Chapter 4 Turbo Codes

- 4.1 Introduction of Turbo Codes
- 4.2 Encoding of Turbo Codes
- 4.3 Decoding of Turbo Codes (Turbo Decoding)
- 4.4 Performance Analysis

§ 4.1 Introduction of Turbo Codes

- Invented by C. Berrou, A. Glavieux and P. Thitimajshima in 1993 [1].

- Integrate a couple of conv. codes in a parallel encoding structure. The two conv. codes are called the constituent codes of a turbo code.

- Exploit the interplay between the decoders of the two constituent codes in a soft information exchange decoding mechanism.

- Such a decoding mechanism is called turbo decoding, turbo decoding is NOT limited to decode turbo codes, but to any concatenated (serial or parallel) code.

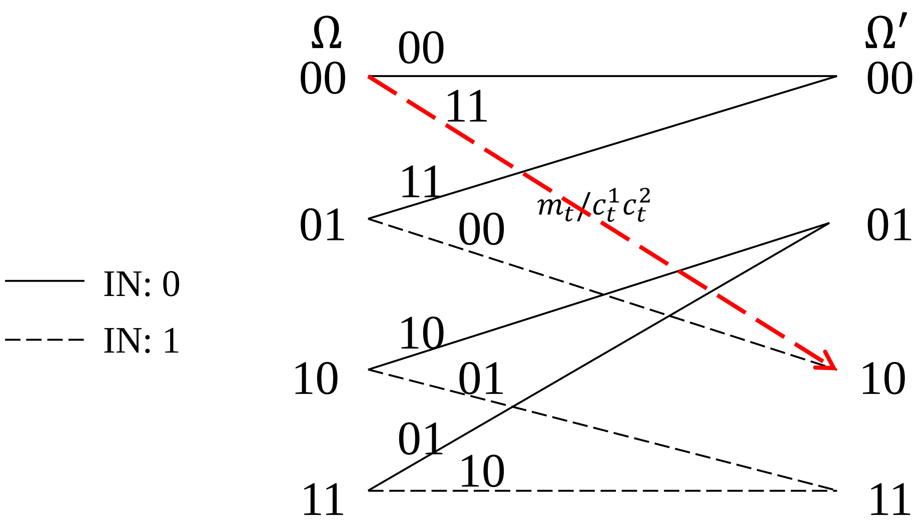
- Shannon capacity can be approached with the existence of error floor.

[1] C. Berrou, A. Glavieux and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: turbo codes," *Proc. ICC'93*, pp. 1064-1047, Geneva, May 1993.

§ 4.1 Introduction of Turbo Codes

Why do we need code concatenation?

In BCJR decoding of a conv. code,



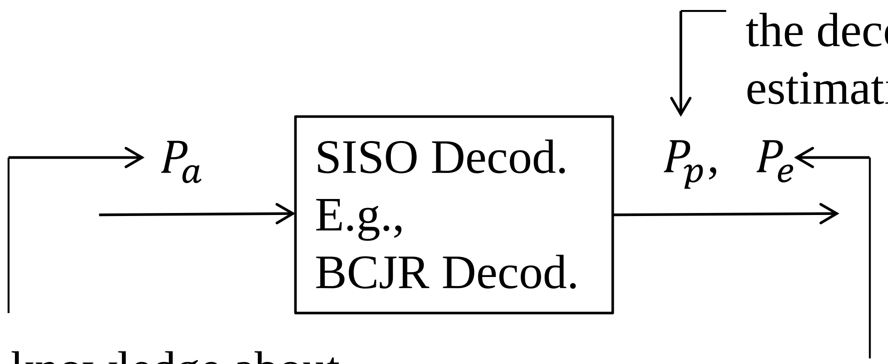
a priori prob.

$$\Gamma_{\Omega\to\Omega'} = \boxed{P_a(m_t)} \cdot \underbrace{P_{ch}(c_t^1) \cdot P_{ch}(c_t^2)}_{\text{channel observations}}$$

With a single conv. code, we do not have any information of information bit m_t and the *a priori* prob. $P_a(m_t = 0) = P_a(m_t = 1) = 0.5$. With a couple of conv. codes that share the same information bits (but in different permutations), one decoder can gain *a priori* prob. of information bits m_t from the output of the other decoder, and vice versa. As a result, BCJR decoding of each constituent code can be improved.

