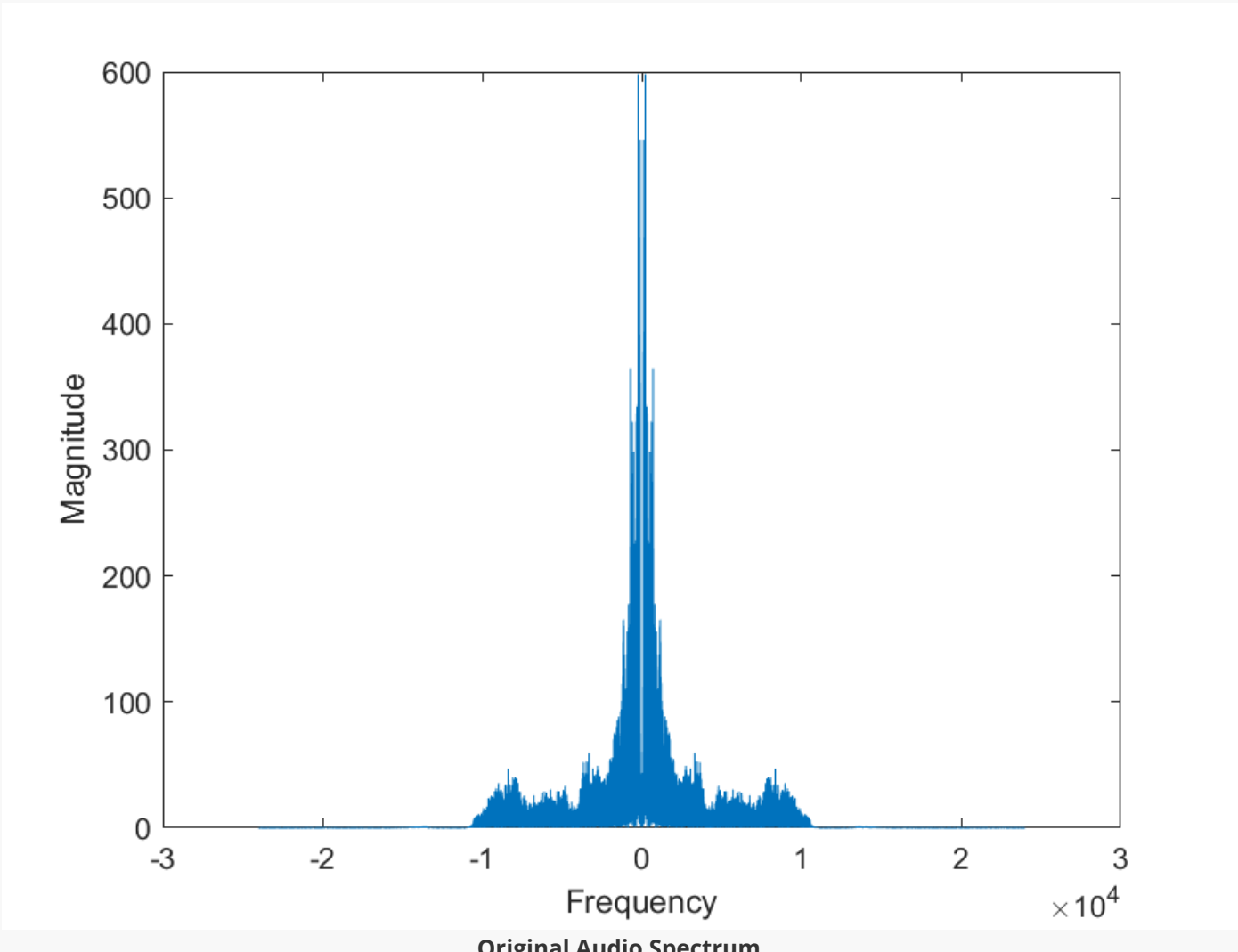
Analogue Communication Theory

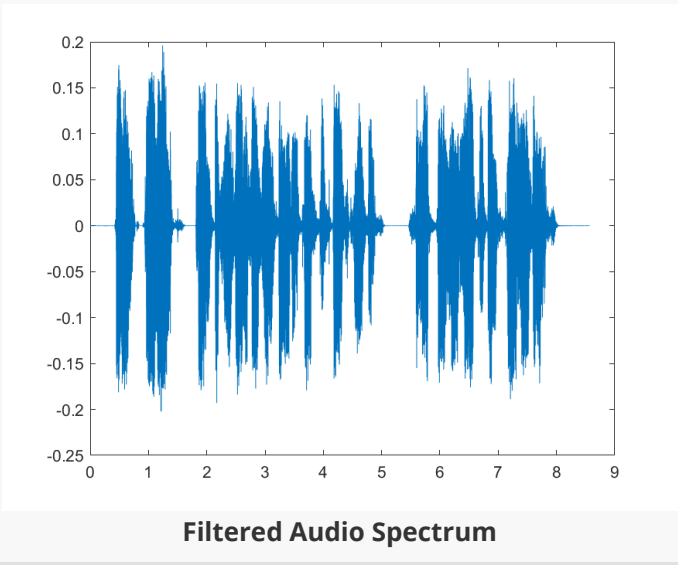
Final Lab Project

- Ahmed Osama 6078
- Yehia Salah 6218
- Youssef Hassan 6259
- Noureldein Hazem 6261
- Amr Ayman 6273
- Omar Mohamed 6290
- Youssef Sabry 6239

