Lab 3: Digital Modulation and Demodulation

Report Due: 21:00, 5/10, 2019

1 Overview

In digital modulation, an analog carrier signal is modulated by a discrete signal. Digital modulation methods can be considered as digital-to-analog conversion and the corresponding demodulation as analog-to-digital conversion.

2 Experiments

In this section, we want to implement each processing block of modulation and demodulation. After completing the questions of this section, you can simulate the transmission system by yourself.

1. **Symbol Mapping:** First of all, we will find some constellation sets and map bits to the symbols of sets.

   Write a Matlab function to modulate an incoming binary sequence using different constellations.

   ```
   symbol_sequence = symbol_mapper(binary_sequence, M, d, ...
                                   constellation, mapping)
   ```

   - Inputs:
     - `M`: The number of points in the signal constellation and `M = 2,4,8,16` for `'PAM'` and `'PSK'`, `M = 4,16` for `'QAM'`.
     - `d`: The minimum distance among the constellation.
     - `constellation`: `'PAM'`, `'PSK'` or `'QAM'`.
     - `mapping`: `'Binary'` or `'Gray'`.

   For Gray code mapping of PAM, please assign `00` $\cdots$ `0` to the leftest constellation point. Gray code mapping of PSK and QAM is shown in Figure 1 and Figure 2. If input `M` is not the power of 2 or if input `constellation` is not defined, your function should be able to throw error and display message. Please consider 4-PSK and 4-QAM as QPSK.

   Plot all the constellations, and show the bits of each symbol on the constellation points.

2. **Pulse Shaping:** Please see the page 25-26 of Comm_Lab_Week_06_ver_20190327.pdf. We will design $p(t)$ and $q(t)$ such that $u_m = \hat{u}_m$ for all $m \in \mathbb{Z}$. Note that pulse function $p(t)$ satisfies $g(t) = p(t) * q(t)$. One possible choice of $q(t)$ is to use matched filtering, i.e.

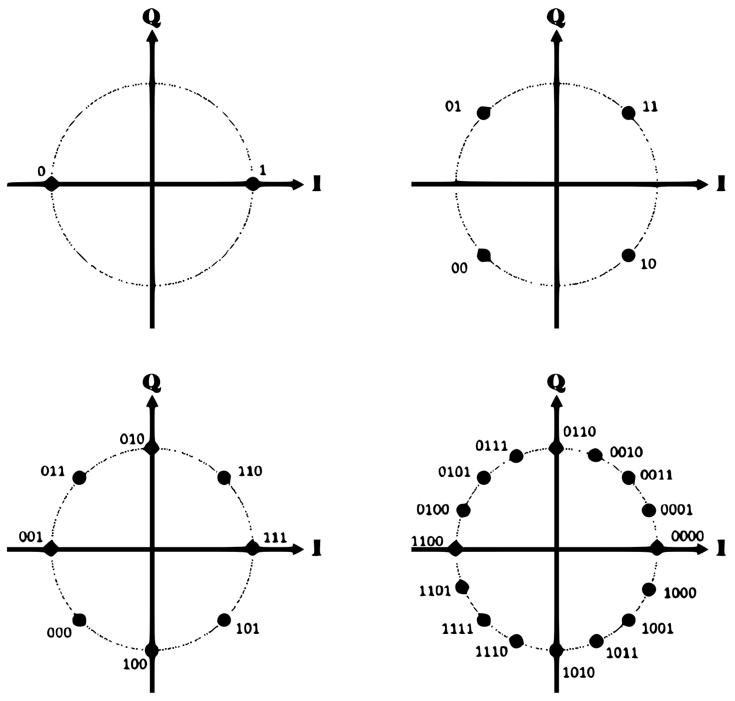

Figure 1: Gray Code Mapping of PSK

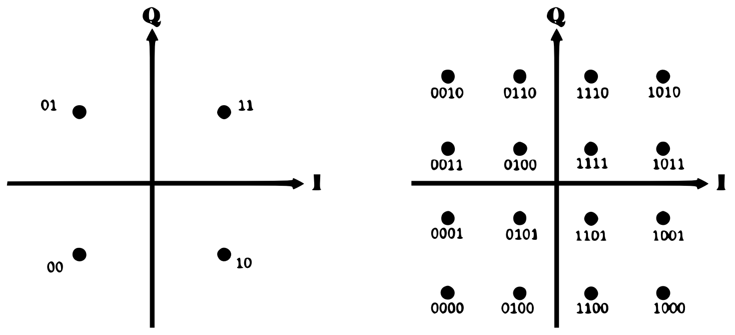

Figure 2: Gray Code Mapping of QAM

$q(t) = p^*(-t)$. Pulse shaping filter must be chosen carefully not to introduce intersymbol interference.

In Matlab, we simulate the continuous time by oversampling. Here, parameters `W`, `oversampling_factor`, `pulse_duration`, and carrier frequency are just for the convenience of simulation. They are not the data applied in real world.

(a) Set the operational bandwidth `W = 50` Hz. Given the following arguments (you can adjust by yourself),

```
oversampling_factor = 1000;
T_os = 1/oversampling_factor; % time spacing after oversampling
pulse_duration = 1; % 1 sec
t_axis = (-pulse_duration/2 : T_os : pulse_duration/2 - T_os);
```

choose **rectangular function** as $g(t)$. Plot the function and its Fourier transform. Validate the Nyquist criterion in the time-domain and the frequency-domain.

