EE 5501 Prof. N. Jindal
Digital Communication Nov. 5, 2009

**Homework 6**
Due: Thursday, Nov. 12, 11:15 AM

1. In class we have found the log-likelihood ratio of each information bit by computing the likelihood's of the trellis transitions associated with different information bits. In this problem we will see that the same can be done by computing the likelihood of each codeword (although this is very computationally inefficient). We assume that $K$ information bits plus 2 terminating bits are sent, so the information bit sequence $\vec{u}$ is length $K+2$ and the received symbol vector $\vec{y}$ is length $2(K+2)$.

   (a) Show that the likelihood of a particular information bit sequence $\vec{u}$ conditioned on the received symbols $\vec{y}$ can be written as:

   $$\mathbb{P}[\vec{u}|\vec{y}] = \mathbb{P}[\vec{y}|\vec{u}]\frac{\mathbb{P}[\vec{u}]}{\mathbb{P}[\vec{y}]}$$

   (b) Show that

   $$\mathbb{P}[\vec{y}|\vec{u}] = \frac{1}{\sqrt{(2\pi\sigma^2)^{2(K+2)}}}\exp\left[-\frac{1}{2\sigma^2}||\vec{y}-\vec{c}(\vec{u})||^2\right]$$

   where $\vec{c}(\vec{u})$ is the codeword (including the energy scaling) corresponding to information bit sequence $\vec{u}$.

   (c) Explain why the LLR of the $k$-th information bit can be written as:

   $$\mathbb{P}[u_k = 0|\vec{y}] = \sum_{\vec{u}:u_k=0}\mathbb{P}[\vec{u}|\vec{y}].$$

   (d) Combine the results of the previous parts to show

   $$\begin{aligned}\text{LLR}(u_k) &\triangleq \log\left(\frac{\mathbb{P}\left[u_k=0|\vec{y}\right]}{\mathbb{P}\left[u_k=1|\vec{y}\right]}\right)\\ &= \log\left(\frac{\sum_{\vec{u}:u_k=0}\exp\left[-\frac{1}{2\sigma^2}||\vec{y}-\vec{c}(\vec{u})||^2\right]}{\sum_{\vec{u}:u_k=1}\exp\left[-\frac{1}{2\sigma^2}||\vec{y}-\vec{c}(\vec{u})||^2\right]}\right).\end{aligned}$$

   This (inefficient) computation of the information bit LLR is performed in a file that has been provided to you to help you verify the correctness of your BCJR code (see next question).

2. In this problem you will implement the BCJR algorithm for the four state, $R = 1/2$ *systematic* binary convolutional code (recursive, systematic [7,5]) that was discussed in class (the trellis for this code is given on pg. 313 of the textbook).

   To get you started the following Matlab files are provided:

   - `encode_75_recursive_function.m`

     This function encodes the information bits that are provided to it, and also adds the necessary termination bits so that the codeword always ends in state 00.

   - `maxstar.m`

     This function implements the maxstar function.

   - `bcjr_bruteforce_75_recursive.m`

     This function computes LLR's in the brute force manner derived in problem 1 of this assignment, and should be used to verify the correctness of your BCJR algorithm.

   - `test_bcjr_75_recursive.m`

     This file generates random information bits, encodes them, creates a received signal, and calls your BCJR function and the brute-force LLR function. This file should be used to test your BCJR algorithm against the brute force algorithm.

   - `bcjr_function_75_recursive.m`

     This function will implement the BCJR algorithm. Only inputs and outputs are specified.

   Your assignment is to fill out the BCJR function (bcjr_function_75_recursive.m). (You should also go through the files that are provided to make sure you understand what they are doing.)

   Note: You should implement BCJR in the log-domain. In order to prevent numerical overflow when performing the forwards and backwards recursion, you should periodically (at every step, or every few steps) re-center the $a_k$ (and $b_k$) values about zero. Recall that in the log-domain, at any step $k$ you can add a constant to all of the $a_k(s)$ values without affecting the algorithm.

3. Once you have written the BCJR algorithm, your final assignment is to implement the $R = 1/3$ parallel concatenated turbo code that uses the recursive, systematic [7, 5] code as its component code. The file parallel_turbo_75_recursive_shell.m has been provided to you to serve as a shell. The file encodes the information bits (including the definition of an interleaver), creates the received symbols, and plots the corresponding bit error curves.

   Using the BCJR function from the previous section, you should be able to define the turbo decoder. Generate a plot of the bit error probability vs. $E_b/N_0$ for 0 to 5 dB after one, three, six, and thirteen iterations.