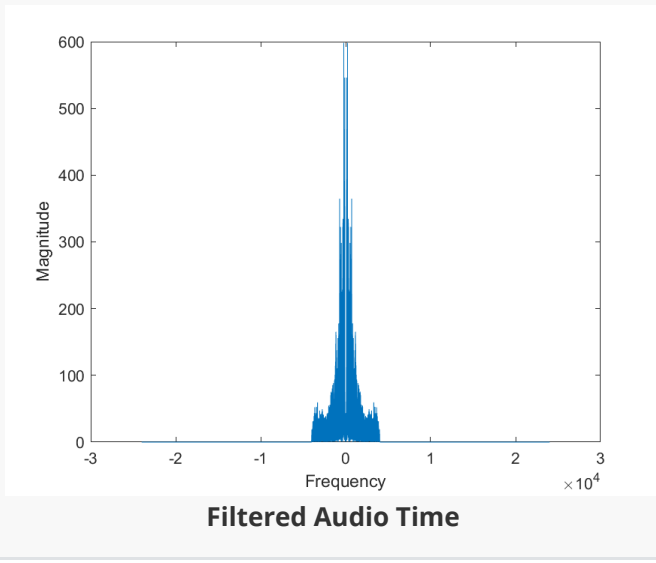
Original and Filtered Audio



Original Audio Spectrum



Filtered Audio Spectrum



Filtered Audio Time

DSB Modulation

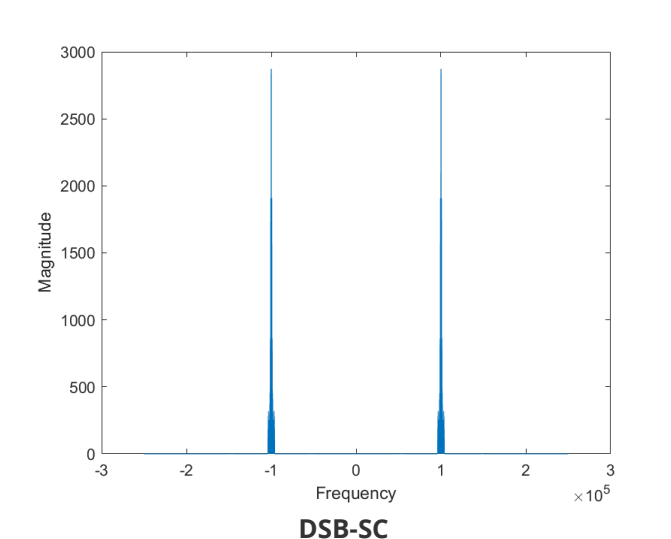
Modulation

```
classdef dsbmod
    properties
        fs_original
        resampled_msg
        carrier
    end
    methods
        function obj = dsbmod(msgsig,fc,fs_original,Xnyquist)
            obj.fs_original = fs_original;
            new_fs = fc*Xnyquist;
            [p,q] = rat(new_fs/fs_original);
            obj.resampled_msg = resample(msgsig,p,q);
            msg_len = length(obj.resampled_msg);
            obj.carrier = commonspectrum.gen_carrier(fc,0,msg_len/new_fs,msg_len);
        end
        function sc = suppressed_carrier(obj,Ac)
            sc = (Ac .* obj.carrier) .* obj.resampled_msg;
        end
        function tc = transmitted_carrier(obj,Mu,Ac)
            tc = Ac*(1 + Mu * obj.resampled_msg/max(obj.resampled_msg)) .* obj.carrier;
        end
    end
end
```

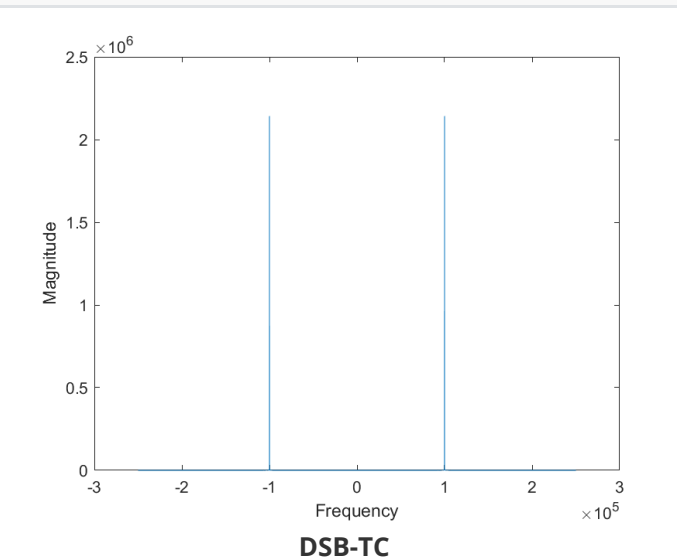
Demodulation

```
classdef dsbdemod
    methods (Static)
    function demodulated=envelope(modulated,fcut,fs)
       temp= abs(hilbert(modulated));
       filtered=lowpass(temp,fcut,fs,'ImpulseResponse','iir');
       %HardCoded
       [p,q]=rat(48/500);
       demodulated=resample(filtered,p,q);
    end
    function demodulated=coherent(modulated,fc,fs,phase,fcut)
        len=length(modulated);
        demod= modulated.*commonspectrum.gen_carrier(fc,phase,len/fs,len);
        filter = generate_filter(length(demod),fs,fcut);
        temp = fftshift(fft(demod)).*filter;
        temp =ifft(ifftshift(temp));
        %HardCoded
       [p,q]=rat(48/500);
       demodulated=resample(temp,p,q);
    end
    end
end
```

