ZJU-UIUC INSTITUTE
Zhejiang University-University of Illinois at Urbana-Champaign Institute
浙江大学伊利诺伊大学厄巴纳香槟校区联合学院

ECE 459

COMMUNICATIONS SYSTEMS

Final Project (Fall 2023)

---

# PERFORMANCE ANALYSIS OF AMPLITUDE AND FREQUENCY MODULATION IN NOISE

---


**Group 5**

| Name | Student ID | Grade |
|---|---|---|
| Yiqin LI | 3200110000 | |
| Peidong YANG | 3200110000 | |
| Zhuohao XU | 3200110000 | |
| Rongjian CHEN | 3200110000 | |
| Tiantian ZHONG | 3200110000 | |

**Instructor**

Prof. Said MIKKI and Prof. Juan ALVAREZ


January 10, 2024

# Abstract

**Keywords** Amplitude Modulation, Frequency Modulation, Noise Analysis

# Contents

# 1 Introduction

## 1.1 The Background

The history of modulation techniques dates back to the early days of radio communication when Amplitude Modulation emerged as the pioneering method. Over time, Frequency Modulation gained prominence due to its resilience against noise and superior audio quality, particularly in broadcasting and mobile communication [1, p. 152].

Amplitude Modulation (AM) is a communication technique that transmits messages by modulating the amplitude of a radio frequency (RF) wave. This modulation is achieved through the combination of the message signal with a high-frequency carrier wave. The resulting modulated waves can be demodulated using either coherent detectors or envelope detectors.

Frequency Modulation (FM), classified as a form of Angle Modulation, involves integrating the message into the phase of an RF signal. Demodulation of the FM signal can be accomplished through the utilization of differentiators or slope circuits.

Both AM and FM present distinct advantages and trade-offs, necessitating a comprehensive performance analysis. Such an analysis is crucial for a thorough understanding of their strengths and limitations within contemporary communication systems.

## 1.2 Objectives and Purposes

This project is designed to conduct a comprehensive analysis of the performance of Frequency Modulation (FM) and Amplitude Modulation (AM) communication systems in the presence of noise. The methodology involves constructing an envelope-modulated AM and narrow-band FM communication system, with the subsequent application of Additive White Gaussian Noise (AWGN) to the system. Specifically, the demodulation of AM signals is to be carried out using envelope detectors.

The team is tasked with simulating the modulation and demodulation processes for both systems and subsequently comparing the spectra and waveforms of the message signals and demodulated signals. The chosen message signals encompass both a multi-tone message and a Text-to-Speech (TTS)-generated voice recording. Theoretical and experimental pre- and post-detection signal-to-noise ratios (SNRs) are to be obtained to facilitate a comprehensive performance analysis.

Furthermore, the team is required to meticulously observe disparities between input and output signals, as well as their spectra, in order to conclude the distinctive characteristics of AM and FM modulation. A comparative analysis of anti-noise performance is also imperative, involving the measurement of the signal-to-noise ratio (SNR). The evaluation of simulation performance itself is to be conducted by comparing theoretical and experimental pre- and post-detection SNRs.

This project aims to equip the team with an in-depth understanding of both the modulation and demodulation processes, enabling a nuanced comprehension of the intricacies involved in implementing communication systems using Python. Specifically, the team is expected to demonstrate proficiency in performing Fourier Transform and Hilbert Transform, as well as implementing filters and envelope detectors.

## 1.3 Literature Review

The textbook by Haykin and Moher [1, Sec. 3.1] presents an intuitive method of envelope modulation. In this approach, the modulated signal is derived by combining a carrier wave with an amplified and DC-shifted message signal. For the demodulation process, the team references Haykin's work, particularly [1, Fig. 9.8]. Ulrich [2] introduces an alternative method of envelope detection utilizing Hilbert Transformation. This technique, when employed in conjunction with the `scipy.signal` package, streamlines the implementation of envelope detector design.

The Direct Method of Frequency Modulation (FM) signal generation, as illustrated in [1, Fig. 4.7], involves components such as an integrator, a phase modulator, and a local oscillator. The demodulation process for FM is also elucidated in the same textbook, specifically in [1, Fig. 9.13].

