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

This chapter presents the methodology employed in the project using Python 3. The simulations were executed within the Jupyter Notebook environment, leveraging essential packages for numerical computation, signal analysis, and plotting, namely `numpy`, `scipy`, and `matplotlib`.

2.1 Choice of Message Signal

Two distinct message signals were selected for thorough investigation 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 recording of a male voice reading "ECE 459 is so interesting."

The first signal, a multi-tone sinusoidal waveform, was chosen for its simplicity as a periodic function and its unique attributes as a linear combination of two sinusoidal functions. This particular choice facilitates the observation of distortion induced by noise, given the relatively straightforward frequency spectra of sinusoidal functions.

The utilization of a TTS-generated male voice recording, on the other hand, introduces a more realistic scenario, allowing the team to experiment with the designed communication system in a practical context. Such recording has no background noise between words, which enables the team to better observe the distortion by the simulated noise process.

2.2 Additive White Gaussian Noise

Additive White Gaussian Noise (AWGN) refers to a Gaussian process that can be directly added to a signal to approximate a noise-distorted signal in a real-life communication channel.

According to [1, Sec. 8.10], a noise process is called white noise if it has zero mean and its power spectral density satisfies

$$S_W(f) = \frac{N_0}{2}. \tag{2.1}$$

The power spectrum of a white noise process is shown in Figure 2.1.

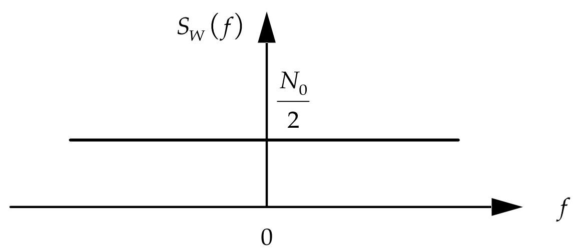

Figure 2.1 The power spectrum of a white noise process.

2.3 AM Simulation

2.3.1 Envelope Modulation

Consider a message signal $m(t)$ and carrier wave $A_c \cos(2\pi f_c t + \phi)$ where ϕ is the phase delay of the local oscillator. In this project, we pick $\phi = 0$ for simplicity. The modulation of a message signal, denoted as $m(t)$, can be achieved through envelope modulation, represented by the equation:

$$s(t) = A_c[1 + k_a m(t)] \cos(2\pi f_c t + \phi) \tag{2.2}$$

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

$$|k_a m(t)| < 1, \quad \text{for all } t. \tag{2.3}$$

The product of modulation sensitivity k_a and amplitude of message signal A_m

$$\mu = k_a A_m \tag{2.4}$$

is called the modulation factor, or as stated by [5], the *modulation index*. It can be alternatively expressed as

$$\mu = \frac{A_m}{A_c}. \tag{2.5}$$

In this project, the AM modulation index is specified as $\mu = 0.3$.

The implementation of amplitude modulation (AM) can be illustrated through the block diagram depicted in Figure 2.2. The message signal is amplified by a gain k_a and is added a DC offset. The carrier wave is then multiplied with the scaled and shifted message signal, which produces the envelope-modulated wave that is to be transmitted through the channel.

2.3.2 Envelope Detection

Diverging from coherent detection, envelope detection dispenses with the necessity of multiplying the received signal by the carrier wave. Instead, the received signal undergoes filtration by a Band Pass Filter (BPF) to eliminate noise beyond the desired bandwidth, and subsequently, an envelope detector

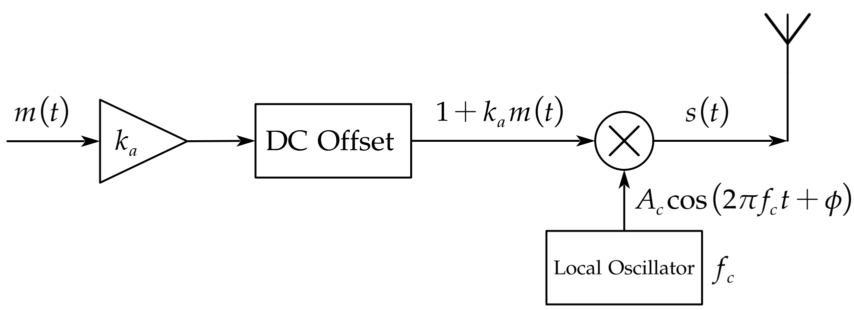

Figure 2.2 Block diagram illustrating the process of envelope modulation.

facilitates the recovery of the message signal. This procedural sequence is depicted in Figure 2.3.

