ECOLE POLYTECHNIQUE FEDERALE DE LAUSANNE
School of Computer and Communication Sciences

Software-Defined Radio:
A Hands-On Course

Final Exam
December 21, 2016

Name:

Note:

- You have 2h45' to work at the exam.
- The exam is closed book, but you are allowed one A4 page (double-sided) of hand-written notes. 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.
- There is a `MATLAB` part and a paper-and-pencil part.
- 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.
- All the needed files are on Moodle in the folder `sdr_final_2016`.
- You will upload (to Moodle) your solution to the problems that require writing `MATLAB` code. You can do so in a single archive, or by uploading multiple files.
- The problems can be solved in any order.

Problem 1. 14 p. (Paper and Pencil)

Consider a toy (2D) GPS system that has 2 satellites on a circular orbit of radius R, see figure. The receiver is on the surface of a circle of radius r, (a toy Earth), with $r < R$.

Since the orbit is circular, the easiest way to describe the position of a satellite is to use polar coordinates $(\rho(t), \phi(t))$, where t refers to the GPS time. Suppose that the position of satellite 1 is $\left(R, \frac{2\pi}{T}t\right)$ and that of satellite 2 is $\left(R, \frac{2\pi}{T}t + \frac{\pi}{2}\right)$. We suppose that a receiver has this information. The figure shows the positions at GPS time $t = 0$. It takes T seconds for a satellite to make a revolution.

For the purpose of making distance measurements (range/pseudorange), each satellite sends a pseudonoise (PN) sequence (e.g. a C/A code) with chip of duration T_c. For both satellites, a new cycle of the PN sequence starts at GPS time $t = 0$. We assume that each satellite's clock is perfectly synchronized with the GPS time.

The receiver clock t_r has an unknown offset, i.e., $t_r = t + b$. Suppose that we want a pseudorange measurement at $t_r = \tau$. The receiver makes the following observations:

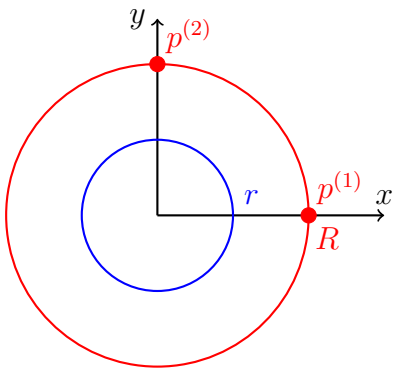
- The start of a new PN sequence, transmitted at $t = 0$, is received at $t_r = \tau^{(1)} < \tau$ and $t_r = \tau^{(2)} < \tau$, respectively.
- The number of chips between τ and $\tau^{(i)}$ is $n^{(i)}$, $i = 1, 2$.

(a) Using the parameters defined so far, express the time of flight $T_f^{(1)}(\tau^{(1)})$. This is the time of flight for a signal transmitted by satellite 1 and received at receiver time $t_r = \tau^{(1)}$.

(b) More generally, express $T_f^{(1)}(\tau)$. (Hint: sketch the various events on the GPS time line.)

(c) Let p be the receiver position. Describe, in polar coordinates, the position $p^{(1)}$ of satellite 1 for which $\|p^{(1)} - p\| = c\, T_f^{(1)}(\tau)$, where c is the speed of light.

(d) Write down the equations, the solution of which allows us to determine the receiver position p and the offset b at $t_r = \tau$. Note: To avoid dependencies from previous questions, it is sufficient that you write those equations using notation introduced thus far. For instance, you may assume that the functions $T_f^{(i)}$, $i = 1, 2$ are known.



Problem 2. 8 p. (`MATLAB`)

In this problem we assume to be in GF(2) and your solution is allowed to contain a `for` loop.

Your task is to write a script, called `backSubstitution.m`, that solves for `X` the equation `T*X = A`. The solution should rely on back-substitution. `A` and `T` are given in the files `A.mat` and `T.mat`, respectively. `T` is a lower triangular matrix with 1s on the diagonal. Your solution is not allowed to use `mldivide` or any other Matlab equation solver.

Problem 3. 14 p. (`MATLAB`)

To obtain full points, your solution should not contain `for` loops.

