Laboratory work instructions in Digital Communications 1

Spring 2009

Daniel Aronsson

Note! Only the last part of this laboratory work will be conducted in the lab room. This means that the greater part of the laboration will be done at e.g. your home computer *prior to the laboratory part.* When you come to the lab room to do the last part of this assignment, the instructor will check that you have done and understood the earlier parts. Only after showing graphs of your results will you be allowed access to the laboration setup. Therefore, carefully read this instruction in good time for your laboratory session.

1 Introduction

The goal of this laboratory work is to

- increase your understanding for how the different parts of a communications system work, and
- verify that the measured error performance of the system matches theoretical expressions.

In this laboratory work you will get a chance to study the individual parts of a complete communication system. These parts are schematically depicted in Figure 1. In the first part of the laboratory work, everything – even the channel – will be simulated in Matlab. This means that the only thing you need for this part is a standard computer (about 1 GHz and 500 MB RAM is recommended).

The second part will be performed in the radio lab at Ångström, 7K1401. Here, data will be sent over an actual radio link. The data is output and recorded by the computer's sound card. Since the sound card is unable to play low frequencies, your signal will be digitally modulated on a 2 kHz carrier. The radio equipment will then modulate that signal on a carrier of about 1 GHz. In fact, instead of using the radio interface, you might just as well send your data acoustically. After you finish Section 3, you can connect a speaker and a microphone to your sound card and try to send your data as a sound wave!

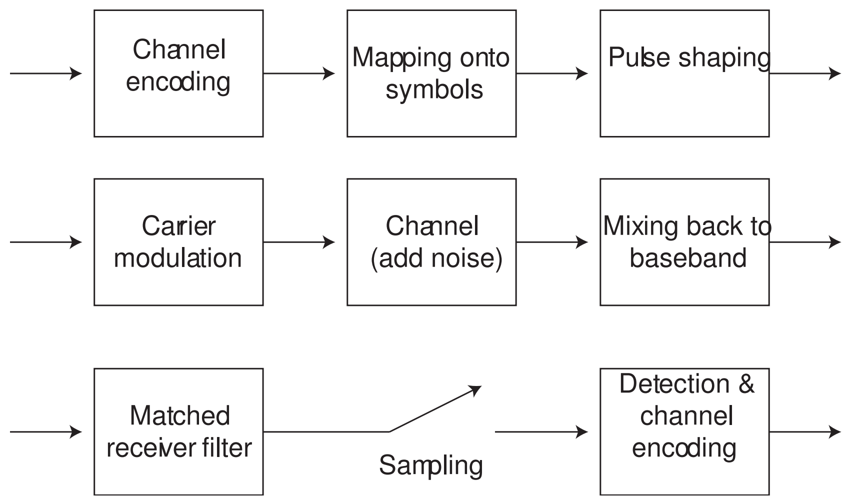

Figure 1: The different blocks in the digital communication system.

2 Preparatory work

Among other functionality, the script that you will run in the laboratory work requires eight Matlab functions: two modulators and two demodulators (QPSK and 16QAM), and two channel encoders and two channel decoders (Hamming 7,4 and Hamming 15,11). You will test the performances of all four combinations of these and compare them with theoretical results. The functions are available on the homepage. Download them and make sure they conform to the below properties:

- The output from your encoders should be longer than the input bit stream by a factor $1/R_c = n/k$ (15/11 for Hamming 15-11 a.s.o). The reverse goes for the decoders.
- The output from your modulators should be shorter than the input bit stream by a factor $log_2 M$ (3 for 8PSK a.s.o). The reverse goes for the demodulators.
- Make sure your decoders undo what your encoders do. `c=my_encoder(m); m2=my_decoder(c)`. Also change a few of the coded bits `c` and verify that the decoded bits `m2` are equal to the original bits `m`.
- Make sure your demodulators undo what your modulators do. `[I,Q]=my_modulator(m); m2=my_demodulator(I,Q)`. Also add a little (just a little) noise to the symbols `[I,Q]` and verify that the detected bits `m2` are equal to `m`.
- Everything, inputs as well as outputs, should be column vectors.

3 The laboration part 1 - simulation

In order to perform this laboration you will need to download a few files to your home directory. Download the following files from the course homepage and save them in the directory:

- the main script `dc1_comsys.m`,
- two pairs of channel encoders and decoders, `encode_*.m` and `decode_*.m` (you downloaded these earlier),
- two pairs of modulators and demodulators, `mod_*.m` and `det_*.m` (you downloaded these earlier),
- the square-root-raised-cosine pulse shaping function `srrc.m`,
- a function for cross-correlation, `xcorr_dc1.m`, needed by the receiver, and
- a random sequence of QPSK symbols, `trainingsequence.mat`, needed by the receiver to find the start of the transmitted waveform.

This section will take you on a guided tour in the provided Matlab script, `dc1_comsys`. You will study the different signals both in the time domain and the frequency domain and hopefully you will acquire a better understanding for how a digital communication system works.

Let us begin.

