Software-Defined Radio
Lab 2: Data modulation and transmission

Version 0.2

Anton Blad and Mikael Olofsson

May 11, 2011

1 Introduction

In this lab, you will build a transmitter and corresponding receiver for simple DBPSK modulated data framed with a correlation sequence in the beginning. The receiver prints the estimated SNR of the received data, as well as the computed bit error rate. You will then modify the transmitter to include an AWGN channel model and verify that the measured SNR and bit error rate correspond to the theoretical values. You will further modify the system to use DQPSK modulation instead of DBPSK.

2 Setup

The following hardware is needed for this lab:

- A computer with USB 2.0
- A USRP with an LFTX/LFRX daughterboard pair at the A side
- An oscilloscope with two channels

You need GNU Radio installed. The lab is tested and ensured to work with GNU Radio 3.3.0, and assumes that it is installed in `/usr/local/gnuradio-3.3.0`.

2.1 Initialization

If you are doing this lab in CommSys' research lab:

Log in to one of the lab computers as the user *SDR lab user*. Your lab teacher will give you the password. Your lab computer is prepared with all needed files in the directory

```
/home/sdrlabuser/labfolder/lab2
```

Open a terminal and cd to that directory. You do not need to worry about making changes to files there. That directory will be updated for the next occation.

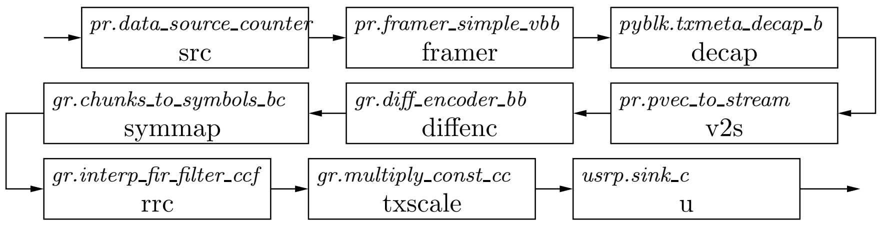

Figure 1: Simple transmitter for differential PSK modulated data.

If you are not following these instructions in CommSys' research lab:

Open a terminal. Create a directory `labfolder` somewhere and cd into it. If you have done Lab 1, you should already have that directory. Download and extract the accompanying lab files by executing the following commands:

```
$ wget http://www.commsys.isy.liu.se/SDR/labs/gr/lab2.tar.gz
$ tar zxf lab2.tar.gz
$ cd lab2
```

All files needed in this lab are now in the directory `lab2` that you just created.

3 Data transmission

3.1 Noise-free transmission link

The simple data transmitter is shown in Fig. 1. It consists of a data source generating a stream of data vectors that are all equal. Each data vector is framed with a simple correlation sequence at the beginning, then converted to a stream of bits. The stream of bits is packed to a number of bits per sample, corresponding to the chosen modulation, then differentially encoded, mapped to a complex symbol, and then passed through a root-raised-cosine shaping filter. The signal is finally modulated to the carrier frequency by the USRP.

On the receiver side, the structure is shown in Fig. 2. The baseband processing includes and AGC and matched filter followed by the `gr_mpsk_receiver_cc` block that does frequency correction and timing recovery. The output of the block contains complex symbol estimates. The symbol estimates are decoded and unpacked to a bit stream. A frame correlator searches for the correlation sequence, and frame sync samples the frame when the correlation sequence is found. The correlation sequence is removed, the header is stripped from the data vectors, and the resulting data stream is fed to a bit error rate computation block.

In order to provide reference data, a copy of the data generation chain in the transmitter is also included in the receiver. Also, the fftsink and constsink instances are used to visualize the spectrum and constellation of the received signal. Finally, the symbol estimates are fed to an SNR estimation block. The main loop of the program prints the estimated SNR and computed bit error rate regularly.

Use the files `psk_tx.py` and `psk_rx.py` as stubs for building the flowgraphs in Fig. 1 and Fig. 2, respectively. Design the transmitter and receiver according to the following requirements:

- Use a USRP transmitter interpolation rate of 128 and a receiver decimation rate of 64.

