John Renner

ELEC6603

Fall ‘24

An Interactive MATLAB Model for Simulating and Demonstrating Amplitude Modulation

INTRODUCTION

This project aims to design and implement a prototype radio transmission and reception system using Amplitude Modulation (AM) in MATLAB. The system will simulate the AM process by allowing the user to record a short message, which will then be modulated, demodulated, and played back. Various parameters such as amplitude, carrier frequency, and noise level will be adjustable, giving users control over the system's behavior. Additionally, the system will provide visual outputs, including time-domain and frequency-domain plots, along with performance metrics such as Signal-to-Noise Ratio (SNR), Peak Signal-to-Noise Ratio (PSNR), Mean Squared Error (MSE), and correlation, to demonstrate the accuracy of the demodulation process. This prototype will serve as an interactive tool to illustrate how AM works and how different factors, such as modulation error and noise, can impact the quality of the transmitted signal.

BACKGROUND/LITERATURE REVIEW

Amplitude Modulation is a technique where the amplitude of a high frequency carrier signal is varied according to a lower frequency message signal. The modulated signal can then be transmitted over a distance via radio waves, where it can then be captured and demodulated to retrieve the original message signal.[1]

Amplitude modulation is governed by the following equation:

$$y(t) = x(t)\cos(2w_o\pi t)$$

Where $x(t)$ is the message signal, w_o is the carrier frequency, and $y(t)$ is the modulated signal. Using the modulation property of the Fourier Transform, we can calculate

$$Y(w) = F\{y(t)\} = \frac{1}{2}(X(w - w_o) + X(w + w_o))$$

This reveals an interesting spectral property of the amplitude modulation function. Modulating a carrier signal of frequency w_o creates two spectral copies of the original message centered around $+w_o$ and $-w_o$. These are known as sidebands.

We can use the same modulation property to demodulate or signal. If we again multiply our modulated signal by the carrier signal, we get:

$$z(t) = y(t)\cos(2wopit)$$

Again, using the modulation property:

$$Z(w) = F\{z(t)\} = \frac{1}{2}\big(Y(w - wo) + Y(w + wo)\big)$$

$$Z(w) = \frac{1}{2}\left(\frac{1}{2}\big(X(w - 2wo) + X(w)\big) + \frac{1}{2}\big(X(w + 2wo) + X(w)\big)\right)$$

$$Z(w) = \frac{1}{2}X(w) + \frac{1}{4}\big(X(w - 2wo) + X(w + 2wo)\big)$$

Now we can see that we have a signal consisting of the original message at ½ amplitude, plus side bands located at $+2w_o$ and$-2w_o$. From here, all we need to do to retrieve our original signal is apply a filter to remove the sidebands.[2]

When working with Amplitude Modulation, we need to consider the modulation index given by $\mu = \frac{|A_x|}{|A_c|}$ where A_x and A_c are the amplitudes of the message and carrier signals, respectively. Values of μ greater than 1 result in overmodulation and distortion, while small values of μ close to 0 can result in a weakened signal.[1] [3] [4]

When creating the model, we also need to consider the Nyquist Theorem, a fundamental principle of signal processing that establishes the minimum sample rate needed to accurately reconstruct a continuous signal from discrete samples. According to the theorem, the sampling rate must be at least twice the highest frequency present in the signal. Sampling below this rate can lead to aliasing, where frequencies that are too high to be captures accurately manifest as lower frequencies not present in the original signal.[2] [5]

Once we've completed our simulation, there are a few different tools that we will use to evaluate the quality of our demodulated signal and how closely it represents the original message. The first tool is the correlation between the two signals. The correlation between the two signals is a representation of how similar the two signals are to each other, and is normalized on the interval (-1,1) where values close to 1 indicate closely correlated signals, values close to 0 indicate loosely correlated signals, and negative values indicate inversely correlated signals. The correlation of two signals can be found using MATLAB's corrcoef() command.[6]

