Convolutional codes and Viterbi decoding.

Wednesday, May 2, 2007. Due by: 5 p.m., Monday, May 7, 2007

I. Note

The deliverable of this project is in form of a written report, containing the answers to the questions given in this document in the form you consider more appropriate. You are invited, but not obliged, to use C programming for the simulations of convolutional codes and Matlab for computing the bounds and plotting the results. Your work in individual and you are supposed to carry out the project based on your own knowledge and capabilities.

The report has to be sent to my e-mail account laura.cottatellucci@eurecom.fr

II. Description of the project

Consider the communication system shown in Figure 1.

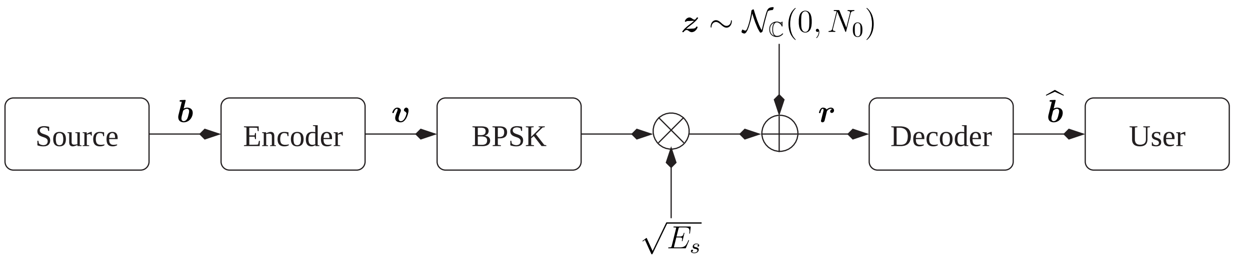

Fig. 1. Communication system

A source generates sequences of $K = 128$ information bits $b[i]$ which are encoded with a binary convolutional code. The parameters of the convolutional codes are defined in Table I.

The encoded output sequence $\boldsymbol{v}[i]$, with[1] $i = 0, 1, \ldots, (K + \nu_{\mathrm{MAX}} - 1)$, is sent to a BPSK modulator with mapping rule $\mu : \{0, 1\} \to \{-1, +1\}$ and then transmitted through an AWGN channel with power spectral density N_0. Let $\boldsymbol{r}[i] = \sqrt{E_s}\mu(\boldsymbol{v}[i]) + \boldsymbol{z}[i]$ be the received sequence where $\boldsymbol{z}[i] = [z_1[i], z_2[i], \ldots z_n[i]]$ is the noise with $z_j[i], j = 1, \ldots n$, i.i.d. and $z_j[i] \sim \mathcal{N}(0, N_0)$. The received sequence $\boldsymbol{r}$ is decoded by a Viterbi algorithm. The decoder provides the user with the decoded information sequence $\widehat{\boldsymbol{b}}$.

[1] n and ν_{MAX} depend on the specific code. ν_{MAX} is due to the trellis termination by transmission of additional ν_{MAX} information bits $b[i] = 0,\ i = K - 1, \ldots, (K + \nu_{\mathrm{MAX}} - 1)$

Name	Memory	Rate	Generators
	ν_ν	$1/n$	(octal notation)
Abu Alghanam	2	1/4	(5,5,7,7)
Berotoglio	2	1/3	(5,7,7)
Beyaztas	3	1/4	(13,13,15,17)
Clavier	3	1/3	(13, 15,17)
Galardini	2	1/2	(5,7)
Gorgoglione	3	1/2	(13,17)
Hdiji	3	1/2	(17,15)
Medina Perlaza	2	1/4	(7,5,7,5)
Melis	2	1/3	(7,5,7)
Murangira	3	1/4	(13,17,15,13)
Ogboh	3	1/3	(17,15,13)
Omar	2	1/2	(3,5)
Quirion	3	1/2	(17,15)
Rout	2	1/4	(5,7,7,5)
Tanninen	3	1/4	(17,13,13,15)
Villegas Ramos	3	1/3	(15,17,13)
Zhang	3	1/4	(17,15,13,13)

TABLE I

CODE PARAMETERS

The project task consists of:

- Write the state diagram of the convolutional code.
- Write system simulator.
- Implement the ML decoder based on the Viterbi algorithm.
- Plot the simulated BER and WER versus[2] $\frac{E_b}{N_0}$ (expressed in dB). The WER must be computed for codewords corresponding to $K = 128$ plus the ν_{MAX} zero bits for trellis termination.
- Compute the bounds on the BER via the symbolic transfer function method and compare on the same chart the BER obtained by simulations with the computed bounds.

