Chapter 4 Binary Capacity-Approaching Codes

This chapter is an introduction to modern coding theory. The encoding and decoding principles of several classes of capacity-approaching codes (including Turbo codes, LDPC codes, and RA codes) will be discussed in detail. They can be viewed as codes defined on graphs. The general graphical models will be addressed with emphasis on Forney-style graph.

4.1 Introduction

It has been universally recognized that turbo-like codes are the most successful known class of codes, which can achieve near-capacity performance over AWGN and many other channels. The common feature of all codes in this family is the feasibility of an iterative decoder, which performs close to the ML decoder provided that SNR is sufficiently high. Typical examples of turbo-like codes are listed below.

- **Turbo Codes** (by Claude Berrou and A. Glavieux)
  - *Block turbo codes* (*or, Turbo product codes*)
  - *Asymmetric turbo codes* (by Costello and Massey)
  - *Serially concatenated codes* with iterative decoding
  - *Mixed turbo codes* (by Divsalar, *et al*)
  - *Doped turbo codes* (by ten Brink)
  - *Concatenated Tree (CT) Codes* (by Li Ping)
- **Low-Density Parity-Check (LDPC) Codes**
  - *Regular codes* (by Gallager)
  - *Irregular LDPC codes* (by Luby, Richardson, *et al*)
- **Repeat-Accumulate (RA) Codes** (by McEliece, Divsalar, Jin)

4.1.1 The Turbo-Era State of the Art on the AWGN Channel

The performance of turbo-like codes on the power-limited AWGN channel is shown in Fig. 4.1, where some classical FEC codes are also included for comparison.

- Turbo codes, also called parallel concatenated convolutional codes (PCCC), were proposed by Berrou and Glavieux in the ICC'93. Performance within 0.5 dB of the channel capacity limit for BPSK was demonstrated.
- Since 1993, turbo codes have been widely studied and adopted in several communication systems, and the inherent concepts of the "**turbo principle**" have been applied to topics other than error correction coding, such as single-user (equalization) and multi-user detection.
- The significance of turbo codes becomes clear upon observing their performance. Figure 4.2 shows simulation results of the original rate $R = 1/2$ turbo code presented in [1].

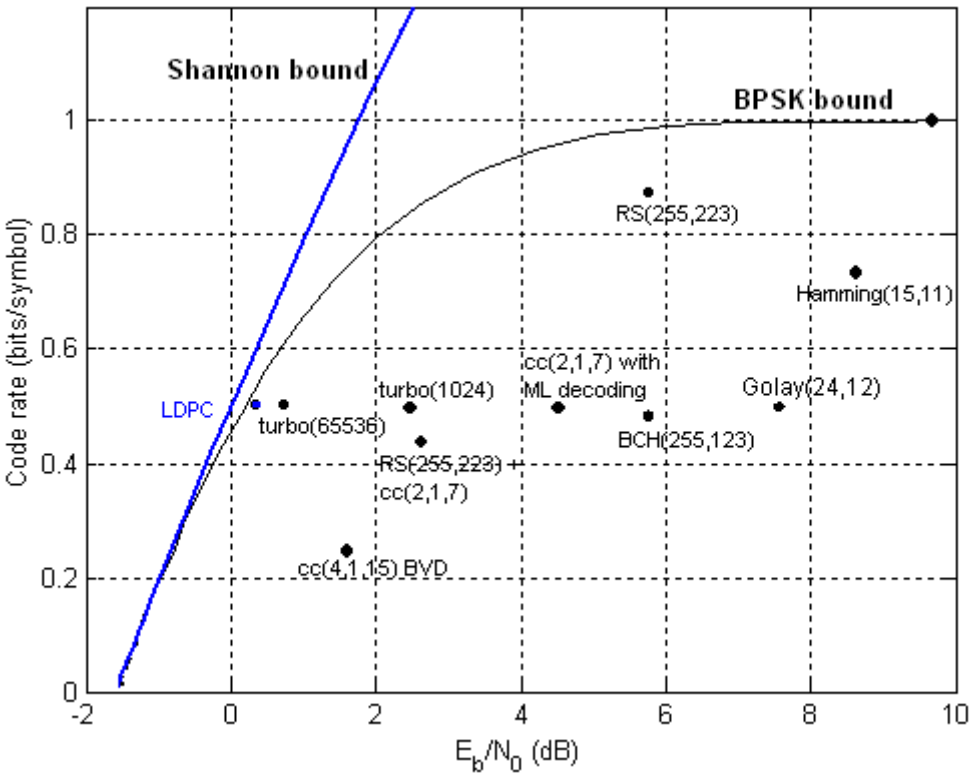


Fig. 4.1 Milestones in the drive towards channel capacity achieved over the past 50 years.

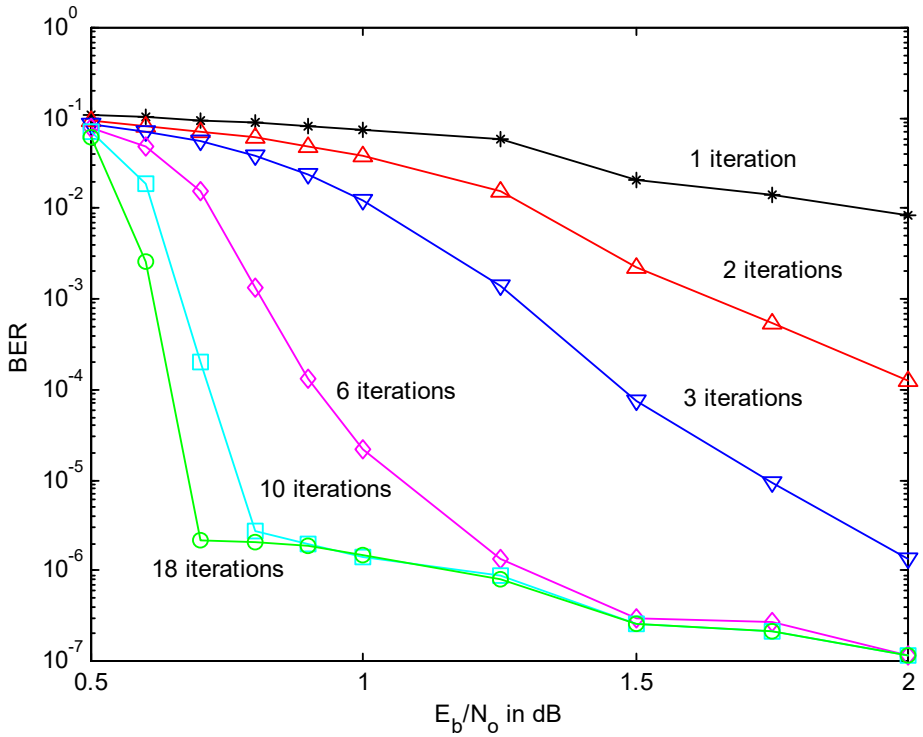
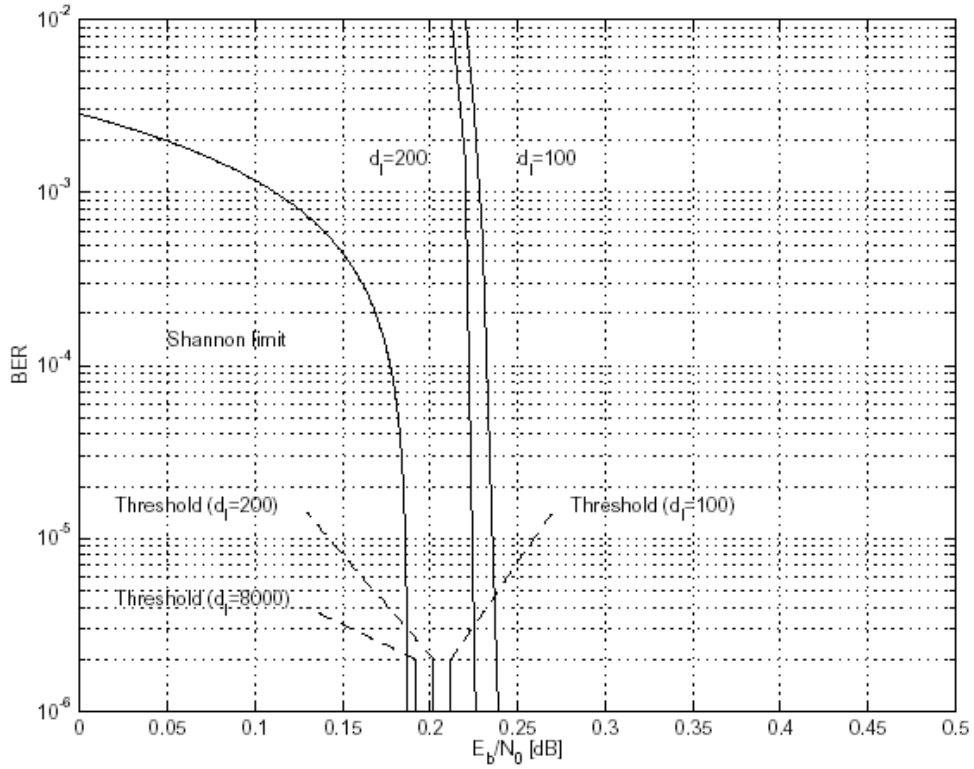


Figure 4.2 Performance of a rate-1/2 binary turbo code with 16-state RSC codes as constituent codes. Block interleaver size = 65536.

- It has been reported [Chung01] that a well-designed irregular LDPC code of length 10^7 has the performance within 0.04dB of the Shannon limit at BER=10^{-6}.



Irregular LDPC Codes, Density Evolution
(Chung, Forney, Richardson, Urbanke, 2001)

For non-binary codes, the coded modulation schemes based on Turbo-like codes can also achieve near capacity performance on both AWGN and fading channels. Fig. 4.3 compares the Turbo-TCM scheme to the conventional TCM scheme. It is seen that the improvement made by coding becomes evident. The bit-interleaved coded modulation (BICM) and multilevel coding (MLC) schemes using LDPC codes exhibit similar or even better performance.

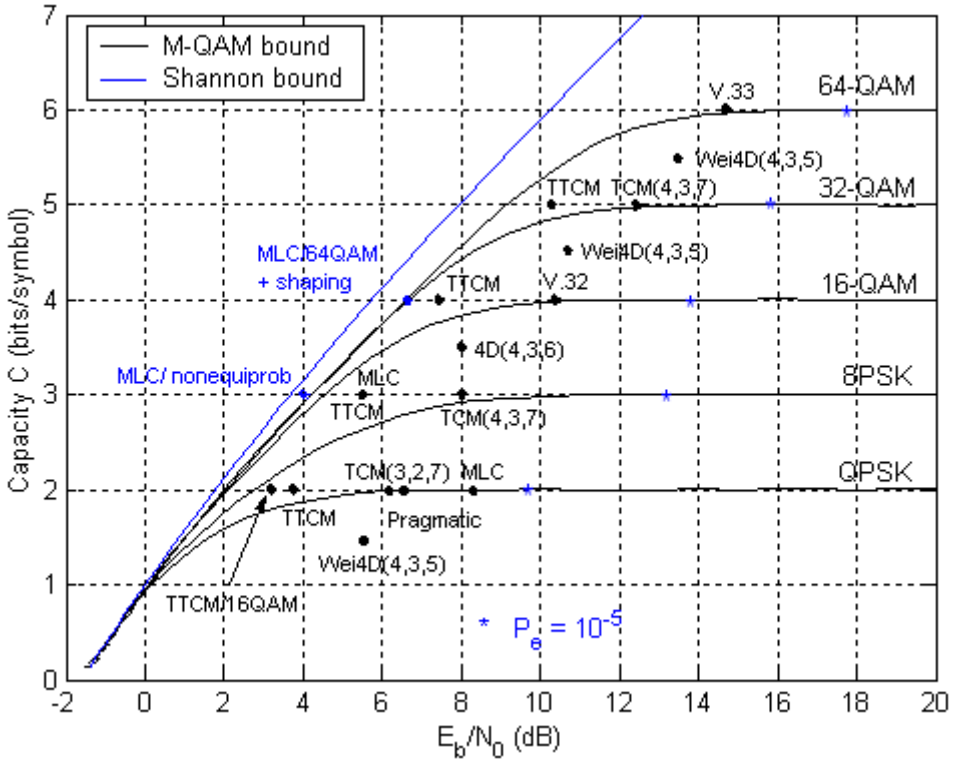


Figure 4.3 Performance of various coded modulation schemes based on turbo-like codes.

4.2 Binary Turbo Codes

4.2.1 The Invent of Turbo Codes

Shannon 理论证明，随机码是好码，但是它的译码却太复杂。因此，多少年来随机编码理论一直是作为分析与证明编码定理的主要方法，而如何在构造码上发挥作用却并未引起人们的足够重视。直到 1993 年，Turbo 码的发现，才较好地解决了这一问题，为 Shannon 随机编码理论的应用研究奠定了基础。

Turbo 码，又称并行级连卷积码(PCCC)，是由 C.Berrou 等在 ICC'93 会议上提出的[1]。它巧妙地将卷积码和随机交织器结合在一起，实现了随机编码的思想，同时，采用软输出迭代译码来逼近最大似然译码。

关于 turbo 码的发展历程，C. Berrou 等在[?]中给出了详细的说明。因为 C. Berrou 主要从事的是通信集成电路的研究，所以他们将 SOVA 译码器看作是“信噪比放大器”，从而将电子放大器中的反馈技术应用于串行级联的软输出译码器，并且为了使两个译码器工作于相同的时钟，以简化时钟电路设计，就提出了并行级联方式，这导致了 turbo 码的发明。

- Serially Concatenated Coding

Fig. 4.4 depicts a serially concatenated coding scheme, where the outer encoder / decoder and inner encoder/decoder operate on ***different clocks***. To solve this problem, Berrou proposed ***parallel concatenated coding*** schemes.

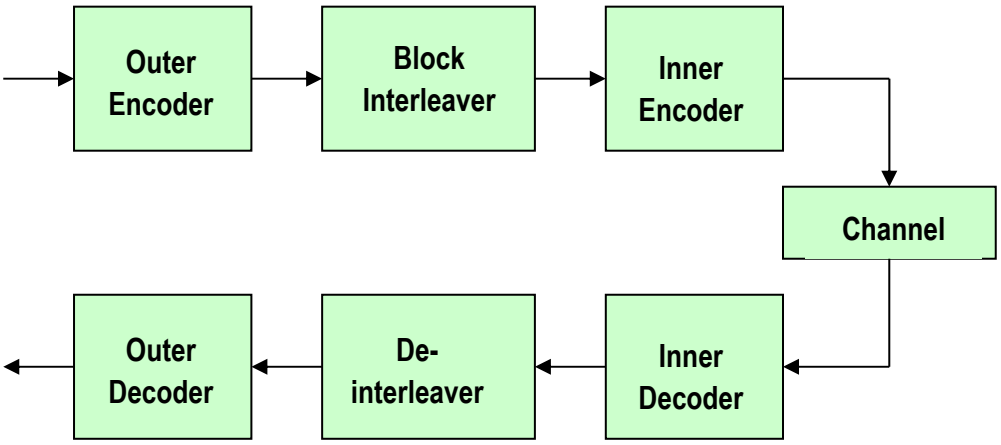


Figure 4.4 Serially concatenated coding scheme

■ Features of turbo codes

- Parallel concatenated coding
- Recursive convolutional encoders
- Pseudo-random interleaving
- Iterative decoding

For details, see [1][2], and the references therein.

4.2.2 Structure of Turbo Encoder

The original turbo code is a parallel concatenation of two ***Recursive Systematic Convolutional*** (RSC) codes. Fig. 4.5 depicts a typical turbo encoder, where two constituent RSC encoders are concatenated in parallel by a pseudo-random interleaver. The interleaver is used to permute the input bits such that the two encoders are operating on the same block of input bits, but in a different order.

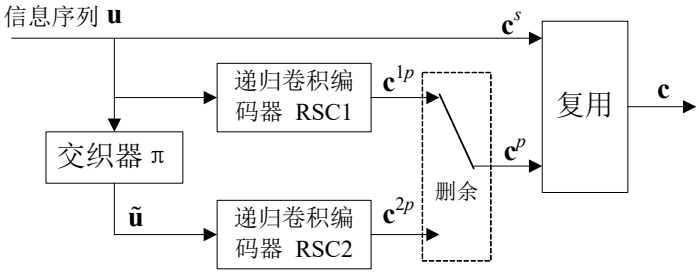


Figure 4.5 Turbo 码编码器的基本结构

对于一个长为 N 的二进制序列，我们用整数集合 $\mathcal{I}=\{1,2,...,N\}$ 来表示其比特位置索引。则长为 N 的交织器可以被定义为索引集合 $\mathcal{I}=\{1,2,...,N\}$ 上的置换：

$$\pi:\mathcal{I}\to\mathcal{I}.$$

通常，我们用 $\pi(k)$ 与 k 分别表示一个元素（比特）在原序列与交织后序列中的位置索引。例如，记交织前的原序列为 $x[k]$，交织后的序列为 $y[k]$，则有

$$y[k]=x[\pi(k)],\ k\in\mathcal{I}$$

交织器通常采用下列方式来表示：

$$\pi=\begin{pmatrix}1 & 2 & \cdots & N\\ \pi(1) & \pi(2) & \cdots & \pi(N)\end{pmatrix}$$

为简单起见，我们有时也用置换向量$\boldsymbol{\pi}=(\pi(0),\pi(1),\cdots,\pi(N-1))$来表示。

Let $\mathbf{u}=(u_1,u_2,...,u_N)$ be an information bit sequence of length *N*, where $u_k\in\{0,1\},1\le k\le N$. 它一方面直接进入第一个分量编码器 RSC1 进行卷积编码，另一方面经过交织后变为长度相同但比特位置重新排列的序列。Let $\tilde{\mathbf{u}}=(\tilde{u}_1,\tilde{u}_2,...,\tilde{u}_N)$ be the permuted sequence by the interleaver corresponding to the original information sequence $\mathbf{u}=(u_1,u_2,...,u_N)$, according to the mapping $\tilde{u}_k=u_{\pi(k)},1\le k\le N$. 交织后的序列$\tilde{\mathbf{u}}$进入第二个分量编码器 RSC2 进行编码，这样我们就得到了两个不同的校验序列$\mathbf{c}^{1p}$和$\mathbf{c}^{2p}$。假定图 4.5 中两个分量编码器的码率均是 1/2, Since both encoders are systematic and operate on the same set of input bits, it is only necessary to transmit the input bits once, and the overall code has rate R_c = 1/3. The overall codeword consists of the systematic bit sequence $\mathbf{c}^s$ = **u** and two parity bit sequences $\mathbf{c}^{1p}$ and $\mathbf{c}^{2p}$; that is, the codeword is given by $\mathbf{c}=(\mathbf{c}_1,\mathbf{c}_2,...,\mathbf{c}_N)$, where $\mathbf{c}_k=(c_k^s,c_k^{1p},c_k^{2p}),1\le k\le N$, is the output code block at time *k*.

