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	3200112322	
Peidong YANG	3200115555	
Zhuohao XU	3200115233	
Rongjian CHEN	3200110745	
Tiantian ZHONG	3200110643	

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]. Though Ulrich [2] introduces an alternative method of envelope detection utilizing Hilbert Transformation, this non-causal operation is not implementable in analog circuits. A more common implementation in real circuit design is to apply a diode detector (whether full-bridge or half-bridge rectifier) cascaded by an LPF.

This project applies the Direct Method of FM signal generation as illustrated in [1, Fig. 4.7], which involves components such as an integrator, a phase modulator, and a local oscillator. [1, Fig. 9.13] also elucidated the demodulation process that could be applied to the project.

The measurement of pre- and post-detection SNRs are mentioned in [1, Sec. 9.5 & 9.7], where the theoretical SNRs are also give. This allows the team to evaluate both the performance of the channels and the quality of the simulation.

Ideal filters are not physically implementable as they are non-causal [3, p. 428]. However, the Butterworth filter offers a practical solution for simulating real-world filters, as discussed in the works of Storr [4] and Khetarpal et al. [5]. Implementation of the Butterworth filter can be achieved using the `scipy.signal` package.

2 Methodology

This chapter presents the methodology and the relevant theories employed in the project. 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 The Message Signals

2.1.1 The Choices of Message Signals

Two distinct message signals were selected for thorough investigation in this project:

1. The multi-tone sinusoidal signal $\sum_{i=1}^{n} m(t) = A_i \cos(2\pi f_i t + \phi_i)$.
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. The signal is plotted as Figure 2.1.

2.1.2 Bandwidth of Signals

It is of significance to measure the bandwidth of message signals. The bandwidth can be determined by analytical calculations or by visually analyzing the spectrums.

As is stated in [3, Sec. 7.2.3] and illustrated in Figure 2.2, the bandwidth of a band-limited signal, B_T, is the minimum frequency component that the signal has almost zero magnitude at any $f > B_W$, which means

$$S(f) = \begin{cases} \not\approx 0, & \text{for } |f| = B_T \\ \approx 0, & \text{for any } |f| > B_T \end{cases}. \tag{2.1}$$

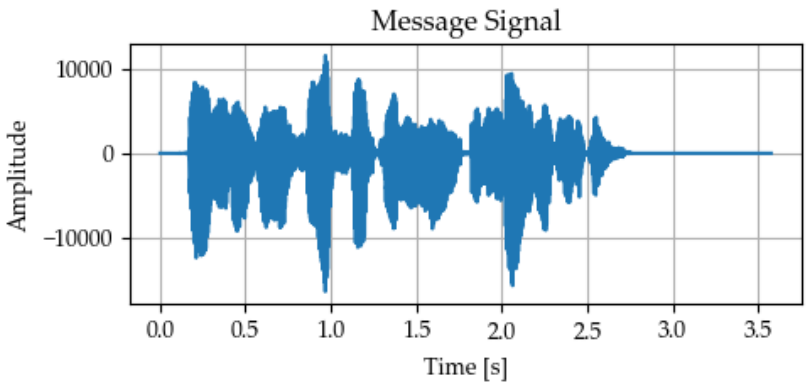

Figure 2.1 The waveform of the TTS-generated recording.

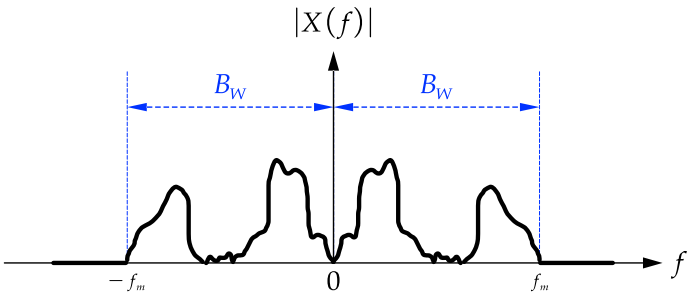

Figure 2.2 The bandwidth of the signal is B_W.

