Week 6 : Session 1.

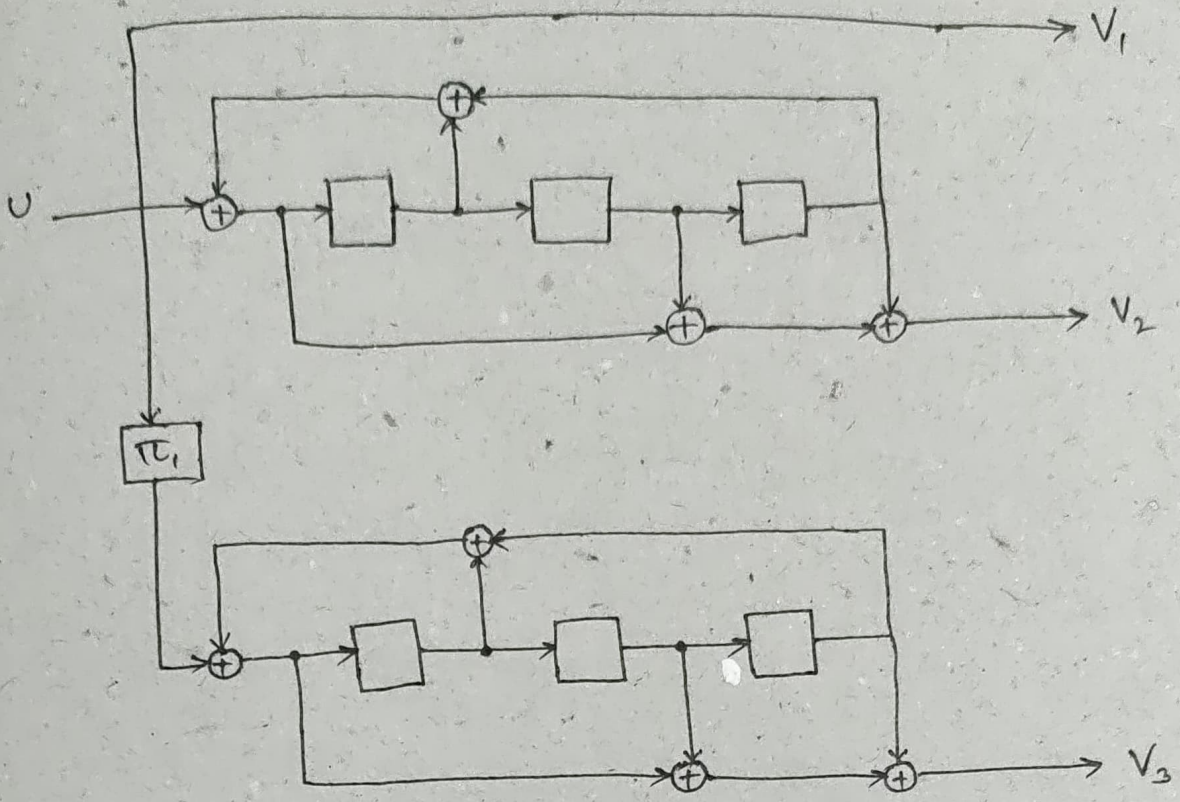
TURBO CODES : AN INTRODUCTION

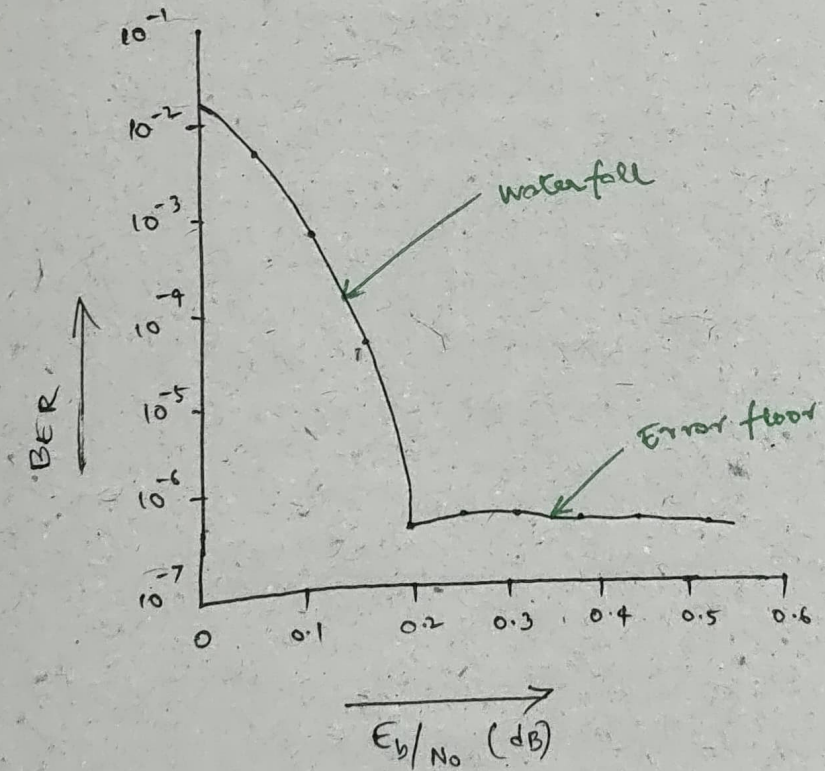
Turbo code is a class of concatenated codes. Using simple convolutional codes, we can create very powerful error correcting codes.

- Shannon's noisy channel coding theorem implies that, arbitrarily low decoding error probabilities can be achieved at any transmission rate R less than the channel capacity C, by using long block lengths.
- Shannon proved that the average performance of a randomly chosen ensemble of codes results in an exponentially decreasing decoding error probability with increasing block length.
- Typically, the code designs contain a large amount of structure, since it can be used to guarantee good minimum distance properties for the code as well as it is easier to decode a highly structured code. However, structure does not always result in the best distance properties for a code.
- Turbo codes have random like code design with just enough structure to allow for an efficient iterative decoding method.

Encoder structure consists of

- Recursive systematic convolutional encoder as constituent encoders in a parallel concatenation scheme.
- An interleaver denoted by π_1.
- Example of rate $R = \frac{1}{3}$ turbo code is shown below.





BER performance of rate $R = \frac{1}{3}$ turbo code with input information block length of 65536 bits.

- The best performance at moderate BER's, down to about 10^{-5} is achieved with short constraint length constituent encoders, typically $\nu = 4$ or less.
- If the constituent encoders of a turbo code are same, it is known as a Symmetric turbo code, otherwise it is an asymmetric turbo code.
- Recursive symmetric encoders give much better performance than non recursive systematic encoders when used as constituent encoders in a Turbo code.
- Bits can be punctured from the parity sequences in order to produce higher code rates.