In physical circuitry, the envelope detector typically comprises a diode and an LPF [6]. However, in Python, this functionality is realized through a Hilbert transform process [2], [7].

Consider a single-tone sinusoidal signal with frequency f_m modulated by a carrier wave with frequency f_c. The modulated wave, denoted as $g(t)$, is expressed as

$$g(t) = \sin(2\pi f_c t)\sin(2\pi f_m t) \tag{2.6}$$

where $f_c \gg f_m$. By trigonometric identity, $g(t)$ can be alternatively expressed as

$$g(t) = -\frac{1}{2}\{\cos[2\pi(f_c + f_m)t] - \cos[2\pi(f_c - f_m)t]\}. \tag{2.7}$$

The Hilbert transform introduces a $-\frac{\pi}{2}$ to the phase of the signal, which yields

$$\begin{aligned}\tilde{g}(t) &= -\frac{1}{2}\left\{\cos\left[2\pi(f_c + f_m)t - \frac{\pi}{2}\right] - \cos\left[2\pi(f_c - f_m)t - \frac{\pi}{2}\right]\right\} \\ &= -\frac{1}{2}\left\{\sin\left(2\pi f_c t - \frac{\pi}{2}\right)\sin(2\pi f_m t)\right\} \\ &= -\frac{1}{2}\cos(2\pi f_c t)\sin(2\pi f_m t)\end{aligned} \tag{2.8}$$

The analytic signal of $g(t)$, denoted as $\aleph\{g(t)\}$, can be expressed as a combination of $g(t)$ and its Hilbert transform $\tilde{g}(t)$, which is

$$\aleph\{g(t)\} = g(t) + j\tilde{g}(t) = \sin(2\pi f_c t)\sin(2\pi f_m t) - j\cos(2\pi f_c t)\sin(2\pi f_m t). \tag{2.9}$$

The magnitude of the analytic signal can be calculated by

$$|\aleph\{g(t)\}| = \sqrt{\aleph\{g(t)\}\aleph^*\{g(t)\}} = |\sin(2\pi f_m t)| \tag{2.10}$$

which precisely equals to the original $g(t)$.

Thus, the envelope of an envelope-modulated signal can be derived by calculating the magnitude of the Hilbert transform of the signal,

$$\text{envelop of } s(t) = |\tilde{s}(t)| \tag{2.11}$$

where $\tilde{s}(t)$ is the Hilbert transform of $s(t)$.

This methodology is applied by the team in the project.

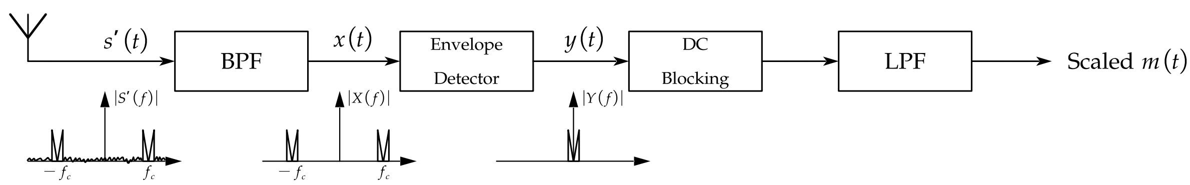

Figure 2.3 Block diagram illustrating the process of envelope demodulation.

2.3.3 SNR Calculation and Measurement

Pre-Detection SNR

As is stated in Haykin and Moher [1, Eq. (9.26)], the theoretical pre-detection SNR in an envelope modulation receiver can be calculated by

$$\mathrm{SNR}^{\mathrm{AM}}_{\text{pre, theoretical}} = \frac{A_c^2(1+k_a^2)P}{2N_0B_T} \tag{2.12}$$

where A_c is the carrier amplitude, k_a is the modulation sensitivity, P is the power of message signal, N_0 is twice the power spectral density of white noise and B_T is the noise bandwidth of the BPF.

