Appendix B
Turbo Decoder

Corresponding to the Turbo code with two component RSC codes, there is usually an iterative decoder with two component decoders operating cooperatively.

In present, the decoding algorithms of component decoder mainly include[70]:

1. Maximum a posteriori (MAP) algorithm,
2. Log-MAP algorithm,
3. Soft output Viterbi algorithm (SOVA).

Log-MAP algorithm replaces most of multiplications in an MAP algorithm with additions so as to have better properties for a hardware implementation. SOVA is a soft output implementation of Viterbi algorithm (VA), which has lower implementation complexity too. But their performance is worse than the MAP algorithm. Therefore, only the MAP algorithm is sketched here and applied to the iterative decoding in the simulations of Chapter 5 and 7 in fact.

B.1 MAP Decoding Algorithm

Viterbi algorithm is a maximum likelihood method which minimizes the probability of sequence error for CCs, but does not necessarily minimize BER for the underlying information sequence. In 1974, Bahl, et al. proposed an optimal decoding method, denoted BCJR algorithm [71] for linear codes, which minimizes BER. The BCJR algorithm is similar to the VA except that it performs both forward and backward recursions and produces a likelihood ratio for the information bit instead of considering the most-likely state transitions due to an input 1 versus those due to an input 0.

Let $y_k = (y_k^s, y_k^p)$ be a noise version of $(x_k^s, x_k^p) = (x_k^s, x_k^{1p}, x_k^{2p})$ as in Chapter 5, where $k = 1, 2, \cdots, N$. Then, $\mathbf{y} = \mathbf{y}_1^N = (y_1, y_2, \cdots, y_N)$ is the noisy received codeword, and $y_a^b = (y_a, y_{a+1}, \cdots, y_b)$ $(1 \leq a \leq b \leq N)$ is one part of $\mathbf{y}$. Assume that the trellis termination is executed and N' denotes $N - t$, let the transmission of the symbols ± 1 over the channel, corresponding to the information bit u_k where $1 \leq k \leq N'$.

The decoding problem is to estimate the information sequence $\mathbf{u}$ from the received sequence $\mathbf{y}$. The MAP decoder decides $u_k = +1$ if $P(u_k = +1\,|\mathbf{y}) \geq P(u_k = -1\,|\mathbf{y})$, or $u_k = -1$ otherwise. The decision $\hat{u}_k$ is given by $\hat{u}_k = \text{sign}[L(u_k)]$, where $L(u_k)$ is log a posteriori probability (LAPP) ratio defined as:

$$L(u_k) = \log(\frac{P(u_k = +1\,|\mathbf{y})}{P(u_k = -1\,|\mathbf{y})}). \tag{B.1}$$

Incorporating the code's trellis, this may be written as:

$$L(u_k) = \log(\frac{\sum_{S^+} P(s_{k-1} = s', s_k = s, \mathbf{y})/P(\mathbf{y})}{\sum_{S^-} P(s_{k-1} = s', s_k = s, \mathbf{y})/P(\mathbf{y})}) = \log(\frac{\sum_{S^+} P(s', s, \mathbf{y})}{\sum_{S^-} P(s', s, \mathbf{y})}), \tag{B.2}$$

where $s_k \in S$ is the state of the encoder at time s, and S is the set of all 2^{K-1} component encoder states. K is the constraint length of each component code. S^+ is the set of ordered pairs (s', s) corresponding to all state transitions $(s_{k-1} = s') \rightarrow (s_k = s)$ caused by data input $u_k = +1$, and S^- is similarly defined for $u_k = -1$. In terms of BCJR algorithm [71], $P(s', s, \mathbf{y})$ can be expressed by:

$$P(s', s, \mathbf{y}) = \alpha_{k-1}(s')\gamma_k(s', s)\beta_k(s) \tag{B.3}$$

where $\gamma_k(s', s)$ is defined as the branch transition probability $P(s_k = s, y_k\,|s_{k-1} = s')$, which can be determined from the transition probabilities of channel and encoder trellis.

