Convolutional Coding and the Viterbi Algorithm

SSY125 Project*

1 Introduction

1.1 Project Overview

The goal of this project is to provide you with a deeper understanding of how channel coding can be used to improve the performance of a digital communication system. Your main task is to implement, evaluate, and compare several options for different encoders and modulation formats. In order to successfully complete this task, you have to program certain algorithms (like the Viterbi algorithm), as well as answer specific questions with the help of concepts introduced in the lectures (like channel capacity).
The deliverables of the project are split into two parts. For the first part (due **Friday, December 8, 2023, 11:59pm**), you have to submit the source code of your implementation of the Viterbi algorithm together with a figure showing the requested simulation results. For the second part (due **Friday, January 5, 2024, 11:59pm**), the source code and a final report have to be handed in. The report should document all your answers and simulation results, as well as a detailed discussion thereof. For more details, see Section 3 below.

1.2 Learning Outcomes

After completing this project, you should be able to

- successfully set up and run a simulation environment efficiently in MATLAB,
- understand why and how MATLAB can be used to simulate the performance of a system that eventually transmits and receives analog waveforms,
- use the concept of channel capacity as a benchmarking tool,
- encode a bit stream with a convolutional encoder and understand the relationship between the encoder and a trellis diagram,
- implement the Viterbi algorithm and understand how it works,
- explain and quantify the performance difference between hard and soft decoding, and
- analyze fundamental trade-offs between power, bit rate, bandwidth, and bit-error rate in a coded digital communication system and in particular make fair comparisons between uncoded and coded systems.

*Lecturer: Prof. Alexandre Graell i Amat (e-mail: alexandre.graell@chalmers.se). Teaching assistant: Mohammad Farsi , (e-mail: farsim@chalmers.se).

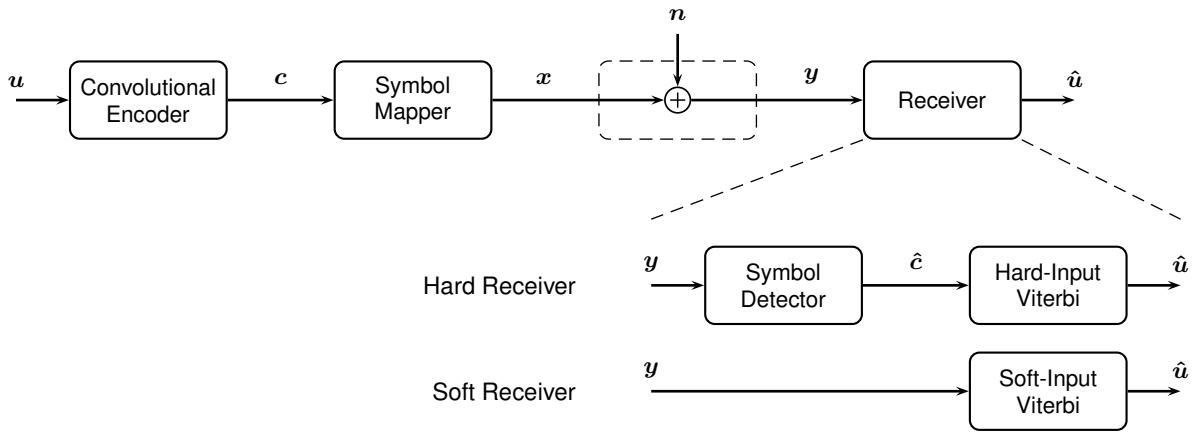


Figure 1: Block diagram of the system model including the two receiver types that are used in this project.

1.3 A Note on the Oral Exam

On Friday, January 12, 2024, there will also be a short oral examination based on the content of the project. In order to give you a feeling for what to expect in this oral examination, we include several *example questions* in the following description. These questions may also be helpful for you to check if you have understood all the concepts needed to implement the entire simulation environment. However, the example questions are not meant to be answered in the report!

2 System Overview

In Fig. 1, a block diagram of the considered system model is shown. The system consists of a convolutional encoder, a symbol mapper, an additive noise channel and a receiver (both hard and soft). In the following, we describe the individual blocks in more detail. In your report, please *do not* repeat the information given in this section.

