ECE300 – Project 1

Jonathan Lam

October 16, 2020

Simulating analog modulation schemes in MATLAB

Contents

List of Listings

List of Figures

List of Tables

1 Overview

Analog modulation schemes are at the core of all modern communication schemes. In this project, four common modulation schemes – conventional amplitude modulation, single side-band amplitude modulation, phase modulation, and frequency modulation – are implemented in MATLAB. In this report, we first review some of the modulation theory. We demonstrate that these signals faithfully recreate the original message after modulation and demodulation, and that the modulation is correct. Then, after adding additive white gaussian noise (AWGN) random processes, simulations are performed to analyze the effect of modulation scheme parameters and noise variance on the signal-to-noise ratio (SNR) on the different schemes. The relationship between SNR and noise power, and the relationships between SNR and the various parameters of the modulation schemes, generally agreed with the theory. The theoretical SNR calculations are compared with the experimentally-computed SNR values. SNR values comparable to theoretical values are obtained for all but FM. In addition to the requirements of this project, some additional checks were performed to verify the behavior of the modulated signals and also generally agreed with the theory.

1.1 Procedure

The assignment called for the following:

1. Read in an audio signal (and its sampling rate).
2. Write modulator/demodulator implementations for each modulation scheme. Make sure that the behavior seems correct. Verify Carson's rule.
3. Generate three zero-mean, different-variance AWGN noise processes. Add these to the modulated outputs of each system, demodulate them, and calculate the SNR of the demodulated signals experimentally. Compare these with the theoretical results.
4. Choose a noise process for each of conventional AM, FM, and PM which makes the demodulated output qualitatively noisy (but recognizable). Vary the modulation indices of each of these schemes by a factor of two in both directions, and calculate the SNR of the demodulated signals experimentally. Compare these with the theoretical results.

Reading in the audio file is described in 8. The implementations of the modulator and demodulator functions for each scheme are described in 2. Plots of the results appear in 3 and are discussed in 4.

1.2 Execution environments and resource usage

In order to emulate reasonable frequencies for the modulation schemes, I originally had the carrier and sampling frequencies set fairly high. The carrier frequency was set around 600kHz for amplitude modulation and around 1MHz for angle modulation, and the sampling rates were set to ten times those frequencies. Even though this AM carrier is on the lower end of the AM spectrum, and the FM carriers are far lower than real carriers, the lower end of which is at 88MHz [1], this was quite heavy on my computers. This caused many vectors with dozens of millions of double floating-point samples, which took dozens of seconds to run and took over 30GB of RAM when running on my Intel S5520UR server.

Another problem I encountered with these high frequencies is that there was often significant numerical error, especially with my FM signal. Interestingly, this happened very clearly when advancing the sampling rate to over 5MHz, or the carrier frequency to over 500kHz.

As a result of both the resource consumption issues and numerical errors associated with the multi-megahertz frequency ranges, I lowered the carrier and sampling frequencies for the angle modulation. While this becomes even less unrealistic for angle modulation, it did help alleviate both of these problems. A plot of the FM demodulated signal is Figure 16, and a plot of the FM demodulated signal with the lower carrier and sampling frequency is Figure 5.

2 Review of modulation schemes

Conventional AM, SSB AM, FM, and PM schemes are covered in this section. We also covered DSB AM and vestigial AM in class, but these are not covered here.

2.1 Conventional (CONV) AM modulation

2.1.1 Overview

Conventional AM (henceforth denoted CONV) is amplitude modulation with an offset, such that the modulation envelope is always positive and thus can be cheaply extracted with an envelope detector. Given a message signal $m(t)$, where $|m(t)| \leq 1$, modulation index $0 \leq a \leq 1$ (greater values of a are possible but cause overmodulation, or clipping), carrier frequency f_c, and carrier amplitude A_c, the modulation involves scaling and shifting m, and then mixing this with a carrier tone. The modulated signal $u(t)$ is:

$$u_{conv}(t) = A_c(1 + am(t)) \cos(2\pi f_c t) \tag{1}$$

Demodulation can be performed in two ways:

1. By mixing with the carrier signal (i.e., an I-Q demodulator with only an I component), which can be obtained by using a narrowband filter similar to in DSB-SC AM demodulation.
2. By rectifying and low-pass-filtering the signal (i.e., performing an envelope detection). Either full-wave or half-wave rectification can be used. This method is ubiquitous because it is cheap to implement in hardware.

Additionally, the demodulator has to:

- DC block (e.g., with a HPF) to remove the offset
- Divide the output by aA_c to get back to the original output amplitude (power)
- Scale the output up in order to compensate for energy lost during the LPF/HPF

2.1.2 Analysis

Let P_u be the total transmitted (bandpass) (modulated signal) power, P_c is the power in the transmitted signal sent in the carrier, P_y is the power in the transmitted signal for the message signal, and P_m is the power of the original normalized message signal.

$$P_u = P_c + P_m \tag{2}$$

$$P_c = \frac{A_c^2}{2} \tag{3}$$

$$P_y = \frac{a^2 A_c^2}{2} P_m \tag{4}$$

Since $P_m < 1$ (since $|m(t)| < 1$) and $|a| < 1$, at least half of the power is always sent in the carrier.

The SNR can be estimated by assuming an I-Q demodulation scheme. This doesn't account for rectification noise and is thus not perfect, but it is good enough. Let $r(t)$ be the noiseless received

message signal after mixing with the carrier tone (and ignoring the carrier power, which can be removed with a DC block). Let $y(t)$ denote the message part of the signal, and $n(t)$ denote the additive noise incurred during transmission.

$$r(t) = \frac{1}{2}A_c(1 + am(t))\cos(2\pi f_c t) + n_I(t)\cos(2\pi f_c t) + n_Q(t)\sin(2\pi f_c t) \tag{5}$$

$$\rightarrow \frac{1}{2}A_c(am(t)) + n_I(t) \qquad \text{(after DC block and mixing with cosine)}$$

$$P_y = \frac{a^2A^2}{4}P_m \tag{6}$$

$$P_r = \frac{2N_0W}{4} \tag{7}$$

$$\text{SNR} = \frac{a^2A_c^2}{2N_0W}P_m \tag{8}$$

The calculation for noise power was derived in the analog domain. See 10 for details about the digital domain calculation.

2.1.3 Implementation details

The modulation function is fairly straightforward. Notes on demodulation:

```
% modulates a signal using the conventional AM scheme
% params:
% f_c   = frequency of carrier (Hz)
% A_c   = carrier amplitude
% a     = modulation index
% sig_m = message signal
% f_s_m = message sampling frequency (Hz)
% f_s_c = carrier sampling frequency (Hz)
% returns:
% sig_c = conventional AM-modulated signal
function sig_c = conv_mod(f_c, A_c, a, sig_m, f_s_m, f_s_c)
    duration = length(sig_m) / f_s_m;
    t_m = linspace(0, duration, length(sig_m));
    t_c = linspace(0, duration, f_s_c * duration);

    % upsample sig_m
    sig_m_us = interp1(t_m, sig_m, t_c);

    % generate conventional AM-modulated signal
    sig_c = A_c * (1 + sig_m_us * a) .* cos(2 * pi * f_c * t_c);
end
```

Listing 1: Conventional AM modulation

- I perform the latter demodulation scheme because it is more common.
- Half-wave rectification was arbitrarily chosen for my implementation of the demodulation.
- The LPF and HPF are the frequency response of first-order systems.
- A HPF with an arbitrarily low cutoff frequency was used to help alleviate a low-frequency noise caused by rectification.
- An additional "DC block" is performed by subtracting the mean of the filtered signal, since the HPF doesn't always perfectly center the signal.
- The constant 3.3 was empirically determined to recover the energy lost from the rectification and LPF. This constant scaler seemed to work pretty well for a variety of input samples. I wasn't able to find a way to do this analytically, especially because rectification is a nonlinear function.
- The original signal amplitude was recovered by dividing by $A_c a$.

Note that the SNR equation for conventional AM doesn't assume that the original amplitude is restored (i.e., by dividing by $A_c a$); thus, when computing SNR experimentally, we have to re-multiply the scaled output by $A_c a$.

```
% demodulates a signal modulated using the conventional AM scheme
% params:
% sig_c = conventional-AM modulated signal
% A_c   = carrier amplitude
% a     = modulation index
% f_s_c = carrier sampling rate (Hz)
% tau   = LPF cutoff (Hz)
% returns:
% sig_c = demodulated signal at lower frequency
function sig_m = conv_demod(sig_c, A_c, a, f_s_m, f_s_c, tau)
    % half-wave rectification
    sig_c(sig_c < 0) = 0;

    % low-pass filtering in freq. domain
    wd = linspace(-pi, pi, length(sig_c));
    f_c = wd * f_s_c / (2 * pi);
    lpf = (1 + f_c/tau*1j).^-1;             % LPF frequency response
    hpf = (1 - 20*f_c.^-1*1j).^-1;          % HPF to get rid of offset
                                            % and rectifier noise
    ft_c = fftshift(fft(sig_c)) / f_s_c;    % freq. domain rectified signal
    ft_c = lpf .^ 2 .* hpf .* ft_c;         % apply filter in freq. domain

    sig_c = ifft(ifftshift(ft_c) * f_s_c);

    % downsample
    duration = length(sig_c) / f_s_c;
    t_c = linspace(0, duration, f_s_c * duration);
    t_m = linspace(0, duration, f_s_m * duration);
    sig_m = real(interp1(t_c, sig_c, t_m));

    % DC block
    sig_m = sig_m - mean(sig_m);

    % scaling; the extra factor is empirically determined to account for
    % the power in the carrier lost when filtering
    sig_m = 3.3 * sig_m / A_c / a;
end
```

Listing 2: Conventional AM demodulation

2.2 SSB AM modulation

2.2.1 Overview

Single-sideband amplitude modulation aims to be more bandwidth efficient than the other AM schemes. It does this by employing the Hilbert transform to encode some of its information into a quadrature component. Given a message signal $m(t)$, the transmitted modulated signal looks like (some of the terms used in conventional AM will not be redefined here):

$$u_{(u|l)ssb}(t) = A_c \left(m(t)\cos(2\pi f_c t) \mp \hat{m}(t)\sin(2\pi f_c t)\right) + A_p \cos(2\pi f_c t) \tag{9}$$

Both modulation and demodulation occur using an I-Q (de)modulator. A pilot tone is necessary to ensure coherence, and a narrow-band filter and mixer are required in the demodulator to employ the pilot tone. These additions make SSB-AM also power-inefficient (as, like conventional AM, much of the power is in the carrier signal), and expensive (as it requires a mixer circuit and narrow-band filter.)

2.2.2 Analysis

Examining the power transmitted by this signal:

$$P_u = P_c + P_y \tag{10}$$

$$P_c = \frac{A_p^2}{2} \tag{11}$$

$$P_y = \frac{A_c^2}{2}P_m + \frac{A_c^2}{2}P_{\hat{m}} = A_c^2 P_m \tag{12}$$

Again, a large portion of the power is transmitted in the carrier signal. Twice as much power is transmitted in the message part of the signal as in conventional AM (assuming that $a = 1$), but there is also twice as much noise power, which contributes to a SNR that is equal to that of DSB SC (and roughly equal to that of conventional AM, assuming that $a = 1$). Let $r(t)$ again be the received signal,

$$r(t) = (A_c m(t) + n_I(t))\cos(2\pi f_c t) - (\pm A_c \hat{m}(t) + n_Q(t))\sin(2\pi f_c t) \tag{13}$$

$$\rightarrow \frac{A_c}{2}m(t) + \frac{1}{2}n_I(t) \qquad \text{(after mixing with cosine)}$$

$$P_y = \frac{A_c^2}{4}P_m \tag{14}$$

$$P_n = \frac{1}{4}N_0 W \tag{15}$$

$$\text{SNR} = \frac{A_c^2}{N_0 W}P_m \tag{16}$$

Again, the calculation for noise power was derived in the analog domain. See 10 for details about the digital domain calculation.

2.2.3 Implementation details

See 9.2 for the implementation of the Hilbert transform. The modulator follows the definition of SSB fairly closely, and easily allows for both USSB and LSSB. Notes on demodulation:

```
% modulates a signal using the SSB AM scheme; currently assumes phase
% coherence (no pilot tone in modulated signal)
% params:
% f_c   = carrier frequency (Hz)
% A_c   = carrier amplitude
% sig_m = message signal
% f_s_m = message sampling frequency (Hz)
% f_s_c = carrier sampling frequency (Hz)
% ussb  = whether it is USSB (true for USSB, false for LSSB)
% returns:
% sig_c = SSB AM-modulated signal
function sig_c = ssb_mod(f_c, A_c, sig_m, f_s_m, f_s_c, is_ussb)
    duration = length(sig_m) / f_s_m;
    t_m = linspace(0, duration, length(sig_m));
    t_c = linspace(0, duration, f_s_c * duration);

    % upsample sig_m
    sig_m_us = interp1(t_m, sig_m, t_c);

    % generate in-phase component (same as DSB-SC AM-modulated signal)
    sig_dsbsc = sig_m_us .* cos(2 * pi * f_c * t_c);

    % generate quadrature component
    sig_mhat = hilbert_transform(sig_m_us);
    sig_quad = sig_mhat .* sin(2 * pi * f_c * t_c);

    if is_ussb
        sig_c = A_c * (sig_dsbsc - sig_quad);
    else
        sig_c = A_c * (sig_dsbsc + sig_quad);
    end

    % add pilot tone
    A_p = 1;
    sig_c = sig_c + A_p * cos(2 * pi * f_c * t_c);
end
```

Listing 3: SSB AM modulation

- A narrow-band filter would be hard to realistically construct in MATLAB, and I did not get very far with my approximations for one (using only first-order filters). Thus, we abstract away these hardware details, assume phase coherence, and use a pure generated cosine and sine with no phase offset for the I-Q demodulation.
- A first-order LPF and HPF were designed for this function.

- The LPF is largely intended to filter out the double-carrier-frequency signal generated from I-Q mixing, and the HPF reduces any low-frequency noise and the DC component caused by the I-Q mixing.

```
% demodulates a signal modulated using the SSB AM scheme; currently
% assumes phase coherence (no pilot tone in modulated signal)
% params:
% sig_c = SSB AM-modulated signal
% f_c   = carrier frequency (Hz)
% A_c   = carrier amplitude
% f_s_m = baseband sampling frequency
% f_s_c = carrier sampling frequency (Hz)
% tau   = LPF cutoff frequency
% returns:
% sig_m = demodulated signal at baseband sampling frequency
function sig_m = ssb_demod(sig_c, f_c, A_c, f_s_m, f_s_c, tau)
    duration = length(sig_c) / f_s_c;
    t_c = linspace(0, duration, f_s_c * duration);
    t_m = linspace(0, duration, f_s_m * duration);

    % demodulate in-phase component
    sig_c = sig_c .* cos(2 * pi * f_c * t_c);

    % low-pass filtering in freq. domain; set cutoff frequency to carrier
    % frequency
    wd = linspace(-pi, pi, length(sig_c));
    f_c = wd * f_s_c / (2 * pi);
    lpf = (1 + f_c/tau*1j).^-1;             % LPF frequency response
    hpf = (1 - 20*f_c.^-1*1j).^-1;          % HPF to remove low frequency
                                            % distortion from pilot
    ft_c = fftshift(fft(sig_c)) / f_s_c;    % freq. domain rectified signal
    ft_c = lpf .* hpf .* ft_c;              % apply filter in freq. domain
    sig_c = ifft(ifftshift(ft_c) * f_s_c);

    % scale and downsample
    sig_m = real(interp1(t_c, sig_c, t_m));

    % scale back up to original amplitude
    sig_m = 2 * sig_m / A_c;
end
```

Listing 4: SSB AM demodulation

2.3 PM Modulation

2.3.1 Overview

Phase modulation involves encoding the message into the phase $\phi(t)$ of the carrier signal:

$$\phi(t) = k_p m(t) \tag{17}$$

where k_p is an arbitrary scaling constant. We also define the modulation index, β_p:

$$\beta_p \coloneqq k_p \max |m(t)| \tag{18}$$

This modulation index indicates the maximum phase deviation, $\Delta\phi_{max}$. If we adjust the message signal so that its maximum value has unity amplitude, it makes the maximum deviation simply k_p. The modulated signal looks like:

$$\begin{aligned} u_{pm}(t) &= A_c \cos(2\pi f_c t + \phi(t)) && (19)\\ &= A_c(\cos\phi\cos(2\pi f_c t) - \sin\phi\sin(2\pi f_c t)) && (20)\\ &\approx A_c(\cos(2\pi f_c t) - \phi\sin(2\pi f_c t)) \qquad (\phi \ll 1) && (21) \end{aligned}$$

These three forms of the equation lend them to three modes of demodulation (and also multiple modes of modulation).