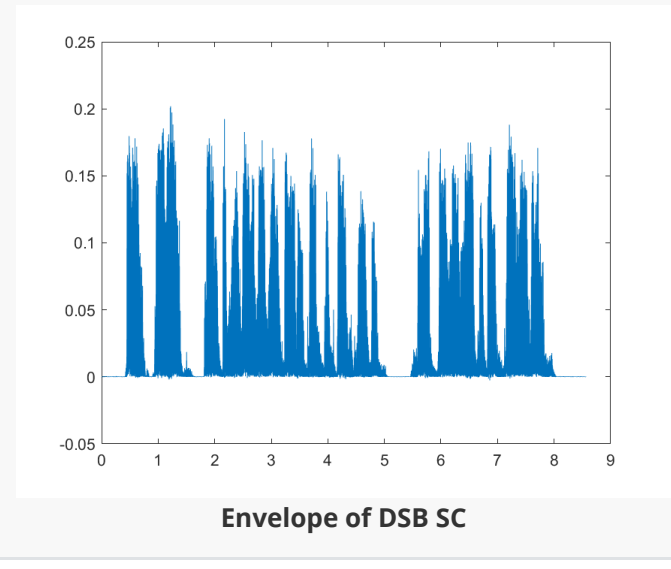
Figures



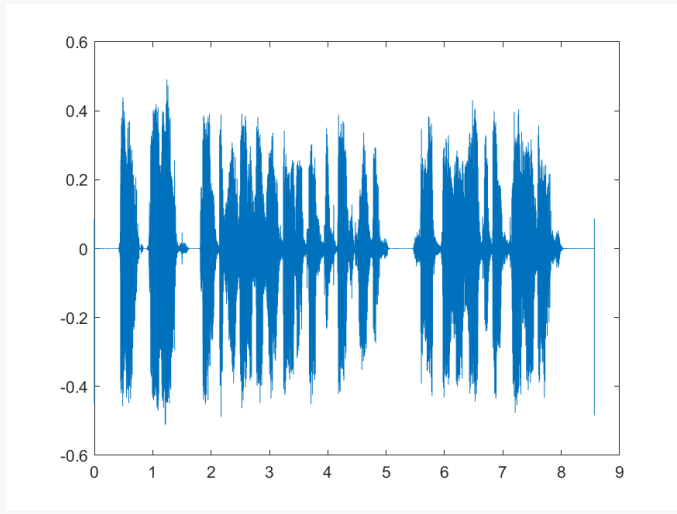
DSB-SC



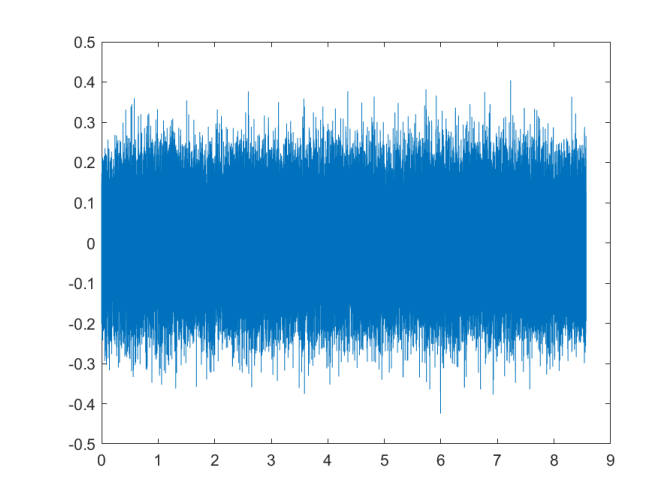
DSB-TC



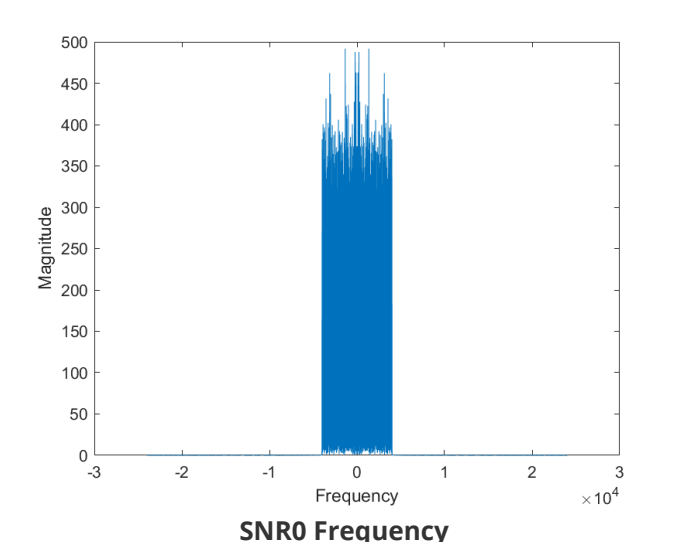
Envelope of DSB SC



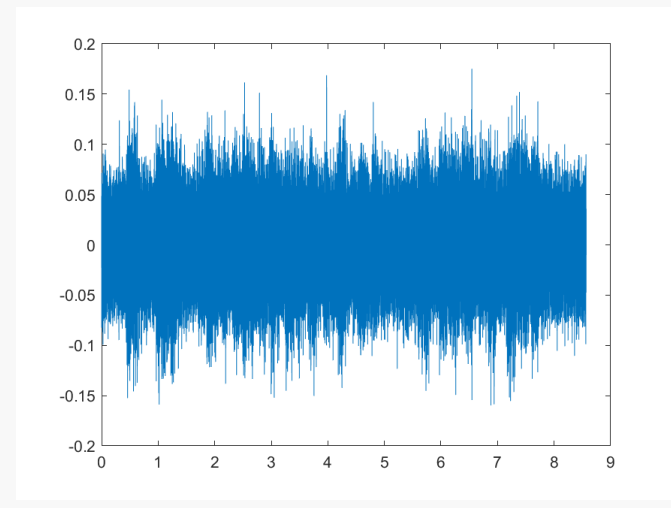
Envelope of DSB TC



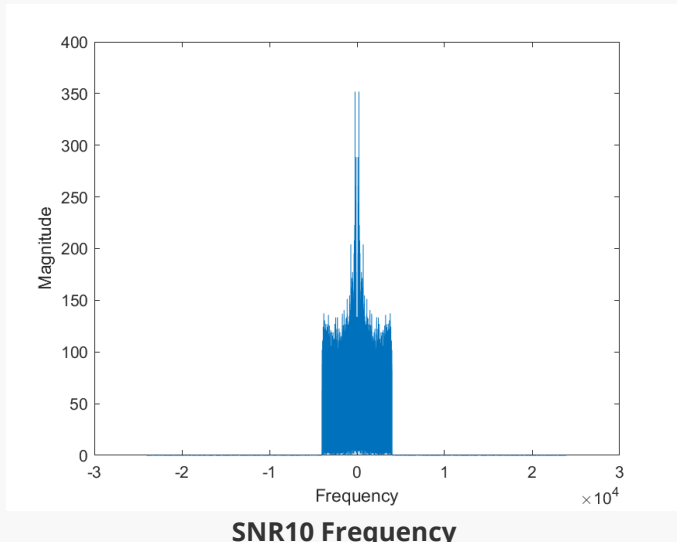
SNR0 Time



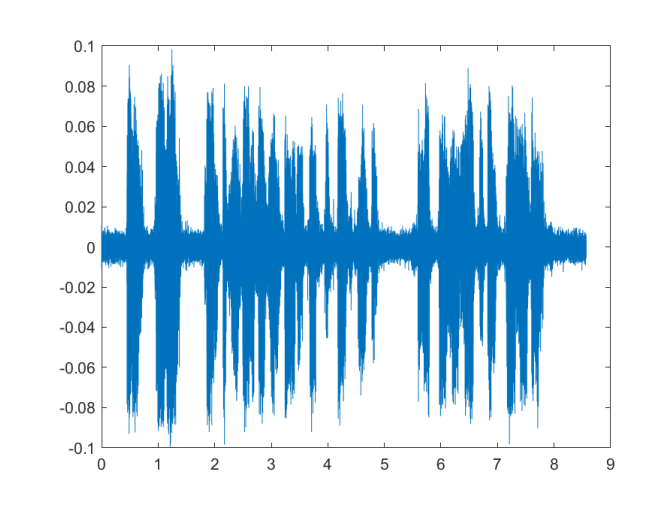
SNR0 Frequency



