Chapter 6 Turbo Codes

- 6.1 Introduction of Turbo Codes
- 6.2 Encoding of Turbo Codes
- 6.3 Decoding of Turbo Codes (Turbo Decoding)
- 6.4 Performance Analysis

§ 6.1 Introduction of Turbo Codes

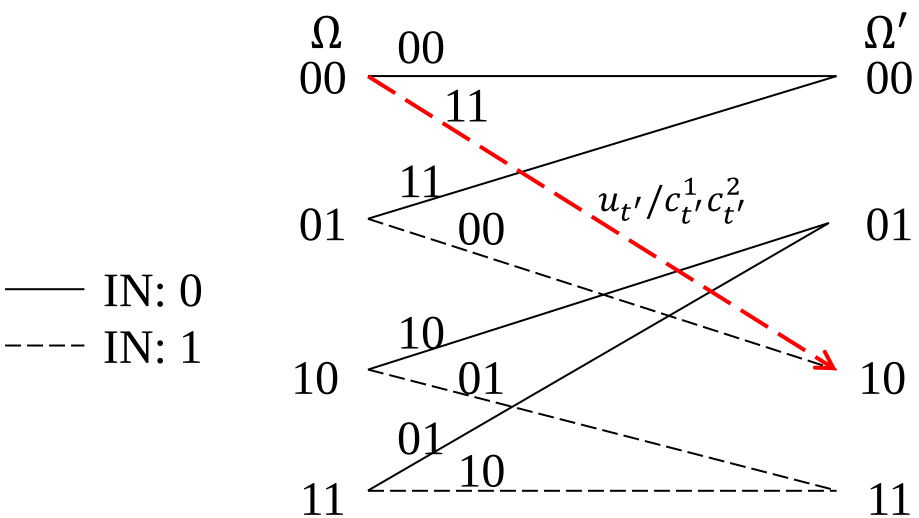
- Invented by C. Berrou, A. Glavieux and P. Thitimajshima in 1993 [1].
- Integrate a couple of conv. codes in a parallel encoding structure. The two conv. codes are called the constituent codes of a turbo code.
- Exploit the interplay between the decoders of the two constituent codes in a soft information exchange decoding mechanism.
- Such a decoding mechanism is called turbo decoding, turbo decoding is NOT limited to decode turbo codes, but to any (serially or parallelly) concatenated 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.

§ 6.1 Introduction of Turbo Codes

Why do we need code concatenation?

In BCJR decoding of a conv. code,



a priori prob.

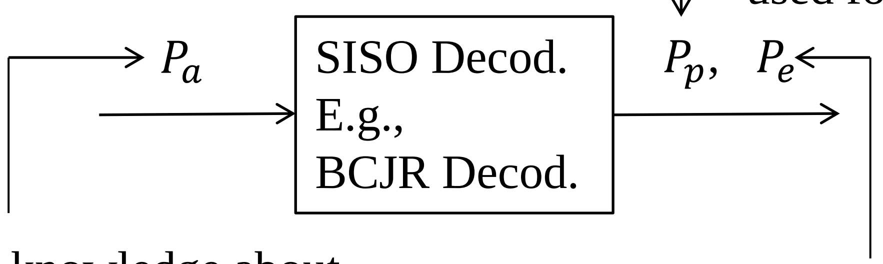
$$\Gamma_{\Omega\to\Omega'} = \boxed{P_a(u_{t'})} \cdot \underline{P_{\text{ch}}(c_{t'}^1) \cdot P_{\text{ch}}(c_{t'}^2)}$$

channel observations

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

§ 6.1 Introduction of Turbo Codes

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



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

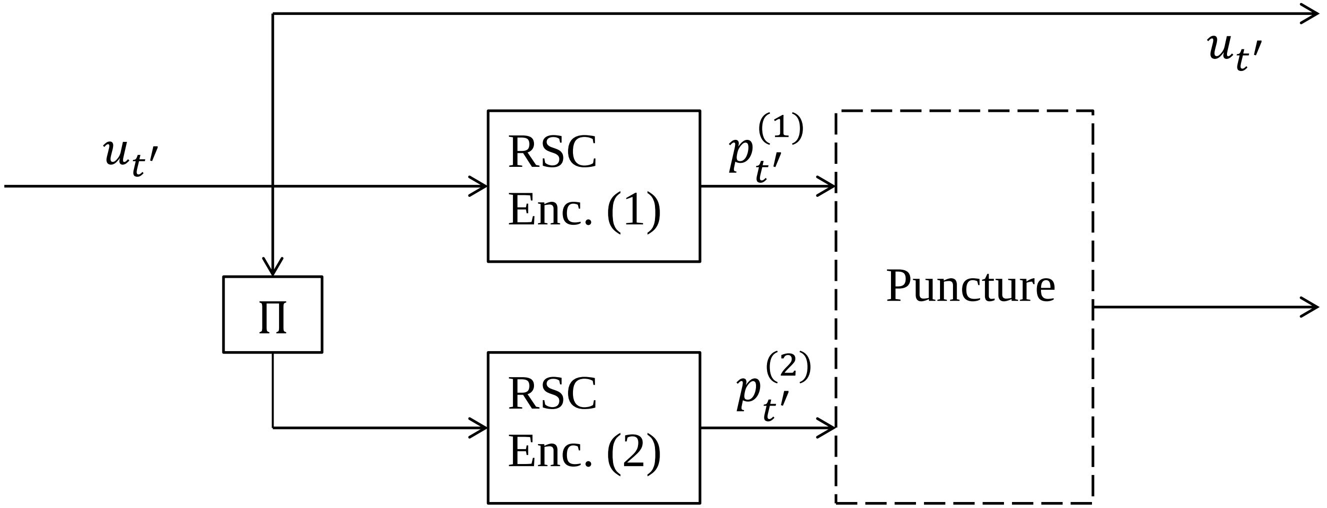
Extrinsic prob.: $P_e = \frac{P_p}{P_a}$, the extra knowledge (excluding the *a priori* prob.) delivered by the SISO decoder.