In addition to correlation, we will also look at the mean squared error (MSE). The MSE provides another figure for how closely the signals match each other by taking the difference between the signals at each sample point, squaring the difference, and then averaging these together.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(x[n]c[n]$$

We will also evaluate the Signal to Noise Ratio (SNR) and Peak Signal to Noise Ratio (PSNR) of our demodulated signal. The SNR is the ratio between the signal level and the noise level.[7] PSNR is similar, but looks at the peak value of the signal.[8]

$$SNR = 10\log_{10}(\frac{P_{signal}}{P_{noise}})$$

$$PSNR = 10\log_{10}(\frac{A_{peak}}{MSE})$$

METHODOLOGY

The following parameters will be set by the user prior to running the code, so they will be defined within the first block of code: Carrier frequency, carrier amplitude, recorded message length, message amplitude, upper frequency limit of the message signal, level of added noise. Based on these parameters, we can then determine the necessary sampling frequency. The sampling frequency will need to account for the upper side band of the demodulation signal, which will be two times the carrier frequency, plus the upper frequency limit of the message. Therefore, the sampling frequency will need to be at least two times this limit, and ideally slightly higher. However, the higher the sampling frequency, the more samples will be required for computation, and the more computationally costly the program will become. Therefore, it's best to keep the carrier frequency, as well as the length of the message being recorded, should be minimized as much as possible.

```
%% _____USER DEFINED PARAMETERS_____
messageLength = 3; % Message time in seconds
fc = 50000; % Carrier Frequency
message_flimit = 10000;
message_amp = .9;
carrier_amp = 1;
noise_factor = 0;
%% _____AXES, CREATE CARRIER/NOISE_____
fs = 2*(fc*2+message_flimit)*1.2; % Sampling frequency
n = fs * messageLength; % Number of samples
t = (0:n-1)*(1/fs); % time axis
w = fs * (-n/2:(n/2)-1) / n; % 2-sided freq axis
carrier = carrier_amp * cos(fc*2*pi*t); % Create Carrier signal
noise = noise_factor*randn(1, n)/100; % Create Noise Signal
```

Moving onto the input side of our model, we then need to generate a message signal and a carrier signal, and then multiply them. The carrier signal will consist of a cosine function, and will incorporate the user defined parameters mentioned above. Creating the message signal is more involved, as we will need to record and capture audio. However, recording audio

at the sample rate determined above would be impractical and inefficient, as that sample rate is may lie in the order of 10^5. Additionally, the signal for our model is being recorded on a low-quality microphone, the bandwidth is likely to be even less. Therefore, we will record audio at a lower sample rate, and then up sample the signal in order to be able to display on the same axes as our other signals. However, we will also apply a band pass filter with an upper cutoff frequency settable by the user.

```
%% ___________Audio Recording_____________________________________________
recObj = audiorecorder(12000, 16, 1); % 12kHz, 16 bit, 1 channel
disp("SBegin recording.")
recordblocking(recObj, messageLength);
disp("End of recording.")
disp("Processing audio...")
audio = getaudiodata(recObj);
audio = (resample(audio, fs, 12000))'; % Resample audio at fs
audio = audio * message_amp / max(audio); % normalize signal based on user-defined
modulation index
message = bandpass(audio, [100 message_flimit], fs); % Audio Signal
```

Once the message and carrier signals have been created, the carrier will be modulated by multiplying the amplitudes of the two signals together. This signal now represents the radio wave that will physically transmit the message. We will then add a small amount of noise to this signal using MATLAB's randn() function, which approximates a Gaussian distribution.

