C Tech Information Technologies Inc.

Summer Internship

08/07/2024 - 12/07/2024

Week 3 Report:

12/07/2024

Fatih Mehmet Çetin

Purpose:

The purpose of this document is to provide an overview of my third week's activities during my internship. I aim to document the tasks I was responsible with, the skills I learned, and my experiences in my third week in the company.

Introduction:

I intern at the “Communications System Design” department within the company. The department consists of 4 engineers and myself as an intern. My third week had passed with learning and implementing more improved mathematical & digital communication terms such as Orthogonal Frequency Division Multiplexing, RRC Filtering, DFFT and IFFT in Matlab.

Work Done:

The very first task I assigned myself to was to work on and fix some codes I’ve written in the past two weeks. I have altered the modulation and demodulation codes I’ve written in the first week **(Appendices 1 to 6)**. When the input was divided into the bits per symbol ratio, if the remainder was not zero, the modules used to give an error. I fixed the issue by basically splitting the remainder from the original data. For instance, if I was performing QPSK and the input bits' length was 5, the new code would just give the output as 2 symbols for the first 4 bits and splitting the last remainder bit off.

My second task was to learn what orthogonal frequency multiplexing division (OFDM) was. Basically, it was a digital modulation technique used in many communication systems, including wireless networks, digital television, and broadband internet services. OFDM used subcarriers in frequency domain to use a limited bandwidth more efficiently. OFDM uses the property of orthogonality of signals. Inverse and discrete Fourier transform is used in the OFDM algorithm to obtain a time domain carrier signal and subcarriers in frequency domain. I've created a matlab module that simulated the OFDM process. I graphed bit error rates for different modulation schemes (BPSK, QPSK, 8-PSK and 16-PSK). Here is the figure I obtained **(Figure 3.1)**:

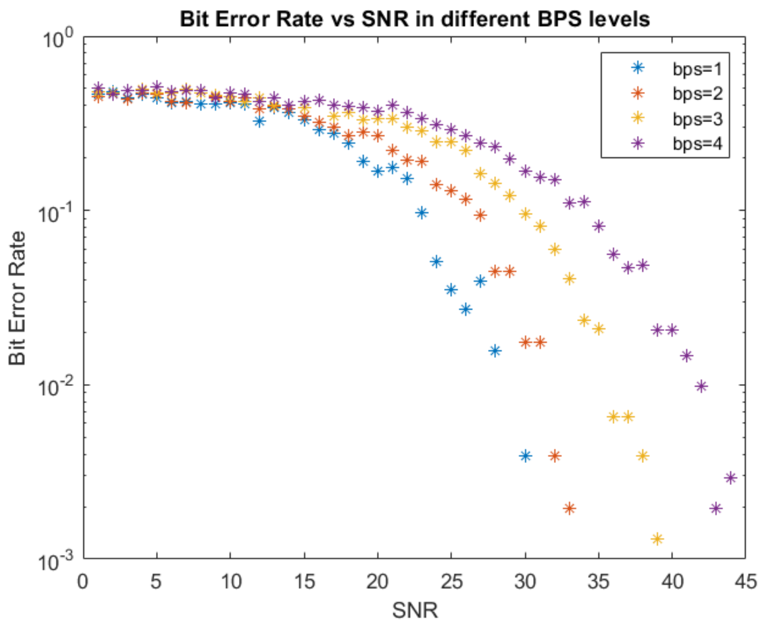
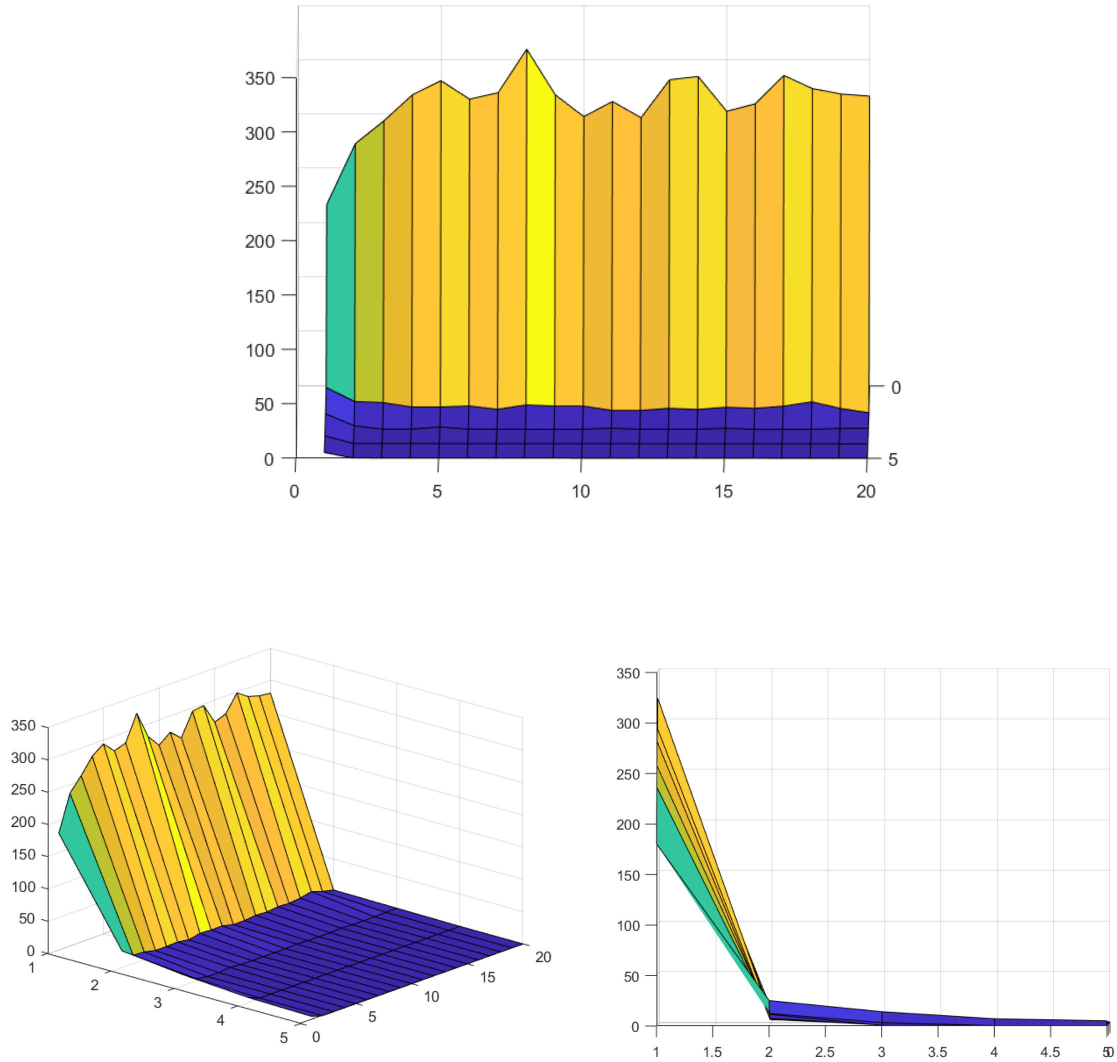


Figure 3.1: Bit Error Rates for different modulation schemes vs SNR

The graph showed that bit error rate was higher in higher bps values. The algorithm for the code went like this: I first modulated raw bits. Secondly I performed an IFFT operation to obtain a time domain signal. Then I added a cyclic prefix to the obtained time domain signal. Then I transmitted the symbols to the channel. The I did the reverse order of what I did so far, I removed the CP, performed a discrete FFT operartion and the demodulated the symbols **(Appendix 7)**.

I manipulated **appendix 7** on and on in order to have a better understanding about the OFDM process. I changed variables, plotted and observed inner variables within the system. This type of a test environment was pretty useful for me in order to understand OFDM better. I spent more than a day in this module changing and observing variables and outputs.

Then I did a test module where I modulated, upsampled, matched filtered, channeled, matched filtered again, downsampled and demodulated a signal. But this time I changed the filtlength and the sps parameters of the rrc filter **(Appendix 8)**. Here you can see the 3D-graph I obtained where filtlength changes from 1 to 20 and the sample per symbol changes from 1 to 5 **(Figure 3.2)**:



Figures 3.2.1 to 3.2.3: Error Rate for 10000 bits with different filterlength and sps parameters

The error counts are for 10 000 bits. We can clearly observe how important sps parameter is in an RRC filter. Sps value directly affects the error rate.