§ 6.2 Encoding of Turbo Codes

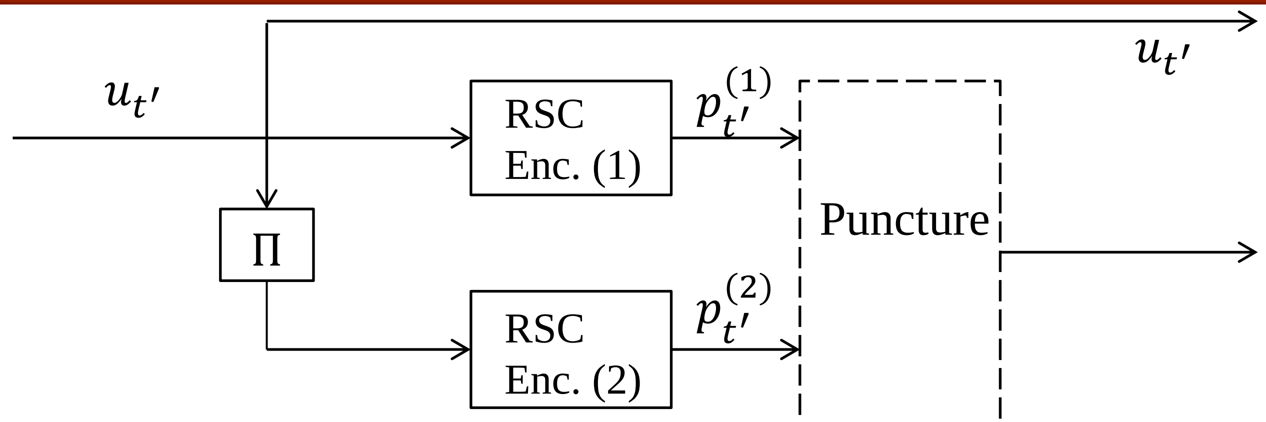
Constituent codes: Recursive Systematic Conv. (RSC) codes. Normally, the two constituent codes are the same.

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 code rate.



§ 6.2 Encoding of Turbo Codes



- Given the binary message sequence as $\bar{u} = [u_1,\ u_2,\ \cdots\ u_k]$, output of the turbo encoder should be

$$\bar{c} = [u_1\ p_1^{(1)}\ p_1^{(2)}\ \ u_2\ p_2^{(1)}\ p_2^{(2)}\ \cdots\ u_{t'}\ p_{t'}^{(1)}\ p_{t'}^{(2)}\ \cdots\ u_k\ p_k^{(1)}\ p_k^{(2)}].$$

- Rate of the turbo code is 1/3. To increase the rate to 1/2, 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.

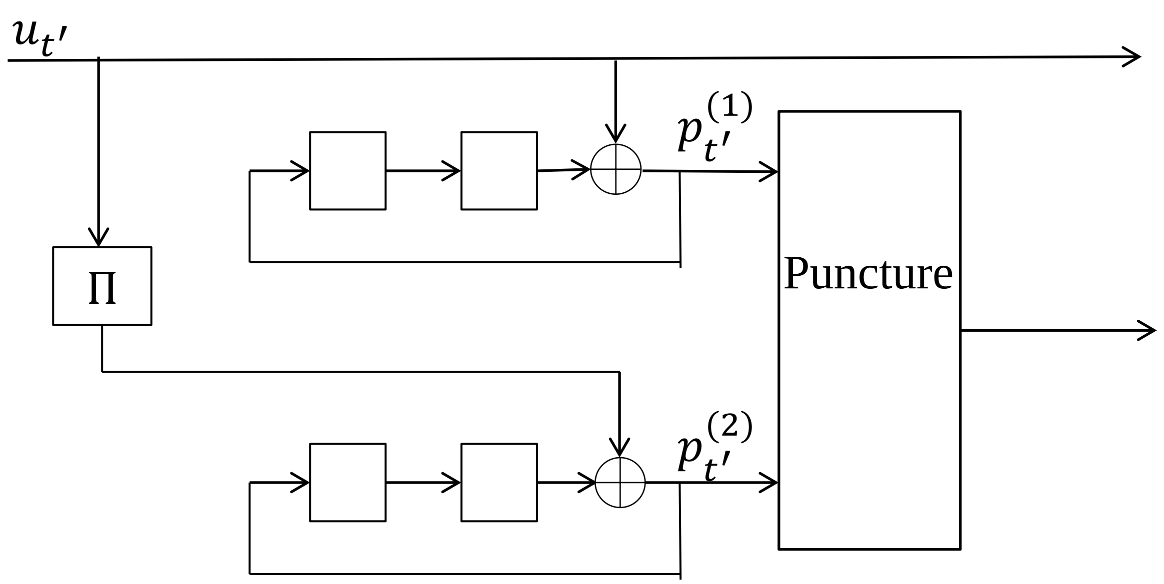
- After puncturing, output of the turbo encoder should be

$$\bar{c} = [u_1\ p_1^{(1)}\ \ u_2\ p_2^{(2)}\ \ \cdots\ \underline{u_k\ p_k^{(1)}}(\underline{u_k\ p_k^{(2)}})]$$

when k is odd — when k is even.