[2]Not $\frac{E_s}{N_0}$!

III. Hints

A. The simulator

Part of the simulator is provided in the package included to this project description. Copy the package in your local working directory and uncompress it with the following command:

```
unzip tpcc_2007.zip
```

The package consists of a set of .c files:

- `tp_random.c` contains libraries for random number generation.
- `tp_alloc.c` contains a useful function for handling memory allocation.
- `tp_convolutional.c` contains the functions concerning convolutional codes. Some of the functions contained in this file with their interfaces have been declared but they are empty. Your work consists in filling each function with the required C code.
- `tp_main.c` is the simulator main file. It contains the main() and some other functions (e.g. the BPSK modulator and the error counter). In some of these files small piece of C code must be added.

Additional information concerning the simulator and the data structure are provided as comments in the source code.

B. Convolutional Codes

A convolutional code is characterized by the parameters $(n, k, \boldsymbol{G}(D))$ where $\frac{k}{n}$ is the code rate and $\boldsymbol{G}$ is the generator matrix.

In order to simplify the simulator we only consider codes with $k = 1$, i.e., one information bit enters in the encoder for each trellis step. Therefore the generator matrix consists of a single row of impulse responses (generators) $(g_1(D), g_2(D), \ldots, g_n(D))$. The code generator matrix expressed in octal format is assigned in Table 1.

The function `Initialize_convolutional_code()` converts the generators into the more useful binary format (code provided). The generators fully determine the structure of the convolutional code, the trellis structure, and the trellis transition labels. However, it is more convenient to represent the trellis by using the Forward and Backward matrix data structures (already declared in the provided C code). These matrices are filled by a function contained in the source code according to the following convention (refer to Figure 2)

- The state of the convolutional code is the content of its memory, e.g. a code with memory 2 has 4 states that can be represented either by the binary labels $\{00, 01, 10, 11\}$ or by the decimal labels $\{0, 1, 2, 3\}$. In the simulator use the decimal representation to indicate a certain state s.

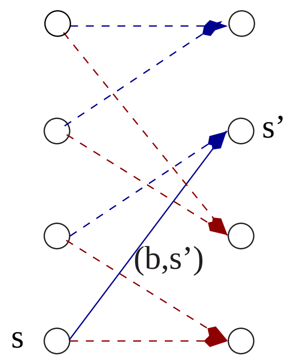

Fig. 2. Trellis

- When a bit enter in the encoder, it generates a transition and a set of n output bits, one bit per generator. Write the output bits using the notation $(v_n, \ldots, v_1)$ where v_i is the output generated by the generator $g_i(D)$. Refer to the output using labels in decimal format. This will be useful when indexing a matrix (e.g. if $n = 3$ and the output is $(v_3, v_2, v_1) = (1, 0, 1)$ then the decimal index of the transition output is $\boldsymbol{v} = 5.$)
- After the transition the encoder reaches the state s'.

1) Forward Matrix: Let S denote the number of states. The forward matrix $\boldsymbol{F}$ is a $S \times 2 \times 2$ matrix of integers. It describes the trellis when moving form the left to the right. Its contents is described as follows:

$$\boldsymbol{F}(s, b, 0) = s'$$
$$\boldsymbol{F}(s, b, 1) = v$$

for $b \in (0, 1)$ and $s \in \{0, 1, \ldots, S - 1\}$.