- Bits can also be punctured from the information sequence. If some of the information bits are punctured, it is known as partially systematic turbo code. If all the information bits are punctured, it is known as non systematic turbo code.
- Additional constituent codes and interleavers can be used to produce lower rate codes. For example, rate $r = \frac{1}{4}$ can be achieved with three constituent codes and two interleavers. This is known as a multiple turbo code.
- The interleavers are usually constructed in a random fashion.
- Sub optimum iterative decoding, which employs individual Soft Input Soft Output (SISO) decoders for each of the constituent codes in an iterative manner, achieves performance typically within a few tenths of a dB of overall ML or MAP decoding.
- The best performance is obtained when the BCJR or MAP algorithm is used as the SISO decoder for each constituent code.
- Since MAP decoder uses a forward-backward algorithm, the information is arranged in blocks.

- The first constituent encoder is terminated by appending ν bits to return it to the all-zero state.
- Because the interleaver reorders the input sequence, the second encoder will not normally return to the zero state. However, if desired, modifications can be made to insure termination of both encoders.
- Block codes can also be used as constituent codes in turbo encoders.
- Turbo codes have few disadvantages:
  - a large decoding delay
  - significantly weakened performance at BER's below 10^{-6}.

TURBO DECODING

In the last section, we've seen the encoder structure of Turbo codes. In this section, we'll talk about how to decode Turbo codes.

Before that, we'll briefly talk about BCJR algorithm in log domain. We'll recap the discussion of BCJR algorithm because that's the underlying Soft Input and Soft Output algorithm used for Turbo decoding. And then we'll talk about Turbo decoding algorithm.

