Decoding CDMA $\int$ecret Messages

[*]Or Why Grid searching should always be your first option

Jack Spencer "Danger" Langner
Dept. of Electrical Engineering
Cooper Union
Manhattan, New York
jlangner732@gmail.com

***Abstract*—Each student in the spring 2020 ECE408: Wireless Communications class was given a unique message to decode. The "secret" message was encoded in a way that vaguely resembles CDMA-2000. This document recounts the decoding process along with the secret message and other details that may be of interest.**

***Index Terms*—M sequence, Walsh – Hadamard coding**

I. Introduction

THROUGHOUT the Spring 2020 semester in ECE-408: Wireless Communications, the class has covered an array of standards dealing with wireless communications, this has come as a shock to many especially the instructor of the course. Battling through these emotions, we were able to cover Code Division Multiple Access (CDMA) which has been included in many standards. CDMA allows multiple users to communicate on the same carrier frequency by encoding an individual's data on different channels that theoretically do not interfere with each other. Using this principle, the instructor encoded a "secret" message to each student. The task for the student was then to recover the message that was encoded according to a CDMA scheme.

The paper is organized in the following manner. In section II the CDMA system that was used to encode the message is discussed. Then in section III the strategy used to decode the message is described. Finally, in section IV the "secret" message along with other details are revealed along with any significance the message may have.

II. The CDMA System

The system that was used to encode the "secret" message is depicted in fig 1 and I will describe the operation. We begin with the binary data streams, one of which is all 0's. This all 0 stream represents the pilot which is our best friend because the pilot never changes and thus allows us to compensate for idealities. The other bit stream is the data we care which is the secret message. Both bit streams are then Binary Phase Shift Keying (BPSK) modulated according to the following mapping $\{0, 1\} \rightarrow \{+1, -1\}$, which is represented by the BPSK block in Fig. 1. Next, the triangles represent the gain of the individual channels, which is just one in this case, no need to get too fancy and introduce a source of nonlinearity so early in the transmit chain. The 8-ary Walsh Channel block is where the magic of CDMA happens. Walsh channels are built on the rows of a Hadamard matrix, or Walsh-Hadamard matrix. Walsh-Hadamard matrices are nice because each of the rows are orthognal to one another, so a linear combination of the columns can be recovered by multiplying by the same Hadamard matrix. In this case, the 8×8 Hadamard matrix is used to create the Walsh channels and is shown below:

$$W_8 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \\ w_4 \\ w_5 \\ w_6 \\ w_7 \end{bmatrix}$$

In order to create the distinct channels, the BPSK data streams are assigned to a specific channel which is denoted by w_i where $i \in \{0, 7\}$ and each w_i denotes a different row of W_8. Then each BPSK symbol is used to scale the corresponding row of W_8. Furthermore, since one BPSK symbol is mapped to 8 symbols because of the transform, this group of 8 symbols is then called a chip, so one BPSK data symbol gets mapped to one chip.

An example using channels 0 and 5. We know channel 0 carries the pilot, which will be a stream of only 1's under BPSK. So the output is $[1 \cdot w_0, 1 \cdot w_0] = [w_0, w_0]$. Channel 5 carries the actual data which we will assume takes the form of $[0, 1] \rightarrow [+1, -1]$ which is then used to multiply w_5. The result is $[+1 \cdot w_5, -1 \cdot w_5] = [w_5, -w_5]$. Combining the two channels gives $[w_0 + w_5, w_0 - w_5]$ which is what will be seen at the receiver. At the receiver, the two chips are then multiplied on the right by W_8, which gives the following:

$$\begin{bmatrix} w_0 + w_5 \\ w_0 - w_5 \end{bmatrix} W_8 = \begin{bmatrix} 8 & 0 & 0 & 0 & 0 & 8 & 0 & 0 \\ 8 & 0 & 0 & 0 & 0 & -8 & 0 & 0 \end{bmatrix}$$

The 0's mean no information is present on that channel, and the sign of the 8's represents the BPSK symbol that was received.

That was a digression I wasn't expecting, continuing on we then have the Switch block which handles how data is formatted into frames. Each frame has length 255 and we already know each chip has length 8. Furthermore, since the data bit stream is derived from ascii characters we know that