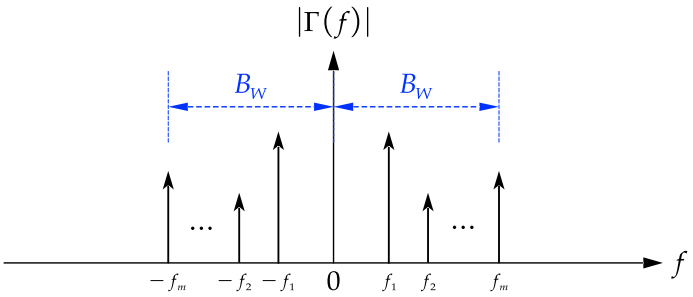

Figure 2.3 The spectrum of a multi-tone signal $\gamma(t)$. B_T is the bandwidth of the signal.

Consider the multi-tone sinusoidal signal $\gamma(t) = A_i \cos\left(2\pi f_i t + \phi_i\right)$. Its Fourier Transform is

$$\Gamma(f) = \sum_{i=1}^{n} \frac{jA_i}{2} \left\{\delta\left[2\pi\left(f + f_i\right)\right] - \delta\left[2\pi\left(f - f_i\right)\right]\right\} \tag{2.2}$$

Hence the spectrum of $\gamma(t)$ should be similar to Figure 2.3. It is apparent that by Equation (2.1), the bandwidth is $B_T = f_m$.

The bandwidth of the TTS-generated recording can be determined by its spectrum. Note that the recording is generated using the male's voice, which typically ranges from 80 Hz to 200 Hz. The spectrum of the recording is plotted as Figure 2.4, where it is observed that the majority of power is concentrated within 1 kHz. This is an important guidance for determining the cut-off frequency of the LPFs we used in the project.

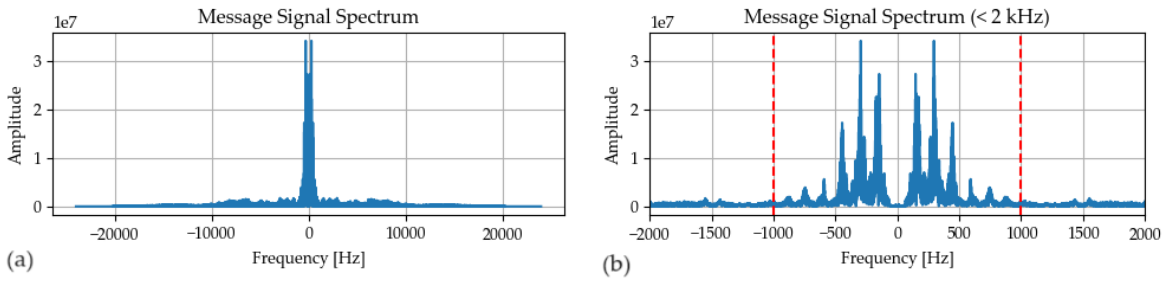

Figure 2.4 The spectrum of the audio recording. (a) The total spectrum. (b) The spectrum within 2 kHz.

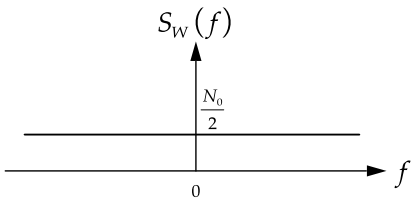

Figure 2.5 The power spectrum of a white noise process.

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.

A Gaussian process denoted as $\mathrm{N}\left[\mu, \sigma^2\right]$ has the probability density function as

$$f_{Gaussian}(u) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{(u-v)^2}{2\sigma^2}\right]. \tag{2.3}$$

It has mean μ and standard deviation σ^2.

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

$$S_W(f) = \frac{N_0}{2}. \tag{2.4}$$

The power spectrum of a white noise process is shown in Figure 2.5.

In this project, the AWGN process will be simulated using `numpy.random.normal()` function. Figure 2.6 shows a Python-generated Gaussian noise process and its frequency spectrum.