§ 4.1 Introduction of Turbo Codes

***A posteriori* prob.**: knowledge about the information bits after the decoding. It is used for estimation.



***A priori* prob.**: knowledge about the information bits before the decoding. It is also called the intrinsic prob.

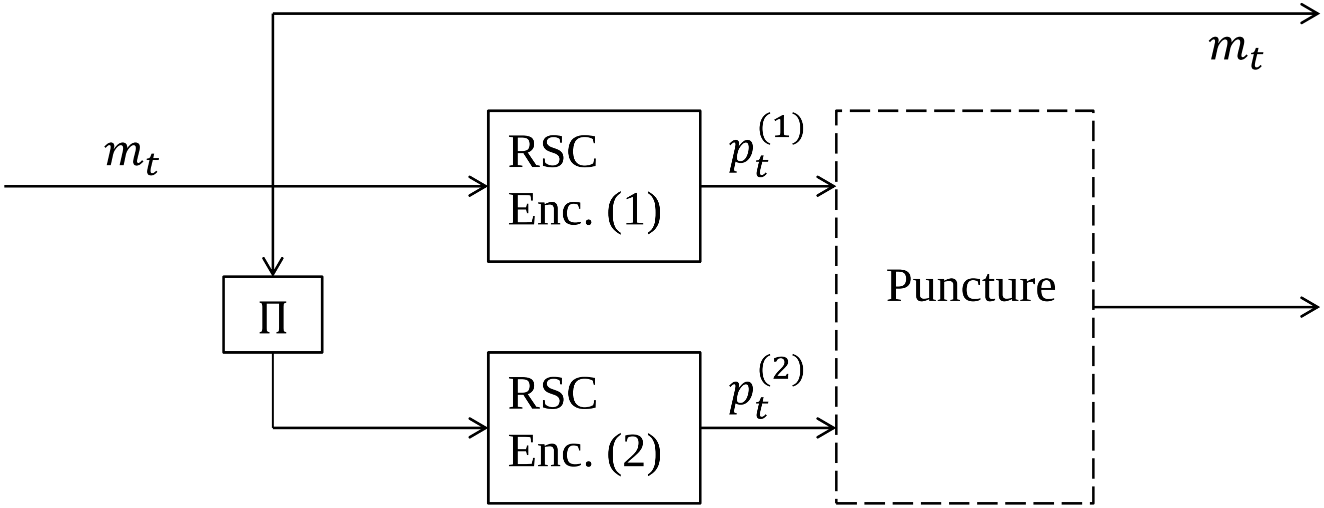
Extrinsic prob.: $P_e = \frac{P_p}{P_a}$, extra knowledge (excluding the *a priori* prob.) delivered by the SISO decoder.

§ 4.2 Encoding of Turbo Codes

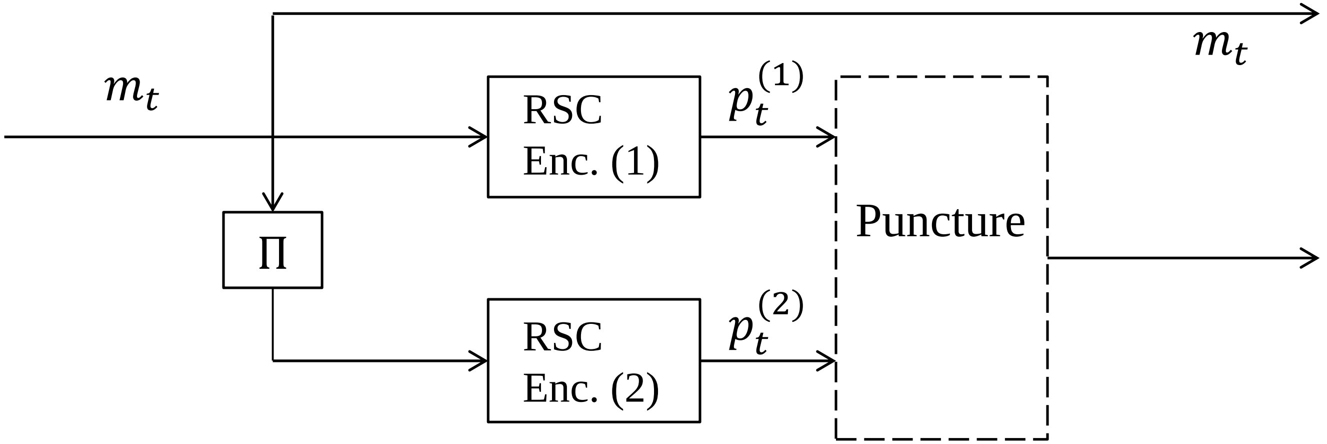
Constituent codes: Recursive Systematic Conv. (RSC) codes. Normally, the two constituent codes are the same.

