ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE
School of Computer and Communication Sciences

Software-Defined Radio: A Hands-On Course — Midterm

October 30, 2013

Name:

Note:

- You have 1 h 30 min to work at the midterm.
- The exam is closed book (no notes allowed). You are only allowed to use the work-stations in the laboratory (not your own laptops). Resources from the internet as well as code written outside this exam are not allowed.
- Some parts require writing MATLAB code that you will upload on Moodle.
- 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.

Good luck!

Problem 1. (8 p.) Let $s(t)$ be defined as

$$s(t) = \sin(2\pi f_1 t) + \sin(2\pi f_2 t)$$

for $t \in [0, T]$, and zero otherwise. The parameters are $f_1 = 100\text{Hz}$, $f_2 = 300\text{Hz}$, and $T = 10\text{s}$. Write a MATLAB script named `p1.m` which displays the frequency spectrum of $s(t)$, sampled with sampling frequency $f_s = 1000\text{Hz}$. The graph should have $f = 0$ in the middle, and the axes should be labeled in Hz. Do not forget to upload `p1.m` on Moodle.

Problem 2. (8 p.) Let $s(t)$ be defined as

$$s(t) = \sin(2\pi f_1 t) + \sin(2\pi f_2 t)$$

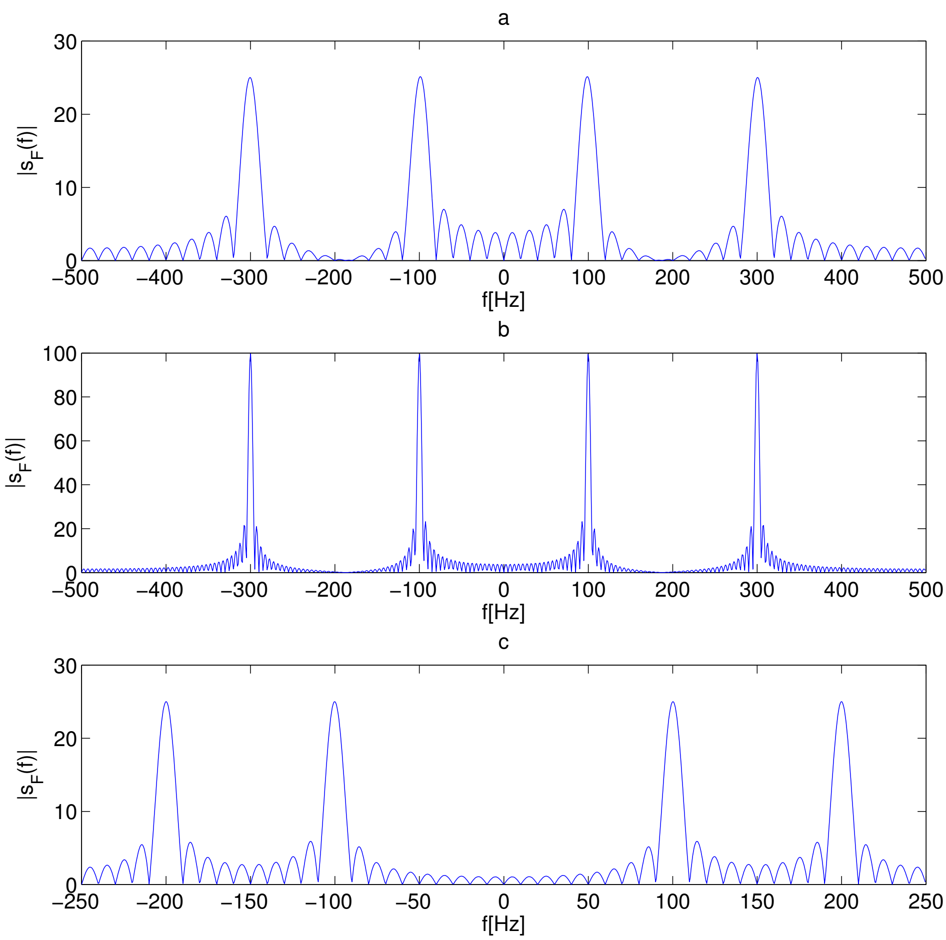
for $t \in [0, T]$, and zero otherwise, with frequencies $f_1 = 100\text{Hz}$ and $f_2 = 300\text{Hz}$. Each of the following figures shows the absolute value of the spectrum of $s(t)$, obtained via FFT.

(a) Consider the top figure first. The peaks are sinc functions. Explain why.

(b) Determine the sampling frequency f_s and the duration T that were used for the top figure.

(c) Describe what parameter(s) have changed for the middle figure. (The frequencies f_1 and f_2 did not change.)

(d) Now consider the bottom figure. Notice the location of the peaks. Describe what parameter(s) have changed. (The frequencies f_1 and f_2 did not change.)