To demodulate the signal, we will multiply our modulated signal by the carrier. We will then pass this signal through another band pass filter, whose upper cutoff frequency matches that of the input filter. At this point, the entire process is complete, and we can generate visual or audio samples to compare the demodulated message with the original and help visualize how the process occurred.

```
%% _________________Modulation Function_____________________________________
disp("Performing Amplitude Modulation/Demodulation...")
mod_signal = (message).*carrier + noise; % Modulate Carrier signal
mod_signal = bandpass(mod_signal, [(fc-message_flimit), (fc+message_flimit)], fs);
mod_signal_fourier = fftshift(abs(fft(mod_signal))/n);
demod_signal = 2*mod_signal.*carrier; % Demodulate
demod_signal_fourier = fftshift(abs(fft(demod_signal))/n);
demod_signal = bandpass(demod_signal, [100 message_flimit], fs); % Retrieve original
signal
disp("Calculating FFTs...")
in_fourier = fftshift(abs(fft(message))/n); % Input FFT
in_fourier = smoothdata(in_fourier, "movmean", 20); % smooth data
in_fourier = 20*log(in_fourier);
out_fourier = fftshift(abs(fft(demod_signal))/n); % Output FFT
out_fourier = smoothdata(out_fourier, "movmean", 20); % smooth data
out_fourier = 20*log(out_fourier);
```

In addition to playing back both the original message and the demodulated signal so that we can hear the similarities and differences between the two, we will also generate time-domain

and frequency-domain plots of the original message and the demodulated message, which will allow us to visually evaluate how closely these signals match each other. We will generate “close-up” waveforms of the original message signal, the carrier signal, the modulated signal, and the demodulated signal, which will illustrate how the message signal is encoded onto the carrier signal. We will also generate a frequency-domain plot of the modulated and demodulated signals in order to demonstrate the sidebands created by the process.

```
%% _________________PLAYBACK_____________________________
message_downsample = resample(message, 12000, fs);
demod_signal_downsample = resample(demod_signal, 12000, fs);
disp("Playing Original Message...")
soundsc(message_downsample, 12000)
pause(messageLength + 2)
disp("Playing Demodulated Message...")
soundsc(demod_signal_downsample, 12000)
```

Finally, we will calculate the SNR, PSNR, correlation coefficient, and MSE of the demodulated signal, as well as the modulation index of the modulation process. These will provide subjective measurements for evaluating the performance of the system. SNR and PSNR will provide insight into how the system responds to different levels of noise, while correlation coefficient and MSE will gauge how accurately our demodulated signal approximates the original signal. Finally, modulation index will be calculated to ensure we’re not introducing any modulation error.

```
%% _____________________ANALYSIS_____________________________

disp('*********************************************')
message_power = mean(message.^2);
noise_power = mean((message - demod_signal).^2);
snr_value = 10 * log10(message_power / noise_power);
disp(['SNR (Demodulated Signal): ', num2str(snr_value), ' dB']);
mse_value = mean((message - demod_signal).^2);
disp(['MSE: ', num2str(mse_value)]);
correlation = corrcoef(message, demod_signal);
correlation_value = correlation(1,2);
disp(['Correlation Coefficient: ', correlation_value]);
peak_amplitude = max(abs(message));
psnr_value = 10 * log10(peak_amplitude^2 / mse_value);
disp(['PSNR: ', num2str(psnr_value), ' dB']);
mod_index = max(abs(message))/max(abs(carrier+noise));
disp(['Modulation Index: ', num2str(mod_index)]);
```

The visual plots, calculations, and buttons to initiate playback will all be programmed to appear within a windowed figure so that we can review the results after running the code.

RESULTS

This section presents the results of the amplitude modulation model when simulated under varying parameters. The different simulations and their associated parameters can be

found in the table below, along with figures for SNR, MSE, etc. This section also includes screenshots and some analysis of specific simulations.