As a forward recursive probability function, $\alpha_k(s) \equiv P(s_k = s, \mathbf{y}_1^k)$ is computed recursively as:

$$\alpha_k(s) = \sum_{s' \in S} \alpha_{k-1}(s')\gamma_k(s', s), \tag{B.4}$$

with initial conditions $\alpha_0(0) = 1$ and $\alpha_0(s \neq 0) = 0$, i.e., the encoder is expected to start in state 0.

As a backward recursive probability function, $\beta_k(s) \equiv P(\mathbf{y}_{k+1}^N\,|s_k = s)$ is computed in a backward recursion as:

$$\beta_{k-1}(s') = \sum_{s \in S} \beta_k(s)\gamma_k(s', s), \tag{B.5}$$

with boundary conditions $\beta_N(0) = 1$ and $\beta_N(s \neq 0) = 0$, i.e., the encoder is expected to end in state 0 after N input bits, implying that the last t input bits, called termination bits, are so selected.

Then, $L(u_k)$ can be expressed by:

$$L(u_k) = \log(\frac{\sum_{S^+} \alpha_{k-1}(s')\gamma_k(s', s)\beta_k(s)}{\sum_{S^-} \alpha_{k-1}(s')\gamma_k(s', s)\beta_k(s)}). \tag{B.6}$$

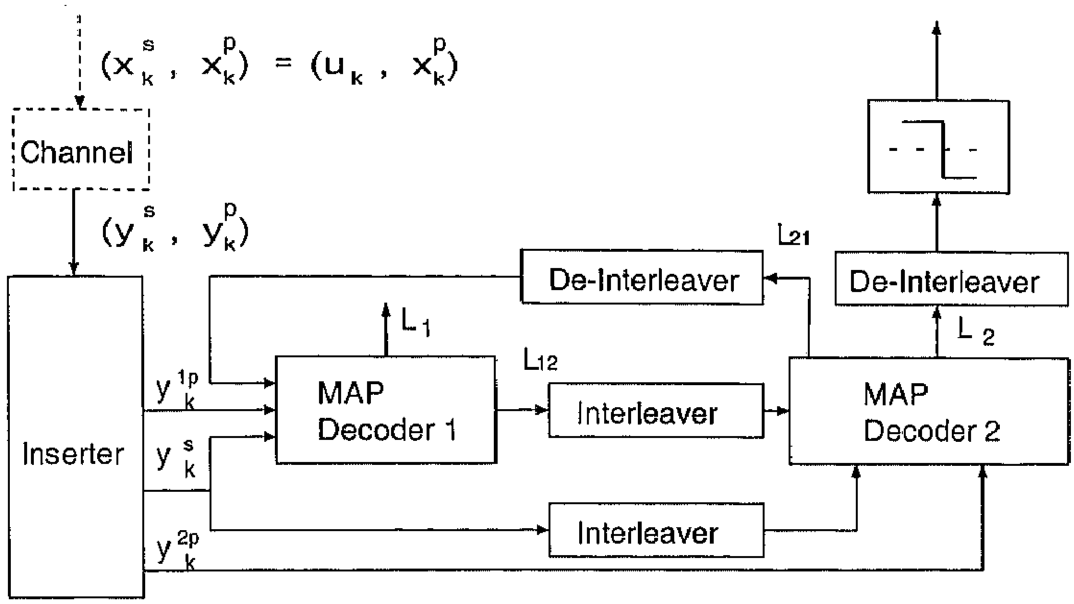

Figure B.1: **Two dimensional frame-based Turbo decoder**

B.2 Iterative Decoding Algorithm

From Bayes' rule, the LAPP ratio for an arbitrary MAP decoder can be written as:

$$L(u_k) = \log(\frac{P(\mathrm{y}\,|u_k = +1)}{P(\mathrm{y}\,|u_k = -1)}) + \log(\frac{P(u_k = +1)}{P(u_k = -1)}), \tag{B.7}$$