Following the notation in Figure 2.3, the actual pre-detection SNR can be measured as

$$\mathrm{SNR}^{\mathrm{AM}}_{\text{pre, measured}} = \frac{\mathbb{E}[x(t)]}{\text{noise power}}. \tag{2.13}$$

Post-Detection SNR

The textbook [1, Eq. (9.23)] states that the theoretical post-detection SNR in an envelope modulation receiver is expressed as

$$\mathrm{SNR}^{\mathrm{AM}}_{\text{post, theoretical}} = \frac{A_c^2k_a^2P}{2N_0W} \tag{2.14}$$

which is a valid approximation only for high SNR and $0 < k_a < 100\%$.

The actual post-detection SNR can be expressed as, if following the notation in Figure 2.3,

$$\mathrm{SNR}^{\mathrm{AM}}_{\text{post, measured}} = \frac{\mathbb{E}[y(t)]}{\text{noise power}}. \tag{2.15}$$

Relation between Pre- and Post-Detection SNRs

2.4 FM Simulation

2.4.1 Narrow-Band FM Modulation

The textbook [1, Sec. 4.1] states that the message signal $m(t)$ will be phase-modulated with a carrier signal $A_c\cos(2\pi f_ct)$, which obtains the frequency modulated wave

$$s(t) = A_c\cos\left[2\pi f_ct + 2\pi k_f\int_0^t m(\tau)\mathrm{d}\tau\right] \tag{2.16}$$

where the instantaneous frequency is

$$f_i(t) = f_c + k_fm(t). \tag{2.17}$$

Here k_f denotes the modulation sensitivity which determines the frequency deviation by

$$\Delta f = k_fA_m. \tag{2.18}$$

By Carson's rule, [1, Sec. 4.6], [8], the transmission bandwidth of an FM wave for a single-tone modulating wave is estimated as

$$B_T = 2\Delta f + 2f_{m,\max} \tag{2.19}$$

where $f_{m,\max}$ is the highest modulating frequency. The ratio of Δf and f_m is defined as modulation index,

$$\beta = \frac{\Delta f}{f_m}. \tag{2.20}$$

In this project, the modulation index is specified as $\beta = 3$.

Such modulation can be done using a direct method [1, Sec. 4.7], where the frequency modulator contains an oscillator directly controllable by the message signal. Figure 2.4 illustrates this process: the message signal is taken integral and sent to the controllable oscillator to produce a phase-variant signal, which is the modulated wave.

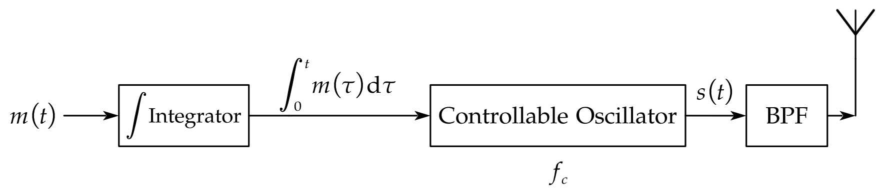

Figure 2.4 The direct method of narrow-band FM modulation.

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 in Python

Both Amplitude Modulation (AM) and Frequency Modulation (FM) communication systems necessitate the incorporation of filters. In practical scenarios, however, the realization of ideal filters poses inherent challenges. Nevertheless, the Butterworth Filter exhibits characteristics closely approximating those of an ideal filter. Figures 2.5 and 2.6 depict the frequency responses of Butterworth LPF and BPF with varying orders, illustrating that an increased order (n) results in a more pronounced roll-off slope. As expounded by [3], [9], a Butterworth Low Pass Filter manifests a maximally flat frequency response within its passband, swiftly attenuating beyond the cut-off frequency. This advantageous attribute empowers the design of filters with minimal distortion, albeit at the expense of an indeterminate phase delay induced by the inherent properties of Infinite Impulse Response (IIR) filters.