Outline:

- Review of BCJR algorithm in log domain
- Turbo decoding for rate $R = \frac{1}{3}$ code.

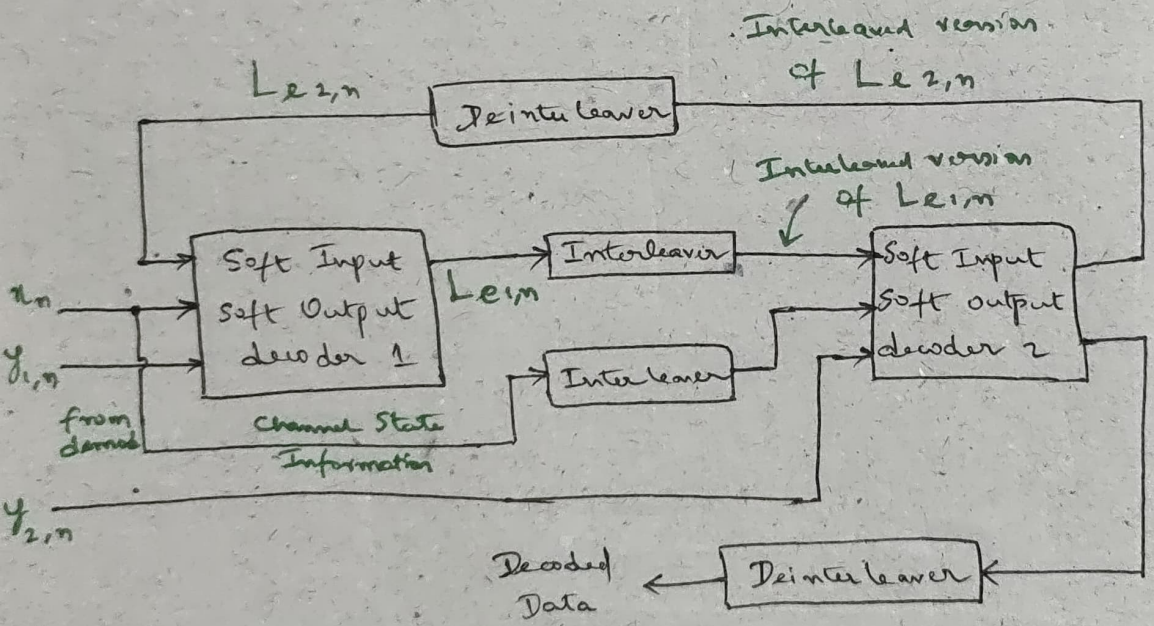


Fig: Block diagram of Turbo decoding algorithm

$L_e \rightarrow$ Extrinsic information

$L_{APP} \rightarrow$ APP value based on the MAP.

SISO decoder 1 : (INPUTS)

- x_n → Noisy version of the information sequence.
- $y_{1,n}$ → Noisy received version of the Parity sequence coming out of encoder 1.
- Third input to SISO decoder is the apriori knowledge that the second decoder is providing to the first decoder.

SISO decoder 2 : (INPUTS)

- Interleaved version of the Noisy received version of the information sequence (x_n).
- $y_{2,n}$ → Received parity bits corresponding to the second parity sequence.
- Apriori information from decoder 1.

BCJR' Algorithm :

The underlying algorithm which is used in Turbo decoding is the BCJR algorithm. It computes the aposteriori probability, using that we compute the extrinsic information. In the previous section, we talked about BCJR algorithm in the probability domain. We could implement the same in Log domain also.

- Define $\max^*(.)$ function :

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

$$\max^*(x,y,z) \equiv \ln\left(e^x + e^y + e^z\right) = \max^*\left[\max^*(x,y), z\right]$$

- Branch metrics :

$$\gamma_k^*(s',s) \equiv \ln \gamma_k(s',s) = \frac{u_k L_a(u_k)}{2} + \frac{L_c}{2} x_k v_k$$

⊙ Forward metrics :

$$\alpha^*_{k+1}(s) \equiv \ln \alpha_{k+1}(s) = \max^*_{s' \in \sigma_k} \left[\gamma^*_k(s', s) + \alpha^*_k(s') \right]$$

⊙ Forward metrics initialization :

$$\alpha^*_0(s) \equiv \ln \alpha_0(s) = \begin{cases} 0, & s = 0 \\ -\infty, & s \neq 0 \end{cases}$$

