In the Name of God

Communication Systems (25751-4)

Python Assignment 03

Fall Semester 1402-03

Department of Electrical Engineering

Sharif University of Technology

Instructor: Dr. M. Pakravan

Due on Bahman 13, 1402 at 23:59

1 Slotted-ALOHA Capacity

The idealized slotted aloha model divides time into several slots, each equal to the packet transmission time. All nodes are perfectly synchronized and transmit at the start of each slot. The transmission probability for each user in each slot is a value p. For any time slot, three scenarios are possible:

1. More than one node transmits, resulting in a collision slot. The receiver cannot decode them correctly.
2. Only one node transmits and the receiver can decode it correctly.
3. No node transmits in the current slot.

1.1 Slotted ALOHA Simulation

Write a function that takes three parameters: the number of users, the number of time slots, and the probability p that a user will transmit in a slot. The function should return the fraction of successful slots (slots where only one user transmitted) over the total number of slots.

For this purpose, fill in the function below:

```
def simulate_aloha(n_users, n_timeslots, prob):
    pass
```

1.2 Verifying Slotted ALOHA Capacity

Run your simulation over 1000 values of p between 0 and 1 with $n = 10$ users and $T = 10000$ timeslots. Plot the timeslot success ratio as a function of p. Verify that the resulting plot matches the theoretical curve.

2 MAC protocol: Preamble

A preamble is a waveform sent at the beginning of a packet that is used to indicate the start of a packet. The preamble waveform is agreed to ahead-of-time, so the signal carries no data. Receivers will listen for the preamble, and when it's detected the receivers will start to demodulate the rest of the packet. Preambles can also be used to synchronize multiple clients for MAC protocols that require synchronization. In this part, we will explore how to detect and

synchronize to a preamble. Both detection and synchronization can be done using a **correlation**. A correlation of two discrete signals x and y is an infinite-length discrete signal which is defined as

$$(x \star y)[k] = \sum_{n=-\infty}^{\infty} x^*[n] \cdot y[n+k]$$

In other words, at each index k, you take conjugate of the first signal x and offset the second signal y by k indices and then sum the resulting two signals. Intuitively, the correlation measures the similarity of two signals for every offset of the one signal relative to the other. For signals with finite length, the correlation is zero for sufficiently large $|k|$, so often we'll consider the correlation to be finite as well.

So how do we use the correlation to detect a preamble? Let x be the preamble signal, and let y be the signal that the receiver receives. For now, we'll let y be finite length. If the preamble is not received, then we can model y as a random signal, and the correlation between x and y will also be random. However, if the receiver did receive the preamble in the signal y, then we can model y as x with some offset plus noise. The correlation between x and y will then have a large amplitude at the time in y when the preamble started. So if we look for large peaks in the correlation we can detect and then also synchronize to a given preamble.

You are required to complete the Python notebook (`CHW03_Q2.ipynb`) according to the provided instructions and generate the expected outputs.

3 GPS signal

PRN stands for PseudoRandom Noise. It is a binary sequence of length 16 bits that is included in the data stream of a GPS satellite signal. The purpose of this sequence is to ensure that the data signal is spread across the bandwidth of the satellite signal, preventing interference between different signals.

3.1

Generate a sequence of codes with size 16 (PRN). (Use `random.seed` to get the same sequence of pseudorandom numbers each time you run the program.)

3.2

The transmited GPS signal uses binary phase shift keying, signal changed by shift π while it turns from 1 to -1, which can be done by multiplying the carefully constructed PRN above with the carrier! This works because $\sin(x - \pi) = -\sin(x)$, in other words it's exactly a 180 degree phase shift. We should see this as blips in sine wave every time our data changes from 1 to 0 or visa-versa. An example is shown in figure 1.

Generate a Binary Phase Shift Keying (BPSK) modulation scheme and plot the PRN sequence and the signal over time. An example is shown in figure 2.