2.3 AM Simulation

2.3.1 Envelope Modulation

Consider a message signal $m(t)$ and carrier wave $A_c \cos\left(2\pi f_c t + \phi\right)$ 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:

$$s(t) = A_c\left[1 + k_a m(t)\right] \cos\left(2\pi f_c t + \phi\right) \tag{2.5}$$

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]:

$$|k_a m(t)| < 1, \quad \text{for all } t. \tag{2.6}$$

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

$$\mu = k_a A_m \tag{2.7}$$

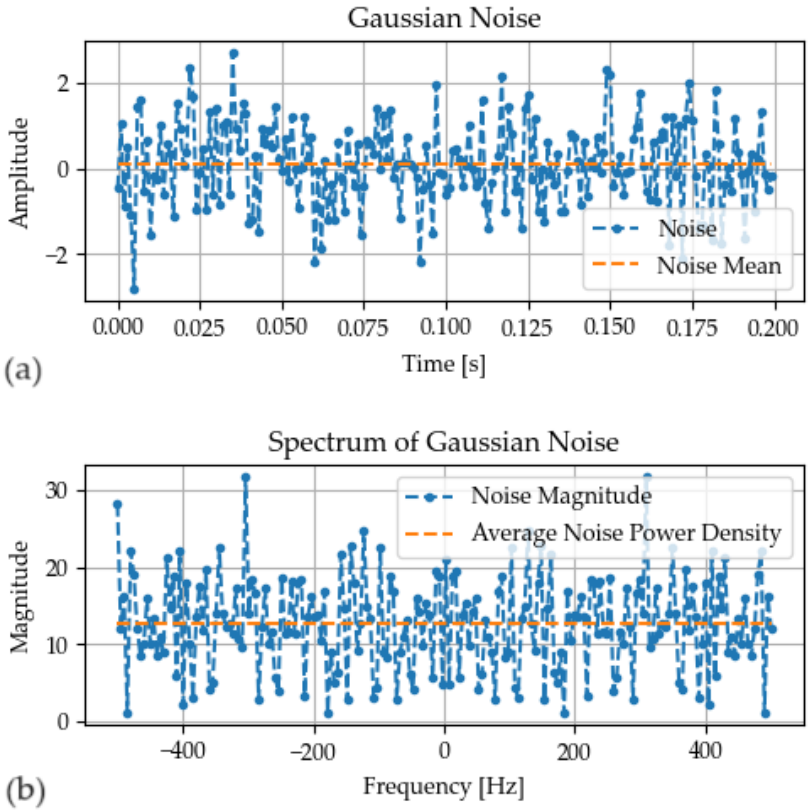

Figure 2.6 The Python-generated Gaussian process N $\left[\mu = 0, \sigma^2 = 1\right]$ and its spectrum.

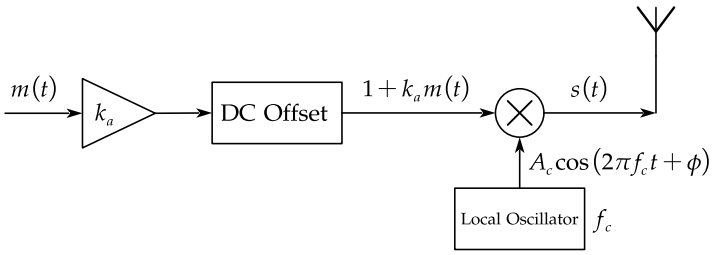

Figure 2.7 Block diagram illustrating the process of envelope modulation.

is called the modulation factor, or as stated by [6], the modulation index, which can be alternatively expressed as

$$\mu = \frac{A_m}{A_c}. \tag{2.8}$$

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

The implementation of amplitude modulation can be illustrated through the block diagram depicted in Figure 2.7. 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

