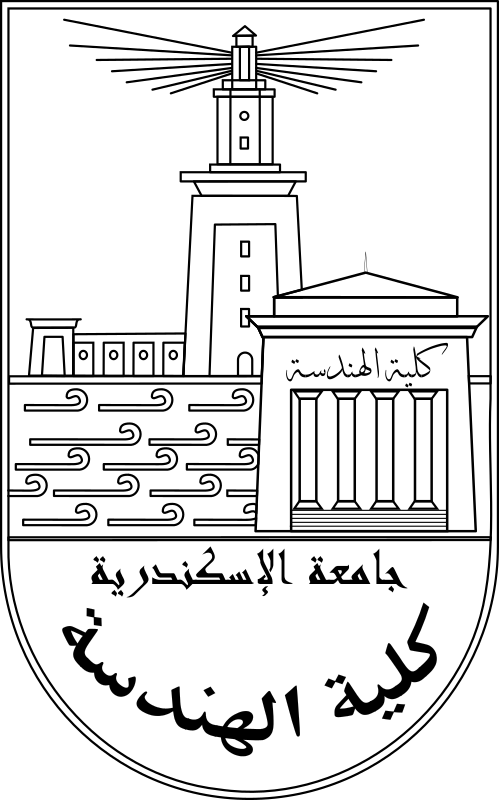
**Final Project - Analogue Communications**

Fadi Sarwat Farouk – 7432

Peter Anton Naguib – 7406

Yassin Medhat Tamam – 7574

Zeyad Ahmed Fathalla – 7621

Yamen Mohamed Saad – 7577

# Amplitude Modulation (DSB-TC and DSB-SC)

## Code

clear all;

close all;

[y, fs] = audioread('eric.wav');

Y = fftshift(fft(y));

f = linspace(-fs/2, fs/2, length(Y));

% Plot the spectrum

plot(f, abs(Y)/length(Y));

title('Spectrum of m');

figure();

% Apply filter

bw = 4000;

filt = ones(size(Y));

filt(f > bw|f<-bw) = 0;

y\_filter = Y .\* filt;

plot(f, abs(y\_filter)/length(y\_filter));

title('filtered signal spectrum');

% Inverse transform

y\_filtered\_time = ifft(ifftshift(y\_filter));

% Ensure data type and scaling

y\_filtered\_time = real(y\_filtered\_time); % Ensure real values

y\_filtered\_time = double(y\_filtered\_time); % Convert to double if necessary

%% Normalize if values are outside the range [-1, 1]

max\_val = max(abs(y\_filtered\_time));

if max\_val > 1

y\_filtered\_time = y\_filtered\_time / max\_val;

end

fc = 100000;

U = 0.5;

Am = max(y\_filtered\_time);

Ac = Am/U; %modulationindex = Am/Ac

new\_fs = 5 \* fc;

resampled\_signal = resample(y\_filtered\_time, new\_fs, fs);

t1 = linspace(0, length(resampled\_signal) / new\_fs, length(resampled\_signal));

t1 = t1';

carrier = Ac .\* cos(2 \* pi \* fc \* t1);

DSB\_SC = resampled\_signal .\* carrier;

DSB\_TC = (1 + U \* resampled\_signal/Am ) .\* carrier;

%% Plot DSB-SC spectrum

figure();

subplot(1,2,1)

DSB\_SC\_spectrum = fftshift(fft(DSB\_SC));

f\_DSB\_SC =linspace(- new\_fs/2, new\_fs/2, length(DSB\_SC));

plot(f\_DSB\_SC, abs(DSB\_SC\_spectrum)/length(DSB\_SC\_spectrum));

xlabel('Frequency (Hz)');

ylabel('Magnitude');

title('DSB-SC Modulated Signal Spectrum');

% Plot DSB-TC spectrum

subplot(1,2,2)