Interleaver (Π): Generate a different information sequence (a permuted sequence) as the input to the RSC encoder (2). Normally, it is a random interleaver.

Puncture: Control the rate of the turbo code.



§ 4.2 Encoding of Turbo Codes



- Given the binary message sequence as $\bar{m} = [m_1,\ m_2,\ \cdots\ m_k]$, output of the turbo encoder should be

$$\bar{c} = [m_1\, p_1^{(1)}\, p_1^{(2)}\ \ m_2\, p_2^{(1)}\, p_2^{(2)}\ \cdots\ m_t\, p_t^{(1)}\, p_t^{(2)}\ \cdots\ m_k\, p_k^{(1)}\, p_k^{(2)}].$$

- Rate of the turbo code is 1/3. To increase the rate to ½, we can use puncturing whose pattern can be represented by

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

puncture $p_t^{(2)}$ when t is odd

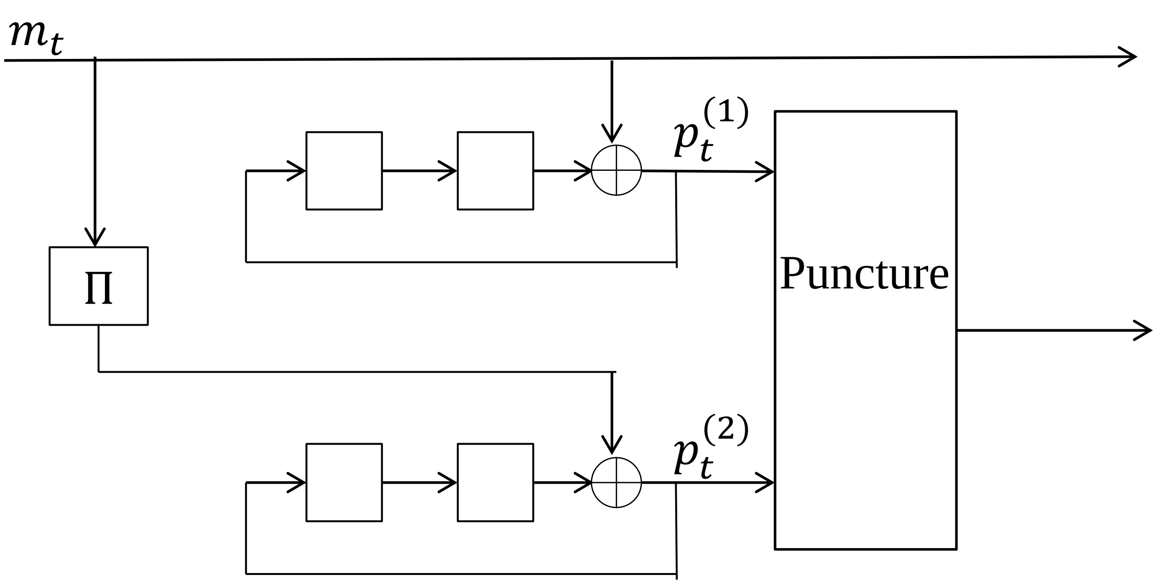
puncture $p_t^{(1)}$ when t is even.

- After puncturing, output of the turbo encoder should be

$$\bar{c} = [m_1\, p_1^{(1)}\ \ m_2\, p_2^{(2)}\ \cdots\ \underbrace{m_k\, p_k^{(1)}}_{\text{when } k \text{ is odd}}\underbrace{(m_k\, p_k^{(2)})}_{\text{when } k \text{ is even.}}]$$

§ 4.2 Encoding of Turbo Codes

Example 4.1 Given the turbo encoder shown below with constituent code of conv. $(1, {}^{1}/_{1+x^2})$. The puncturing pattern is $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$. The interleaving pattern is {8, 3, 7, 6, 9, 1, 10, 5, 2, 4}. Determine the turbo codeword of message vector $\bar{m}$ = [1 0 0 1 0 1 1 0 0 0].