Example Questions: What are the lengths of each of the vectors shown in the block diagram? Are the elements discrete, continuous, complex, ...? How does one measure the performance of such a system?

2.1 Convolutional Encoder

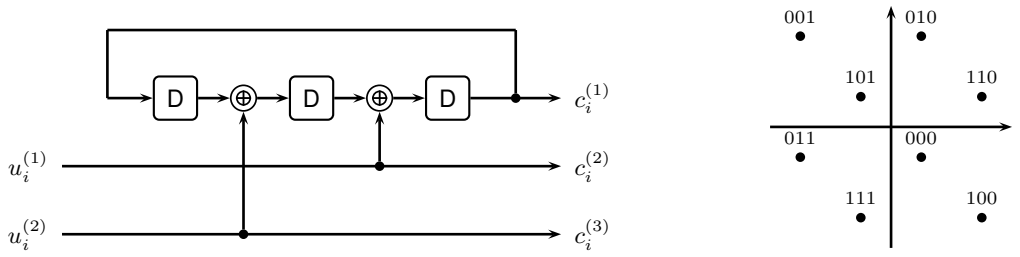
This block encodes the random bit vector $\boldsymbol{u}$ into a (longer) bit vector $\boldsymbol{c}$. We assume that the convolutional encoder is not zero-terminated. In this project, we use the following different encoders:

- $\mathcal{E}_1$: Rate-1/2 convolutional encoder, generator polynomial $\boldsymbol{G} = (1 + D^2, 1 + D + D^2)$
- $\mathcal{E}_2$: Rate-1/2 convolutional encoder, generator polynomial $\boldsymbol{G} = (1 + D^2 + D^3 + D^4, 1 + D^2 + D^3)$
- $\mathcal{E}_3$: Rate-1/2 convolutional encoder, generator polynomial $\boldsymbol{G} = (1 + D^3 + D^4, 1 + D + D^3 + D^4)$
- $\mathcal{E}_4$: Rate-2/3 convolutional encoder, shown in Fig. 2 (a)

Example Questions: Are the encoders systematic? Can you draw a trellis diagram corresponding to a given encoder? How long is the vector $\boldsymbol{c}$ for the three encoders?

2.2 Symbol Mapper

The mapper takes the encoded bit sequence $\boldsymbol{c}$ and outputs a sequence of symbols $\boldsymbol{x}$. Each symbol is taken from particular constellation, and the relationship between bits and symbols is given according to a specific labeling or symbol mapping. In this project, we use the following constellations and mappings:



(a) Rate-2/3 convolutional encoder $\mathcal{E}_4$ *(b)* Const. $\mathcal{X}_{\mathsf{AMPM}}$

Figure 2: The encoder $\mathcal{E}_4$ and constellation $\mathcal{X}_{\mathsf{AMPM}}$. The symbol mapping is done according to $L_i \rightarrow x_i \in \mathcal{X}_{\mathsf{AMPM}}$, where $L_i = (c_i^{(1)} c_i^{(2)} c_i^{(3)})$ corresponds to the label shown next to the constellation points. For example: $(010) \rightarrow a + j \cdot 3a$, or $(110) \rightarrow 3a + j \cdot a$, where a is a constant to adjust the energy of the constellation.

- $\mathcal{X}_{\mathsf{BPSK}}$: BPSK with mapping $0 \rightarrow 1$ and $1 \rightarrow -1$
- $\mathcal{X}_{\mathsf{QPSK}}$: QPSK with a Gray mapping
- $\mathcal{X}_{\mathsf{AMPM}}$: AMPM with a predefined bit mapping, as shown in Fig. 2 (b)

Example Questions: What is the average energy per symbol or per information bit? What is the minimum distance normalized by the average energy for each constellation? Can you draw the optimal symbol decision regions assuming additive white Gaussian noise?