Problem 3. (3 p.) Let a and b be two row vectors of the same length. Write (on paper) the MATLAB code to compute the inner product between a and b.

(a) using `xcorr`;

(b) using `conv`;

(c) using vector multiplication.

Which of the three methods is more efficient? Motivate your answer.

Problem 4. (3 p.) The sequence

```
0 0 0 0 -1 0 -1 0 -2 2 2 -2 -3 0 1
```

is the result of the `MATLAB` command `xcorr(s1,s2)`, where `s1` and `s2` are binary sequences taking value in $\{\pm 1\}$. Determine:

(a) the longest between `s1` and `s2`;

(b) `length(s1)+length(s2)`;

(c) `length(s1)` and `length(s2)`;

Problem 5. (12 p.) For this problem, you need to download the files available on Moodle, under the "Midterm 2013" link.

The code on the next page has created a channel output sequence called `received`. Your task is to write the code for a receiver that attempts to recover the bits that were contained in `bitsR` and `bitsI` when we ran the code. The two sequences of bits were mapped to QAM modulation symbols, then transmitted over a channel. In addition to adding white Gaussian noise, the channel also introduced a delay and a rotation. (Same rotation for all symbols.) Study the code on the next page to understand how the bits are mapped into symbols and how the channel acts on the symbol sequence. Afterwards, open the file `p5.m` and complete as following. Do not forget to upload the program on Moodle.

(a) Use the training sequence to estimate the rotation and the start of the symbol sequence within `received`

(b) Isolate the noisy symbol subsequence from the rest of `received` and correct for the phase rotation that occurred during transmission.

(c) Plot the noisy symbols on the complex plane, before and after rotation correction. (For the corrected signal, you should see 4-QAM clouds.)

(d) Decode the bit sequence. You can compare your result with the bit sequences stored in `bitsR.mat` and `bitsI.mat`, respectively. (In our implementation, all the bits are correctly decoded.)

```
%% Produce a Received Signal
%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% initialize
clear all;close all;
nSymbols=1000; % number of transmitted symbols
maxDelay=500;
sigma=0.2; % noise satandard deviation
load training;

%% create the bit and symbol sequence
bitsR=2*randi(2,1,nSymbols)-3; % bits mapped to the real axis
bitsI=2*randi(2,1,nSymbols)-3; % bits mapped to the imaginary axis
symbols=bitsR+1j*bitsI;

%% transmitted signal
signal=[training symbols]; % taining sequence followed by symbol sequence

%% generate the channel's parameters
phi=random('unif', 0, 2*pi) % random phase
delay=randi(maxDelay+1)-1; % a random channel-delay is generated
lengthReceived=(delay+length(training)+nSymbols); % rec. signal's length

%% received signal
% start with the noise and then add the delay and rotated signal
received=(randn(1,lengthReceived)+1j*randn(1,lengthReceived))*sigma/sqrt(2);
signalRotated=exp(1j*phi)*signal; % rotated signal
received(1+delay:delay+length(signal))=...
    received(1+delay:delay+length(signal))+signalRotated; % add the signal

%% save the received signal
save 'received' received
```

Problem 6. (3 p.) The following script simulates the transmission over an AWGN channel of a vector `y` of complex QAM symbols. The performed operations are: producing the samples of the modulated pulse train, adding the noise, and obtaining sufficient statistics about the transmitted symbols. The signal to noise variance ratio in dB is `Es_sigma2`. The code contains three errors. Modify the corresponding lines in order to correct these errors.

Note: All variables are assumed to be defined.

```
%% producing the pulse train; USF is the upsampling factor
% and h is the sampled pulse

y_up = upsample(y, USF-1);
z=conv(y_up,h);

%% simulating the awgn channel

Es=var(y);
sigma = sqrt(Es*10^(-Es_sigma2/10));
noise = sigma * randn(size(z));
r=z+noise;

%% obtaining the sufficient statistics at the receiver

y_rec = conv(r, h);
x = y_rec(length(h):USF:end-length(h)+1);
```

Problem 7. (3 p.) The following MATLAB operations can be performed more efficiently. Write the equivalent code. Note: Each operation can be rewritten in one line.

(a)
```
%% determining the square of each element of vector X

x2=zeros(size(x));
for i=1:length(x);
    x2(i)=x(i)^2;
end
```

(b)
```
%% finding the position of the maximum element of vector X

pos=1;
mVal=x(1);
for i=2:length(x)
    if x(i)>mVal
        pos=i;
        mVal=x(i);
    end
end
```

(c)
```
%% mapping a vector of integers between 0 and 3, denoted X,
% to QAM constellation points
% you know that qammap=[-1+1i, -1-1i, 1+1i, 1-1i]

x_qam=zeros(size(x));
for i=1:length(x)
    x_qam(i)=qammap(x(i)+1);
end
```