230/236 Wireless Networking
Project Description

Kasper B. Rasmussen

Due: December 6, 2012 @ 11:59pm PST

Project Objectives and Guidelines The aim of the course project is to create a link level Monte-Carlo simulator, to study the performance of convolutional codes over wireless channels. Please note the following:

- Students should work on the project in groups of two. The group should work jointly on all tasks of the project. Group members are expected to cooperate on the project work with nearly equal load.
- The deadline for forming the groups is *Thursday October 18, 2012*. Email your group names to me (kbrasmus@ics.uci.edu) before the deadline.
- Matlab should be used as the programming language in the project. The code must be readable and well documented. Please maintain a proper coding style.
- Each group should type their own solution for the project task. Any copying will be considered cheating and dealt with accordingly. Name your solution file `groupnumber-project.pdf`. You will get group numbers when you email me the group members.
- You solution should include a brief description of how to run the code and obtain the required results. Name your description file `readme.txt`.
- You are required to submit all your Matlab code in addition to your solution file and description file. Put all the files the dropbox directory named `groupnumber-project`. All materials must be submitted before the deadline.
- The project final grade will depend on the correctness and completeness of the solution, the neatness of the code, the efficiency/speed of the code, the clearness of the output, and the implementation of additional and innovative features.
- Enjoy and Good Luck!

Simulator Design (Convolutional Codes: Design and Performance)

You will develop a link level simulator using Matlab for a basic wireless communications system. The simulator should consist of the following blocks:

- Bernoulli source that generates a sequence of 0's and 1's. The source data is generated in frames of size F = 1000 bits.

- Convolutional encoder with the following parameters: rate $R_c = 1/N$; generator polynomials $g_i (1 \leq i \leq N)$; number of memory elements M.
- Block or pseudo-random interleaver (it is up to you to select what you think is better).
- BPSK modulator with the following mapping of encoded bits to symbols: 0 mapped to $+1$ and 1 mapped to -1.
- Wireless channel with the following parameters: AWGN with variance E_b/N_o; flat Rayleigh fading with unity average power and Jakes power spectral density.
- BPSK demodulator and detector.
- Block or pseudo-random deinterleaver.
- Viterbi convolutional decoder with the following parameters: hard decision decoding (HDD) or soft decision decoding (SDD); with full CSI info or without CSI info.
- For each simulated frame, the number of decoded bit errors should be calculated to obtain a corresponding bit error rate (BER) value.
- For each SNR (E_b/N_o) level, the simulation chain should be executed over a large enough number of frames K in order to get reliable statistics at the corresponding BER level. You have to decide how to set the value of K in various cases. You can control the SNR level by controlling the AWGN variance (i.e., assume E_b normalized to 1).
- For generating a complete BER plot for a given set of parameters, you have to run the simulation chain from $E_b/N_o = 0$ dB until you reach a target BER $P_{\text{target}} = 105$.

Simulator Implementation For the implementation of the simulator, please note the following:

- Feel free to use any needed functions from the Matlab toolboxes including the Communications Toolbox (e.g., rayleighchan.m, convenc.m, vitdec.m). Check Matlab help and demos in order to learn more about the Communications toolbox. Note: You should clearly indicate as part of your solution what functions you developed on your own and what functions you used from the toolboxes.
- For modelling AWGN and flat Rayleigh fading, you can use functions from the Communications toolbox. Make sure that the normalized fading power is equal to 1. For flat fading, assume Doppler shifts that correspond to mobile speeds of $v = 3$ km/hr (low mobility) and $v = 120$ km/hr (high mobility) with carrier frequency $f_c = 900$ MHz.
- Plot BER in log scale as a function of E_b/N_o in dB where E_b is energy per source information bit and not per coded bit. Take care of proper normalization for fair comparisons.
- Your simulator can have two modes of operation. One mode for demonstration purposes in case results are required for a given set of input parameters. Another mode for automation purposes in case results are required based on a search over a wide set of possible input parameters.

- In order to make your simulator more user friendly, provide the values of input parameters either via a configuration file or via a graphical user interface (GUI). The user should not need to reenter all input parameters in case only one of the parameters need to be changed.
- Creating a GUI for your implementation is regarded as an additional feature especially for demonstration purposes.