§ 4.2 Encoding of Turbo Codes

The original message vector

$$\bar{m} = [1\ 0\ 0\ 1\ 0\ 1\ 1\ 0\ 0\ 0]$$

Output of the 1st constituent code is:

$$\bar{p}^{(1)} = [1\ 0\ 1\ 1\ 1\ 0\ 0\ 0\ 0\ 0]$$

After interleaving, the permuted message vector becomes

$$\bar{m}' = [0\ 0\ 1\ 1\ 0\ 1\ 0\ 0\ 0\ 1]$$

Output of the 2nd constituent code is:

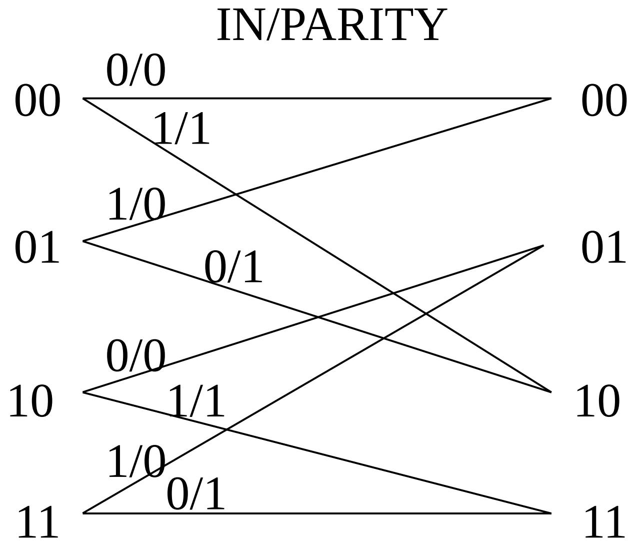
$$\bar{p}^{(2)} = [0\ 0\ 1\ 1\ 1\ 0\ 1\ 0\ 1\ 1]$$

Before puncturing, the turbo codeword is

$$\bar{c} = [110\ 000\ 011\ 111\ 011\ 100\ 101\ 000\ 001\ 001]$$

After puncturing, the turbo codeword is

$$\bar{c} = [11\ 00\ 01\ 11\ 01\ 10\ 10\ 00\ 00\ 01]$$



Trellis of the code

§ 4.3 Decoding of Turbo Codes

- **Parameterization**
  - Turbo codeword $\bar{c} = [m_1\, p_1^{(1)}\, p_1^{(2)},\ m_2\, p_2^{(1)}\, p_2^{(2)},\ \cdots,\ m_k\, p_k^{(1)}\, p_k^{(2)}]$.
  - Assume the turbo codeword is transmitted using BPSK.
  - Received symbol vector

$$\bar{r} = [r_1^{(0)}\, r_1^{(1)}\, r_1^{(2)}, r_2^{(0)} r_2^{(1)}\, r_2^{(2)},\ \cdots, r_k^{(0)}\, r_k^{(1)}\, r_k^{(2)}].$$

  - Interleaved message vector

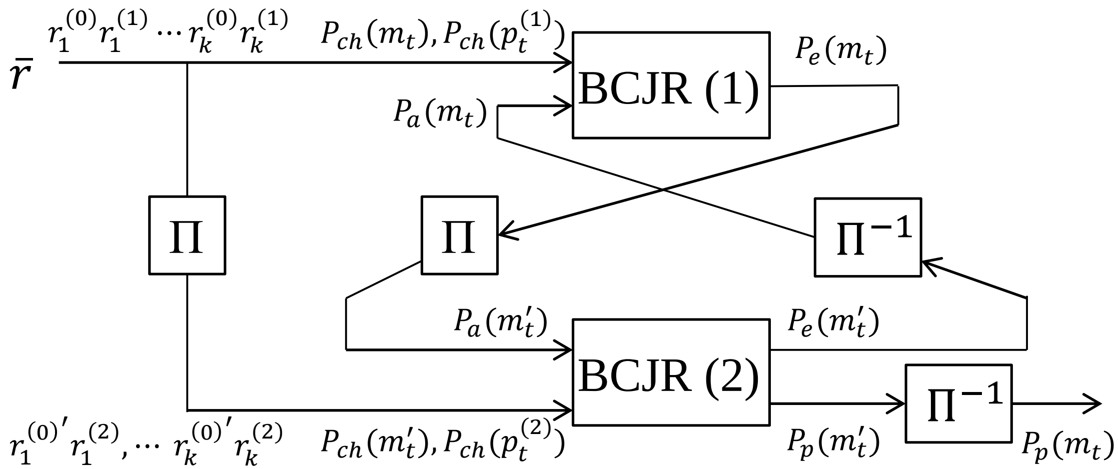
$$\bar{m}' = \Pi(\bar{m}) = [m_1',\ m_2',\ \cdots,\ m_k'].$$

