ECOLE POLYTECHNIQUE FEDERALE DE LAUSANNE
School of Computer and Communication Sciences

Software-Defined Radio:
A Hands-On Course

Assignment date: Oct 5, 2016
Due date: Oct 12, 2016

Coded vs. Uncoded Image Transmission

In this assignment you will transmit an image over a simulated AWGN channel and assess the performance of convolutional coding vs. uncoded transmission. The following MATLAB functions may be useful: `imread`, `imshow`, `bitshift`, `reshape`, `poly2trellis`, `convenc` and `vitdec`.

Exercise 1. *Image reduction.* Implement the function `imgreduce` defined as follows:

```
% IMGREDUCE Reduces image size and grayscale resolution
%    Y = IMGREDUCE(X, G, B) reduces the grayscale image X by
%    downsampling the image by a factor G in each dimension and by
%    reducing the grayscale resolution to B bits per pixel.
%    X is an m x n matrix where m and n are the height and the width of
%    the image, respectively, measured in pixels. The entries of X are
%    values of type UINT8 (unsigned 8-bit integers) between 0 and 255,
%    where 0 means black and 255 means white; values in between stand
%    for intermediate shades of gray. G must be a positive integer, B
%    must be an integer between 1 and 8.
%    The output Y is a UINT8 matrix  obtained by taking every G-th point
%    of X (horizontally and vertically), starting with X(1,1), and scaling
%    the values to lie in the interval [0, 2^B-1] such that 0 corresponds
%    to black and 2^B-1 corresponds to white.
```

Note that `UINT8` variables in MATLAB behave similar to integer values in C or Java with respect to elementary arithmetic operations, in particular division.

You may test your function using the image file `elaine.512.tiff`, available on the course web site.

EXERCISE 2. *Serializing.* Implement the functions `img2bits` and `bits2img` defined as follows:

```
% IMG2BITS Converts an image to a sequence of bits
%    S = IMG2BITS(X, B) returns a row vector S that is obtained by
%    serializing the image matrix X column by column and then converting
%    each value to its binary representation. The conversion to bits is
%    'right-msb'. The number of bits per pixel is given by B; B must be
%    larger than or equal ceil(log2(M+1)), where M is the largest value
%    of X.


% BITS2IMG Converts a vector of bits to an image
%    X = BITS2IMG(S, B, M, N) returns an M x N matrix X whose entries
%    are of type UINT8 that is obtained by reconstructing the image from
%    the bit sequence S. B is the number of bits per pixel, and M and N
%    are the height and width of the image, respectively. The number of
%    elements in S must be equal to M*N*B.
```

You may use the functions `my_de2bi` and `my_bi2de` from your own communication toolbox, the functions `de2bi` and `bi2de` from the MATLAB communication toolbox, or the MATLAB functions `dec2bin` and `bin2dec` (but the latter two have the MSB on the left).

EXERCISE 3. Implement the function `transmit_bpsk` defined below:

```
% TRANSMIT_BPSK Simulate BPSK transmission across the AWGN channel
%    [S_EST, BER] = TRANSMIT_BPSK(S, Es_sigma2) simulates the
%    transmission of the bit sequence S over an additive white
%    Gaussian noise channel using BPSK modulation.
%    S is a row or column vector with the bits to be transmitted.
%    Es_sigma2 is the ratio, expressed in dB, of the energy per
%    symbol to noise variance at the output of the matched filter
%    of the receiver.
%
%    The function has two return values: S_EST is a vector
%    (same dimensions as input S) containing the bit sequence
%    estimated by the receiver; BER is the  bit error rate, i.e,
%    the fraction of bits that were received incorrectly.
```

You may assume the discrete symbol-time AWGN channel model, i.e., there is no need to convert symbols into samples (but you can if you wish). To simulate the Gaussian noise

you may use the `awgn` function from Matlab Communication Toolbox or, if not available, the standard Matlab function `randn`[1].

Exercise 4. Modify the `transmit_bpsk` function to include channel coding using a convolutional encoder in the transmitter and a Viterbi decoder in the receiver. The new function should behave according to the following interface:

```
% TRANSMIT_CODED_BPSK Simulate convolutionally encoded BPSK
%    transmission across the AWGN channel with Viterbi decoding
%
%    [S_EST, BER] = TRANSMIT_CODED_BPSK(S, Es_sigma2)
%    simulates the transmission of the bit sequence S
%    across the  additive white Gaussian noise channel
%    using convolutional coding and BPSK modulation.
%    The convolutional code used has rate 1/2 and octal generators [5 7].
%    A Viterbi decoder with soft decisions is used at the receiver.
%    S is a row or column vector with the bits to be transmitted.
%    Es_sigma2 is the ratio, expressed in dB, of the energy per
%    symbol to noise variance at the output of the matched
%    filter of the receiver.
%
%    The function has two return values:
%    S_EST is a vector containing the bit sequence estimated
%    by the receiver  (same dimensions as input S).
%    BER is the bit error rate, i.e., the fraction of the
%    transmitted bits that were incorrectly decoded.
```

To learn how we can represent a convolutional code in Matlab, search for *Polynomial description of a convolutional encoder* in Matlab's documentation. You will also need the functions `poly2trellis` and `convenc`. Notice that the constraint length of this code is 3 and the rate 1/2, see Fig. 1.

For the Viterbi decoder, use Matlab function `vitdec`. Use a value of 15 for the `TBLEN` parameter (this "traceback" length determines after how many backward steps through the trellis the decoder makes a final decision. This method is suboptimal, since the whole trellis is not traversed, but it allows one to start decoding before all the bits have arrived at the receiver). For the other options, have a look at the examples section in the Matlab help.