2) Backward Matrix: The Backward matrix $\boldsymbol{B}$ is a $S \times 2 \times 3$ matrix of integers. It describes the trellis when moving from the right to the left. Its contents is described as follows:

$$\boldsymbol{B}(s', t, 0) = s$$
$$\boldsymbol{B}(s', t, 1) = v$$
$$\boldsymbol{B}(s', t, 2) = b$$

where $t \in (0, 1)$ is the transition index (information bit that determine the transition (s, s')) and $s' \in \{0, 1, \ldots S - 1\}$.

3) Encoder: The encoder processes blocks (or frames) of length K sources per bit time. By default the initial state of the encoder is 0. Remember to append a frame of ν_{MAX} bits equal to zero to the block of K bits, so that the whole frame has length $K + \nu_{\text{MAX}}$. This will terminate the trellis in the state 0.

4) Decoder: Implement the Viterbi decoder. As a hint you can use the following data structures:

- Two vectors $\boldsymbol{a}^0$ and $\boldsymbol{a}^1$ of S elements, in which the path metrics are stored.
- Two arrays $\boldsymbol{Z}^0, \boldsymbol{Z}^1$ of size $S \times (K + \nu_{\mathrm{MAX}})$ needed for the trace-back recursion.

Implement the Viterbi Algorithm according to the following pseudo code: let $\boldsymbol{r} = (r_0, r_1, \ldots r_{n(K+\nu_{\mathrm{MAX}}-1)})$ denote the received signal sequence from the AWGN channel.

```
**Path metric recursion**
Let $a^0[0] = 0, a^0[s] = -\infty, \forall s = 1, \ldots, S-1$, where $\infty$ denotes a very large number (e.g. $10^{10}$).
For $i = 0, \ldots, K + \nu_{\mathrm{MAX}} - 1$
      Compute the values of the $i$-th branch metric, given by
      $$\omega_i(v) = \sum_{\ell=0}^{n-1} r_{ni+\ell}\mu(v_\ell)$$
      for $v = 0, \ldots, 2^n - 1$ ($(v_{n-1}, v_{n-2}, \ldots, v_0)$ is the binary representation of the integer $v$ and $\mu : \{0,1\} \to \{+1,-1\}$ is the usual binary antipodal modulation mapping).
      let $a = -\infty$
      for $s' = 0, \ldots, S-1$
            let $\widehat{t} = \mathrm{argmax}_{t=0,1}\{\omega_i(\boldsymbol{B}(s',t,1)) + \boldsymbol{a}^0[\boldsymbol{B}(s',t,0)]\}$
            let $\boldsymbol{a}^1[s'] = \omega_i(\boldsymbol{B}(s',\widehat{t},1)) + \boldsymbol{a}^0[\boldsymbol{B}(s',\widehat{t},0)]$
            let $\boldsymbol{Z}^0[s',i] = \boldsymbol{B}(s',\widehat{t},0)$
            let $\boldsymbol{Z}^1[s',i] = \boldsymbol{B}(s',\widehat{t},2)$
            let $a = \max\{a, \boldsymbol{a}^1[s']\}$
      end loop
      for $s' = 0, \ldots, S-1$
            let $\boldsymbol{a}^0[s'] = \boldsymbol{a}^1[s'] - a$
      end loop
end loop
**Trace-back recursion**
let $s = 0$
for $i = K + \nu - 1, \ldots 0$
      let $\widehat{b}(i) = \boldsymbol{Z}^1[s,i]$
      let $s = \boldsymbol{Z}^0[s,i]$
end loop
```

The ML decoded information bit sequence is given by $(\widehat{b}_0, \widehat{b}_1, \ldots, \widehat{b}_{K+\nu_{\mathrm{MAX}-1}})$ where the last ν_{MAX} must be equal to zero (since we are forced the encoder to terminate all paths in the state zero) and have to be discarded from the total error count (i.e., only the first K information bits have to be considered in the computation of the BER for each simulated block of information).