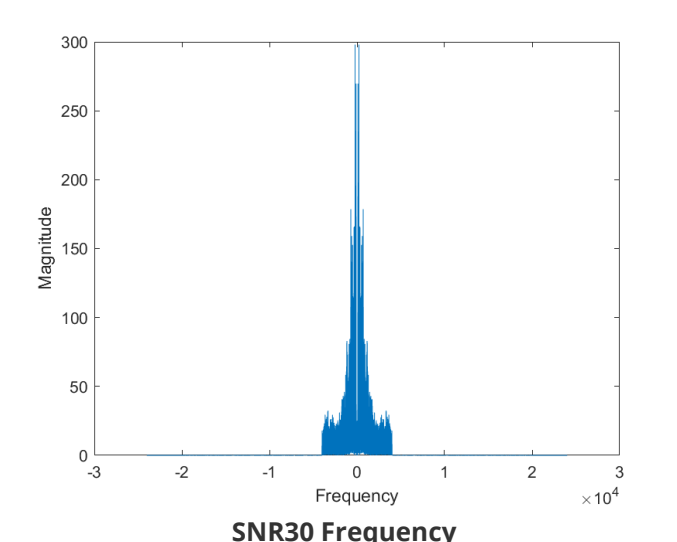
SNR10 Time



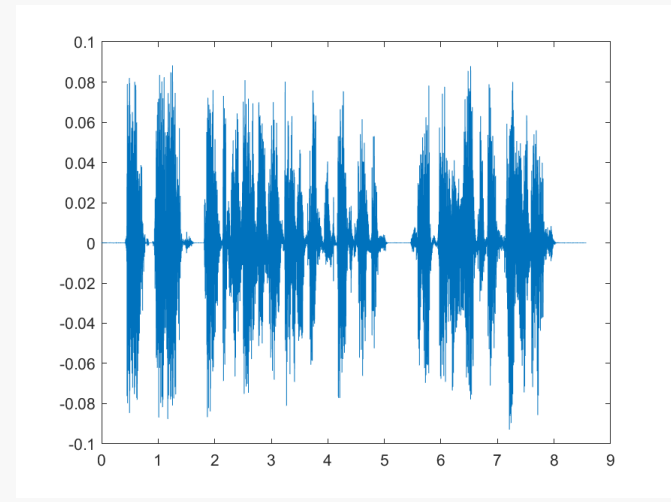
SNR10 Frequency



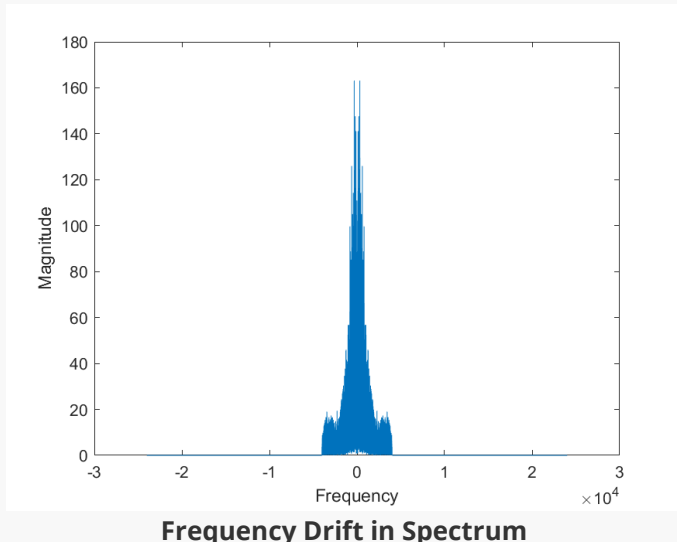
SNR30 Time



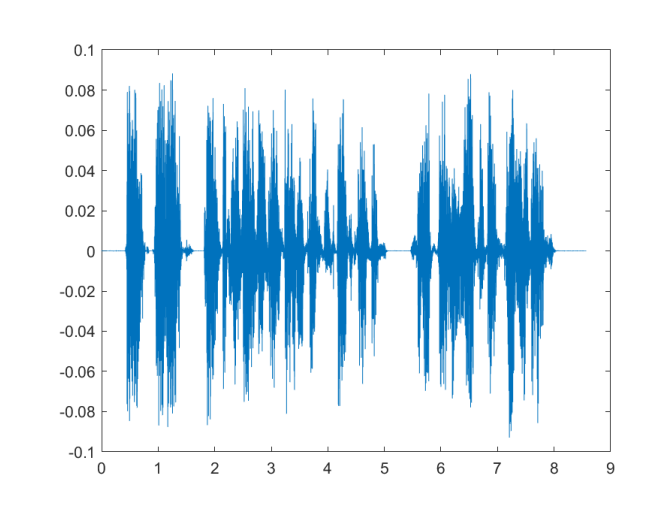
SNR30 Frequency



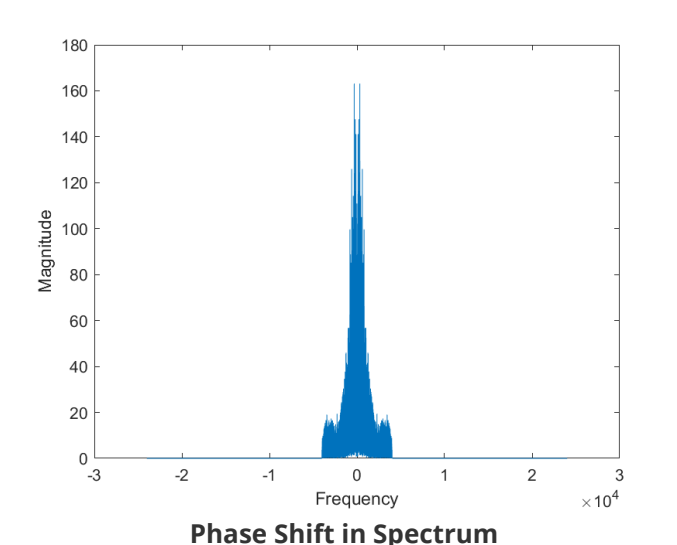
Frequency Drift in Time



Frequency Drift in Spectrum



Phase Shift in Time



Phase Shift in Spectrum

Conclusions for DSB Modulation:

- Envelope Detector should only be used for transmitted carrier only
- Frequency Error is called Frequency Drift

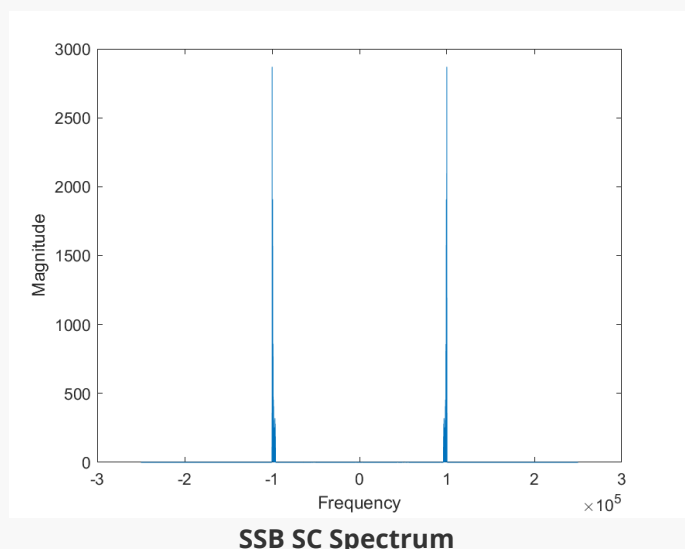
SSB Modulation

Modulation:

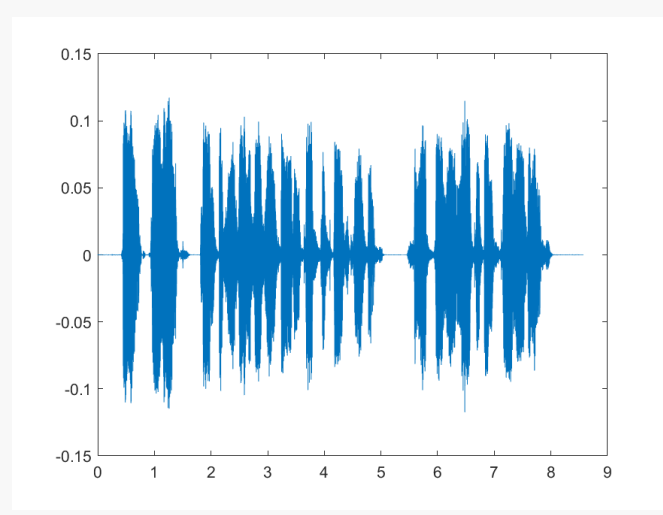
```
classdef ssbmod
    methods (Static)
        function ideal = idealSC(dsbsc,fc)
            len=length(dsbsc);
            filter=generate_filter(len,5*fc,fc);
            %filter off by one
            ssbf=fftshift(fft(dsbsc)).*filter;
            ideal=real(ifft(ifftshift(ssbf)));
        end

        function practical = practicalSC(dsbsc,fc)
            fs = fc * 5;
            flow = (fc-4e3)/ (fs/2);
            fhigh = fc/ (fs/2);
            fcut = [flow, fhigh];
            [b, a] = butter(4,fcut);
            practical = filter(b, a, dsbsc);
        end
        function tc = transmitted_carrier(dsbtc,fc)
           len=length(dsbtc);
           filter=generate_filter(len,5*fc,fc);
           r = find(filter==1,1,'last');
           filter(r+1) = 1;
            ssbf=fftshift(fft(dsbtc)).*filter;
            tc=ifft(ifftshift(ssbf));
        end
    end
end
```

Demodulation:

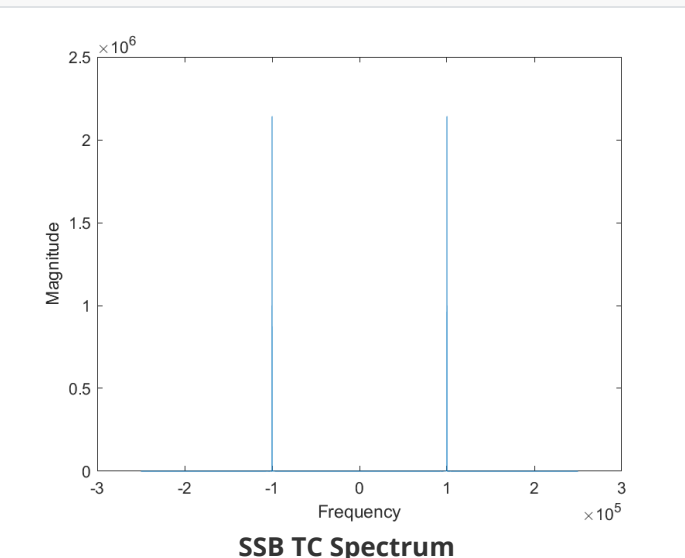
```
classdef ssbdemod
    methods (Static)
    function demodulated = envelope(modulated,fcut,fs)
       temp= abs(hilbert(modulated));
       demod = temp - mean(temp);
       filter=generate_filter(length(demod),fs,fcut);

       filtered=real(ifft(ifftshift(fftshift(fft(demod)).*filter)));
       %HardCoded
       [p,q]=rat(48/500);
       demodulated=resample(filtered,p,q);
    end
    function demodulated = coherent(modulated,fc,fs,phase,fcut)
        len = length(modulated);
        demod = modulated.*commonspectrum.gen_carrier(fc,phase,len/fs,len);
        filter = generate_filter(length(demod),fs,fcut);
        temp = fftshift(fft(demod)).*filter;
        filtered =ifft(ifftshift(temp));
        %HardCoded
       [p,q] = rat(48/500);
       demodulated = resample(filtered,p,q);
    end
    end
end
```