File Name	Message Amplitude	Carrier Amplitude	Carrier Frequency (kHz)	Noise Factor	Message Upper Freq Limit (kHz)	Correlation Coefficient	Mean Squared Error (MSE)	Signal-Noise-Ratio (SNR) (dB)	Peak Signal-Noise-Ratio (PSNR) (dB)	Mod Index
Control	0.75	1	50	5	5	1	0.0000	37.60	53.03	0.71
noise	0.75	1	50	10	5	0.97	0.0008	12.58	29.20	0.54
noise2	0.75	1	50	30	5	0.79	0.007	2.23	20.79	0.38
modErrorHigh	1	0.5	50	0	5	1	0.0085	2.50	20.72	2.00
modErrorLow	0.15	1	50	0	5	1	0.0000	41.10	57.89	0.16
overlappingBands	0.75	1	8	0	7	1	0.0000	41.82	60.24	0.74
overlappingBands2	0.75	1	3	0	2.5	1	0.0000	43.08	60.51	0.75
Mix	0.4	1	30	3	5	0.86	0.0008	4.62	24.85	0.41

Table 1. Simulation Parameters and Resultant Calculations

The “Control” simulation file represents typical operation of the model with no error introduced. The demodulated signal, when played back, sounds like an accurate representation of the original message. This is confirmed by the time-domain and frequency domain plots, where the output closely tracks the input, as well as the correlation coefficient, MSE, and SNR (1.00, 0.0000, and 37.60 dB respectively.) The close-up time-domain plots also provide a useful tool for visualizing the AM process, particularly the time-domain plot that shows the original message and demodulated message superimposed onto the modulated carrier signal.