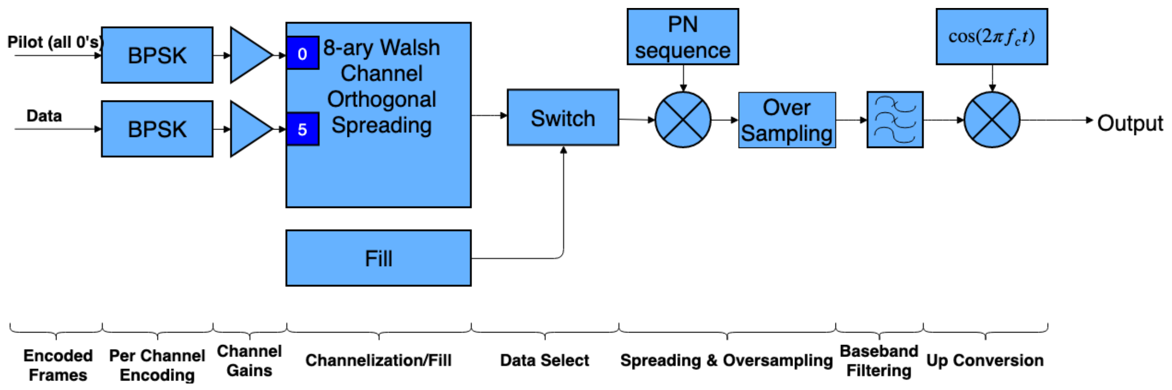

Fig. 1. CDMA Encoding System

a single character occupies 64 chips. Additionally, a character has to be sent within one frame, so this means that at most 3 characters can be sent per frame. Thus 192 chips out of the total 255 correspond to data, the remaining 63 chips are simply filled with 0's on the data stream. So the Switch determines whether data of fill is included in the frame. The frame structure is illustrated below in Fig. 2

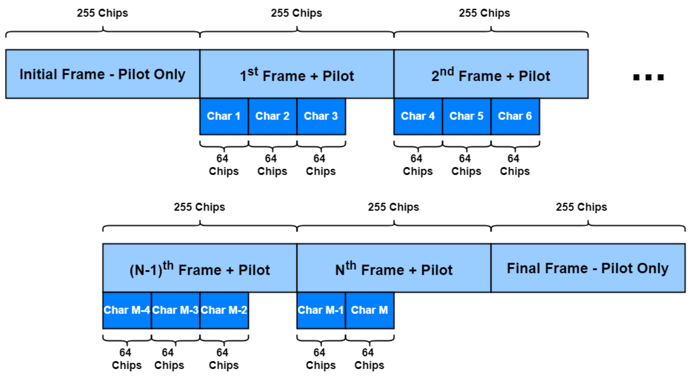

Fig. 2. CDMA Framing Scheme

After framing, the 255 chips are multiplied by a Pseudo-Noise (PN) sequence. A PN sequence, or a maximal length sequence (M sequence) is a binary sequence that has many magical properties. PN sequence are generated from a Linear Feedback Shift Register (LFSR) with a given number of memory elements (n) as described in [1] . With the correct set up a sequence of length $2^n - 1$ can be created and if the LFSR runs past $2^n - 1$, then the sequence repeats. Furthermore the sequence that is generated will have the property that is has perfect correlation with itself and essentially zero correlation with any offset of it self. So each frame is multiplied, or scrambled, by the same PN sequence. The PN sequence was constructed according to the article we were told to follow. A visualization of the LFSR structure used is given in Fig. 3 Each block in Fig. 3 represents a memory element and the number inside the block represents its position. Furthermore, this realization is how the PN sequence was generated to scramble the frames. Following the scrambling, the frames are up sampled by a factor of 4 and passed through a pulse shaping

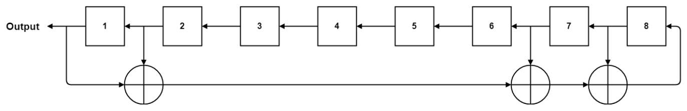

Fig. 3. PN Sequence LFSR

filter. Here the pulse shaping filter is a finite impulse response square root raised cosine filter which was obtained from the MATLAB function rcosine(1,4,'fir/sqrt',0.75), but this is deprecated so a better choice is rcosdesign(0.75,6,4,'sqrt'). Fig.

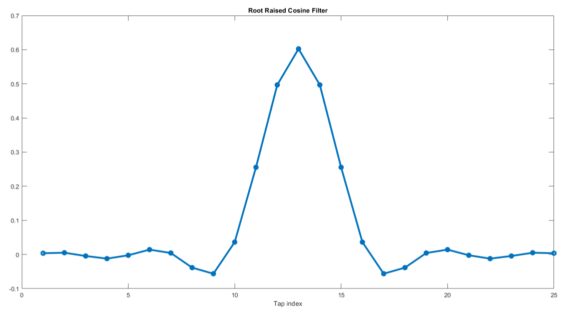

Fig. 4. Impulse Response of RRC Filter Used

4 depicts the response of the filter used in the system. Finally, the signal is mixed with a sinusoidal wave with an unknown frequency and phase offset. This was done specifically to make our lives harder. At this point every part of the CDMA encoder has been covered.