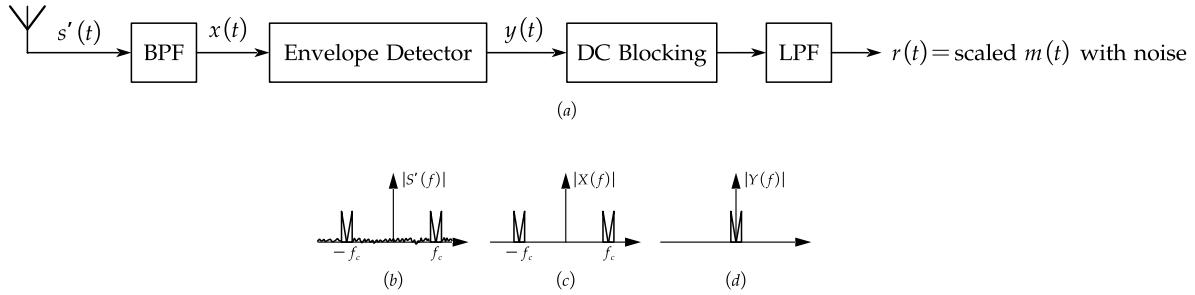

Figure 2.8 The envelope demodulation process. (a) The block diagram of the demodulation process. (b) The spectrum of the received noisy signal $s'(t)$. (c) The spectrum of the band-pass filtered signal $x(t)$. (d) The spectrum of envelope detected signal $y(t)$.

Filter (BPF) to eliminate noise beyond the desired bandwidth, and subsequently, an envelope detector facilitates the recovery of the message signal. This procedural sequence is depicted in Figure 2.8.

A common way to build the envelope detector in Python is to apply Hilbert transform [2], [7]. The envelope of an envelope-modulated signal $s\,(t)$ can be derived by calculating the magnitude of the Hilbert transform of the signal,

$$\text{envelop of } s\,(t) = |\tilde{s}\,(t)| \tag{2.9}$$

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

However, as is mentioned in [3, p. 428], the Hilbert Transform is not a causal process, so is not able to be realized using analog circuits. Instead, in physical circuitry, the envelope detector typically comprises a diode and an LPF [8].

Figure 2.9 illustrates a typical diode detector circuit with a full-wave rectifier. As is discussed in [1, pp. 111–112], on the positive segment of the signal, the diode is forward-biased and the capacitor is charged quickly. On the negative segment, the diode becomes reverse-biased and the capacitor discharges slowly through the resistor. When the input signal is greater than the output signal, the capacitor charges again; when the input signal is smaller, the capacitor discharges. In this way, the RC circuit can filter out the high-frequency carrier wave component and approximate the message signal. The time constant of the RC filter, $\tau = RC$, should satisfy

$$RC \ll \frac{1}{f_c}. \tag{2.10}$$

The charging and discharging process is visualized in Figure 2.10 which takes a square wave as an example message signal. The output from the envelope detector is a series of saw-tooth wave that approximates the input square-wave message signal.

2.3.3 SNR Calculation and Measurement

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

$$\text{SNR}^{\text{AM}}_{\text{pre, theoretical}} = \frac{A_c^2(1 + k_a^2)P}{2N_0B_T} \tag{2.11}$$

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.

The actual pre-detection SNR can be obtained by measuring the power of the obtained signal and

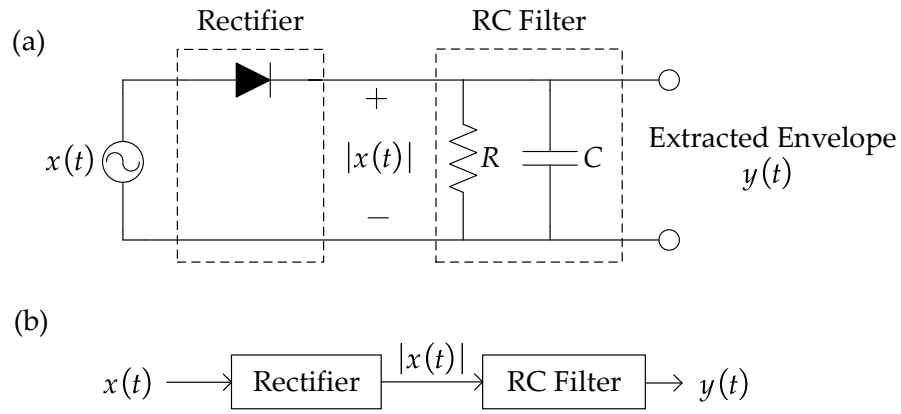

Figure 2.9 The diode detector. (a) The circuit of the typical diode detector. (b) The block diagram of the diode detector.

