CT111: Introduction to Communication Systems

A Software Project

The main project[1] of CT111 is on software implementation and performance analysis/evaluation of the convolution coding scheme. Specifically, you will implement the convolutional encoder, whose output will be passed through the BSC and BEC and the Gaussian channels, and for each channel, the received sequence will be decoded using a convolutional decoding algorithm. You are required to generate a working software as well as an analysis of the performance of this channel encoding/decoding scheme.

Project Groups and Evaluations

- ▷ For this project, you are required to work in groups. The groups will be formed based on the student IDs — the number of the students per group will be 12. Thus, ID 001 to 012 will be in Group 1, IDs 013 to 024 will be in group 2 and so on.
- ▷ On the due date of the project (to be announced), each group will make a presentation of its project work.
  - → At the time of evaluation, each group should come prepared with a (e.g., Powerpoint) presentation of the work accomplished.
  - → The presentation should be of professional quality, and it should have
    1. *an introduction section:* This should highlight the key achievements, the obstacles faced and solved, and the algorithmic insights gained, etc.,
    2. *a technical description section:* This should describe the main elements of the logic underlying the software,
    3. *a numerical results section:* This should describe the details of "what-if" experiments performed, and the numerical performance results obtained as a result of these experiments. This should also contain the validation/verification of these results (that demonstrates that there is no conceptual or programmatic "bug" in the software),

[1] This project is a main *attraction* of CT111. If you do the project well, you're likely to start feeling the passion for the subject and start climbing Step 3 (i.e., get a feel of "why to study CT[111]?"). This project combines the best of multiple "worlds", e.g., the development and analysis of efficient algorithms, machine learning and AI, communication theory, probability and estimation theory, coding theory, etc. Furthermore, this is not only a coding/programming exercise — here, you will need to understand the underlying theory well. Keep in mind that the programming/coding is a "cheap" job, many people can do it once clear instructions are provided. In contrast, the skills such as an in-depth conceptual understanding, mathematical insights and the ability to develop new algorithms are the 'premium" skills — very few possess these.

4. *a conclusion/summary section:* This should describes the key lessons learnt, the main achievement that the group collectively is proud of accomplishing and the main dissatisfaction that the group has (i.e., the work that the group would have liked to do but could not complete).

→ You should also have a working code that you will need to demonstrate during the presentation. Any member of the group may be selected to explain the logic of any module of the program.

▷ You can use either C++, Matlab or Python for this project.

→ *After* you successfully complete your project, you are encouraged, as a general programming exercise, to convert the code you've written to some other programming language.

▷ A mentor TA will be assigned to each group. The mentor TA will supervise and evaluate a total of three groups.

→ Please consult your mentor TA if you have difficulties or hurdles. More importantly, keep the mentor TA abreast of your individual progress and the group progress — the mentor TA will monitor your individual performance relative to the group (who has been the leader versus the laggard) and it's in your benefit to interact with the mentor TA as frequently as possible. You're also encouraged to meet and interact with the course instructor as well regarding the project.

→ A question you should ask before you walk into the presentation venue: *will the course instuctor and the mentor TA recognize me as someone who has been proactive during this project development?* If the answer[2] to this question is *yes*, you're likely to do well in the presentation.

Honor Code

You will be required to submit the following pledge at the time of the interim and final evaluations of the project work. This should be the opening page of your project presentation slides.

- We declare that
  - → The work that we are presenting is our own work.
  - → We have not copied the work (the code, the results, etc.) that someone else has done.
  - → Concepts, understanding and insights we will be describing are our own.
  - → We make this pledge truthfully. We know that violation of this solemn pledge can carry grave[3] consequences.
- Signed by: all the members of the project group.

[2] In general, it is in your best interest, both from the learning point of view as well as for getting a good evaluation, that you do not sit on the sidelines or be satisfied with some secondary tasks (preparing the presentation, understanding someone else's code/logic, etc.). Instead you should insist on engaging in the actual program development and working as if this is your own individual project. Do not hope or expect that someone else in your group will develop the algorithms or do a thorough study of the algorithm performance, instead be proactive and take up such task yourself.

