Turbo Codes: Parallel Concatenated Convolutional Codes (PCCCs) with Iterative Decoding

EECS 869: Error Control Coding

Fall 2017

In this project, we will be using the CCSDS turbo encoder shown in Figure 1, which consists of two identical *constituent* encoders fed by the same information block (albeit in permuted order). We will be using the rate-1/3 configuration. We have seen this constituent encoder earlier in the semester and have already drawn its trellis diagram and listed its look-up tables; for your convenience, the trellis diagram is shown in Figure 4 and the rate-1/3 trellis look-up tables are shown in Table 1. The trellis is already specified for you in the file `RunPcccSimulationTemplate.m`.

Complete the following tasks. You should submit an e-mail with three .m file attachments (use the e-mail address esp@eecs.ku.edu).

1. **De-multiplex the single received data stream into the *a priori* LLRs that will be fed to the SISOs.** As we have discussed in class, it is important yet non-obvious how properly handle the inputs and outputs of SISO a and SISO b. Figure 2 helps establish some of the notation we used during our discussion of this issue. Figure 3 shows one possible correct configuration for a turbo decoder. Notice in particular that SISO b receives only one *a priori* data stream via its "**c**" input port. Accordingly, it uses a look-up table for $\mathbf{c}_{\mathrm{B}}(e)$ that has one less column than $\mathbf{c}_{\mathrm{A}}(e)$, which is also how Table 1 is presented. You may use some other correct turbo decoder configuration if you like. Either way, there is a TODO in `RunPcccSimulationTemplate.m` for you to prepare $\boldsymbol{\lambda}_{\mathrm{A}}(c;\mathrm{I})$ and $\boldsymbol{\lambda}_{\mathrm{B}}(c;\mathrm{I})$ for their respective SISOs.

2. **Implement the Turbo (PCCC) Decoder using the log-based SISO algorithm.** Implement the turbo (PCCC) decoder in Figure 3, or some other correct version, using the log-based SISO algorithm. In the e-mail distribution, you have a template decoder `PcccSisoLogDecoderTemplate.m`. You have already implemented the log-based SISO algorithm in a previous project. Therefore, your task is simply to "wire together" the appropriate modules. You do not need to run a full BER simulation; however, the template simulation file is set up to do so. You should verify that your full PCCC decoder works properly.

3. **Implement the Turbo (PCCC) Decoder using the max–log SISO algorithm.** This is a straightforward extension of the previous task. You should add scale factors K_1 and K_2 to help improve the performance. Place your implementation in a separate file called `PcccSisoMaxLogDecoderXXX.m`.

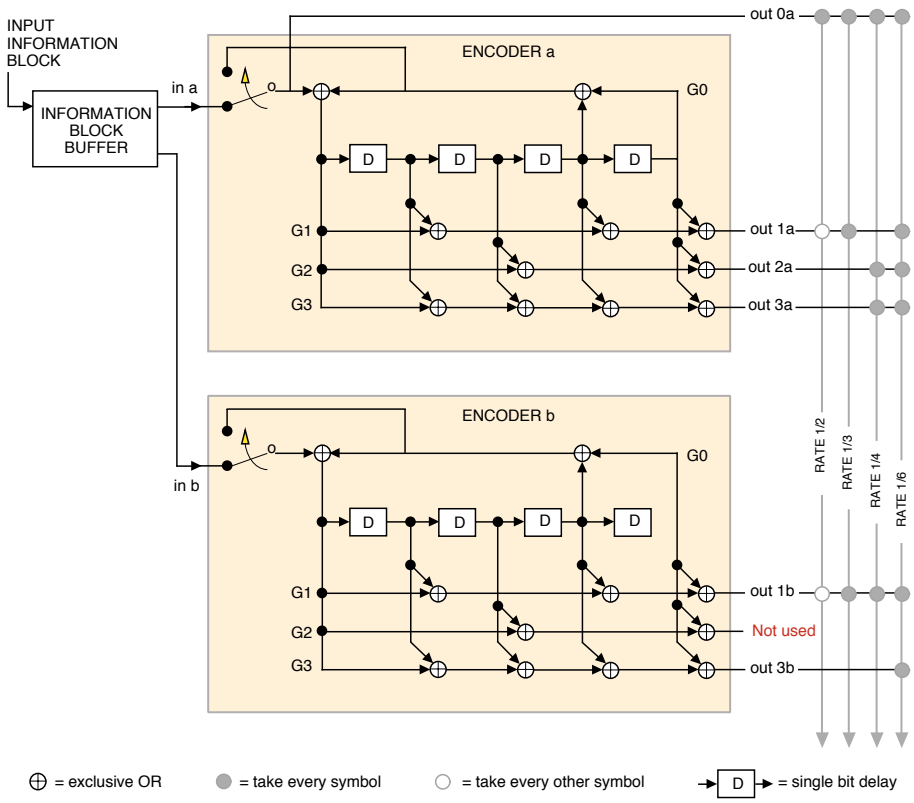

Figure 1: CCSDS turbo encoder. We will be using the rate-1/3 configuration in this project.