  - Interleaved (information) symbol vector

$$\left[r_1^{(0)'}, r_2^{(0)'}, \cdots, r_k^{(0)'}\right] = \Pi([r_1^{(0)},\ r_2^{(0)},\ \cdots,\ r_k^{(0)}]).$$

§ 4.3 Decoding of Turbo Codes

Turbo decoding structure



- In BCJR (1), trellis transition probability is determined by

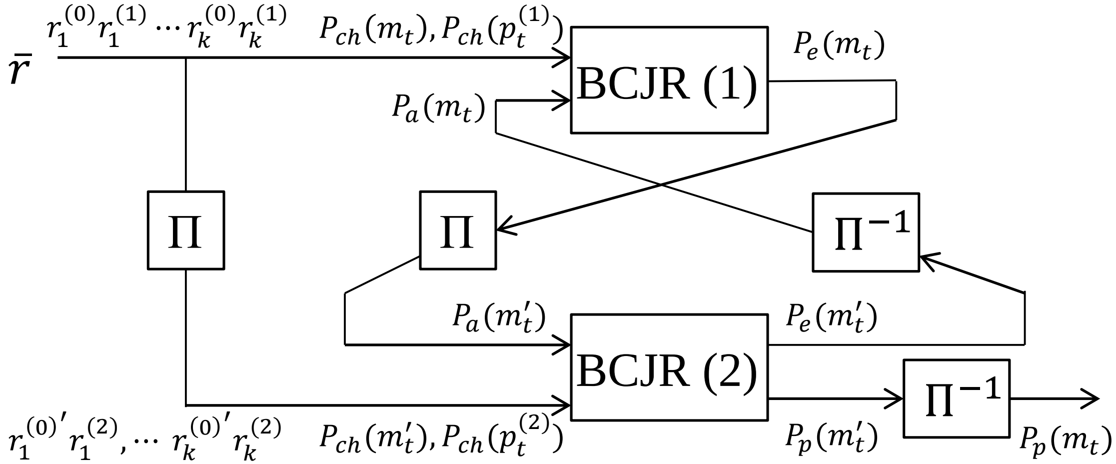
$$\Gamma_{\Omega\to\Omega'} = P_a(m_t)P_{ch}(m_t)P_{ch}(p_t^{(1)}).$$

In BCJR (2), trellis transition probability is determined by

$$\Gamma_{\Omega\to\Omega'} = P_a(m_t')P_{ch}(m_t')P_{ch}(p_t^{(2)}).$$

§ 4.3 Decoding of Turbo Codes

Turbo decoding structure



- At the beginning of iterations, knowledge of information bits m_t is not available, and $P_a(m_t)$ are initialized as

$$P_a(m_t = 0) = P_a(m_t = 1) = ½.$$

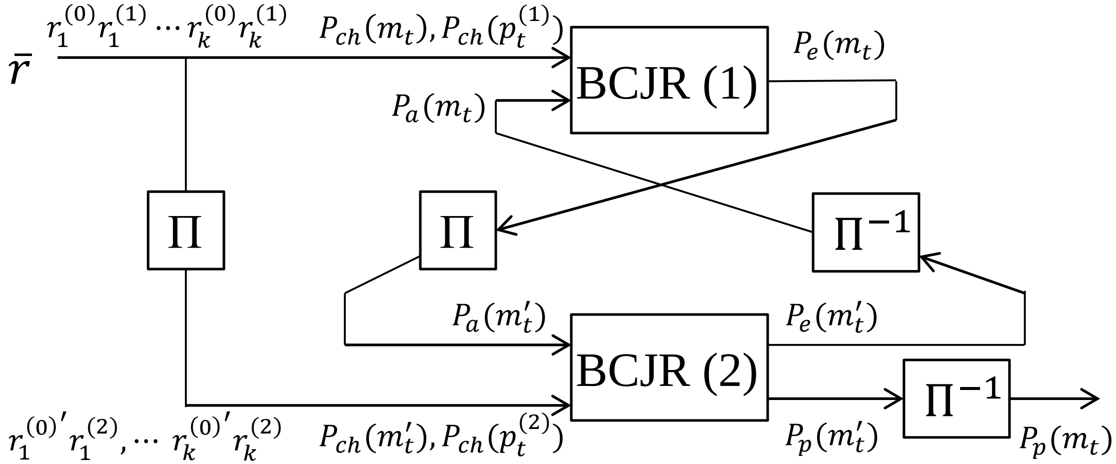
- Once BCJR (1) delivers $P_e(m_t)$, knowledge of interleaved information bits m'_t will be gained by mapping

$$\Pi(P_e(m_t)) \rightarrow P_a(m'_t),$$

and BCJR (2) starts its decoding with $P_a(m'_t)$, $P_{ch}(m'_t)$ and $P_{ch}(p_t^{(2)})$.