(a) Write a function `socketToH` that takes a socket matrix S and produces the corresponding parity check matrix H. Hint: below is a sample S and the corresponding H.

$$S = \begin{pmatrix} 2 & 2 \\ 5 & 2 \\ 3 & 1 \\ 4 & 2 \\ 4 & 1 \\ 1 & 1 \end{pmatrix} \qquad H = \begin{pmatrix} 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \end{pmatrix}.$$

(b) Write a function `socketToM` that takes a socket matrix S and produces a matrix M that serves the following purpose. Suppose that v is a *row* vector that contains the variable-node-to-check-node messages. (In class we have used *column* vectors!). We want $u = vM$ to be the row vector that, at position i, contains the sum of the messages that go to check node i. We are asking that you find M directly, not by first constructing its transpose M^T. For the above S, this matrix is

$$M = \begin{pmatrix} 0 & 1 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 0 \end{pmatrix}.$$

(c) Now suppose that you want to multiply (rather than add) the messages that go into a check node. For this, we use the command `prod(v(I))`. This can be done, assuming that M has the same number of 1s in each column. (We assume that it is the case.) Write the function `produceI` that takes M and produces I. For the above M, the desired matrix is

$$I = \begin{pmatrix} 3 & 1 \\ 5 & 2 \\ 6 & 4 \end{pmatrix}.$$

Problem 4. 14 p. (MATLAB)

Write a MATLAB function according to the following header. A template is given in `ofdmSymbols.m`

```
% OFDMSYMBOLS = ofdmSymbols(dataTx, noCarriers, cpLength, powerMask)
% This function generates OFDM symbols with a power spectrum shaped
% according to the energy per carrier given by powerMask. The
% variable dataTx contains the information symbols, and should be a
% row vector of length a multiple of noCarriers. The number of
% carriers and the length of the cyclic prefix are given by
% noCarriers and cpLength respectively. The powerMask and
% noCarriers have the same dimension. The columns of OFDMSYMBOLS
% are OFDM symbols.
```

Hint: to test your solution, you may want to run the script `testOFDM.m`. You should obtain the figure below.

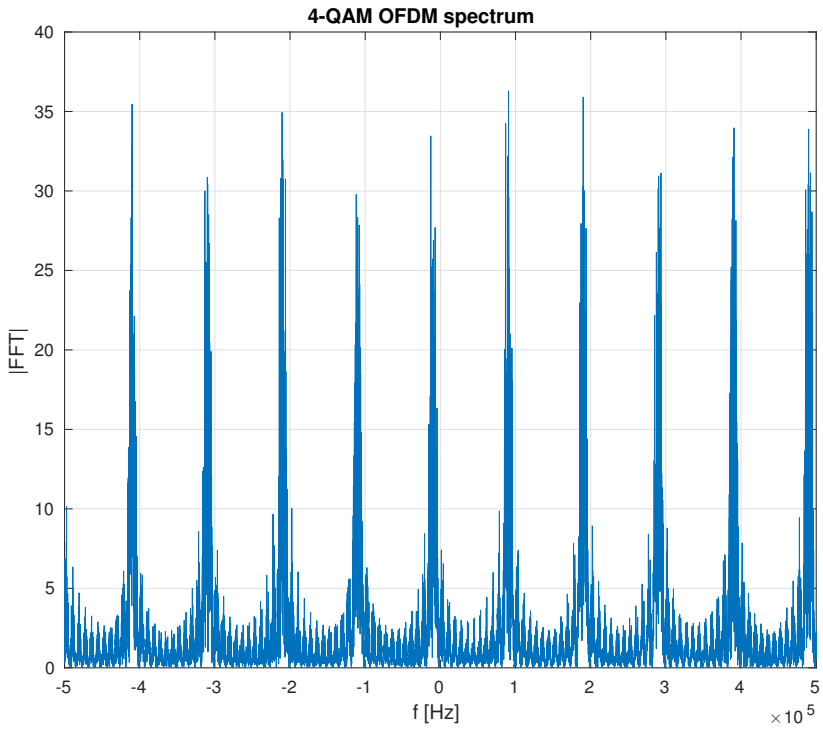


Figure 1: Example output of the function `testOFDM.m`

Problem 5. 10 p. (`MATLAB`)

Load the AM-modulated signal via the command `load('amSignal.mat')`. Using the FFT, find the carrier frequency, which is the strongest component in the positive spectrum. Remove the negative frequencies, and shift the spectrum to baseband. Your script, called `basebandEquivalent.m`, should produce the (time-domain) baseband equivalent of the given AM-modulated signal.

Note: no need to worry about complex-valued scaling factors.