where the second term represents a priori information. Since $P(u_k = +1) = P(u_k = -1)$ typically, the a priori term is usually zero for conventional decoders. However, for iterative decoders, Decoder 1 (D1) receives extrinsic or soft information for each u_k from Decoder 2 (D2), which serves as a priori information. Similarly, D2 receives extrinsic information from D1 and the decoding iteration proceeds as D1 $\rightarrow$ D2 $\rightarrow$ D1 $\rightarrow$ D2 $\rightarrow \cdots$, with the previous decoder passing extrinsic information along to the next decoder at each half-iteration except for the first. The idea behind extrinsic information is that D2 provides extrinsic information to D1 for each u_k, by using only information not available to D1, i.e., y^{2p}. D1 does likewise for D2.

Figure B.1 shows how this algorithm is incorporated into an iterative decoder employing two MAP decoders in a serial concatenation scheme. When the redundant information of a given encoder is not emitted, the corresponding decoder input is set to zero by the inserter. L_{12} is extrinsic information from D1 to D2, L_{21} is extrinsic information from D2 to D1 so that the results of component decoders are exchanged through the feed forward and feedback loop. After a certain number of iterations, the soft outputs of both MAP decoders stop producing further performance improvements. Then the last stage of decoding makes a hard decision after the de-interleaver. The following shows how interleavers and de-interleavers are involved in arranging systematic, parity, and extrinsic information in the proper sequence for each decoder.

From its definition, $\gamma_k(s', s)$ can be written as:

$$\gamma_k(s', s) = P(s\,|s')P(y_k\,|s', s) = P(u_k)P(y_k\,|u_k), \tag{B.8}$$

where the event u_k corresponds to an event $s' \to s$. Defining:

$$L^e(u_k) \equiv \log(\frac{P(u_k = +1)}{P(u_k = -1)}), \tag{B.9}$$

then,

$$P(u_k) = (\frac{\exp[-L^e(u_k)/2]}{1 + \exp[-L^e(u_k)]}) \exp[u_k L^e(u_k)/2] \equiv A_k \exp[u_k L^e(u_k)/2]. \tag{B.10}$$

As for $P(y_k|u_k)$, $y_k = (y_k^s, y_k^p)$ and $x_k = (x_k^s, x_k^p) = (u_k, x_k^p)$. For the Turbo code with two rate $\frac{1}{2}$ component codes over AWGN channel, we have:

$$\begin{aligned} P(y_k|u_k) &\propto \exp[-\frac{(y_k^s - u_k)^2}{2\sigma^2} - \frac{(y_k^s - x_k^p)^2}{2\sigma^2}] \\ &= \exp[-\frac{y_k^{s2} + u_k^2 + y_k^{p2} + x_k^{p2}}{2\sigma^2}] \exp[\frac{u_k y_k^s + x_k^p y_k^p}{\sigma^2}] \\ &\equiv B_k \exp[\frac{u_k y_k^s + x_k^p y_k^p}{\sigma^2}], \end{aligned} \tag{B.11}$$

where σ^2 is the variance of the Gaussian noise, so that:

$$\gamma_k(s', s) \propto A_k B_k \exp[u_k L^e(u_k)/2] \exp[\frac{u_k y_k^s + x_k^p y_k^p}{\sigma^2}]. \tag{B.12}$$

From (B.6), the factor $A_k B_k$ will be canceled as it is independent of u_k. Due to $\frac{E_s}{N_0/2} = \frac{1}{\sigma^2}$, $\sigma^2 = N_0/2E_s$, where $E_s = RE_b$ is energy per channel bit. Then, we have:

$$\begin{aligned} \gamma_k(s', s) &\propto \exp[\frac{1}{2}u_k(L^e(u_k) + L_s y_k^s) + \frac{1}{2}L_s y_k^p x_k^p] \\ &= \exp[\frac{1}{2}u_k(L^e(u_k) + L_s y_k^s)]\gamma_k^e(s', s), \end{aligned} \tag{B.13}$$

where $L_s \equiv 4E_s/N_0$, which is called the reliability value of the channel, and $\gamma_k^e(s', s) \equiv \exp[\frac{1}{2}L_s y_k^p x_k^p]$.