[3] E.g., Your submissions will be spot-checked and the entire project group will be awarded 0 marks even if there is a slightest hint of plagiarism found during this check. It will be an *extremely* foolish mistake to engage in the plagiarism, or to accept someone in your group who is attempting to plagiarize.

Basic:

Each group is required to complete this section.

1. **Encoder:** A convolutional encoder with constraint lengths $K = 3$, the generator polynomials $G_1 = [1, 1, 1]$ and $G_2 = [1, 0, 1]$, and rate $r = k/n = 1/2$. See [1, Slide 15].

   - The length k of the input to the encoder, i.e., the k information bits that are sent to the encoder, should be a user-configurable input. The encoder should work for any values of k.
   - Once the user of the software specifies a value of k, the encoder should generate a block of k information bits as Bernouilli($p = 0.5$) random variables. This information bit-block should be passed as the input to the encoder. The encoder append two zeros to this block (to bring the Trellis to the all-zeros starting state S_0 [1, Slide 17]) and it should generate $n = 2k + 4$ bits at the output.
   - The encoder can be implemented either as a finite state machine [2, Fig. 8-3] or as a *linear shift-invariant system (LSI)* [2, Eq. 8.3].

2. **Channel:** The output of the encoder of length $n = 2(k + 2)$ bits should be passed through each of the following three models of communication channel.

   - the binary erasure channel BEC(p),
   - the binary symmetric channel BSC(p),
   - the Gaussian channel that introduces Gaussian-distributed noise n with zero mean and variance $\sigma_n^2 = \frac{1}{2\,r\,\gamma_b^{lin}}$, where γ_b^{lin} is the per-bit SNR in the linear scale (the per-bit SNR in dB is defined as $\gamma_b = 10\log_{10}(\gamma_b^{lin})$). The input to the Gaussian channel should be BPSK modulated, i.e., a bit $b \in \{0, 1\}$ obtained at the output of the encoder should be converted to a voltage level $s \in \{-1, 1\}$ volts as $s = 2b - 1$. The output of the channel is $r = s + n$.

3. **Decoder:** The output of the channel should be sent to the Viterbi Algorithm [1, Slides 20, 21, and 22], and [3].

   - The Viterbi Algorithm (VA) is based on the Trellis representation of the transmitted codeword of length n bits. The trellis has 2^{K-1} states [1, Slide 19].
   - Following are the main computational aspects[4] of the VA:
     - ▷ *Calculation of branch metrices:* The Trellis has eight branches connecting the four states at bit index i to the four states at bit index $i + 1$ [1, Slide 19]. The branch metric of each branch is calculated as the *distance* between the channel output and the branch output. This distance is the Hamming distance for the BEC and the BSC channels and it is the Euclidean distance for the Gaussian channel.
     - ▷ *Calculation of the path metrices:* The path metric for State S_j ($j \in \{0, 1, 2, 3\}$) at bit index i is calculated by adding to each of the two path metrices at index $i - 1$ the branch metrices of the two branches that terminate at State S_j at index i. The surviving path at State S_j at index i is the one with the smallest path metric. At each bit index i, the Trellis has four surviving paths — one corresponding to each of the four states at index i.

[4] The development of a correct VA program is the main challenging work of this project. Besides the correctness of the logic, also pay attention to the time complexity of your code (the program should run *fast*).

▷ *Traceback through the Trellis:* After completing the forward sweep through the Trellis, a traceback is needed to obtain the decoded bits. See [1, Slide 20].

4. **A Monte Carlo Simulation:** Your presentation should contain the probability of bit error $P_b(\gamma_b)$ at the output of the convolutional decoder. Develop[5] a Monte-Carlo simulation to numerically evaluate this probability. In the simulation, set the length k of the information bit block to be 500 bits, and the number of Monte-Carlo simulations to be $N_{sim} = 20000$.

   - *For the Gaussian noise channel:* obtain the Monte-Carlo experimental result for each value of per-bit SNR in dB, $\gamma_b = 10\log_{10}(\gamma_b^{lin}) \in \{0, 0.5, 1.0, 1.5, \ldots, 8\}$.
   - *For the BSC and the BEC channels:* Convert each value of γ_b above to p as follows: $p = Q(\sqrt{2\,r\,\gamma_b^{lin}})$.