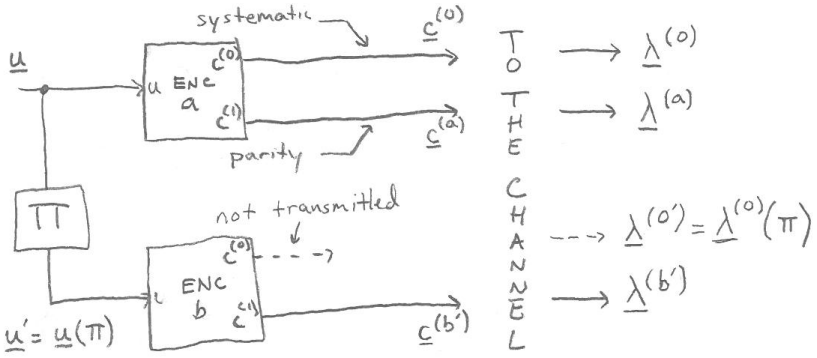

Figure 2: Turbo encoder with the notation used in the midterm.

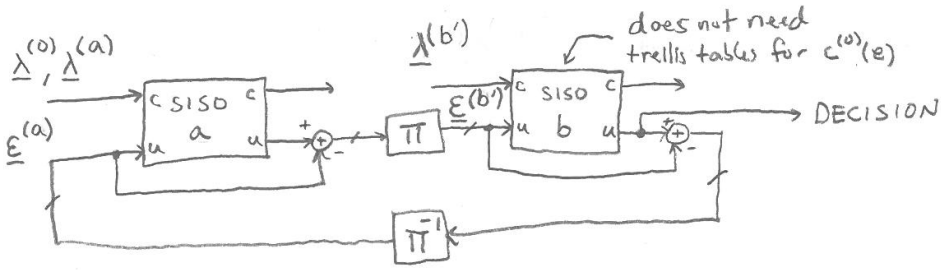

Figure 3: Correct implementation of the turbo decoder with the notation used in the midterm.

