ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE
School of Computer and Communication Sciences

Handout 11 Modern Digital Communications
Midterm Exam November 8, 2017

Name:

Note:

- You have 2 hours to work at the exam.
- The exam is closed book, but your are allowed one sheet (one single-sided page) of handwritten notes. 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 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 MATLAB and close all the tabs (previously written code).
3. From Moodle, download the file `mdc_midterm_2017.zip`. Unzip the file to create the directory `mdc_midterm_2017`. For the rest of the exam you are required to work inside that directory.
4. Turn your WiFi off until you are ready to upload your solutions.
5. Wait until you receive the go-ahead signal.

PROBLEM 1. 8 points (Paper and Pencil)

Read the following code, where `TBDi` ($i = 1, 2, 3$) is a place-holder for a number that you will determine (see the questions below).

```
close all; clear all;

L = 30; % number of bits transmitted
up = 5; % upsampling factor and pulse length
p = [1 1 -1  1 -1]; % pulse
Es = 9; % symbol energy

start = TBD1;
stepSize = TBD2;
channelNoiseVariance = TBD3;

% create the bits
b = randi(2,1,L)-1;
% map them into BPSK symbols
s = sqrt(Es)*(1-2*b);

% create the transmitted signal
sUP = upsample(s,up);
x = conv(sUP,p);

% add WGN
channelNoise = sqrt(channelNoiseVariance)*randn(1,length(x));
r = x + channelNoise;

% receiver front-end
y = conv(r,fliplr(conj(p)));
suffStat = y(start:stepSize:end-(up-1));
```

The following questions can be answered in any order, but keep in mind that the pulse `p` is not normalized.

1. Specify TBD1 and TBD2, so that `suffStat` contains the sufficient statistics (matched filter output sampled at the correct places).
2. In the absence of channel noise, what is the alphabet of `suffStat`?
3. Specify the value of TBD3, so that the variance of the noise in `suffStat` is 1.

PROBLEM 2. 12 points (MATLAB)

The file `mf_output.mat` contains the samples of a signal at the output of the corresponding matched filter, before downsampling. The transmitted symbols are drawn from a 4-QAM constellation and the upsampling factor is 50. We assume that the channel introduces an unknown delay of d samples, with $0 < d < 50$. Note: Your code should contain no loops.

1. Plot the eye diagram for the in-phase component (i.e. the real part of `mf_output`).

2. Modify the code that you have written, so that you plot three eye-diagrams using the command `subplot`. In the first and second subplot, plot the eye diagram of the in-phase and quadrature component, respectively. (The quadrature component is the imaginary part of `mf_output`). In the third subplot, plot the eye diagram of the absolute value of `mf_output`.

3. From the previous three plots, you should see that either one can be used to estimate d. Write a code that estimates d based on one of the eye diagrams (choose the method that you find more convenient to code).

4. Once you find your estimate of the delay d, use it to downsample `mf_output` and do a scatterplot of the received symbols.

PROBLEM 3. 9 points (MATLAB)

The file `rx_signal.mat` contains the samples of a signal at the output of the channel. The transmitted data symbols are drawn from a 4-QAM constellation and they are preceded by a preamble of BPSK symbols. The preamble symbols are stored in the file `preamble.mat`. The transmitted symbols (preamble + data symbols) are pulse shaped with a root-raised-cosine pulse truncated to a total length of 8 symbols, having a rolloff factor $\beta = 0.22$ and using an upsampling factor of 50. We assume that the channel delays the transmitted signal. Note: Your code should contain no loops.

1. Process the received signal by passing it through the appropriate matched filter.
2. Find the start of the data symbols by correlating the matched-filter output (before downsampling) with the upsampled version of the preamble.
3. Downsample the matched-filter output accordingly and plot the resulting constellation of received (noisy) symbols (data symbols only).

PROBLEM 4. 11 points (Paper and Pencil / MATLAB)

Let p be the vector that contains the samples of a pulse $p(t)$, taken at $t = nT_s$, $n = 0, 1, \ldots, L-1$. You can assume that the conditions of the sampling theorem are fulfilled, so that you can reconstruct $p(t)$ from p.

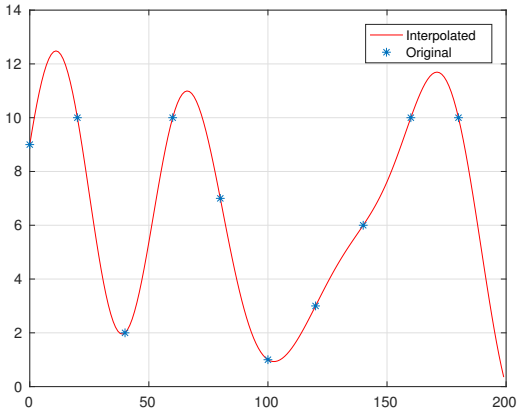
Let k be some positive integer. We are interested in the vector q that contains the samples of $p(t)$, taken at $t = n\frac{T_s}{k}$, $n = 0, \ldots, Lk-1$. We want to construct q via the command `conv(a,b)` for some vector `a` and some vector `b`.

1. Using the reconstruction formula of the sampling theorem, write down on paper the expression for $p(n\frac{T_s}{k})$.

2. Rewrite your expression so that it is a convolution between two sequences. Specify the two sequences.

3. In `interpolate_incomplete.m` complete the lines that define the two vectors `a` and `b` that contain those two sequences. One sequence has to be truncated since it has infinite length. For now, do not optimize the length of the truncated sequence, and do now worry about the relative positions of the sequences inside the vectors. The code then computes `y=conv(a,b)`. Notice that `y` contains `q`.

4. To find which position of `y` contains the first component of `q`, use the fact that, for a convolution, if you delay an input by some amount, the output is delayed by the same amount. In the code, specify the value of the index `ind` such that `y(ind)=q(1)`. If you have it right, when you run your code, you should obtain the figure below:



5. Bonus question: what is the length of the shortest truncated sequence (question 3 above) for which there is no performance degradation?