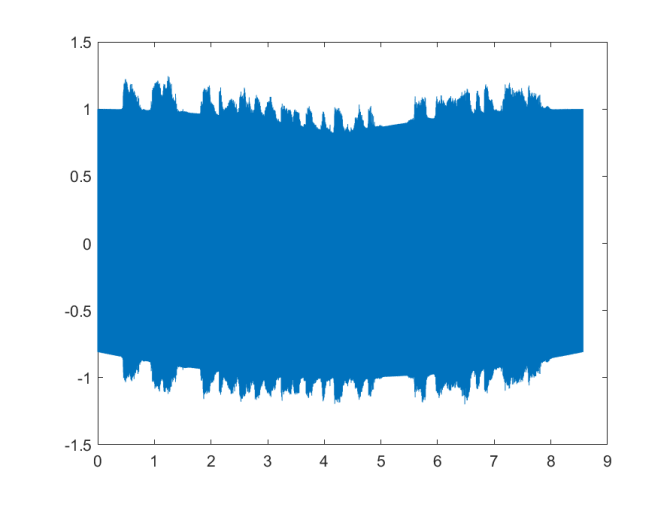
SSB SC Spectrum



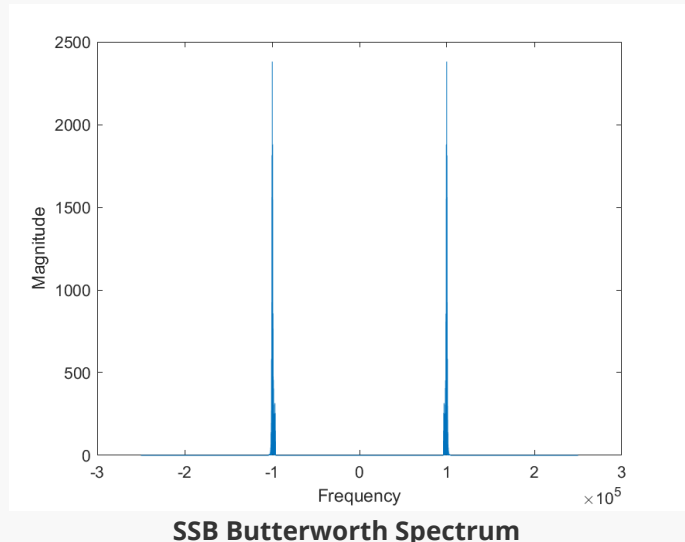
SSB SC Time



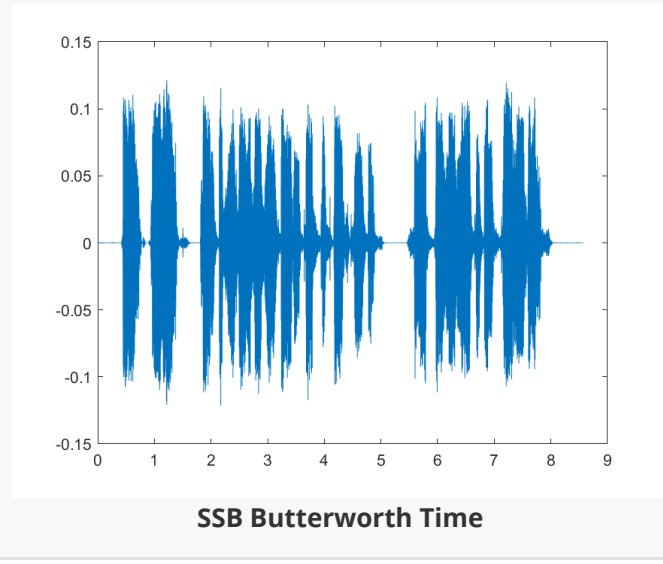
SSB TC Spectrum



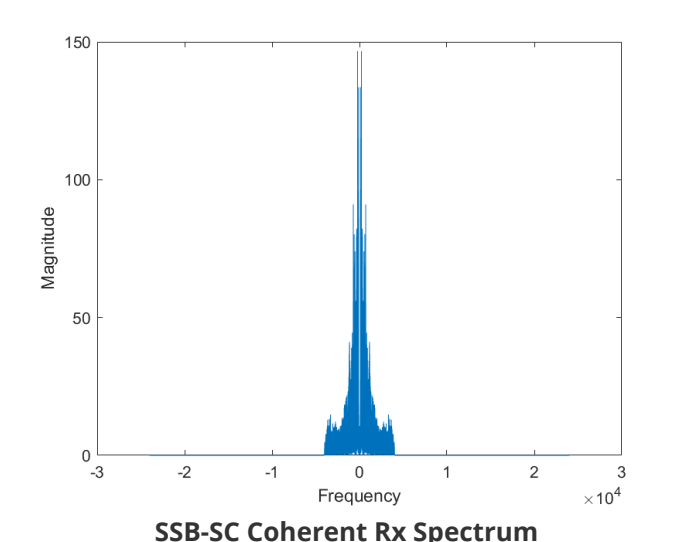
SSB TC Time



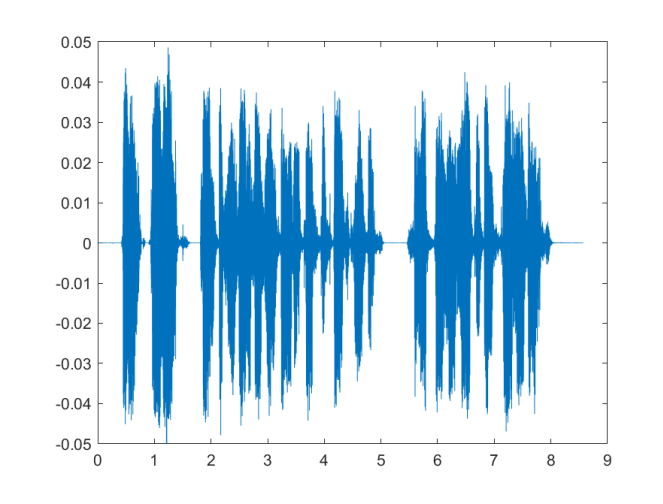
SSB Butterworth Spectrum



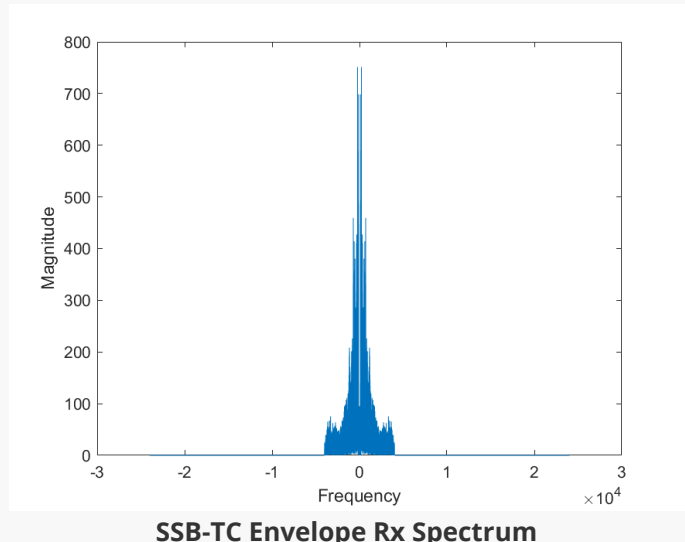
SSB Butterworth Time



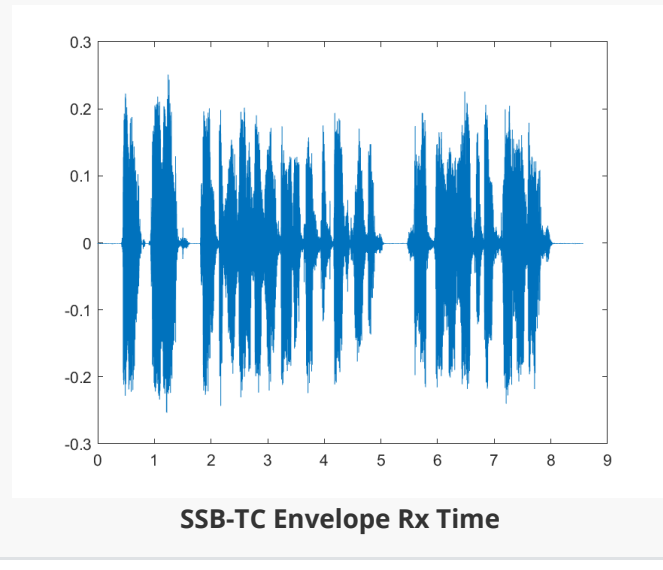
SSB-SC Coherent Rx Spectrum



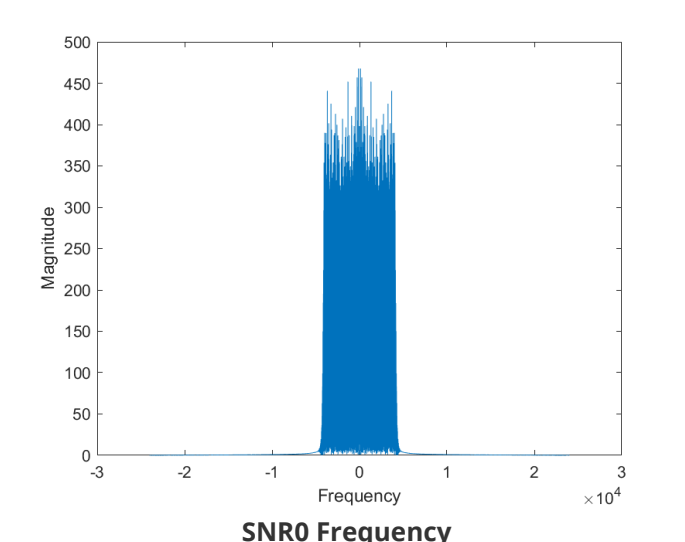
SSB-SC Coherent Rx Time



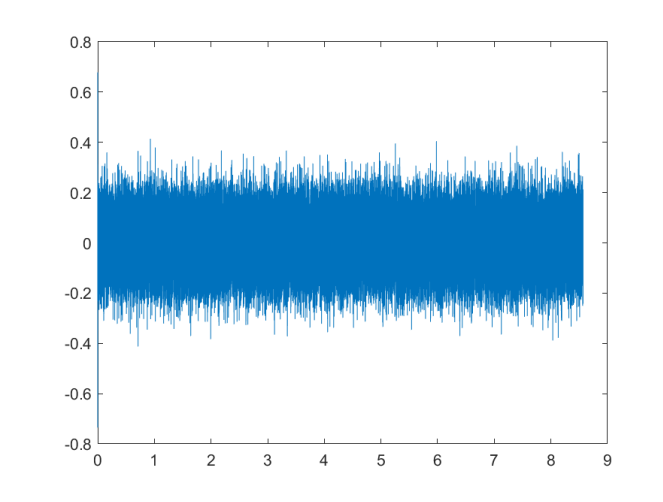
SSB-TC Envelope Rx Spectrum



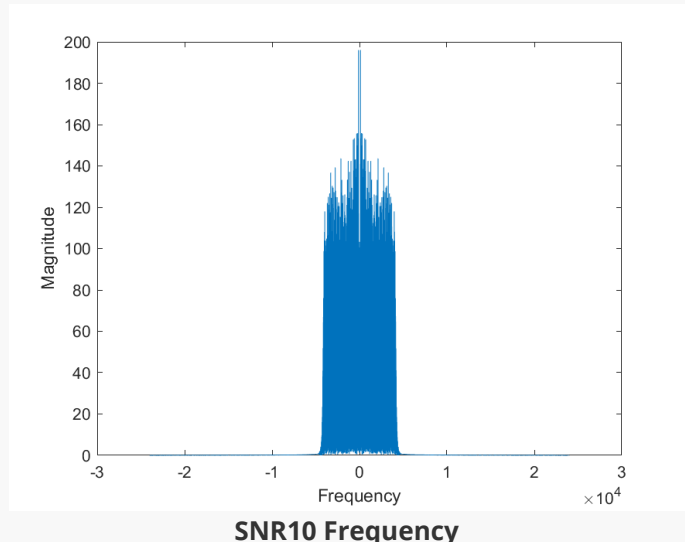
SSB-TC Envelope Rx Time



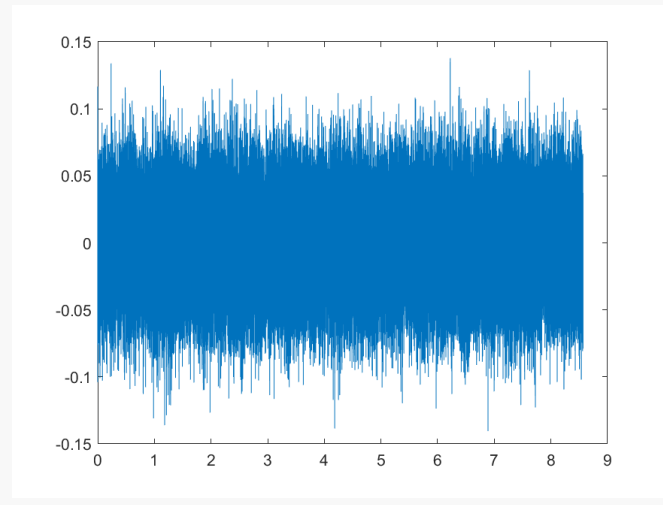
SNR0 Frequency



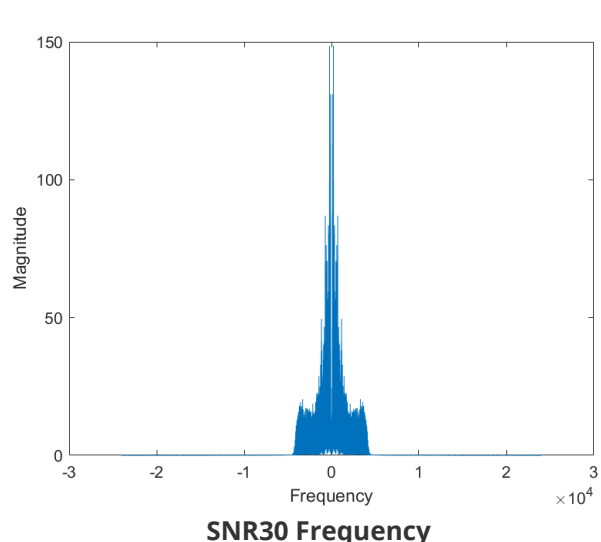
SNR0 Time



