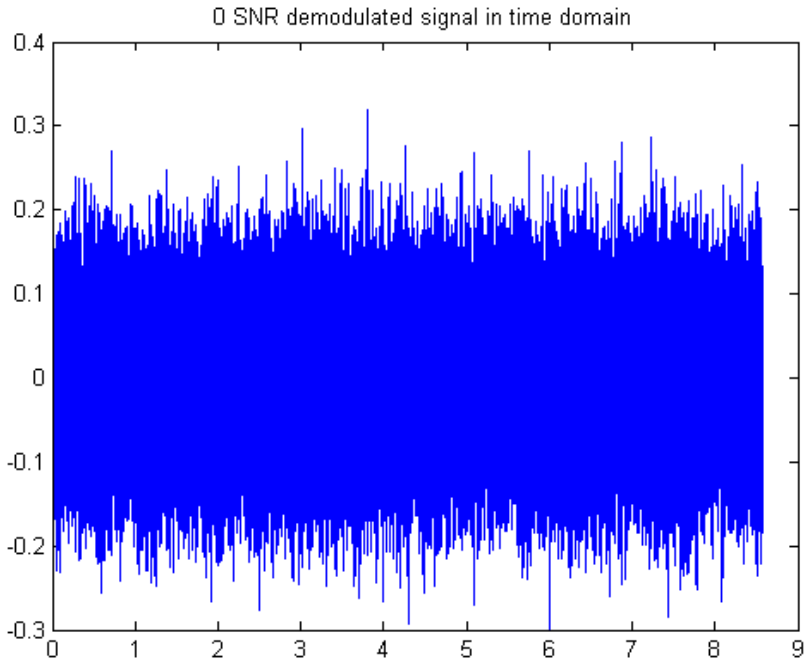

SSBLSB Frequency Domain
x 10^-4
8
7
6
5
4
3
2
1
0
-2.5
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2
2.5
x 10^5

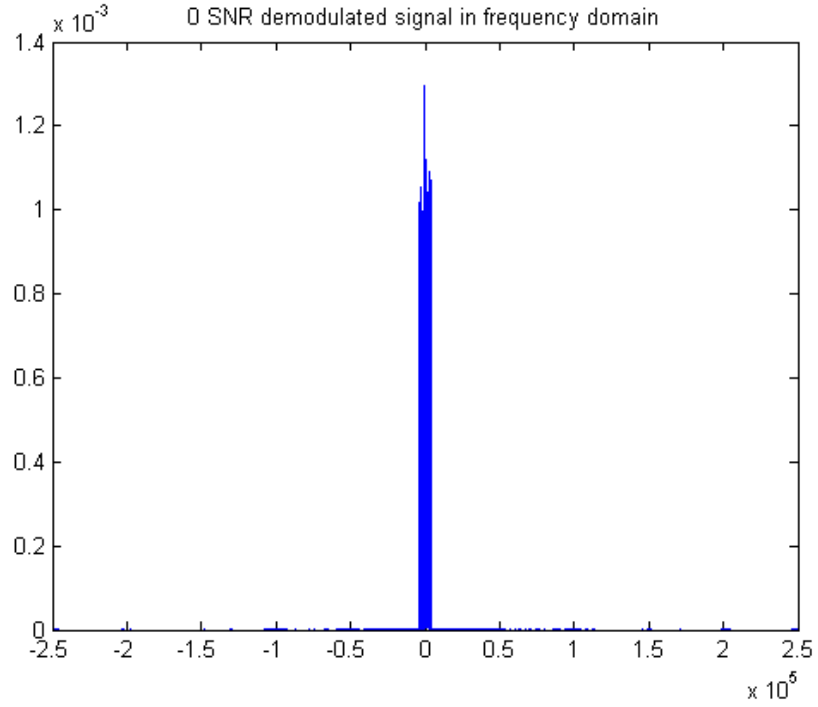

Demodulated Signal using ideal filter in Time Domain
0.1
0.05
0
-0.05
-0.1
-0.15
0
1
2
3
4
5
6
7
8
9