⊙ Backward metrics :

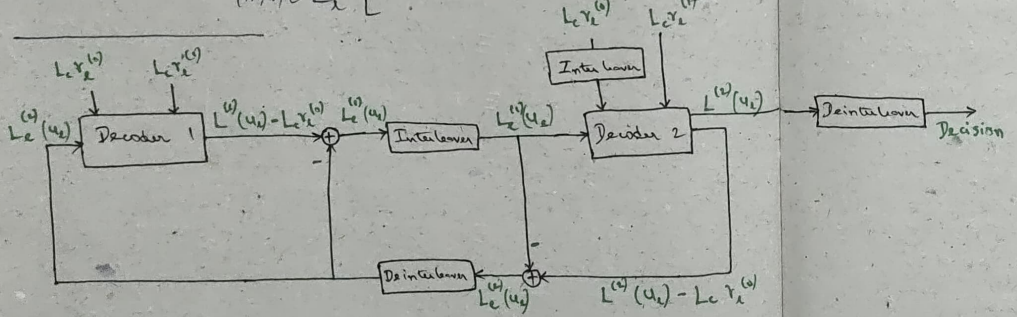
$$\beta^*_k(s') \equiv \ln \beta_k(s') = \max^*_{s \in \sigma_{k+1}} \left[\gamma^*_k(s', s) + \beta^*_{k+1}(s) \right]$$

⊙ Backward metrics initialization :

$$\beta^*_K(s) \equiv \ln \beta_K(s) = \begin{cases} 0, & s = 0 \\ -\infty, & s \neq 0 \end{cases}$$

⊙ APP L-Value :

$$L(u_k) = \max^*_{(s',s) \in \Sigma^+_k} \left[\beta^*_{k+1}(s) + \gamma^*_k(s', s) + \alpha^*_k(s') \right] - \max^*_{(s',s) \in \Sigma^-_k} \left[\beta^*_{k+1}(s) + \gamma^*_k(s', s) + \alpha^*_k(s') \right]$$



Turbo decoding

⊙ Figure shows the basic structure of an iterative turbo decoder for a rate 1/3 turbo code.

⊙ It employs two SISO decoders using the MAP algorithm.

⊙ At each time unit k, three output values are received from the channel,

- One for the information bit $u_k = v_k^{(0)}$, denoted $y_k^{(0)}$
- And two for the parity bits $v_k^{(1)}$ and $v_k^{(2)}$, denoted $y_k^{(1)}$ and $y_k^{(2)}$.

⊙ The $3K$ - dimensional received vector is denoted by

$$y = \left(y_0^{(0)} y_0^{(1)} y_0^{(2)}, y_1^{(0)} y_1^{(1)} y_1^{(2)}, \ldots, y_{K-1}^{(0)} y_{K-1}^{(1)} y_{K-1}^{(2)} \right).$$

⊙ Assume 0 is mapped to -1 and 1 is mapped to +1

⊙ Then for an AWGN channel, we define the log-likelihood ratio (L-value) $L\left(u_k \middle| y_k^{(0)}\right)$. (before decoding) of a transmitted information bit u_k given the received value $y_k^{(0)}$ as

$$L\left(u_k \middle| y_k^{(0)}\right) = \ln \frac{P\left(u_k = +1 \middle| y_k^{(0)}\right)}{P\left(u_k = -1 \middle| y_k^{(0)}\right)}$$

$$= \ln \frac{P\left(y_k^{(0)} \middle| u_k = +1\right) P(u_k = +1)}{P\left(y_k^{(0)} \middle| u_k = -1\right) P(u_k = -1)}$$

$$= \ln \frac{P\left(y_k^{(0)} \middle| u_k = +1\right)}{P\left(y_k^{(0)} \middle| u_k = -1\right)} + \ln \frac{P(u_k = +1)}{P(u_k = -1)}$$

$$= \ln \frac{e^{-(E_s/N_0)\left(y_k^{(0)} - 1\right)^2}}{e^{-(E_s/N_0)\left(y_k^{(0)} + 1\right)^2}} + \ln \frac{P(u_k = +1)}{P(u_k = -1)}$$

$$= -\frac{E_s}{N_0}\left\{(r_i^{(0)}-1)^2 - (r_i^{(0)}+1)^2\right\} + \ln \frac{P(u_i=+1)}{P(u_i=-1)}$$

$$= 4\frac{E_s}{N_0} r_i^{(0)} + \ln \frac{P(u_i=+1)}{P(u_i=-1)}$$

$$= L_c r_i^{(0)} + L_a(u_i)$$

where,

$\frac{E_s}{N_0}$ → channel SNR

u_i and $r_i^{(0)}$ → Both have been normalized by a factor of $\sqrt{E_s}$.