§ 4.3 Decoding of Turbo Codes

Turbo decoding structure



- Once BCJR (2) delivers $P_e(m_t')$, knowledge of information bits m_t will be gained by mapping

$$\Pi^{-1}(P_e(m_t')) \rightarrow P_a(m_t),$$

  and BCJR (1) performs another round of decoding with $P_a(m_t)$, $P_{ch}(m_t)$ and $P_{ch}(p_t^{(1)})$.
- After a sufficient number of iterations, decision will be made based on the *a posteriori* prob. $P_p(m_t)$ that is the deinterleaved version of output of BCJR (2), $P_p(m_t')$.
- If parity bits $p_t^{(1)}$ (or $p_t^{(2)}$) have been punctured, the channel observations become

$$P_{ch}\left(p_t^{(1)} = 0\right) = P_{ch}\left(p_t^{(1)} = 1\right) = ½, \text{(or } P_{ch}\left(p_t^{(2)} = 0\right) = P_{ch}\left(p_t^{(2)} = 1\right) = ½).$$

  And all the channel observations remain unchanged during the whole iterative process.

§ 4.3 Decoding of Turbo Codes

Example 4.2. Message vector $\bar{m} = [1\ 0\ 0\ 1\ 0\ 1\ 1\ 0\ 0\ 0]$

Transmitted codeword $\bar{c} = [11\ 00\ 01\ 11\ 01\ 10\ 10\ 00\ 00\ 01]$

Received symbol $\bar{r}$ = [1.66, 2.49, -2.35, -1.39, 0.22, 1.27, -0.41, 0.30,

-2.00, 1.16, 1.70, -1.69, 0.90, -0.38, -3.28, -0.82, 0.12, -1.30, -3.31, 2.28].

After iteration 1:

$P_e(m_t = 0)$	0.01	0.32	0.99	0.04	0.84	0.69	0.37	0.32	0.92	0.32
$P_e(m_t = 1)$	0.99	0.68	0.01	0.96	0.16	0.31	0.63	0.68	0.08	0.68
$P_e(m_t' = 0)$	0.50	0.82	0.50	0.08	0.50	0.67	0.50	0.23	0.50	0.03
$P_e(m_t' = 1)$	0.50	0.18	0.50	0.92	0.50	0.33	0.50	0.77	0.50	0.97
$P_p(m_t = 0)$	0.00	0.27	1.00	0.49	1.00	0.31	0.28	0.02	1.00	0.07
$P_p(m_t = 1)$	1.00	0.73	0.00	0.51	0.00	0.69	0.72	0.98	0.00	0.93

§ 4.3 Decoding of Turbo Codes

After iteration 2:

$P_e(m_t = 0)$	0.00	0.93	0.99	0.01	0.91	0.07	0.37	0.93	0.93	0.93
$P_e(m_t = 1)$	1.00	0.07	0.01	0.99	0.09	0.93	0.63	0.07	0.07	0.07
$P_e(m_t' = 0)$	0.50	0.99	0.50	0.04	0.50	0.14	0.50	0.34	0.50	0.01
$P_e(m_t' = 1)$	0.50	0.01	0.50	0.96	0.50	0.86	0.50	0.66	0.50	0.99
$P_p(m_t = 0)$	0.00	0.92	1.00	0.11	1.00	0.01	0.28	0.32	1.00	0.68
$P_p(m_t = 1)$	1.00	0.08	0.00	0.89	0.00	0.99	0.72	0.68	0.00	0.32

After iteration 3:

$P_e(m_t = 0)$	0.00	0.97	0.99	0.01	0.94	0.04	0.37	0.96	0.93	0.96
$P_e(m_t = 1)$	1.00	0.03	0.01	0.99	0.06	0.96	0.63	0.04	0.07	0.04
$P_e(m_t' = 0)$	0.50	0.99	0.50	0.03	0.50	0.09	0.50	0.37	0.50	0.01
$P_e(m_t' = 1)$	0.50	0.01	0.50	0.97	0.50	0.91	0.50	0.63	0.50	0.99
$P_p(m_t = 0)$	0.00	0.96	1.00	0.10	1.00	0.00	0.28	0.49	1.00	0.82
$P_p(m_t = 1)$	1.00	0.04	0.00	0.90	0.00	1.00	0.72	0.51	0.00	0.18