C. *Other Hints*

- Read the parameters of the convolutional code in Table I, then set the corresponding variables in the C code.
- Write the Viterbi decoder and check its performance first of all without adding noise. The number of error after decoding must be zero.
- Plot your curves versus $\frac{E_b}{N_0}$ in dB scale. The noise is generated with variance $N_0 = 1$. Then, in order to vary $\frac{E_b}{N_0}$ you must vary the symbol energy E_s, i.e. the transmitted symbols are $\pm\sqrt{E_s}$. Recall the relation between E_b and E_s (that depends on the coding rate) in order to map the desired value of $\frac{E_b}{N_0}$ into the value of SNR $\frac{E_s}{N_0}$ that will be used to scale the transmitted symbols.
- BER and WER can be determined at the same time, by generating many frames of $K = 128$ information bits (plus the trellis termination bits). For every value of $\frac{E_b}{N_0}$ many frames need to be transmitted in the simulation. Bits error and word errors are counted. A frame is in error if one or more information bits are in error after decoding.
- When running simulations, consider as reliable results only results obtained with a sufficient number of errors. A good value is between 100 and 200 bit errors. If the actual BER of the system is 10^{-4} it is necessary to transmit at least 10^6 information bits, that is, roughly 10000 frames of length 128 information bits. In order to save time simulation a loop exit condition is needed. Set the maximum number of transmitted frames to a very large number (e.g. 10^7). After decoding each frame, test the number of counted bit errors and word error accumulated at that point. If both are larger than a threshold, e.g. 100, the condition to exit from the simulation loop is satisfied and the the BER and WER can be determined.

IV. Numerical Computation of the Bit Error Probability Bound

In order to compute the upper bound on the bit error probability $P_b(E)$ it is necessary to determine the IOWEF $A(W, D)$. Let us adopt the same approach as in the course handouts. Let $\boldsymbol{x}(W, D)$ be the vector of the nonzero states and $\boldsymbol{x}_0(W, D)$ be the vector of the inputs. As shown in the handout it is possible to write the linear system

$$\boldsymbol{x}(W, D) = \boldsymbol{T}(W, D)\boldsymbol{x}(W, D) + \boldsymbol{x}_0(W, D)$$

and determine the nonzero states as solution of the system. The output-zero state can be reached from the nonzero states through the relation $\boldsymbol{C}(W, D)\boldsymbol{x}(W, D)$ where $\boldsymbol{C}(W, D)$ is row vector whose j-th element is the label of the transition in the modified graph that brings from the j-th nonzero state to the output zero state. Then,

$$A(W,D)=\boldsymbol{C}(W,D)(\boldsymbol{I}-\boldsymbol{T}(W,D))^{-1}\boldsymbol{x}_0.$$

The bit error probabilities $P_b(E)$ is bounded by

$$P_b(E)\leq\frac{1}{k}\frac{\partial}{\partial D}A(D,W)|_{W=\mathrm{e}^{-\frac{RE_b}{N_0}},D=1}$$

In order to calculate the union bound avoiding the symbolic inversion of the matrix $(\boldsymbol{I}-\boldsymbol{T}(W,D))$ we can approximate the derivative of $A(D,W)$ with respect to D in $D=1$ by the incremental ratio

$$\frac{\partial}{\partial D}A(W,D)|_{W=\mathrm{e}^{-\frac{RE_b}{N_0}},D=1}\approx\frac{1}{\epsilon}\left(A(1+\epsilon,\mathrm{e}^{-\frac{RE_b}{N_0}})-A(1,\mathrm{e}^{-\frac{RE_b}{N_0}})\right)$$

where ϵ is a small number (e.g. $\epsilon=0.01$).

In any case, either if $\frac{\partial}{\partial D}A(W,D)|_{W=\mathrm{e}^{-\frac{RE_b}{N_0}},D=1}$ is evaluated in a symbolic form or by the above numerical method, since $A(W,D)$ is a power-series, it is meaningful only when it is evaluated inside the region of convergence. It turns out that for $\frac{E_b}{N_0}$ smaller than a certain threshold the bound on $P_b(E)$ assumes values which may be greater than 1 or negative. In these case we are outside the convergence domain and we shall replace the bound on the BER by the value $\frac{1}{2}$ which is the trivial bound obtained by deciding randomly on every information bit by disregarding the decoder outcome.