为了提高 turbo 码的码率，除可以选用高码率(e.g., rate-2/3)的分量码外，还可以采用删余（puncturing）技术从这两个校验序列中周期性地删除一些校验位，然后再与信息序列（systematic sequence）$\mathbf{c}^s=\mathbf{u}$复用在一起送给数据调制器。例如，在图 4.5 中，In order to increase the rate of the turbo code to R_c = 1/2, the two parity sequences can be punctured according to the following puncturing pattern（删余矩阵）：

$$\mathbf{P}=\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}$$

where the 0's in the *m*th row of **P** indicates that the corresponding bit of the *m*th parity sequence will not be transmitted. 即上述**P**矩阵表示删去来自 RSC1 的校验序列$\mathbf{c}^{1p}$的偶数位置比特与来自 RSC2 的校验序列$\mathbf{c}^{2p}$的奇数位置比特。这样，采用删余技术后，turbo 编码器在 *k* 时刻的输出为$\mathbf{c}_k=(c_k^s,c_k^p)$，其中$c_k^p$由$c_k^{1p}$和$c_k^{2p}$交替组成:

$$c_k^p = \begin{cases} c_k^{1p}, & \text{if } k \text{ is even number} \\ c_k^{2p}, & \text{if } k \text{ is odd number} \end{cases}。$$

■ Recursive Systematic Encoder

从图 4.5 可以看出，在 turbo 编码方案中两个分量码采用并行级联方式，并且编码器结构为递归形式而不是传统的非递归编码器。一个递归编码器具有下面两个特点：

➢An RSC encoder has an infinite impulse response.

➢An arbitrary input will cause a “good” (high weight) output with high probability.

从差错控制编码理论可知，非系统卷积码（NSC）的 BER 性能在高信噪比时比约束长度相同的非递归系统码要好，而在低信噪比时情况却正好相反。递归系统卷积（RSC）码综合了 NSC 码和系统码的特性，虽然它与 NSC 码具有相同的 trellis 结构和自由距离，但是在高码率（$R_c \geq 2/3$）的情况下，对任何信噪比，它的性能均比等效的 NSC 码要好。因此，在 turbo 码中采用 RSC 码作为分量码。一个生成多项式为（37，21）（八进制表示）的 16 状态 RSC 编码器结构如图 4.6 所示，其生成矩阵表示为 $G(D) = \left[1, \dfrac{1+D^4}{1+D+D^2+D^3+D^4}\right]$。

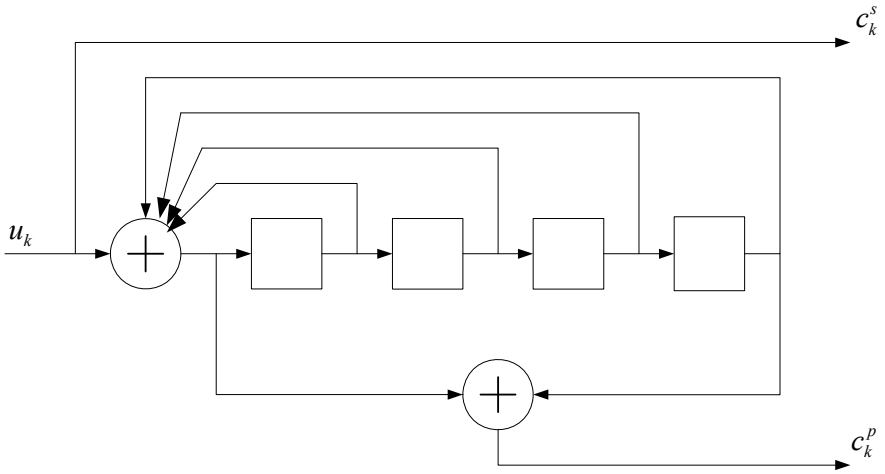


图 4.6　G = (37，21)的 RSC 编码器

■ Pseudo-Random Interleaver

交织器虽然仅仅是在 RSC2 编码之前将信息序列中的 N 个比特的位置进行随机置换，但它却起着关键的作用，很大程度地影响着 turbo 码的性能。

Shannon showed that large block-length random codes achieve channel capacity. However，its high decoding complexity prohibits its applications in practice. 因此，多少年来随机编码理论一直是作为分析与证明编码定理的主要方法，而如何在构造码上发挥作

用却并未引起人们的足够重视。It has been recognized that codes with structure don't perform as well as random codes. The turbo coding method provides an effective solution to this problem:

Make the code appear random, while maintaining enough structure to permit decoding.

In turbo codes, this is achieved by the use of random interleaver. Interleaving has a role in shaping the weight distribution of the code, leading to that ***Turbo codes possess random-like properties.***

另一方面，通过随机交织，使得编码序列在长为 2N 或 3N（不使用删余）比特的范围内具有记忆性，从而由简单的短码得到了长码。当交织器充分大时，turbo 码就具有近似于随机长码的特性。

■ 一个例子

下面通过一个简单例子来具体说明Turbo码编码器的工作过程。

考虑图4.7（a）所示的Turbo码编码器，其中两个分量编码器均是4状态，码率为1/2的 $(7,5)_8$ RSC编码器，其生成矩阵为

$$G(D)=\left[1,\ \frac{1+D^2}{1+D+D^2}\right]$$

分量编码器的trellis图如图4.7（b）所示。

令**u**表示turbo编码器的输入信息比特序列，$\tilde{\mathbf{u}}$ 表示交织器的输出序列。假定采用一个size为7的伪随机交织器

$$\pi_7 = \quad (4,\ 1,\ 6,\ 3,\ 5,\ 7,\ 2)\ ,$$

则有 $\tilde{u}_1=u_4,\tilde{u}_2=u_1$ 等等。

如果输入序列 $\mathbf{u}=(1011001)$，则 $\mathbf{c}^s=(1011001)$，图4.7（b）中上面的RSC编码器的状态转移为：

$$0\xrightarrow{1/11}2\xrightarrow{0/01}3\xrightarrow{1/10}3\xrightarrow{1/10}3\xrightarrow{0/01}1\xrightarrow{0/00}2\xrightarrow{1/10}1$$

对应的输出校验序列 $\mathbf{c}^{1p}=(1100100)$。

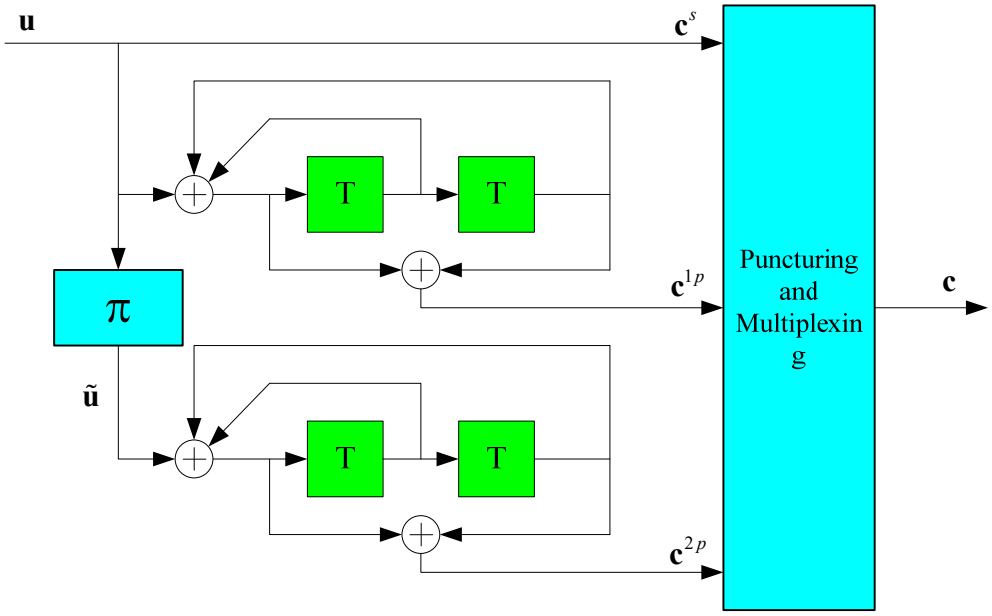
经过交织后的序列 $\tilde{\mathbf{u}}=(1101010)$，从而下面的RSC编码器的输出校验序列 $\mathbf{c}^{2p}=(1000000)$。

所以，未经删余的码率为1/3的turbo码编码器输出码字为

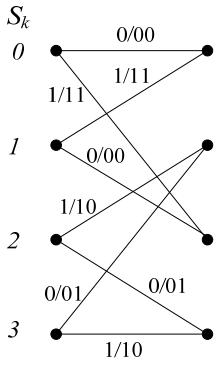
$$\mathbf{c} = (111,010,100,100,010,000,100)$$

若采用 $\mathbf{P} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ 的删余矩阵，则删余后的码率为1/2的turbo码的输出码字为

$$\mathbf{c} = (11,00,10,10,01,00,10)$$



（a）



Branch label: u_k/c_k

(b) 分量码的 trellis 图

图 4.7　(7, 5) Turbo 码编码器

4.2.3 A Heuristic Discussion of the Basic Rationale Behind Turbo codes

It is well known that in a coded system, the code performance is dominated by low weight code words. With the use of RSC encoders and interleavers, the probability that both encoders produce low weight outputs is very low. This is because that the pseudorandom interleaver permutes the input bits, the two input sequences **u** and **ũ** are almost always

different, though of the same weight, and the two encoders will (with high probability) produce parity sequences of different weights.

This can be generalized to multiple parallel concatenation of RSC codes. Refer to Fig. 4.8. Let w be the total parity weight obtained after concatenating. Then,

$$w = \Sigma \text{ the parity weights of } M \text{ component codes} \qquad (4.1)$$

With the use of recursive component codes and random interleaving, the right hand side of (4.1) involves (approximately) independent and identically distributed random variables. The resulting code will have properties very close to those of random codes, and the latter are well-known "good" codes.

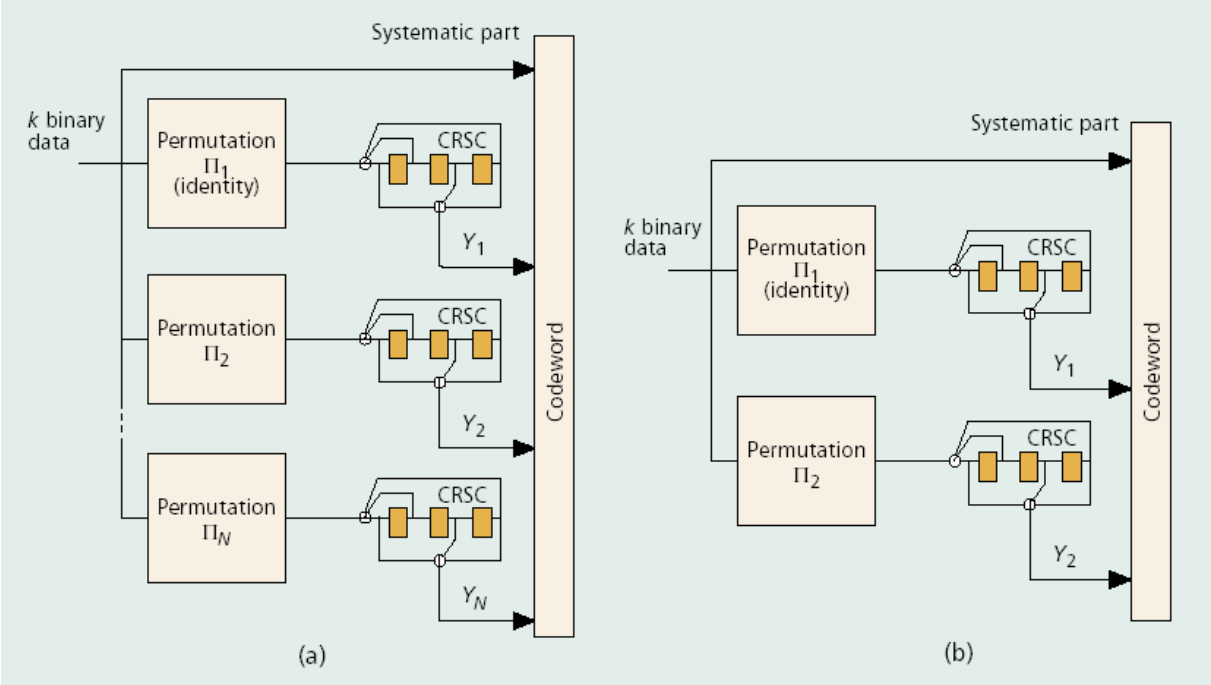


Figure 4.8 *a) In this multiple parallel concatenation of Circular RSC (CRSC) codes, the block containing k information bits is encoded N times. The probability that the sequence remains of the RTZ(return to zero) type after the N pseudorandom permutations is very low. The properties of this multiconcatenated code are very close to those of random codes. b) The number of encodings can be limited to two, provided that permutation Π_2 is judiciously devised. This is a classical turbo code.*

As Dave Forney pointed out: **Rather than attacking error exponents, they (turbo codes) attack multiplicities**.

Remark: ***Good long codes should have a distribution which mimics that of random coding rather than have a large minimum distance.***

Since the constituent encoders are realized in recursive systematic form, a nonzero input bit is required to return the encoder to the all-zero state and thus all detours are associated with information sequences of weight 2 or greater, at least one for departure, and one for the merger. The ML decoding performance analysis shows that the error paths with the minimum information weight $w_{min} = 2$ and their compound paths dominate the BER performance of turbo codes. Let z_{min} denote the lowest weight of the parity check sequence in error paths of

RSC component encoder generated by an information sequence with $w_{\min} = 2$. Then we have the following definition of the *effective free distance* for (un-punctured) turbo codes.

$$d_{\text{free,eff}} = 2 + 2z_{\min}$$

The effective free distance plays a role similar to that of the free distance for convolutional codes.

4.2.4 Structure of Turbo Decoder

如前所述，采用删余技术时 turbo 编码器在 k 时刻的输出为 $\mathbf{c}_k = (c_k^s, c_k^p)$，其中 c_k^p 由 c_k^{1p} 和 c_k^{2p} 交替组成。如果采用 BPSK 调制，则信道上的发送符号为

$$\mathbf{x}_k = (x_k^s, x_k^p) = \left((2c_k^s - 1)\sqrt{E_s}, (2c_k^p - 1)\sqrt{E_s}\right), \quad 1 \le k \le N$$

经过信道传输、相干解调，接收机匹配滤波器在 k 时刻的输出采样值为 $\mathbf{y}_k = (y_k^s, y_k^p)$，译码器的任务就是从此接收序列来估计发送符号。

Turbo 码译码器的基本结构如图 4.9 所示。它由两个软输入软输出（SISO）译码器 dec1 和 dec2 串行级联组成，交织器与编码器中所使用的交织器相同。译码器 dec1 对分量码 RSC1 进行最佳译码，产生关于信息序列 **u** 中每一比特的后验概率信息，并将其中的“新信息”经过交织送给 dec2；译码器 dec2 将此信息作为先验信息，对分量码 RSC2 进行最佳译码，产生关于交织后的信息序列中每一比特的后验概率信息，然后将其中的“新信息”经过解交织送给 dec1，进行下一次译码。这样，经过多次迭代，dec1 或 dec2 新产生的外信息趋于稳定，后验概率比渐进值逼近于对整个码的最大似然译码。

When the series of iterations halts, after either a fixed number of iterations or when a termination criterion is satisfied, the output from the turbo decoder is given by the de-interleaved a-posteriori LLRs of the second component decoder. The sign of these *a-posteriori* LLRs gives the hard decision output $\hat{\mathbf{u}}$.

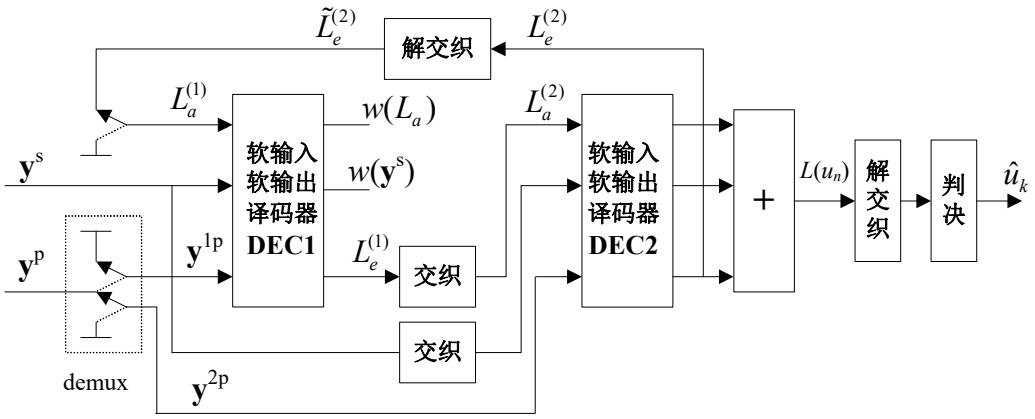


图 4.9　Turbo 码译码器的结构. $\mathbf{y}^s$ is the received systematic bit sequence and $\mathbf{y}^{mp}, m = 1, 2$, are the received parity-check sequences corresponding to the mth constituent encoder.

