Massachusetts Institute of Technology
18.413: Error-Correcting Codes Laboratory
Professor Daniel A. Spielman

Handout
April 1, 2004

Small Project 3

Due: April 8, 2004

In this project, you are going to implement and test a simple convolutional code. This project has two parts:

- Implementation
- Generating Data

1 Implementation

Your implementation will have five parts:

A. encoder

B. channel simulator,

C. decoder,

D. evaluator that compares the output of the decoder to the input to the encoder.

E. code to drive all these

These parts should all be separable so that you can test each individually. This will also allow you change components. Many of you will be using parts of these components in your final projects.

B. **encoder** You should use the recursive convolutional encoder presented in todays lecture (from Figure 2). The encoder should take as input a string of 0s and 1s, and output the corresponding codeword. Since the convolutional encoder is rate 1/2, this means that on an input of k bits, the encoder should output $2k$ bits.

C. **channel simulator**,

We'll just use the Gaussian Channel that we've used before.

C. **decoder**

You should implement the BCJR decoder presented in class today. You should provide hard outputs–that is a guess of 0 or 1 for each bit.

D. **evaluator**:

This will compare the output of the decoder to the input to the encoder, and compute the bit eror rate. Note that this comparison should only be performed for the message bits.

E. **driver code**:

For each noise level at which you simulate code performance, you should run enough simulations to obtain at least 100 runs in which the decoder produces at least 1 error. (and you can stop at 100)

For this project, you should produce a fresh codeword for each run (and it is easy to do so).

2 Generating Data

You should perform your simulations with $k = 10,000$. That is, each run should have $10,000$ message bits.

You should run simulations on the Gaussian channel at signal-to-noise ratio E_b/N_0: 0, 1, 2, 3, 4, 5 and 6 dB. You should report the observed bit error rate, as well as plot the resulting data in a table with the logarithm of the BER on the Y-axis (in matlab `semilogy`) and the signal-to-noise ratio in dB on the X-axis.

3 Reporting

Submit all code, plots, and the data used to generate plots (unless it is absurdly big). Start with an overview explaining what is to follow, and how your code is organized. To the extent that you generated data by interaction, rather than merely running the code you've submitted, explain what you did.

Note that I'm going to actually try to read your code, so please document it well!

4 Collaboration

For this project you are free to discuss how you do this with others in the class (and especially to get technical help). But, you should write your own code. You must acknowledge your collaborators.