⊙ In the case of a transmitted parity bit $v_i^{(j)}$, given the received value $(r_i^{(j)})$, $j = 1, 2$, the L-value (before decoding) is given by

$$L\left(v_i^{(j)} \mid r_i^{(j)}\right) = L_c r_i^{(j)} + L_a\left(v_i^{(j)}\right)$$

$$= L_c r_i^{(j)}, \quad j = 1, 2 \quad \text{(why ?)}$$

⊙ $L_a(u_i)$ also equals 0 for the first iteration of decoder 1, but that thereafter the apriori L-values of the information bits are replaced by extrinsic L-values from the other decoder.

⊙ The received soft channel L-values $L_c r_i^{(0)}$ for u_i and $L_c r_i^{(1)}$ for $v_i^{(1)}$ enter decoder 1, while the (properly interleaved) received soft channel L-values $L_c r_i^{(0)}$ for u_i and the received soft channel L-values $L_c r_i^{(2)}$ for $v_i^{(2)}$ enter decoder 2.

⊙ The output of decoder 1 contains two terms :

$$* \; L^{(1)}(u_l) = \ln \left[\frac{P(u_l = +1 \mid r_1, La^{(1)})}{P(u_l = -1 \mid r_1, La^{(1)})} \right]$$

$$* \; L_e^{(1)}(u_l) = L^{(1)}(u_l) - \left[L_c r_l^{(0)} + L_e^{(2)}(u_l) \right]$$

where,

$L^{(1)}(u_l)$ → Aposteriori L-value (after decoding) of each information bit produced by decoder 1 given the (partial) received vector

$$r_1 \triangleq \left[r_0^{(0)} r_0^{(1)}, r_1^{(0)} r_1^{(1)}, \ldots, r_{K-1}^{(0)} r_{K-1}^{(1)} \right]$$

and the Apriori input vector

$$L_a^{(1)} \triangleq \left[La^{(1)}(u_0), La^{(1)}(u_1), \ldots, La^{(1)}(u_{K-1}) \right].$$

for decoder 1.

$L_e^{(1)}(u_l)$ → Extrinsic Aposteriori L-value (after decoding) associated with each information bit produced by decoder 1, which after interleaving, is passed to the input of decoder 2 as the apriori value $La^{(2)}(u_l)$.

⊙ Subtracting the terms, viz., $L_c r_l^{(0)} + L_e^{(2)}(u_l)$ from $L^{(1)}(u_l)$, removes the effect of the current information bit u_l from $L^{(1)}(u_l)$, and thus providing an independent estimate of the information bit u_l to decoder 2 in addition to the received soft channel L-values at time l.

⊙ Similarly, the output of decoder 2 contains two terms.

* $L^{(2)}(u_k) = \ln\left[P(u_k = +1 \mid r_2, L_a^{(2)}) \Big/ P(u_k = -1 \mid r_2, L_a^{(2)}) \right]$,

where r_2 is the (partial) received vector and

$L_a^{(2)}$ is the apriori input vector for decoder 2

* And,

$L_e^{(2)}(u_k) = L^{(2)}(u_k) - \left[L_c r_k^{(0)} + L_e^{(1)}(u_k) \right]$, and the extrinsic aposteriori L-values $L_e^{(2)}(u_k)$ produced by decoder 2, after deinterleaving, are passed back to the input of decoder 1 as the apriori values $L_a^{(1)}(u_k)$.

⊙ Thus the input to each decoder contains three terms.

* The soft channel L-values $L_c r_k^{(0)}$ (information bit)
* The soft channel L-values $L_c r_k^{(1)}$ (or) $L_c r_k^{(2)}$ (parity bit)
* The extrinsic aposteriori L-values $L_e^{(2)}(u_k) = L_a^{(1)}(u_k)$ (or) $L_e^{(1)}(u_k) = L_a^{(2)}(u_k)$ passed from the other decoder.