- If we differentiate the form in (19) (e.g., using an inductor), then we get:

  $$u'_{pm}(t) = -A_c(2\pi f_c + \phi'(t))\sin(2\pi f_c t + \phi(t)) \tag{22}$$

  This has the same form as conventional AM, where the offset is $2\pi f_c$ (sufficiently large) and the message signal is $\phi'(t)$. By rectifying and LPF-ing, this can produce the envelope, $\phi'(t)$. This can then be integrated (e.g., by using a capacitor), then divided by k_p in order to recover the original message. This method would introduce some rectification noise, but is cheap to implement like conventional AM. This method also does not require sending an additional pilot tone.

- The form in (20) is an I-Q decomposition of the PM-modulated signal, which can be I-Q demodulated to reproduce $I = \cos\phi$ and $Q = \sin\phi$. ϕ can be recovered with $\arctan(Q/I)$, and then we can divide by k_p. I haven't tried implementing this, but I've heard from peers that the arctangent may sometimes introduce numeric instabilities using this method.

- (20) can be approximated as (21) with the assumption that $|\phi| \ll 1$. This is called the **narrow band approximation**. When demodulating, we only need recover the Q-component, and don't require the use of an arctangent function, so this is simpler than the second method.

For phase coherence, an additional pilot tone term $A_p\cos(2\pi f_c t)$ should be added. This is required for the latter two methods (I-Q demodulation), but is not required for the former (envelope detection).

PM-signal generation can be implemented as an I-Q demodulation (one of the two latter equation forms); the last form is easiest to implement, but only can be used when the modulation index is small. However, there are ways to use this method even with a wide band signal, namely by frequency division and multiplication.

Unlike amplitude-modulated signals, angle-modulated signals (PM and FM) take up infinite bandwidth because of their continuous frequency (phase) deviations. This can be seen mathematically as the Fourier transform of angle-modulated signals involves the Bessel functions, and thus have infinite bandwidth. However, Carson's rule states that the majority of an angle-modulated signal's power ($\geq 98\%$) lies in a bandwidth of:

$$W_{carson} = 2(\beta + 1)W \tag{23}$$

where $\beta = \beta_p$ for phase modulation and $\beta = \beta_f$ for frequency modulation.

2.3.2 Analysis

When assuming the narrow band approximation, we can estimate that both $A_c + A_p$ is the amplitude of the in-phase carrier, and $A_c\beta_p$ is the amplitude of the quadrature carrier. Thus:

$$P_u = \frac{(A_c + A_p)^2}{2} + \frac{A_c^2\beta_p^2 P_m}{2} \tag{24}$$

Since the modulated signal is narrow band (by assumption), then β_p is small and the first term dominates. I defer the derivation of the noise power and SNR to the class notes, or section 5.3 of [2].

$$P_m = k_p^2 P_m \tag{25}$$

$$P_n = \frac{2WN_0}{A_c^2} \tag{26}$$

$$\text{SNR} = \frac{k_p^2 A_c^2}{2}\frac{P_m}{N_0 W} = \frac{A_c^2}{2}\left(\frac{\beta_p}{\max|m(t)|}\right)^2 \frac{P_m}{N_0 W} \tag{27}$$

Again, the calculation for noise power was derived in the analog domain. See 10 for details about the digital domain calculation.

2.3.3 Implementation details

The modulation is fairly straightforward. We calculate the phase function, and add this phase pointwise to the carrier signal. (In practice, it would be easier to implement by mixing a narrow band signal with the carrier frequency using the narrow band approximation, but this is MATLAB.) A pilot tone is added to the signal.
Notes on demodulation:

- We use the narrow band approximation here in order to simplify the calculation. This means an I-Q demodulation (the third PM demodulation method described) by only mixing with the quadrature carrier.
- Similar to in SSB, we generated a pilot tone in the modulation "for show": this would be necessary for coherence with the I-Q demodulation, but actually extracting this tone in MATLAB with a realistic filter would be hard. Thus, again we don't use this pilot tone in the demodulator, but we generate a quadrature carrier (sine wave) and assume coherence.

```
% modulates a signal using the PM scheme
% params:
% f_c   = frequency of carrier
% A_c   = amplitude of carrier
% sig_m = message signal
% k     = k_p, phase deviation coefficient
% f_s_m = downsampled sampling rate
% f_s_c = updampled sampling rate
% returns:
% sig_c = PM-modulated signal at upsampled rate
function sig_c = pm_mod(f_c, A_c, sig_m, k, f_s_m, f_s_c)
    duration = length(sig_m) / f_s_m;
    t_m = linspace(0, duration, length(sig_m));
    t_c = linspace(0, duration, f_s_c * duration);

    % upsample sig_m, multiply by k to get phase
    sig_phi = k * interp1(t_m, sig_m, t_c);

    % generate phase-modulated signal w/ pilot tone
    A_p = 1;
    sig_c = A_c * cos(2 * pi * f_c * t_c + sig_phi) ...
          + A_p * cos(2 * pi * f_c * t_c);
end
```

Listing 5: PM modulation

- As with the previous modulation schemes, a first-order LPF and HPF were used to filter the components. The LPF was applied twice to get rid of some extra noise at higher frequencies.
- To regain the original amplitude, we multiply by a factor of 2. This can be explained by the power losses in the demodulation. Firstly, half of the power should be in the I component, which we completely disregard. Another half of the power is lost when mixing with the carrier. Losing power by a factor of 4 means an amplitude attenuation by a factor of 2.

```
% demodulates a signal modulated using the PM scheme
% params:
% f_c   = frequency of carrier
% A_c   = amplitude of carrier
% sig_c = PM-modulated signal
% k     = k_p, phase deviation coefficient
% f_s_m = downsampled sampling rate
% f_s_c = upsampled sampling rate
% tau   = LPF cutoff frequency
% returns:
% sig_m = demodulated signal at downsampled rate
function sig_m = pm_demod(f_c, A_c, sig_c, k, f_s_m, f_s_c, tau)
    duration = length(sig_c) / f_s_c;
    t_c = linspace(0, duration, f_s_c * duration);
    t_m = linspace(0, duration, f_s_m * duration);

    % mix with sine; negative because quadrature component is negative
    sig_c = -sig_c .* sin(2 * pi * f_c * t_c);

    % LPF and HPF
    wd = linspace(-pi, pi, length(sig_c));
    f_c = wd * f_s_c / (2 * pi);
    hpf = (1 - 20*f_c.^-1*1j).^-1;          % HPF
    lpf = (1 + f_c/tau*1j).^-1;             % LPF frequency response
    ft_c = fftshift(fft(sig_c));            % freq. domain rectified signal
    ft_c = lpf .^ 2 .* hpf .* ft_c;         % apply filter in freq. domain
    sig_c = ifft(ifftshift(ft_c));

    % downsample
    sig_m = real(interp1(t_c, sig_c, t_m) / (k * A_c));

    % restore amplitude
    sig_m = sig_m * 2;
end
```

Listing 6: PM demodulation

2.4 FM Modulation

2.4.1 Overview

Frequency modulation involves encoding the message into the frequency of the carrier signal. This first requires the concept of instantaneous frequency, which can be defined as a function of the phase ϕ:

$$f_{inst} = \frac{1}{2\pi}\frac{d\phi}{dt} \tag{28}$$

Now, say that we wish to encode the message signal as a in the frequency, i.e.:

$$f_{inst} = k_f m(t) \tag{29}$$

Here, k_f is the frequency deviation coefficient from the carrier (I.e., if the message has an amplitude of a, then the instantaneous frequency of the modulated signal will be $f_c + a$ at that moment; intuitively, $k_f \max|m(t)| = \Delta f_{max}$) Then we can express the phase as a function of the message signal:

$$\phi(t) = 2\pi k_f \int_{-\infty}^{t} m(t)\,dt \tag{30}$$

Then, we can apply the same principles behind PM to modulate this phase. This generates the modulated signal:

$$u_{fm}(t) = A_c \cos\left(2\pi f_c t + 2\pi k_f \int_{-\infty}^{t} m(t)\,dt\right) \tag{31}$$

Similar to PM, we can define a modulation factor k_f that indicates the phase shift:

$$\beta_f \coloneqq \frac{k_f \max|m(t)|}{W} = \frac{\Delta f_{max}}{f_{m,max}} \tag{32}$$

where W is the bandwidth of m(t) (i.e., the highest frequency of the baseband signal $m(t)$). This can intuitively be interpreted to mean the bandwidth efficiency of a modulated signal: a higher β_f indicates a higher carrier modulation for the same input bandwidth (and thus lower bandwidth efficiency). β_f can also be roughly interpreted as the maximum phase deviation (just like β_p for PM), since, for the pure tone $m_{pure}(t) = \cos(2\pi f_0 t)$ ($\max|m(t)| = 1$, $\beta = k_f/f_0$):

$$\begin{aligned} u(t) &= A_c \cos\left(2\pi f_c t + 2\pi k_p \int_{-\infty}^{\infty} \cos(2\pi f_0 t)\,dt\right) \\ &= A_c \cos\left(2\pi f_c t + \frac{2\pi k_p}{2\pi f_0}\sin(2\pi f_0 t)\right) = A_c \cos\left(2\pi f_c t + \beta_f \sin(2\pi f_0 t)\right) \end{aligned} \tag{33}$$

Clearly, $\beta_f = \Delta\phi_{max}$ in this case.

Demodulation of an FM-modulated signal can be performed identically to a PM demodulator, except that it would require an additional division by $2\pi k_f$ at the conclusion and a differentiation stage. Note that the first demodulation scheme involves a differentiation, envelope detection, and an integration stage; if we add another differentiation stage at the end of this, it would cancel out the integration stage and make it a simple two-step process very similar to conventional AM. Since differentiation and envelope detection are both easy to implement in hardware, this makes conventional AM and FM the most commonly implemented modulation schemes.

2.4.2 Analysis

As with PM, I defer the derivation of the noise power and SNR to the class notes, or section 5.3 of [2].

$$P_m = k_f^2 P_m \tag{34}$$

$$P_n = \frac{2W^3 N_0}{3A_c^2} \tag{35}$$

$$\text{SNR} = \frac{3k_f^2 A_c^2}{2W^2} \frac{P_m}{N_0 W} = \frac{3A_c^2}{2} \left(\frac{\beta_f}{\max |m(t)|} \right)^2 \frac{P_m}{N_0 W} \tag{36}$$

Again, the calculation for noise power was derived in the analog domain. See 10 for details about the digital domain calculation. What is a little confusing is that the bandwidth W in the first term in (36) can be in analog units (it should be the same value used when defining k_f), whereas the W in the second term indicates the cutoff frequency and should be in digital radian units when working in a sampled space. The equivalent expression using β_f is a little nicer because β_f is invariant to the domain (digital or analog), and so it hides this ugliness.

2.4.3 Implementation details

This uses the cheap method involving differentiation and conventional AM demodulation (envelope detection). The modulation implementation is pretty straightforward. The only design decision was to use `cumsum` as a simple running integral.
Notes about demodulation:

- Differentiation is performed in the time domain using a difference equation approximation. It is padded with a single zero to maintain the length.
- Like in the conventional AM case, half-wave rectification is arbitrarily chosen (as opposed to full-wave rectification).
- A first-order LPF and HPF were used to filter the circuit. As in the previous examples, the LPF is primarily to filter out the double frequency carrier and the DC components created by the carrier mixing.
- The LPF was arbitrarily applied multiple times, similar to the SSB case, in order to reduce additional noise at higher frequencies.

```
% modulates a signal using the FM scheme
% params:
% f_c   = frequency of carrier
% A_c   = amplitude of carrier
% sig_m = message signal
% k     = k_f, frequency deviation coefficient
% f_s_m = downsampled sampling rate
% f_s_c = updampled sampling rate
% returns:
% sig_c = FM-modulated signal at upsampled rate
function sig_c = fm_mod(f_c, A_c, sig_m, k, f_s_m, f_s_c)
    duration = length(sig_m) / f_s_m;
    t_m = linspace(0, duration, length(sig_m));
    t_c = linspace(0, duration, f_s_c * duration);

    % upsample sig_m, multiply by k, integrate and multiply by 2pi to get
    % phase (in rad)
    sig_phi = 2 * pi * k * interp1(t_m, cumsum(sig_m) / f_s_m, t_c);

    % generate phase-modulated signal (since FM is basically PM)
    sig_c = A_c * cos(2 * pi * f_c * t_c + sig_phi);
end
```

Listing 7: FM modulation

```
% demodulates a signal modulated using the FM scheme
% params:
% sig_m = message signal
% A_c   = amplitude of carrier
% f_s_m = downsampled sampling rate
% f_s_c = updampled sampling rate
% k     = k_f, frequency deviation coefficient
% tau   = LPF cutoff frequency
% returns:
% sig_m = FM-demodulated signal at downsampled rate
function sig_m = fm_demod(sig_c, A_c, f_s_m, f_s_c, k, tau)
    % differentiate (same as multiplying by jw in freq domain)
    sig_c = [(diff(sig_c) * f_s_c) 0];

    % rectify signal
    sig_c(sig_c < 0) = 0;

    % lpf and hpf
    wd = linspace(-pi, pi, length(sig_c));
    f_c = wd * f_s_c / (2 * pi);
    hpf = (1 - 20*f_c.^-1*1j).^-1;          % HPF
    lpf = (1 + f_c/tau*1j).^-1;             % LPF frequency response
    ft_c = fftshift(fft(sig_c));            % freq. domain rectified signal

    % apply LPF multiple times in a row to completely get rid of carrier
    % frequency
    ft_c = lpf .^ 2 .* hpf .* ft_c;         % apply filter in freq. domain
    sig_c = ifft(ifftshift(ft_c));

    % factor of 3.4 found empirically (similar to conventional AM demod)
    sig_c = 3.4 * real(sig_c) / (2 * pi * k * A_c);

    % downsample sig_c
    duration = length(sig_c) / f_s_c;
    t_m = linspace(0, duration, f_s_m * duration);
    t_c = linspace(0, duration, length(sig_c));
    sig_m = interp1(t_c, sig_c, t_m);
end
```

Listing 8: FM demodulation

2.5 Modulation schemes comparison

See Table 1 for a general comparison between modulation schemes.

	CONV	SSB	FM
PM	PM more robust to noise CONV cheaper	Similar I-Q demod scheme SSB more bandwidth efficient PM more robust to noise	FM cheaper FM more common
FM	FM more robust to noise Similar cheap demod scheme	FM cheaper SSB more bandwidth efficient PM more robust to noise	
SSB	CONV cheaper SSB use half bandwidth SSB transmits double signal power		

Table 1: Comparison of modulation schemes

Power Conventional AM uses half the power of SSB. Angle-modulation schemes are usually more power-efficient because they can increase β to increase SNR instead of increasing A_c.

Bandwidth Conventional AM uses double the bandwidth of SSB. Angle-modulation schemes are usually less bandwidth-efficient because they use a larger bandwidth (larger β) to increase SNR without increasing A_c.

SNR (and modulation index) In all of the schemes, increasing the carrier amplitude increases the SNR (but is less power efficient). In SSB AM, this is the only way to affect the SNR (assuming the message power is fixed). In conventional AM, increasing a also increases the SNR, but this also decreases power efficiency, and it is limited by the fact that $0 \leq a \leq 1$ so that no clipping occurs at demodulation. In the angle modulation schemes, in addition to the option of increasing the carrier amplitude, increasing the modulation index β increases SNR at no power cost, and is not as limited as a is; however, this uses a larger bandwidth.

2.6 "Fairly" comparing different modulation schemes

The "design problem" of this assignment is that there are multiple parameters to vary, and most do not affect the noiseless demodulated output (but affect the SNR when noise is added to the modulated signal). For example, the demodulated signal of a no-noise modulated signal is invariant to increases in A_c, but the SNR is directly related. In other words, if we set the criterion for choosing parameters to be such that the demodulated signal has the same power, there are infinitely-many possible combinations of parameters to choose from. This can be seen to make the estimated SNRs exactly the same in the simulations below (see Table 5), and the experimental SNR values are very close.

The assignment asks us to "ensure that the signal power will be equal at the output in each [modulated] scheme." This means that the criterion for choosing parameters is such that the *power of the component of the modulated signal encoding the message* is equal between each modulated scheme. This is easily done with conventional AM and SSB AM, where the power of the message

signal component P_y are given by (derived in their respective sections):

$$P_{y,conv} = \frac{A_c^2 a^2}{2} P_m \tag{37}$$

$$P_{y,ssb} = A_c^2 P_m \tag{38}$$

In this case, the modulation index a conventional AM is arbitrarily set to 0.5 (which allows us to double it later without worrying about clipping), and the carrier amplitude A_c is arbitrarily set to 4. To match this, in the SSB case $A_{c,ssb} \leftarrow A_{c,conv}/\sqrt{8}$.

For the angle-modulated cases, I am not sure how to solve it because of the complex nature of the Fourier transform and PSD (i.e., with the Bessel functions). Thus we give up on this idea of fair comparison and just give the FM and PM the same A_c as the conventional AM case.