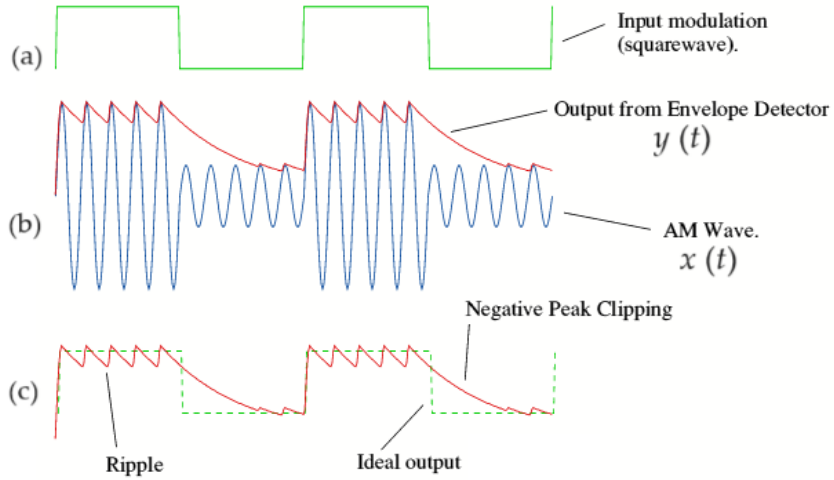

Figure 2.10 Ripple and peak clipping effects [9, Fig. 9.3]. (a)

that of noise. Since the filtering process will filter both the useful signal and the noise, the SNR calculation cannot simply apply the specified noise power used in AWGN generation. Instead, there are two ways of measuring the noise in this simulation:

1. Calculating as the power of difference between the resulting signal under an ideal channel and that under a noisy channel.
2. Calculating the power of noise that passes all the filters.

Following the notation in Figure 2.8, the actual pre-detection SNR can be measured as

$$\mathrm{SNR}^{\mathrm{AM}}_{\text{pre, measured}} = \frac{\mathbb{E}[x^2(t)]}{\text{filter noise power}}. \tag{2.12}$$

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

$$\mathrm{SNR}^{\mathrm{AM}}_{\text{post, theoretical}} = \frac{A_c^2 k_a^2 P}{2N_0 W} \tag{2.13}$$

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

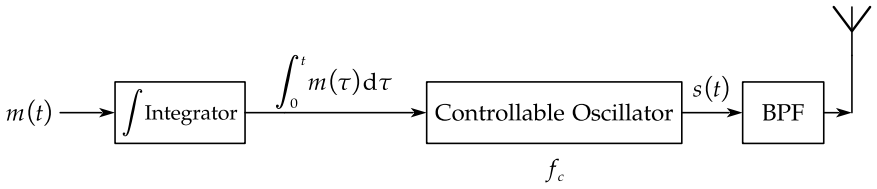

Figure 2.11 The direct method of narrow-band FM modulation.

The actual post-detection SNR can be expressed as, if following the notation in Figure 2.8,

$$\mathrm{SNR}^{\mathrm{AM}}_{\text{post, measured}} = \frac{\mathbb{E}[r^2(t)]}{\mathbb{E}\left[[r(t) - m(t)]^2\right]}. \tag{2.14}$$

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 $A_c \cos(2\pi f_c t)$, which obtains the frequency modulated wave

$$s(t) = A_c \cos\left[2\pi f_c t + 2\pi k_f \int_0^t m(\tau)\mathrm{d}\tau\right] \tag{2.15}$$

where the instantaneous frequency is

$$f_i(t) = f_c + k_f m(t). \tag{2.16}$$

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

$$\Delta f = k_f A_m. \tag{2.17}$$

By Carson's rule [1, Sec. 4.6], [10], the transmission bandwidth of an FM wave for a frequency-modulated signal is estimated as

$$B_T = 2\Delta f + 2f_{m,\max} \tag{2.18}$$

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

$$\beta = \frac{\Delta f}{f_m}. \tag{2.19}$$

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

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 2.11 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.