§ 6.2 Encoding of Turbo Codes

Example 6.1 Given the turbo encoder shown below with constituent code of the $(1, 1/5)_8$ conv. code. 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 $\bar{u}$ = [1 0 0 1 0 1 1 0 0 0].



§ 6.2 Encoding of Turbo Codes

The original message vector

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

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

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

Output of the 2nd constituent code is:

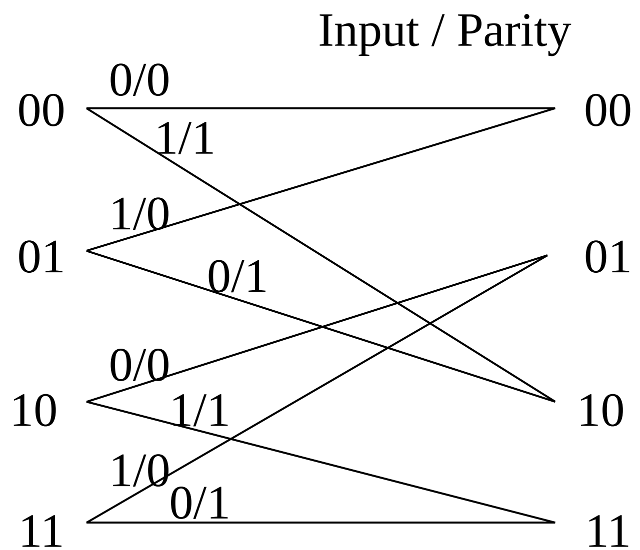
$$\bar{p}^{(2)} = [0\ 0\ 1\ 1\ 1\ 0\ 1\ 0\ 1\ 1]$$

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 $(1, 1/5)_8$ conv. code

§ 6.3 Decoding of Turbo Codes

- **Parameterization**
  - Turbo codeword $\bar{c} = [u_1\, p_1^{(1)}\, p_1^{(2)},\ u_2\, p_2^{(1)}\, p_2^{(2)},\ \cdots,\ u_k\, p_k^{(1)}\, p_k^{(2)}]$.
  - Assume the turbo codeword is transmitted using BPSK.
  - Received symbol vector

$$\bar{y} = [y_1^{(0)}\, y_1^{(1)}\, y_1^{(2)}, y_2^{(0)}\, y_2^{(1)}\, y_2^{(2)},\ \cdots, y_k^{(0)}\, y_k^{(1)}\, y_k^{(2)}].$$

  - Interleaved message vector

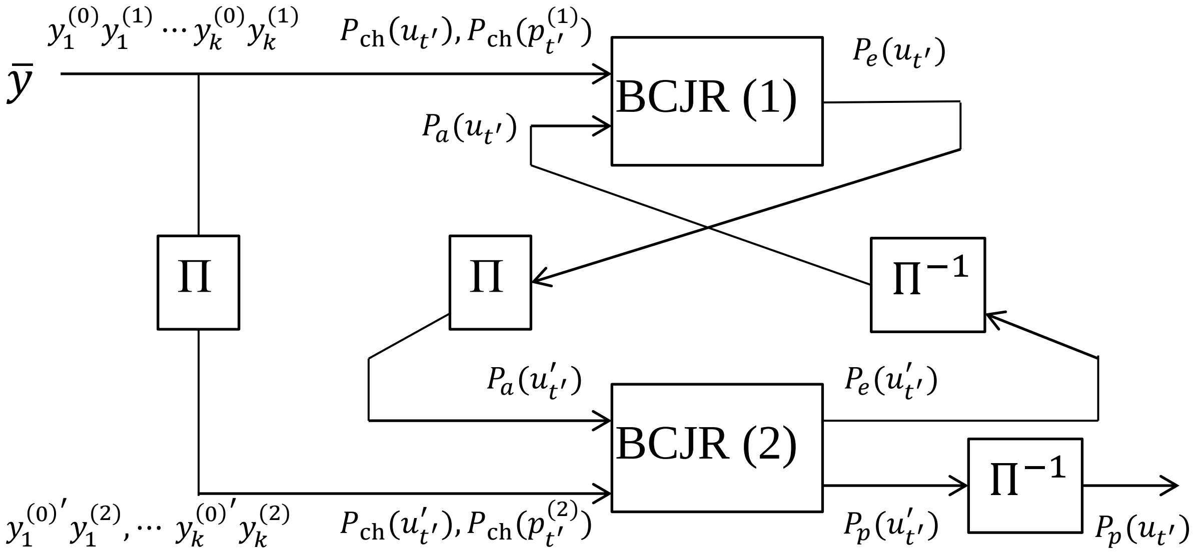
$$\bar{u}' = \Pi(\bar{u}) = [u_1',\ u_2',\ \cdots,\ u_k'].$$