下面我们对此迭代译码原理做一简要的讨论。假定 turbo 码译码器的接收序列为 $\mathbf{y}=(\mathbf{y}^s,\mathbf{y}^p)$，where $\mathbf{y}^s$ is the received systematic bit sequence and $\mathbf{y}^p$ is the received parity-check sequence. 冗余信息$\mathbf{y}^p$经解复用后得到两个观测序列$\mathbf{y}^{1p}=\{y_k^{1p}\}$和$\mathbf{y}^{2p}=\{y_k^{2p}\}$，$1\le k\le N$，分别送给 dec1 和 dec2。这里，$y_k^{mp}, m=1 \text{ or } 2$, is the noisy observation of x_k^{mp} if x_k^{mp} is not punctured; otherwise, set $y_k^{mp}=0$ (implying no observation obtained). 于是，两个软输出译码器的输入序列分别为：

dec1: $\mathbf{y}^{(1)}=(\mathbf{y}^s,\mathbf{y}^{1p})$,

dec2: $\mathbf{y}^{(2)}=(\mathbf{y}^s,\mathbf{y}^{2p})$

为了使译码后的比特错误概率最小，根据最大后验概率(MAP)译码准则，turbo 译码器的最佳译码策略是，根据接收序列 $\mathbf{y}$ 计算后验概率（APP） $P(u_k)=P\left(u_k\mid\mathbf{y}^{(1)},\mathbf{y}^{(2)}\right)$：

$$\text{MAP rule:}\quad \hat{u}_k=\arg\max_{u\in\{0,1\}}P(u_k=u\mid\mathbf{y}^s,\mathbf{y}^{1p},\mathbf{y}^{2p}) \tag{4.2}$$

显然，式(4.2)的计算对于码长稍微长一点的码来说，计算复杂度太高。在 turbo 码的译码方案中，巧妙地采用了一种次优译码规则，将$\mathbf{y}^{(1)}$和$\mathbf{y}^{(2)}$分开考虑，由两个分量码译码器分别计算后验概率$P\left(u_k\mid\mathbf{y}^{(1)},\mathbf{L}_e^{(2)}\right)$和$P\left(u_k\mid\mathbf{y}^{(2)},\mathbf{L}_e^{(1)}\right)$，然后通过 dec1 和 dec2 之间的多次迭代，使它们收敛于 MAP 译码的$P\left(u_k\mid\mathbf{y}^{(1)},\mathbf{y}^{(2)}\right)$，从而达到逼近 Shannon 限的性能。这里，$\mathbf{L}_e^{(1)}$和$\mathbf{L}_e^{(2)}$为附加信息，其中$\mathbf{L}_e^{(1)}$由 dec1 提供，在 dec2 中用作先验信息；$\mathbf{L}_e^{(2)}$由 dec2 提供，在 dec1 中用作先验信息。具体推导如下：

$$\begin{aligned}P(u_k=u\mid\mathbf{y}^s,\mathbf{y}^{1p},\mathbf{y}^{2p})&=\sum_{\mathbf{u}:u_k=u}P(\mathbf{u}\mid\mathbf{y}^s,\mathbf{y}^{1p},\mathbf{y}^{2p})\\&\propto\sum_{\mathbf{u}:u_k=u}p(\mathbf{y}^s,\mathbf{y}^{1p},\mathbf{y}^{2p}\mid\mathbf{u})P(\mathbf{u})\\&=\sum_{\mathbf{u}:u_k=u}p(\mathbf{y}^{2p}\mid\mathbf{u})p(\mathbf{y}^s,\mathbf{y}^{1p}\mid\mathbf{u})P(\mathbf{u})\\&\propto\sum_{\mathbf{u}:u_k=u}p(\mathbf{y}^{2p}\mid\mathbf{u})P(\mathbf{u}\mid\mathbf{y}^s,\mathbf{y}^{1p})\end{aligned}$$

From distribution separation,

$$P(\mathbf{u}\mid\mathbf{y}^s,\mathbf{y}^{1p})\approx\prod_{n=1}^{N}P(u_n\mid\mathbf{y}^s,\mathbf{y}^{1p})\propto\prod_{n=1}^{N}p(\mathbf{y}^s,\mathbf{y}^{1p}\mid u_n)P(u_n)$$

Thus,

$$P(u_k = u \mid \mathbf{y}^s, \mathbf{y}^{1p}, \mathbf{y}^{2p}) \propto \sum_{\mathbf{u}:u_k=u} p(\mathbf{y}^{2p} \mid \mathbf{u}) \underbrace{\prod_{n=1}^{N} p(\mathbf{y}^s, \mathbf{y}^{1p} \mid u_n) P(u_n)}$$

Used as the priori probability for the 2nd decoder

Since the separation assumption destroyed optimality of the equation, the algorithm is iterated several times through the two decoders.

需要说明的是，通过迭代译码提高编码方案性能的思想，早在 60 年代 Gallager 提出的低密度校验码（LDPC）中已经萌芽[Gallager1963]。在那里 Gallager 将信息分为内信息和外信息两种，并指出只有外信息对迭代译码是有用的。但限于当时的实验条件与研究方法，他没有给出迭代算法正确性的证明，只是通过一个例子说明了长的 LDPC 码可以达到 BSC 信道的容量。否则，信道编码理论的发展历史可能要被重新书写。

另外，既然只有外信息可用于迭代，那么在迭代译码中，分量译码器之间的信息交换应不相关，即要求交织器应最大程度地置乱原信息序列。

关于 $P\left(u_k \mid \mathbf{y}^{(1)}, \mathbf{L}_e^{(2)}\right)$ 和 $P\left(u_k \mid \mathbf{y}^{(2)}, \mathbf{L}_e^{(1)}\right)$ 的求解，目前已有多种方法，它们构成了 turbo 码的不同译码算法，我们将在后续章节中详细介绍。下面我们以 BCJR 的前向-后向 MAP 软输出算法为例来讨论 turbo 码的译码。

4.3 Principle of Iterative Decoding Algorithm

4.3.1 Log Likelihood Ratios

The concept of Log-Likelihood Ratios (LLRs) was shown by Robertson [] to simplify the passing of information from one component decoder to the other in the iterative decoding of turbo codes, and so is now widely used in the turbo coding literature. Let u be in GF(2) with elements {+1, -1}, where '+1' is the null element under the addition operation. The LLR or L-value of the binary variable u is defined to be

$$L(u) \triangleq \ln\left(\frac{P(u=+1)}{P(u=-1)}\right) \tag{4.3}$$

Here, $P(u = i)$ denotes the probability that the random variable u takes on the value i. Notice that the two possible values for the bit u are taken to be +1 and -1, rather than 0 and 1. This definition of the two values of a binary variable makes no conceptual difference, but it slightly simplifies the mathematics in the derivations which follow.

Figure 4.10 shows how $L(u)$ varies as the probability of u=+1 varies. It can be seen from this figure that the sign of the LLR $L(u)$ of a bit u is the hard decision, and the magnitude of the LLR gives the reliability of this decision (i.e., indication of how likely it is that the sign of the LLR gives the correct value of u.)

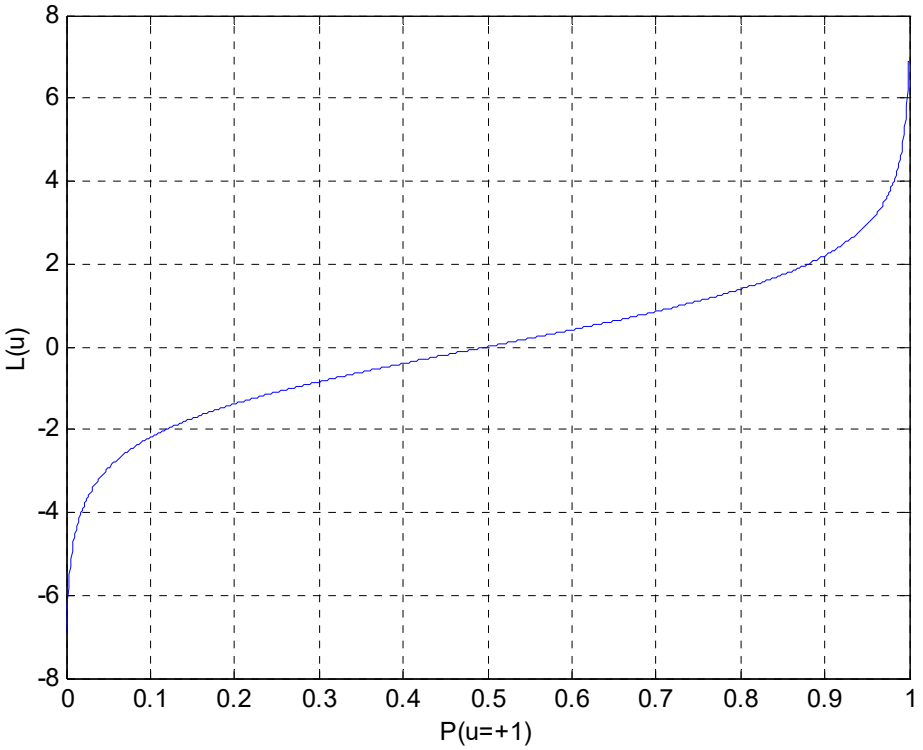


Figure 4.10 Log Likelihood Ratio $L(u)$ versus the Probability of $u = +1$

For a data bit u_k, given the LLR $L(u_k)$, it is possible to calculate the probability that $u_k = +1$ or $u_k = -1$. Noting that $P(u_k = -1) = 1 - P(u_k = +1)$, and taking the exponent of both sides in (4.3) we have

$$e^{L(u_k)} = \frac{P(u_k = +1)}{1 - P(u_k = +1)}$$

so

$$P(u_k = +1) = \frac{e^{L(u_k)}}{1 + e^{L(u_k)}} = \frac{1}{1 + e^{-L(u_k)}}$$

and

$$P(u_k = -1) = \frac{1}{1 + e^{L(u_k)}} = \frac{e^{-L(u_k)}}{1 + e^{-L(u_k)}}$$

Hence, we can write

$$P(u_k = \pm 1) = \left(\frac{e^{-L(u_k)/2}}{1 + e^{-L(u_k)}} \right) \cdot e^{\pm L(u_k)/2} = C_k \cdot e^{u_k L(u_k)/2} \tag{4.4}$$

where $C_k = \frac{e^{-L(u_k)/2}}{1 + e^{-L(u_k)}}$ depends only on the LLR $L(u_k)$ and not on whether u_k is +1 or –1, so it

can be treated as a constant. Eq. (4.4) can also be expressed as

$$P(u_k = \pm 1) = \frac{e^{\pm L(u_k)/2}}{e^{L(u_k)/2} + e^{-L(u_k)/2}}$$

We define the soft bit $\lambda(u_k)$ as the expectation of u_k. From the discussion above, we have

$$\begin{aligned}\lambda(u_k) &= \mathsf{E}[u_k] \\ &= (+1)\cdot P(u_k = +1) + (-1)\cdot P(u_k = -1) \\ &= \tanh\left(\frac{L(u_k)}{2}\right)\end{aligned}$$

If $P(u_k)$ is a random variable in the range of (0, 1), then $\lambda(u_k)$ is a random variable (r.v.) in the range (-1, +1). For two binary random variables u_1 and u_2, their GF(2) addition $u_1 \oplus u_2$ transforms into $u_1 \cdot u_2$. Since $E[u_1 \cdot u_2] = E[u_1] \cdot E[u_2] = \lambda(u_1) \cdot \lambda(u_2)$, the LLR of the sum $L(u_1 \oplus u_2)$ equals

$$2\tanh^{-1}\left(\tanh\left(\frac{L(u_1)}{2}\right)\cdot\tanh\left(\frac{L(u_2)}{2}\right)\right) = L(u_1) \otimes L(u_2)$$

abbreviated by the **boxplus operation** ⊗. This relation is often used in the decoding of LDPC codes, and appeared in the following form.

定义（Box-Plus 运算）：对于两个二元独立随机变量 u 和 v，它们的 box-plus 运算定义为

$$L(u) \otimes L(v) = L(u \oplus v) = \log\frac{P(u \oplus v = 0)}{P(u \oplus v = 1)} = \log\frac{e^{L(u)+L(v)} + 1}{e^{L(u)} + e^{L(v)}}$$

As well as the LLR $L(u_k)$ based on the unconditional probabilities $P(u_k = \pm 1)$, we are also interested in LLRs based on conditional probabilities. For example, we often use the conditional LLRs based on the probability that the receiver's matched filter output would be y_k given that the corresponding transmitted bit $x_k = +1$ or -1. This conditional LLR, denoted by $P(y_k \mid x_k)$, is defined as

$$L(y_k \mid x_k) \triangleq \ln\left(\frac{P(y_k \mid x_k = +1)}{P(y_k \mid x_k = -1)}\right) \tag{4.5}$$

If we assume that x_k is transmitted over the (possibly fading) AWGN channel using BPSK

modulation, then we have

$$p(y_k \mid x_k = +1) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{E_s}{2\sigma^2}(y_k - a)^2 \right)$$

and

$$p(y_k \mid x_k = -1) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{E_s}{2\sigma^2}(y_k + a)^2 \right)$$

where E_s is the average transmitted energy per symbol, $\sigma^2 = N_0/2$ is the variance of noise samples and a is the fading amplitude.
Thus, with the channel model above, (4.5) can be written as

$$\begin{aligned} L(y_k \mid x_k) &= \ln\left(\frac{\exp\left(-\frac{E_s}{2\sigma^2}(y_k - a)^2 \right)}{\exp\left(-\frac{E_s}{2\sigma^2}(y_k + a)^2 \right)} \right) \\ &= -\frac{E_s}{2\sigma^2}(y_k - a)^2 + \frac{E_s}{2\sigma^2}(y_k + a)^2 \\ &= \frac{4aE_s}{2\sigma^2} y_k \\ &= L_c y_k \end{aligned}$$

where

$$L_c \triangleq \frac{4aE_s}{N_0} \tag{4.6}$$

is defined as the *channel reliability value*, and depends only on the SNR and fading amplitude of the channel. Hence, for BPSK over a (possibly fading) Gaussian channel, the conditional LLR $L(y_k \mid x_k)$, which is referred to as the *soft output* of the channel, is simply the matched filter output y_k multiplied by the channel reliability value L_c.

4.3.2 The Maximum A Posteriori Algorithm

Having introduced log likelihood ratios, we now proceed to describe the operation of the Maximum A-Posteriori algorithm, which is one of the possible soft-in soft-out component decoders that can be used in an iterative turbo decoder.

4.3.2.1 Introduction

In 1974 an algorithm, which is known as the Maximum *A-Posteriori* (MAP) algorithm, was proposed by Bahl, Cocke, Jelinek and Raviv for estimating the a-posteriori probabilities of the states and the transitions of a Markov source observed, when subjected to memoryless noise. This algorithm has also become known as the BCJR algorithm, named after its inventors. They showed, how the algorithm could be used for decoding both block and

convolutional codes. When employed for decoding convolutional codes, the algorithm is optimal in terms of minimizing the decoded bit error rate, unlike the Viterbi algorithm, which minimizes the probability of an incorrect path through the trellis being selected by the decoder.

However the MAP algorithm provides not only the estimated bit sequence, but also the probabilities for each bit that it has been decoded correctly. This is essential for the iterative decoding of turbo codes.

We often use Bayes' rule in this section. This rule may be briefly stated as

$$P(u,v) = P(u \mid v)P(v)$$

A usedul consequence of Bayes's rule is that

$$P(u,v \mid w) = P(u \mid v,w)P(v \mid w)$$

4.3.2.2 Symbol-by-Symbol MAP decoding algorithm for a convolutional code

Consider the system model shown in Fig. 4.11. Let $\mathbf{u} = (u_1, u_2, ..., u_N), u_k \in \{0,1\}$ be the input information sequence, which is encoded by a rate-1/2 (either systematic or non-systematic) convolutional encoder into a coded symbol sequence $\mathbf{c} = (\mathbf{c}_1, \mathbf{c}_2, ..., \mathbf{c}_N)$, where $\mathbf{c}_k = (c_k^s, c_k^p), 1 \le k \le N$. The coded symbols, $c_k^s \in \{0,1\}$ and $c_k^p \in \{0,1\}$, are then modulated in BPSK format, producing the transmitted symbols, x_k^s and x_k^p.

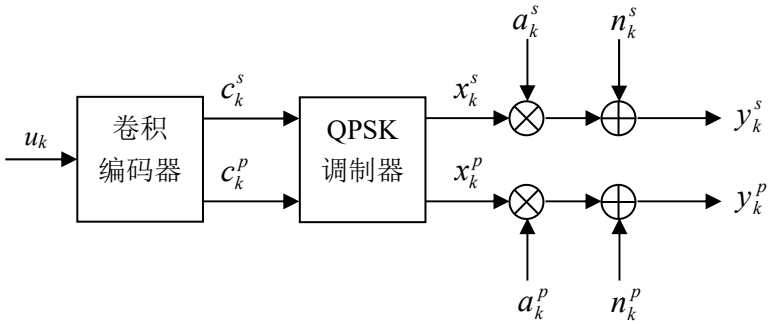


图 4.11　信道模型

With the channel model shown in Fig. 4.11, the received sequence $\mathbf{y} = \mathbf{y}_1^N = (y_1, y_2 \cdots, y_k, \cdots, y_N)$ consists of N symbol pairs $\mathbf{y}_k = (y_k^s, y_k^p)$, where

$$y_k^s = a_k^s x_k^s + n_k^s = a_k^s (2c_k^s - 1)\sqrt{E_s} + n_k^s \tag{4.7}$$

$$y_k^p = a_k^p x_k^p + n_k^p = a_k^p (2c_k^p - 1)\sqrt{E_s} + n_k^p$$

式中 E_s 是每发送符号的平均能量；a_k^s和a_k^p 为信道衰落因子，对于 AWGN 信道，$a_k^s = a_k^p = 1$；　对于 Rayleigh 衰落信道，a_k^s和a_k^p 为两个 Reyleigh 随机变量。n_k^s和n_k^p 是两

个独立同分布的高斯噪声样值，它们的均值为 0，方差 $\sigma_n^2 = N_0/2$。

式(4.7)可以等效地写为：

$$y_k^s = a_k^s(2c_k^s - 1) + n_k^s$$
$$y_k^p = a_k^p(2c_k^p - 1) + n_k^p$$

式中，噪声 n_k^s and n_k^p 的方差成为 $\sigma^2 = \sigma_n^2 / E_s = N_0/(2E_s)$.

Assume that the convolutional encoder used has *v* memory elements and a constraint length of *K*. Let $S_k = (a_k, a_{k-1}, \cdots, a_{k-v+1})$ be the encoder state at time *k*. Denote by $\mathcal{S}$ the set of encoder states and by $M = |\mathcal{S}|$ the number of states. In the following, we will consider the optimal decoding of **u** from **y**.

考虑图 4.12 所示的软输入软输出（SISO）译码器，它能为每一译码比特提供对数似然比输出。

图 4.12　软输入软输出译码器框图

图中 MAP 译码器的输入序列为 $\mathbf{y} = \mathbf{y}_1^N = (\mathbf{y}_1, \mathbf{y}_2 \cdots, \mathbf{y}_k, \cdots, \mathbf{y}_N)$，其中 $\mathbf{y}_k = (y_k^s, y_k^p)$。$L_a(u_k)$ 是关于 u_k 的先验信息，$L(u_k)$ 是关于 u_k 的对数 APP(似然)比。它们的定义如下：

$$L_a(u_k) \equiv \ln \frac{P(u_k = 1)}{P(u_k = 0)}$$

$$L(u_k) \equiv \ln \frac{P(u_k = 1|\mathbf{y}_1^N)}{P(u_k = 0|\mathbf{y}_1^N)} \tag{4.8}$$

MAP 译码器的任务就是求解式(4.8)，然后按照下列规则进行判决：

$$\hat{u}_k = \begin{cases} 1, & L(u_k) \geq 0 \\ 0, & L(u_k) < 0 \end{cases} \tag{4.9}$$

下面我们利用 BCJR 算法对式(4.8)的计算方法进行推导。

A branch at the *k*th section in the trellis diagram can be specified by $b_k \equiv (S_{k-1}, u_k, \mathbf{c}_k, S_k)$, where u_k and $\mathbf{c}_k$ are the information and coded symbols associated with the state transition $S_{k-1} \to S_k$. Denote by $B_k(s', s)$ the set of all the parallel branches

connecting $S_{k-1} = s' \in \mathcal{S}$ and $S_k = s \in \mathcal{S}$.