Then my supervisor tasked me with writing a header detection module for a long data by using correlation. He said to me that in real life receivers have no idea when the transmitted data starts. So, I was responsible with writing a module that will help receivers to identify when the header or the transmitted data starts.

I then started to think on the problem the managed to finally bring together a header detection module that took the long data, the header and the actual package length as inputs and gave the actual package as an output **(Appendix 9)**. I used the mathematical operation correlation for this process to detect the header. This module was definitely one of the hardest modules I've written at my internship.

Then I decided to implement this header detection via correlation technique on the module named test I've written in the previous weeks. But before I did that, I wanted to add an RRC filtering and de-filtering process to the test module. The operation was a complete success, I managed to add an RRC filter successfully to the test module.

I changed some old submodules and created some new ones for the main test module **(Appendices 10 to 13)**. Finally, I was able to change the test module such that it could detect original data from the raw inputted noisy data. This was a huge success and meant that I successfully managed to add a header detection by correlation algorithm to my repertoire.

As for the final work I did during week 3, I tested out different headers to see which one would be detected at a higher rate. You can see the headers and the responses of the test module to those headers **(Figures 3.3 to 3.6) (Appendix 14)**.

```
header1=[ones(1,15),zeros(1,15)];
header2=[zeros(1,15),ones(1,15)];
header3=[ones(1,30)];
header4=[zeros(1,30)];
header5=[1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 ];
header6=[0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 ];
header7=[zeros(1,10),ones(1,10),zeros(1,10)];
header8=[ones(1,10),zeros(1,10),ones(1,10)];
header9=randi([0 1],1,30);
```

Figure 3.3: Different Headers for the Simulation

```
avg =

    0.0057    0.0345    0.7252    1.0000    0.5635    0.6270    0.0485    0.0005         0
```

Figure 3.4: The average error rate for the headers (left: header1; right: header9)

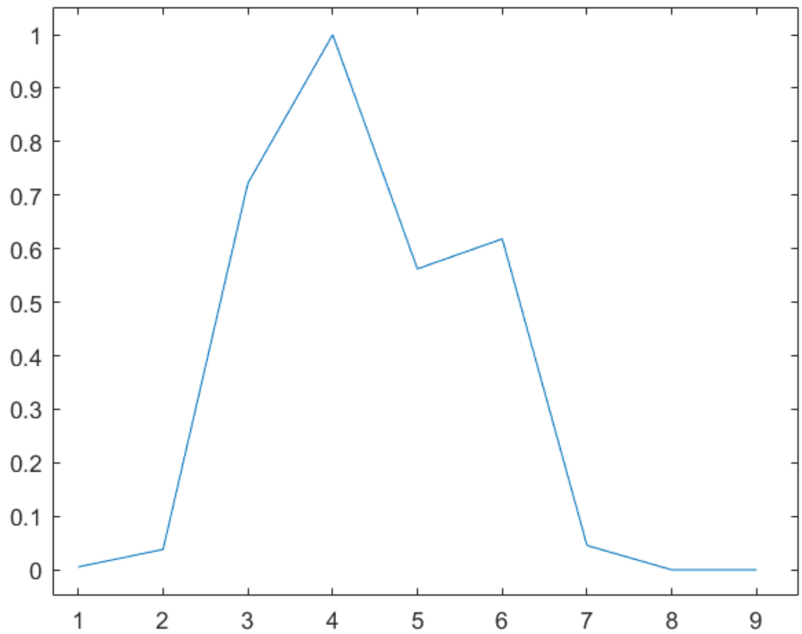


Figure 3.5: Plot of the average error rate for the headers (1: header1; 9: header9)