The assumption that signal power will be equal in each modulation scheme is unfair because this is (probably) not what happens in the real world: different modulation schemes probably get transmitted at vastly different signal powers and different transmitted powers. In other words, I give up on the idea of directly trying to compare the different modulation schemes to each other with some absolute measure (e.g., comparing their SNRs at some given noise variance level to determine which is overall more resistant to noise), but rather observe the general trends w.r.t. changes in their parameters.

3 Results

	CONV	SSB	PM	FM
modulation index	$a = 0.5$	–	$k_p = 0.1$	$k_f = 10W \approx 60650$Hz
carrier amplitude A_c	4	$\sqrt{2}$	4	4
carrier frequency f_c	500000	500000	500000	400000
sampling rate of modulated signal $f_{s,c}$	2000000	2000000	5000000	5000000
USSB/LSSB	–	LSSB	–	–
noise variance (when varying modulation index)	1	–	1	0.5

Table 2: Parameters for modulation schemes

		CONV	SSB	PM	FM
Total transmitted power	Experimental	8.1044	0.6031	12.4990	8.000
	Theoretical	8.1042	0.6044	12.5000	8.000
	% error	0.00%	0.22%	0.01%	0.00%
Demodulated power	Experimental	0.0546	0.0513	0.0498	0.0553
	Theoretical	0.0522	0.0522	0.0522	0.0522
	% error	4.60%	1.72%	4.60%	5.94%

Table 3: Expected vs. theoretical power of modulated (transmitted) and demodulated signals

	PM	FM
% power of signal in Carson's bandwidth	100.00%	100.00%

Table 4: Percent power of signal in Carson's bandwidth for angle-modulated signals

		$\sigma = 0.1$	$\sigma = 0.5$	$\sigma = 1$
CONV SNR (dB)	Experimental	22.52	15.12	9.83
	Theoretical	31.99 (28.98†)	18.01 (15.00†)	11.99 (8.98†)
SSB SNR (dB)	Experimental	21.40	13.55	8.11
	Theoretical	31.99 (28.98†)	18.01 (15.00†)	11.99 (8.98†)
PM SNR (dB)	Experimental	20.93	9.66	3.79
	Theoretical	21.99	8.01	1.99
FM SNR (dB)	Experimental	19.46	9.97	2.32
	Theoretical	66.76	52.78	47.76

Table 5: Effect of varying noise variance on SNR. †using an extra scaling factor of 2

		Modulation index (a, β_p, or β_f)		
		0.5*original	original	2*original
CONV SNR (dB)	Experimental	4.06	9.83	14.79
	Theoretical	5.97 (2.96†)	11.99 (8.98†)	18.01 (15.00†)
PM SNR (dB)	Experimental	-2.26	3.72	9.64
	Theoretical	-4.03	1.98	8.01
FM SNR (dB)	Experimental	3.43	9.97	13.54
	Theoretical	45.93	51.95	57.97

Table 6: Effect of varying modulation index on SNR. †using an extra scaling factor of 2