Using Bayes' rule, the *a posteriori* probability can be expressed as

$$P(u_k = i \mid \mathbf{y}_1^N) = \sum_{\forall (s',s) \in B_k^i} P(u_k = i, S_{k-1} = s', S_k = s \mid \mathbf{y}_1^N)$$

$$= \sum_{\forall (s',s) \in B_k^i} \frac{p(u_k = i, S_{k-1} = s', S_k = s, \mathbf{y}_1^N)}{p(\mathbf{y}_1^N)}$$

where B_k^i is the set of transitions $S_{k-1} \to S_k$ that are caused by the input $u_k=i$. Thus (4.8) can be written as

$$L(u_k) = \ln \frac{\sum_{\forall (s',s) \in B_k^1} p(u_k = 1, S_{k-1} = s', S_k = s, \mathbf{y}_1^N) / p(\mathbf{y}_1^N)}{\sum_{\forall (s',s) \in B_k^0} p(u_k = 0, S_{k-1} = s', S_k = s, \mathbf{y}_1^N) / p(\mathbf{y}_1^N)} \tag{4.10}$$

The sequence $\mathbf{y}$ in $p(u_k = i, S_{k-1} = s', S_k = s, \mathbf{y}_1^N)$ can be written as

$$p\left(u_k, S_{k-1}, S_k, (\mathbf{y}_1, ..., \mathbf{y}_{k-1}), \mathbf{y}_k, (\mathbf{y}_{k+1}, ..., \mathbf{y}_N)\right) = p\left(u_k, S_{k-1}, S_k, \mathbf{y}_1^{k-1}, \mathbf{y}_k, \mathbf{y}_{k+1}^N\right)$$

where $\mathbf{y}_t^l = (\mathbf{y}_t, \mathbf{y}_{t+1}, ..., \mathbf{y}_l)$ represents the portion of the received sequence $\mathbf{y}$ in the time interval t to l.

Applying Bayes' rule to this expression yields the following decomposition of $p(u_k = i, S_{k-1} = s', S_k = s, \mathbf{y}_1^N)$:

$$p\left(u_k = i, S_{k-1} = s', S_k = s, \mathbf{y}_1^N\right) = p\left(u_k = i, S_k = s, S_{k-1} = s', \mathbf{y}_1^{k-1}, \mathbf{y}_k, \mathbf{y}_{k+1}^N\right)$$

$$= p\left(\mathbf{y}_{k+1}^N \mid \mathbf{y}_k, \mathbf{y}_1^{k-1}, u_k = i, S_k = s, S_{k-1} = s\right) \cdot p\left(\mathbf{y}_k, \mathbf{y}_1^{k-1}, u_k = i, S_k = s, S_{k-1} = s'\right)$$

$$= p\left(\mathbf{y}_{k+1}^N \mid S_k = s\right) \cdot p\left(\mathbf{y}_k, \mathbf{y}_1^{k-1}, u_k = i, S_k = s, S_{k-1} = s'\right) \tag{4.11}$$

$$= p\left(\mathbf{y}_{k+1}^N \mid S_k = s\right) \cdot p\left(u_k = i, S_k = s, \mathbf{y}_k \mid S_{k-1} = s', \mathbf{y}_1^{k-1}\right) \cdot p\left(S_{k-1} = s', \mathbf{y}_1^{k-1}\right)$$

$$= p\left(S_{k-1} = s', \mathbf{y}_1^{k-1}\right) \cdot p\left(u_k = i, S_k = s, \mathbf{y}_k \mid S_{k-1} = s'\right) \cdot p\left(\mathbf{y}_{k+1}^N \mid S_k = s\right)$$

$$= \alpha_{k-1}(s') \cdot \gamma_k^i(s', s) \cdot \beta_k(s) \tag{4.12}$$

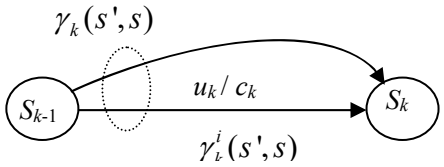
where

$\alpha_k(s) \equiv p(S_k = s, \mathbf{y}_1^k)$, called the forward state metric, is the joint of the encoder state at time k being $S_k=s$ and receiving the sequence $\mathbf{y}_1^k$;

$\beta_k(s) \equiv p(\mathbf{y}_{k+1}^N | S_k = s)$ is the backward state metric; and

$\gamma_k^i(s',s) \equiv p\left(u_k = i, S_k = s, \mathbf{y}_k \mid S_{k-1} = s'\right)$ is the metric of the trellis branch connecting $S_{k-1} = s'$ to $S_k = s$ with $u_k = i$.

Equation (4.11) follows from the Markov property of the encoder; i.e., if $S_k = s$ is known, events after time k are independent of the history $\mathbf{y}_1^k$ up to S_k.



Define

$$\gamma_k(s',s) \equiv p\left(S_k = s, \mathbf{y}_k \mid S_{k-1} = s'\right)$$

$$= \sum_{\forall b_k \in B_k(s',s)} \gamma_k^i(s',s) \tag{4.13}$$

as the state transition probability from $S_{k-1} = s'$ to $S_k = s$. If there exit parallel transitions between $S_{k-1} = s'$ and $S_k = s$ in the trellis diagram, then $\gamma_k(s',s)$ is equal to the sum of the metrics of the corresponding parallel branches.

Substituting (4.12) into (4.10), we have

$$L(u_k) = \ln \frac{\sum_{\forall (s',s) \in B_k^1} \alpha_{k-1}(s') \cdot \gamma_k^1(s',s) \cdot \beta_k(s) / p(\mathbf{y}_1^N)}{\sum_{\forall (s',s) \in B_k^0} \alpha_{k-1}(s') \cdot \gamma_k^0(s',s) \cdot \beta_k(s) / p(\mathbf{y}_1^N)} \tag{4.14}$$

下面我们来求$\alpha_k(s), \beta_k(s)$和$\gamma_k(s',s)$。By definition, the term $\alpha_k(s)$ can be computed via the recursion

$$\begin{aligned}
\alpha_k(s) &= \sum_{s' \in \mathcal{S}} p(S_k = s, S_{k-1} = s', \mathbf{y}_1^k) \\
&= \sum_{s' \in \mathcal{S}} p(S_{k-1} = s', \mathbf{y}_1^{k-1}) \cdot p(S_k = s, \mathbf{y}_k \mid S_{k-1} = s', \mathbf{y}_1^{k-1}) \\
&= \sum_{s' \in \mathcal{S}} p(S_{k-1} = s', \mathbf{y}_1^{k-1}) \cdot p(S_k = s, \mathbf{y}_k \mid S_{k-1} = s') \\
&= \sum_{s' \in \mathcal{S}} \alpha_{k-1}(s') \cdot \gamma_k(s',s)
\end{aligned} \tag{4.15}$$

with the initial value $\alpha_0(s) = P(S_0 = s)$.

Similarly, $\beta_k(s)$ can be obtained via the following backward recursion

$$\beta_{k-1}(s') = \sum_{s\in\mathcal{S}} p(S_k = s, \mathbf{y}_k^N \mid S_{k-1} = s')$$
$$= \sum_{s\in\mathcal{S}} p(\mathbf{y}_{k+1}^N \mid S_k = s)\cdot p(S_k = s, \mathbf{y}_k \mid S_{k-1} = s')$$
$$= \sum_{s\in\mathcal{S}} \beta_k(s)\cdot\gamma_k(s', s) \tag{4.16}$$

with boundary condition $\beta_N(s) = P(S_N = s)$.

The term $\gamma_k^i(s', s)$ can be further decomposed into

$$\gamma_k^i(s', s) = P(u_k = i \mid S_{k-1} = s')\cdot p(S_k = s, \mathbf{y}_k \mid S_{k-1} = s', u_k = i)$$
$$= P(u_k = i \mid S_{k-1} = s')\cdot P(S_k = s \mid S_{k-1} = s', u_k = i)\cdot p(\mathbf{y}_k \mid S_k = s, S_{k-1} = s', u_k = i)$$
$$= P(u_k = i)\cdot P(S_k = s \mid S_{k-1} = s', u_k = i)\cdot p(\mathbf{y}_k \mid \mathbf{c}_k) \tag{4.17}$$

where $P(u_k)$ is the *a priori* probability of $u_k = i$, and $P(S_k = s \mid S_{k-1} = s', u_k = i)$ is the probability that a state transition $S_{k-1} \to S_k$ with input $u_k = i$ exists, which is equal to either 1 or 0. $p(\mathbf{y}_k \mid \mathbf{c}_k) \equiv p(\mathbf{y}_k \mid S_k = s, S_{k-1} = s', u_k = i)$ is the probability of receiving $\mathbf{y}_k$ given the specific trellis branch, which is determined by the channel transition probability. For the discerte memoryless AWGN channel, it can be calculated as

$$p(\mathbf{y}_k \mid \mathbf{c}_k) = p(y_k^s, y_k^p \mid c_k^s, c_k^p)$$
$$= p(y_k^s \mid c_k^s)\cdot p(y_k^p \mid c_k^p)$$
$$= \frac{1}{\sqrt{2\pi}\sigma}\exp\left\{-\frac{E_s}{2\sigma^2}\left[y_k^s - (2c_k^s - 1)\right]^2\right\}$$
$$\cdot\frac{1}{\sqrt{2\pi}\sigma}\exp\left\{-\frac{E_s}{2\sigma^2}\left[y_k^p - (2c_k^p - 1)\right]^2\right\}$$
$$= \frac{1}{2\pi\sigma^2}\exp\left\{-\frac{(y_k^s)^2 - 2y_k^s(2c_k^s - 1) + 1}{2\sigma^2}E_s\right\}\exp\left\{-\frac{(y_k^p)^2 - 2y_k^p(2c_k^p - 1) + 1}{2\sigma^2}E_s\right\}$$
$$= \frac{1}{2\pi\sigma^2}\exp\left\{-\frac{(y_k^s)^2 + (y_k^p)^2 + 2}{2\sigma^2}E_s\right\}\exp\left\{-\frac{E_s(y_k^s + y_k^p)}{\sigma^2}\right\}\exp\left\{\frac{2E_s(y_k^s c_k^s + y_k^p c_k^p)}{\sigma^2}\right\}$$
$$= A_k\exp\left\{\frac{2E_s(y_k^s c_k^s + y_k^p c_k^p)}{\sigma^2}\right\}$$

$$= A_k \exp\left\{\frac{4E_s(y_k^s c_k^s + y_k^p c_k^p)}{N_0}\right\} \tag{4.18}$$

where A_k is a constant that is independent of code bits. As aforementioned, let

$$L_c = 4E_s / N_0 \tag{4.19}$$

which is called the reliability value of the channel. Then (4.18) can be written as

$$p(y_k \mid S_k = s, S_{k-1} = s', u_k = i) = A_k \exp\left\{L_c y_k^s c_k^s + L_c y_k^p c_k^p\right\} \tag{4.20}$$

Therefore,

$$\gamma_k^i(s',s) = \begin{cases} P(u_k = i) \cdot A_k \exp\{L_c y_k^s c_k^s + L_c y_k^p c_k^p\}, & \text{if } P(S_k = s \mid S_{k-1} = s', u_k = i) = 1 \\ 0, & \text{otherwise} \end{cases} \tag{4.21}$$

$$= P(u_k = i) \cdot A_k \exp\{L_c y_k^s c_k^s + L_c y_k^p c_k^p\} \cdot \delta_k^i(s',s)$$

where $\delta_k^i(s',s) \equiv P(S_k = s \mid S_{k-1} = s', u_k = i)$.

至此，如果将式(4.14)分子分母中的 $p(\mathbf{y}_1^N)$ 约掉， $L(u_k)$ 的求解已基本完成。然而，由于式(4.20)是从连续随机变量的概率密度计算得到， $\gamma_k(s',s)$ 的值可能大于 1，这会使得式(4.15)、(4.16)产生上溢出，导致整个算法不稳定。另外，由于下列原因，也可能导致下溢出：

Some state metric becomes vanishingly small as k increases, resulting in errors due to precision limitations. This is evident by considering the summation

$$\sum_{s \in \mathcal{S}} \alpha_k(s) = p(\mathbf{y}_1^k)$$

The probability of receiving a particular sequence, $p(\mathbf{y}_1^k)$, becomes very small after a few time instants k.

因此，有必要对 $\alpha_k(s), \beta_k(s)$ 进行归一化处理。

令

$$\tilde{\alpha}_k(s) = \alpha_k(s) / p(\mathbf{y}_1^k)$$
$$\tilde{\beta}_k(s) = \beta_k(s) / p(\mathbf{y}_{k+1}^N | \mathbf{y}_1^k) \tag{4.22}$$

因为 $p(\mathbf{y}_1^k) = \sum_s p(S_k = s, \mathbf{y}_1^k)$，所以

$$\tilde{\alpha}_k(s) = \alpha_k(s) / \sum_s \alpha_k(s) \tag{4.23}$$

将式(4.15)代入上式，并且分子分母同除以 $p(\mathbf{y}_1^{k-1})$，得到

$$\tilde{\alpha}_k(s) = \frac{\sum_{s'} \alpha_{k-1}(s')\gamma_k(s',s) / p(\mathbf{y}_1^{k-1})}{\sum_s \sum_{s'} \alpha_{k-1}(s')\gamma_k(s',s) / p(\mathbf{y}_1^{k-1})}$$

$$= \frac{\sum_{s'} \tilde{\alpha}_{k-1}(s')\gamma_k(s',s)}{\sum_{s}\sum_{s'} \tilde{\alpha}_{k-1}(s')\gamma_k(s',s)} \tag{4.24}$$

对于 $\tilde{\beta}_k(s)$，考虑到 $p(\mathbf{y}_k^N|\mathbf{y}_1^{k-1}) = p(\mathbf{y}_{k+1}^N|\mathbf{y}_1^k)p(\mathbf{y}_1^k) / p(\mathbf{y}_1^{k-1})$，于是有

$$\tilde{\beta}_{k-1}(s') = \frac{\beta_{k-1}(s')}{p(\mathbf{y}_k^N \mid \mathbf{y}_1^{k-1})} = \frac{\sum_s \beta_k(s)\gamma_k(s',s)}{p(\mathbf{y}_{k+1}^N|\mathbf{y}_1^k)p(\mathbf{y}_1^k) / p(\mathbf{y}_1^{k-1})}$$

$$= \frac{\sum_s \beta_k(s)\gamma_k(s',s) / p(\mathbf{y}_{k+1}^N|\mathbf{y}_1^k)}{\sum_s \alpha_k(s) / p(\mathbf{y}_1^{k-1})}$$

$$= \frac{\sum_s \tilde{\beta}_k(s)\gamma_k(s',s)}{\sum_s\sum_{s'} \alpha_{k-1}(s')\gamma_k(s',s) / p(\mathbf{y}_1^{k-1})}$$

$$= \frac{\sum_s \tilde{\beta}_k(s)\gamma_k(s',s)}{\sum_s\sum_{s'} \tilde{\alpha}_{k-1}(s')\gamma_k(s',s)} \tag{4.25}$$

合并式(4.12)和(4.22)得

$$p(u_k = i, S_{k-1} = s', S_k = s, \mathbf{y}_1^N) = \tilde{\alpha}_{k-1}(s')p(\mathbf{y}_1^{k-1}) \cdot \gamma_k^i(s',s) \cdot \tilde{\beta}_k(s)p(\mathbf{y}_{k+1}^N \mid \mathbf{y}_1^k)$$

$$= \tilde{\alpha}_{k-1}(s') \cdot \gamma_k^i(s',s) \cdot \tilde{\beta}_k(s) \cdot p(\mathbf{y}_1^{k-1})p(\mathbf{y}_1^N) / p(\mathbf{y}_1^k)$$

$$= \tilde{\alpha}_{k-1}(s/) \cdot \gamma_k^i(s',s) \cdot \tilde{\beta}_k(s) \cdot p(\mathbf{y}_1^N) / p(y_k \mid \mathbf{y}_1^{k-1}) \tag{4.26}$$

将上式代入式(4.10)，并且分子分母同乘以因子 $p(y_k|\mathbf{y}_1^{k-1})$，便得最终计算公式

$$L(u_k) = \ln\left(\frac{\sum_{\forall(s',s)\in B_k^1} \tilde{\alpha}_{k-1}(s') \cdot \gamma_k^1(s',s) \cdot \tilde{\beta}_k(s)}{\sum_{\forall(s',s)\in B_k^0} \tilde{\alpha}_{k-1}(s') \cdot \gamma_k^0(s',s) \cdot \tilde{\beta}_k(s)}\right) \tag{4.27}$$

这样，我们就完成了分量码的 MAP 译码算法的推导。$\tilde{\alpha}_k(s)$和$\tilde{\beta}_k(s)$ 的递推运算示意于图 4.13，其中 $\tilde{\alpha}_k(s)$ 的初始条件为(假定 RSC 编码器的初始状态为零状态)：

$$\tilde{\alpha}_0(0) = 1, \quad \tilde{\alpha}_0(s \neq 0) = 0 \tag{4.28}$$

如果编码器在每帧编码完成之后通过结尾（termination）处理也回到零状态,那么 $\tilde{\beta}_k(s)$ 递推的初始条件为：

$$\tilde{\beta}_N(0) = 1, \quad \tilde{\beta}_N(s \neq 0) = 0 \tag{4.29}$$

否则，应设定为：

$$\tilde{\beta}_N(s) = 1/2^v \quad \forall s \in \mathcal{S} \tag{4.30}$$

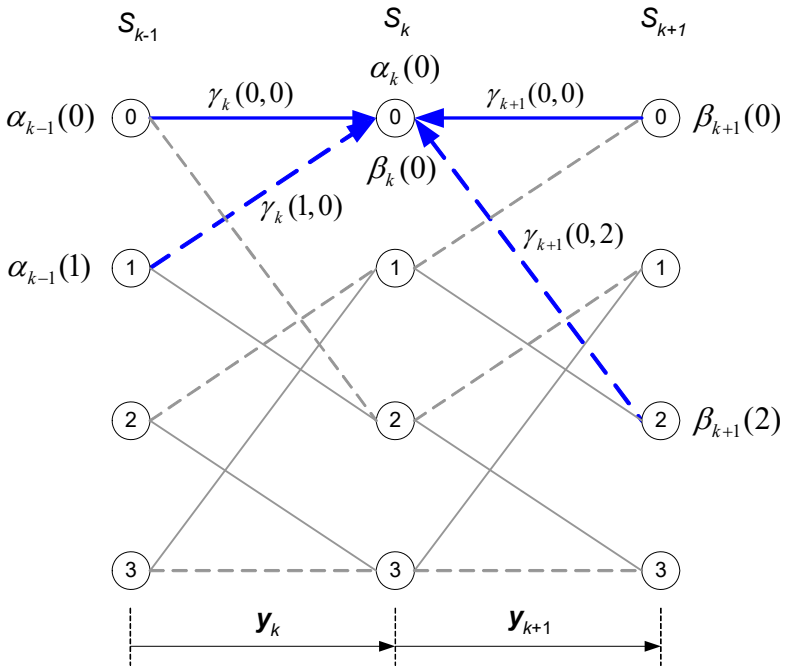


图 4.13 $\tilde{\alpha}_k(s)$和$\tilde{\beta}_k(s)$ 的递推计算示意图. $\alpha_k(0) = \alpha_{k-1}(0)\gamma_k(0,0) + \alpha_{k-1}(1)\gamma_k(1,0)$

and $\beta_k(0) = \beta_{k+1}(0)\gamma_{k+1}(0,0) + \beta_{k+1}(2)\gamma_{k+1}(0,2)$

Note: The MAP algorithm is often called the BCJR algorithm or forward-backward algorithm. They are equivalent to the Baum-Welch algorithm used in Hidden Markov models.