Combining (B.6) and (B.13), we obtain:

$$L(u_k) = L_s y_k^s + L^e(u_k) + \log(\frac{\sum_{S^+} \alpha_{k-1}(s')\gamma_k^e(s', s)\beta_k(s)}{\sum_{S^-} \alpha_{k-1}(s')\gamma_k^e(s', s)\beta_k(s)}) \tag{B.14}$$

The first term of (B.14) is called a channel value, the second term represents any a priori information about u_k provided by the previous decoder, and the third term represents extrinsic information that can be passed on to the subsequent decoder. The extrinsic information is a function of the redundant information introduced by the encoder. It does not contain the information decoder input y_k^s. This quantity may be used to improve the a priori probability estimate for the next decoding stage.

For D1, $L^e(u_k)$ is $L_{21}^e(u_{DI[k]})$, i.e., the deinterleaved extrinsic information from the previous D2 iteration. Thus, on any given iteration, D1 computes:

$$L_1(u_k) = L_s y_k^s + L_{21}^e(u_{DI[k]}) + L_{12}^e(u_k), \tag{B.15}$$

where $L_{12}^e(u_k)$ is corresponding to the third term in (B.14) that is to be used as the extrinsic information passed from D1 to D2, expressed by:

$$L_{12}^e(u_k) = \log\left(\frac{\sum_{S^+} \alpha_{k-1}^{(1)}(s')\gamma_k^e(s',s)\beta_k^{(1)}(s)}{\sum_{S^-} \alpha_{k-1}^{(1)}(s')\gamma_k^e(s',s)\beta_k^{(1)}(s)}\right). \tag{B.16}$$

For D2, $L^e(u_k)$ is $L_{12}^e(u_{I[k]})$, i.e., the interleaved extrinsic information from the previous D1 iteration. Then, on any given iteration, D2 computes:

$$L_2(u_k) = L_s y_{I[k]}^s + L_{12}^e(u_{I[k]}) + L_{21}^e(u_k), \tag{B.17}$$

where $L_{21}^e(u_k)$ is corresponding to the third term in (B.14) that is to be used as the extrinsic information passed from D2 to D1, expressed by:

$$L_{21}^e(u_k) = \log\left(\frac{\sum_{S^+} \alpha_{k-1}^{(2)}(s')\gamma_k^e(s',s)\beta_k^{(2)}(s)}{\sum_{S^-} \alpha_{k-1}^{(2)}(s')\gamma_k^e(s',s)\beta_k^{(2)}(s)}\right). \tag{B.18}$$

Note that $\alpha_{k-1}^{(1)}(s')$, $\alpha_{k-1}^{(2)}(s')$ and $\beta_k^{(1)}(s)$, $\beta_k^{(2)}(s)$ can be computed in terms of (B.4) and (B.5), respectively.

From the above equations, the iterative decoding algorithm is illustrated as Figure B.2. Here, some parameters should be initialized firstly.

For D1:

- $\alpha_0^{(1)}(0) = 1$ and $\alpha_0^{(1)}(s \neq 0) = 0$,
- $\beta_N^{(1)}(0) = 1$ and $\beta_N^{(1)}(s \neq 0) = 0$,
- $L_{21}^e(u_k) = 0$ for $k = 1, 2, \cdots, N$.

For D2:

- $\alpha_0^{(2)}(0) = 1$ and $\alpha_0^{(2)}(s \neq 0) = 0$,
- $\beta_N^{(2)}(s) = \alpha_N^{(2)}(s)$ for all s (after computation of $\alpha_N^{(2)}(s)$ in the first iteration),
- $L_{12}^e(u_k)$ is to be determined from D1 after the first half-iteration and so need not be initialized.