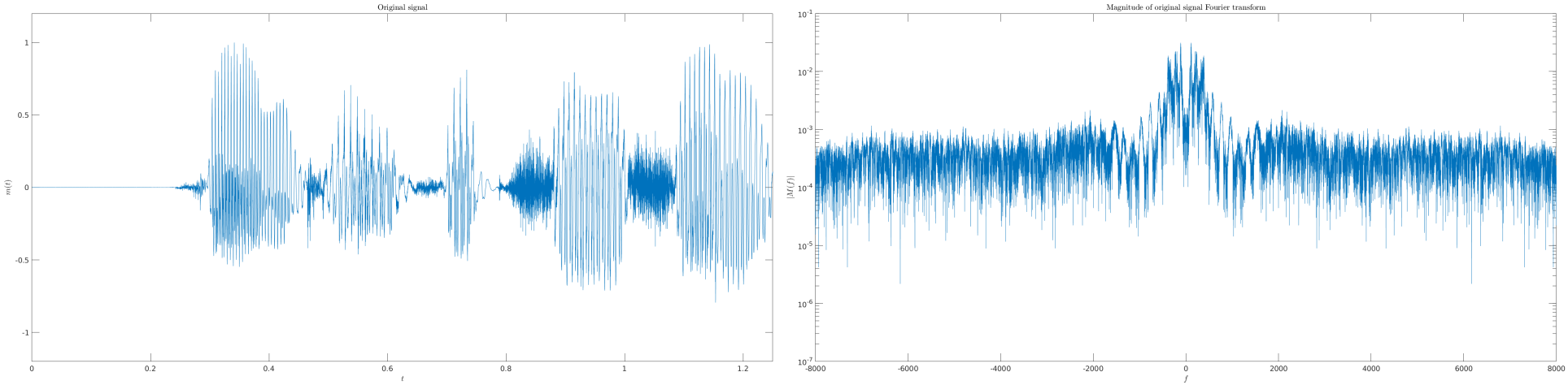

Figure 1: Original message signal

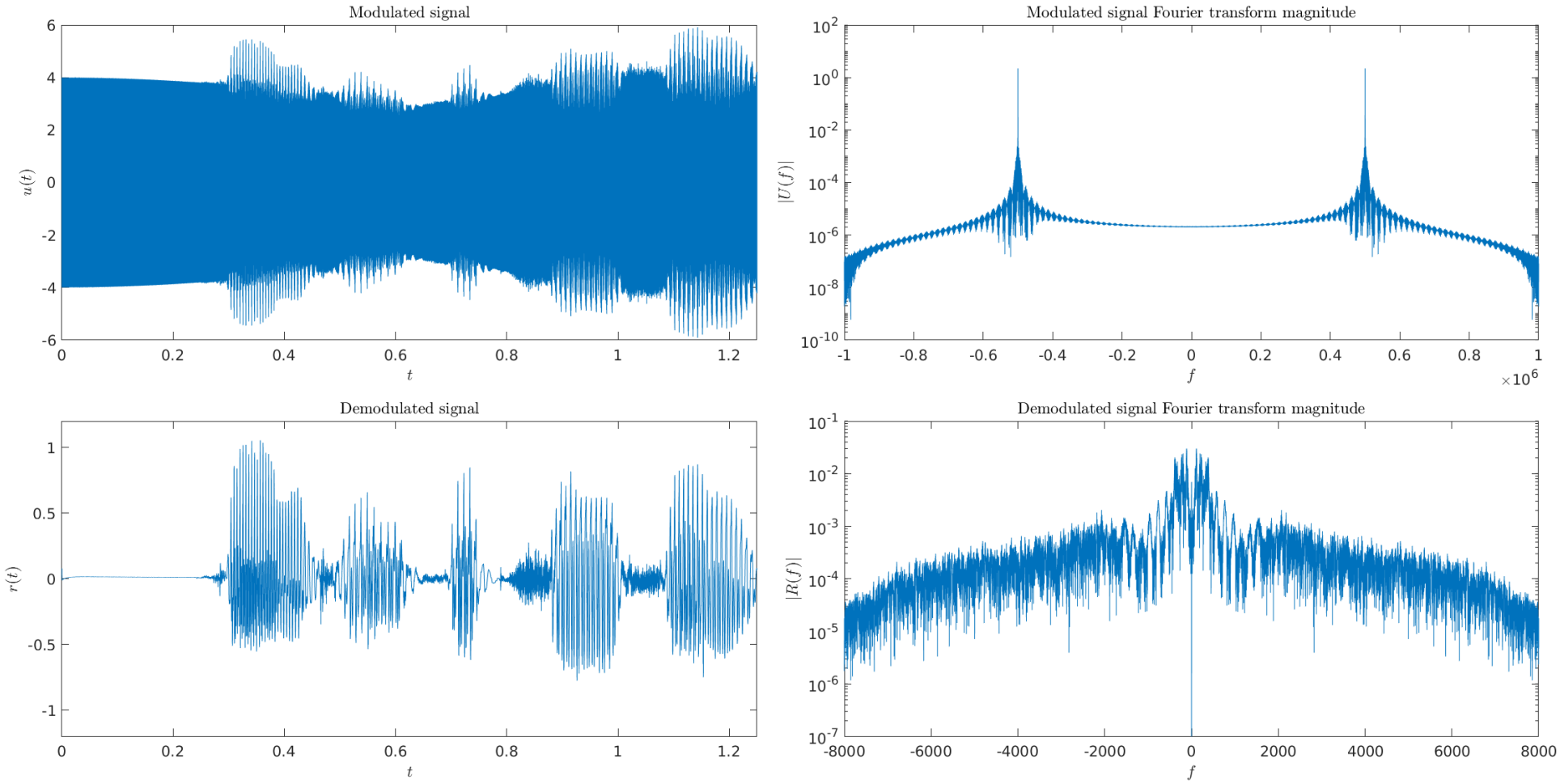

Figure 2: Conventional AM modulated and demodulated signals

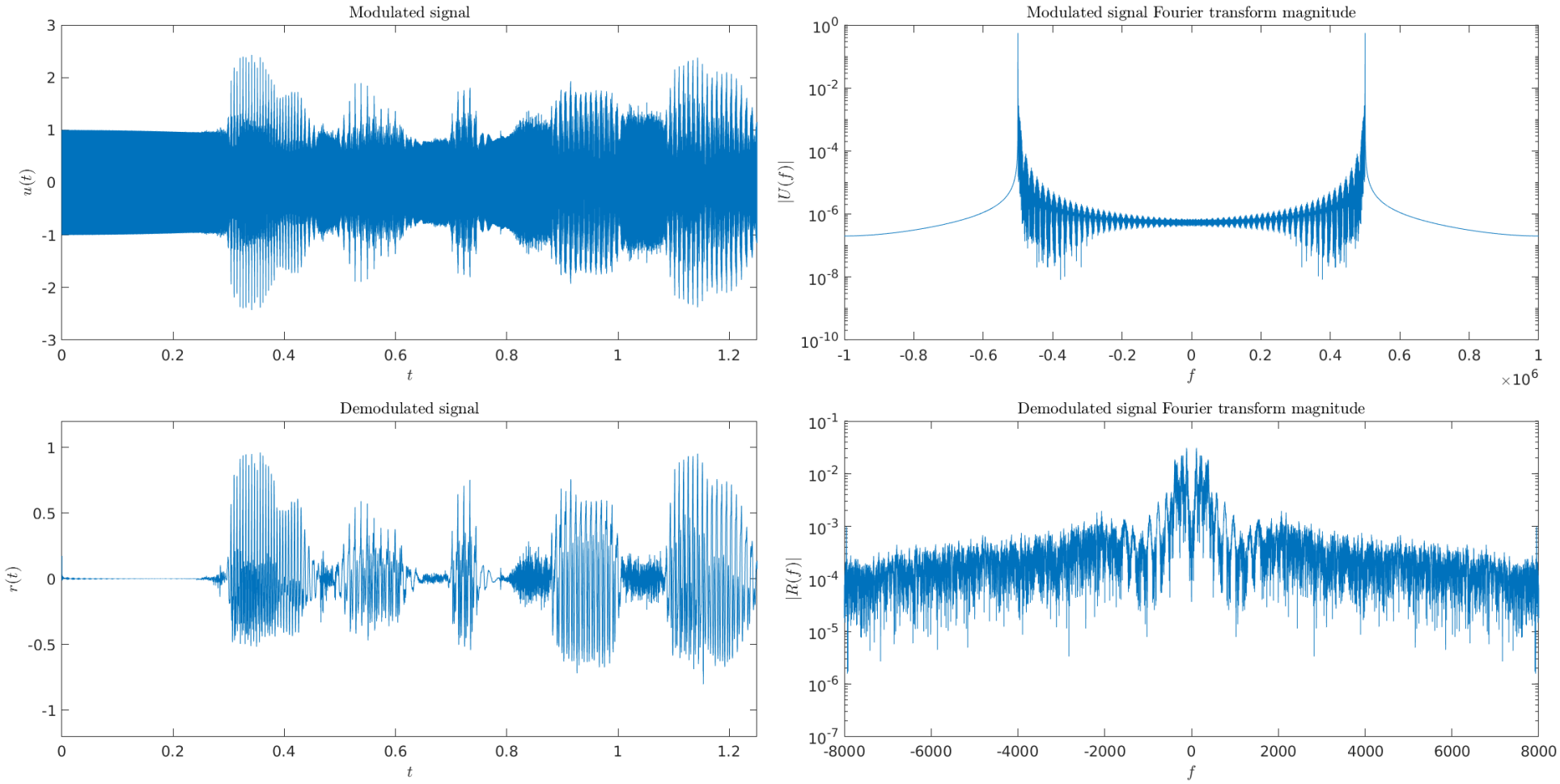

Figure 3: SSB AM modulated and demodulated signals

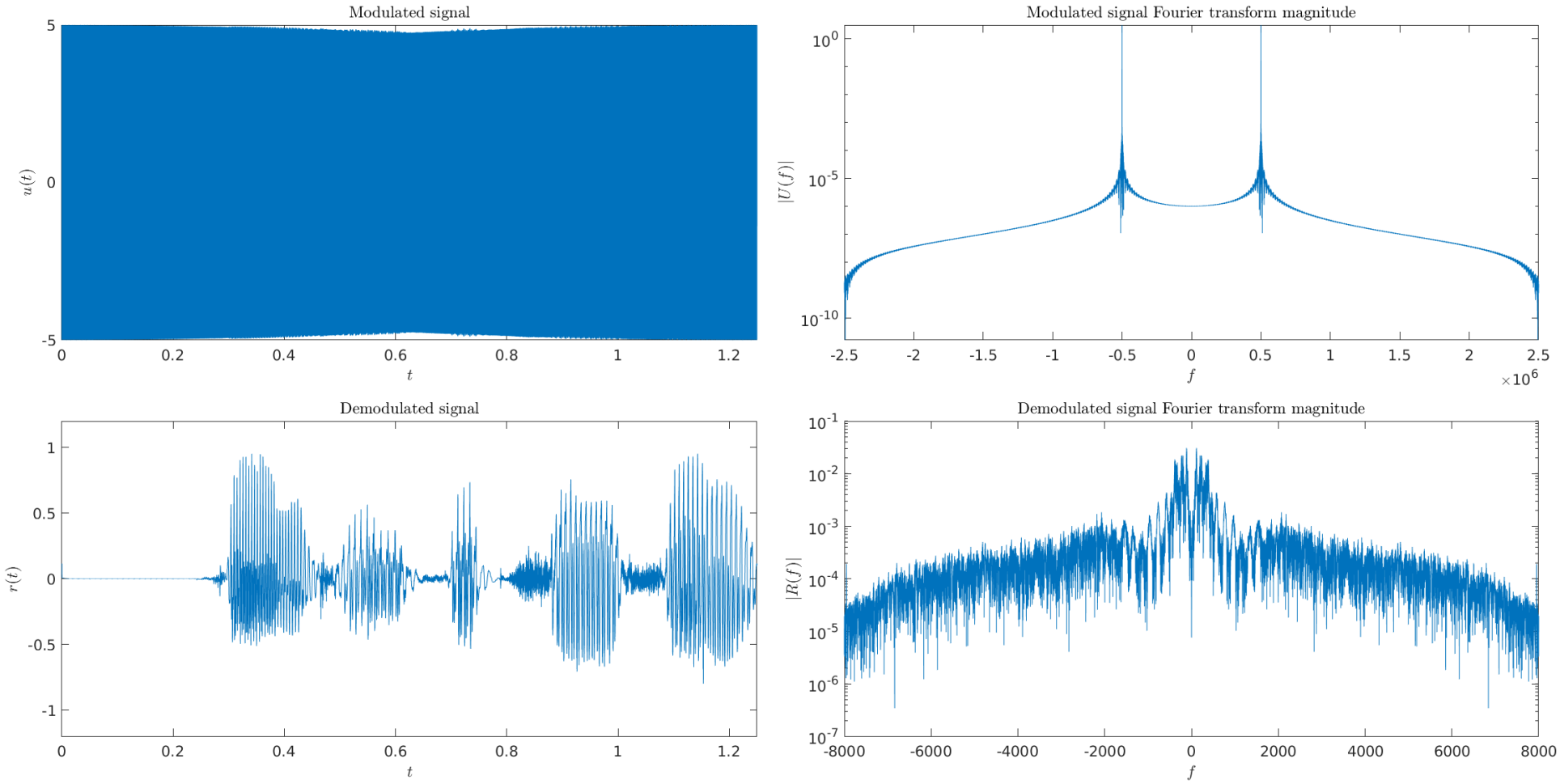

Figure 4: PM modulated and demodulated signals

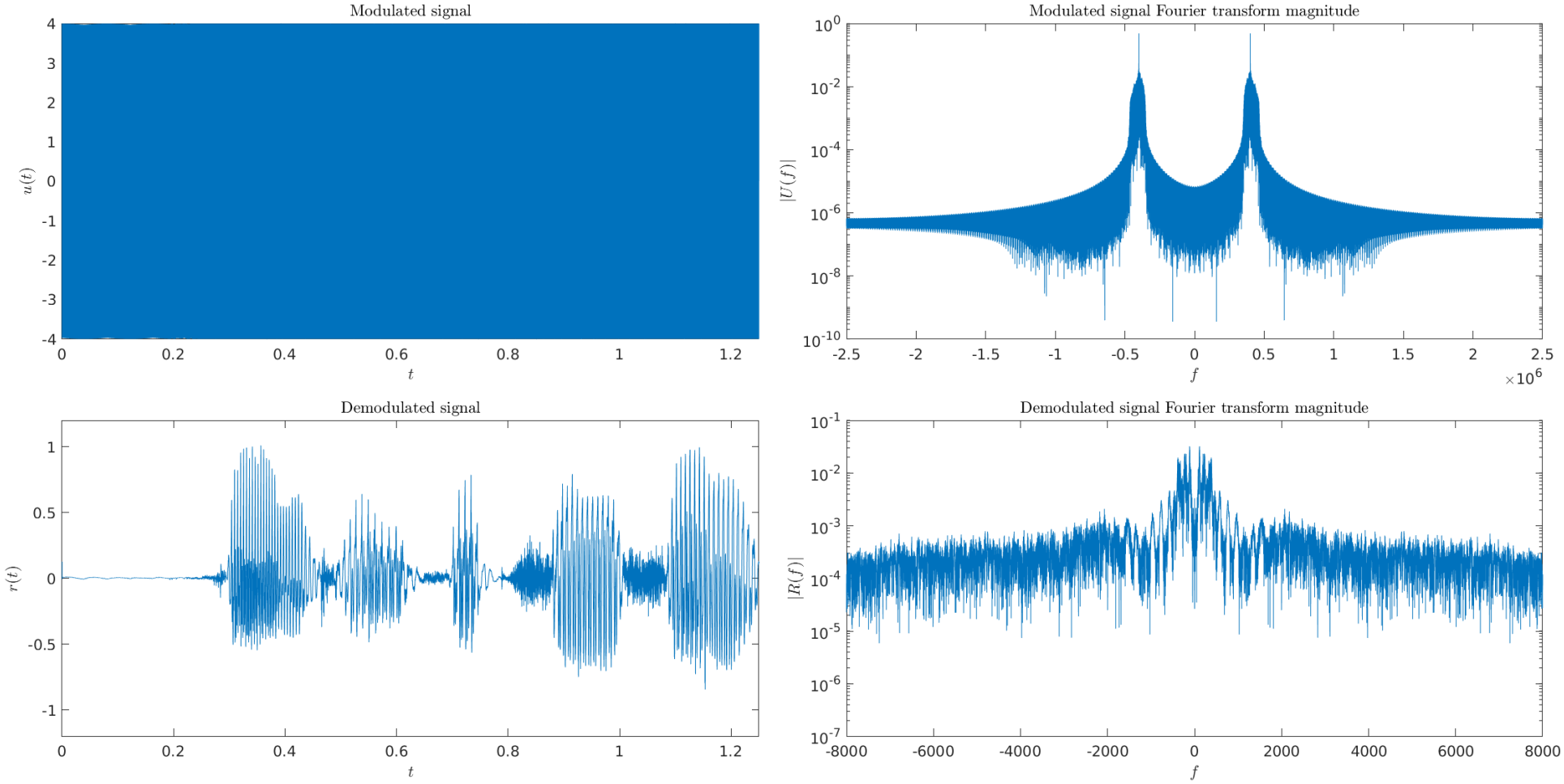

Figure 5: FM modulated and demodulated signals

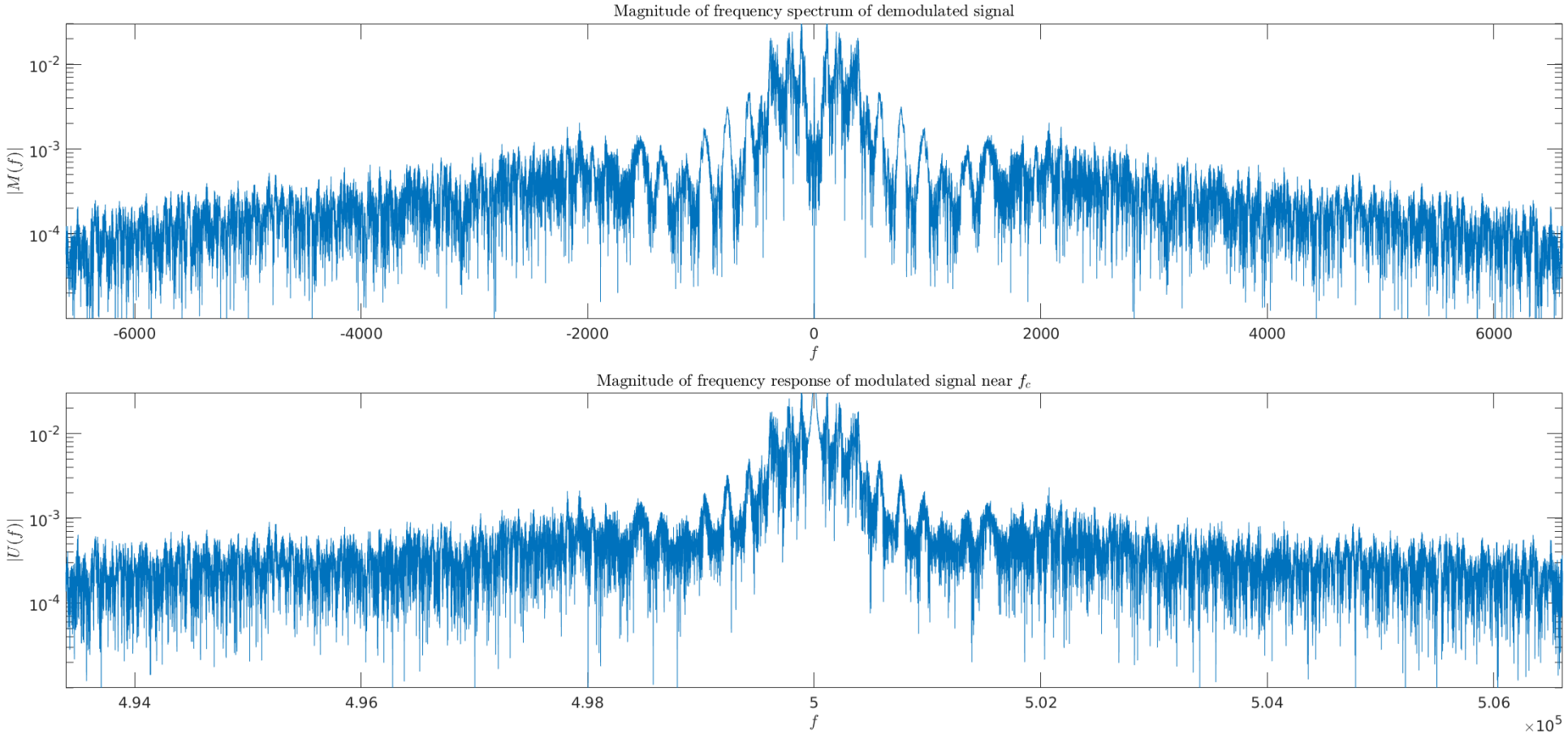

Figure 6: Close-up comparison of conventional AM baseband and modulated spectra

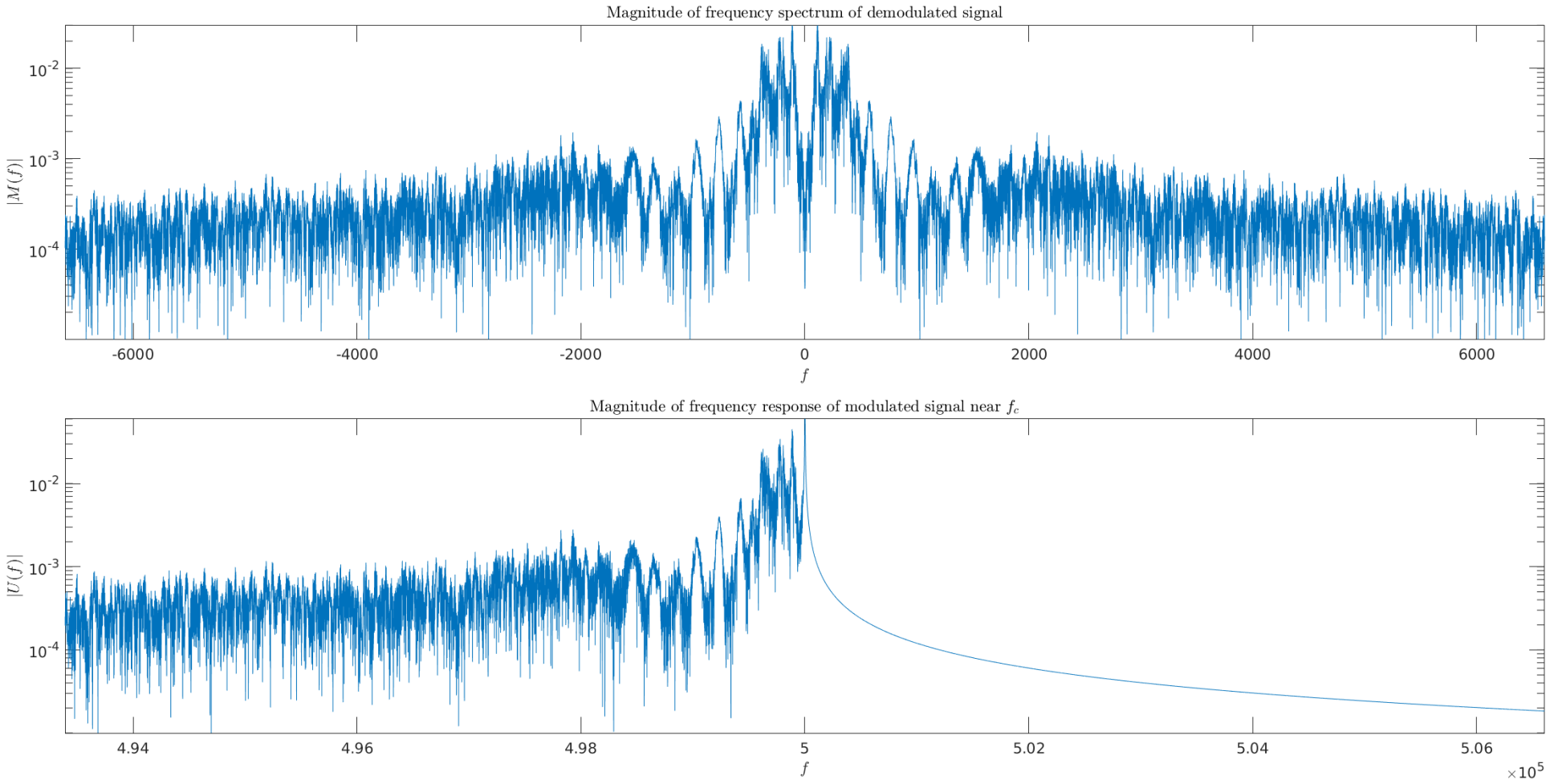

Figure 7: Close-up comparison of SSB AM baseband and modulated spectra

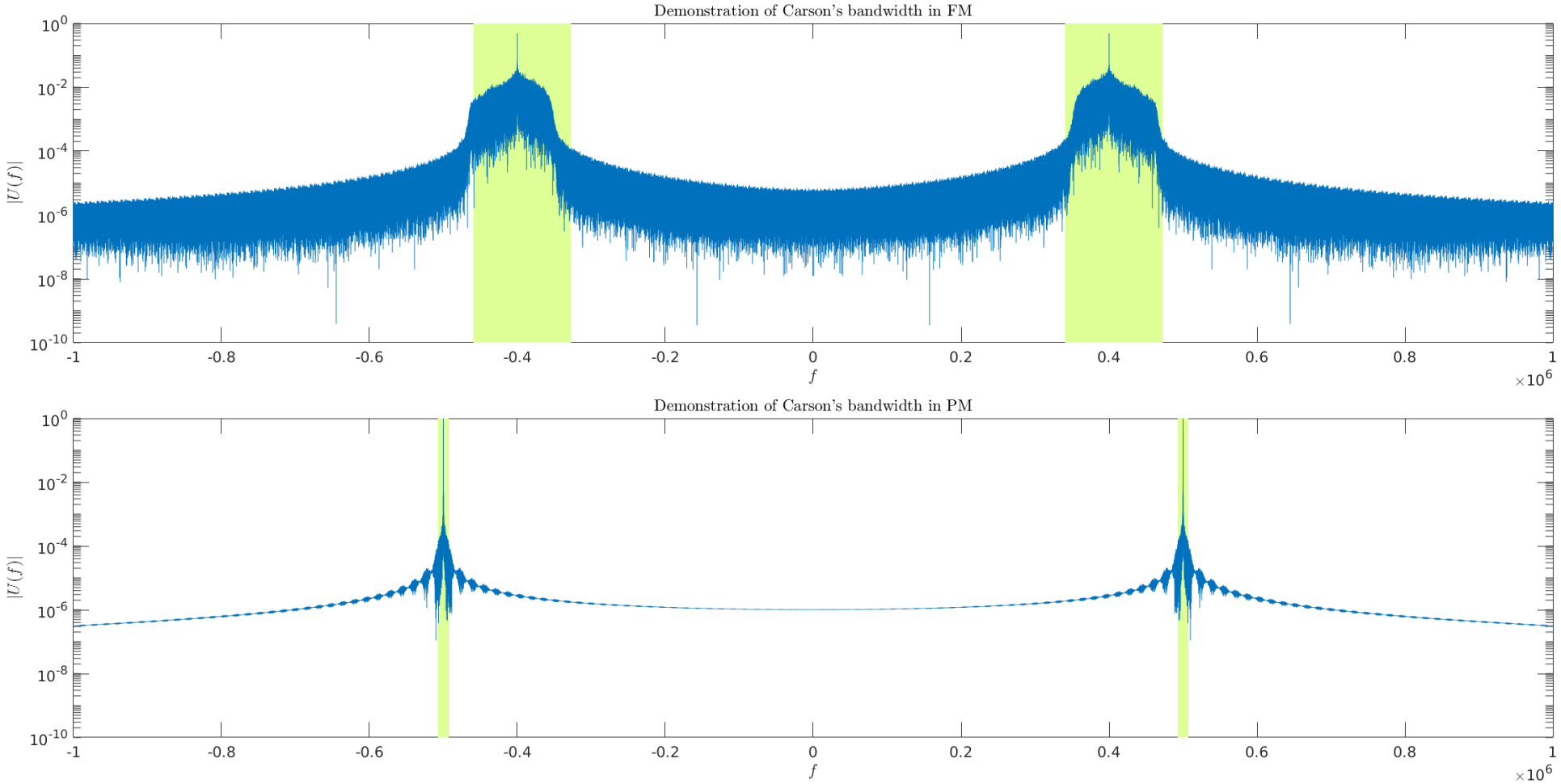

Figure 8: Demonstration of Carson's rule

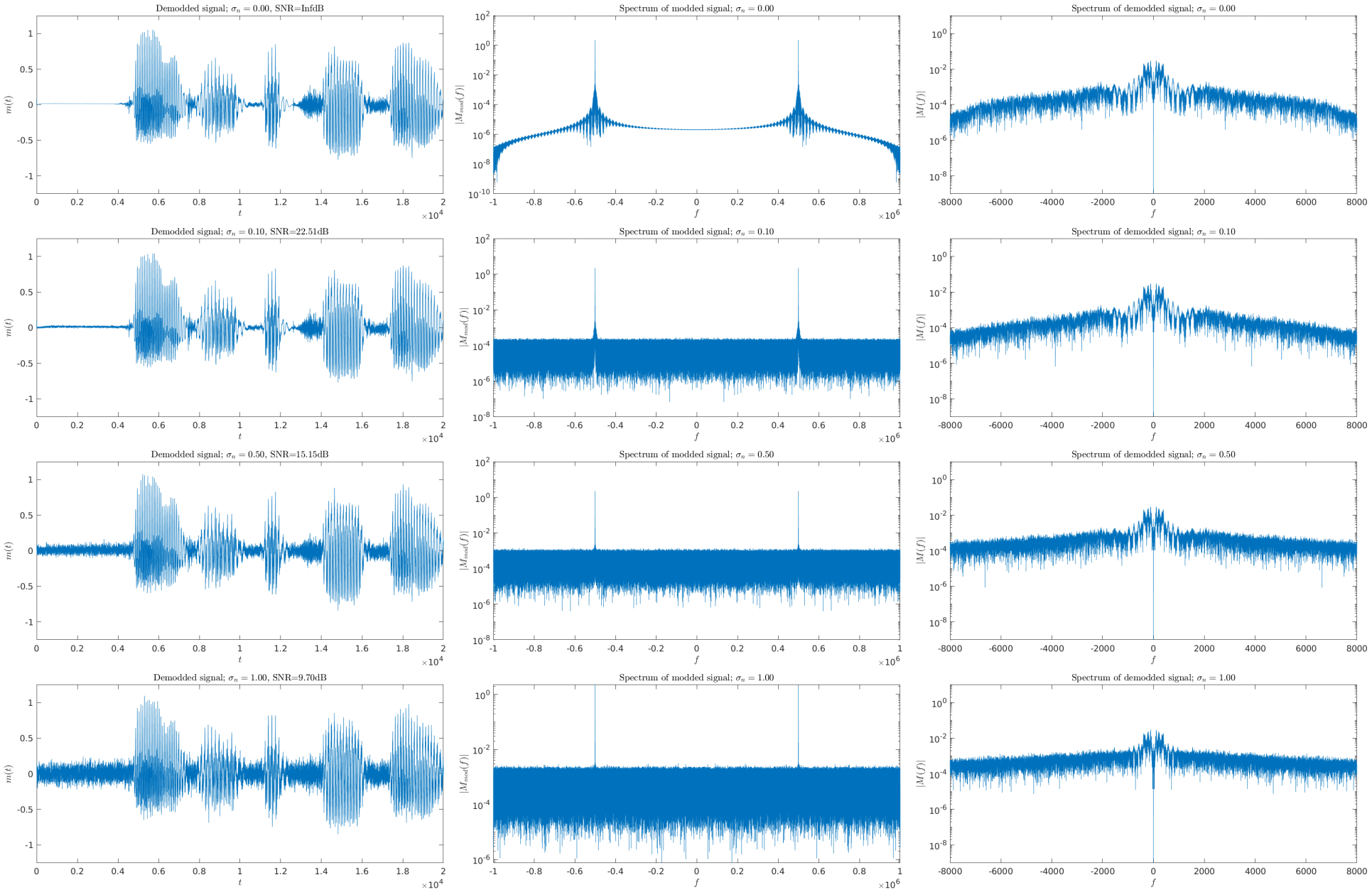

Figure 9: Effect of varying noise variance on conventional AM

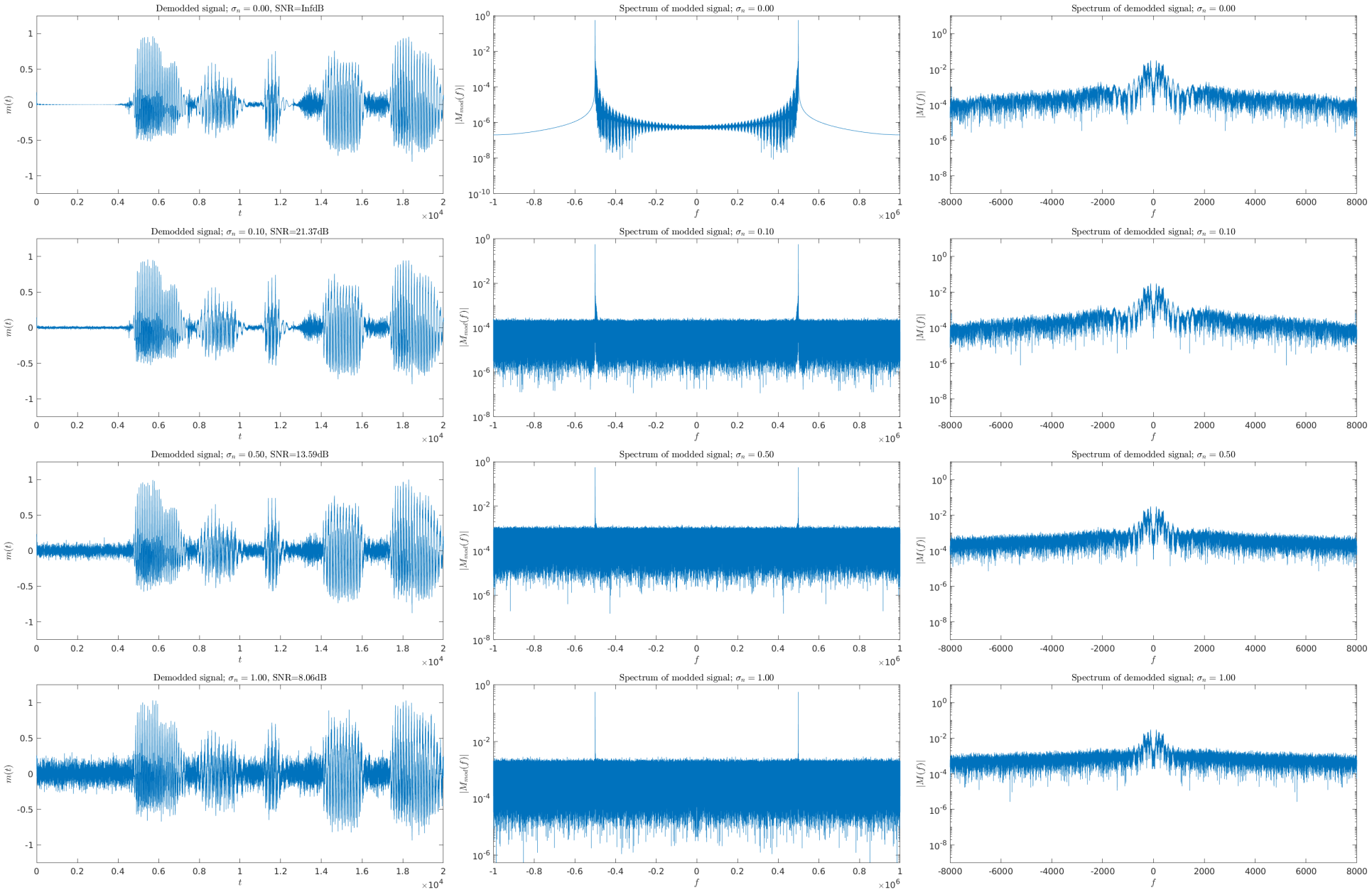

Figure 10: Effect of varying noise variance on SSB AM

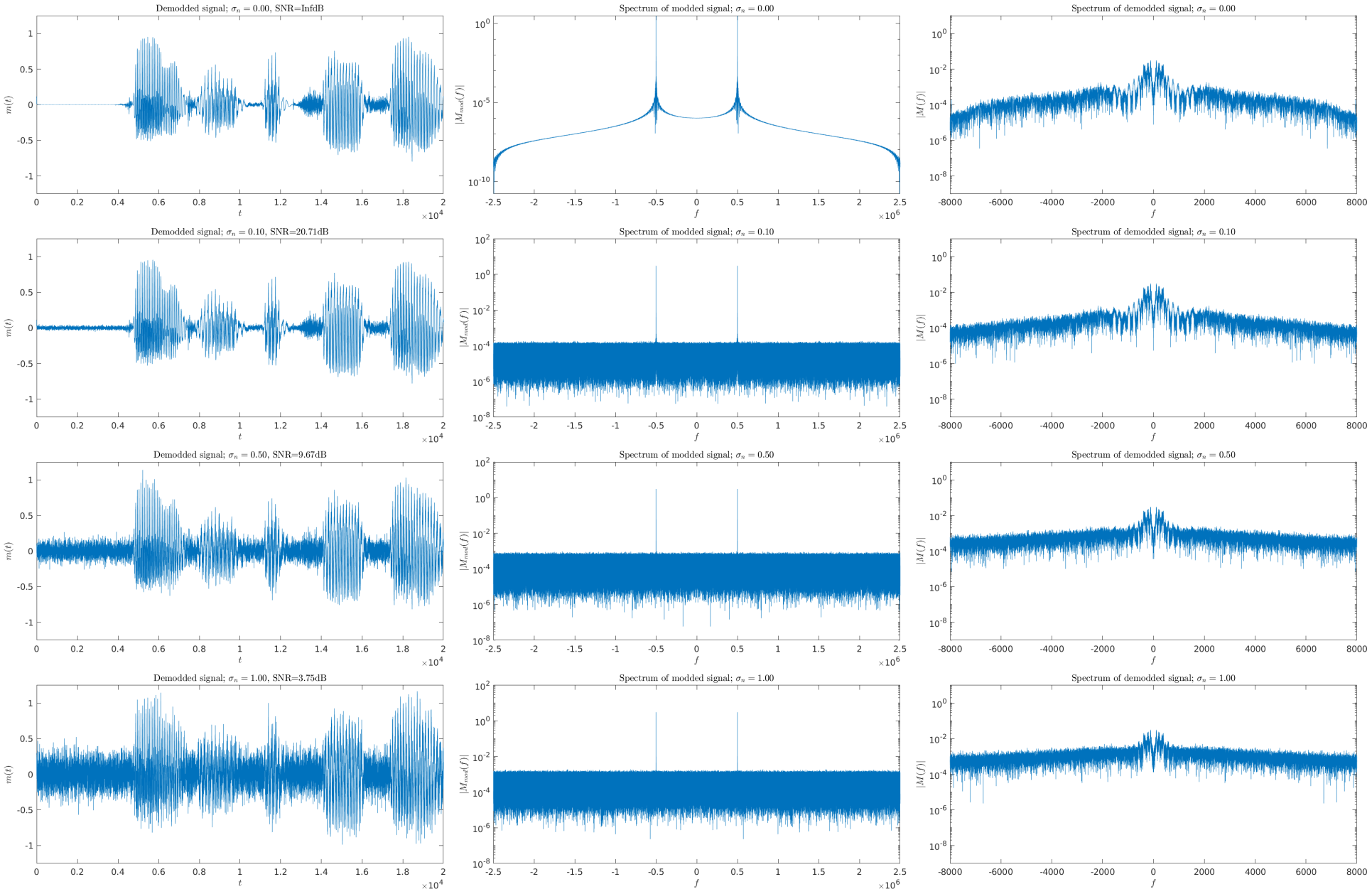

Figure 11: Effect of varying noise variance on PM

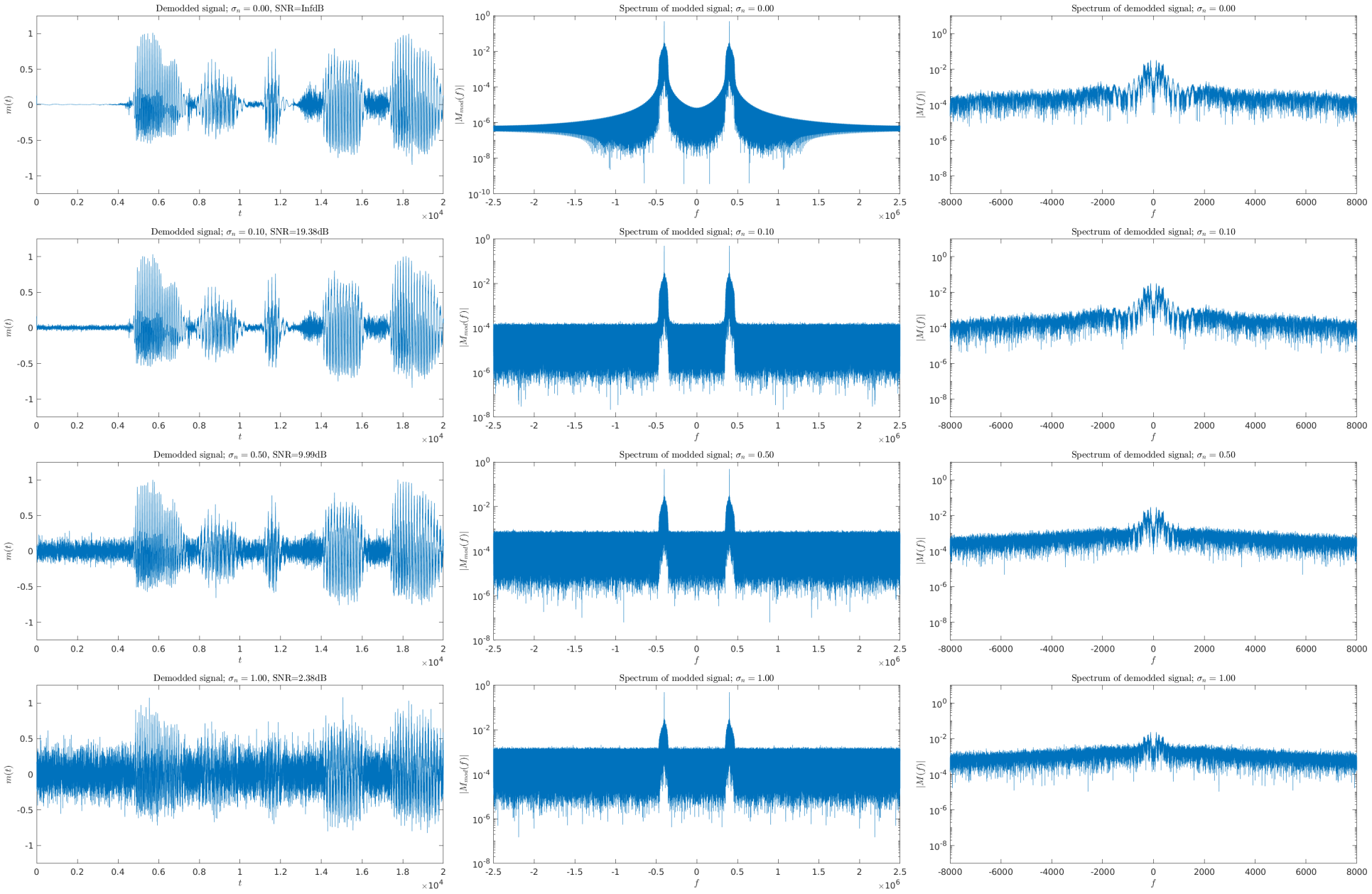

Figure 12: Effect of varying noise variance on FM

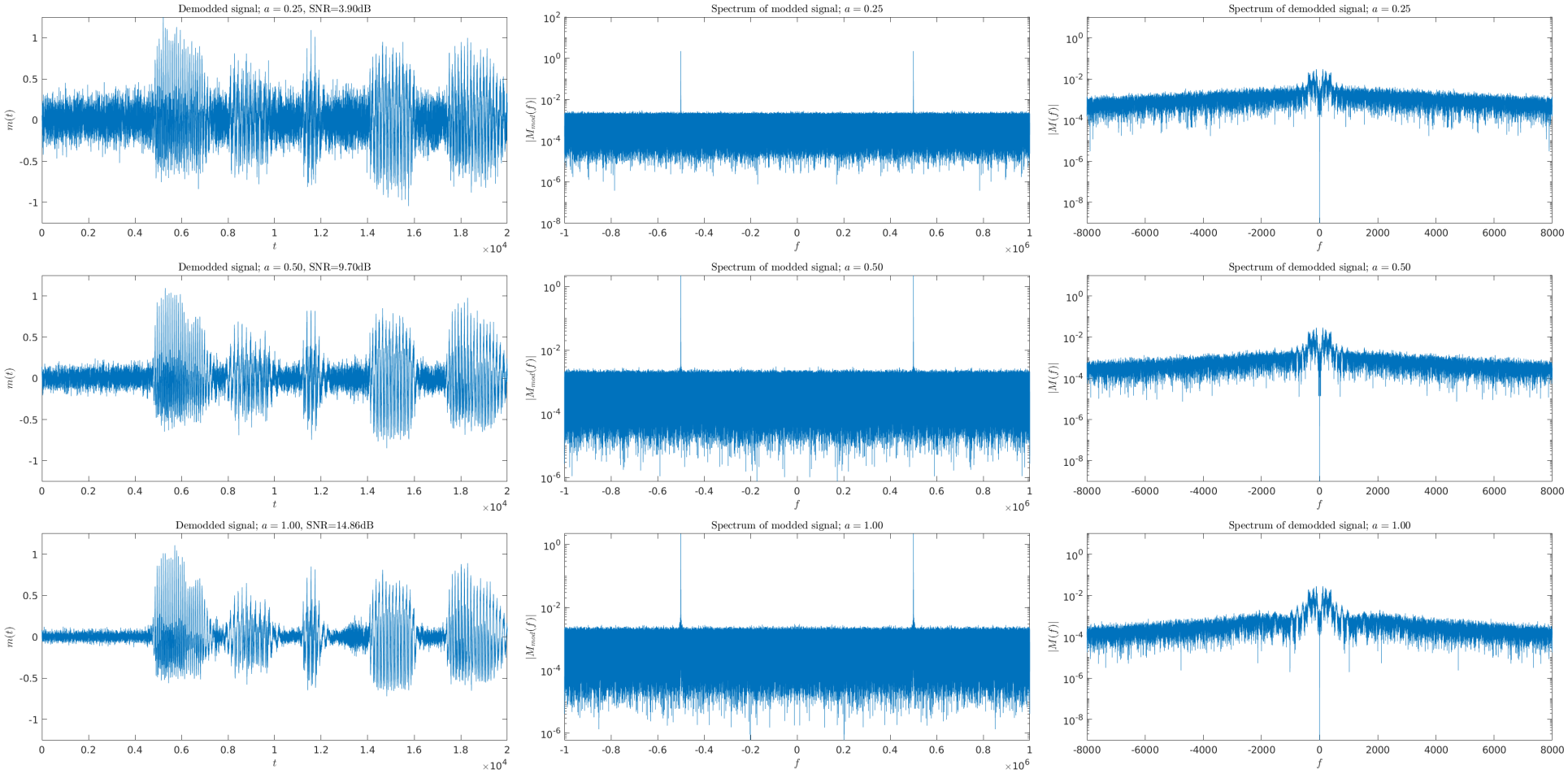

Figure 13: Effect of varying modulation index on conventional AM

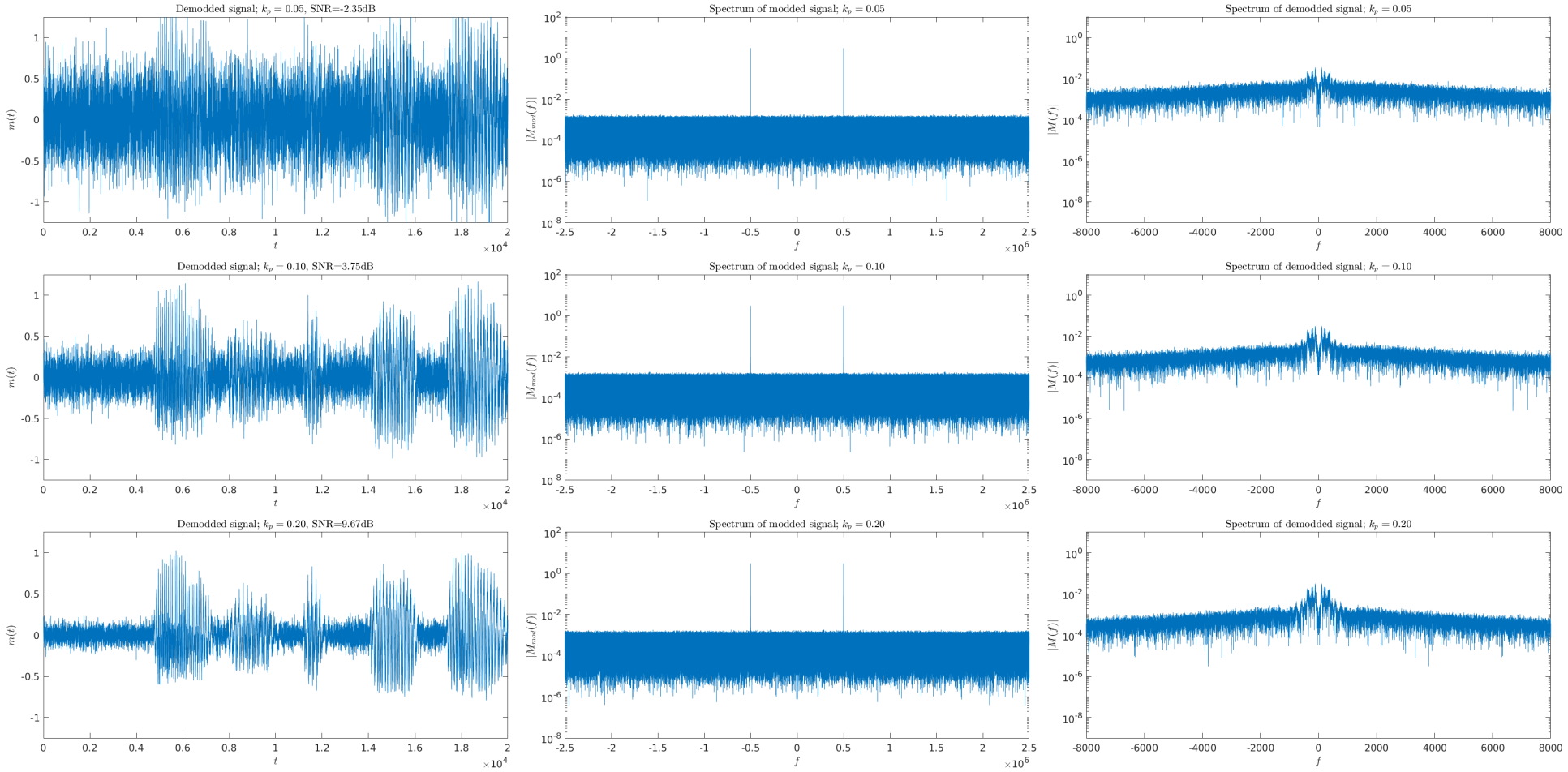

Figure 14: Effect of varying modulation index on PM

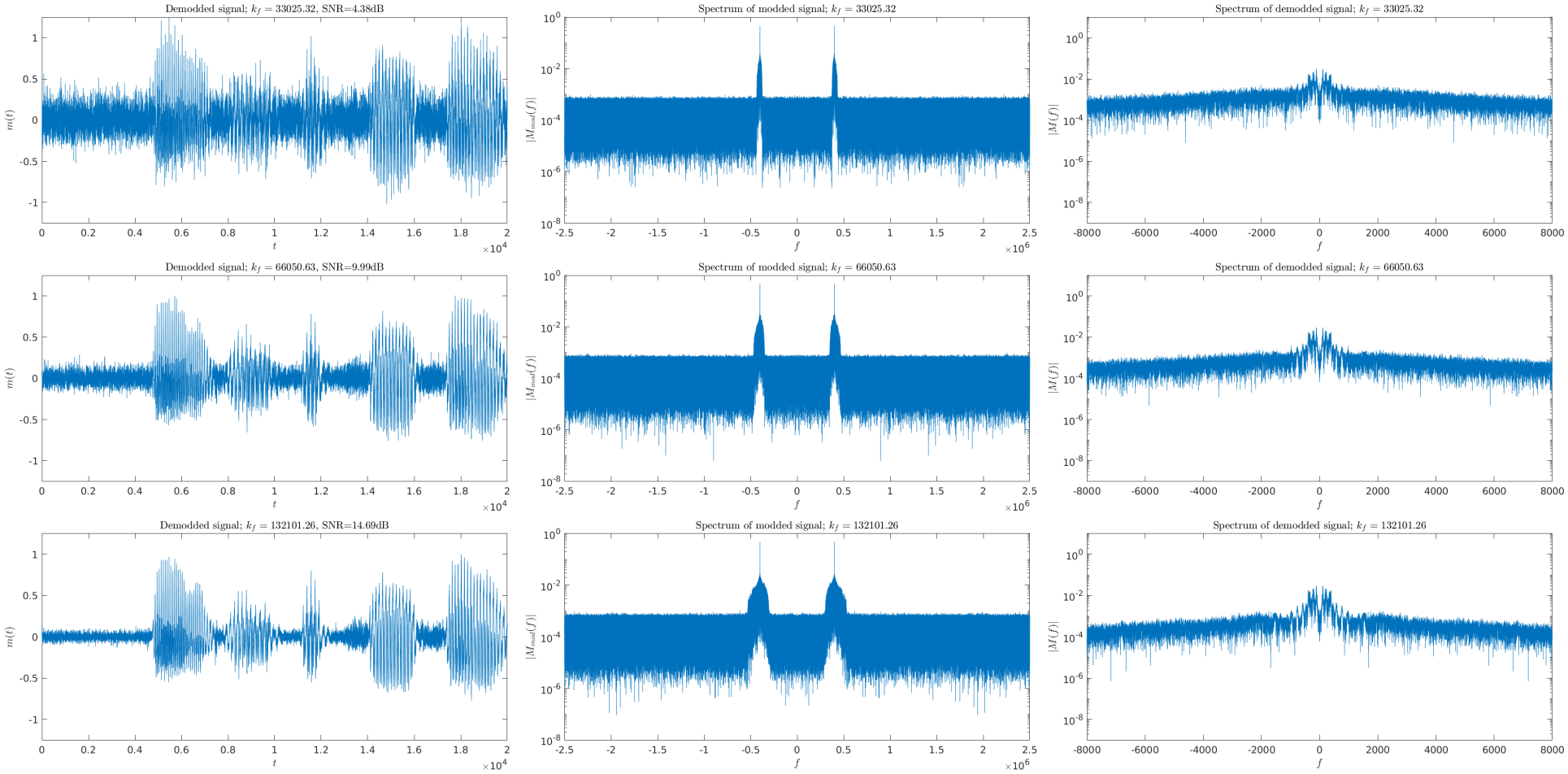

Figure 15: Effect of varying modulation index on FM

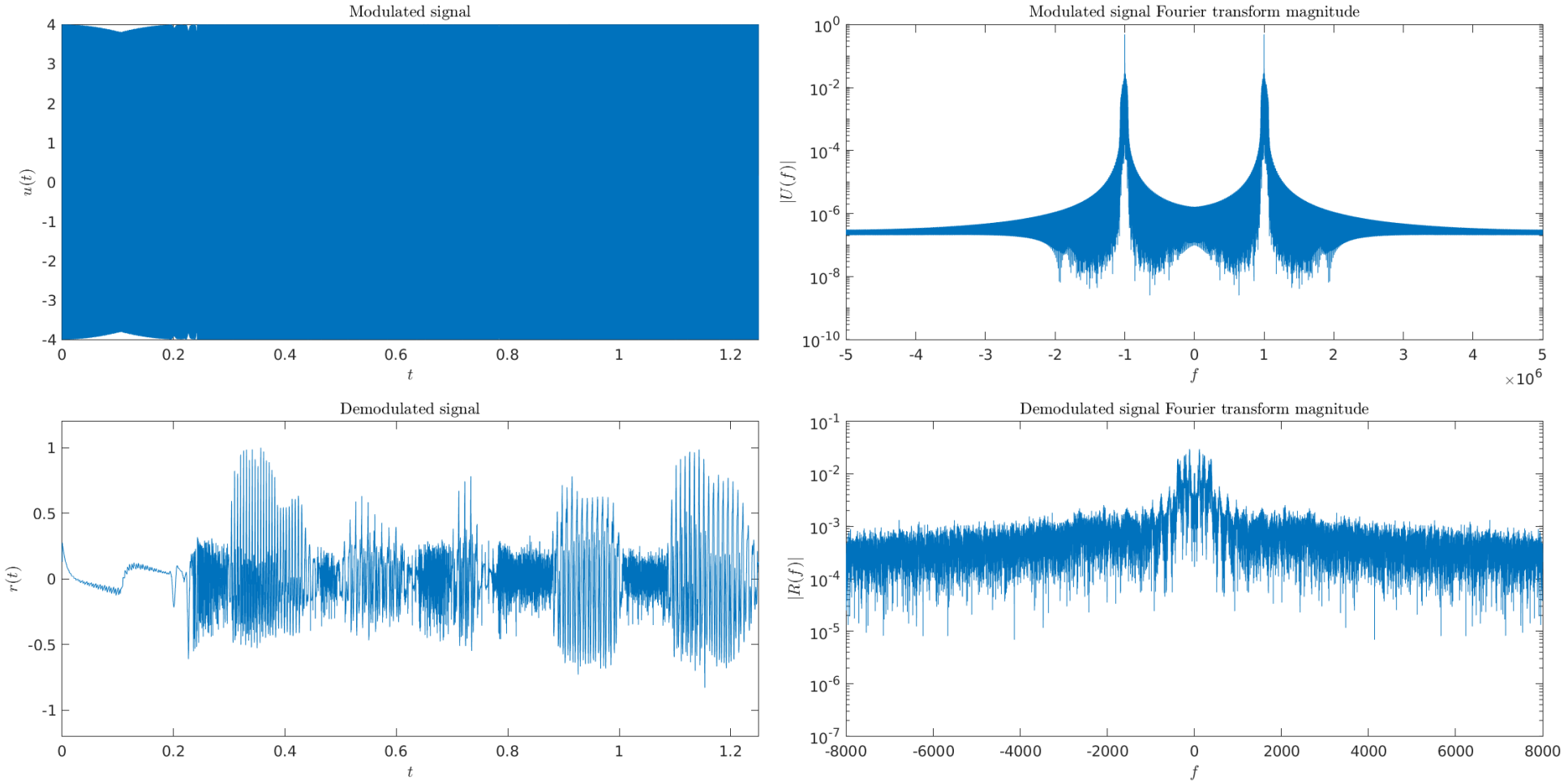

Figure 16: Increased distortion with higher frequencies with FM ($f_c = 1\text{MHz}$, $f_{s,sc} = 10\text{MHz}$)

4 Discussion

4.1 The message signal

The original message signal was scaled to have unity maximum amplitude, and it has a bandwidth (calculated using `obw`) of 6605Hz. Figure 1 is a plot of this signal and its Fourier transform.

4.2 Verification of the modulation schemes

The first order of business was to verify that each modulation and demodulation scheme works. The parameters for each of the modulation schemes is shown in Table 2. The modulated and demodulated signals were plotted in the time and frequency domains in Figures 2 (conventional AM), 3 (SSB AM), 4 (PM), and 5 (FM).