Project Requirements The project requirements are kept broad in order to allow for creativity in terms of what to do and how to do it. The project requirements can be divided into basic features and additional features. The following are the main project requirements:

1. Use your simulator in order to search for the the generator polynomials of the best convolutional codes with $R = 1/2$ and $M = 2, M = 3$. To do this, you have to simulate the performance of all possible codes and select the ones that have the best BER in the range around 10^{-3}. Assume HDD and AWGN.
   - Provide the generator polynomials of the best codes for each value of M.
   - Sketch schematics for the encoders of the best convolutional codes.
   - Present complete BER curves for the best convolutional codes over AWGN channels.
   - Present complete BER curves for the best convolutional codes over wireless channels flat fading (assume low mobility).
   - Providing results for $M = 4$ and $M = 8$ will be regarded as additional features.
2. Repeat the previous parts for convolutional encoders with $R = 1/3$.
3. Answer the following analysis questions based on your results and facts from the literature (books and papers):
   - Compare (and comment on) the performance of convolutional codes with different values of M.
   - Compare the performance of convolutional codes with/without fading in the channel. Comment.
   - Compare (and comment on) the performance of convolutional codes with different rates ($R = 1/2$ versus $R = 1/3$).
   - Are the best encoders over AWGN channels also the best encoders over flat fading channels? Justify your answer either intuitively or via simulations.
4. For the best convolutional code with $R = 1/2$ and $M = 3$, perform necessary simulations and present complete BER curves in order to answer the following questions:
   - Compare (and comment on) the performance between HDD and SDD over AWGN channel.
   - Compare (and comment on) the performance between HDD with/without CSI and SDD with/without CSI over flat fading channel (low mobility).
   - Compare (and comment on) the cases with and without interleaving over flat fading channel (low mobility). Assume HDD.

- Compare (and comment on) the cases with low mobility flat fading channel and high mobility flat fading channel. Assume HDD.

5. The following are additional features that you can add to your implementation and analysis:
   - Benchmark your results, whenever possible, against results published in the literature (books and papers). This is very important to justify the correctness of your work. Include clear referencing whenever applicable.
   - For the best convolutional code with $R = 1/2$ and $M = 3$, compare the performance with a repetition code having the same coding rate. Perform necessary simulations and present complete BER curves. Assume HDD and AWGN channel.
   - For the best convolutional code with $R = 1/3$ and $M = 3$, perform necessary simulations in order to search for the best puncturing matrix to obtain a code with rate $R = 1/2$. You have to determine how to do the decoding when puncturing is performed at the transmitter side. Compare the performance of the best $R = 1/2$ code with the obtained $R = 1/2$ punctured code over AWGN channel with HDD.
   - For the best convolutional code with $R = 1/2$ and $M = 3$, investigate the impact of channel estimation errors on the decoder performance with SDD. Model CSI error via some statistical model (e.g., Gaussian estimation noise with zero mean and given variance) and include results for various levels (variances) of error. Compare with HDD, SDD with full perfect CSI, and SDD without CSI.
   - Feel free to add any other special features, e.g., include diversity, higher order modulation, etc.

Deliverables

- All code that you wrote. Your code should be well documented.
- A "readme" file that describes the functionality of your simulator and how to run it. This is like a basic user manual for your simulator.
- A project report that includes the following:
  - For each student in the group: First Name, Last Name, Student ID and Group Number (will be assigned).
  - A brief overview of the main features of your implementation (up to 1 page).
  - A brief overview of the structure/logic of your code implementation, e.g., how did you set the value of K, how did you automate the code search, etc. (up to 2 pages).
  - A description of the design decisions you made and a short justification for them. If you found vagueness in the project description, list how you solved them in your implementation.
  - A description of additional features of your implementation. Bonus points will be given for innovative special features.

  - Results and analysis for all required cases in addition to any additional cases that you considered. Group curves into plots in a logical way that allows you to make necessary comparisons in order to answer the questions. Make sure to include clear captions and legends for each figure.
  - Indicate all references that you used to help in your implementation or to support your answers/analysis.

Grade Distribution

- 65%: Implementation of required features, generation of required results, and correctness of analysis.
- 15%: Structure and completeness of the project report.
- 10%: Structure and documentation of the project code.
- 10%: Benchmarking your results against published results from the literature.
- Up to 30% bonus: Implementation of more additional features.