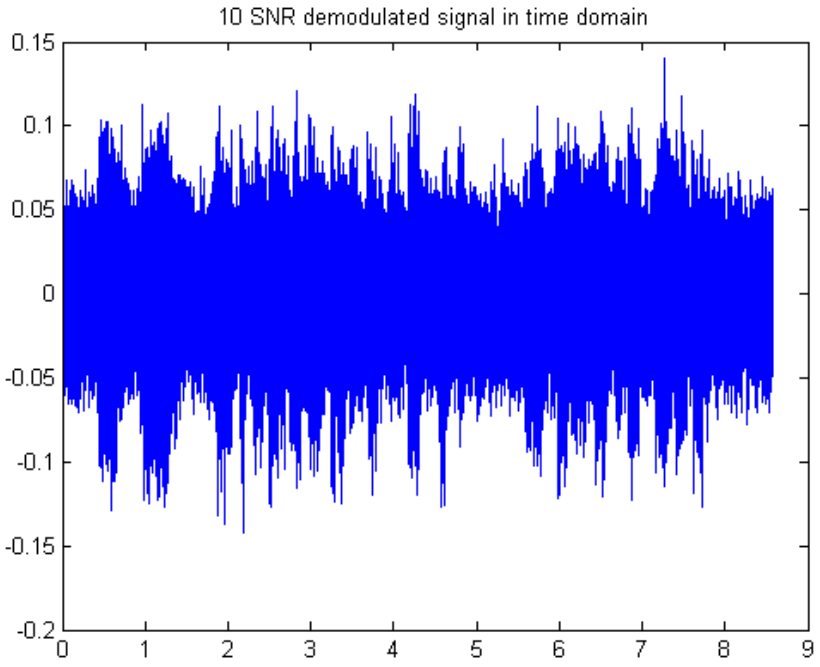

x 10^-4
Demodulated Signal using ideal filter in Frequency Domain
8
7
6
5
4
3
2
1
0
-2.5
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2
2.5
x 10^5

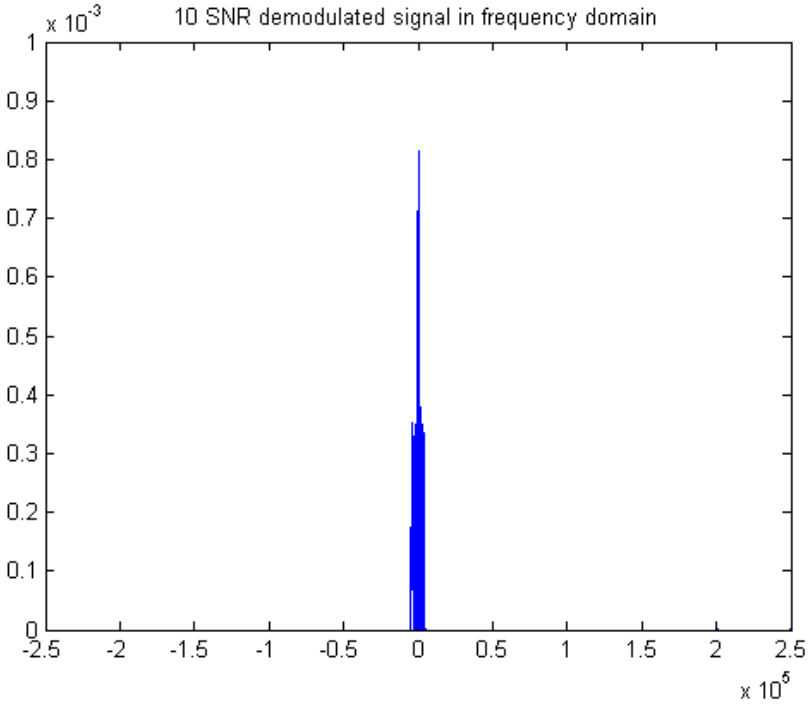

Demodulated Signal using butter in Time Domain
0.1
0.08
0.06
0.04
0.02
0
-0.02
-0.04
-0.06
-0.08
-0.1
0
1
2
3
4
5
6
7
8
9