Also, the total transmitted power of the modulated signals, and the power of the demodulated signals, for each scheme are given in Table 3. It is clear from this table that the values match the theoretical values. (However, this doesn't mean much in the case of the demodulated conventional AM and demodulated FM cases, because there was an empirically-determined scalar multiplier in the demodulator used to restore the original amplitude.)

The outputs from each scheme are clearly audibly recognizable. Conventional AM and FM both have a very slight staticky sound, which is probably due to rectification noise.

4.2.1 Conventional AM

It is clear from the modulated signal plotted in the time domain that $A_c = 4$, and the message signal is clearly visible as the envelope of the modulated signal. In the Fourier transform of the modulated signal, we can see that there are two spikes at the carrier frequency as expected. The demodulated signal looks comparable to the original signal, as well as its Fourier transform; the only difference is that the DC offset is explicitly zero (from the demodulation function), so there is a (nonconsequential) negative spike at $f = 0$.

In addition, Figure 6 is a plot of the Fourier transform of the modulated signal shown zoomed horizontally and centered horizontally at $f = f_c$ and compared to the Fourier transform of the baseband message signal. This clearly demonstrates the function of mixing with a pure tone at the carrier frequency.

4.2.2 SSB AM

Similar to conventional AM, the envelope of the message signal is clearly visible in the carrier. The Fourier transform clearly shows a difference between the lower and upper sidebands, as expected. The output signal looks similar to the message signal in both time and frequency domain.

Similarly to conventional AM, Figure 7 is a plot of the Fourier transform of the modulate signal shown zoomed horizontally and centered horizontally at $f = f_c$ and compared to the Fourier transform of the baseband message signal. It is clear that all of the power is stored in the lower side band, as expected.

4.2.3 PM

As expected, the modulated signal is hard to see: the amplitude is more or less constant, with only a small visible distortion. The Fourier transform of the modulated signal shows narrow peaks at