DSB\_TC\_spectrum = fftshift(fft(DSB\_TC));

f\_DSB\_TC=linspace(-new\_fs/2,new\_fs/2,length(DSB\_TC));

plot(f\_DSB\_TC, abs(DSB\_TC\_spectrum)/length(DSB\_TC\_spectrum));

xlabel('Frequency (Hz)');

ylabel('Magnitude');

title('DSB-TC Modulated Signal Spectrum');

%% envelop for DSB-SC

envelope\_DSB\_SC = abs(hilbert(DSB\_SC));

figure;

plot(t1, DSB\_SC);

hold on;

plot(t1,-envelope\_DSB\_SC,'r-',t1, envelope\_DSB\_SC,'-r','Linewidth',1.5);

hold off;

title('DSB\_sc with envellope');

ylim([-2 2]);

xlim([2 2.5]);

%envelop for DSB-TC

envelope\_DSB\_TC = abs(hilbert(DSB\_TC));

figure;

plot(t1, DSB\_TC);

hold on;

plot(t1, -envelope\_DSB\_TC ,'r-', t1, envelope\_DSB\_TC, '-r', 'Linewidth', 1.5);

hold off;

title('DSB\_Tc with envellope');

ylim([-2 2]);

xlim([2 2.5]);

demod\_DSB\_SC=envelope\_DSB\_SC.\*cos(2 \* pi \* fc \* t1);

%% plot demodulation signal

figure;

subplot(2, 1, 1);

plot(t1, envelope\_DSB\_TC);

xlabel('Time');

ylabel('Amplitude');

title('Received Signal - DSB-TC');

subplot(2, 1, 2);

plot(t1, demod\_DSB\_SC);

xlabel('Time');

ylabel('Amplitude');

title('Received Signal - DSB-SC');

% resample the envelope DSB\_SC to hear it

recived\_sig\_DSB\_SC = resample(envelope\_DSB\_SC, fc, fs);

% resample the envelope DSB\_TC to hear it

recived\_sig\_DSB\_TC = resample(envelope\_DSB\_TC, fc, fs);

%% sound the three msgs

sounds={y\_filtered\_time,envelope\_DSB\_SC,envelope\_DSB\_TC};

for i = 1:length(sounds)

sound(sounds{i}, fs);

pause(10);

end

%% Coherent detection

% when noise is added to DSB\_SC with SNR = 0, 10, and 30

snr\_values = [0, 10, 30];

for snr\_dB = snr\_values

% Add noise to DSB\_SC

noisy\_DSB\_SC = awgn(DSB\_SC, snr\_dB);

demodulated\_noisy = noisy\_DSB\_SC .\* cos(2\*pi\*fc\*t1);

f = new\_fs / 2 \* linspace(-1,1,length(DSB\_SC));

% Coherent detection with noise

demodulated\_noisy = double(real(demodulated\_noisy));

demodulated\_noisy = fftshift(fft(demodulated\_noisy));

demodulated\_noisy(f >= bw | f <= -bw) = 0;

demodulated\_noisy = ifft(ifftshift(demodulated\_noisy));

L = length(demodulated\_noisy);

% Plot received waveform and spectrum for each SNR value

figure;

subplot(2, 1, 1);

plot(t1, demodulated\_noisy);

title(['Received Signal with SNR = ' num2str(snr\_dB) ' dB - Time Domain']);

xlabel('Time (s)');

ylabel('Amplitude');

subplot(2, 1, 2);

plot(f, abs(fftshift(fft(demodulated\_noisy))) / L);

title(['Spectrum with SNR = ' num2str(snr\_dB) ' dB']);

xlabel('Frequency (Hz)');

ylabel('Magnitude');

demodulated\_noisy\_sound = resample(abs(demodulated\_noisy), fs, new\_fs);

sound(abs(demodulated\_noisy\_sound), fs);

pause(10);

end

%% coherent detection with frequency error

fc\_error=100100;

for snr\_dB = snr\_values

% Add noise to DSB\_SC

noisy\_DSB\_SC = awgn(DSB\_SC, snr\_dB);

demodulated\_noisy = noisy\_DSB\_SC .\* cos(2\*pi\*fc\_error\*t1);

f = new\_fs / 2 \* linspace(-1,1,length(DSB\_SC));

