Introduction

Virtually all modern communication systems are based on digital data transmission, e.g. mobile networks like GSM, UMTS and LTE, or broadcast systems like DVB. To detect the data in the receiver, the distortions that are introduced during the transmission must be estimated and corrected. Today, these tasks are mostly performed by digital signal processors.

In this lab course, you will learn to design digital receivers for realistic wireless systems, with a focus on signal processing aspects. You will start by modelling a very simple transmission system where the signal is only disturbed by additive noise. Realistic channels however introduce many other distortions into the signal, for example an unknown timing offset in the A/D converter. Therefore, in each lab you will adapt the receiver to more and more realistic channels by adding components like synchronization units and estimators for unknown channel parameters. In the end, you will have built a receiver that could be used in a realistic communication system.

As we are only concerned with the digital part of the receiver, we start our examination behind the A/D converter. In each lab, you will be given time-discrete signals which model the output of the A/D converter and thus the input into the digital signal processor. The signals contain images as payload data, so by recovering the data and displaying the received image on the screen, you will get a qualitative impression whether your receiver works properly or not. In order to quantify the performance of your receiver, you will also receive signals that contain a known bit sequence so you can calculate the bit error rate.

MATLAB

Throughout this course we will use MATLAB, which is a programming language specialized on numerical computing.

If you are unfamiliar with MATLAB, we suggest that you start with reading the "Getting Started" section in the MATLAB documentation, which can be found at `http://www.mathworks.com/access/helpdesk/help/helpdesk.html`.
There are also many tutorials available in the internet, for example `http://www.cyclismo.org/tutorial/matlab/`.

Here is a brief list of some useful commands that you might need during the course:

`help <functionname>`	Displays help for the function `functionname`
`zeros(m,n)`	Returns the $m \times n$-matrix of zeros
`ones(m,n)`	Returns the $m \times n$-matrix of ones
`length(x)`	The length of the vector `x`
`size(x)`	The size of the matrix `x` (number of rows and number of columns)
`reshape(x,m,n)`	Returns the $m \times n$-matrix whose elements are taken columnwise from `x`
`repmat(x,m,n)`	Repeats the matrix `x` $m \times n$ times
`circshift(x,m)`	Circularly shifts the elements of the vector `x`
`angle(x)`	The argument of a complex number
`max(x)`	The maximum element of the vector `x`
`sum(x)`	The sum of the elements of `x`
`mod(x,y)`	The remainder of the division `x/y`
`plot(y)`	Plot the data in the vector `y`
`conv(x,y)`	Returns the convolution of `x` and `y`
`awgn(x,snr)`	Add White Gausian Noise to transmitted signal `x`
`bi2de(bits)`	Converts a vector of bits to an integer

Furthermore, many standard mathematical functions like `abs`, `sin`, `exp` and so on are available for (complex-valued) matrix arguments; they return a matrix of the same size where the function is applied elementwise. The variable `pi` is predefined with π, the variables `i` and `j` are predefined with the complex number $\sqrt{-1}$.

Most available operators have their usual meaning, as known from other programming languages like C/C++. Some operators, however, have a different meaning when used with vectors or matrices as arguments. For example, `a*b` denotes a matrix multiplication, whereas `a.*b` is an elementwise multiplication. Since we are mainly dealing with complex-valued vectors, you also need to differentiate between the transpose operators `'` and `.'`. The prime `'` is the hermitian-transpose operator, i.e. `a'` is the transpose and complex-conjugate of `a`, whereas `a.'` is simply the transpose of `a`.

MATLAB provides an in-program help for each function, which can be accessed by the `help <function>` command.

Instructions

For the most tasks we provide you with some code to help you focus on the important things. You can download these files from the EPFL Moodle page [1] of the course. Please prepare a small report (as a PDF document) with your answers. Reason all your answers. If you have programmed any MATLAB code, document your code with inline comments and upload all scripts and resource files required for execution together in a compressed archive on the Moodle page.

[1]URL: http://moodle.epfl.ch