The realization of ideal filters poses challenges due to the discontinuous frequency response. However, the Butterworth filter offers a practical solution for simulating real-world filters, as discussed in the works of Storr [3] and Khetarpal et al. [4]. Implementation of the Butterworth filter can be achieved using the `scipy.signal` package.

# 2 Methodology

The project simulates the AM and FM communication systems respectively which applies Python 3. The work is done using Jupyter Notebook. Packages imported include `numpy` for numerical computation, `scipy` for signal analysis, and `matplotlib` for plotting.

## 2.1 Choice of Message Signal

Two message signals are tested in this project:

1. The multi-tone sinusoidal signal $m_1(t) = A_1 \cos(2\pi f_1 t + \phi_1) + A_2 \cos(2\pi f_2 t + \phi_2)$.
2. The TTS-generated male voice recording.

The choice of the multi-tone signal lies in its simplicity as a periodic function and special properties as a linear combination of two sinusoidal functions. Such choice allows the team to observe the distortion by noise in an easier way, as the sinusoidal functions have relatively simple frequency spectrums. The male voice recording enables the team to practice the designed communication system in a more realistic scenario.

Generating the multi-tone signal can be achieved by Python expression

```
t = np.arange(0, duration, 1/fs) # time vector
message = A1 * np.cos(2 * np.pi * f1 * t + phi1) + A2 * np.cos(2 * np.pi * f2 * t + phi2)
```

and generating the voice signal from a wave audio file can be achieved by

```
fs, audio_data = wavfile.read('audio.wav') # fs is the sampling rate of the audio file
t = np.arange(0, len(data)/fs, 1/fs) # time vector
```

## 2.2 AM Simulation

## 2.3 Envelope Modulation

The modulation of a message signal, denoted as $m(t)$, can be achieved through envelope modulation, represented by the equation:

$$s(t) = A_c[1 + k_a m(t)] \cos(2\pi f_c t) \tag{2.1}$$

In this expression, $k_a$ denotes the modulation sensitivity, $A_c$ corresponds to the amplitude of the carrier wave, and $f_c$ represents the frequency of the carrier wave. It is imperative to ensure that the carrier wave frequency $f_c$ significantly surpasses the highest frequency component, denoted as $W$, of the message signal $m(t)$ to prevent aliasing. This condition can be expressed as $f_c \gg W$. Moreover, the choice of the modulation sensitivity, $k_a$, needs to adhere to the constraint:

$$|k_a m(t)| < 1, \quad \text{for all } t \tag{2.2}$$

This condition is crucial to prevent envelope distortion, as outlined by Haykin and Moher [1, pp. 101-102].

The implementation of amplitude modulation (AM) can be illustrated through the block diagram depicted in Figure 2.1. For practical implementation in Python, the AM modulation process can be realized using the following code snippet:

```
am_signal = Ac*(1 + ka*data) * np.cos(2*np.pi*fc*t)
```

Here, `data` refers to the array containing samples of the message signal, and `ka` denotes the modulation sensitivity.

### 2.3.1 Envelope Detection

### 2.3.2 SNR Calculation and Measurement

## 2.4 FM Simulation

### 2.4.1 Narrow-Band FM Modulation

### 2.4.2 Narrow-Band FM Demodulation

### 2.4.3 SNR Calculation and Measurement

## 2.5 Filter Design

### 2.5.1 Ideal Filters

### 2.5.2 Butterworth Filter

Both AM and FM communication systems require filters. Yet in real life, ideal filters are not implementable. However, the Butterworth Filter can behave closely to the ideal filter. According to [3], [5], a Butterworth Low Pass Filter has a maximum flat frequency response within its passband and rolls off quickly at the cut-off frequency.

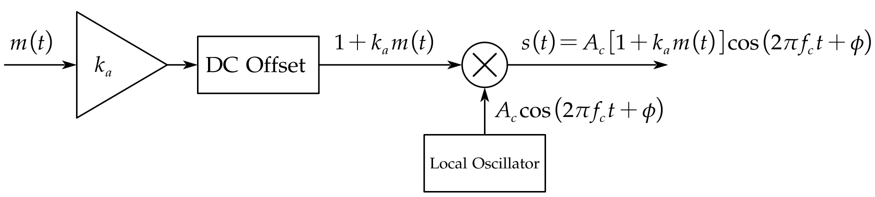


**Figure 2.1** Block diagram illustrating the process of envelope modulation.

The transfer function of a Butterworth Low Pass Filter is expressed as

$$|H(\omega)|^2 = \frac{1}{1 + \left(\frac{\omega}{\omega_c}\right)^{2n}} \tag{2.3}$$

where $n$ is the order of the filter and $\omega_c$ is the cut-off frequency. The higher the order $n$, the closer the Butterworth Filter behaves like an ideal filter. Similarly, the transfer function of a Butterworth Band Pass Filter is???????