3.1 From bits to waveform – coding, modulation, and pulse shaping

There are four places in `dc1_comsys` that contain function calls to modulators/detectors and encoders/decoders. These are marked 'INSERT FUNCTION CALL HERE'. Uncomment these rows and insert the appropriate function names. There are four combinations of codes and modulation formats. You will investigate all of them, so begin with whatever combination you feel like.

Now you should be ready to go. We will first make a stop just after the initial signals have been set up so that we can study them more carefully.

Insert a `return` command immediately before the 'create pass-band signal' section. This will stop the code at this point when you run the code. Set the *coderate* and *M* (number of symbols in the constellation) variables in the beginning of the script to match your choices of modulator and encoder. Also make sure that the *Nbits* variable is divisible with the *k* value of your encoder. Set the *channel* variable in the top of the script to 'simulated'. Save and run.

Hopefully the code will run for a few seconds and then stop without producing an error message. Block one to three have now been executed. We begin by studying what happens in blocks one and two. In block one, our original uncoded message, m, is channel encoded by the encoder you chose in the beginning. Block two maps your coded bits onto symbols. This is performed by the modulator which you also chose in the beginning.

EXERCISE 1 **Vector lengths**

What is the ratio between the length of the original uncoded bits m and the length of the coded bits c?

What is the ratio between the length of the coded bits vector c and the I/Q signals?

Another thing that happens in this block is that a few known symbols are added to the beginning of the symbol sequence. The reason for adding such a *training sequence* is that we need a way to find the signal at the receiver side of the communication system. More on that later. For now, it will suffice for you to remember that the first few symbols in our symbol vector is known to the receiver (and hence they don't carry any information).

For technical reasons (soundcard startup) we will have to add a few zeros in the very beginning of the symbol sequence, but if you study the code you'll see that the script also adds some zeros *after* the symbol sequence. The reason is that in the real radio setup, there will be a delay between the transmission and the reception. The trailing zeros account for the delay so that the whole messages can be received before the transmission is terminated.

Block three takes care of the pulse shaping. The pulse that we use here is a *square root raised cosine pulse*. It is stored in the variable *pulse*.

The generated pulse shaped information signal is called *tx_s_bb*. More of these cryptically named signals will come along, so a presentation of the naming convention might be appropriate at this point:

Naming convention: The first part of the variable name describes whether the signal is due for transmission or have been received (*tx_* or *rx_*). The last part is for 'baseband' or 'passband' (*_bb* or *_pb*). The middle part stands for 'signal' or 'noise'.
At the end of the script, the received signals will be filtered to form *rx_(s/n)_bb_filtered*. After that, the filtered signals are sampled at symbol rate to form the final *rx_(s/n)_bb_sampled*.
In order to measure the noise power (and from it calculate the E_b/N_0), we'll send a sequence of zeros and save the recorded sequence as *rx_n_pb*. By measuring the power of the mixed down, filtered, and sampled noise, *rx_n_bb_sampled*, we'll get a good estimate of the noise power. There are no *tx_n_pb*/*tx_n_bb* signals since they would consist of only silence. Note that the *rx_s_pb*/*rx_s_bb* are your information signal distorted by noise. The ratio between the power of *rx_s_bb_sampled* and *rx_n_bb_sampled* will therefore be E_s/N_0, where E_s is the symbol energy and N_0 is the noise spectral power density.

The *rx_trainingsequence* signal, generated last in block three, is the short training sequence of known symbols, upsampled and filtered *twofold* through the filter *pulse*. This signal will be used on the receiver side to locate the transmitted waveform in the received signal.

Before we leave this section we will study the impact of the pulse shape on the power spectral density of the signal. Note that the rolloff of the SRRC pulse is set to 0.5 in the script. We will now study how the rolloff impacts the power spectrum.

EXERCISE 2 **Impact of the rolloff factor**

Plot the pulse shape *pulse* by issuing the command `plot(pulse)`. Also plot the power spectrum of the transmitted baseband signal (*tx_s_bb*) with the command `plot(f,20*log10(abs(fft(tx_s_bb))))` and zoom in on the *first null*. The first null is the frequency at which the big "bulge" in the middle of the spectrum dips to zero.

Now change the rolloff factor of the filter to 1 (the last argument in the function call `srrc(fs,R,Nfilter,rolloff)`). Rerun the script. Again, plot the pulse shape and the power spectrum zoomed in on the first null.

Finally, set the rolloff to a very low value, say 0.001, and produce the two plots. Do the spectra look the way you expected? (What does the theoretical spectrum look like? Look up the expression for a SRRC pulse in the textbook.). Where are the first nulls in the respective plots? Are they located where the theoretical expression say the should be? How do you explain the side lobes (should a SRRC pulse have side lobes?)? What would we do to reduce the sidelobes?

We have arrived at our first stop. We will now see what happens when we modulate the carrier and send and receive the signal.

3.2 From baseband to passband, and back

Remove the `return` command and instead put it after block seven, before the correlate and measure section. Reset the rolloff factor of the SRRC filter to 0.5. Save and run.

Your workspace will now contain more signals that were mentioned in the naming convention part. Which part of the script is executed depends on how the *channel* variable is set, but essentially what happens is that block four moves the signal from the baseband to the passband (carrier modulation), block five lets the channel affect the signal, and block six mixes the signal back to the baseband. Since we're now running the system in simulated mode, the latter part of the `if` statement will be executed.