  - Interleaved (information) symbol vector

$$\left[y_1^{(0)\prime},\ y_2^{(0)\prime}, \cdots,\ y_k^{(0)\prime}\right] = \Pi([y_1^{(0)},\ y_2^{(0)},\ \cdots,\ y_k^{(0)}]).$$

§ 6.3 Decoding of Turbo Codes

Turbo decoding structure



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

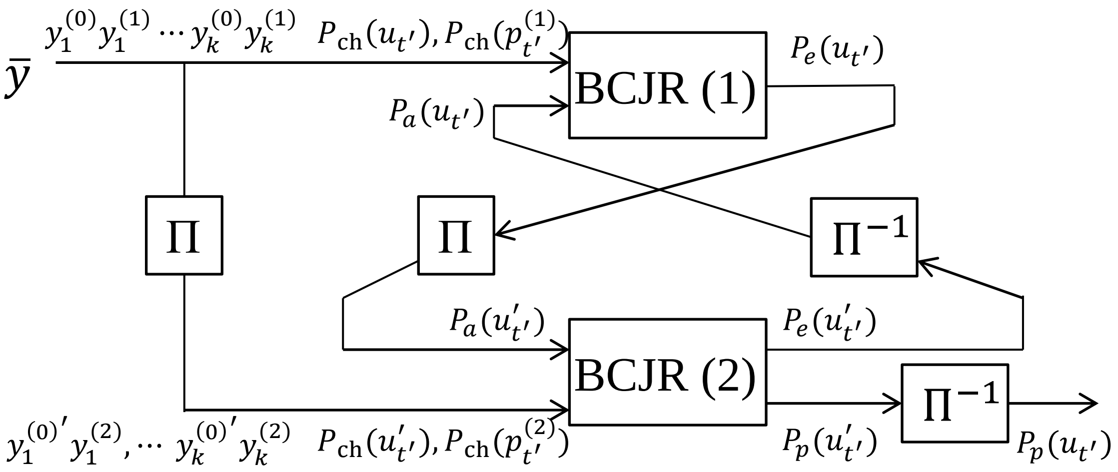
$$\Gamma_{\Omega\to\Omega'} = P_a(u_{t'})P_{\text{ch}}(u_{t'})P_{\text{ch}}(p_{t'}^{(1)}).$$

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

$$\Gamma_{\Omega\to\Omega'} = P_a(u'_{t'})P_{\text{ch}}(u'_{t'})P_{\text{ch}}(p_{t'}^{(2)}).$$

§ 6.3 Decoding of Turbo Codes

Turbo decoding structure



- At the beginning of iterations, knowledge of information bits $u_{t'}$ is not available, and $P_a(u_{t'})$ are initialized as

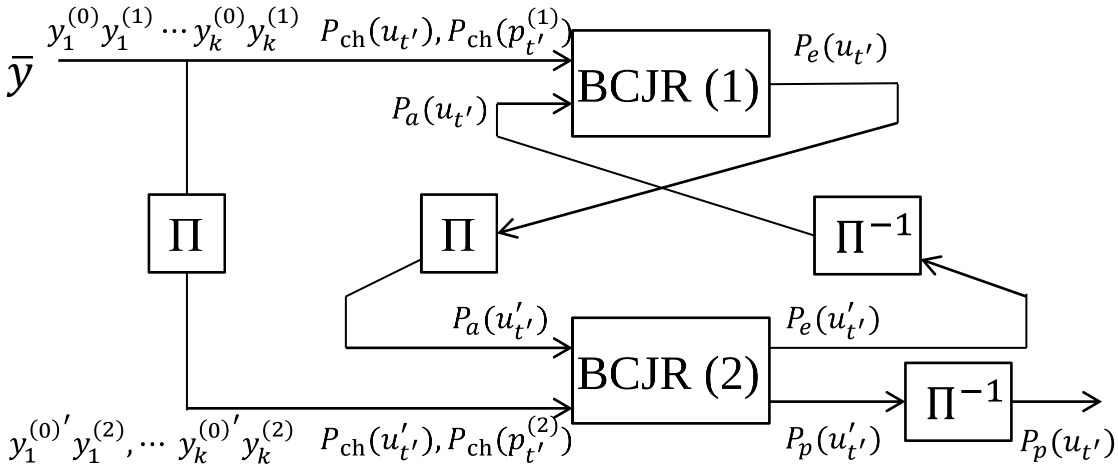
$$P_a(u_{t'} = 0) = P_a(u_{t'} = 1) = 1/2.$$

- Once BCJR (1) delivers $P_e(u_{t'})$, knowledge of interleaved information bits $u'_{t'}$ will be gained by mapping

$$\Pi(P_e(u_{t'})) \rightarrow P_a(u'_{t'}),$$

and BCJR (2) starts its decoding with $P_a(u'_{t'})$, $P_{\text{ch}}(u'_{t'})$ and $P_{\text{ch}}(p_{t'}^{(2)})$.