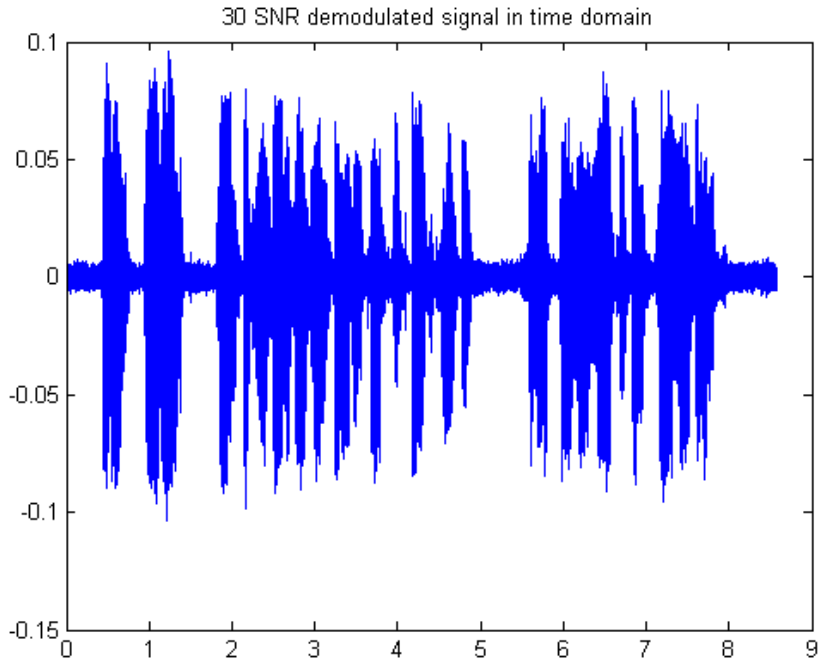

Demodulated Signal using butter in Frequency Domain
x 10^-4
8
7
6
5
4
3
2
1
0
-2.5
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2
2.5
x 10^5