2.4.2 Narrow-Band FM Demodulation

The key to FM demodulation is to extract the phase of the received signal. Considering an ideal channel, the received signal should equal the modulated signal $s(t)$ in Eq. (2.15). Taking the derivative

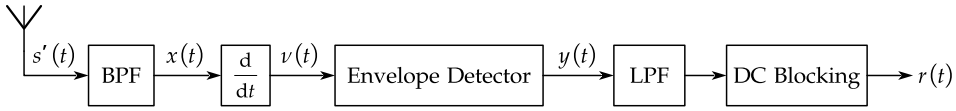

Figure 2.12 The demodulation process of frequency-modulated signals.

of $s(t)$ yields

$$v(t) = \frac{\mathrm{d}}{\mathrm{d}t} s(t) = -A_c \left[2\pi f_c + 2\pi k_f m(t)\right] \sin\left[2\pi f_c t + 2\pi k_f \int_0^t m(\tau)\,\mathrm{d}\tau\right]. \tag{2.20}$$

An envelope detector could remove the sinusoidal term and get

$$y(t) = -A_c \left[2\pi f_c + 2\pi k_f m(t)\right] \tag{2.21}$$

which only contains the scaled version of $m(t)$ and some DC offset. Hence by applying a DC blocking circuit and a proper gain to $y(t)$, the message signal can be concluded.

In practice where the channel is noisy, a BPF will be applied before differentiating and an LPF will be applied before concluding the output, both of which suppress noise introduced by the channel. The complete demodulation process is shown in Figure 2.12.

2.4.3 SNR Calculation and Measurement

As discussed in [1, Sec. 9.7], the theoretical pre- and post-detection SNR is expressed as

$$\mathrm{SNR}^{\mathrm{FM}}_{\text{pre, theoretical}} = \frac{A_c^2}{2N_0 B_T} \tag{2.22}$$

$$\mathrm{SNR}^{\mathrm{FM}}_{\text{post, theoretical}} = \frac{3A_c^2 k_f^2 P}{2N_0 W} \tag{2.23}$$

where P is the power of the message signal, B_T is the noise bandwidth of the BPF, and W is the message bandwidth.

The actual SNRs can be calculated by measuring the power of the signal and noise, which can be concluded as

$$\mathrm{SNR}^{\mathrm{FM}}_{\text{pre, measured}} = \frac{\mathbb{E}\left[x^2(t)\right]}{\text{filter noise power}} \tag{2.24}$$

$$\mathrm{SNR}^{\mathrm{FM}}_{\text{post, measured}} = \frac{\mathbb{E}[r^2(t)]}{\mathbb{E}\left[\left[r(t) - m(t)\right]^2\right]}. \tag{2.25}$$

2.5 Filter Design

2.5.1 Ideal Filters

The filters remove the unnecessary frequency components of its input signals. Figure 2.13 shows the impulse response of an ideal LPF and BPF.

The expressions for ideal LPF and BPF in the frequency domain are

$$H_{LPF}(f) = \text{rect}\left(\frac{f}{2f_{cut}}\right) \tag{2.26}$$

$$H_{BPF}(f) = \text{rect}\left(\frac{f - f_{cut}}{2f_{cut}}\right) + \text{rect}\left(\frac{f + f_{cut}}{2f_{cut}}\right) \tag{2.27}$$

By inverse Fourier transformation, their time-domain impulse response can be obtained as

$$h_{LPF}(t) = 2f_{cut}\text{sinc}(2f_{cut}t) \tag{2.28}$$

$$h_{BPF}(t) = 2f_{cut}\text{sinc}(2f_{cut}t)\,\text{e}^{j2\pi f_{cut}t} - 2f_{cut}\text{sinc}(-2f_{cut}t)\,\text{e}^{-j2\pi f_{cut}t} \tag{2.29}$$

which contains non-periodic and infinitely expanding sinc terms. This indicates the impulse response to be associated with the future input, which violates the causality. Thus the ideal LPF and BPF are not causal and are not physically implementable.