2.3 Additive Gaussian Noise Channel

The channel adds i.i.d. complex Gaussian noise to each component in the transmitted symbol vector $\boldsymbol{x}$. That is, let x_i be the ith transmitted symbol, then y_i is given by $y_i = x_i + n_i$, where $n_i = n_i^{\mathsf{Re}} + j \cdot n_i^{\mathsf{Im}}$, and both noise components $n_i^{\mathsf{Re}}, n_i^{\mathsf{Im}}$ are real Gaussian random variables with zero mean and a certain variance σ^2.

Example Questions: Which components usually follow after the symbol mapper in a real communication system? Why can we assume that they are not there? Does that have any implications for the obtained performance results with respect to a real communication system? What is the relationship between σ^2 and $\mathsf{N_0}$ (the one-sided power spectral density of an additive white Gaussian noise (AWGN) channel)?

2.4 Hard Receiver

This type of receiver performs decoding in two steps. First, a symbol detector tries to recover the transmitted symbols and outputs a vector of detected (coded) bits $\hat{\boldsymbol{c}}$. This vector is then passed to a hard-input Viterbi decoder, which outputs an estimate of the transmitted bit vector.

Example Questions: Why is this receiver structure not optimal? Can you think of another way to pass information about the symbols/coded bits from the symbol detector to the decoder?

2.5 Soft Receiver

The soft receiver utilizes the channel output directly and passes it to the soft-input Viterbi decoder. Again, the decoder tries to recover the transmitted bit vector and outputs an estimate $\hat{\boldsymbol{u}}$.

Example Questions: What is the cost function that the soft-input Viterbi decoder minimizes? Is this receiver structure optimal? What is the difference between the hard-input Viterbi and soft-input Viterbi decoder from an implementation point-of-view?

3 Tasks

Both deliverables should be uploaded in Canvas. One submission per group is sufficient. *Important: It is not allowed to use any MATLAB function in the Communication Systems Toolbox except for* `bi2de`, `de2bi` *and* `qfunc`. *All algorithms have to be implemented by yourselves.*

3.1 Part I - Due Friday, December 8, 2023, 11:59pm

Using $\mathcal{E}_1$ and $\mathcal{X}_{\mathsf{QPSK}}$ with the Gray mapping, evaluate the performance of the system for uncoded and coded transmission. For coded transmission, consider only the hard receiver. Also plot the theoretical BER for the uncoded system.

For this task, you are required to hand in *one BER vs* $\mathsf{E_b/N_0}$ *plot* showing your simulation results in order for us to see that the simulation environment and Viterbi algorithm are working properly. *Only show results for BER between* 10^{-4} *and* 1 (command `ylim([1e-4 1])`). For an example of how such a plot may look like, see Fig. 3, but note that the red curve in that plot corresponds to a different coded system than what you will use in this task.
You are also required to hand in *all* your MATLAB code. Make sure that it can be easily executed by the TAs (e.g., have a file called *main.m* or include a short list of instructions how to generate your curves). Code that does not run does not give points.

Hint: It is convenient to make the Viterbi algorithm general to account for any trellis structure. As a model, the MATLAB way to specify a trellis structure can be used (see documentation for `poly2trellis`).

3.2 Part II - Due Friday, January 5, 2024, 11:59pm

The second part consists of the following subproblems.

(1) Hard vs. Soft Receiver (3 Points)

- Using $\mathcal{E}_2$ and $\mathcal{X}_{\mathsf{QPSK}}$ with the Gray mapping, evaluate the performance of the system for uncoded and coded transmission. For coded transmission, consider both the hard and soft receiver. For the coded system with the *soft* receiver, also plot an upper bound on the theoretical BER. What is the coding gain at a BER of 10^{-4}? What is the asymptotic coding gain? Why is there a difference between hard and soft decoding and how much is it? What is the minimum $\mathsf{E_b/N_0}$ for reliable communication of a system operating at the same spectral efficiency as this system?

