Frequency Modulation and Demodulation

1st Aidan Sharpe
Electrical and Computer Engineering
Rowan University
Glassboro, United States
sharpe23@students.rowan.edu

2nd Elise Heim
Electrical and Computer Engineering
Rowan University
Glassboro, United States
heimel27@students.rowan.edu

3rd John Leahy
Electrical and Computer Engineering
Rowan University
Glassboro, United States
leahyj92@students.rowan.edu

I. Introduction

Frequency Modulation (FM) is a widely used technique in electrical communication systems, valued for its resilience to noise and audio signal transmission. This exercise aims to explore the principles of FM communication systems, by modeling and analyzing system performance. By simulating a single-tone FM system, both in ideal conditions and under the influence of Gaussian noise, this project provides a hands-on approach to understanding the behavior of FM modulation. This analysis is crucial for applications ranging from radio broadcasting to modern wireless communication technologies, where maintaining signal integrity in noisy environments is crucial.

II. Frequency Modulation

Given any message signal $m(t)$, we can create its corresponding frequency modulated signal using

$$s(t) = A_c \cos(2\pi f_c t + \beta_f m(t)). \tag{1}$$

where A_c is the carrier amplitude, f_c is the carrier frequency, and β_f is the modulation index. Essentially, $m(t)$ alters the phase of the carrier signal such that the difference between the instantaneous frequency and the carrier frequency is proportional to the amplitude of $m(t)$.

An example of an FM signal is seen in figure 1. While it is difficult to make out the individual waves and their frequencies, there are clearly areas where the waves seem to "bunch up" or "spread out". The "bunched up" areas correspond to a higher frequency, and therefore a higher message amplitude, and the more "spread out" areas correspond to a lower frequency and therefore a lower message amplitude.

III. The Modulation Index

The modulation index β_f determines how much the amplitude of the message affects the frequency deviation. Since we are using a sinusoidal signal to modulate $m(t)$, the frequency spectrum can be found by using a Bessel function. At the zeros of the Bessel function, the carrier frequency has a small amplitude, and therefore almost all transmission power is being used to transmit the message. This behavior is known as the "null carrier" effect, and it is desirable, as it improves the efficiency of the transmission. The smallest value for β_f that creates a null carrier effect is $\beta_f = 2.40483$.

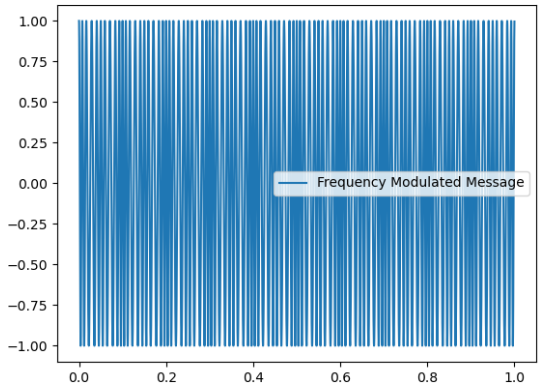

Fig. 1. Frequency modulated signal

We modeled our FM system with two different modulation indices $\beta_f = 2.40483$ and $\beta_f = 5$, and we compared the results, seen in figure 2. In the left plot, there is no frequency contribution at the carrier frequency $f_c = 500$[Hz]. In the right plot, however, there is a spike at the carrier frequency.

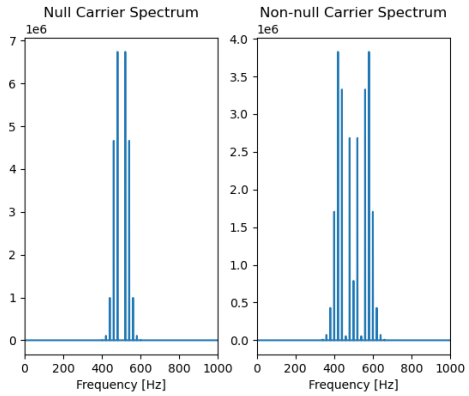

Fig. 2. Comparing null carrier and non-null carrier spectra

IV. Demodulation

Demodulation is the process of recovering the original message from a modulated signal. Put simply, it is the reverse