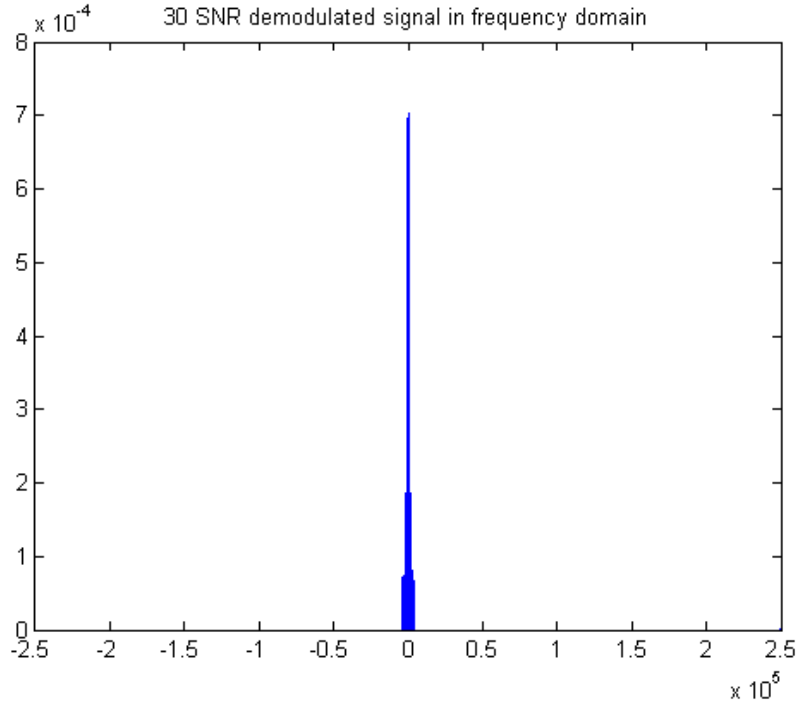

0 SNR demodulated signal in time domain
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
0
1
2
3
4
5
6
7
8
9

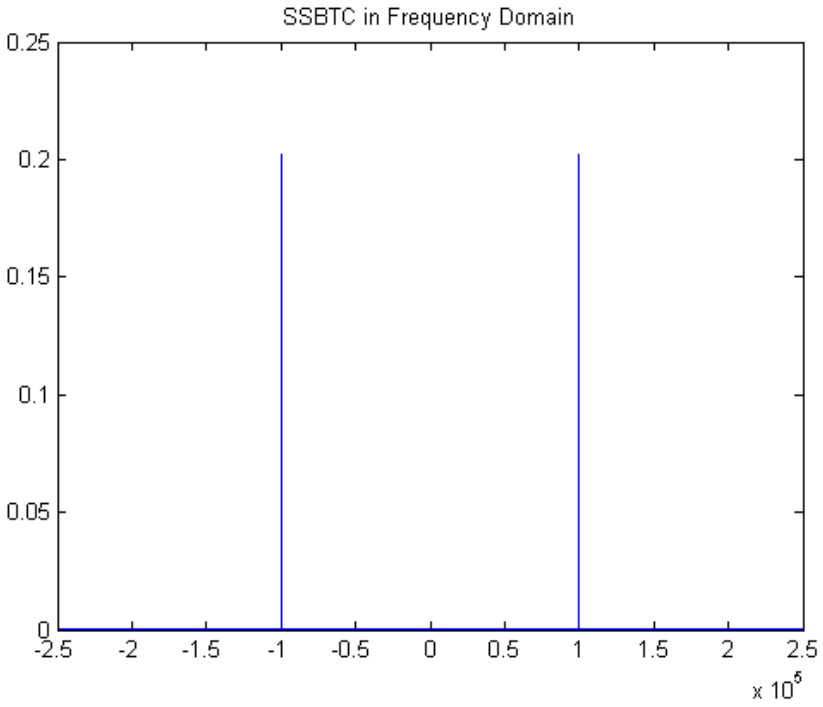

x 10^-3
0 SNR demodulated signal in frequency domain
1.4
1.2
1
0.8
0.6
0.4
0.2
0
-2.5
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2
2.5
x 10^5

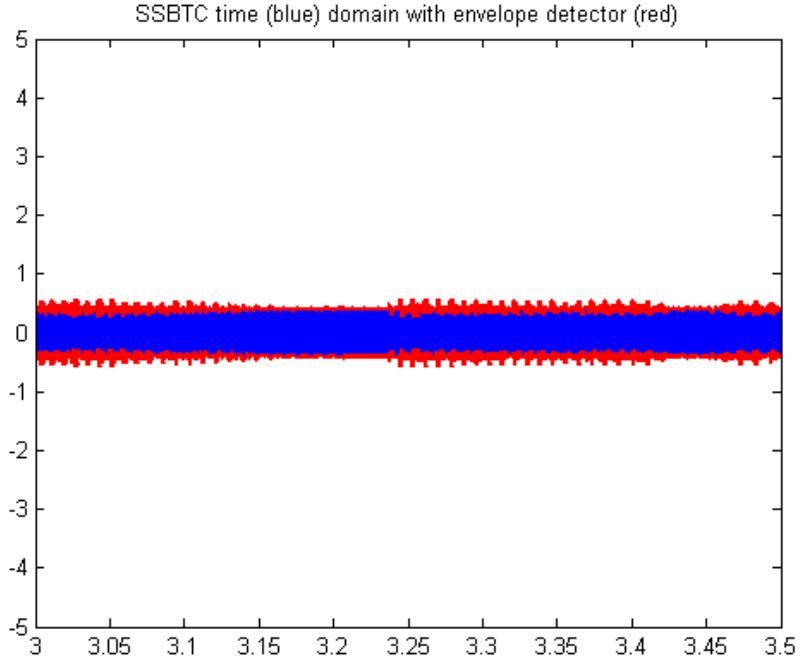

10 SNR demodulated signal in time domain
0.15
0.1
0.05
0
-0.05
-0.1
-0.15
-0.2
0
1
2
3
4
5
6
7
8
9


10 SNR demodulated signal in frequency domain
x 10^-3
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
-2.5
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2
2.5
x 10^5


30 SNR demodulated signal in time domain
0.1
0.05
0
-0.05
-0.1
-0.15
0
1
2
3
4
5
6
7
8
9


30 SNR demodulated signal in frequency domain
x 10^-4
8
7
6
5
4
3
2
1
0
-2.5
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2
2.5
x 10^5





Published with MATLAB® R2013a