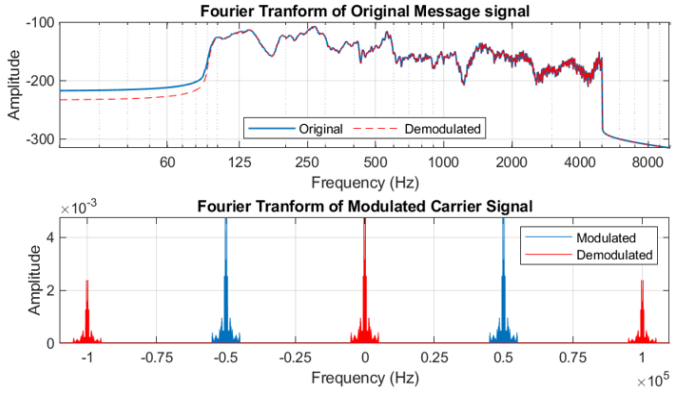


Figure 2. "Control" Frequency Plots

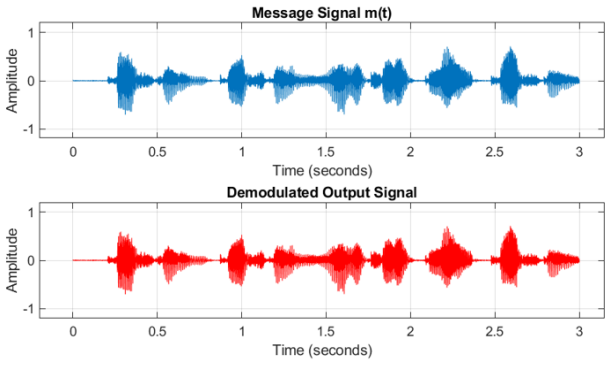


Figure 1. "Control" Time Plots

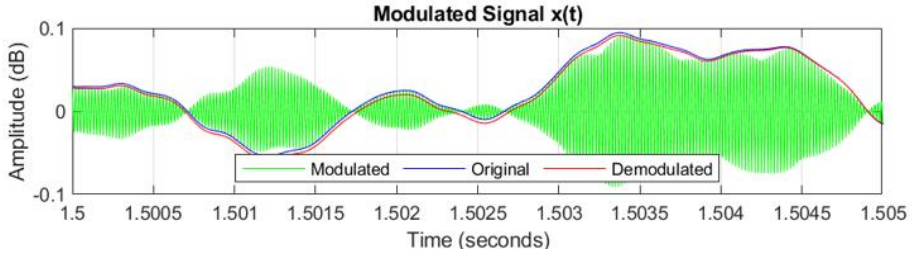
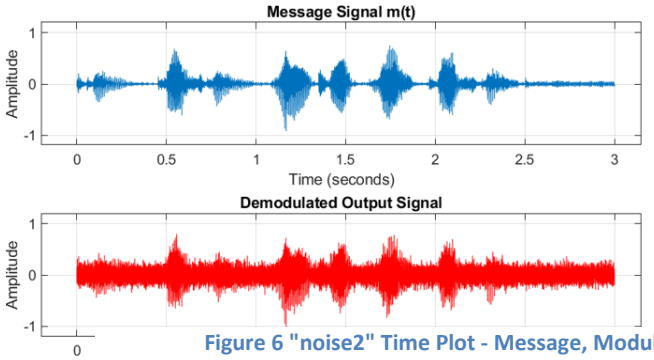


Figure 3 "Control" Time Plot - Message, Modulated Carrier, and Demodulated Output Signal

The “noise” and “noise2” simulations introduce small amounts of noise to the carrier. The “noise” simulation uses a noise factor of 10, while “noise2” uses a noise factor of 30. The noise severely degraded the signal, as evidenced by a correlation coefficient of 0.79 and a SNR of 2.23 dB for “noise2”. When played back, the original signal is audible above the noise floor, but only barely. The frequency-domain plot and time-domain plot reveal this as well.



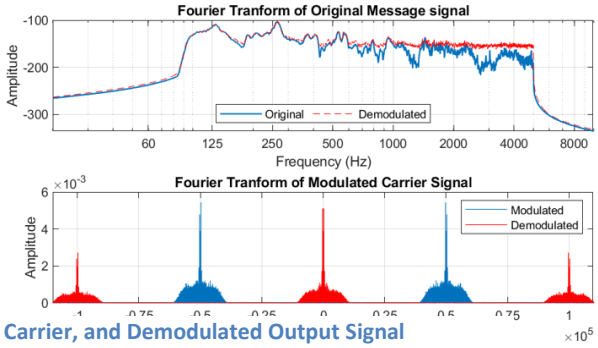
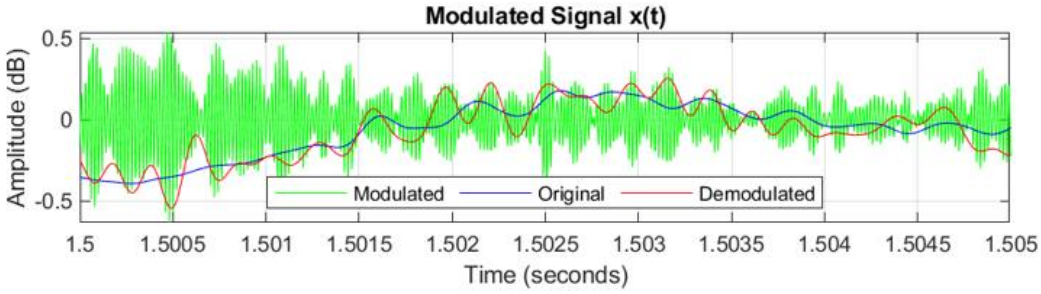


Figure 6 "noise2" Time Plot - Message, Modulated Carrier, and Demodulated Output Signal

Figure 5. "noise2" Time Plots

Figure 4 "noise2" Frequency Plots



The “modErrorLow” and “modErrorHigh” files seek to introduce modulation error into the process by changing the relationship between the message and carrier amplitudes, and therefore the modulation index. In “modErrorLow”, the message and carrier amplitudes are set to 0.15 and 1.00 respectively to attain a modulation index of 2.00, while in “modErrorHigh” they are set to 1.00 and 0.50 respectively to attain a modulation index of 0.15 However, this only resulted in a MSE of 0.0085 for the “modErrorHigh” simulation, and a MSE of 0.0000 for the