§ 6.3 Decoding of Turbo Codes



- Once BCJR (2) delivers $P_e(u'_{t'})$, knowledge of information bits $u_{t'}$ will be gained by mapping

$$\Pi^{-1}(P_e(u'_{t'})) \rightarrow P_a(u_{t'}),$$

  and BCJR (1) performs another round of decoding with $P_a(u_{t'})$, $P_{\text{ch}}(u_{t'})$ and $P_{\text{ch}}(p_{t'}^{(1)})$.
- After a sufficient number of iterations, decisions will be made based on the *a posteriori* prob. $P_p(u_{t'})$ that are the deinterleaved version of output of BCJR (2), $P_p(u'_{t'})$.
- If parity bits $p_{t'}^{(1)}$ (or $p_{t'}^{(2)}$) have been punctured, the channel observations become

$$P_{\text{ch}}\left(p_{t'}^{(1)} = 0\right) = P_{\text{ch}}\left(p_{t'}^{(1)} = 1\right) = 1/2, \text{(or } P_{\text{ch}}\left(p_{t'}^{(2)} = 0\right) = P_{\text{ch}}\left(p_{t'}^{(2)} = 1\right) = 1/2).$$

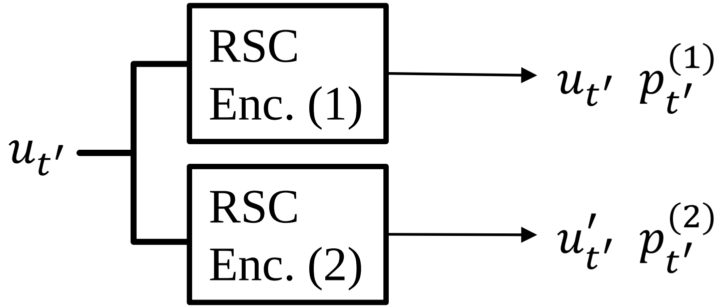
  And all the channel observations remain unchanged during the whole iterative process.

§ 6.3 Decoding of Turbo Codes

Advantage of systematic constituent codes

Using RSC

Encoding:



Transmission (coding rate is 1/3): $u_1\ p_1^{(1)} p_1^{(2)}\ u_2\ p_2^{(1)} p_2^{(2)}\ u_3\ p_3^{(1)} p_3^{(2)}\ \dots\dots\ u_{t'}\ p_{t'}^{(1)} p_{t'}^{(2)}\ \dots\dots$

Decoding of the constituent codes:

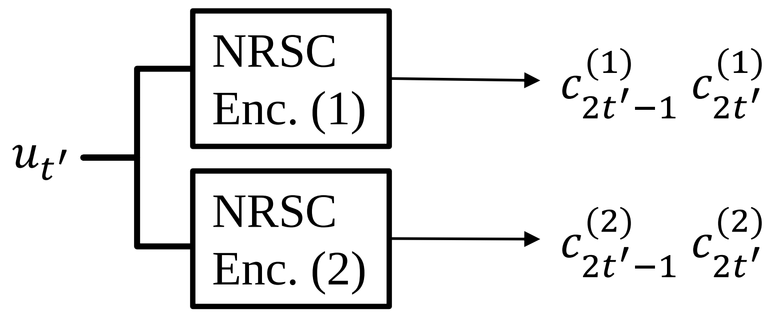
$P_{\text{ch}}(u_{t'}), P_{\text{ch}}(p_{t'}^{(1)})$ are used in the 1st decoder, which is rate 1/2.

$P_{\text{ch}}(u'_{t'}), P_{\text{ch}}(p_{t'}^{(2)})$ are used in the 2nd decoder, which is rate 1/2.

§ 6.3 Decoding of Turbo Codes

<u>Using non-systematic constituent codes</u>

Encoding:



Transmission
(coding rate is 1/4): $c^{(1)}_1\ c^{(1)}_2\ c^{(2)}_1\ c^{(2)}_2\ c^{(1)}_3\ c^{(1)}_4\ c^{(2)}_3\ c^{(2)}_4 \ \ldots\ldots\ c^{(1)}_{2t'-1}\ c^{(1)}_{2t'}\ c^{(2)}_{2t'-1}\ c^{(2)}_{2t'} \ \ldots\ldots$

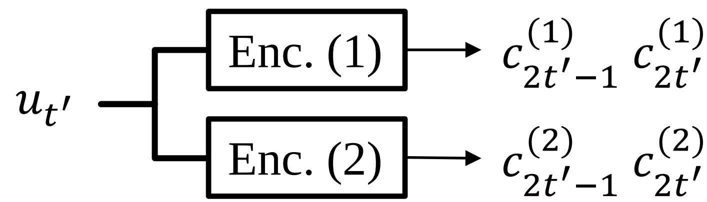
Decoding of the constituent codes :

$P_{\text{ch}}\left(c^{(1)}_{2t'-1}\right), P_{\text{ch}}(c^{(1)}_{2t'})$ are used in the 1st decoder, which is rate 1/2.

$P_{\text{ch}}\left(c^{(2)}_{2t'-1}\right), P_{\text{ch}}(c^{(2)}_{2t'})$ are used in the 2nd decoder, which is rate 1/2.