⊙ In the initial iteration of decoder 1, the extrinsic aposteriori L-values $L_e^{(2)}(u_k) = L_a^{(1)}(u_k)$ are just the original apriori L-values $L_a(u_k)$, which are all equal to 0 for equally likely information bits.

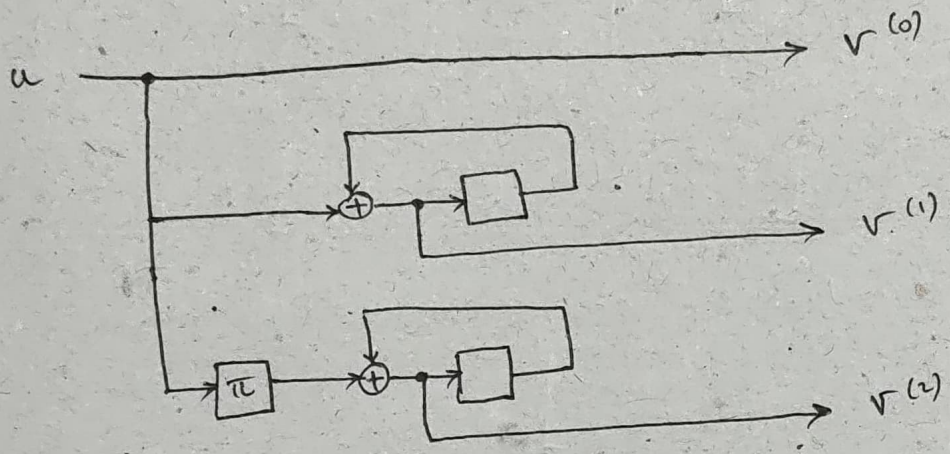
⊙ Thus the extrinsic L-values, passed from one decoder to the other during the iterative decoding process are treated like new sets of apriori probabilities by the MAP algorithm.

- Decoding proceeds iteratively, with each decoder passing its respective extrinsic L-values back to the other decoder.
- Decoding stops after sufficient number of iterations.
- At the output of decoder 2, the decoded information bits are determined from the aposteriori L-values $L^{(2)}(u_k)$.
- Positive L-values are decoded as "+1" and negative L-values are decoded as "-1".

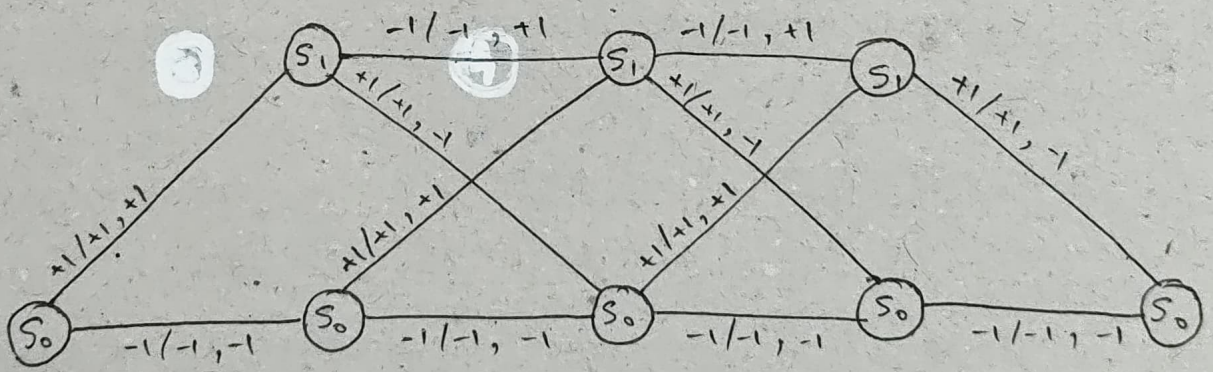
Example:

Rate $R = \frac{1}{3}$ turbo code using constituent encoder $G(D)$.

$$G(D) = \left[1 \quad \frac{1}{1+D}\right]$$



(a) Encoder diagram.