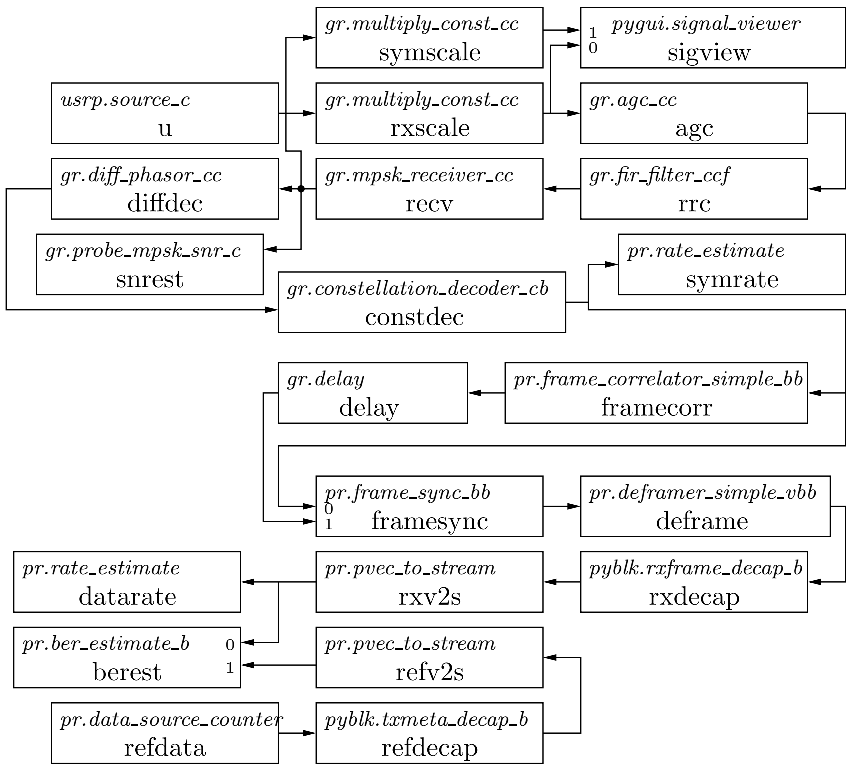

Figure 2: Simple receiver for differential PSK modulated data.

- The usable data rate shall be at least 180 kb/s. Remember that the correlation sequence adds some overhead.
- Use differential BPSK modulation.
- Use a correlation sequence of 64 bits (defined in the stub files).
- Use root-raised cosine filters with an excess bandwidth of 0.35.
- In order for the timing recovery to work, the number of samples per symbol must be at least 4.
- Use the following parameters for the mpsk_receiver block:
  - costas_alpha = 0.1
  - costas_beta = 0.25 * costas_alpha * costas_alpha
  - costas_fmin = -0.1
  - costas_fmax = 0.1
  - mm_mu = 0.5
  - mm_gain_mu = 0.01
  - mm_omega = samples_per_symbol
  - mm_gain_omega = 0.25 * mm_gain_mu * mm_gain_mu
  - mm_omega_rel = 0.005
- Adjust the number of required correct bits in the frame correlator such that the probability of a false positive correlation (assuming random data) is at most 10^{-6}, and the probability of a successful correlation is at least 99.9% for an SNR of 5 dB. You may use octave or matlab and the bundled `cseq.m` for this.

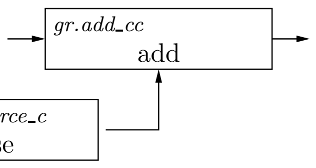

Figure 3: Simple model for AWGN channel.

To design the root-raised-cosine wave-shaping filters, use the function `gr.firdes.root_raised_cosine`. This function requires both the sampling frequency and the symbol rate, but depends in reality only on the relation between the two. For a given number of samples per symbol (*sps*), the parameters for the root-raised-cosine filters can be determined according to the following guidelines:

- gain = sps
- sampling_freq = sps
- symbol_rate = 1
- alpha = 0.35
- ntaps = 11*sps

Run the transmitter and receiver and verify that the transmission is flaw-less and that the rate corresponds to the expected.

3.2 Simulating a channel

GNU Radio provides complete channel models for both AWGN and different fading channels. However, in this lab, you will build your own channel model using a noise source and adder. Model an AWGN channel according to Fig. 3, and insert it before the shaping filter in the transmitter. This placement adds the distortion to the symbols rather than the samples (which would be more appropriate). However, it assures that all the added noise is in the same band as the signal, rather than spread out over the whole spectrum.

The parameter of the noise source can be set through the `set_amplitude()` function. This sets the standard deviation if Gaussian noise is specified. In your program, you should be able to specify the desired SNR of the channel (assuming a unit signal energy), and the noise source should be set accordingly.

3.3 Optional: Increasing the data rate

Increase the data rate to the double by changing the transmitter and receiver to use QPSK instead of BPSK.

4 Resources

- Online USRP FAQ:
  `http://gnuradio.org/redmine/wiki/gnuradio/UsrpFAQIntro`
- Firas Abbas Hamza, *The USRP under 1.5X Magnifying Lens!*:
  `http://gnuradio.org/redmine/attachments/129/USRP_Documentation.pdf`
  Third-party documentation of USRP.
- GNU Radio C++ block documentation:
  `file:///usr/local/gnuradio-3.3.0/share/doc/gnuradio-3.3.0/html/index.html`
- OpenRD C++ block documentation:
  `file:///usr/local/openrd/doc/index.html`
- Firas Abbas, *Simple Gnuradio User Manual*:
  `http://rapidshare.com/files/72169239/Simple-Gnuradio-User-Manual-v1.0.pdf`
  Third-party documentation of GNU Radio. This one is a bit out of date, but still useful. A printed copy is available at the lab desk.