§ 6.3 Decoding of Turbo Codes

<u>Using non-systematic constituent codes</u>



In realizing a rate 1/3 coded transmission, we may puncture one coded bit in each time instant. E.g.,(i) puncture $c_{2t'}^{(2)}$ as

$$c_1^{(1)}\ c_2^{(1)}\ c_1^{(2)}\ c_3^{(1)}\ c_4^{(1)}\ c_3^{(2)}\ \ldots\ldots\ c_{2t'-1}^{(1)}\ c_{2t'}^{(1)}\ c_{2t'-1}^{(2)}\ \ldots\ldots$$

or (ii) puncture $c_{2t'-1}^{(2)}$ when $t'$is odd, and $c_{2t'-1}^{(1)}$ when t' is even, as

$$c_1^{(1)}\ c_2^{(1)}\ c_2^{(2)}\ c_4^{(1)}\ c_3^{(2)}\ c_4^{(2)}\ \ldots\ldots\ c_{2t'-1}^{(1)}\ c_{2t'}^{(1)}\ c_{2t'}^{(2)}\ (c_{2t'}^{(1)}\ c_{2t'-1}^{(2)}\ c_{2t'}^{(2)})\ \ldots\ldots$$

Decoding of the constituent codes:

In case (i)

$P_{\text{ch}}\left(c_{2t'-1}^{(1)}\right), P_{\text{ch}}(c_{2t'}^{(1)})$ are used in the 1st decoder, which is rate 1/2.

$P_{\text{ch}}\left(c_{2t'-1}^{(2)}\right)$ are used in the 2nd decoder, which is forced to be rate 1.

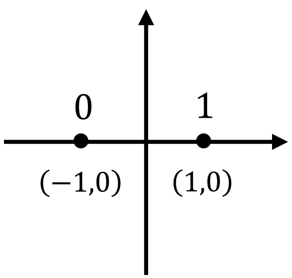
In case (ii)

The 1st and 2nd decoders are forced to be rate 2/3.

§ 6.3 Decoding of Turbo Codes

Example 6.2 Message vector $\bar{u}$ = [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 $\bar{y}$ = [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:

$P_e(u_{t'}=0)$	0.01	0.32	0.99	0.04	0.84	0.69	0.37	0.32	0.92	0.32
$P_e(u_{t'}=1)$	0.99	0.68	0.01	0.96	0.16	0.31	0.63	0.68	0.08	0.68
$P_e(u'_{t'}=0)$	0.50	0.82	0.50	0.08	0.50	0.67	0.50	0.23	0.50	0.03
$P_e(u'_{t'}=1)$	0.50	0.18	0.50	0.92	0.50	0.33	0.50	0.77	0.50	0.97
$P_p(u_{t'}=0)$	0.00	0.27	1.00	0.49	1.00	0.31	0.28	0.02	1.00	0.07
$P_p(u_{t'}=1)$	1.00	0.73	0.00	0.51	0.00	0.69	0.72	0.98	0.00	0.93

Note*:* Only the real part of the received symbols are preserved here.

§ 6.3 Decoding of Turbo Codes

After iteration 2:

$P_e(u_{t'} = 0)$	0.00	0.93	0.99	0.01	0.91	0.07	0.37	0.93	0.93	0.93
$P_e(u_{t'} = 1)$	1.00	0.07	0.01	0.99	0.09	0.93	0.63	0.07	0.07	0.07
$P_e(u'_{t'} = 0)$	0.50	0.99	0.50	0.04	0.50	0.14	0.50	0.34	0.50	0.01
$P_e(u'_{t'} = 1)$	0.50	0.01	0.50	0.96	0.50	0.86	0.50	0.66	0.50	0.99
$P_p(u_{t'} = 0)$	0.00	0.92	1.00	0.11	1.00	0.01	0.28	0.32	1.00	0.68
$P_p(u_{t'} = 1)$	1.00	0.08	0.00	0.89	0.00	0.99	0.72	0.68	0.00	0.32

After iteration 3:

$P_e(u_{t'} = 0)$	0.00	0.97	0.99	0.01	0.94	0.04	0.37	0.96	0.93	0.96
$P_e(u_{t'} = 1)$	1.00	0.03	0.01	0.99	0.06	0.96	0.63	0.04	0.07	0.04
$P_e(u'_{t'} = 0)$	0.50	0.99	0.50	0.03	0.50	0.09	0.50	0.37	0.50	0.01
$P_e(u'_{t'} = 1)$	0.50	0.01	0.50	0.97	0.50	0.91	0.50	0.63	0.50	0.99
$P_p(u_{t'} = 0)$	0.00	0.96	1.00	0.10	1.00	0.00	0.28	0.49	1.00	0.82
$P_p(u_{t'} = 1)$	1.00	0.04	0.00	0.90	0.00	1.00	0.72	0.51	0.00	0.18

§ 6.3 Decoding of Turbo Codes

After iteration 4:

$P_e(u_{t'} = 0)$	0.00	0.97	0.99	0.01	0.94	0.04	0.37	0.97	0.93	0.97
$P_e(u_{t'} = 1)$	1.00	0.03	0.01	0.99	0.06	0.96	0.63	0.03	0.07	0.03
$P_e(u'_{t'} = 0)$	0.50	0.99	0.50	0.03	0.50	0.09	0.50	0.37	0.50	0.01
$P_e(u'_{t'} = 1)$	0.50	0.01	0.50	0.97	0.50	0.91	0.50	0.63	0.50	0.99
$P_p(u_{t'} = 0)$	0.00	0.97	1.00	0.10	1.00	0.00	0.28	0.51	1.00	0.82
$P_p(u_{t'} = 1)$	1.00	0.03	0.00	0.90	0.00	1.00	0.72	0.49	0.00	0.18

<u>Observations</u>:

(i) The iterative decoding corrects 3 errors;

(ii) Let $L_p(u_{t'}) = \ln \frac{P_p(u_{t'}=0)}{P_p(u_{t'}=1)}$. If $|L_p(u_{t'})|$ is greater, the decision made on $u_{t'}$ will be more confident;

(iii) The decisions made on the message bits become more confident as the iteration progresses.