In the digital realm, Python facilitates the implementation of filters as digital IIR filters, with each filter in the z-domain expressed as the quotient of two polynomials:

$$H_{\text{filter}}(z) = \frac{\sum_{p=0}^{n} a_p z^{-p}}{\sum_{q=0}^{n} b_q z^{-q}} \tag{2.21}$$

The response of the filter is uniquely determined by coefficients a_i and b_j $(0 \le i, j \le n)$, where these coefficients can be computed using the `scipy.signal.butter()` function, as documented by [10]–[12]. A practical instantiation of an IIR filter is exemplified below:

```
# Generate an 8-order 40 Hz to 80 Hz IIR BPF, sample rate = fs
b, a = butter(N=8, [40, 80], btype='band', fs=fs)
# Apply the filter to the message signal
filtered_signal = filtfilt(b, a, message)
```

Figures 2.5 and 2.6 showcase the frequency response of a Butterworth LPF and BPF, respectively, each featuring a cut-off frequency of $\omega_c = 100\,\mathrm{rad/s}$. The black dashed line denotes the $-3\,\mathrm{dB}$ amplitude.

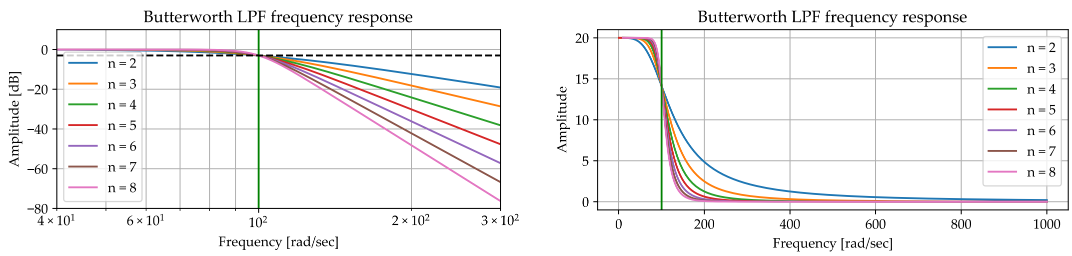

Figure 2.5 The frequency response of a Butterworth LPF with cut-off frequency $\omega_c = 100\,\mathrm{rad/s}$. The black dash line shows the $-3\,\mathrm{dB}$ amplitude.

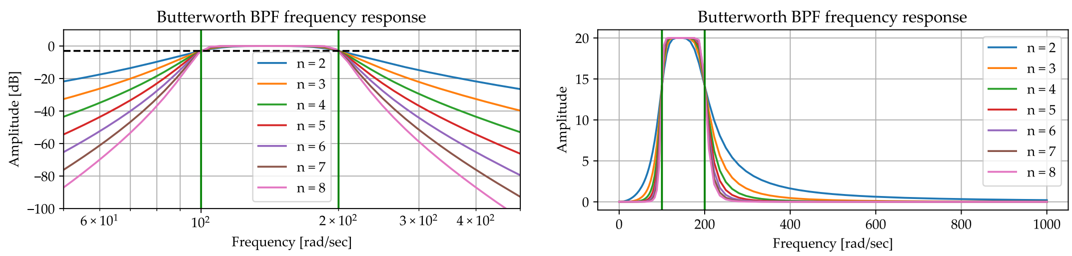

Figure 2.6 The frequency response of a Butterworth BPF with pass-band from 100 Hz to 200 Hz. The black dash line shows the $-3\,\mathrm{dB}$ amplitude.

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] Sasmita. "Modulation Index or Modulation Factor of AM Wave," Electronics Post. (May 19, 2020), [Online]. Available: https://electronicspost.com/modulation-index-or-modulation-factor-of-am-wave/ (visited on 01/04/2024).

[6] "Analog Communication - AM Demodulators," tutorialspoint. (n.d.), [Online]. Available: https://www.tutorialspoint.com/analog_communication/analog_communication_am_demodulators.htm (visited on 01/02/2024).

[7] 西笑生. "Python 提取信号的包络 [Get envelope of a signal in Python]," CSDN Blog. (Mar. 10, 2023), [Online]. Available: https://blog.csdn.net/flyfish1986/article/details/129444260 (visited on 12/29/2023).

[8] "Carson's Rule," DAEnotes. (Nov. 12, 2017), [Online]. Available: https://www.daenotes.com/electronics/communication-system/carsons-rule (visited on 01/04/2024).

[9] 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.

[10] 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).

[11] 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).

[12] 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).