The MAP algorithm can be summarized as follows.

- **Initialization**

  $\tilde{\alpha}_0(S_0 = 0) = 1, \quad \tilde{\alpha}_0(S_0 = s) = 0$ for $\forall s \neq 0$

  $\tilde{\beta}_N(0) = 1, \quad \tilde{\beta}_N(\forall s \neq 0) = 0$

- **Forward recursion**

  For $k = 1$ to N, do

  1) For $i = 0, 1$, compute and store the branch metrics $\gamma_k^i(s', s)$ as in (4.21);

  2) For $s \in \mathcal{S}$, compute and store the forward metrics $\tilde{\alpha}_k(s)$ as in (4.24)

- **Backward recursion**

For $k = N$ -1 to 0, do

Compute and store the backward metrics $\tilde{\beta}_k(s)$, $\forall s \in \mathcal{S}$, as in (4.25), using the branch metric computed in the forward recursion.

- **Output**

  For k = 1 to N, compute and output the log APP ratio $L(u_k)$ as in (4.27).

4.3.3 Iterative Decoding of Turbo Codes

利用 Bayes 规则，由式(4.8)我们有

$$L(u_k) = \ln \frac{p(\mathbf{y}_1^N \mid u_k = 1)}{p(\mathbf{y}_1^N \mid u_k = 0)} + \ln \frac{P(u_k = 1)}{P(u_k = 0)}$$

$$= \ln \frac{p(\mathbf{y}_1^N \mid u_k = 1)}{p(\mathbf{y}_1^N \mid u_k = 0)} + L_a(u_k) \tag{4.31}$$

其中 $L_a(u_k)$ 是关于 u_k 的先验信息。在以往的译码方案中，通常认为先验等概，因而 $L_a(u_k) = 0$。而在迭代译码方案中，Extrinsic information from another decoder, $L_e(u_k)$, can be used as the priori information, $L_a(u_k)$。为了能使迭代继续进行，当前译码器应从式(4.31)的第一项中提取出新的外信息并且提供给下一级译码器。下面我们以上一节中的 MAP 算法为例，说明如何在两个分量码译码器之间传递软信息。

由 $L_a(u_k)$ 的定义式(4.8)可得

$$\exp\{L_a(u_k)\} = \frac{P(u_k = 1)}{1 - P(u_k = 1)}$$

从上式我们得到

$$P(u_k = 1) = \frac{\exp\{L_a(u_k)\}}{1 + \exp\{L_a(u_k)\}}$$

因为 $P(u_k = 0) = 1 - P(u_k = 1)$，所以有

$$P(u_k = 0) = \frac{1}{1 + \exp\{L_a(u_k)\}}$$

从而，

$$P(u_k = i) = \frac{\exp\{i \cdot L_a(u_k)\}}{1 + \exp\{L_a(u_k)\}} = D_k \exp\{iL_a(u_k)\} \tag{4.32}$$

其中 $D_k = \dfrac{1}{1+\exp\{L_a(u_k)\}}$ 为常量。

Substituting (4.32) into (4.21), the branch transition probability can be written as

$$\gamma_k^i(s',s) = A_k D_k \exp\left\{iL_a(u_k) + L_c y_k^s c_k^s + L_c y_k^p c_k^p\right\} P(S_k = s \mid S_{k-1} = s', u_k = i)$$

For a systematic encoder, $c_k^s = u_k = i$. Thus

$$\gamma_k^i(s',s) = K \exp\left\{iL_a(u_k) + L_c y_k^s u_k\right\} \cdot \exp\left\{L_c y_k^p c_k^p\right\} P(S_k = s \mid S_{k-1} = s', u_k = i) \tag{4.33}$$

where $K = A_k D_k$ is a constant that is independent of the value of u_k.

Let

$$\gamma_{k,e}^i(s',s) = \exp\left\{L_c y_k^p c_k^p\right\} \cdot P(S_k = s \mid S_{k-1} = s', u_k = i)$$

Then we have

$$\gamma_k^i(s',s) \propto \exp\left\{iL_a(u_k) + iL_c y_k^s\right\} \cdot \gamma_{k,e}^i(s',s) \tag{4.34}$$

where "$\propto$" denotes equality up to a scale factor.

Combining (4.34) and (4.27) yields

$$\begin{aligned} L(u_k) &= \ln \frac{\sum_{\forall(s',s)\in B_k^1} \tilde{\alpha}_{k-1}(s') \cdot \gamma_{k,e}^1(s',s) \cdot \tilde{\beta}_k(s) \cdot \exp\{L_a(u_k) + L_c y_k^s\}}{\sum_{\forall(s',s)\in B_k^0} \tilde{\alpha}_{k-1}(s') \cdot \gamma_{k,e}^0(s',s) \cdot \tilde{\beta}_k(s)} \\ &= L_c y_k^s + L_a(u_k) + \ln \frac{\sum_{\forall(s',s)\in B_k^1} \tilde{\alpha}_{k-1}(s') \cdot \gamma_{k,e}^1(s',s) \cdot \tilde{\beta}_k(s)}{\sum_{\forall(s',s)\in B_k^0} \tilde{\alpha}_{k-1}(s') \cdot \gamma_{k,e}^0(s',s) \cdot \tilde{\beta}_k(s)} \\ &= L_c y_k^s + L_a(u_k) + L_e(u_k) \end{aligned} \tag{4.35}$$

The first term in the R.H.S. of (4.35) is the weighted channel measurement (which is sometimes called the channel value), the second term represents any *a priori* information about u_k provided by a previous decoder，and the third term represents extrinsic information that can be passed on to a subsequent decoder. Notice that only extrinsic information contains new information produced by the current decoder. 从 $\gamma_{k,e}^i(s',s)$ 的定义可以看出，外信息只包含当前分量编码器校验位的软输出信息，也只有这一部分信息对其它分量码译码器来说是不具有的新信息，因此可被用作先验信息。

由(4.35)我们可以得到 Turbo 译码器的另一种实现结构，如图 4.14 所示。

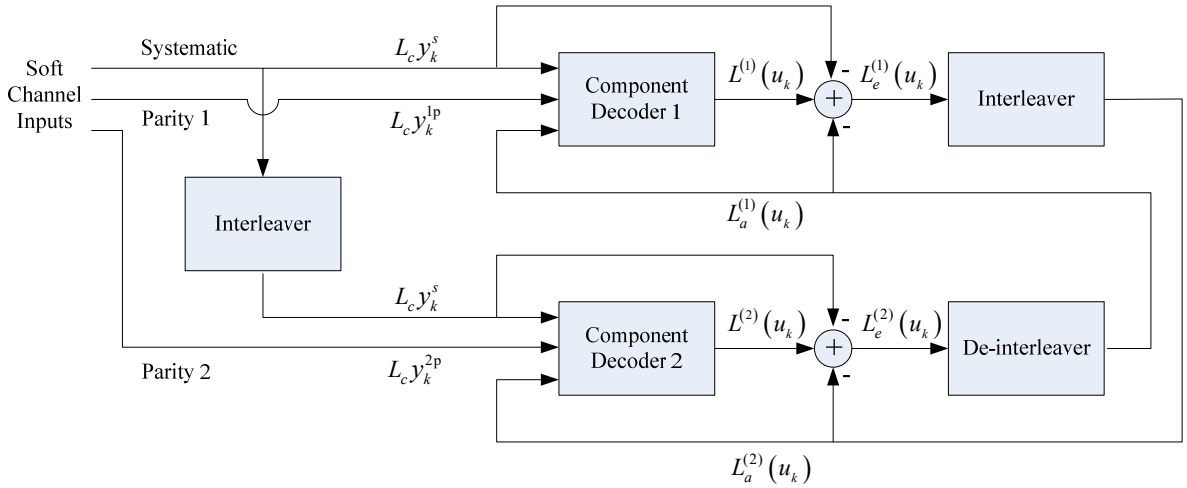


Figure 4.14 Turbo decoder schematic

对于图 4.14 所示的 turbo 译码器，如果分量码译码器 dec1 和 dec2 均采用上述 MAP 译码算法，则从式(4.35)可知它们在第 i 次迭代的软输出分别为：

$$\text{dec1:}\quad \left[L^{(1)}(u_k)\right]^{(i)} = L_c y_k^s + \left[L_e^{(2)}(u_{\pi^{-1}(k)})\right]^{(i-1)} + \left[L_e^{(1)}(u_k)\right]^{(i)}$$

$$\text{dec2:}\quad \left[L^{(2)}(u_k)\right]^{(i)} = L_c y_k^s + \left[L_e^{(1)}(u_{\pi(k)})\right]^{(i)} + \left[L_e^{(2)}(u_k)\right]^{(i)}$$

其中$\left[L_e^{(2)}(u_k)\right]^{(i-1)}$是前一次迭代中 dec2 给出的关于信息符号 u_k 的外信息，在本次迭代中被 dec1 用作先验信息，i.e, $\left[L_a^{(1)}(u_{\pi(k)})\right]^{(i)} = \left[L_e^{(2)}(u_k)\right]^{(i-1)}$；$\left[L_e^{(1)}(u_k)\right]^{(i)}$是 dec1 新产生的外信息，即式(4.35)中的第三项；$\left[L_e^{(1)}(u_k)\right]^{(i)}$经交织后作为从 dec1 到 dec2 的外信息，在 dec2 中被用作先验信息，i.e, $\left[L_a^{(2)}(u_k)\right]^{(i)} = \left[L_e^{(1)}(u_{\pi(k)})\right]^{(i)}$。整个迭代中软信息的转移过程为:

$$dec1 \rightarrow dec2 \rightarrow dec1 \rightarrow dec2 \rightarrow \cdots$$

Clearly, extrinsic information is used to update the *a priori* probabilities. Turbo codes get their name because the decoder uses feedback, like a ***turbo engine***.

■ A summary: **The Turbo Principle**

1) Each decoder produces the extrinsic information for each data bit u.

$$L_{extrisic}(u) = L_{output_APP}(u) - L_{input}(u)$$

L is called the logarithm of the APP ratio.

2) The extrinsic information is used as *a priori* information by the other decoder.

3) Decoding continues for a set number of iterations.

The Key Point here is what is passed by one decoder to the other is the result of its work about estimation of u, but not taking its own input into account.

■ MAP algorithm for a non-systematic encoder

For a non-systematic encoder, by letting

$$\tilde{\gamma}_k^i(s',s) = \exp\left\{L_c y_k^s c_k^s + L_c y_k^p c_k^p\right\} \cdot P(S_k = s \mid S_{k-1} = s', u_k = i)$$

(so $\gamma_k^i(s',s) = K \exp\left\{iL_a(u_k)\right\} \cdot \tilde{\gamma}_k^i(s',s)$)

we can obtain the log APP ratio as follows.

$$L(u_k) = L_a(u_k) + \ln \frac{\sum_{\forall (s',s) \in B_k^1} \tilde{\alpha}_{k-1}(s') \cdot \tilde{\gamma}_k^1(s',s) \cdot \tilde{\beta}_k(s)}{\sum_{\forall (s',s) \in B_k^0} \tilde{\alpha}_{k-1}(s') \cdot \tilde{\gamma}_k^0(s',s) \cdot \tilde{\beta}_k(s)} = L_a(u_k) + L_e(u_k)$$

4.3.4 Generalizations of the APP Algorithm

We now extend the above algorithm to the decoding of a rate-*k*/*n* convolutional code whose trellis diagram may exist parallel branches.

■ Extrinsic information for encoder output symbols

Let $B_k(c)$ be the set of transitions on which the output symbol c can occur; i.e., $B_k(c) = \left\{b_k = (u_k, c_k, S_k, S_{k-1}) \in B_k : c_k = c\right\}$. For the rate-1/2 convolutional encoder, denoting the coded output symbol by $c_k = (c_k^{(1)}, c_k^{(2)})$, the APP of the output coded bits can be calculated as

$$P(c_k^{(1)} = j \mid \mathbf{y}_1^N) = \sum_{\forall b_k \in B_k : c_k^{(1)} = j} P(u_k = i, S_{k-1} = s', S_k = s \mid \mathbf{y}_1^N)$$

Therefore, we have

$$L(c_k^{(1)}) = \ln \frac{\sum_{\forall b_k \in B_k : c_k^{(1)} = 1} \tilde{\alpha}_{k-1}(s') \cdot \gamma_k^i(s',s) \cdot \tilde{\beta}_k(s)}{\sum_{\forall b_k \in B_k : c_k^{(1)} = 0} \tilde{\alpha}_{k-1}(s') \cdot \gamma_k^i(s',s) \cdot \tilde{\beta}_k(s)}$$

$$= L_c y_k^{(1)} + \ln \frac{\sum_{\forall b_k \in B_k : c_k^{(1)} = 1} \tilde{\alpha}_{k-1}(s') \cdot \breve{\gamma}_k^i(s',s) \cdot \tilde{\beta}_k(s)}{\sum_{\forall b_k \in B_k : c_k^{(1)} = 0} \tilde{\alpha}_{k-1}(s') \cdot \breve{\gamma}_k^i(s',s) \cdot \tilde{\beta}_k(s)}$$

where $\breve{\gamma}_k^i(s',s) = \exp\left\{iL_a(u_k) + L_c y_k^{(2)} c_k^{(2)}\right\} \cdot P(S_k = s \mid S_{k-1} = s', u_k = i)$.

The extrinsic information for $c_k^{(1)}$ is given by

$$L_e(c_k^{(1)}) = L(c_k^{(1)}) - L_c y_k^{(1)} - L_a(u_k)$$

APP and extrinsic information for the code bit $c_k^{(2)}$ can be obtained in a similar way.

■ General APP decoder

If extrinsic information regarding both the encoder input and output is available, the branch transition metric becomes

$$\gamma_k^i(s',s) = K \exp\left\{ iL_a(u_k) + c_k^{(1)} L_a(c_k^{(1)}) + c_k^{(2)} L_a(c_k^{(2)}) + L_c y_k^{(1)} c_k^{(1)} + L_c y_k^{(2)} c_k^{(2)} \right\}$$

$$\cdot P(S_k = s \mid S_{k-1} = s', u_k = i)$$

Then, the APP and extrinsic information can be calculated in a way as aforementioned.

A general APP detector module is shown in Figure 4.15. The inputs to the decoder include the received sequence **y**, priori information regarding the encoder input $\mathbf{L}_a(\mathbf{u})$, and priori information regarding the encoder output $\mathbf{L}_a(\mathbf{c})$. The outputs are $L(u_k)$ and $L(c_k)$, the log APP ratios corresponding to the encoder input and output, respectively.

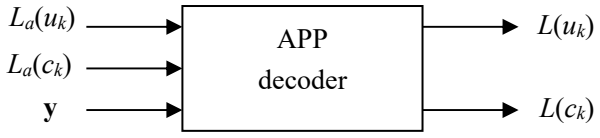


Fig. 4.15

4.3.5 An Alternative View on Extrinsic Information

In the original paper on turbo codes, the extrinsic information is interpreted as the output of a Gaussian meta-channel. In the decoding procedure, the extrinsic information is used as a Gaussian-distributed input to the decoder: $\mathbf{y}_k = (y_k^s, y_k^p, L_{a,k})$. The branch metric is then calculated as

$$\gamma_k^i(s',s) = P(u_k = i) \cdot P(S_k = s \mid S_{k-1} = s', u_k = i) \cdot p(\mathbf{y}_k \mid S_k = s, S_{k-1} = s', u_k = i)$$

$$= \frac{1}{2} P(S_k = s \mid S_{k-1} = s', u_k = i) \cdot p(y_k^s \mid c_k^s) p(y_k^p \mid c_k^p) p(L_{a,k} \mid u_k)$$

The interleaver allows the component decoders to generate separate estimates of APP's based on approximately uncorrelated data sources.

In general, the first approach makes a better use of the extrinsic information.

4.3.6 Decoding Algorithms in the Log Domain

Computations involved in the above APP algorithms will become simpler if the decoder operates in the log domain. Let us begin the discussion with the following definition.

$$E(x,y) \equiv \ln\left(e^x + e^y\right)$$

$$= \ln\left[e^x \left(1 + e^{y-x}\right) \right]$$

$$= x + \ln\left(1 + e^{y-x}\right)$$

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

$$= \max(x, y) + f_c\left(|x - y|\right) \tag{4.36}$$

where $f_c\left(|x-y|\right) = \ln\left(1 + e^{-|x-y|}\right)$ is a correction function. In practice, it can be realized using a lookup table.

For a set of real numbers $\{\delta_1, \delta_2, ..., \delta_q\}$, by noting that $\delta = e^{\ln\delta}$, we have

$$\begin{aligned}
\ln\left(\sum_{i=1}^{q} e^{\delta_i}\right) &= \ln\left(e^{\delta_1} + e^{\delta_2} + \cdots + e^{\delta_{q-2}} + e^{\delta_{q-1}} + e^{\delta_q}\right) \\
&= \ln\left(e^{\delta_1} + e^{\delta_2} + \cdots + e^{\delta_{q-2}} + e^{\ln\left(e^{\delta_{q-1}} + e^{\delta_q}\right)}\right) \\
&= \ln\left(e^{\delta_1} + e^{\delta_2} + \cdots + e^{\delta_{q-2}} + e^{E\left(\delta_{q-1}, \delta_q\right)}\right) \\
&= \ln\left(e^{\delta_1} + e^{\delta_2} + \cdots + e^{\ln\left[e^{\delta_{q-2}} + e^{E\left(\delta_{q-1}, \delta_q\right)}\right]}\right) \\
&= E\left(\delta_1, E\left(\delta_2, \cdots, E\left(\delta_{q-2}, E(\delta_{q-1}, \delta_q)\right)\cdots\right)\right)
\end{aligned} \tag{4.37}$$

This shows that $\ln\left(e^{\delta_1} + e^{\delta_2} + \cdots + e^{\delta_i}\right)$ can be computed recursively. Suppose that $\ln\left(e^{\delta_1} + e^{\delta_2} + \cdots + e^{\delta_{i-1}}\right)$ has been computed. Let

$$\Delta = \ln\left(e^{\delta_1} + e^{\delta_2} + \cdots + e^{\delta_{i-1}}\right)$$

Then

$$\begin{aligned}
\ln\left(e^{\delta_1} + e^{\delta_2} + \cdots + e^{\delta_i}\right) &= \ln\left(e^{\Delta} + e^{\delta_q}\right) \\
&= \max\left\{\Delta, \delta_q\right\} + f_c\left(|\Delta - \delta_q|\right)
\end{aligned} \tag{4.38}$$

4.3.6.1 Max-Log-MAP Decoding Algorithm

It is a suboptimal version of the aforementioned MAP algorithm, which is devised to provide an efficient trade-off between error performance and decoding computational complexity.