§ 6.3 Decoding of Turbo Codes

Log-MAP BCJR

Define function

$$\max{}^*(x,y) \triangleq \ln(e^x + e^y) = \max(x,y) + \ln\left(1 + e^{-|x-y|}\right)$$

whose multivariate version can be defined recursively as

$$\max{}^*(x,y,z) \triangleq \ln(e^x + e^y + e^z) = \max{}^*(\max{}^*(x,y)\,, z)$$

Similarly, let X be a finite set so that

$$\max{}^*(X) \triangleq \ln \sum_{x \in X} e^x$$

Define BCJR metrics in log domain

$$\Gamma^*_{\Omega \to \Omega'} = \ln \Gamma_{\Omega \to \Omega'}$$

$$A^*_{t'}(\Omega) = \ln A_{t'}(\Omega) = \ln \sum_{(\Omega_0, \Omega)} e^{\left[A^*_{t'-1}(\Omega_0) + \Gamma^*_{\Omega_0 \to \Omega}\right]}$$

$$= \max{}^*\left(\left\{A^*_{t'-1}(\Omega_0) + \Gamma^*_{\Omega_0 \to \Omega} \mid \forall(\Omega_0, \Omega)\right\}\right)$$

§ 6.3 Decoding of Turbo Codes

Similarly, $B^*_{t'+1}(\Omega') = \ln B_{t'+1}(\Omega') = \ln \sum_{(\Omega',\Omega'')} e^{\left[B^*_{t'+2}(\Omega'')+\Gamma^*_{\Omega'\to\Omega''}\right]}$

$$= \max{}^*\left(\left\{B^*_{t'+2}(\Omega'') + \Gamma^*_{\Omega'\to\Omega''} \mid \forall(\Omega',\Omega'')\right\}\right)$$

$$L(u_{t'}) = \max_{(\Omega\to\Omega')_0}{}^* \left(A^*_{t'}(\Omega) + \Gamma^*_{\Omega\to\Omega'} + B^*_{t'+1}(\Omega')\right)$$
$$- \max_{(\Omega\to\Omega')_1}{}^* \left(A^*_{t'}(\Omega) + \Gamma^*_{\Omega\to\Omega'} + B^*_{t'+1}(\Omega')\right)$$

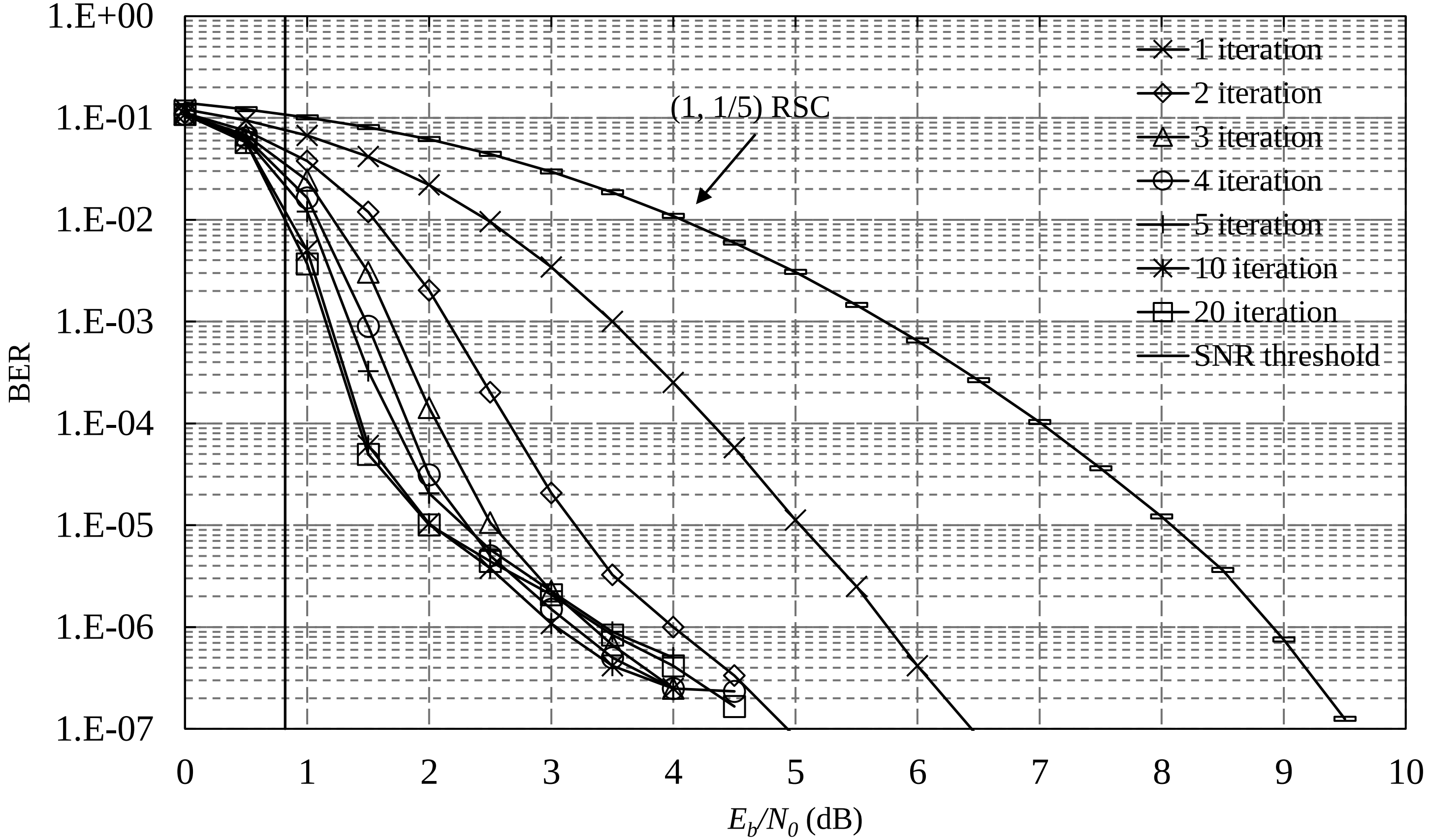
The max-log-MAP algorithm is to replace $\max^*(\cdot)$ by $\max(\cdot)$.