(b) Following (a), change $g(t)$ to **sinc function**. Use the definition of the sinc function in lecture slide,

$$g(t) = \frac{sin(\pi t/T)}{\pi t/T} = sinc(\frac{t}{T})$$

Plot the function and its Fourier transform. Validate the Nyquist criterion in the time-domain and the frequency-domain.

(c) Following (a), change $g(t)$ to **raised cosine function**. Plot the function and its Fourier transform. Validate the Nyquist criterion in the time-domain and the frequency-domain.

(d) Observe the power decay in time-domain and bandwidth in frequency-domain of different pulses. Describe which pulse you will choose for pulse shaping.

(e) Write a Matlab function to implement the pulse shaping filter $p(t)$. **Note that the pulse function** $p(t)$ **could be different from the function** $g(t)$. Your own pulse shaping function should have the following arguments,

```
y = pulse_shaper(x, pulse_shape, W)
```

- Input:
  - `x`: Input symbols after modulation (QPSK, 16-QAM, etc.).
  - `pulse_shape`: `'sinc'` or `'raised cosine'` or others.
  - `W`: The operational bandwidth.
- Output:
  - `y`: The signal after pulse-shaping

Because there are `T/T_os` points between pulse $p(m\text{T})$ and $p((m+1)\text{T})$, we have to oversample the symbol sequence by `T/T_os` and then we can convolve the oversampled symbol sequence with pulse $p(t)$.

(f) Generate a 20-bits random binary sequence, and let the [0 1] with equal probability. Modulate the binary sequence using Gray-coded QPSK with `d = 2`, and then pass the symbols through the pulse shaper.

Plot real part and imaginary part of the output signal and indicate what pulse do you use.

3. **Baseband $\leftrightarrow$ Passband:** Following 2(f), here we will implement upconverter and downconverter. Assume carrier frequency is 100 Hz.

(a) You have a baseband signal $x_b(t) = x_b^{(I)}(t) + jx_b^{(Q)}(t)$ complex output signal of pulse shaper. Multiply $x_b^{(I)}(t)$ with in-phase carrier and multiply $x_b^{(Q)}(t)$ with quadrature-phase carrier.

$$x(t) = x_b^{(I)}(t)\sqrt{2}cos(2\pi f_c t) - x_b^{(Q)}(t)\sqrt{2}sin(2\pi f_c t)$$

Plot the output signal $x(t)$.

(b) In real world, we have noisy channels. Assume we pass the transmitted signal through AWGN channel with SNR = 25 dB. Plot the noisy signal.

(c) At receiver, we have to recover the baseband signal according to the received passband noisy signal. Multiply the received signal with in-phase carrier, and multiply the received signal with quadrature-phase carrier. Next, the I/Q channels can be obtained from lowpass filtering. You can implement lowpass by convolve signal with a sinc function.

Plot real part and imaginary part of the output signal after lowpass filtering.

4. **MAP/ML/MD:** In this part we will have to demodulate the received noisy signal. With different prior probability of [0 1] and detection rule, we may get different demodulated symbols.

   Here, to simplifying the analysis of communication in the presence of noise, we only need to focus on the constellation domain. The transmitted symbol sequence does not go throuth **pulse shaping** and **Baseband** $\leftrightarrow$ **Passband** process. We directly add circularly symmetric complex Gaussian noise to the transmitted symbol sequence.

   (a) Plot the constellation of noisy symbol sequence with SNR = $\{0,10,20\}$.

   (b) Write a Matlab function to demap the symbol sequence to binary sequence. Your demapper function should have the following arguments,

```
binary_sequence = symbol_demapper(symbol_sequence, M, d, ...
                                  constellation, mapping, decision_rule)
```

   where `decision_rule` is `'MD'`.

   (Bonus) `decision_rule`: `'MAP'` and `'ML'`.

   (c) First, you should generate a random binary sequence. Next, go through the process, binary seqence $\rightarrow$ symbol mapping $\rightarrow$ add circularly symmetric complex Gaussian noise $\rightarrow$ symbol demapping $\rightarrow$ detection (MD rule)

   For each case, plot constellations, symbol error rate, and bit error rate with SNR = $0 \sim 25$ dB.

   - case 1: QPSK + binary code vs. QPSK + Gray code
   - case 2: 2-PAM + Gray code vs. 4-PAM Gray code
   - case 3: QPSK + Gray code vs. 8-PSK + Gray code
   - case 4: 4-QAM + Gray code vs. 16-QAM + Gray code
   - case 5: 16-PAM + Gray code vs. 16-PSK + Gray code + 16-QAM + Gray code

5. **Modulation and Demodulation:** In Lab 2, we have completed the source coding of `'handel.ogg'`. In this part, please pass the output of source coding in Lab 2 to the process mentioned in 4(c). Describe what it sounds like.

3 Lab Report

There is no format requirements for your lab report. In the report, you should address the results of the exercises mentioned above. You should also include your simulation program in the appendix of the report. Include whatever discussions about the new findings during the lab exercise, or the problems encountered and how are those solved. Do not limit yourself to the exercises specified here. You are highly encouraged to play around with your simulation program on self-initiated extra lab exercises/discussions. For example, you can record your own voice or search for sound clips with no copyright at Freesound.org (`https://freesound.org/`).