ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE
School of Computer and Communication Sciences

Handout 21 Modern Digital Communications
Final Exam December 20, 2023

Name:

Note:

- You have 2 hours to work on the exam.
- The exam is closed book, but you are allowed one double-sided A4 page of handwritten notes, with NO processing (plain and pure handwriting on paper). Resources from the internet as well as code written outside this exam are not allowed (unless the code is written on the sheet of handwritten notes).
- The code will be evaluated according to the usual criteria, namely correctness, speed, form, and readability. Short comments that allow us to follow what you are doing will improve readability.
- The problems can be solved in any order.
- You will upload (to Moodle) your solution to the problems that require writing MATLAB or Python code. Do so in a single archive.

To get started with the exam, do the following:

1. Close all the windows and programs on your laptop.
2. Launch the MATLAB/Python editor and close all the tabs (previously written code).
3. From Moodle, download the file `mdc_final_2023.zip`. Unzip the file to create the directory `mdc_final_2023`. For the rest of the exam you are required to work inside that directory. The MATLAB/Python files for Problem n, are found in subfolder pn.
4. Turn your WiFi off until you are ready to upload your solutions.
5. Wait until you receive the go-ahead signal.

Problem 1. 23 points (MATLAB/Python)

In this problem you are asked to implement an OFDM transmitter and its corresponding receiver. The OFDM system should have $N = 256$ carriers and a cyclic prefix (CP) of length $L = 20$. The duration of an OFDM block, without the CP, is $T = 10^{-6}$ [seconds]. The data symbols to be transmitted are stored in the file `data_symbols.mat`. For transmitting these data symbols, you should use **only** the carriers corresponding to the baseband-equivalent frequencies of 30 MHz, 80 MHz, 110 MHz, -20 MHz, -70 MHz, and -100 MHz. You should also insert an OFDM block containing a training sequence which will be used for estimating the channel. Since we want to estimate the channel on all the carriers, you should use the (same) training symbol `1+1j` on all N carriers. Before the training sequence OFDM block, you should also insert a P/N sequence (stored in the file `pn_sequence.mat`), which will be used for synchronization purposes at the receiver. Your implementation should follow these steps:

1. Generate the transmitted OFDM samples, having the structure described above and depicted as well in Figure 1.

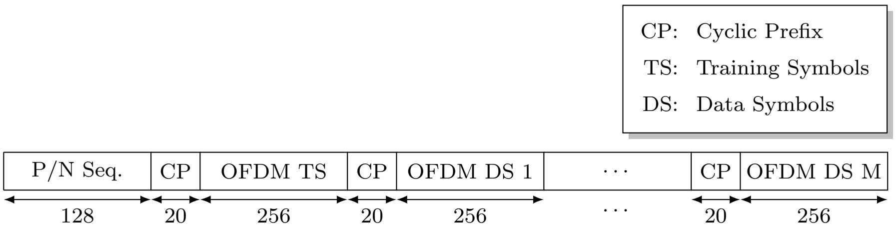

Figure 1: Structure of the transmitted signal

2. Generate complex white Gaussian noise samples, with zero mean, and such that the noise variance after the DFT at the receiver is 0.3.
3. Pass the samples generated in step 1 through the ISI (Inter-Symbol Interference) channel (provided in the given script) and then add the noise generated in step 2.
4. At the receiver, using the P/N sequence, determine the beginning of the OFDM blocks.
5. Implement the OFDM receiver.
6. Using the training sequence, estimate the channel (lambdas). Compute as well the theoretical lambdas and plot both the estimated and theoretical values on the same figure to check if they agree.
7. Correct for the effect of lambdas, and then plot the obtained constellation. You should see a noisy 4-QAM constellation.
8. Take decisions in order to estimate the transmitted data symbols.
9. Compute the symbol error rate (SER). If your implementation was correct, SER should be 0.