% Coherent detection with noise

demodulated\_noisy = double(real(demodulated\_noisy));

demodulated\_noisy = fftshift(fft(demodulated\_noisy));

demodulated\_noisy(f >= bw | f <= -bw) = 0;

demodulated\_noisy = ifft(ifftshift(demodulated\_noisy));

L = length(demodulated\_noisy);

% Plot received waveform and spectrum for each SNR value

figure;

subplot(2, 1, 1);

plot(t1, demodulated\_noisy);

title(['Received Signal with SNR (FE) = ' num2str(snr\_dB) ' dB - Time Domain']);

xlabel('Time (s)');

ylabel('Amplitude');

subplot(2, 1, 2);

plot(f, abs(fftshift(fft(demodulated\_noisy))) / L);

title(['Spectrum with SNR (FE) = ' num2str(snr\_dB) ' dB']);

xlabel('Frequency (Hz)');

ylabel('Magnitude');

demodulated\_noisy\_sound = resample(abs(demodulated\_noisy), fs, new\_fs);

sound(abs(demodulated\_noisy\_sound), fs);

pause(10);

end

%% phase error

for snr\_dB = snr\_values

% Add noise to DSB\_SC

noisy\_DSB\_SC = awgn(DSB\_SC, snr\_dB);

demodulated\_noisy = noisy\_DSB\_SC .\* cos(2\*pi\*fc\*t1+20);

f = new\_fs / 2 \* linspace(-1,1,length(DSB\_SC));

% Coherent detection with noise

demodulated\_noisy = double(real(demodulated\_noisy));

demodulated\_noisy = fftshift(fft(demodulated\_noisy));

demodulated\_noisy(f >= bw | f <= -bw) = 0;

demodulated\_noisy = ifft(ifftshift(demodulated\_noisy));

L = length(demodulated\_noisy);

% Plot received waveform and spectrum for each SNR value

figure;

subplot(2, 1, 1);

plot(t1, demodulated\_noisy);

title(['Received Signal with SNR (PE) = ' num2str(snr\_dB) ' dB - Time Domain']);

xlabel('Time (s)');

ylabel('Amplitude');

subplot(2, 1, 2);

plot(f, abs(fftshift(fft(demodulated\_noisy))) / L);

title(['Spectrum with SNR (PE) = ' num2str(snr\_dB) ' dB']);

xlabel('Frequency (Hz)');

ylabel('Magnitude');

demodulated\_noisy\_sound = resample(abs(demodulated\_noisy), fs, new\_fs);

sound(abs(demodulated\_noisy\_sound), fs);

pause(10);

end

## Outputs

|  |  |
| --- | --- |
|  |  |
|  | A screen shot of a graph  Description automatically generated |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

## Questions

**1) What observation can you make of this or which type of modulation the envelope detector can be used with?**

In the DSB-TC the envelop detection worked and it was clear since there is no phase reversals in the DSB-SC the envelop detection wasn't optimal since the modulation index is critical and the coherent detection for it worked better.

**2) Do you have a name for this phenomenon?**

Frequency Offset

Top of Form

Bottom of Form

# Single Side Band (SSB)

## Code

%% 1-2. Same as DSB

[y, fs] = audioread('eric.wav');

L = length(y);

Y = fftshift(fft(y));

f = linspace(-fs/2, fs/2, L);

% Plot the spectrum

figure;

subplot(2, 1, 1);

plot(f, abs(Y) / L);

title('Original Spectrum of m');

xlabel('Frequency (Hz)');

ylabel('Magnitude');

% Apply filter

bw = 4000;

Y(f >= bw | f <= -bw) = 0;

subplot(2, 1, 2);

plot(f, abs(Y) / L);

title('Filtered Spectrum of m');

xlabel('Frequency (Hz)');

ylabel('Magnitude');