3.3

Check the concideness between codes and graph. An example is shown in figure 3

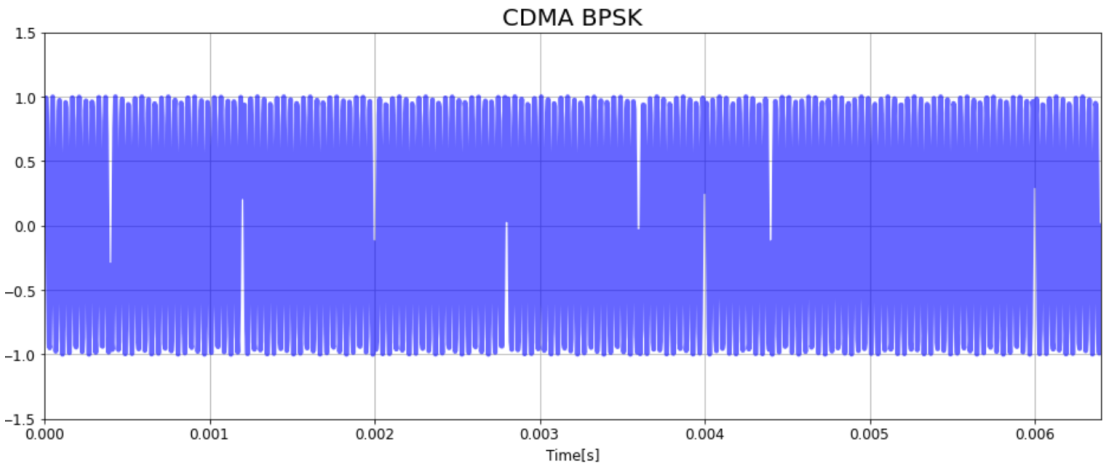

Figure 1: BPSK signal

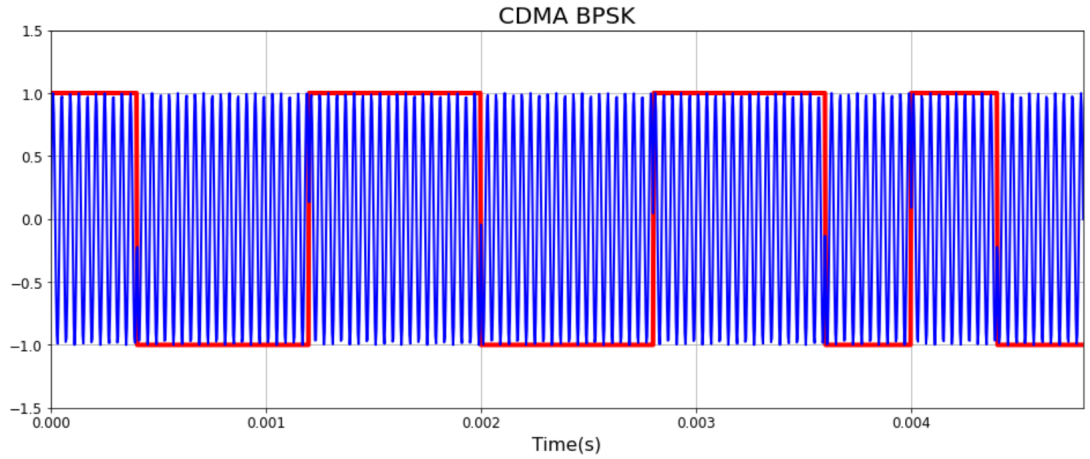

Figure 2: PRN sequence and the BPSK signal

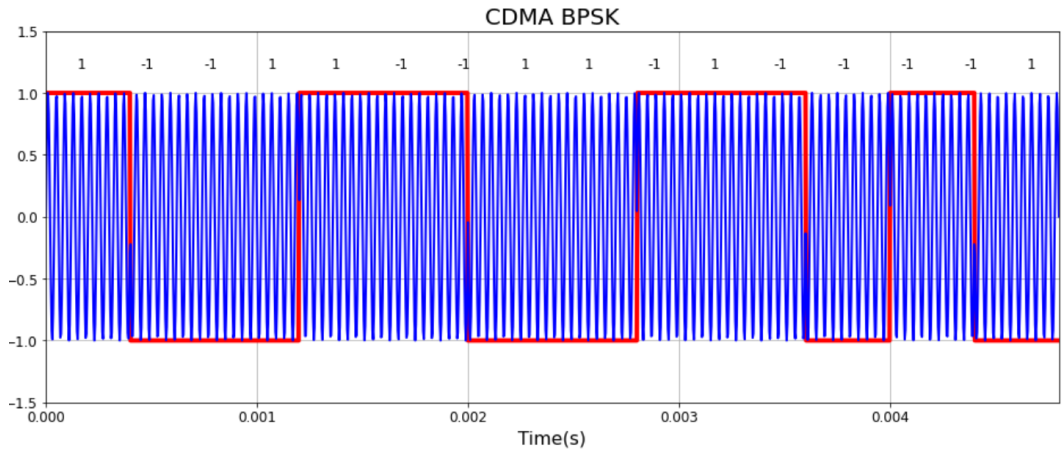

Figure 3: concideness between codes and graph