§ 4.3 Decoding of Turbo Codes

After iteration 4:

$P_e(m_t=0)$	0.00	0.97	0.99	0.01	0.94	0.04	0.37	0.97	0.93	0.97
$P_e(m_t=1)$	1.00	0.03	0.01	0.99	0.06	0.96	0.63	0.03	0.07	0.03
$P_e(m_t'=0)$	0.50	0.99	0.50	0.03	0.50	0.09	0.50	0.37	0.50	0.01
$P_e(m_t'=1)$	0.50	0.01	0.50	0.97	0.50	0.91	0.50	0.63	0.50	0.99
$P_p(m_t=0)$	0.00	0.97	1.00	0.10	1.00	0.00	0.28	0.51	1.00	0.82
$P_p(m_t=1)$	1.00	0.03	0.00	0.90	0.00	1.00	0.72	0.49	0.00	0.18

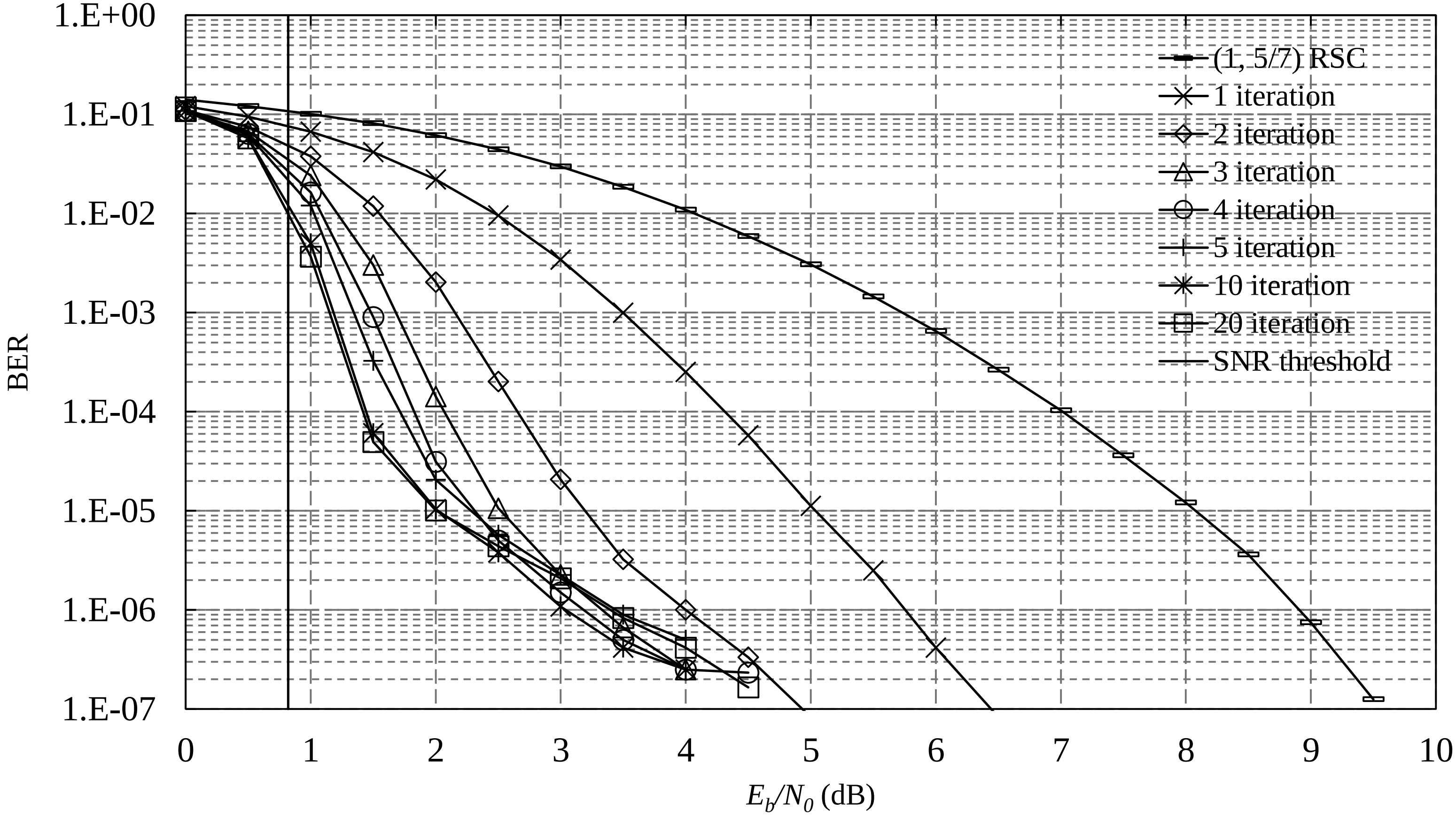
§ 4.3 Decoding of Turbo Codes

Remark: Turbo decoding efficiency can be improved by the so called log-MAP algorithm or the max-log-MAP algorithm [2]. Both of the algorithms deal with log-likelihood ratios rather than probabilities. The max-log-MAP algorithm has a computational complexity of not more than three times of Viterbi algorithm, but suffers a slight performance loss compared to BCJR and log-MAP algorithms.

[2] T. K. Moon, *Error correction coding-Mathematical Methods and Algorithms.*, John Wiley & Sons Press, 2005.

§ 4.4 Performance Analysis

BER performance of rate half turbo code with constituent code of (1, 5/7) RSC over AWGN channel using BPSK.



§ 4.4 Performance Analysis

Q: Why there is an error floor?

- The bit error rate (BER) (denoted as P_b) of a conv. code (and turbo code) is determined by

$$P_b \leq \sum_{i=1}^{2^k} \frac{w_i}{k} Q\left(\sqrt{\frac{2d_i \cdot R \cdot E_b}{N_0}}\right).$$

  - Let $\bar{m}_i$ denote a message vector and $\bar{c}_i$ denote its corresponding codeword, $w_i = weight(\bar{m}_i)$ and $d_i = weight(\bar{c}_i)$.
  - $k = length(\bar{m}_i)$ and there are 2^k codewords in the code book.
  - R is the rate of the code.
  - $\frac{E_b}{N_0}$ — signal-to-noise ratio (SNR).

$$\boxed{Q \text{ function as } Q(x) \triangleq \frac{1}{\sqrt{2\pi}} \int_x^{\infty} e^{-\frac{u^2}{2}} du.}$$

§ 4.4 Performance Analysis

