ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE
School of Computer and Communication Sciences

Software-Defined Radio: Final Exam
A Hands-On Course December 21, 2011

Name:

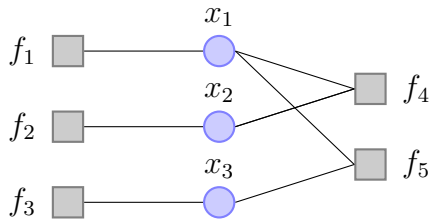
Note:

- You have 2 h 45 min to work at the exam.
- There are four problems that can be solved in any order.
- The exam is closed book (no notes allowed). You are only allowed to use the workstations in the laboratory (not your own laptops). Resources from the internet as well as code written outside this exam are not allowed.
- 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.
- One problem requires a handwritten solution that we will take at the end of the exam. The rest requires writing Matlab code that you will upload on Moodle (as a single archive).
- The Matlab files referenced below are available on Moodle.

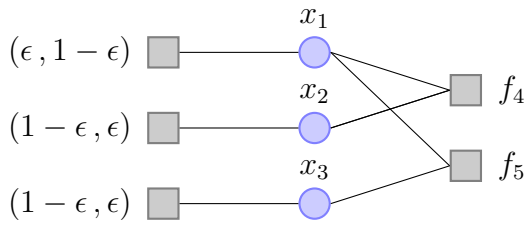
Start by downloading from Moodle and unzipping the file with all the data required for the different problems.

Problem 1. 20 p. (LDPC) (This is a "paper and pencil" problem.) Consider a binary code described by a parity check matrix H. This means that $\boldsymbol{x} = (x_1, \ldots, x_n)^T$, $x_i \in \{0, 1\}$, is a codeword iff $H\boldsymbol{x} = 0$, where operations are mod 2. The code is used to communicate across a binary symmetric channel of crossover probability ϵ, $0 < \epsilon < 0.5$.

(a) 5 p. Find the parity check matrix H knowing that the factor graph to compute the a posteriori probability is the one shown below.



(b) 10 p. A function of a binary variable like $f_i(x_i)$, $i = 1, 2, 3$, can be described by a vector of length 2 of the form $(f_i(0), f_i(1))$. Let $f_i(x_i)$, $i = 1, 2, 3$, be as in the following figure.



Work out by hand the sum-product algorithm to find the marginals of the a posteriori probability of X_i. (We use capital letters to denote random variables.)

Hint: With the notation introduced in class, the update rules are:

$$V_{n\to m}(x_n) = \prod_{m'\in\mathcal{V}(n)\setminus m} F_{m'\to n}(x_n)$$

$$F_{m\to n}(x_n) = \sum_{\sim x_n} f_m(\boldsymbol{x}) \prod_{n'\in\mathcal{F}(m)\setminus n} V_{n'\to m}(x'_n)$$

(c) 2 p. Using your result from the previous question, determine the most likely decision $\hat{x}_i$ for x_i, $i = 1, 2, 3$.

(d) 3 p. Describe $(f_i(0), f_i(1))$, $i = 1, 2, 3$, when the received vector is $\boldsymbol{y} = (0, 1, 0)^T$.

Problem 2. 20 p. (OFDM) In this assignment you will recover the text message sent through a finite-response AWGN channel using OFDM.

The signal has been obtained as follows: First we have produced a text message (hopefully you will find out what it is) and have converted the text message into bits using `de2bi(unicode2native(),8)` (with the text message as argument), which gives a matrix of bits. Those bits have been serialized row-by-row and the result has been converted into a sequence of integers taking values in $\{0, 1, \ldots, m\}$ for some value of m that you will determine. Then we have converted the integers into symbols using the functions `my_qammap.m` and `my_modulator.m`. The symbol sequence has been used to form blocks of size 14. The blocks of size 14 have been extended to blocks of size 16 by inserting 2 zeros at locations that you will determine (same location for all blocks). One block of the same length has been inserted in front to act as a preamble. We have then transformed each block via the IFFT of size 16 and added to each block the cyclic prefix of length 4. The resulting blocks have been serialized and sent over a finite-response AWGN channel. The channel output is the signal `rx_signal_noisy` that you obtain by loading `rx_signal.mat`. You may load it at this point. Doing so will also generate the channel coefficients `D` in the frequency domain.

Your assignment consists in completing the MATLAB file `ofdm_2011_ex.m` and answering a few questions. Your answers may also be written (as a comment) inside `ofdm_2011_ex.m`. Specifically:

(a) 2 p. How many OFDM symbols (of length $16 + 4$) have been received (including the preamble)?

(b) 5 p. By visually analyzing the received signal, determine in which positions we have inserted the 2 zeros.

(c) 4 p. Do channel equalization in the frequency domain. The FFT of the channel impulse response is given to you in the vector `D`. (The result is the OFDM symbol before IFFF plus additive white Gaussian noise.)

(d) 4 p. By looking at the signals, determine the modulation scheme used for the preamble and that used for the data. Write your answer as a comment in `ofdm_2011_ex.m`.

(e) 5 p. Recover and print the text message. Hint: To convert the equalized symbol sequence into integers from the alphabet $\{0, 1, \ldots, m\}$ you should use `my_qammap.m` and `my_demodulator.m`, both of which are provided.

Problem 3. 10 p. (LDPC) Complete the function `[H, Mc, Mv]=generateMatrices(socket)` that takes a socket matrix and produces the corresponding parity check matrix H, Mc and Mv. All 3 matrices shall be declared as sparse.

Problem 4. 10 p. (GPS)

- 5 p. Complete the function `iterateE.m` that determines the eccentric anomaly E of a satellite starting from the orbit eccentricity e and the mean anomaly M. The relation between these quantities is

$$E - e \sin E = M.$$

  The problem should be solved iteratively, starting with $E_0 = M$. The iterations should stop when the difference between two consecutively determined E values is smaller than the parameter $E_{tolerance}$.

- 5 p. Complete the function `t=timeInterval(taus, ind1, B)`. This function computes the signal (in seconds) it took a satellite to send the signal portion that in the receiver lies between sample number `ind1` and the start of bit number `B`. The vector `taus` contains the sample positions that mark the start of a bit. So `taus(1)` marks the start of the first bit etc.

  The function skeleton contains the interval T_b it takes to transmit a bit. You may assume that the duration of a bit at the receiver varies slowly so that it may be considered as constant over 2 bits. You can test your function by loading the vector `taus` stored in the file `taus.mat`. (`ind1` and `B` should be greater than zero and `ind1` should be smaller than `taus(B)`.)