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. Figure 2.14 illustrates an analog circuit design of a first-order Butterworth LPF which is composed of an operational amplifier. According to [11], the general n-th order Butterworth LPF has a transfer function

$$H_n(j\omega) = \frac{1}{\sqrt{1 + \epsilon^2\left(\frac{\omega}{\omega_c}\right)^{2n}}} \tag{2.30}$$

where ϵ is the maximum pass-band gain and ω_c is the cut-off frequency. Figures 2.15 and 2.16 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], [4], 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:

$$H_{\text{digital}}(z) = \frac{\sum_{p=0}^{n} a_p z^{-p}}{\sum_{q=0}^{n} b_q z^{-q}}. \tag{2.31}$$

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 [12]–[14].

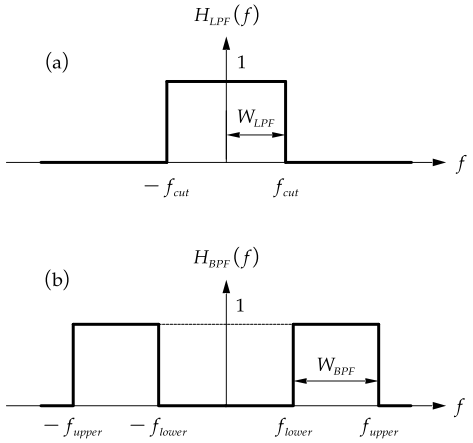

Figure 2.13 The impulse response of ideal LPF and BPF. (a) The impulse response of an ideal LPF. W_{LPF} shows the bandwidth of the LPF. (b) The impulse response of an ideal BPF. W_{BPF} shows the bandwidth of the BPF.

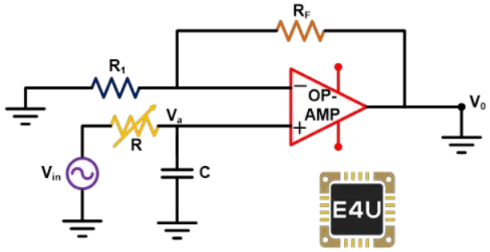

Figure 2.14 A first-order Butterworth LPF circuit [11].

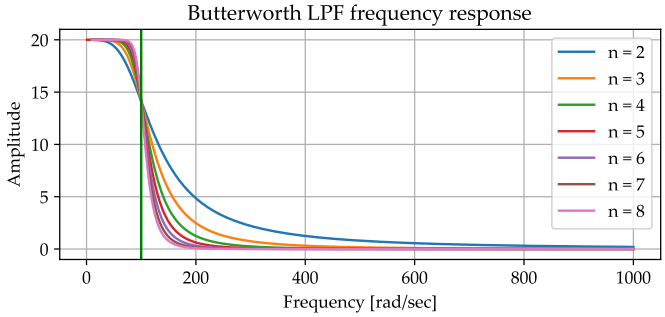

Figure 2.15 The frequency response of a Butterworth LPF with cut-off frequency $\omega_c = 100\,\text{rad/s}$.

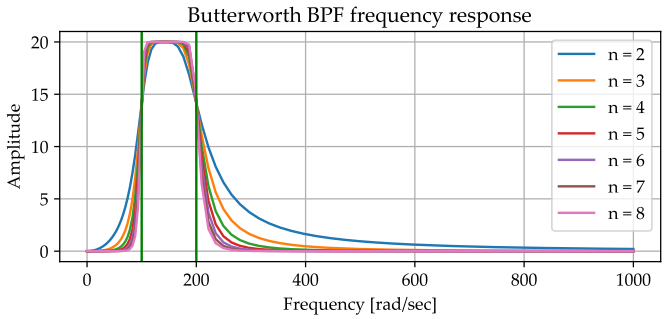

Figure 2.16 The frequency response of a Butterworth BPF with pass-band from 100 Hz to 200 Hz.

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

[4] 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).

[5] 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).

[6] 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).

[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] "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).

[9] J. Lesurf. "The Envelope Detector." (n.d.), [Online]. Available: https://www.winlab.rutgers.edu/~crose/322_html/envelope_detector.html (visited on 01/02/2024).

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

[11] "Butterworth Filter: What is it? (Design & Applications) | Electrical4U," https://www.electrical4u.com/. (Apr. 16, 2021), [Online]. Available: https://www.electrical4u.com/butterworth-filter/ (visited on 01/08/2024).

[12] 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).

[13] 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).

[14] 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).