PROBLEM 2. 20 points (Paper and Pencil + MATLAB/Python)

Consider the transmitted OFDM samples having exactly the same structure (and values) as in the previous problem. These samples go trough an ISI channel $\boldsymbol{h}$ as in the previous problem as well (and white Gaussian noise is added after that).

1. Compute (theoretically, by hand) the values of the cyclic prefix (CP) of the training sequence OFDM block. *Hint*: You might find useful the fact that $\sum_{k=0}^{M-1} a^k = \frac{a^M-1}{a-1}$ where a is a non-zero scalar, $a \neq 1$.

2. Let $\boldsymbol{x} = (x_0, \ldots, x_{Q-1})^T$ be the samples of the transmitted P/N sequence ($Q = 128$), and $\boldsymbol{h} = (h_0, \ldots, h_{P-1})^T$ the channel impulse response (sample-level), where $P = 20$. Let $\boldsymbol{y} = (y_0, \ldots, y_{Q+P-2})^T$ be the noisy received values corresponding to the transmitted P/N sequence. Write down the expression of y_n (for $n = 0, \ldots, Q + P - 2$) as a convolution (in the form of a summation) between the transmitted samples and the channel coefficients (and plus noise, of course). $\boldsymbol{z} = (z_0, \ldots, z_{Q+P-2})^T$ are the corresponding samples of the noise at the receiver.

3. Write down $\boldsymbol{y}$ as a function of $\boldsymbol{h}$, $\boldsymbol{z}$ and a matrix that you need to find and specify.

4. Explain why it was important to compute and verify the values of the CP at point 1), and which influence these values would have had in your formula at point 3).

5. Derive the least squares (LS) estimate $\hat{\boldsymbol{h}}$ of $\boldsymbol{h}$ based on $\boldsymbol{y}$. We ask you to write down all the steps of your derivation, not only the final expression of the estimate.

6. The file `y.mat` contains the samples of $\boldsymbol{y}$. Implement the LS estimate $\hat{\boldsymbol{h}}$ of $\boldsymbol{h}$ based on $\boldsymbol{y}$. Plot the resulting $\hat{\boldsymbol{h}}$ on the same figure with the actual channel $\boldsymbol{h}$ (given in the script) and check if your estimate is correct. If your derivations and code are correct, you should find an almost perfect fit between the estimated $\hat{\boldsymbol{h}}$ and the actual $\boldsymbol{h}$. You might find useful the command `toeplitz(c,r)` (`scipy.linalg.toeplitz(c, r)`) which returns a non-symmetric Toeplitz matrix with `c` as its first column and `r` as its first row. You might also find useful the commands:
   `m\n` (for MATLAB) and
   `numpy.linalg.lstsq(m, n, rcond=None)[0]` (for Python).
   Of course, you need to figure out what to use for the matrices `m` and `n`.

Problem 3. 17 points (MATLAB/Python)

The file `ofdm_rx_symbols.mat` contains the symbols at the output of an OFDM receiver, after the correction with the channel coefficients (lambdas). The corresponding transmitted symbols were taken from a 4-QAM constellation. All the OFDM carriers ($N = 256$) were used for transmission. Due to some residual carrier frequency offset, the 4-QAM symbols in the given file are rotated and one cannot proceed to decode them directly. (You can convince yourself by plotting them.) In this problem we ask you to implement the following strategy for decoding:

1. The symbols of the first OFDM block are rotated, in average, with some small enough angle (you should plot them to see this), such that we can directly proceed to decode them, without making any correction. Decode the symbols of this first OFDM block, and then using the decoded symbols, estimate the rotation (some average value) which affected the first OFDM block. Correct with this value all the following OFDM blocks.

2. Keep proceeding like this for each subsequent OFDM block until you finish decoding all the blocks.

3. Plot the rotation you have obtained as explained above for all the blocks and check if it was more or less constant over all the blocks.

4. Using the file `data_symbols.mat` which contains the transmitted 4-QAM symbols, compute the symbol error rate (SER). If your implementation was correct, you should get SER = 0.