(2) Encoder Comparison (5 Points)

- Compare the performance of the three encoders $\mathcal{E}_1$, $\mathcal{E}_2$, $\mathcal{E}_3$ for $\mathcal{X}_{\mathsf{QPSK}}$ with the Gray mapping. Consider only the soft receiver. For all encoders, also plot an upper bound on the theoretical BER. What is the coding gain for the three codes at a BER of 10^{-4}? What is the asymptotic coding gain? Explain the performance differences between the three coded systems and also comment on the system complexity.

(3) Coding can Increase Efficiency (7 Points)

- Make a fair comparison between the following three systems in terms of spectral efficiency, power efficiency, and bit error rate. In all cases, evaluate both the performance for uncoded and coded transmission. For coded transmission, consider *only* the soft receiver.
  - System 1: $\mathcal{E}_3$ and $\mathcal{X}_{\mathsf{BPSK}}$
  - System 2: $\mathcal{E}_3$ and $\mathcal{X}_{\mathsf{QPSK}}$ with the Gray mapping
  - System 3: $\mathcal{E}_4$ and $\mathcal{X}_{\mathsf{AMPM}}$

Is any of the systems strictly better than another one? How far away are these systems from capacity? In particular comment on the differences between system 1 and system 2.

For all subproblems, you are also required to hand in *all* your MATLAB code. Make sure that it can be easily executed by the TAs (e.g., have a file called *main.m* or include a short list of instructions how to generate your curves). Code that does not run does not give points.

4 Report Guidelines

For part II, you have to hand in a project report, in which you show us that you have understood and solved all of the above subproblems (1)–(3). Here are some guidelines and recommendations on how to write your report:

- Focus on obtained results and the questions in the tasks. Do not review details about algorithms.
- *All* results have to be discussed. Plotting curves without a discussion will not give you any points.
- To present your results, you are limited to *one BER vs.* $\mathsf{E_b/N_0}$ *plot per subproblem*, that is, three such plots in total. For an example of how such a plot may look like, see Fig. 3. (You may of course use other figures if you wish to illustrate something.)
- As your x-axis, use $\mathsf{E_b/N_0}$ in [dB] consistently in your entire report!
- For this project, you should only show results between BER of 10^{-4} and 1 (command `ylim([1e-4 1])`). As a rule of thumb, you have to evaluate more than 100 error events in order to get statistically significant results. That means you have to transmit at least $100/10^{-4} = 10^6$ bits. *Never* plot simulation results that are not statistically significant.
- Place figures alongside your text, in the top or bottom part of the page. Do not place all figures in an appendix or in the beginning.
- The report should not exceed 6 pages (including all figures).
- Keep in mind that the actual writing of the report and the simulations take time! In particular, part I of the project should give you a good idea about how much time you need to finish all the simulations in part II.
- LaTeX is highly recommended for the report.
- The report should be written according to the rules contained in [1].