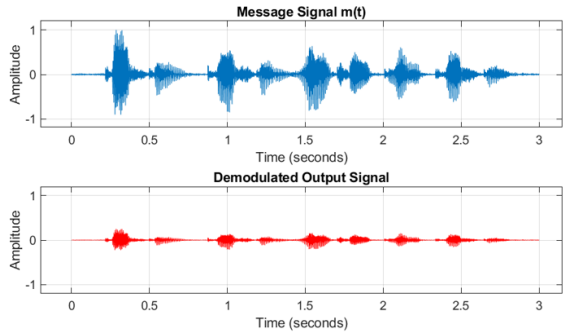


Figure 8. "modErrorHigh" Time Plots

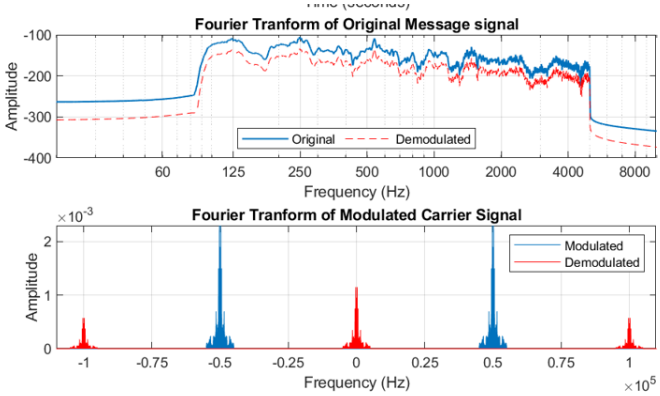


Figure 7 "modErrorHigh" Frequency Plots

"modErroLow" simulation. Both simulations resulted in a correlation coefficient of 1.00. The "modErrorHigh" simulation had a SNR of 2.50 dB, which was similar to that of "noise2" while "modErrorLow" had an SNR of 41.10 dB, closer to that of "Control." However, these distortions are not readily apparent when listening to the playback of the files, and the waveforms, despite appearing at different amplitudes due to the large difference in carrier and message amplitude, appear similar.

The "overlappingBands" and "overlappingBands2" simulations explore what happens when the carrier frequency is reduced low enough that the sidebands begin to overlap with each other. Carrier frequencies of 8 kHz and 3 Khz were used. The frequency-domain plots below illustrate the overlap. However, no significant distortion is audible when listening to the playback of the files, and the correlation coefficient, MSE, SNR, and PSNR for these simulations is similar to that of the "Control" simulation.