According to The SciPy Community [6]–[9], the filters can be realized in the following way: ?????????????

```
# Low Pass Filter
# Get coefficients for butter filter
b, a = signal.butter(4, 100, 'low', analog=True)
w, h = signal.freqs(b, a)
sos = signal.butter(10, 15, 'hp', fs=1000, output='sos')
filtered = signal.sosfilt(sos, sig)

# Band Pass Filter
# Get coefficients for butter filter
b, a = signal.butter(4, [50, 100], 'band', analog=True)
w, h = signal.freqs(b, a)
sos = signal.butter(10, 15, 'hp', fs=1000, output='sos')
filtered = signal.sosfilt(sos, sig)
```

## 2.6 Additive White Gaussian Noise

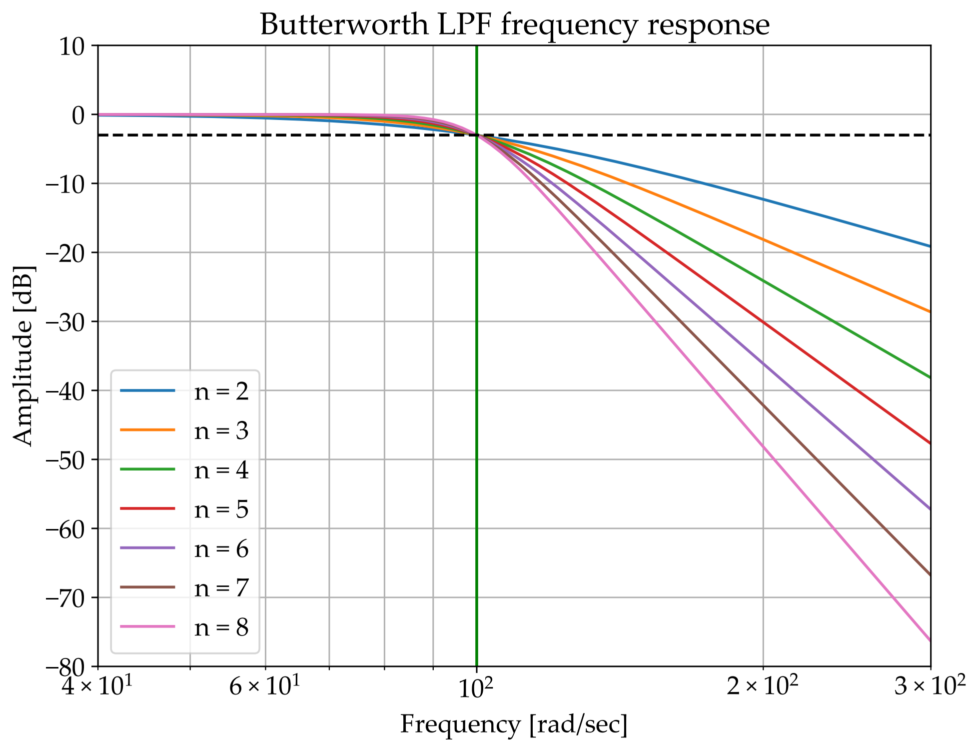


**Figure 2.2** The frequency response of a Butterworth LPF with cut-off frequency $\omega_c = 100\,\text{rad/s}$. The black dash line shows the $-3\,\text{dB}$ amplitude.