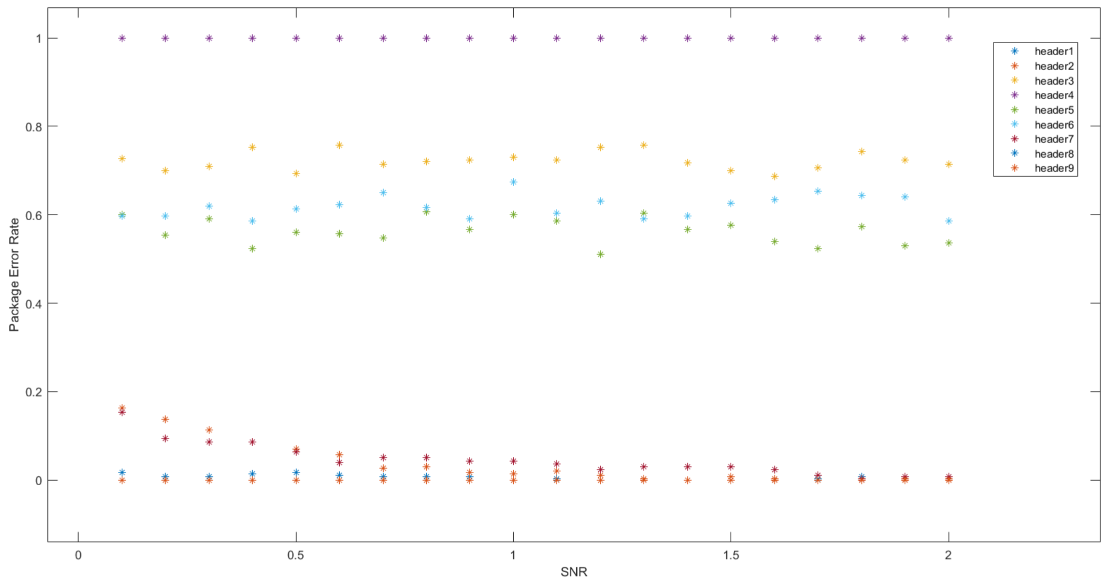


Figure 3.6: Plot of the error rates for the different headers

The plot meant that using asymmetric and unrepeated patterns more lead to a more accurate header detection by the system. This means that using only zeros and ones for a header was a bad idea. Headers should have a combination of them both as well as obtaining from repeated sequences.

Conclusion:

The third week had passed with learning and implementing more improved mathematical & digital communication terms such as Orthogonal Frequency Division Multiplexing, RRC Filtering, DFFT and IFFT in Matlab. I believe I got better at Matlab if I compare myself to the previous week. This week did not only contribute to my software simulation tool skills, but also, I learned significant theoretical information on signals and systems as well as mathematics which will help me when I take those courses in Bilkent.

Appendices:

1. https://github.com/fmcetin7/CTech-Internship/blob/main/week3matlabcodes/qpsk.m
2. https://github.com/fmcetin7/CTech-Internship/blob/main/week3matlabcodes/opsk.m
3. https://github.com/fmcetin7/CTech-Internship/blob/main/week3matlabcodes/spsk.m
4. https://github.com/fmcetin7/CTech-Internship/blob/main/week3matlabcodes/deqpsk.m
5. https://github.com/fmcetin7/CTech-Internship/blob/main/week3matlabcodes/deopsk.m
6. https://github.com/fmcetin7/CTech-Internship/blob/main/week3matlabcodes/despsk.m
7. https://github.com/fmcetin7/CTech-Internship/blob/main/week3matlabcodes/ofdmcp.m
8. https://github.com/fmcetin7/CTech-Internship/blob/main/week3matlabcodes/spsfiltlength.m
9. https://github.com/fmcetin7/CTech-Internship/blob/main/week3matlabcodes/detectheader.m
10. https://github.com/fmcetin7/CTech-Internship/blob/main/week3matlabcodes/packed.m
11. https://github.com/fmcetin7/CTech-Internship/blob/main/week3matlabcodes/test.m
12. https://github.com/fmcetin7/CTech-Internship/blob/main/week3matlabcodes/transformheader.m
13. https://github.com/fmcetin7/CTech-Internship/blob/main/week3matlabcodes/upsample.m
14. https://github.com/fmcetin7/CTech-Internship/blob/main/week3matlabcodes/headertest.m