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 **Analogue Communication Theory Final Lab Project** Original and Filtered Audio **DSB** Modulation Conclusions for DSB Modulation: SSB Modulation **NBFM Modulation** Conclusions for NBFM: **Original and Filtered Audio** 600 500 400 Magnitude 300 200 100 0 -3 -2 -1 2 $\times 10^4\,$ Frequency **Original Audio Spectrum** 500 0.05 400 300 -0.05 200 -0.1 100 -0.2 -0.25 **Filtered Audio Time Filtered Audio Spectrum 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); 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); 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** 2.5 3000 2500 2000 Magnitude 1500 1000 0.5 500 -2 -2 Frequency Frequency **DSB-SC DSB-TC** 0.25 0.2 0.1 0.05 -0.05 L -0.6 **Envelope of DSB SC Envelope of DSB TC** 500 450 0.4 400 350 0 250 -0.1 200 150 100 -0.4 50 -2 Frequency **SNR0 Time SNR0 Frequency** 400 350 0.15 300 0.05 250 0 200 -0.05 150 100 -0.15 -2 -1 Frequency **SNR10 Time SNR10 Frequency** 250 200 150 100 50 -0.1 ^{__}0 -2 **SNR30 Time SNR30 Frequency** 0.08 160 0.06 140 0.04 0.02 Magnitude 100 80 -0.02 60 -0.04 40 -0.06 -0.08 20 $\times 10^4$ **Frequency Drift in Time Frequency Drift in Spectrum** 180 160 140 0.04 120 0.02 100 80 -0.02 60 -0.04 40 -0.06 -0.08 20 -0.1 $\times 10^4$ **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); 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 2500 2000 0.05 1500 1000 Frequency **SSB SC Spectrum** 2.5 × 10⁶ 0 -0.5 -2 Frequency $\times 10^5$ **SSB TC Time SSB TC Spectrum** 2000 0.05 -0.05 500 -0.1 Frequency $\times 10^5$ **SSB Butterworth Time SSB Butterworth Spectrum** 0.05 0.04 0.03 0.02 0.01 Magnitude -0.01 50 -0.02 -0.03 -0.040 ^L -0.05 Frequency $\times 10^4$ **SSB-SC Coherent Rx Time** SSB-SC Coherent Rx Spectrum 700 0.2 0.1 500 400 300 -0.1 200 -0.2 100 -3 **SSB-TC Envelope Rx Time SSB-TC Envelope Rx Spectrum** 500 450 0.6 400 350 300 250 0 200 -0.2 150 100 -0.6 50 -0.8 -2 Frequency **SNR0 Time SNR0 Frequency** 200 180 160 140 0.05 120 100 80 -0.05 60 40 -0.1 20 -0.15 -2 **SNR10 Time SNR10 Frequency** 150 0.06 0.04 Magnitude 0 50 -0.04 -0.06 ^L Frequency $\times 10^4$ **SNR30 Time SNR30 Frequency 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); 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))); demodulated=resample(filtered,p,q); end end end

Figures

1.8

0.4

0.2

Modulated FM Spectrum

5000 4500

4000 3500

1500

1000

500

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

 $eta = rac{\Delta f}{f_m} << 1$