In the Max-Log-MAP algorithm, we use the following Max-Log approximation:

$$\ln\left(\sum_{i=1}^{q} e^{\delta_i}\right) \cong \max_{1 \le i \le q}\left\{\delta_i\right\} \tag{4.39}$$

With this approximation, we have the following derivation. For (4.24), we have

$$\begin{aligned}
A_k(s) &\triangleq \ln \tilde{\alpha}_k(s) \\
&= \ln\left(\sum_{s'} \tilde{\alpha}_{k-1}(s')\gamma_k(s', s)\right) - \ln\left(\sum_{s}\sum_{s'} \tilde{\alpha}_{k-1}(s')\gamma_k(s', s)\right)
\end{aligned}$$

$$= \ln \alpha_k^{'}(s) - \ln\left(\sum_s \alpha_k^{'}(s) \right)$$

Here

$$\ln \alpha_k^{'}(s) = \ln\left(\sum_{s'} \tilde{\alpha}_{k-1}(s')\gamma_k(s',s) \right)$$

$$= \ln\left(\sum_{s'} e^{\ln \tilde{\alpha}_{k-1}(s') + \ln \gamma_k(s',s)} \right)$$

$$= \ln\left(\sum_{s'} e^{A_{k-1}(s') + \Gamma_k(s',s)} \right)$$

$$\cong \max_{s'} \left\{ A_{k-1}(s') + \Gamma_k(s',s) \right\}$$

and

$$\ln\left(\sum_s \alpha_k^{'}(s) \right) = \ln\left(\sum_s e^{\ln \alpha_k^{'}(s)} \right) \cong \max_s \left\{ \ln \alpha_k^{'}(s) \right\}$$

Hence, the forward recursion of (4.24) can be expressed in the log domain as

$$A_k(s) \cong \max_{s'} \left\{ A_{k-1}(s') + \Gamma_k(s',s) \right\} - \max_s \left\{ \max_{s'} \left\{ A_{k-1}(s') + \Gamma_k(s',s) \right\} \right\}, \text{ or}$$

$$\ln \tilde{\alpha}_k(s) \cong \max_{s'} \left\{ \ln \tilde{\alpha}_{k-1}(s) + \ln \gamma_k(s',s) \right\} - \max_s \left\{ \max_{s'} \left\{ \ln \tilde{\alpha}_{k-1}(s') + \ln \gamma_k(s',s) \right\} \right\}$$

Similarly, the backward state metric in (4.25) can be expressed as

$$\ln \tilde{\beta}_{k-1}(s') \cong \max_s \left\{ \ln \tilde{\beta}_k(s) + \ln \gamma_k(s',s) \right\} - \max_s \left\{ \max_{s'} \left\{ \ln \tilde{\alpha}_{k-1}(s') + \ln \gamma_k(s',s) \right\} \right\},$$

or equivalently,

$$B_{k-1}(s') \cong \max_s \left\{ B_k(s) + \Gamma_k(s',s) \right\} - \max_s \left\{ \max_{s'} \left\{ A_{k-1}(s') + \Gamma_k(s',s) \right\} \right\}$$

Clearly, the forward and backward state metrics can be computed recursively with initial conditions:

$$A_0(0) = 0,\ A_0(s) = -\infty \text{ for } \forall s \neq 0$$

$$B_N(0) = 0,\ B_N(s) = -\infty \text{ for } \forall s \neq 0$$

Analogously, the final LLR calculation in (4.27) is approximated by

$$L(u_k) = \ln\left(\frac{\sum_{\forall (s',s) \in B_k^1} e^{\ln \tilde{\alpha}_{k-1}(s') + \ln \gamma_k^1(s',s) + \ln \tilde{\beta}_k(s)}}{\sum_{\forall (s',s) \in B_k^0} e^{\ln \tilde{\alpha}_{k-1}(s') + \ln \gamma_k^0(s',s) + \ln \tilde{\beta}_k(s)}} \right)$$

$$= \ln\left(\sum_{\forall (s',s) \in B_k^1} e^{\ln \tilde{\alpha}_{k-1}(s') + \ln \gamma_k^1(s',s) + \ln \tilde{\beta}_k(s)} \right) - \ln\left(\sum_{\forall (s',s) \in B_k^0} e^{\ln \tilde{\alpha}_{k-1}(s') + \ln \gamma_k^0(s',s) + \ln \tilde{\beta}_k(s)} \right)$$

$$\cong \max_{\forall(s',s)\in B_k^1}\left\{\ln\tilde{\alpha}_{k-1}(s')+\ln\gamma_k^1(s',s)+\ln\tilde{\beta}_k(s)\right\}-\max_{\forall(s',s)\in B_k^0}\left\{\ln\tilde{\alpha}_{k-1}(s')+\ln\gamma_k^0(s',s)+\ln\tilde{\beta}_k(s)\right\}$$

$$=\max_{\forall(s',s)\in B_k^1}\left\{A_{k-1}(s')+\Gamma_k^1(s',s)+B_k(s)\right\}-\max_{\forall(s',s)\in B_k^0}\left\{A_{k-1}(s')+\Gamma_k^0(s',s)+B_k(s)\right\} \quad (4.40)$$

which can be implemented by executing add-compare-select-subtract (ACSS) process.

From the discussion above, it can be shown that the Max-Log-MAP algorithm is equivalent to the bidirectional Viterbi algorithm (VA).

4.3.6.2 Log-MAP Decoding Algorithm

The error performance of Max-Log-MAP is very closed to that of MAP algorithm. But it produces soft-output values inferior to that of the MAP algorithm, resulting in the degraded performance. To overcome this problem, the Max-Log approximation can be corrected by adding a term called *correction function*, as shown in (4.36) and (4.38).

4.3.7 Turbo 码的迭代译码性能

根据上述 MAP 译码算法，我们对 1/2 码率的 16 状态 turbo 码进行了计算机仿真，其结果如图 4.16、图 4.17 所示。仿真所用交织器大小分别为 420、1024、4096、16000 比特，两个分量码相同，生成多项式均为 G=(37，21)$_8$。另外，图 4.18 中给出了采用 Max-Log-MAP 算法(即 SOVA 算法)时的 turbo 码的性能曲线，从中我们可以看出标准 MAP 算法与其简化算法之间的性能差异。

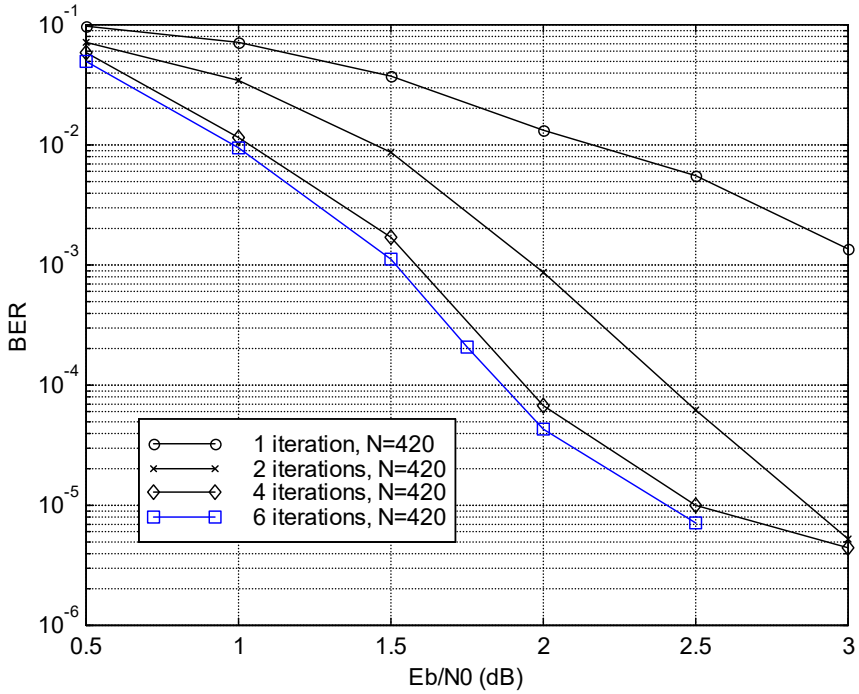


图 4.16　使用 BCJR 算法时 turbo 码在 AWGN 信道上的性能

（交织器的长度=420 比特）

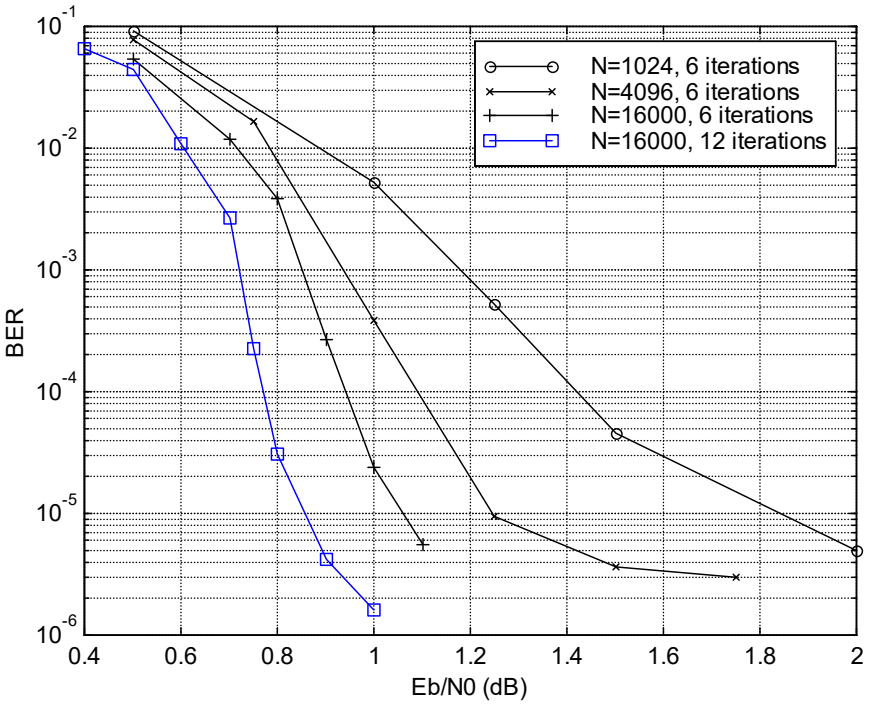


图 4.17　使用 BCJR 算法时，交织器长度对 turbo 码性能的影响

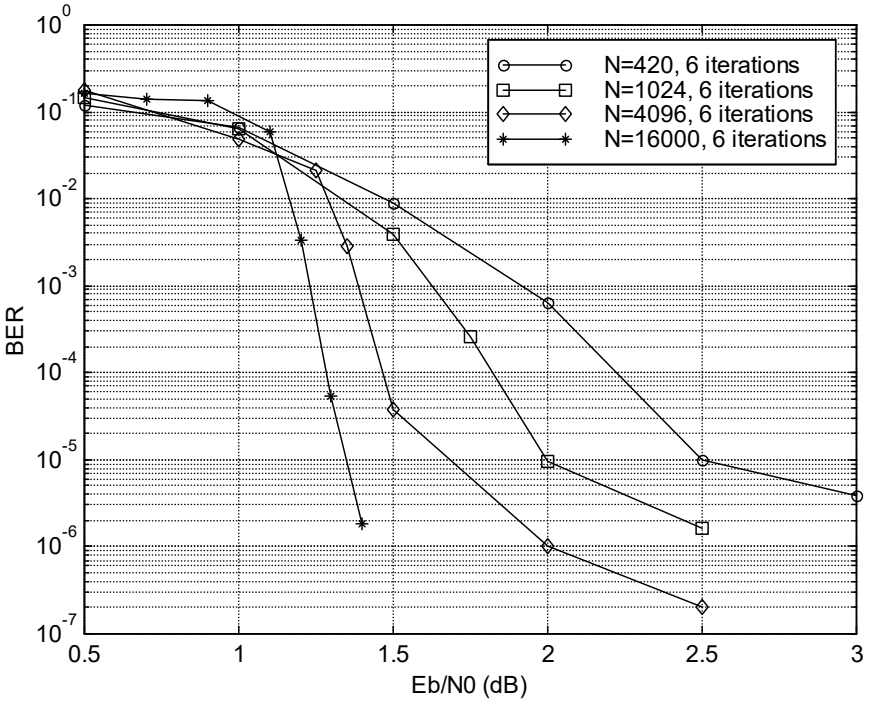
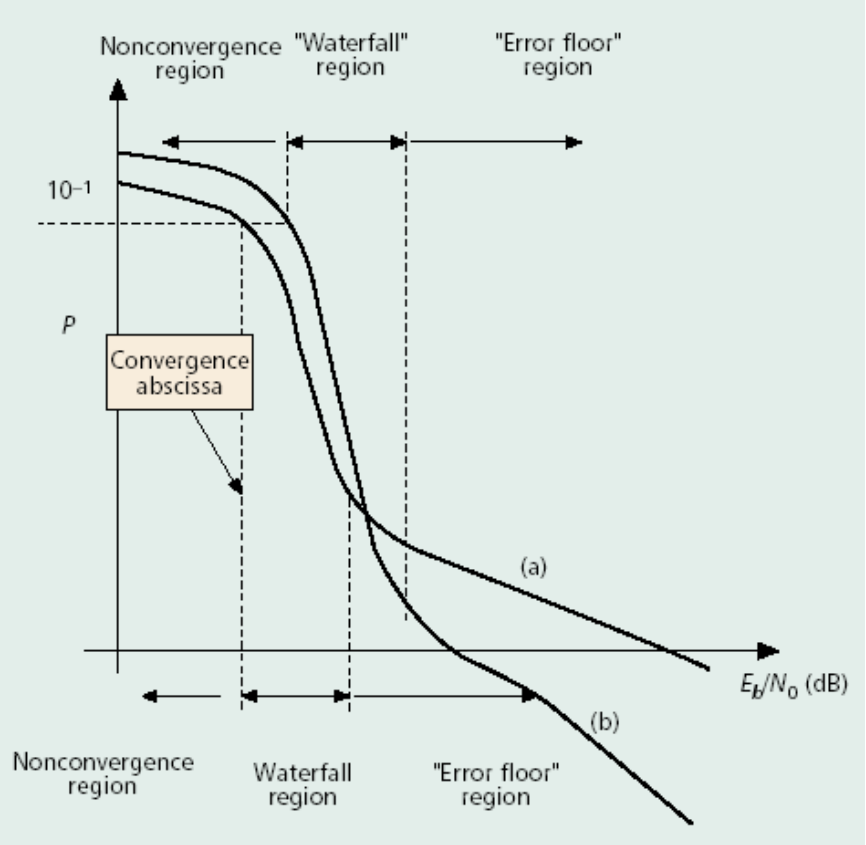


图 4.18　使用 Max-Log-MAP 算法时 turbo 码在 AWGN 信道上的性能

■ **Performance Features of Turbo-Like Codes**



■ *外信息的统计特征*

在 Turbo 码的迭代译码过程中，外信息 $\mathbf{L}_e$ 是一个随机变量序列，其统计特征随迭代次数的不同而变化。图 4.27 给出了（G1=37，G2=21）16 状态 Turbo 码的外信息 $L_e(u_k=1)$ 在不同迭代次数时的 Histogram. 这里 E_b/N_0=0.8dB. 我们可以看出随着迭代次数的增加，$L_e(u_k=1)$ → Gaussian-like。

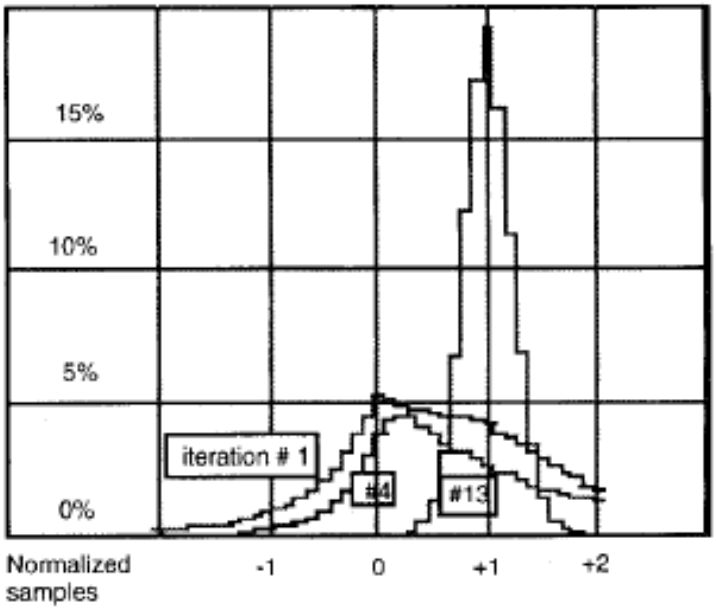


Fig. 11. Histograms of extrinsic information z_k after iterations #1, 4, 13 at $E_b/N_0 = 0.8$ dB; all information bits $d_k = 1$.

4.4 ML Performance Analysis of Binary Turbo Codes

This section is focused on the performance evaluation of binary turbo codes under maximum-likelihood (ML) decoding. This analysis allows us to predict the performance (especially in the high SNR regime) of a turbo code without resorting to computer simulations. Since the error performance of coded communication systems rarely admits exact expressions, tight analytical upper and lower bounds serve as a useful theoretical and engineering tool for assessing performance.

4.4.1 Classes of Existing Upper Bounds on the ML Decoding Error Probability

- *Union bound*
- *Duman and Salehi bound* (DS2)
  It is a modified version of Gallager bound.
- *Tangential-sphere bound*: It gives the tightest result.

The above bounds are compared in Fig. 4.20. In general, it is considered to be tight as long as it is close enough to the simulation results based on the iterative decoding, pre-assuming that the iterative decoding technique performs close to ML decoding for moderate values of E_b/N_0.

Notice that the bounds based on the union bounding technique give useless results for E_b/N_0 below the value corresponding to the cutoff rate R_0 of the channel.