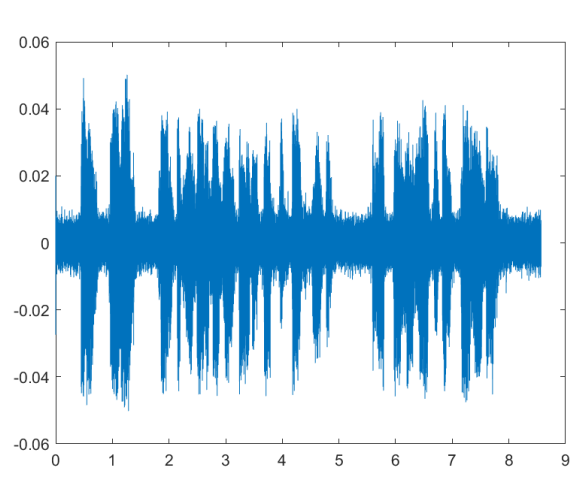
SNR10 Frequency



SNR10 Time



SNR30 Frequency



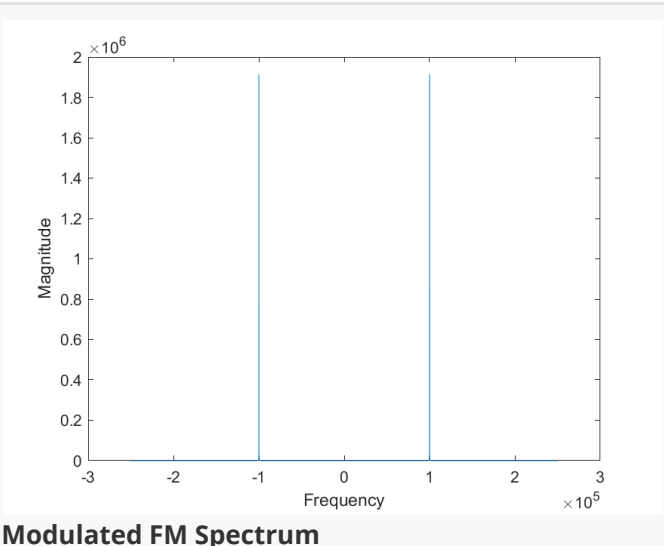
SNR30 Time

NBFM Modulation

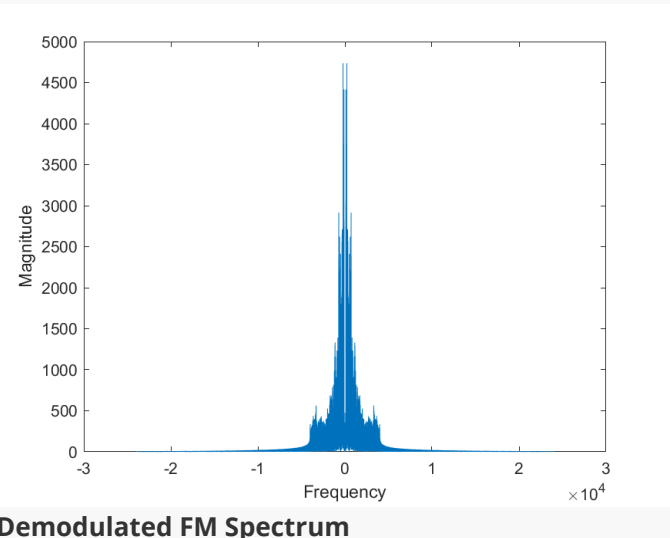
Modulation and Demodulation:

```
classdef NBFM
    methods(Static)
        function modulated = modFM(audio,fc,fs,kf)
            dsbobj = dsbmod(audio.filtered_data,1e5,audio.fs,5);
            % Integration of Msg
            int_M = cumsum(dsbobj.resampled_msg)/fs;
            t=linspace(0,length(int_M)/fs,length(int_M));
            % Generating FM Modded Msg
            modulated = cos(2*pi*fc*t'+ 2*pi*(kf)*int_M);
        end
        function demodulated = demodFM(moddedFM,fs,kf)
            [p,q] = rat(48/500);
            % Discriminator to Convert to AM
            differentiated = diff(moddedFM)*fs/kf/2*pi;%fs/kf/2*pi is a scaling factor
            differentiated = [0;differentiated];
            % Envelope Detection
            envelop=abs(hilbert(differentiated));
            % Remove DC
            demod = envelop - mean(envelop);
            % Remove High Freqs
            filter=generate_filter(length(demod),5e5,4e3);
            filtered=real(ifft(ifftshift(fftshift(fft(demod)).*filter)));
            % Downsample Again
            demodulated=resample(filtered,p,q);
        end
    end
end
```

Figures



Modulated FM Spectrum



Demodulated FM Spectrum

Conclusions for NBFM:

- Spectrum of NBFM takes the shape of DSB-TC
- For Narrow Band Modulation , the Modulation Index β should be less than 1

$$\beta = \frac{\Delta f}{f_m} << 1$$