the carrier frequency; they aren't quite as narrow as in the AM cases, but since $\beta_p = k_p = 0.1$ is small, they are fairly narrow. The demodulated output looks as expected.

4.2.4 FM

Much of the same arguments as PM apply. The main difference is that the Fourier transform of the modulated signal looks messier; this is because $\beta_p = 10$ is larger, so more bandwidth is used in this case.

4.3 Verifying Carson's rule for angle modulation

To do this, I simply overlaid a plot of the Carson bandwidth (shaded in green) on top of the plots of the modulated PM and FM signals in the frequency domain in Figure 8. These were calculated using the Carson bandwidth formula:

$$W_{carson} = 2(\beta + 1)W$$

It is visually apparent that most of the power lies in these bounds; to further verify this, I used the following snippet to experimentally calculate the amount of power in both Carson bandwidths (this is for PM, the result for FM is analogous):

```
bandpower(sig_pm_modulated, f_s_c_pm, ...
          [f_c_pm - (beta_p + 1) * W], f_c_pm + (beta_p + 1) * W) ...
    / bandpower(sig_pm_modulated)
```

As shown in Table 4, both calculations give approximately 100% of the power is in the Carson bandwidth.

4.4 Effect of varying noise variance on SNR

First, to review the SNR results from 2:

$$\begin{aligned}
SNR_{conv} &= \frac{a^2 A_c^2}{2} \frac{P_m}{N_0 W} \\
SNR_{ssb} &= A_c^2 \frac{P_m}{N_0 W} \\
SNR_{pm} &= \frac{A_c^2}{2} \left(\frac{\beta_p}{\max |m(t)|} \right)^2 \frac{P_m}{N_0 W} \\
SNR_{fm} &= \frac{3A_c^2}{2} \left(\frac{\beta_f}{\max |m(t)|} \right)^2 \frac{P_m}{N_0 W}
\end{aligned}$$

A few things to note:

- As shown in 10, the noise power $N_0 W$ in all of these analog domain equations should be replaced with $N_0 W / f_s$ for use in the digital (sampled) domain.
- In the case of a fixed message signal, P_m and W are fixed.

- a, A_c, β_p, β_f, and $(1/\sqrt{N_0} = \frac{1}{\sigma\sqrt{2}})$ are all related to the SNR by a quadratic relationship. Thus, changing any of these should have the same result on the SNR (albeit different results on power, bandwidth, clipping, etc.).