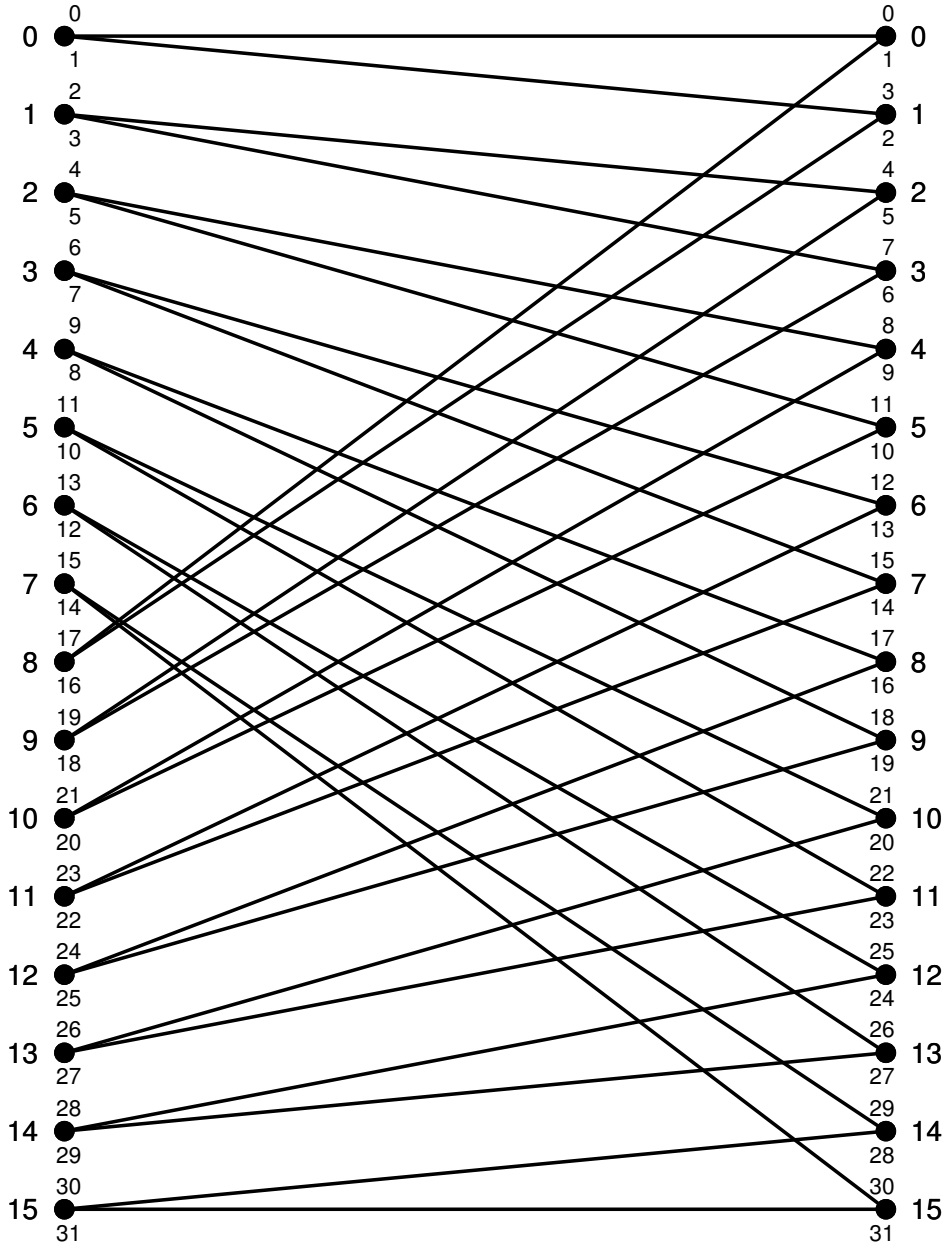

Figure 4: Trellis diagram for the CCSDS constituent convolutional code. Each starting state is labeled with $s^{\mathrm{S}} \in \{0, 1, \ldots, 15\}$ and each ending state is labeled with $s^{\mathrm{E}} \in \{0, 1, \ldots, 15\}$. On the left side of the trellis, there are two edges leaving each starting state; these are labeled with $e^{\mathrm{L}} \in \{0, 1, \ldots, 31\}$. On the right side of the trellis, there are two edges entering each ending state; these are labeled with $e^{\mathrm{R}} \in \{0, 1, \ldots, 31\}$. Therefore, a given edge has four labels associated with it: e^{L}, s^{S}, s^{E}, and e^{R}. These labels are listed in table form on the following page, along with the edge information bit, $u(e)$, and the edge code symbols, $\mathbf{c}(e)$ [the latter vary with the code rate, R, and with Encoder A vs. Encoder B]. The trellis table is sorted (indexed) according to the left edge index, e^{L}. In the specification of the trellis algorithms, there are mathematical expressions such as $\{e : s^{\mathrm{S}}(e) = s\}$, which means "the set of edges such that the starting state of the edge is equal to a desired state s." Because of the labeling scheme above, such a statement translates to "the two e^{L} values grouped with a given s^{S}." Or similarly, the statement $\{e : s^{\mathrm{E}}(e) = s\}$ means "the set of edges such that the ending state of the edge is equal to a desired state s." This translates to "the two e^{R} values grouped with a given s^{E}." Finally, the statement $\{e : u(e) = 0\}$, which means "the set of edges such that the edge information bit is zero," translates to "every other edge, starting with edge zero" whether the trellis is viewed from the left or right edge labels.

e^{L}	$s^{\mathrm{S}}(e^{\mathrm{L}})$	$s^{\mathrm{E}}(e^{\mathrm{L}})$	$e^{\mathrm{R}}(e^{\mathrm{L}})$	$u(e^{\mathrm{L}})$	$\mathbf{c}_{\mathrm{A}}(e^{\mathrm{L}})$		$\mathbf{c}_{\mathrm{B}}(e^{\mathrm{L}})$
0	0	0	0	0	0	0	0
1	0	1	3	1	1	1	1
2	1	2	4	0	0	1	1
3	1	3	7	1	1	0	0
4	2	4	8	0	0	0	0
5	2	5	11	1	1	1	1
6	3	6	12	0	0	1	1
7	3	7	15	1	1	0	0
8	4	9	18	0	0	0	0
9	4	8	17	1	1	1	1
10	5	11	22	0	0	1	1
11	5	10	21	1	1	0	0
12	6	13	26	0	0	0	0
13	6	12	25	1	1	1	1
14	7	15	30	0	0	1	1
15	7	14	29	1	1	0	0
16	8	1	2	0	0	0	0
17	8	0	1	1	1	1	1
18	9	3	6	0	0	1	1
19	9	2	5	1	1	0	0
20	10	5	10	0	0	0	0
21	10	4	9	1	1	1	1
22	11	7	14	0	0	1	1
23	11	6	13	1	1	0	0
24	12	8	16	0	0	0	0
25	12	9	19	1	1	1	1
26	13	10	20	0	0	1	1
27	13	11	23	1	1	0	0
28	14	12	24	0	0	0	0
29	14	13	27	1	1	1	1
30	15	14	28	0	0	1	1
31	15	15	31	1	1	0	0

Table 1: Trellis table for the CCSDS constituent convolutional code in the rate-1/3 configuration. The code symbols, $\mathbf{c}(e)$, are listed as binary vectors but in the algorithm implementation they may used in decimal form, $C(e)$, using the standard conversion between binary and decimal. The first column of $\mathbf{c}_{\mathrm{A}}(e)$ matches $u(e)$ because Encoder A contains the systematic output; although Encoder B receives $u(e)$ as its input (in permuted order), it does not have $u(e)$ as one of its outputs and thus $\mathbf{c}_{\mathrm{B}}(e)$ has one less column. Antipodal symbols, $a(u(e))$ and $a(\mathbf{c}(e))$, are not listed in the table because they can be derived when needed via the relation $a(x) = 2x - 1$.