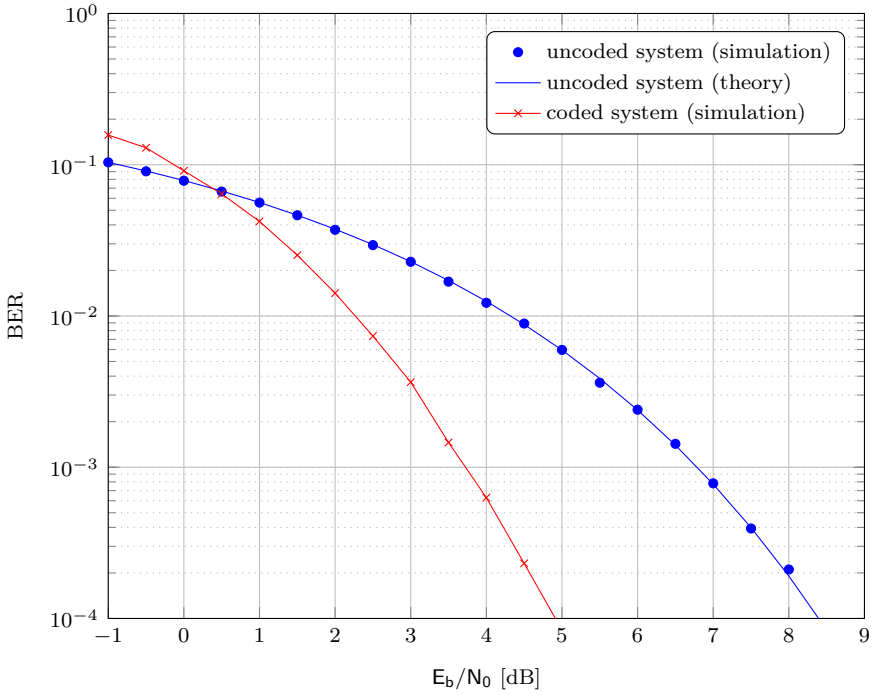


Figure 3: A typical BER vs. $\mathsf{E_b}/\mathsf{N_0}$ plot. Note that you only need to simulate down to a BER of 10^{-4} for this project.

5 Note on the Upper Bound

You are required to plot an upper bound on the theoretical BER for the coded and uncoded system. For the coded system, *this is only required for the soft receiver*, since theoretical performance evaluation for the hard receiver is not part of the lecture content.
For the soft receiver, the necessary background is developed in the lecture notes. In general, it is shown that the bit error probability can be upper bounded by

$$P_{\mathsf{b}}^{\mathsf{SOFT}} \leq \frac{1}{K} \sum_{d=d_{\mathsf{min}}}^{N} \sum_{w=1}^{K} w A_{w,d} \mathsf{Q}\left(\sqrt{\frac{2dR_{\mathsf{c}}\mathsf{E_b}}{\mathsf{N_0}}}\right),$$

where all variables are defined in the lecture notes. To make your life easier, we remark here that the above inequality can also be written as

$$P_{\mathsf{b}}^{\mathsf{SOFT}} \leq \sum_{d=d_{\mathsf{min}}}^{N} \tilde{A}_d \mathsf{Q}\left(\sqrt{\frac{2dR_{\mathsf{c}}\mathsf{E_b}}{\mathsf{N_0}}}\right).$$

Moreover, you probably have to truncate the above sum to a reasonable number of terms. One last hint: you are allowed to use the `distspec` function in MATLAB to help with the evaluation of the bound.

6 Matlab

6.1 Built-in MATLAB Functions

MATLAB has several functions that may be useful in this project. A short list of them follows below. The `help` command is perhaps the most useful of all commands. When still in doubt after reading the help files, you can do as the pros: *use the force, read the source.* The command `which cmd` gives the path to the m-file that implements the command `cmd`, and `open cmd` opens the source of the m-file in the editor window.

- `bitxor`, `bitand`, `bitshift` Bit-wise operations on MATLAB floating-point matrices

- `find` Find indices of nonzero elements of a vector.
- `save load` Save and load data in .mat format.
- `size` To get size of an array.
- `ind2sub` Multiple subscripts from linear index.
- `zeros(N,M)` Generate an N-by-M matrix of zeros.
- `ones(N,M)` Generate an N-by-M matrix of ones.

6.2 Simulation Chain Skeleton

In order to produce smooth BER curves, we provide you with an adaptive simulation chain that you can use as a starting point for your implementation. It is contained in the file `project_plain.m`, which you can find on Canvas.

6.3 Efficient Coding in MATLAB

The simulation time can be drastically decreased by writing efficient MATLAB code. One way to do this is to vectorize as many computations as possible. Generating and encoding bits one-by-one will require a lot of simulation time. For instance, the following code is not efficient in MATLAB:

```
N = 1000; % Number of codewords
k = 5; % Number of information bits per codeword
c = zeros(N, n); % n is the block length
for i = 1:N
  for j = 1:k
    m(j) = randb(1); % Generate a block of k information bits
  end
  c(i, :) = mod2(m*G); % G is the generator matrix (assumed known)
end
```

Much better is:

```
N = 1000; % Number of code words
k = 5; % Number of information bits per code word m = randb(N, k);
m = randb(N, k);
c = mod2(m*G);
```

You may also want to have a look `help parfor` to make use of all processors in your computer.

References

[1] Dept. Language and Communication. “Student Writing Manual”. Chalmers University of Technology, 2007