EXERCISE 3 **Passband signal propreties**

Plot the spectrum of the transmitted passband signal (tx_s_pb) (with the command in Exercise 2). Does it look as you expected? Where is the centre frequency? Now also plot the spectrum of the baseband signal, tx_s_bb. In terms of symmetry, is there a difference between the spectra of the baseband and the passband signals (look closely)? Also plot the spectrum of the received signal (passband or baseband) and note the impact of the noise.

Note that the rx_n_bb/rx_n_pb signals are measurements of the noise only. Their purpose is only to allow us to measure the noise power and they are not very interesting to look at.

Block seven filters the signal with a filter matched to the SRRC pulse shape.

Exercise 4 **Noise suppression**

Plot the *spectra* of the unfiltered received signal rx_s_bb and the filtered received signal $rx\_s\_bb\_filtered$ and watch how the noise is suppressed. Any reflections on how the spectrum of $rx\_s\_bb\_filtered$ looks?

The rest of the script will search for the known training sequence by correlating the received and filtered $rx\_n\_bb\_filtered$ with $rx\_trainingsequence$. Much could be said about correlation but suffice it to say that what it basically does is to match every possible delayed version of the received signal with the training signal and sees where the best fit is. It will also tell us how much the signal was scaled and phase-shifted during transmission, which gives us the opportunity to scale back and derotate the signal. Sampling the rescaled and derotated filtered signal at the symbol rate will then produce the correlation metrics that your detectors will operate on.

Block eight will finally perform the detection and channel decoding. A few final lines compute the E_b/N_0 and the bit error rate (BER). Before you run the entire script there is one more thing to do.

Exercise 5 **Theoretical BER expressions**

Find or derive expressions for the theoretical bit error rates for your four channel code+modulation format combinations. Plot the BER curves versus E_b/N_0 and have them at hand when you do your measurements.

Now you are ready to start measuring.

Exercise 6 **BER simulations**

Remove the `return` command. Make sure all variables are set according to the modulators/encoders that you use. Save and run the script (in simulation mode). The script will produce a BER value and a value for E_b/N_0. Was the value of E_b/N_0 within your range of interest? Remember that since the number of bits is approximately 10000, the lowest possible BER is about 10^{-4}, so if you set the noise power too low, the BER value will be zero. Therefore, always set the noise power high enough that the BER is higher than 10^{-4}. Play around with the noise power (*Npow*) variable in the script to produce different E_b/N_0. Try to make an even spacing of the E_b/N_0 values (how much does E_b/N_0 change when you double the noise power?). Plot the BER/(E_b/N_0) pairs (as unconnected stars or circles) together with the theoretical curves (as solid lines). Theoretical values should match experimental values very well. Produce at least 10 points for each modulator/encoder combination. Remember to use logarithmic scale on the y axis (use the command `semilogy` for plotting), and also use the command `grid on` to make it easier to read the plots. Change the modulation/code combination and rerun the script until you acquire plots for all four combinations. Bring the plots to the lab room.

4 The laboration part 2 - The radio lab

When you arrive at the lab room (Ångström 7K1401), the instructor will discuss the questions in Section 3 and verify that your plots look okay. **The entire group needs to be present.** The task here consists of getting acquainted with the radio equipment and performing the same task as in Exercise 6, but now the *channel* variable should be set to 'soundcard'. Also, you only need to collect data and plot BER curves for *one* of the four modulator/coder combinations. The clever/lazy students will have inserted a `for` loop in their script, that automatically performs the measurements and collects the data points. Collect at least 10 BER/(E_b/N_0) pairs.

4.1 Laboration setup

The radio equipment consists of

- *a computer:* You run your code exactly as before but now you set the *channel* variable to 'soundcard'.
- *mixers:* The signal coming from the computers' sound card is first mixed up to 70 MHz, then up to about 1 GHz. The reverse is done for the received signal. The same 70 MHz and 930 MHz are used by the sender mixer and the receiver mixer. That way we don't have to worry about oscillator mismatch between sender and receiver. In a "real world" system, where the receiver and sender are far apart, we cannot use this approach. Instead we'll have to use e.g. a phase-locked loop (PLL).
- *a noise generator:* We have to add noise, otherwise the E_b/N_0 would be too high for our BER measurements.
- *antennas:* These are so called dipole antennas. The reception is best when the antennas are perpendicular to the distance between the antennas (connect an oscilloscope and try to turn them during a transmission).

We will also hook up a spectrum analyzer to see the spectra of the signals during transmissions.

5 The report

The report should contain

- short answers to the exercises in section 3.
- plots of your simulated measurements, four plots in total. Plot the theoretical curves in the same plots as their corresponding simulated curves.
- one plot from your radio measurements. Plot the theoretical curve in the same plot.

It should be possible to read and comprehend your report without having this instruction at hand. Therefore, make sure that you specify also the question and/or purpose when answering questions. Keep the report short and clean. Preferably use LaTeX. No handwriting.

Good luck!