%% 3. Filtered signal in time domain (Inverse transform)

y\_filtered\_time = ifft(ifftshift(Y));

t1 = linspace(0, length(y\_filtered\_time) / fs, length(y\_filtered\_time));

t1 = t1';

y\_filtered\_time = real(double(y\_filtered\_time));

figure; plot(t1, y\_filtered\_time);

title('Filtered Signal of m Time Domain');

xlabel('Time (s)');

ylabel('Amplitude');

fc = 100000;

U = 0.5;

Am = max(y\_filtered\_time);

Ac = Am/U; % modulationindex = Am/Ac

new\_fs = 5 \* fc;

%% 4. Plot DSB-SC spectrum

message\_for\_sound = resample(y\_filtered\_time, new\_fs, fs);

% sound(abs(message\_for\_sound), fs);

message = resample(y\_filtered\_time, new\_fs, fs);

t1 = linspace(0, length(message) / new\_fs, length(message));

t1 = t1';

L = length(message);

carrier = Ac .\* cos(2\*pi\*fc\*t1);

DSB\_SC = message .\* carrier;

DSB\_SC\_spectrum = fftshift(fft(DSB\_SC));

f\_DSB\_SC = new\_fs/2 \* linspace(-1, 1, length(DSB\_SC));

figure;

subplot(2, 1, 1);

plot(f\_DSB\_SC, abs(DSB\_SC\_spectrum) / L);

title('DSB-SC Modulated Signal Spectrum');

xlabel('Frequency (Hz)');

ylabel('Magnitude');

% 5. Obtain SSB\_LSB

SSB\_LSB = DSB\_SC;

L = length(SSB\_LSB);

f = new\_fs / 2 \* linspace(-1, 1, L);

F = fftshift(fft(SSB\_LSB));

F(f>=fc | f<=-fc) = 0;

SSB\_LSB = ifft(ifftshift(F));

% Plot SSB\_LSB spectrum

subplot(2, 1, 2);

plot(f, abs(F) / L);

title('SSB LSB Modulated Signal Spectrum');

xlabel('Frequency (Hz)');

ylabel('Magnitude');

%% 6. Coherent detection with no noise interference

demodulated\_signal\_ideal = SSB\_LSB .\* cos(2\*pi\*fc\*t1);

demodulated\_signal\_ideal\_sound = resample(abs(demodulated\_signal\_ideal), fs, new\_fs);

sound(abs(demodulated\_signal\_ideal\_sound), fs);

demodulated\_signal\_ideal = fftshift(fft(demodulated\_signal\_ideal));

demodulated\_signal\_ideal(f >= fc | f <= -fc) = 0;

demodulated\_signal\_ideal = ifft(ifftshift(demodulated\_signal\_ideal));

% Plot received waveform and spectrum

subplot(2, 1, 1);

plot(t1, demodulated\_signal\_ideal);

title('Received Signal (CD) in Time domain');

xlabel('Time (s)');

ylabel('Amplitude');

% Spectrum of received signal

L = length(demodulated\_signal\_ideal);

Y\_demodulated\_ideal = fftshift(fft(demodulated\_signal\_ideal));

f\_demodulated\_ideal = linspace(-new\_fs/2, new\_fs/2, L);

subplot(2, 1, 2);

plot(f\_demodulated\_ideal, abs(Y\_demodulated\_ideal) / L);

title('Spectrum of Received Signal (CD)');

xlabel('Frequency (Hz)');

ylabel('Magnitude');

xlim([-10000 10000]);

%% 7. Repeat steps 5 and 6 with Butterworth filter

[b, a] = butter(4, fc / (new\_fs / 2), 'low');

SSB\_LSB\_butter = filtfilt(b, a, demodulated\_signal\_ideal);

L = length(SSB\_LSB\_butter);

% Plot the spectrum of SSB-LSB signal using Butterworth filter

Y\_SSB\_LSB\_butter = fftshift(fft(SSB\_LSB\_butter));

f\_SSB\_LSB\_butter = linspace(-fs/2, fs/2, length(Y\_SSB\_LSB\_butter));

figure;

subplot(3, 1, 1);

plot(f\_SSB\_LSB\_butter, abs(Y\_SSB\_LSB\_butter) / L);

title('Spectrum of SSB-LSB Signal (Butterworth Filter)');

xlabel('Frequency (Hz)');

ylabel('Magnitude');