Note:

For binary-input symmetric channels, $R_0 = 1 - \log_2\left[1 + \sum_y \sqrt{P(y|0)P(y|1)}\right]$

For the BPSK signaling over an AWGN channel,

$$R_0 = 1 - \log_2\left[1 + \int_{-\infty}^{\infty} \sqrt{\frac{1}{2\pi\sigma^2}\exp\left(-\frac{(y-\sqrt{E_s})^2 + (y+\sqrt{E_s})^2}{2\sigma^2}\right)}\, dy\right]$$

$$= 1 - \log_2\left[1 + \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{y^2 + E_s}{2\sigma^2}\right) dy\right]$$

$$= 1 - \log_2(1 + e^{-E_s/N_0})$$

Therefore, we have

$$\frac{E_b}{N_0} = -\frac{\ln(2^{1-R_0} - 1)}{R_0} \tag{4.41}$$

For R_c=1/3, $\frac{E_b}{N_0} = \begin{cases} -0.5\text{dB}, & \text{(correspondong to capacity)} \\ 2.03\text{dB}, & \text{(correspondong to cutoff rate)} \end{cases}$

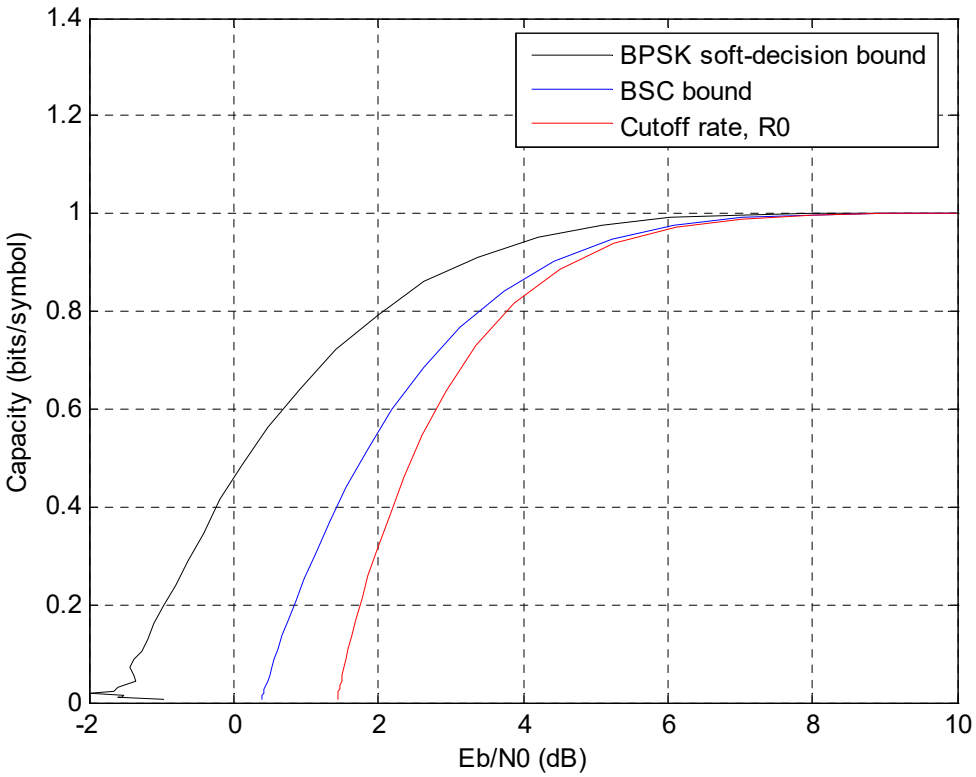


Figure 4.19 C, R_0 vs. Eb/N0

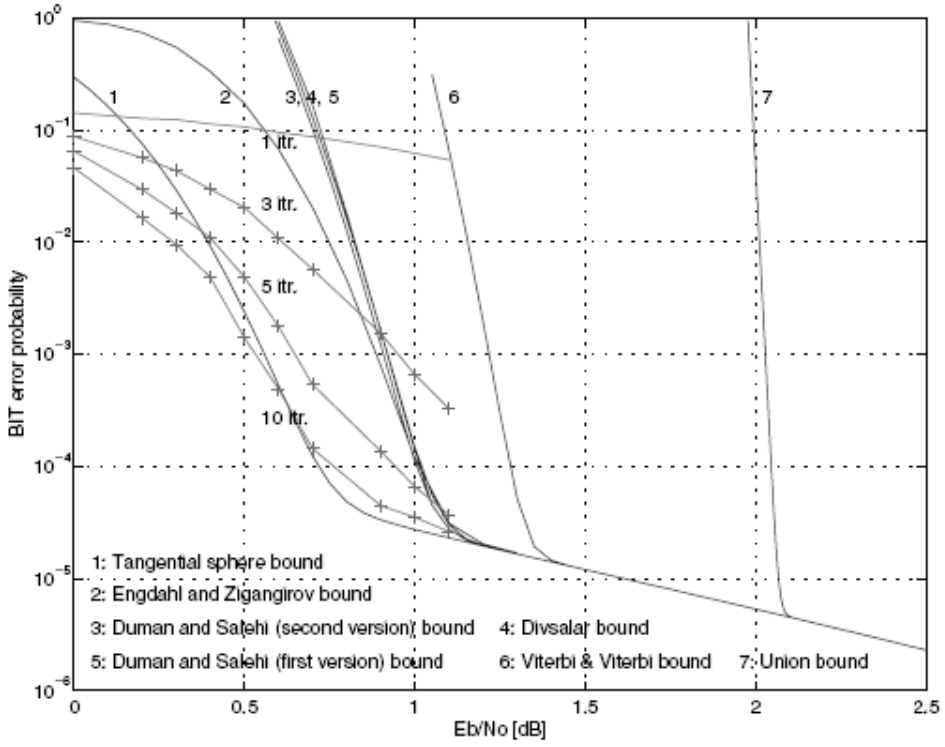


Fig. 4.20. Comparison of various upper bounds on the BER performance for a rate-1/3 turbo code. The interleaver size is 1000, and the generator for both component codes is $G(D)=\left[1 \quad \frac{1+D^4}{1+D+D^2+D^3+D^4}\right]$. Assuming BPSK signaling on the AWGN channel.

The upper bounds on the ensemble performance of turbo codes for the AWGN channel and ML decoding can be broadly classified into two categories:

1) The ensemble is generated by a uniform choice of the interleaver of length N and of component codes taken from the set of time-varying RSC codes of memory length m.
2) The ensemble of codes is generated by a uniform interleaver of length N and the component codes are fixed RSC codes.

In the following subsections, we will derive the performance bounds for the 2nd case; i.e., we assume that the component codes are fixed (block or convolutional) codes.

4.4.2 The Union Bound

Consider the use of a binary linear code $\mathcal{C}$ of length N over an AWGN channel. With BPSK modulation, the transmitted sequence corresponding to the ith binary-valued codeword $\mathbf{c}_i=[c_{i1}\ c_{i2}\cdots c_{in}]$ is given by $\mathbf{s}_i=[s_{i1}\ s_{i2}\cdots s_{in}],\ i=1,2,\cdots M$, where

$$s_{ik}=\begin{cases}-\sqrt{E_s}, & \text{if } c_{ik}=1\\ \sqrt{E_s}, & \text{if } c_{ik}=0\end{cases}$$

From Chapter 2, the probability that an ML detector selects $\mathbf{s}_j$ instead of $\mathbf{s}_i$ given that $\mathbf{s}_i$ is the transmitted vector is equal to

$$P_2(\mathbf{s}_i \to \mathbf{s}_j) = Q\left(\sqrt{\frac{d_{ij}^2}{2N_0}}\right) \tag{4.42}$$

where

$$d_{ij}^2 = |\mathbf{s}_i - \mathbf{s}_j|^2$$

If $\mathbf{s}_i$ and $\mathbf{s}_j$ differ in d coordinates, i.e., $d_H(\mathbf{c}_i, \mathbf{c}_j) = d$, then

$$d_{ij}^2 = \sum_{k=1}^{n} (s_{ik} - s_{jk})^2 = d\left(2\sqrt{E_s}\right)^2 = 4dE_s \tag{4.43}$$

Substituting (4.43) into (4.42) yields

$$P_2(\mathbf{s}_i \to \mathbf{s}_j) = P_2(d) = Q\left(\sqrt{\frac{2dE_s}{N_0}}\right) \tag{4.44}$$

This is the so-called *pairwise error probability* (PEP) between the two codewords with Hamming distance d. A useful property is that

$$P_2(d) \le \frac{1}{2} e^{-dE_s/N_0}$$

Applying the union bound, the probability of decoding error given that $\mathbf{c}$ is the transmitted codeword is given by

$$P(e \mid \mathbf{c}) \le \sum_{c' \ne c} P_2(\mathbf{c} \to \mathbf{c}')$$

$$= \sum_{d=d_{\min}}^{N} A_{d|c} P_2(d)$$

where $A_{d|c}$ is the number of codewords in $\mathcal{C}$ with Hamming distance d from $\mathbf{c}$, $d_{\min}$ is the minimum Hamming distance of code $\mathcal{C}$. For a convolutional code, $d_{\min} = d_f$. Hence, the average probability of error over all codewords is upper-bounded by

$$P(e) = \sum_{c} P(\mathbf{c}) P(e \mid \mathbf{c})$$

$$\le \sum_{d=d_{\min}}^{N} A_d P_2(d) \tag{4.45}$$

where $A_d = \sum_{\mathbf{c}} P(\mathbf{c}) A_{d|\mathbf{c}}$ is the average number of codewords $\mathbf{c}' \ne \mathbf{c}$ at distance d from $\mathbf{c}$. Due to the linearity, A_d is also the number of codewords with Hamming weight d.

It is seen from (4.45) that, in order to obtain the performance bound of turbo codes, we need to calculate the pairwise error probability under the considered channel model and the weight spectrum of turbo codes.

4.4.2.1 *The Distance Spectrum of Parallel Concatenated Codes (PCC)*

For simplicity, let us start with the PCBC schemes (where the block codes are used as constituent codes). With trellis termination, a convolutional code is equivalent to a block code.

■ **Weight Enumerators for Component Codes**

For a (N, K) linear systematic block code $\mathcal{C}$, its *weight enumerating function* (WEF) has the form

$$A(X) = \sum_{d=0}^{N} A_d X^d \tag{4.46}$$

where A_d is the number of codewords with Hamming weight d.

The WEF describes the code weight distribution that only depends on the codewords. It does not provide the contributions of the input information bits to the total Hamming codeword weight. A more detailed weight profile of the code is described by the *input output weight enumerating function* (IOWEF) defined as

$$A(W, X) = \sum_{w=0}^{K} \sum_{d=0}^{N} A_{w,d} W^w X^d$$

where $A_{w,d}$ is the number of codewords with Hamming weight d generated by information words of weight w. For systematic encoders, we can separate codeword weights into information and parity block weights and define the *input-redundancy weight enumerating function* (IRWEF) of the code as

$$A(W, Z) = \sum_{w=0}^{K} \sum_{z=0}^{N-K} A_{w,z} W^w Z^z \tag{4.47}$$

where $A_{w,z}$ is the number of codewords generated by information words of Hamming weight w whose parity check bits have Hamming weight z. It follows immediately that

$$A_d = \sum_{w+z=d} A_{w,z} = \sum_{w=0}^{d} A_{w,d-w} \tag{4.48}$$

Example 1: Consider the (7, 4) Hamming code defined by the following generator matrix

$$\mathbf{G} = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.49}$$

It can be verified that the WEF and IRWEF for the (7, 4) Hamming code above are, respectively,

$$A(X) = 1 + 7X^3 + 7X^4 + X^7 \text{, and}$$

$$A(W, Z) = 1 + W(3Z^2 + Z^3) + W^2(3Z + 3Z^2) + W^3(1 + 3Z) + W^4 Z^3$$

In performance analysis it is often useful to group the terms in the IRWEF according to the weights of information words. The following *conditional weight enumerating function* enumerates the parity block weights for codewords corresponding to the input words of weight *w*:

$$A_w(Z) = \sum_{z=0}^{N-K} A_{w,z} Z^z \tag{4.50}$$

Clearly, $A_w(Z)$ and IRWEF are related by

$$A(W, Z) = \sum_{w=0}^{K} W^w A_w(Z)$$

$$A_w(Z) = \frac{1}{w!} \cdot \frac{\partial^w A(W, Z)}{\partial W^w}\Big|_{W=0}$$

■ **Weight Enumerators and Performance Bounds**

Both $A_w(Z)$ and IRWEF can be used in conjunction with the union bound to compute an upper bound on the bit error probability for ML decoding of the code. Suppose that a length-*N* codeword **x** has a weight-*w* information block of length *K* and a weight-*z* parity block. The selection of **x** by the decoder, as opposed to the correct all-zero codeword, will cause *w* errors out of a total of *K* information bits. From (4.45), we have

$$P_b \le \sum_{w=1}^{K} \frac{w}{K} \sum_{z=0}^{N-K} A_{w,z} P_2(w + z) \tag{4.51}$$

Substituting $d = w + z$, (4.51) can be written as

$$P_b \le \sum_{d=d_{\min}}^{N} \sum_{w=1}^{d} \frac{w}{K} A_{w,d-w} P_2(d) = \sum_{d=d_{\min}}^{N} B_d P_2(d) \tag{4.52}$$

where

$$B_d = \sum_{w+z=d} \frac{w}{K} A_{w,z} \tag{4.53}$$

denotes the average number of bit errors caused by transitions between the all-zero codeword and codewords of weight d. The set of all pairs (d, B_d) is called the *code distance spectrum*. With $P_2(d) \le \frac{1}{2} e^{-dE_s/N_0}$, (4.51) becomes

$$P_b \le \frac{1}{2} \sum_{w=1}^{K} \frac{w}{K} e^{-wR_cE_b/N_0} \sum_{z=0}^{N-K} A_{w,z} e^{-zR_cE_b/N_0}$$

$$= \frac{1}{2} \sum_{w=1}^{K} \frac{w}{K} W^w A_w(Z)\Big|_{W=Z=e^{-R_cE_b/N_0}} \tag{4.54}$$

where R_c is the code rate. Using IOWEF, we can also express (4.52) (or (4.51)) as

$$P_b \le \sum_{w=1}^{K} \frac{w}{K} \sum_{d=d_{\min}}^{N} A_{w,d} Q\left(\sqrt{\frac{2dR_cE_b}{N_0}}\right) \tag{4.55}$$

With the property that $Q(\sqrt{x+y}) \le Q(\sqrt{y}) e^{-x/2}$, a tighter upper bound on bit error probability can be expressed as

$$P_b \le Q\left(\sqrt{\frac{2d_{\min}R_cE_b}{N_0}}\right) e^{d_{\min}R_cE_b/N_0} \sum_{d=d_{\min}}^{N} B_d e^{-dR_cE_b/N_0}$$

$$= Q\left(\sqrt{\frac{2d_{\min}R_cE_b}{N_0}}\right) e^{d_{\min}R_cE_b/N_0} \frac{W}{K} \frac{\partial A(W,Z)}{\partial W}\Big|_{W=Z=\exp(-R_cE_b/N_0)} \tag{4.56}$$

■ Information Weight Distribution of PCC

We now proceed to consider the weight spectrum of PCC. Let $A_w^{C_i}(Z)$, i=1, 2, denote the conditional weight enumerating function for the constituent codes C_i in the PCBC. We want now to obtain the conditional WEF $A_w^{C_P}(Z)$ of the PCBC. For a given interleaver, this operation is exceedingly complicated. To facilitate the analysis, we consider the ensemble performance, where the ensemble of codes is generated by a uniform interleaver of length N and the component codes are fixed block codes.

Definition [Benedetto]: *A uniform interleaver of length K is a probabilistic device which maps a given input word of weight w into all distinct* $\binom{K}{w}$ *permutations of it with equal*

probability $\dfrac{1}{\binom{K}{w}}$.

The uniform interleaver makes independent weight distributions of the parity check sequences generated by the first and second encoders.

With this definition, we can easily obtain the conditional WEF of the overall PCBC by

$$A_w^{C_P}(Z) = \frac{A_w^{C_1}(Z) \cdot A_w^{C_2}(Z)}{\binom{K}{w}} \tag{4.57}$$

For a parallel concatenated code $\mathcal{C}_p$ consisting of M constituent codes and M-1 uniform interleavers of length K, its conditional WEF is related to the conditional WEF of its constituent codes by

$$A_w^{C_P}(Z) = \frac{\prod_{m=1}^{M} A_w^{C_m}(Z)}{\binom{K}{w}^{M-1}} \tag{4.58}$$

As an example, Fig. 4.21 shows the distance spectrum of a turbo code of rate 1/3.

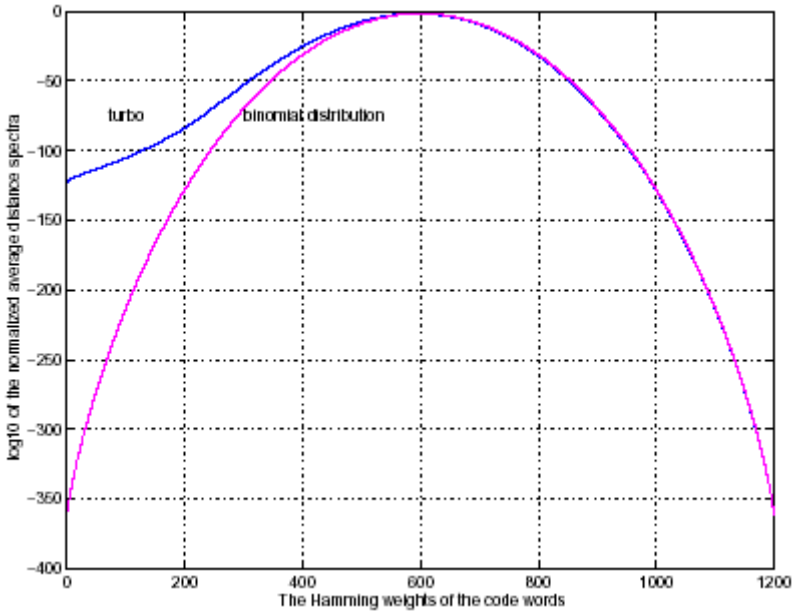


Fig. 4.21. The distance spectrum of a turbo code of rate 1/3. Info length K=400, N=1200, memory=5.

Remark: Unlike random binary block codes that have a binomial distance spectrum and are known to achieve capacity, based on the exponential deviation of the distance spectrum of

turbo codes from the binomial distribution at low Hamming weights, it is not obvious that there exists any decoding method (including the ML decoding), which achieves the Shannon capacity for turbo codes.

Turbo Code Performance Evaluation

Substituting (4.58) into (4.52), we obtain the *upper bound on the bit error probability of the PCC with ML decoding*

$$P_b \le \sum_{d=d_{\min}}^{N} B_d Q\left(\sqrt{\frac{2dRE_b}{N_0}}\right) \tag{4.59}$$

where $B_d = \sum_{w+z=d} \frac{w}{K} A_{w,z}^{C_p} = \sum_{w+z_1+z_2=d} \frac{w}{K} \frac{A_{w,z_1}^{C_1} \cdot A_{w,z_2}^{C_2}}{\binom{K}{w}}$.

Using (4.54), the bit error probability of the PCC is upper-bounded by

$$P_b \le \frac{1}{2} \sum_{w=1}^{K} \frac{w}{K} W^w A_w^{C_p}(Z)\Big|_{W=Z=e^{-R_c E_b/N_0}} \tag{4.60}$$

Notice that the bound above give useless results for E_b/N_0 below the value corresponding to the cutoff rate R_0 of the channel. Tighter bounds for a larger range of E_b/N_0 extending below the cutoff rate threshold have been derived by T. Duman and M. Salehi in [6] and by A. M. Viterbi and A. J. Viterbi in [] based on variants of Gallager's exponential bound.

4.4.3 *Improved Upper Bound – Generalized DS2 Bound*

For the derivation of the upper bound on the block error probability, the binary linear block code $\mathcal{C}$ of length N is partitioned into a set of subcodes $\{\mathcal{C}_d, 1 \le d \le N\}$, where every subcode $\mathcal{C}_d$ includes all the codewords which possess a constant Hamming-weight d (d=1,2,…,N) and also the all-zero codeword. Note that the subcode $\mathcal{C}_d$ are usually nonlinear codes. Let $P_{e|0}(d)$ denote the conditional block error probability associated with the ML decoding of the subcode $\mathcal{C}_d$ given that the all-zero codeword is transmitted. Due to the channel symmery and the linearity of the code, an overall union bound over the subcodes yields

$$P_B(e) = P_B(e \mid 0) \le \sum_{d=d_{\min}}^{N} P_{e|0}(d) \tag{4.61}$$

where $d_{\min}$ is the minimum Hamming distance of the code $\mathcal{C}$.