(b) State diagram

⊙ Consider an input sequence of length $K = 4$, including one termination bit, along with a 2×2 block (row-column) interleaver, resulting in a (12, 3) turbo code with overall rate $R = \frac{1}{4}$.

⊙ The Trellis for the constituent code is shown above, where the branches are labelled using the mapping

$0 \rightarrow -1$ and

$1 \rightarrow +1$.

⊙ The input block is given by the vector $u = [u_0, u_1, u_2, u_3]$ the interleaved input block is

$u' = [u_0', u_1', u_2', u_3'] = [u_0, u_2, u_1, u_3]$.

⊙ The parity vector for the first constituent code is given by $p^{(1)} = [p_0^{(1)}, p_1^{(1)}, p_2^{(1)}, p_3^{(1)}]$.

⊙ The parity vector for the second constituent code is $p^{(2)} = [p_0^{(2)}, p_1^{(2)}, p_2^{(2)}, p_3^{(2)}]$.

⊙ The 12 transmitted bits are represented in a rectangular array, as shown in figure below, where

- the input vector u determines the parity vector $p^{(1)}$ in the first two rows and
- the interleaved input vector u' determines the parity vecto $p^{(2)}$ in the first two columns.

⊙ For purposes of illustration, we assume the particular bit values shown in figure.

⊙ We also assume a channel SNR of $\frac{E_s}{N_0} = \frac{1}{4}$ $(-6.02\ dB)$, so that the received channel L-values, corresponding to the received vector

$$r = \left[r_0^{(0)}\ r_0^{(1)}\ r_0^{(2)},\ r_1^{(0)}\ r_1^{(1)}\ r_1^{(2)},\ r_2^{(0)}\ r_2^{(1)}\ r_2^{(2)},\ r_3^{(0)}\ r_3^{(1)}\ r_3^{(2)} \right]$$

are given by

$$L_c r_\ell^{(j)} = 4\left(\frac{E_s}{N_0}\right) r_\ell^{(j)} = r_\ell^{(j)}, \qquad \ell = 0, 1, 2, 3 \ldots$$
$$j = 0, 1, 2.$$

u_0	u_1	$p_0^{(1)}$	$p_1^{(1)}$
u_2	u_3	$p_2^{(1)}$	$p_3^{(1)}$
$p_0^{(2)}$	$p_2^{(2)}$		
$p_1^{(2)}$	$p_3^{(2)}$		

(12, 3) PCCC

−1	+1	−1	+1
−1	+1	+1	−1
−1	+1		
−1	−1		

Coded values.

+0.8	+1.0	+0.1	−0.5
−1.8	+1.6	+1.1	−1.6
−1.2	+0.2		
+1.2	−1.1		

Received L-values

- In the first iteration of decoder 1 (row decoding), the BCJR algorithm is applied to the trellis of the 2-state (2,1,1) code to compute the aposteriori L-values, $L^{(1)}(u_l)$ for each of the four input bits and the corresponding extrinsic aposteriori L-values $L_e^{(1)}(u_l)$.

- For iterative decoding, extrinsic aposteriori L-values are computed for all input bits, termination bits as well as information bits.

- Before the first iteration of decoding, the apriori L-values of the termination bits are assumed to be zero.

- Similarly, in the first iteration of decoder 2, the BCJR algorithm uses the extrinsic aposteriori L-values $L_e^{(1)}(u_l)$ received from decoder 1 as the apriori L-values, $L_a^{(2)}(u_l)$ to compute the aposteriori L-values $L^{(2)}(u_l)$ for each of the four input bits and the corresponding extrinsic aposteriori L-values $L_e^{(2)}(u_l)$ to pass back to decoder 1.

- Decoding proceeds iteratively in this fashion.

-0.32	-0.38
+0.77	+0.47

Extrinsic L-values after first row decoding

-0.88	-0.69
+0.23	-0.04

Extrinsic L-values after first column decoding

-0.40	-0.07
-0.80	+2.03

Soft output L-values after the first row and column decoding.

-0.01	-0.01
+0.43	+0.77

Extrinsic L-values after second row decoding

-0.98	-0.81
+0.07	-0.21

Extrinsic L-values after second column decoding

-0.19	+0.18
-1.30	+2.16

Soft output L-values after the second row and column decoding.