Three standard deviations were chosen (arbitrarily) for the noise process: 0.1, 0.5, and 1. The overall results for this section are summarized in Table 5. Plots for the modulation schemes can be found at Figures 9 (conventional AM), 10 (SSB AM), 11 (PM), 12 (FM).

It is visually evident from the plots that increasing the variance of the noise decreases the SNR, in all cases. For conventional AM, SSB AM, and PM, the experimental and the theoretical SNRs agreed fairly well.

Due to the nature of the parameter setup for conventional and SSB AM, which were set up so that the signal power in the modulated signal was the same, incidentally the calculated SNRs were also exactly the same. Experimentally, the SNRs between these two schemes turned out pretty close.

For the AM schemes, the theoretical SNR is systematically higher than the experimental SNR. Dan Kim suggested that dividing the theoretical SNR by a factor of 2 (providing a uniform ≈ -3dB shift) seems to make the experimental and theoretical results significantly closer. While the error doesn't seem to be a simple constant shift, the fact that there's an approximately constant decibel shift means that either the experimental noise power is systematically high or the experimental signal power is systematically low. I was debugging by probing the noise and signal powers at several points through the demodulation function, and found that the signal power was always more or less correct, so I suspect that the noise power was incorrect. I'm not sure where this error comes from, or if it's a constant, but here are a few possibilities:

- inherent quantization error
- rectification noise (in the case of conventional AM)
- half-wave rectification, which cuts the signal power in half (in the case of conventional AM)
- the LPF or HPF may cause the retained noise power to be disproportionally higher (w.r.t. signal power) than before filtering

The FM scheme experimental SNR was far from the theoretical value. Theoretically, the modulation index is large and the SNR should be very high. My guess is that there are many sources of numerical and other inherent error, e.g.:

- inherent quantization error
- numerical integration using `cumsum` during modulation
- numerical differentiation using `diff` during demodulation
- half-wave rectification noise during demodulation
- low-pass and high-pass filtering may cause distortions in the ratio of retained noise power to signal power

4.5 Effect of varying modulation index on SNR

The same notes from the previous section apply: decreasing σ by a factor of 2 or increasing modulation factor by a factor of 2 (ignoring clipping or other distortion) should cause the same change in SNR. In our specific case, having $\sigma = 0.5$, $a = 0.5$ in the conventional AM case would have the same SNR as $\sigma = 1$, $a = 1$. Indeed, the theoretical calculation produces the same number: 15.00 (with the 2 correction factor).

The overall results for this section are summarized in Table 6. Plots for the modulation schemes can be found at Figures 13 (conventional AM), 14 (PM), and 15 (FM).

Much of the analysis is exactly the same as in the previous section: the conventional AM, SSB AM, and PM had good agreement between experimental and theoretical SNRs. There was a similar systematic offset between the theoretical and experimental values that could be improved by adding the factor of 2.

4.6 Effect of higher carrier and sampling frequencies

I originally had a carrier frequency of 1MHz and a sampling frequency of 10MHz, but this caused a lot of distortion, which can be seen in Figure 16. It seems that higher frequencies can cause more errors with quantization errors than it might help by improving resolution. However, for the other modulation methods, I didn't have this problem. Therefore, my FM carrier and sampling frequencies are fairly low.

5 Conclusions

A reasonable implementation of multiple analog modulation schemes – conventional AM, SSB AM, FM, and PM – were implemented and simulated in MATLAB. Their operations were verified by a number of methods, including listening to the demodulated signal by ear, examining the modulated signal, checking the power of the modulated and demodulated signals, and comparing the demodulated signal to the original message signal.

These modulation and demodulation functions were used to empirically verify some of the theory. For example, the Carson bandwidth was loosely verified by a visual check and an estimated calculation. The effect of varying the variance of the AWGN noise, and of varying the modulation index, on the SNR was also explored and compared to theoretical values. Most of the calculated SNR values were fairly close to the theoretical values (i.e., most of the errors were less than 10dB); however, the empirical SNRs were systematically lower than the theoretical values. Most of the values were fairly close, but unfortunately I was unable to find a way to make the experimental SNR values for FM match the theoretical values.

These errors may be due to some combination of implementation errors, quantization/rounding errors, and inherent noise in the systems (especially in filtering, rectification, and numerical integration/differentiation stages). More time and effort would be needed to investigate these errors. Despite these errors, the general trends in SNR as a function of the changes in input parameters were correct.

6 Acknowledgments

I would like to acknowledge Prof. Brian Frost for answering my (and my peers') numerous late-night questions, hosting extra office hours, and extending the deadline in order to accommodate my peers and I for this project.

I would also like to acknowledge Anthony Belladonna, Dan Kim, Joshua Yoon, Thodoris Kapouranis, Paul Cucchiara, Philip Blumin, Derek Lee, Steven Lee, and other members of a particular Discord channel for discussing many aspects of the project and staying up with me multiple late nights to work on this project.

I would like to acknowledge Dan Kim for giving the suggestion that multiplying by two in the denominator of the conventional AM SNR equation, which gives closer estimates to the experimental values.

I would like to acknowledge Derek Lee for the following illustrious illustration:

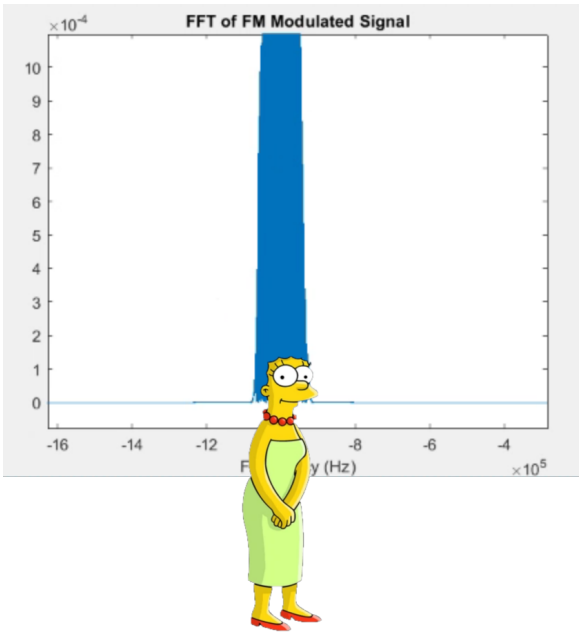

Figure 17: Untitled. Commissioned by Anthony Belladonna.

7 References

[1] "Radio Broadcast Signals." http://hyperphysics.phy-astr.gsu.edu/hbase/Audio/radio.html.

[2] Proakis, J. G., Salehi, M. (2001). *Communication Systems Engineering*. Upper Saddle River, NJ, USA: Prentice-Hall. ISBN: 0130617938

[3] "Sample Audio Files." https://www.opdsupport.com/downloads/miscellaneous/sample-audio-files.

8 Appendix I: Audio

The original audio message signal used throughout this report is a speech sample downloaded from Olympus Professional Dictation Support [3]. The audio is truncated to reduce memory and processor strain. The message is also normalized so that its maximum amplitude is 1, which makes calculating modulation indices simpler.

Sampling rate	16kHz
Bandwidth	6.61kHz
Samples	20000 (after truncation)
Duration	1.25s (after truncation)
Signal power	0.0522
$\max \lvert m(t) \rvert$	1

Table 7: Audio message signal statistics

```
%% Load audio and preprocess

% audio details
audio_filename = 'welcome.wav';

% truncate to limit memory usage and processing time
audio_samples = 20000;

% loads the message and the sampling rate of the message
[sig_m, f_s_m] = audioread(audio_filename);
sig_m = sig_m(1:audio_samples)';

% make maximum magnitude 1 to avoid problems with conventional AM
sig_m = sig_m / max(abs(sig_m));

% get bandwidth
W = obw(sig_m, f_s_m);

% get signal power
P_m = bandpower(sig_m);
```

Listing 9: Loading and preprocessing the audio signal

9 Appendix II: Auxiliary functions

9.1 Estimating signal power in MATLAB

In MATLAB, we only have discrete-time signals, with N discrete samples, sampling frequency F_s, and duration $T = N/F_s$. We can estimate the power of a signal numerically using the root-mean-squared function (squared) `rms` or `bandpower` functions, where `rms(signal)^2=bandpower(signal)`. To show the equivalence of this mean-squared function to the power of a discrete signal:

$$\begin{aligned} \text{rms}^2\{x[n]\} &= \frac{1}{N}\sum x^2[n] = \frac{1}{TF_s}\sum x^2[n] = \frac{1}{T}\sum x^2[n]\frac{1}{F_s} \\ &= \frac{1}{T}\sum x^2[n]\delta t \approx \frac{1}{T}\int x^2(t)\,dt = P_x \end{aligned} \tag{39}$$

Thus, the `bandpower` function was used throughout the program to estimate signal power.

9.2 Hilbert transform

The Hilbert transform is the LTI, norm-preserving magic sauce behind SSB. In layman's terms, this turns the negative part of the baseband signal into the quadrature part of the signal, which allows for packing the same signal as DSB AM into half of the bandwidth.

```
% hilbert transform of a signal; copied from earlier psets; for SSB AM
% params:
% sig_x = input signal
% returns:
% res   = hilbert transform of sig_x
function res = hilbert_transform(sig_x)
    ft_x = fft(sig_x);
    len_x = length(sig_x);

    % get signum of frequency; +1 for 0 to pi, -1 for -pi to 0
    sgn = [ones(1, floor(len_x/2)), -1*ones(1, ceil(len_x/2))];
    res = ifft(-1j * sgn .* ft_x);
end
```

Listing 10: Hilbert transform function

9.3 Lowpass signal

This function is used to lowpass signals by multiplying by the frequency response of a first-order LPF. Used for filtering out noise above the audible range, but can be used with any cutoff frequency.

```
% lowpass: set tau to upper range of audible range
function sig_c = lowpass_audible(sig, f_s, tau)
    wd = linspace(-pi, pi, length(sig));
    f = wd * f_s / (2 * pi);
    ft = fftshift(fft(sig));

    % first-order LPF with cutoff frequency at tau
    lpf = (1 + f/tau*1j).^-1;
    ft = lpf .* ft;

    sig_c = real(ifft(ifftshift(ft)));
end
```

Listing 11: Lowpass function

9.4 Fourier transform plotter

This function provides the frequency axis and Fourier transform of a signal.

```
% helper function to plot the fourier transform of a signal
% params:
% x     = signal
% f_s   = sampling rate
% returns:
% f     = frequency axis
% ft    = fourier transform
function [f, ft] = plot_ft(sig, f_s)
    wd = linspace(-pi, pi, length(sig));
    f = wd * f_s / (2 * pi);
    ft = abs(fftshift(fft(sig)) / f_s);
end
```

Listing 12: Fourier transform plotting function

9.5 Saving figures to PDFs

This function saves figures to PDFs, with arbitrary page dimensions.

```
% helper function for printing to PDFs
function fig_save(fig_title, paper_size)
    global pset_name;
    pset_name = 'ece300_proj1';

    set(gcf(), ...
        'Units', 'centimeters', ...
        'Position', [0 0 paper_size], ...
        'InnerPosition', [0 0 paper_size], ...
        'OuterPosition', [0 0 paper_size], ...
        'PaperPositionMode', 'auto');
    exportgraphics(gcf(), ...
                    sprintf('%s_fig_%s.png', pset_name, fig_title), ...
                    'ContentType', 'image');
end
```

Listing 13: Figure saver helper function

10 Appendix III: Calculating noise power in the digital domain

An AWGN random process $N(t)$ with variance σ^2 has the following autocorrelation function:

$$R_{N,dig}[n] = \sigma^2 \delta[n] \tag{40}$$

By the Weiner-Khinchin theorem, the PSD (in digital radians) of the AWGN noise is given by the DFT of the autocorrelation function:

$$S_{N,dig}(\omega) = \mathcal{F}\{R_N[n]\}(\omega) = \frac{\sigma^2}{2\pi} \tag{41}$$

The total power of the noise process before filtering is the integral of the PSD over its domain:

$$P_{N,dig} = \int_{-\pi}^{\pi} \frac{\sigma^2}{2\pi}\, d\omega = \sigma^2 \tag{42}$$

This can be confirmed empirically by finding the bandpower of an arbitrary AWGN noise signal. E.g., when $\sigma^2 = 4$ `bandpower(2 * randn(1, 1000000))` outputs 4. When we want to find the power of the noise in the baseband (demodulated) signal, then we assume a perfect filter with digital cutoff frequency ω_{cut} (i.e., this would be set to the bandwidth of the message signal). Then we have a rectangular pulse PSD centered at $\omega = 0$, with height $\sigma^2/(2\pi)$ and width $2\omega_{cut}$. The noise power at baseband is the integral (area) under this rectangle, i.e.:

$$P_{N,bb,dig} = \int_{-\pi}^{\pi} (\text{rect}_{\{-\omega_{cut},\omega_{cut}\}})(S_N(\omega))\, d\omega = \frac{\sigma^2}{2\pi}(2\omega_{cut}) \tag{43}$$

Making the conventional substitution for the noise power, $N_0 = 2\sigma^2$:

$$P_{N,bb,dig} = \frac{N_0}{2\pi}\omega_{cut} = \frac{N_0}{2\pi}\frac{2\pi f_{cut}}{f_s} = N_0 \frac{f_{cut}}{f_s} \tag{44}$$

where f_s is the sampling frequency. The final expression here is the most intuitive: it is the product of the noise power with the ratio of the cutoff frequency to the sampling frequency. This contrasts with the analog case, which was covered in class:

$$R_{N,ana}(t) = \sigma^2 \delta(n) \tag{45}$$

$$S_{N,ana}(f) = \sigma^2 \tag{46}$$

$$P_{N,ana} = \int_{-\infty}^{\infty} S_{N,ana}(f)\, df = \infty \tag{47}$$

$$P_{N,bb,ana} = \int_{-f_{cut}}^{f_{cut}} S_{N,ana}(f)\, df = N_0 f_{cut} \tag{48}$$

(If analog angular frequency was used instead of analog linear frequency, then there would also be the additional factor of $1/(2\pi)$.) Both cases involve clipping a flat noise PSD to that within the bandwidth specified by the cutoff frequency.

In summary: a noise power (i.e., in SNR equations) of $N_0 f_{cut}$ (or, equivalently, $N_0 W$) in the analog domain should be replaced with $N_0 f_{cut}/f_s = N_0 \omega_{cut}/(2\pi)$.

Note that most of the demodulation methods involve a LPF at the upsampled rate (i.e., the sampling rate used to sample the modulated signal), and the signal is passed through an additional LPF at the downsampled rate (i.e., the sampling rate used for the baseband audio) to filter out inaudible high-frequency noise. Note that downsampling changes the magnitude of the PSD.