% Coherent detection with Butterworth filter

demodulated\_signal\_butter = SSB\_LSB\_butter .\* carrier;

t = linspace(0, length(demodulated\_signal\_ideal) / fs, length(demodulated\_signal\_ideal));

% Plot received waveform and spectrum

subplot(3, 1, 2);

plot(t, demodulated\_signal\_butter);

title('Received Signal (Coherent Detection with Butterworth) - Time Domain');

xlabel('Time (s)');

ylabel('Amplitude');

% Spectrum of received signal with Butterworth filter

Y\_demodulated\_butter = fftshift(fft(demodulated\_signal\_butter));

f\_demodulated\_butter = linspace(-fs/2, fs/2, length(Y\_demodulated\_butter));

subplot(3, 1, 3);

plot(f\_demodulated\_butter, abs(Y\_demodulated\_butter) / L);

title('Spectrum of Received Signal (Coherent Detection with Butterworth)');

xlabel('Frequency (Hz)');

ylabel('Magnitude');

%% 8. For the ideal filter case, get the received signal again with noise

% when noise is added to SSB-SC with SNR = 0, 10, and 30

snr\_values = [0, 10, 30];

for snr\_dB = snr\_values

% Add noise to SSB-SC

noisy\_SSB\_SC = awgn(SSB\_LSB, snr\_dB);

demodulated\_noisy = noisy\_SSB\_SC .\* cos(2\*pi\*fc\*t1);

% Coherent detection with noise

demodulated\_noisy = double(real(demodulated\_noisy));

demodulated\_noisy = fftshift(fft(demodulated\_noisy));

demodulated\_noisy(f >= bw | f <= -bw) = 0;

demodulated\_noisy = ifft(ifftshift(demodulated\_noisy));

f = new\_fs / 2 \* linspace(-1,1,L);

L = length(demodulated\_noisy);

% Plot received waveform and spectrum for each SNR value

figure;

subplot(2, 1, 1);

plot(t1, demodulated\_noisy);

title(['Received Signal with SNR = ' num2str(snr\_dB) ' dB - Time Domain']);

xlabel('Time (s)');

ylabel('Amplitude');

subplot(2, 1, 2);

plot(f, abs(fftshift(fft(demodulated\_noisy))) / L);

title(['Spectrum with SNR = ' num2str(snr\_dB) ' dB']);

xlabel('Frequency (Hz)');

ylabel('Magnitude');

demodulated\_noisy\_sound = resample(abs(demodulated\_noisy), fs, new\_fs);

sound(abs(demodulated\_noisy\_sound), fs);

end

%% 9. For the ideal filter case, generate a SSB-TC

SSB\_TC = carrier + SSB\_LSB;

L = length(SSB\_TC);

% Spectrum of received signal

Y\_SSB\_TC = fftshift(fft(SSB\_TC));

f\_SSB\_TC = linspace(-new\_fs/2, new\_fs/2, length(Y\_SSB\_TC));

figure;

plot(f\_SSB\_TC, abs(Y\_SSB\_TC) / L);

title('Spectrum of Received SSB-TC Signal');

xlabel('Frequency (Hz)');

ylabel('Magnitude');

% Envelope detection

envelope\_SSB\_TC = abs(hilbert(SSB\_TC));

% Plot received waveform

figure;

plot(t1, SSB\_TC, 'b', t1, -envelope\_SSB\_TC, 'r', t1, envelope\_SSB\_TC, 'r', 'Linewidth', 1.5);

title('Received Message with Envelope Detection');

xlabel('Time (s)');

ylabel('Amplitude');

legend('SSB-TC', 'Envelope');

ylim([-5 5]);

xlim([3 3.5]);

envelope\_SSB\_TC = resample(envelope\_SSB\_TC, fs, new\_fs);

sound(abs(envelope\_SSB\_TC), fs);