III. Decoding the Message

We know how the signal that we received was created and the task was to undo the transmit process in order to recover the message. So, we begin by working backwards with the information we have about the system. Logically the first thing to do is remove the phase and frequency offsets, however these are unknown so we cannot start here. Trying again, we start with the filter and we actually know the structure of the filter so this is promising. Therefore, we begin by passing the entire received message through the root raised cosine filter which will allow us to take advantage of the pulse shaping which leads to a zero crossing property that satisfies Nyquist's criteria. Note, that the same root raised cosine filter is used at both the transmitter and receiver.

Next, we could theoretically down-sample the filtered data by a factor of 4, but we were advised not to do this, so instead I found the PN sequence that was used to scramble the data. In order to this, a PN sequence with all subsequent offsets had to be generated in order to make sure the correct sequence was found. So, the LFSR of Fig. 3 was implemented with all possible initial states. In order to verify that the PN sequences were generated correctly, I took advantage of the correlation property. Specifically, if the PN sequence is BPSK modulated, then the auto-correlation has the property that for the same offset the result will be 255, otherwise the result will be -1. In Fig. 5 it is clear that the main diagonal

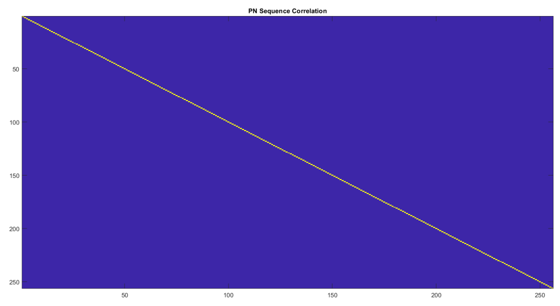

Fig. 5. PN Sequence verification

is yellow while every other element of the matrix is blue. In context, yellow represents a value of 255 and blue is -1. Furthermore, the main diagonal shows the result of the generated sequence being correlated with itself and the off diagonal elements are the correlation with every other offset, so it is clear that the PN sequence was generated correctly at the receiver. Then the PN sequences were correlated with the filtered received. This allows the PN sequence that was used at the transmitter to be determined because the received data contains one PN sequence, so then which ever PN sequence gives good correlation at the receiver will be the same as the one used at the transmitter. Furthermore, we were advised to filter the filtered received data with an up-sampled version of the PN sequences and then look at the sample at an offset 1044 in order to determine the PN sequence. By correlating in this way, the indices that determine the start of every frame that contains data can be determined and the first and last frames that only carry pilot information do not have to be carried through the rest of the decoding process. The result of correlating with the correct PN sequence is shown in Fig. 6 along with a line at y value of 240. The reason the frames do not correlate with a value of 255 is because there is additive white Gaussian noise in the channel which messes up the correlation a little. There is still a clear distinction and it is easy to identify the beginning of the data frames. (I know the threshold should be thicker, but then it would cover some of the peaks.) After determining the start of each frame, the data

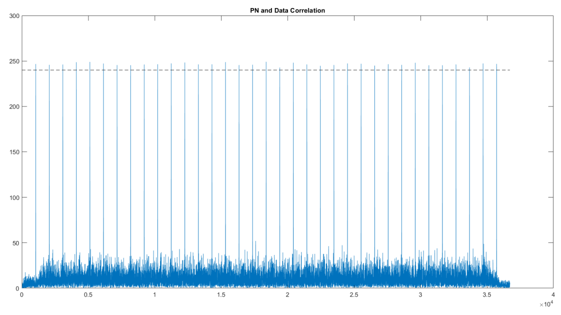

Fig. 6. Correct Correlation

was multiplied by the up-sampled PN sequence and the result was down-sampled. Then the data was grouped into frames of length 255, and within each frame the first 192 samples were considered since we know this is where the data lives. These 192 samples were then grouped into 24 row vectors of length 8, and by finding the average phase of these samples the phase offset could be corrected. After adjusting for the phase offset, the inner product of the row vectors and w_5 was taken to determine the data that was transmitted. After recovering the data on w_5, the data was converted back to binary and then into the correct ASCII character. This process of recovering the data on w_5 was repeated for each frame until the entire "secret" message was recovered.