11 Appendix IV: Source code

The core functions have already been defined. Here is the rest of the code used to produce the plots and results in this document.

```
close all; clear; clc;
set(0, 'defaultTextInterpreter', 'latex');

%% Project details
global pset_name;
pset_name = 'ece300_proj1';

% audio details
audio_filename = 'welcome.wav';
audio_samples = 20000;

%% Load audio

% loads y, Fs
[sig_m, f_s_m] = audioread(audio_filename);
sig_m = sig_m(1:audio_samples)';

% make maximum magnitude 1 to avoid problems with conventional AM
sig_m = sig_m / max(abs(sig_m));

% plot audio
figure('visible', 'off');
tiledlayout(1, 2, 'TileSpacing', 'Compact', 'Padding', 'Compact');

duration = length(sig_m) / f_s_m;
t_m = linspace(0, duration, length(sig_m));

nexttile();
plot(t_m, sig_m);
title('Original signal');
ylabel('$m(t)$');
xlabel('$t$');
xlim([0, duration]);
ylim([-1.2, 1.2]);

[f, ft] = plot_ft(sig_m, f_s_m);
nexttile();
semilogy(f, ft);
title('Magnitude of original signal Fourier transform');
ylabel('$|M(f)|$');
xlabel('$f$');
ylim([10e-8, 10e-2]);

fig_save('sig_m', [80 20]);

% get bandwidth
W = obw(sig_m, f_s_m);

% get signal power
P_m = bandpower(sig_m);

% write original signal to filesig_mod
audiowrite('original.wav', sig_m, f_s_m);

%% Modulation configuration

% conventional AM
a_conv = 0.5;
A_c_conv = 4;
f_c_conv = 500000;
f_s_c_conv = 2000000;

% SSB AM
f_c_ssb = 500000;
A_c_ssb = A_c_conv / sqrt(8);
f_s_c_ssb = 2000000;
is_ussb = false;

% PM
f_c_pm = 500000;
A_c_pm = A_c_conv;
f_s_c_pm = 5000000;
k_p = 0.1;

% FM
f_c_fm = 1e6; %400000;
A_c_fm = A_c_conv;
f_s_c_fm = 1e7; %5000000;
k_f = 10 * W;

% for when viewing the effect of varying noise std.
noise_stds = [0 0.1 0.5 1];

% for when viewing the effect of varying modulation index
noise_std_conv = 1;
noise_std_fm = 0.5;
noise_std_pm = 1;

% for storing the theoretical snrs
var_noise_theo = zeros(4,4);
var_noise_emp = zeros(4,4);
var_mod_ind_theo = zeros(3,3);
var_mod_ind_emp = zeros(3, 3);

%% conventional AM modulation
sig_conv_modded = conv_mod(f_c_conv, A_c_conv, a_conv, ...
    sig_m, f_s_m, f_s_c_conv);
sig_conv_demodded = conv_demod(sig_conv_modded, A_c_conv, ...
    a_conv, f_s_m, f_s_c_conv, W);
plot_modded_demodded(sig_conv_modded, sig_conv_demodded, ...
    f_s_m, f_s_c_conv, 'conv');
audiowrite('conv_demodded.wav', sig_conv_demodded, f_s_m);

%% SSB AM modulation
sig_ssb_modded = ssb_mod(f_c_ssb, A_c_ssb, sig_m, f_s_m, f_s_c_ssb, ...
    is_ussb);
sig_ssb_demodded = ssb_demod(sig_ssb_modded, f_c_ssb, A_c_ssb, ...
    f_s_m, f_s_c_ssb, W);
plot_modded_demodded(sig_ssb_modded, sig_ssb_demodded, f_s_m, ...
    f_s_c_ssb, 'ssb');
audiowrite('ssb_demodded.wav', sig_ssb_demodded, f_s_m);

%% PM Modulation
sig_pm_modded = pm_mod(f_c_pm, A_c_pm, sig_m, k_p, f_s_m, f_s_c_pm);
sig_pm_demodded = pm_demod(f_c_pm, A_c_pm, sig_pm_modded, k_p, ...
    f_s_m, f_s_c_pm, W);
plot_modded_demodded(sig_pm_modded, sig_pm_demodded, f_s_m, ...
    f_s_c_pm, 'pm');
audiowrite('sig_pm_demodded.wav', sig_pm_demodded, f_s_m);

%% FM Modulation
sig_fm_modded = fm_mod(f_c_fm, A_c_fm, sig_m, k_f, f_s_m, f_s_c_fm);
sig_fm_demodded = fm_demod(sig_fm_modded, A_c_fm, f_s_m, f_s_c_fm, ...
    k_f, W);
plot_modded_demodded(sig_fm_modded, sig_fm_demodded, f_s_m, ...
    f_s_c_fm, 'fm_problematic');
audiowrite('sig_fm_demodded.wav', sig_fm_demodded, f_s_m);

%% Conventional AM with noise
demod_fn = @(sig_mod) conv_demod(sig_mod, A_c_conv, a_conv, f_s_m, ...
    f_s_c_conv, W);
[sigs_conv_modded, sigs_conv_demodded, snrs] = apply_noise(noise_stds, ...
    demod_fn, sig_conv_modded, sig_conv_demodded, f_s_m, ...
    f_s_c_conv, W, 'conv');
var_noise_emp(1,:) = snrs;

%% calculate conventional AM theoretical SNR
var_noise_theo(1,:) = 10 * log10(a_conv^2 * A_c_conv^2 * ...
    bandpower(sig_m) * (2 * (2 * noise_stds.^2) * W / f_s_c_conv).^-1);
```

```
%% SSB AM with noise
demod_fn = @(sig_mod) ssb_demod(sig_mod, f_c_ssb, A_c_ssb, f_s_m, ...
    f_s_c_ssb, W);
[sigs_ssb_modded, sigs_ssb_demodded, snrs] = apply_noise(noise_stds, ...
    demod_fn, sig_ssb_modded, sig_ssb_demodded, f_s_m, f_s_c_ssb, ...
    W, 'ssb');
var_noise_emp(2,:) = snrs;

%% calculate SSB AM theoretical SNR
var_noise_theo(2,:) = 10 * log10(A_c_ssb^2 * bandpower(sig_m) * ...
    ((2 * noise_stds.^2) * W / f_s_c_ssb).^-1);

%% PM with noise
demod_fn = @(sig_mod) pm_demod(f_c_pm, A_c_pm, sig_mod, k_p, f_s_m, ...
    f_s_c_pm, W);
[sigs_pm_modded, sigs_pm_demodded, snrs] = apply_noise(noise_stds, ...
    demod_fn, sig_pm_modded, sig_pm_demodded, f_s_m, f_s_c_pm, W, 'pm');
var_noise_emp(3,:) = snrs;

%% calculate PM theoretical SNR
var_noise_theo(3,:) = 10 * log10(k_p^2 * A_c_pm^2 * bandpower(sig_m) * ...
    (2 * (2 * noise_stds.^2) * W / f_s_c_pm).^-1);

%% FM with noise
demod_fn = @(sig_mod) fm_demod(sig_mod, A_c_fm, f_s_m, f_s_c_fm, ...
    k_f, W);
[sigs_fm_modded, sigs_fm_demodded, snrs] = apply_noise(noise_stds, ...
    demod_fn, sig_fm_modded, sig_fm_demodded, f_s_m, f_s_c_fm, W, 'fm');
var_noise_emp(4,:) = snrs;

%% calculate FM theoretical SNR
beta_f = k_f / W;
var_noise_theo(4,:) = 10 * log10(3 * beta_f^2 * A_c_fm^2 * ...
    bandpower(sig_m) * (2 * (2 * noise_stds.^2) * W / f_s_c_fm).^-1);

%% Changing modulation index of Conventional AM
mod_fn = @(sig_m, a) conv_mod(f_c_conv, A_c_conv, a, sig_m, f_s_m, ...
    f_s_c_conv);
demod_fn = @(sig_mod, a) conv_demod(sig_mod, A_c_conv, a, f_s_m, ...
    f_s_c_conv, W);
var_mod_ind_emp(1,:) = apply_modulation_factor( ...
    a_conv, noise_std_conv, mod_fn, demod_fn, sig_m, ...
    sigs_conv_modded(4,:), sigs_conv_demodded(4,:), f_s_m, ...
    f_s_c_conv, W, sig_conv_demodded, 'a', 'conv');

%% calculate conventional AM theoretical SNR
var_mod_ind_theo(1,:) = 10 * log10([a_conv/2 a_conv a_conv*2].^2 * ...
    A_c_conv^2 * bandpower(sig_m) * ( 2 * (2 * noise_std_conv) * W / ...
    f_s_c_conv)^-1);

%% Changing the modulation index of PM
mod_fn = @(sig_m, k_p) pm_mod(f_c_pm, A_c_pm, sig_m, k_p, f_s_m, f_s_c_pm);
demod_fn = @(sig_mod, k_p) pm_demod(f_c_pm, A_c_pm, sig_mod, k_p, ...
    f_s_m, f_s_c_pm, W);
var_mod_ind_emp(2,:) = apply_modulation_factor( ...
    k_p, noise_std_pm, mod_fn, demod_fn, sig_m, sigs_pm_modded(4,:), ...
    sigs_pm_demodded(4,:), f_s_m, f_s_c_pm, W, sig_pm_demodded, ...
    'k_p', 'pm');

%% calculate PM theoretical SNR
var_mod_ind_theo(2,:) = 10 * log10([k_p/2 k_p k_p*2].^2 * A_c_pm^2 * ...
    bandpower(sig_m) * (2 * (2 * noise_std_pm^2) * W / f_s_c_pm).^-1);

%% Changing the modulation index of FM
mod_fn = @(sig_m, k_f) fm_mod(f_c_fm, A_c_fm, sig_m, k_f, ...
    f_s_m, f_s_c_fm);
demod_fn = @(sig_mod, k_f) fm_demod(sig_mod, A_c_fm, f_s_m, ...
    f_s_c_fm, k_f, W);
var_mod_ind_emp(3,:) = apply_modulation_factor( ...
    k_f, noise_std_fm, mod_fn, demod_fn, sig_m, sigs_fm_modded(3,:), ...
    sigs_fm_demodded(3,:), f_s_m, f_s_c_fm, W, sig_fm_demodded, ...
    'k_f', 'fm');

%% calculate FM theoretical SNR
beta_f = [k_f/2 k_f k_f*2] / W;
var_mod_ind_theo(3,:) = 10 * log10(3 * beta_f.^2 * A_c_fm^2 * ...
    bandpower(sig_m) * (2 * (2 * noise_std_fm.^2) * W / f_s_c_fm).^-1);

%% Print out results
fprintf('Theoretical SNRs from varying noise variance');
var_noise_theo(:,2:4)
fprintf('Empirical SNRs from varying noise variance');
var_noise_emp(:,2:4)
fprintf('Theoretical SNRs from varying modulation indices');
var_mod_ind_theo
fprintf('Empirical SNRs from varying modulation indices');
var_mod_ind_emp

% plot modulation modded and demodded signals and their FTs
% params:
% sig_modded    = modulated signal
% sig_demodded  = demodulated signal
% f_s_m         = baseband sampling frequency
% f_s_c         = modulated signal sampling frequency
% figname       = name of figure
function plot_modded_demodded(sig_modded, sig_demodded, f_s_m, f_s_c, ...
    figname)

    figure('visible', 'off');
    tiledlayout(2, 2, 'TileSpacing', 'Compact', 'Padding', 'Compact');

    duration = length(sig_modded) / f_s_c;
    t_c = linspace(0, duration, length(sig_modded));
    t_m = linspace(0, duration, length(sig_demodded));

    % plot modded signal in time domain
    nexttile();
    plot(t_c, sig_modded);
    title('Modulated signal');
    ylabel('$u(t)$');
    xlabel('$t$');
    xlim([0 duration]);

    % plot modded signal in frequency domain
    nexttile();
    [f, ft] = plot_ft(sig_modded, f_s_c);
    semilogy(f, ft);
    title('Modulated signal Fourier transform magnitude');
    ylabel('$|U(f)|$');
    xlabel('$f$');

    % plot demodded signal in time domain
    nexttile();
    plot(t_m, sig_demodded);
    title('Demodulated signal');
    ylabel('$r(t)$');
    xlabel('$t$');
    ylim([-1.2 1.2]);
    xlim([0 duration]);

    % plot demodded signal in frequency domain
    nexttile();
    [f, ft] = plot_ft(sig_demodded, f_s_m);
    semilogy(f, ft);
    title('Demodulated signal Fourier transform magnitude');
    ylabel('$|R(f)|$');
    xlabel('$f$');
    ylim([10e-8, 10e-2]);

    fig_save(figname, [40 20]);
end

% apply noise to signals modulated with different modulation indices
% params:
% mod_index             = modulation index values
% noise_std             = standard deviation of noise to add
% mod_fn                = lambda to modulate a signal
% demod_fn              = lambda to demod a noisy modulated signal
% sig_m                 = message signal
% sig_modded            = reference modded signal
% sig_demodded          = reference demodded signal
% f_s_m                 = baseband sampling frequency
% f_s_c                 = modulated signal sampling frequency
% tau                   = post-demodulation LPF cutoff frequency
% sig_demodded_no_noise = reference no-noise demodulated signal
% variant               = name of modulation index
% mod_type              = modulation scheme
% returns:
% snrs                  = SNRs of noisy modulated signals
function snrs = apply_modulation_factor( ...
    mod_index, noise_std, mod_fn, demod_fn, sig_m, sig_modded, ...
    sig_demodded, f_s_m, f_s_c, tau, sig_demodded_no_noise, variant, ...
    mod_type)

    sigs_modded = zeros(3, length(sig_modded));
    sigs_demodded = zeros(3, length(sig_demodded));

    % half the modulation index
    sigs_modded(1,:) = mod_fn(sig_m, mod_index/2) + ...
        noise_std * randn(1, length(sig_modded));
    sigs_demodded(1,:) = lowpass_audible(demod_fn(sigs_modded(1,:), ...
        mod_index/2), f_s_m, tau);

    % original modulation index
    sigs_modded(2,:) = sig_modded;
    sigs_demodded(2,:) = sig_demodded;

    % double the modulation index
    sigs_modded(3,:) = mod_fn(sig_m, mod_index*2) + ...
        noise_std * randn(1, length(sig_modded));
    sigs_demodded(3,:) = lowpass_audible(demod_fn(sigs_modded(3,:), ...
        mod_index*2), f_s_m, tau);

    % plot
```

```
    snrs = plot_with_noise([mod_index/2, mod_index, mod_index*2], ...
        sigs_modded, sigs_demodded, f_s_m, f_s_c, ...
        sig_demodded_no_noise, variant, mod_type);
end

% apply different noise to a modded signal
% params:
% noise_stds             = list of standard deviations of the noise
% demod_fn               = lambda to demod a noisy signal
% sig_modded_no_noise    = no-noise modded signal reference
% sig_demodded_no_noise  = no-noise demodded signal reference
% f_s_m                  = baseband sampling frequency
% f_s_c                  = modulated signal sampling frequency
% tau                    = cutoff frequency
% mod_type               = modulation scheme
% returns:
% sigs_modded            = modulated signals with applied noise
% sigs_demodded          = demodulated noisy modded signals
% snrs                   = snrs for demodded signals
function [sigs_modded, sigs_demodded, snrs] = apply_noise(noise_stds, ...
    demod_fn, sig_modded_no_noise, sig_demodded_no_noise, f_s_m, f_s_c, ...
    tau, mod_type)

    sigs_modded = zeros(length(noise_stds), ...
        length(sig_modded_no_noise));
    sigs_demodded = zeros(length(noise_stds), ...
        length(sig_demodded_no_noise));

    % first element will be the no-noise variant
    sigs_modded(1,:) = sig_modded_no_noise;
    sigs_demodded(1,:) = sig_demodded_no_noise;

    % loop through each noise process
    for i = 2:length(noise_stds)
        % modulate, add noise
        sigs_modded(i,:) = sig_modded_no_noise + ...
            noise_stds(i) * randn(1, length(sig_modded_no_noise));

        % demod
        sigs_demodded(i,:) = lowpass_audible(...
            demod_fn(sigs_modded(i,:)), f_s_m, tau);
    end

    % plot
    snrs = plot_with_noise(noise_stds, sigs_modded, sigs_demodded, ...
        f_s_m, f_s_c, sig_demodded_no_noise, '\sigma_n', mod_type);
end

% approximates SNR in dB, given the pristine demodded signal and a noisy
% variant; approximates noise as the difference between the two
% params:
% sig_no_noise  = demodded signal with no noise
% sig_noisy     = noisy signal to calculate the SNR of
% returns:
% res           = SNR (in dB)
function res = snr(sig_no_noise, sig_noisy)
    res = 10 * log10(bandpower(sig_no_noise) / ...
                     bandpower(sig_no_noise - sig_noisy));
end

% plots the demodded signal, spectrum of modded signal, and spectrum of
% demodded signal for a series of related signals with one varying
% variable; also calculates SNRs and returns them
% variant
% params:
% variants      = list of varying parameter values
% sigs_modded   = modded signals
% sigs_demodded = demodded signals
% f_s_m         = sample rate of baseband signal
% f_s_c         = sample rate of modulated signal
% sig_snr_ref   = reference for SNR calculation (i.e., no-noise demodded)
% variant       = name of varying parameter
% mod_type      = modulation type (e.g., CONV, FM)
% returns:
% snrs          = calculated SNRs of each demodded signal wrt. sig_snr_ref
function [snrs] = plot_with_noise(variants, sigs_modded, sigs_demodded, ...
    f_s_m, f_s_c, sig_snr_ref, variant, mod_type)

    N = length(variants);
    snrs = zeros(N, 1);

    figure('visible', 'off');
    tiledlayout(N, 3, 'TileSpacing', 'Compact', 'Padding', 'Compact');

    for i = 1:N

        % plot demodded signal
        nexttile();
        plot(sigs_demodded(i,:));
        ylim([-1.25, 1.25]);
        snr_db = snr(sig_snr_ref, sigs_demodded(i,:));
        snrs(i) = snr_db;
        title(sprintf('Demodded signal; $%s=%0.2f$, SNR=%0.2fdB', ...
            variant, variants(i), snr_db));
        ylabel('$m(t)$');
        xlabel('$t$');

        % plot spectrum of modded signal
        nexttile();
        [f, ft] = plot_ft(sigs_modded(i,:), f_s_c);
        semilogy(f, ft);
        title(sprintf('Spectrum of modded signal; $%s=%0.2f$', ...
            variant, variants(i)));
        ylabel('$|M_{mod}(f)|$');
        xlabel('$f$');

        % plot spectrum of demodded signal
        nexttile();
        [f, ft] = plot_ft(sigs_demodded(i,:), f_s_m);
        semilogy(f, ft);
        ylim([10e-10, 10e0]);
        title(sprintf('Spectrum of demodded signal; $%s=%0.2f$', ...
            variant, variants(i)));
        ylabel('$|M(f)|$');
        xlabel('$f$');
    end

    if strcmp(variant, '\sigma_n')
        variant = 'noise_std';
    end
    fig_save(sprintf('%s_var_%s', mod_type, variant), [60, 10 * N]);
end
```