## Outputs

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

# Frequency Modulation (FM) Code

% 1. Read the audio file and find the spectrum

[audio, Fs] = audioread('eric.wav');

N = length(audio);

t = (0:N-1) / Fs;

% Calculate the spectrum

audio\_spectrum\_shifted = fftshift(fft(audio));

% Plot the spectrum

figure;

plot(linspace(-Fs/2, Fs/2, N), abs(audio\_spectrum\_shifted));

title('Original Spectrum');

xlabel('Frequency (Hz)');

ylabel('Magnitude');

% 2. Ideal filter to remove frequencies above 4 KHz

cutoff\_frequency = 4000;

filter = (abs(linspace(-Fs/2, Fs/2, N)) <= cutoff\_frequency);

audio\_spectrum\_filtered = audio\_spectrum\_shifted .\* filter';

% 3. Obtain filtered signal in time and frequency domain

audio\_filtered =real(ifft(ifftshift(audio\_spectrum\_filtered)));

% Plot the filtered spectrum

figure;

plot(linspace(-Fs/2, Fs/2, N), abs(audio\_spectrum\_filtered));

title('Filtered Spectrum');

xlabel('Frequency (Hz)');

ylabel('Magnitude');

% Plot the filtered signal in time domain

figure;

plot(t, audio\_filtered);

title('Filtered Signal in Time Domain');

xlabel('Time (s)');

ylabel('Amplitude');

% 4. Sound the filtered audio signal

sound(abs(audio\_filtered), Fs);

% 5. Generate NBFM signal

kf = 0.2/(2\*pi\*max(abs(cumsum(audio\_filtered)))./Fs);

Ac = 1;

Fc = 100000; % Carrier frequency

resampled\_audio = resample(audio\_filtered, 5\*Fc, Fs);

Fs\_nbfm = 5 \* Fc; % Sampling frequency for NBFM

%calculate time vector

tstart = 0;

tend = tstart + length(resampled\_audio) / Fs\_nbfm;

t1 = linspace(tstart, tend, length(resampled\_audio));

t1 = t1';

%FM modulated signal

NBFM\_signal = Ac \* cos(2\*pi\*Fc\*t1 + 2\*pi\*kf\*cumsum(resampled\_audio)./Fs\_nbfm);

% Plot the NBFM spectrum

L = length(NBFM\_signal);

NBFM\_spectrum\_shifted = real(fftshift(fft(NBFM\_signal)));

f = Fs/2\*linspace(-1,1,L);

figure;

plot(f, NBFM\_spectrum\_shifted)

title('NBFM Spectrum');

xlabel('Frequency (Hz)');

ylabel('Magnitude');

% 6. Condition for NBFM: Frequency deviation (delta\_f) should be much smaller than the message bandwidth (B)

B = cutoff\_frequency;

delta\_f = 75; % Frequency deviation

condition = delta\_f < B;

disp(['Condition for NBFM: ', num2str(condition)]);

% 7. Demodulate NBFM signal

% Discriminator

dy = diff(NBFM\_signal);

dy = [0; dy];

% envelope detector

demodulated\_NBFM = abs(hilbert(dy)) - mean(abs(hilbert(dy)));

% Plot the time-domain signal

figure;

plot(t1,demodulated\_NBFM);

title('Demodulated NBFM Time-Domain Signal');

xlabel('Time (s)');

ylabel('Amplitude');

ylim([-2\*10^-4 2\*10^-4]);

## Outputs

|  |  |
| --- | --- |
|  | A screen shot of a graph  Description automatically generated |
| A screen shot of a computer screen  Description automatically generated |  |
|  | **NBFM Zoomed in** |

## Questions

**1) What can you make out of the resulting plot?**

We realized that the Bandwidth=2fm since beta is very small (condition for NBFM).

**2) what is the condition we needed to achieve NBFM?**

To acheive NBFM we need to satisfy the condition in this equation BW= 2(beta+1) \* Fm that beta <= 0.2