Exercise 5. Now we will put things together and evaluate the performance of the convolutional encoder. Get the image file `elaine.512.tiff`[2] and implement a function `test_convenc` according to the following header:

[1] You will need to multiply the output of `randn` by an appropriate factor so that the resulting variance is in accordance with the desired value of E_s/σ^2.

[2] Available on Moodle

```
% [UNCODED_BER, CODED_BER] = TEST_CONVENC(IMAGE_FILE, Es_sigma2)
% TEST_CONVENC compares the BER and quality of the transmission
% of image IMAGE_FILE over the AWGN with and without convolutional
% coding. The code used has octal generators [5 7].
% BPSK modulation is used for the transmission.
% Es_sigma2 specifies, in dB, the ratio  of the energy per symbol
% to noise variance at the output of the matched filter of the
% receiver.
%
% The function returns the bit error rate for the uncoded scheme
% as first output, and the bit error rate for the coded scheme
% as second output
```

Your function should execute the following steps:

(a) Load the image file into MATLAB.[3]
(b) Using `imgreduce`, downsample the image by a factor 4 and reduce the grayscale resolution to require 6 bits per pixel.
(c) Convert the image to bits.
(d) Using the functions `transmit_bpsk` and `transmit_coded_bpsk` from Exercises 3 and 4, simulate the transmission of both the uncoded and the encoded bit sequence across an AWGN channel.
(e) Display the original image and the results of both the uncoded and the coded transmissions.[4] Find a value of E_s/σ^2 for which uncoded transmission has errors while coded transmission is error free.

Exercise 6. To have a more precise idea of the error correcting capabilities of the convolutional code used throughtout this assignment, we provide the function `err_rates_coded_uncoded.m`, available on Moodle. If run without any arguments, this function plots the BER vs E_s/σ^2 curves for the uncoded and coded BPSK transmission systems. It plots and compares theoretical results with the results obtained from simulation (calling the functions `transmit_bpsk` and `transmit_coded_bpsk`). If your implementation of these functions is correct, then `err_rates_coded_uncoded.m` will run without errors and you should obtain a result similar to that of Figure 2. If the simulated points are too far from the theoretical/reference results, then you can suspect that there is some mistake in your implementation[5].

In order to help you with the implementation, we provide on the course website the compiled versions of the functions you have to write.

[3] Use the MATLAB function `imread`
[4] Use the MATLAB function `imshow`.
[5] For low values of the BER (below 10^{-5}), the distance between the simulated points and the reference results will be larger; this is normal, the precision of the simulation results is limited by the number of transmitted bits, 10^6 by default.

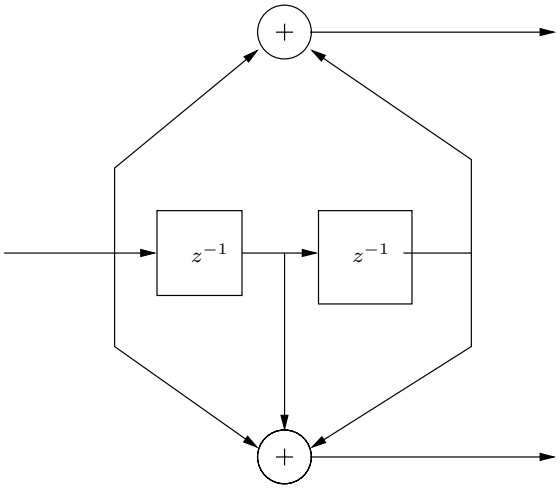

Figure 1: Rate 1/2 Convolutional Encoder

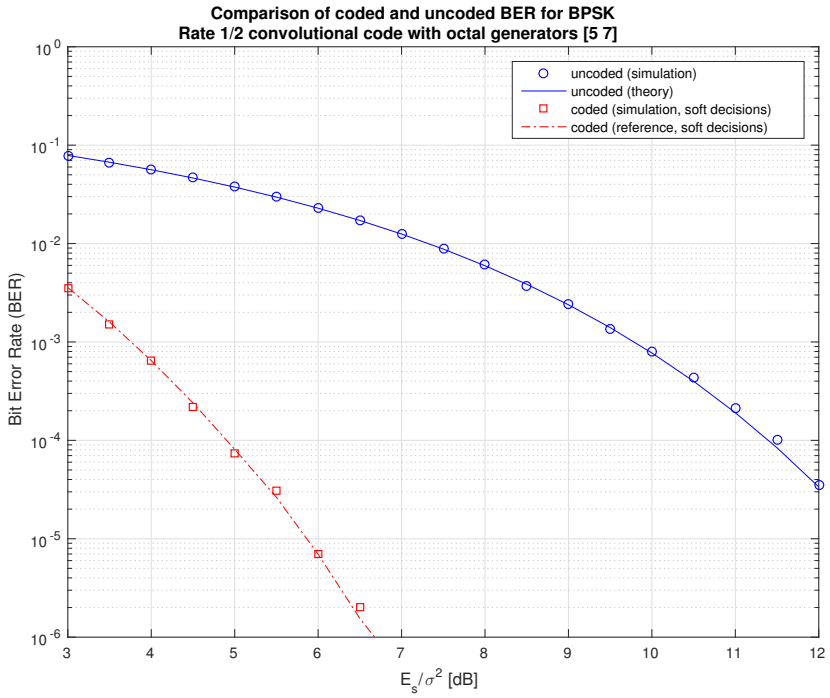

Figure 2: Uncoded and coded bit error rates in BPSK. The coded system uses convolutional coding with rate 1/2, octal generators [5 7] and soft Viterbi decoding.