Remark: Turbo decoding efficiency can be improved by the so-called log-MAP algorithm [1] 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.

[1] C. Berrou and A. Glavieux, "Near optimum error correcting coding and decoding: turbo-codes," in *IEEE Transactions on Communications*, vol. 44, no. 10, pp. 1261-1271, Oct. 1996.
[2] J. Hagenauer, E. Offer und L. Papke, "Iterative Decoding of Binary Block and Convolutional Codes," *IEEE Transactions on Information Theory*, 42, 1996.

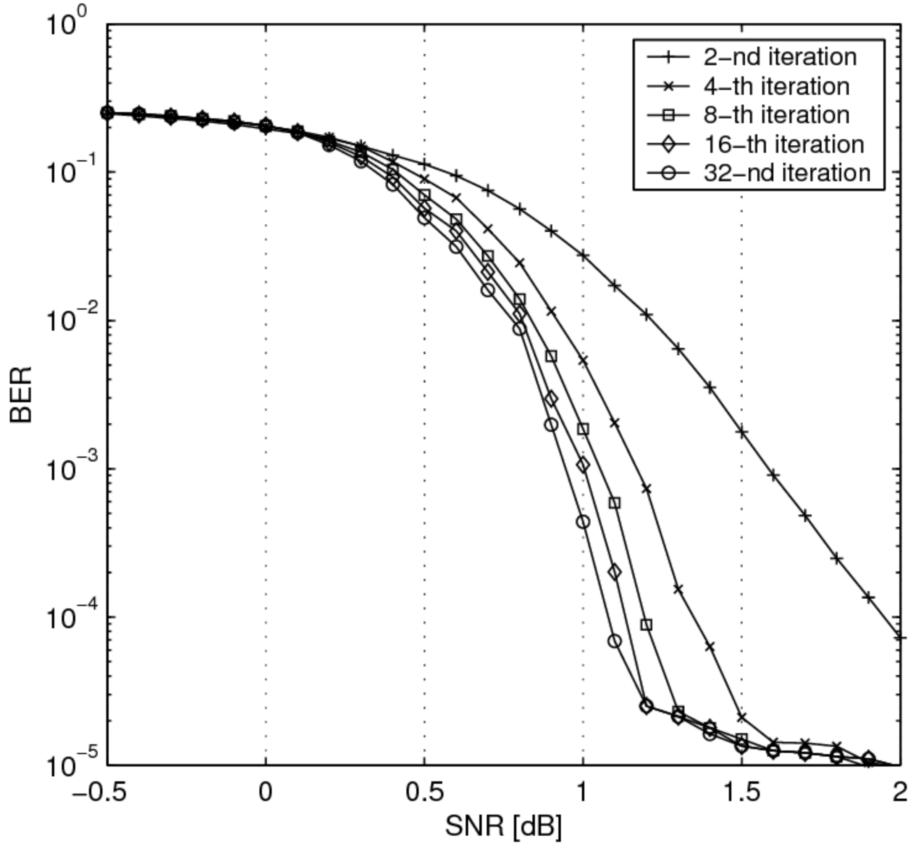
§ 6.4 Performance Analysis

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



§ 6.4 Performance Analysis

The BER performance of the classical turbo code with length 1024 and rate 1/3 [1].



[1] Tasev, Zarko & Popovski, Petar & Maggio, Gian & Kocarev, Ljupco. “Bifurcations and Chaos in Turbo Decoding Algorithms,” 2004.

§ 6.4 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

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

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

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

§ 6.4 Performance Analysis

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

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

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

- $\hat{w}_d$ — 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.

§ 6.4 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

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

- 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.

Remark: The error floor at high SNR is due to a small d_{free}, or alternatively the presence of low weight codewords.

§ 6.4 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 6.3 Following the encoder structure of *Example 6.1*, if the message vector $\bar{u} = [0\ 0\ 0\ 0\ 1]$**,** 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, $\bar{u}' = [1\ 0\ 0\ 0\ 0]$, the output of RSC (2) will now be

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

And the turbo codeword becomes

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