A topic that was discussed, but not necessary was the Walsh codogram. The codogram allows one to determine which Walsh channels are carrying information. Below in Fig. 7 a codogram for a frame is presented. It is clear that Channel 0,

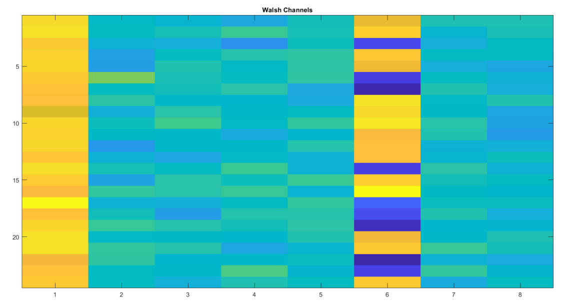

Fig. 7. Example Walsh Codogram

w_0, is carrying a constant signal while Channel 5,w_5, has some clear variation which corresponds approximately to alternating ± 8. Additionally, all other channels have no information as expected. it would be a surprise to have another secret message hidden there.

IV. Hidden Details

Now for the reason why we are all here, the super secret message that gave me so much strife I considered dropping out 3 times. I would like to say first that I was disappointed that I did not get a quote from *Austin Powers*, because that is where my middle name of "Danger" originates from, so I am just a little sad about that, moving on.

The secret message I received was, "Revvin' up your engine, Listen to her howlin' roar, Metal under tension, Beggin' you to touch and go" which is the first verse from the Kenny Loggins song *Danger Zone*. This begs the question, why is this relevant to a class about wireless communications. Well as previously stated I have adopted the middle name "Danger" so the title of the song alludes to my name. Additionally, this is in line with Professor Hoerning's love of rock music and how we both allude to movies, which in this case would be "Top Gun" starring Tom Cruise with a sequel set to be released on December 23, 2020. A note on the release date, "Top Gun: Maverick" was originally going to be released on June 24, 2020 but due to the Covid-19 pandemic was pushed back and the new release date is 2 days after my 24^{th} birthday. Overall, good choice of message.

The other hidden details are the phase and frequency offset. Since the frequency offset is supposed to be small compared to the chip rate of 1 MHz, we can assume that in the first frame there will not be too much drift. Given this, the initial phase offset is approximately -2.69 , or equivalently 3.59, radians, which is enough to make one believe that a different BPSK mapping was used, specifically $\{0, 1\} \rightarrow \{-1, +1\}$ which is exactly opposite of the actual modulation. Then the frequency offset can be determined using the chip rate and the derivative of the phase, $\frac{d\theta}{dt}$. We know that the chip rate is 1 MHz and 255 chips make a frame, which leads to a frame rate of $10^6/255 \approx$ 3921kHz. Then we can calculate the difference in the phase on a frame by frame basis which was calculated to approximately 0.889 radians/frame.

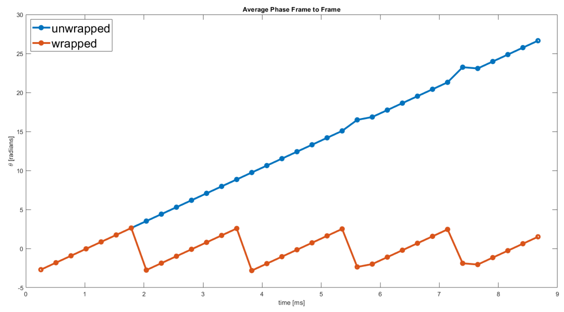

Fig. 8. Average Phase

As seen in Fig. 8 the calculated unwrapped phase is not a straight line at all points, so we approximate with the first few points, and the delta in phase is maintained at 0.889 radians/frame. If we now remember that $\frac{d\theta}{dt} = 2\pi f$, we are almost there, because we take the delta in average phase and multiple by the frame rate to approximate $\frac{d\theta}{dt}$ and then divide by 2π we get a frequency offset of 555 Hz.

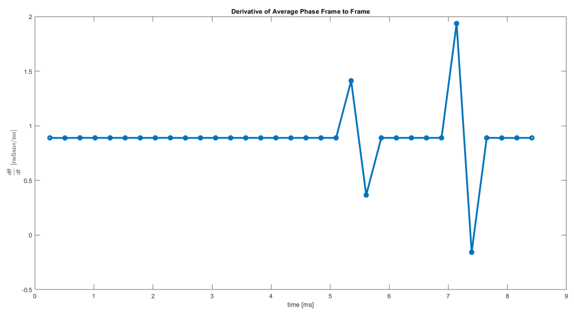

Fig. 9. Derivative of Unwrapped Average Phase

ACKNOWLEDGMENT

Where would I be without Karol Wadolowski, for without his help and guidance I would Shirleybe lost. Additionally, Armaan Kohli and Professor Hoerning for helping me figure out the framing because that's where the heart of this project lies.

REFERENCES

[1] "Linear Feedback Shift Registers," April 5, 2010. Accessed on: March 28, 2020. [Online]. Available: https://web.archive.org/web/20180119232153/http://www.newwaveinstruments.com/resources/articles/m_sequence_linear_feedback_shift_register_lfsr.htm