of modulation. For our setup we utilized a technique called quadrature demodulation. To use quadrature demodulation, the first step is to split the incoming signal into an in-phase (I) component and a quadrature (Q) component. The I component is obtained by multiplying the incoming signal with a local oscillator oscillating at the carrier frequency. The Q component is obtained with the same process, but the oscillator is given a 90° phase-shift prior to the multiplication. The total signal in I-Q form was represented as

$$IQ(t) = s(t)\cos(2\pi f_c t) + js(t)\sin(2\pi f_c t) \tag{2}$$

where f_C is the carrier frequency and $s(t)$ is the incoming FM signal.

Once the signal is decomposed into I-Q form, the message signal becomes much easier to recover. The first step is to take the complex conjugate of the I-Q signal, then calculate the angle above the real axis. Since the effect of $m(t)$ on $s(t)$ is a phase-shift, the angle is directly proportional to the amplitude of the message signal.

We implemented this process in Python, and we were able to demodulate both single- and multi-tone signals, as seen in figure 3. While the amplitude of the original and demodulated signal are not exactly the same, there is a constant of proportionality relating the two signals. Another feature of this demodulation technique is that the phase shift is very small.

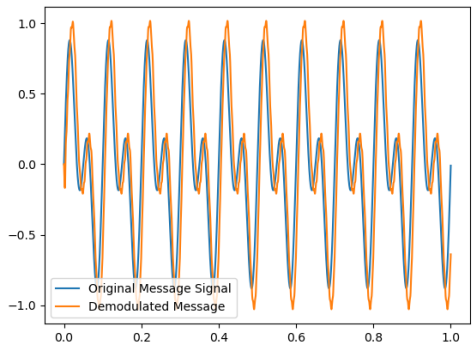

Fig. 3. Original and demodulated message signals

V. Quadrature Amplitude Modulation

QAM is a method of digital modulation widely used in modern electrical communications to transmit information. It relays two analog message signals by modulating the amplitudes of two carrier waves. This is accomplished using the amplitude-shift keying (ASK) digital modulation scheme. In ASK, a symbol is sent by transmitting a fixed-amplitude carrier wave at a fixed frequency for a specific duration of time. The two carrier waves are of the same frequency and are out of phase with each other by 90°, which is known as orthogonality (quadrature). The transmitted signal is created by adding the two carrier waves together. At the receiver, the two waves can be coherently demodulated because of their orthogonality. Another key property is that the modulations are low-frequency/low-bandwidth waveforms compared to the carrier frequency.

VI. Conclusion

We developed a simulation of a FM system including a modulator, channel, and demodulator. We also varied the modulation index to demonstrate the null carrier effect. Analyzing the incoming signal in both the spectral and time domains allowed us to gain a deeper understanding of the frequency modulation technique.

VII. Appendix

```
def lowpass(data, f_cutoff, f_s):
    nyq = 0.5*f_s
    normal_cutoff = f_cutoff/nyq
    b, a = sp.signal.butter(2, normal_cutoff, btype="low", analog=False)
    y = sp.signal.lfilter(b, a, data)
    return y

def quad_demod(s_fm, bandwidth, f_s, f_c, t):
    iq = real_to_iq(s_fm, f_c, bandwidth, f_s, t)
    iq = np.conj(iq)
    return 0.5*np.angle(iq)

def real_to_iq(s_fm, f_c, bandwidth, f_s, t):
    i = lowpass(s_fm * np.cos(2*np.pi*f_c*t), bandwidth, f_s)
    q = lowpass(s_fm * np.sin(2*np.pi*f_c*t), bandwidth, f_s)
    return i + 1j*q

def db(x):
    return 10*np.log10(x)
```