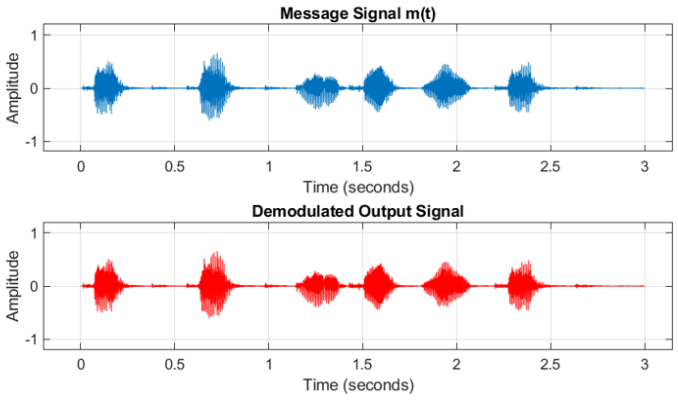


Figure 9 "overlappingBands2" Time Plot

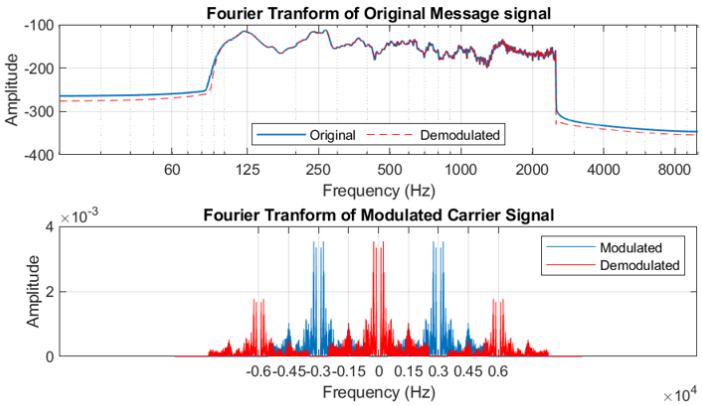


Figure 10 "overlappingBands2" Frequency Plot

Finally, the "mix" simulation mixes elements of the previous simulations, including noise and overlapping sidebands.

CONCLUSION

The model described in this paper provides an interactive and illustrative tool for exploring and understanding amplitude modulation. However, in certain cases the model did not work as expected. Introducing a modulation index greater than 1 did not result in any noticeable distortion in the “modErrorHigh” simulation. Additionally, lowering the carrier frequency to the point where the sidebands begin to overlap did not cause any noticeable distortion either. In the future, the model could be expanded to include different carrier signals at different frequencies modulated by different message signals, to see if any crosstalk occurs as a result of the sideband overlap.

Furthermore, a more rigorous analysis could be done to better evaluate the performance of the system. Due to the way the simulation was coded, a different message signal was recorded and used for each simulation. A more rigorous analysis would use the same recording for each simulation to eliminate differences in the recordings impacting the results. Also, more parameters could be identified for measurement, such as total harmonic distortion.

REFERENCES

[1] "Amplitude Modulation Theory: Time Domain, Frequency Domain," All About Circuits. Accessed: Dec. 6, 2024. [Online]. Available: https://www.allaboutcircuits.com/textbook/radio-frequency-analysis-design/radio frequency-modulation/amplitude-modulation-theory-time-domain-frequency-domain/

[2] M. N. O. Sadiku and W. H. Ali, Signals and Systems: A Primer with MATLAB. Boca Raton, FL, USA: CRC Press, 2016.

[3] "Amplitude and Frequency Modulation," Michigan State University Energy Management Research Group. Accessed: Dec. 6, 2024. [Online].
Available: https://www.egr.msu.edu/emrg/sites/default/files/content/module7_am_fm.pdf

[4] "Amplitude Modulation," ScienceDirect. Accessed: Dec. 6, 2024. [Online].
Available: https://www.sciencedirect.com/topics/engineering/amplitude-modulation

[5] R. D. Ziegler, "Nyquist-Shannon Theorem: Understanding Sampled Systems," *All About Circuits*, [Online]. Available: https://www.allaboutcircuits.com/technical-articles/nyquist-shannon-theorem-understanding-sampled-systems/.

[6] E. I. George, “EECS 206 Laboratory 2: Exploring Signals with MATLAB,” University of Michigan, Ann Arbor, MI, USA. [Online]. Available:
https://www.eecs.umich.edu/courses/eecs206/public/lab/lab2/lab2.pdf

[7] J. F. Afolayan and O. A. Olatunbosun, "Performance Analysis of Amplitude Modulation Scheme for Communication System," International Journal of Trend in Research and Development (IJTRD), pp. 1–6, 2019. [Online].
Available: https://www.ijtrd.com/papers/IJTRD22710.pdf

[8] NextPCB. (n.d.). *What is signal to noise ratio and how to calculate it?* NextPCB. [Online]. Available: https://www.nextpcb.com/blog/what-is-signal-to-noise-ratio-and-how-to-calculate-it.

[9] National Instruments. (n.d.). *Peak signal-to-noise ratio as an image quality metric*. National Instruments. [Online]. Available: https://www.ni.com/en/shop/data-acquisition-and-control/add-ons-for-data-acquisition-and-control/what-is-vision-development-module/peak-signal-to-noise-ratio-as-an-image-quality-metric.html.