- Since $d_i = d_{free}, d_{free}+1, \ \cdots, \ {}^{k}/_{R}$, by grouping terms with the same d_i, the above inequality can be written as:

$$P_b \le \sum_{d=d_{free}}^{k/R} \frac{W_d}{k} Q\left(\sqrt{\frac{2d\cdot R\cdot E_b}{N_0}}\right)$$

$$= \sum_{d=d_{free}}^{k/R} \frac{\hat{w}_d N_d}{k} Q\left(\sqrt{\frac{2d\cdot R\cdot E_b}{N_0}}\right).$$

  - $\hat{w}_d$ — weight of message vectors that correspond to codeword of weight d.
  - N_d — Number of codewords of weight d.
  - W_d — Total weight of message vectors that correspond to codeword of weight d.

§ 4.4 Performance Analysis

- When the SNR ($\frac{E_b}{N_0}$) increases, the asymptotic behavior of P_b is dominated by the first term in the summation as

$$P_b \cong \frac{N_{d_{free}} \widehat{w}_{d_{free}}}{k} Q\left(\sqrt{\frac{2d_{free} \cdot R \cdot E_b}{N_0}}\right).$$

- In the $\log P_b$ vs. $\log \frac{E_b}{N_0}$ graph, d_{free} determines the slope of the BER vs. SNR (dB) curve.

A: The error floor at high SNR is due to a small $\mathbf{d}_{free}$, or alternatively the presence of low weight codewords.

§ 4.4 Performance Analysis

Motivation of having an interleaver between the two encoders: Try to avoid the low weight conv. codewords and subsequently the low weight turbo codeword being produced.

Example 4.3 Following the encoder structure of Example 4.1, if the message vector $\bar{m} = [0\ 0\ 0\ 0\ 1]$**,** the output of the RSC (1) will be

$$\bar{c}_1 = [00\ 00\ 00\ 00\ 11].$$

Without interleaving, the output of RSC (2) will be the same as RSC (1) as $\bar{c}_2 = \bar{c}_1$. And the turbo codeword is

$$\bar{c} = [000\ 000\ 000\ 000\ 111].$$

With interleaving, $\bar{m}' = [1\ 0\ 0\ 0\ 0]$, the output of RSC (2) will now be

$$\bar{c}_2 = [11\ 00\ 01\ 00\ 01].$$

And the turbo codeword becomes

$$\bar{c} = [001\ 000\ 001\ 000\ 111].$$