Then, according to the maximum iterative number T, the iterative computation proceeds. At last, the final decisions may come from either L_1 or L_2, i.e., the outputs of D1 or D2. Figure B.2 chooses L_2 as the decision parameter. If $L_2(u_k) > 0$, then $u_k = +1$, else $u_k = -1$.

The previous description is over AWGN channel, where a branch metric based on the standard Euclidean distance between the received and transmitted signal sequences is used. This metric is optimum for AWGN channel. On fading channel, the decoding metric should optimize the conditional probability $P(y_k|u_k)$. Since this condition gives a fairly complicated metric computation, for practical reasons, the AWGN metric is used as the decoding metric when no CSI is available at the receiver. So the MAP iterative decoding algorithm derived for AWGN channel will be applied to decode Turbo codes for fading channel without CSI.

If ideal CSI is available at the decoder, the branch metric should be modified by replacing the transmitted signal x_k by $a_k x_k$, where a_k is a random variable which represents a channel fading variation. The MAP decoding algorithm has to be modified slightly by changing $L_s \equiv 4E_s/N_0$ to $L_s \equiv 4a_k E_s/N_0$. With this modification, the same decoder structure can be used as before.

This metric is not the optimum one, but is a reasonable tradeoff between the complexity and performance [72], as the optimum metric is very complex. Based on this branch metric, the log-likelihood ratio and extrinsic information for MAP decoding method are modified.

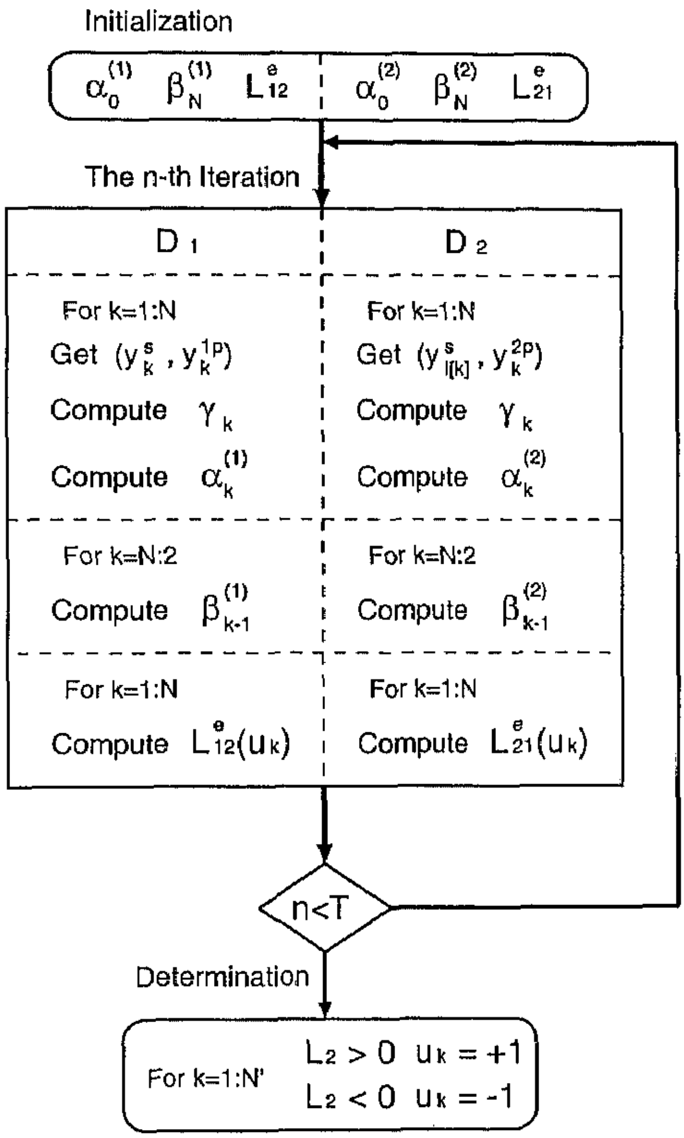

Figure B.2: The flow chart of iterative decoding algorithm