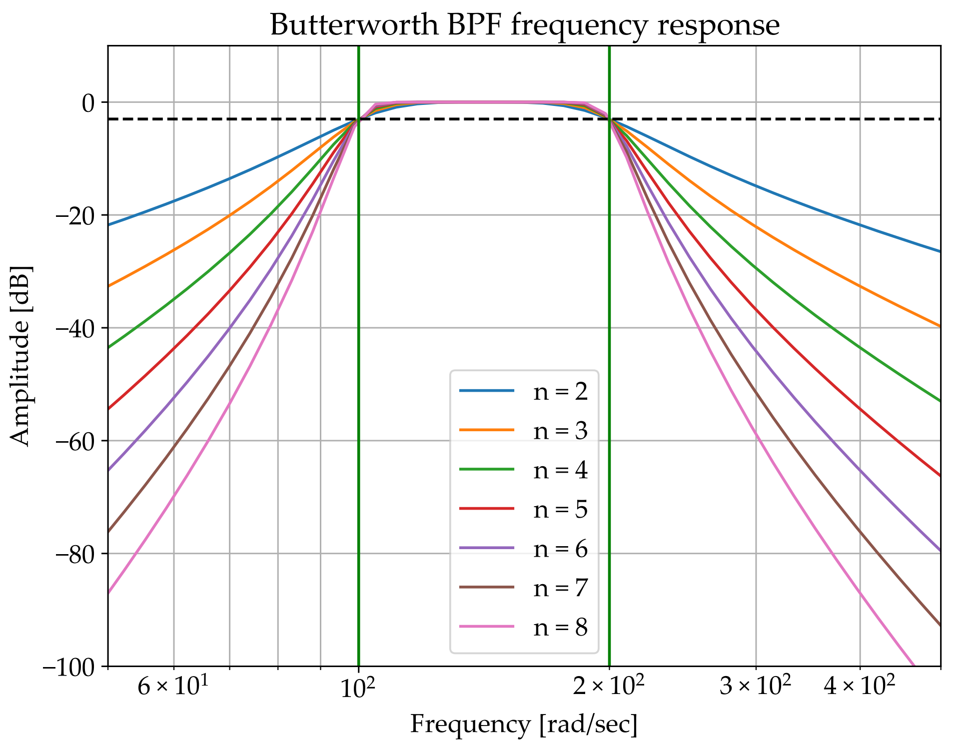


**Figure 2.3** The frequency response of a Butterworth BPF with cut-off frequency $\omega_c = 100\,\text{rad/s}$. The black dash line shows the $-3\,\text{dB}$ amplitude.

# 3 Results and Discussion

# 4 Conclusion

# Statement of Contribution

# Appendix A Python Scripts of This Project

# References


[1] S. Haykin and M. Moher, *Introduction to Analog & Digital Communications*, 2nd ed. NJ: Wiley, 2007, ISBN: 978-0-471-43222-7.

[2] T. Ulrich. "Envelope calculation from the hilbert transform," ResearchGate. (Mar. 17, 2006), [Online]. Available: https://www.researchgate.net/publication/257547765_Envelope_Calculation_from_the_Hilbert_Transform (visited on 12/30/2023).

[3] W. Storr. "Butterworth Filter Design and Low Pass Butterworth Filters," Basic Electronics Tutorials. (Aug. 14, 2013), [Online]. Available: https://www.electronics-tutorials.ws/filter/filter_8.html (visited on 12/31/2023).

[4] V. Khetarpal. "在 Python 中实现低通滤波器 [Implementing low-pass filters in Python]," DelftStack. (Dec. 21, 2022), [Online]. Available: https://www.delftstack.com/zh/howto/python/low-pass-filter-python/ (visited on 12/29/2023).

[5] E. Kudeki and D. C. Munson, *Analog Signals and Systems* (Illinois ECE Series). Upper Saddle River, N.J.: Pearson Prentice Hall, 2009, 512 pp., ISBN: 978-0-13-143506-3.

[6] The SciPy Community. "Scipy.signal.butter," SciPy v1.11.4 Manual. (n.d.), [Online]. Available: https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.butter.html#scipy.signal.butter (visited on 12/27/2023).

[7] The SciPy Community. "Scipy.signal.butter," SciPy v1.11.4 Manual. (n.d.), [Online]. Available: https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.butter.html (visited on 12/27/2023).

[8] The SciPy Community. "Scipy.signal.filtfilt," SciPy v1.11.4 Manual. (n.d.), [Online]. Available: https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.filtfilt.html#scipy.signal.filtfilt (visited on 12/27/2023).

[9] The SciPy Community. "Scipy.signal.lfilter," SciPy v1.11.4 Manual. (n.d.), [Online]. Available: https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.lfilter.html#scipy.signal.lfilter (visited on 12/27/2023).