   Provide all the three results (i.e., for the Gaussian noise, the BSC and the BEC) of $P_b(\gamma_b)$ as a function of γ_b in one single chart.

   - A reference check for the Gaussian noise channel: $P_b(\gamma_b = 5\,dB) \approx 10^{-4}$ (only one bit in average out of 10000 information bits should be in error).
   - A reference check for the BEC channel: $P_b(\gamma_b = 5\,dB) \approx 3.5 \times 10^{-3}$ (only 3 to 4 bits in average out of 1000 information bits should be in error).

Intermediate

Enhance the convolutional encoder and decoder of the Basic section as follows:

1. the constraint length $K = 4$ and rate $r = 1/2$ and the generator polynomials as specified in [4, Table 8-2-1, Second Row].
2. the constraint length $K = 4$ and rate $r = 1/3$ and the generator polynomials as specified in [4, Table 8-2-2, Second Row].

Demonstrate the probability of bit error performance comparison (through Monte Carlo simulations in the BSC and the AWGN channels) of these two coding schemes with $K = 3$, $r = 1/2$ code developed as a part of the Basic section of the Project.

Advanced: Option A (Mathematical Analysis)

Compare the Monte-Carlo performance results obtained for the rate $r = 1/2$ coding scheme implemented in the Basic section of this project and $r = 1/2, K = 4$ scheme in the Intermediate section with the theoretical upper bounds:

1. *For the Gaussian noise channel:* the upper bound is given as [4, Eq. 8-2-27]
$$P_b < \frac{dT(D,N)}{dN}|_{N=1,D=\exp(\gamma_b\,r)}.$$
2. *For the BSC channel:* the upper bound is given as [4, Eq. 8-2-34]
$$P_b < \frac{dT(D,N)}{dN}|_{N=1,D=\sqrt{4p(1-p)}}.$$

   The transfer function $T(D,N)$ is defined in [4, Eq. 8-2-6].

[5] Recall: you have performed Monte-Carlo experiments for a basic BPSK modem and a repetition coding scheme in Lab 3.

Advanced: Option B (Development of a New Algorithm)

A shortcoming of the Viterbi Algorithm is that it does not provide the a-posteriori ratio (the odds) of each decoded bit. The BCJR algorithm (named after its inventors, Bahl, Cocke, Jelinek and Raviv) overcomes this shortcoming. The BCJR algorithm also makes use of the Trellis representation of the convolutional code and, similar to the VA, it performs a forward sweep through the Trellis. However, unlike the VA, it also performs a *backward* sweep through the Trellis (therefore, it is also known as the FBA[6], i.e., the Forward Backward Algorithm). A short but clear description of the BCJR algorithm is provided in [5, Slides 17 to 25] (uploaded on Moodle). Also see the original paper by BCJR [6].

Develop the FBA and show that its performance for $K = 3$, $r = 1/2$ code of the Basic section compares well against the VA (both the FBA and the VA are optimal, i.e., they provide identical $P_b(\gamma_b)$ performance).

References

[1] M. Valenti, "Channel Coding for IEEE 802.16e Mobile WiMAX," June 2009

[2] H. Balakrishnan, J. White, "MIT 6.02 DRAFT Lecture Notes, Chapter 8, Convolution Coding," Fall 2010 (Last update: October 4, 2010)

[3] H. Balakrishnan, J. White, "MIT 6.02 DRAFT Lecture Notes, Chapter 9, Viterbi Decoding of Convolution Codes," Fall 2010 (Last update: October 6, 2010)

[4] J. G. Proakis, "Digital Communications," third edition, 1994.

[5] M. Valenti, "Channel Coding for IEEE 802.16e Mobile WiMAX," June 2009

[6] L. R. Bahl, J. Cocke, F. Jelinek and J. Raviv, "Optimal Decoding of Linear Codes for Minimizing Symbol Error Rate," June 2009

[6] The FBA is the pinnacle of the Bayes' Theorem, and it is considered (so is the VA) as an important algorithm residing at the heart of many ML/AI techniques.