We now develop a tighter upper bound on the $P_{e|0}(d)$, which is first derived by Duman and Salehi based on the Gallager bound. We will call this bound the *DS2 bound*.

It is seen from Chapter 3 that the probability of block error with ML decoding is upper-bounded by the Gallager bound

$$P_B(e\,|\,m) \le \sum_{\mathbf{y}} P_N(\mathbf{y}\,|\,\mathbf{x}_m)\left[\sum_{m'\neq m}\left(\frac{P_N(\mathbf{y}\,|\,\mathbf{x}_{m'})}{P_N(\mathbf{y}\,|\,\mathbf{x}_m)}\right)^{\lambda}\right]^{\rho}, \qquad \lambda>0,\ \rho\ge 0 \tag{4.62}$$

Let $\psi_N^m(\mathbf{y})$ be an arbitrary probability measure of $\mathbf{y}$ (which may depend on the $\mathbf{x}_m$). Then from (4.62) we have

$$P_B(e\,|\,m) \le \sum_{\mathbf{y}} \psi_N^m(\mathbf{y})\left\{\psi_N^m(\mathbf{y})^{-\frac{1}{\rho}} P_N(\mathbf{y}\,|\,\mathbf{x}_m)^{\frac{1}{\rho}} \sum_{m'\neq m}\left(\frac{P_N(\mathbf{y}\,|\,\mathbf{x}_{m'})}{P_N(\mathbf{y}\,|\,\mathbf{x}_m)}\right)^{\lambda}\right\}^{\rho}, \qquad \lambda>0,\ \rho\ge 0 \tag{4.63}$$

By invoking the Jensen's inequality (the expectatin is taken with respect to $\psi_N^m(\mathbf{y})$), (4.63) becomes

$$P_B(e\,|\,m) \le \left\{\sum_{m'\neq m}\sum_{\mathbf{y}} P_N(\mathbf{y}\,|\,\mathbf{x}_m)^{\frac{1}{\rho}} \psi_N^m(\mathbf{y})^{1-\frac{1}{\rho}}\left(\frac{P_N(\mathbf{y}\,|\,\mathbf{x}_{m'})}{P_N(\mathbf{y}\,|\,\mathbf{x}_m)}\right)^{\lambda}\right\}^{\rho} \tag{4.64}$$

For the binary-input AWGN channels, we select $\psi_N(\mathbf{y}) = \prod_{n=1}^{N}\psi(y_n)$. Applying (4.64) to every subcode $\mathcal{C}_d$ yields

$$P_{e|0}(d) \le \left\{\sum_{\substack{\mathbf{x}\in\mathcal{C}_d\\ \mathbf{x}\neq\mathbf{0}}} \int_{-\infty}^{\infty}\cdots\int_{-\infty}^{\infty}\prod_{n=1}^{N}\psi(y_n)^{1-\frac{1}{\rho}} p_0(y_n)^{\frac{1-\lambda\rho}{\rho}} p_{x_n}(y_n)^{\lambda} d\mathbf{y}\right\}^{\rho} \tag{4.65}$$

where x_n equals to "1" or "0" for d and N-d indexes of n, respectively, and $p_{x_n}(y_n) = p_{Y|X}(Y=y_n\,|\,X=x_n)$ denotes the conditional probability density function (pdf) given by

$$p_0(y) = \frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{(y-\sqrt{E_s})^2}{2\sigma^2}\right),$$

$$p_1(y) = \frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{(y+\sqrt{E_s})^2}{2\sigma^2}\right).$$

The right-hand side (R.H.S.) of (4.65) can be written as

$$\left\{A_d\left[\int_{-\infty}^{\infty}\psi(y)^{1-\frac{1}{\rho}} p_0(y)^{\frac{1-\lambda\rho}{\rho}} p_1(y)^{\lambda} dy\right]^{d}\cdot\left[\int_{-\infty}^{\infty}\psi(y)^{1-\frac{1}{\rho}} p_0(y)^{\frac{1}{\rho}} dy\right]^{N-d}\right\}^{\rho} \tag{4.66}$$

Combining (4.65) and (4.66), we obtain the so-called *generalized DS2 upper bound* on the block error probability:

$$P_{e|0}(d) \le (A_d)^{\rho} \left\{ \left[\int_{-\infty}^{\infty} \psi(y)^{1-\frac{1}{\rho}} p_0(y)^{\frac{1}{\rho}} dy \right]^{(1-\delta)\rho} \cdot \left[\int_{-\infty}^{\infty} \psi(y)^{1-\frac{1}{\rho}} p_0(y)^{\frac{1-\lambda\rho}{\rho}} p_1(y)^{\lambda} dy \right]^{\delta\rho} \right\}^N \quad , \quad \lambda>0,$$

$0 \le \rho \le 1$ (4.67)

where $\delta = d/N$ is the *normalized Hamming weight*. Note that ψ is an arbitrary nonnegative function satisfying $\int_{-\infty}^{\infty} \psi(y) dy = 1$. It can be shown that the optimal function (which yields the tightest bound) is given by [5]

$$\psi(y) = \beta p_0(y) \left\{ 1 + \alpha \left[\frac{p_1(y)}{p_0(y)} \right]^{\lambda} \right\}^{\rho} \tag{4.68}$$

where α and β are non-negative numbers. For specific values of λ, ρ, and δ, the parameter α is optimally determined as to satisfy the implicit equation

$$1-\delta = \frac{\int_{-\infty}^{\infty} p_0(y) \left\{ 1 + \alpha \left[\frac{p_1(y)}{p_0(y)} \right]^{\lambda} \right\}^{\rho-1} dy}{\int_{-\infty}^{\infty} p_0(y) \left\{ 1 + \alpha \left[\frac{p_1(y)}{p_0(y)} \right]^{\lambda} \right\}^{\rho} dy}$$

The parameter β is then optimally determined by the following relation

$$\beta = \left\{ \int_{-\infty}^{\infty} p_0(y) \left[1 + \alpha \left(\frac{p_1(y)}{p_0(y)} \right)^{\lambda} \right]^{\rho} dy \right\}^{-1}$$

Notice that the minimization of the bound in (4.67) involves a numerical optimization over λ and ρ for every particular subcode $\mathcal{C}_d$. By applying Jensen inequality, $E[(A_d)^{\rho}] \le (E[A_d])^{\rho}$, we may obtain from (4.67) an upper bound on the block error probability for an ensemble of linear block codes.

Similarly, the upper bound on the bit error probability of a systematic block code $\mathcal{C}$ is given by

$$P_b \le \sum_{d=d_{\min}}^{N} P_b(d) \tag{4.69}$$

where $P_b(d)$ designates the conditional bit error probability of the subcode $\mathcal{C}_d$ with ML decoding when the all-zero codeword is transmitted. The $P_b(d)$ can be calculated as

$$P_b(d) \le (A'_d)^{\rho} \left\{ \left[\int_{-\infty}^{\infty} \psi(y)^{1-\frac{1}{\rho}} p_0(y)^{\frac{1}{\rho}} dy \right]^{(1-\delta)\rho} \cdot \left[\int_{-\infty}^{\infty} \psi(y)^{1-\frac{1}{\rho}} p_0(y)^{\frac{1-\lambda\rho}{\rho}} p_1(y)^{\lambda} dy \right]^{\delta\rho} \right\}^N, \quad \lambda>0,$$

$$0 \le \rho \le 1. \quad (4.70)$$

where $A'_d \equiv \sum_{w=1}^{K} \frac{w}{K} A_{w,d}$, and $A_{w,d}$ are the coefficients of the IOWEF of the subcode $\mathcal{C}_d$.

Let $r_d = \frac{\ln(A'_d)}{N}$. We can express (4.70) in the exponential form as

$$P_b(d) \le e^{-NE(r_d)} \quad (4.71)$$

where

$$E(r_d) = \max_{\lambda>0, 0\le\rho\le1} \{-\rho r_d - (1-\delta)\rho \ln g(\rho) - \delta\rho \ln h(\rho,\lambda)\} \quad (4.72)$$

The function $g(\rho)$ and $h(\rho, \lambda)$ are defined, respectively, to be

$$g(\rho) = \int_{-\infty}^{\infty} \psi(y)^{1-\frac{1}{\rho}} p_0(y)^{\frac{1}{\rho}} dy \quad \text{and}$$

$$h(\rho,\lambda) = \int_{-\infty}^{\infty} \psi(y)^{1-\frac{1}{\rho}} p_0(y)^{\frac{1-\lambda\rho}{\rho}} p_1(y)^{\lambda} dy$$

4.5 Convergence Analysis of Iterative Decoding via EXIT Charts

As we have seen, the ML decoding performance of turbo codes can be analyzed using the distance spectrum and the minimum distance of turbo codes, which can explain the error floor behavior, and to some extent the threshold behavior of these codes. A great challenge is the analysis of the iterative process which has been attempted via the so-called density evolution analysis and which was successful for low-density parity check codes.

The behavior of turbo codes at the onset of the turbo cliff is better understood by a statistical analysis, called the **EX***trinsic* ***I****nformation* ***T****ransfer* (EXIT) *chart*, first presented by ten Brink [8]. The EXIT analysis is related to similar methods which are based on variances [?]. The basic philosophy of both methods is to view the component decoders of the turbo decoder as a statistical processor that transforms an input value, i.e., the a priori (or extrinsic) LLR of the information symbols, into an output value, the recomputed extrinsic LLR. The component decoder is therefore seen as a nonlinear LLR transformer, as indicated in Figure 4.22.

EXIT chart估计的是Turbo code ensemble的平均译码门限，而且假定交织器的长度充分大，以便the a priori values remain fairly uncorrelated from the respective channel observations over many iterations.

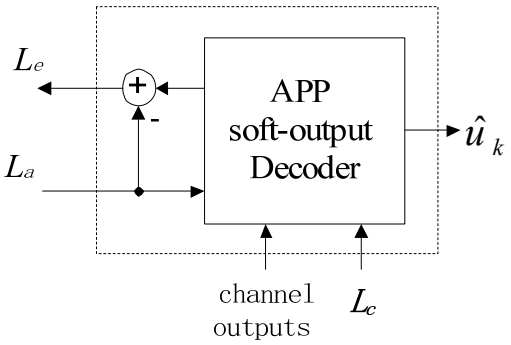


Fig. 4.22 Component decoder seen as extrinsic LLR transformer. L_a and L_e are the input and the output extrinsic LLRs, respectively.

4.5.1 Transfer Characteristics of Constituent Decoders

Simulations show that the extrinsic output LLR values, L_e, (or, *a priori* values, L_a, for the next decoder) has an approximate Gaussian distribution with increasing number of iterations. Thus the *a priori* input L_a to the constituent decoder can be modeled by applying an independent Gaussian random variable n_A with zero mean and variance σ_A^2 in conjunction with the known transmitted systematic bits x, i.e.,

$$L_E = \mu_E x + n_E, \quad n_E \sim \mathcal{N}(0, \sigma_E^2), \text{ or} \tag{4.73}$$

$$L_A = \mu_A x + n_A, \quad n_A \sim \mathcal{N}(0, \sigma_A^2) \tag{4.74}$$

where $x \in \{+1, -1\}$ is the transmitted systematic bit, and μ_E and σ_E^2 are the mean and variance of L_E, respectively. They are related by

$$\mu_E = \frac{\sigma_E^2}{2} \tag{4.75}$$

Note: Under the assumption of an AWGN channel model: $z = x + n$, the LLR of z can be expressed as

$$L_z = \ln \frac{p(z \mid x = +1)}{p(z \mid x = -1)} = \frac{2z}{\sigma_n^2} = \frac{2}{\sigma_n^2}(x + n),$$

which is clearly conditionally Gaussian. We may rewrite it as

$$L_z = \mu_z x + n_z, \text{ with } \mu_z = \frac{2}{\sigma_n^2} \text{ and } n_z = \frac{2}{\sigma_n^2} n \sim \mathcal{N}\left(0, \frac{4}{\sigma_n^2}\right)$$

Denote the variance of n_z by σ_z^2. Then we have $\sigma_z^2 = \frac{4}{\sigma_n^2}$ and $\mu_z = \frac{\sigma_z^2}{2}$.

Thus, under the assumption that $x = +1$ is sent, $L_z \sim \mathcal{N}(\mu_z, 2\mu_z)$. A Gaussian density $\mathcal{N}(\mu, 2\mu)$ is called a consistent Gaussian density.

It can be shown that the following properties are true for all L above:

1) They have a symmetric p.d.f. $p(L \mid x=-1) = p(-L \mid x=+1)$;

2) They satisfy the *consistency condition* $p(-L \mid x) = e^{-x \cdot L} p(L \mid x)$.

With (4.74), the conditional p.d.f. of L_A can be expressed as

$$p(L_A \mid x) = \frac{1}{\sqrt{2\pi}\sigma_A} \exp\left\{ -\frac{\left(L_A - (\sigma_A^2/2)x\right)^2}{2\sigma_A^2} \right\}$$

To measure the reliability of L_A, the mutual information $I_A = I(X; L_A)$ between X and L_A is used, which has proven to be the most accurate and convenient measure.

In the case of Gaussian modelled input LLR values, the mutual information measure, in bits, can be calculated as

$$\begin{aligned}
I_A &= \sum_x P(x) \int_{-\infty}^{\infty} p(L_A \mid x) \log_2 \frac{p(L_A \mid x)}{\sum_{x'} P(x') p(L_A \mid x')} dL_A \\
&= \sum_{x \in \{\pm 1\}} \frac{1}{2} \int_{-\infty}^{\infty} p(L_A \mid X = x) \log_2 \frac{2\, p(L_A \mid X = x)}{p(L_A \mid X = +1) + p(L_A \mid X = -1)} dL_A \\
&= 1 - \frac{1}{2} \sum_{x \in \{\pm 1\}} \int_{-\infty}^{\infty} p(L_A \mid X = x) \log_2 \frac{p(L_A \mid X = +1) + p(L_A \mid X = -1)}{p(L_A \mid X = x)} dL_A \\
&= 1 - \int_{-\infty}^{\infty} p(L_A \mid X = +1) \log_2 \left(1 + \frac{p(L_A \mid X = -1)}{p(L_A \mid X = +1)} \right) dL_A \\
&= 1 - \frac{1}{\sqrt{2\pi}\sigma_A} \int_{-\infty}^{\infty} \exp\left(-\frac{(L_A - \mu_A)^2}{2\sigma_A^2} \right) \log_2 \left(1 + \exp(-L_A) \right) dL_A
\end{aligned} \tag{4.76}$$

where the last line follows from the consistency condition and because $p(L \mid X=-1) = p(-L \mid X=+1)$ for Gaussian densities.

The mutual information, $I_E = I(X; L_E)$, between the output extrinsic LLR and the information bit x is more complex to evaluate since L_E is not exactly Gaussian. The values of L_E corresponding to L_A have to be found via simulating the component decoder. Other than in very simple cases, no closed form or even analytical formulas for the output extrinsic mutual information exist to date. The output extrinsic information I_E is therefore an empirical function of the component decoder, the input I_A, and the operating signal-to-noise ratio, formally given by the EXIT function

$$I_E = T\left(I_A, E_b / N_0\right)$$

To evaluate this function the following steps are executed.

1) A Gaussian modelled input LLR with mutual information value I_A is generated, fed into the decoder, and the output extrinsic LLRs L_E are captured.
2) A numerical evaluation of mutual information between the information symbols x and the captured extrinsic output LLRs L_E yields I_E, calculated as

$$I_E = \frac{1}{2}\sum_{x\in\{\pm 1\}}\int_{-\infty}^{\infty} p(L_E \mid X = x)\log_2\left(\frac{2p(L_E \mid X = x)}{p(L_E \mid X = +1) + p(L_E \mid X = -1)}\right)dL_E$$

where $p(L_E \mid X)$ is the empirical distribution of L_E as measured at the output of the APP decoder. (It can be obtained from histogram measurements.) Clearly, the disadvantage of this method lies in the fact that the component decoder needs to be simulated, and that we are working with empirical distributions.

■ A simpler method

Using Monte Carlo simulation, (4.76) can be calculated simply by

$$I_A = 1 - \mathsf{E}\left\{\log_2\left(1 + e^{-L_A}\right)\right\}$$

where the expectation is over the distribution $P(L_A|x=+1)$. With the symmetry of communication link in $x = \pm 1$ and the assumption of Gaussian-like distribution, I_E can be obtained in a similar fashion. See Fig. 4.23.

$$I_E = 1 - \mathsf{E}\left\{\log_2\left(1 + e^{-L_E}\right)\right\} \approx 1 - \frac{1}{N}\sum_{n=1}^{N}\log_2\left(1 + e^{-x_n L_{E,n}}\right)$$

where $x_n \in \{\pm 1\}$ and $L_{E,n}$ are samples in the experiment. (上式指数中乘以x_n用来correct L_E generated by the input x=-1)

The overall assumption for the EXIT chart is a large interleaver to assure statistical independence.

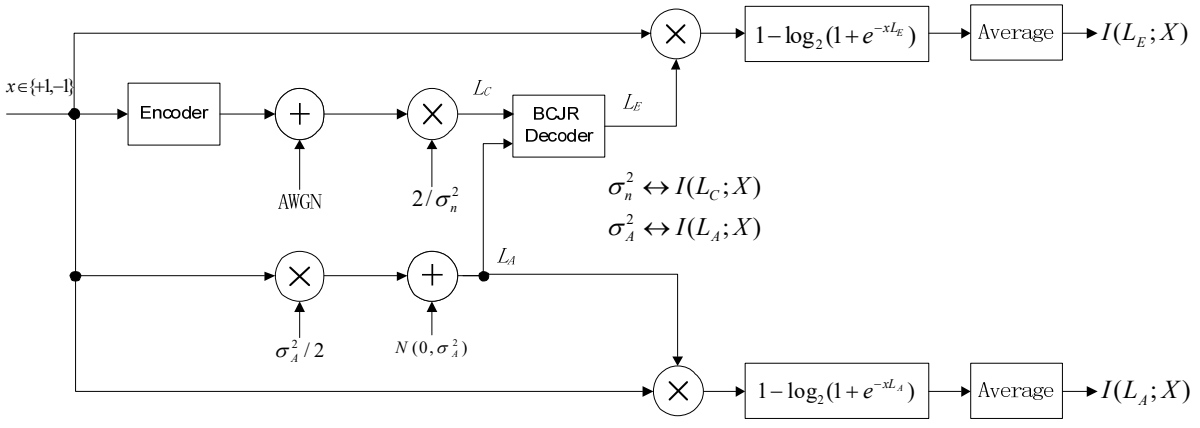


Figure 4.23: Measurement of the mutual information $I(X; L_E)$

Figure 4.24 shows the EXIT function $T\left(I_A, E_b/N_0\right)$ for a memory 4, rate R_c = 1/2 component convolutional code with

$$G(D) = \left[1,\ \frac{1 + D + D^3 + D^4}{1 + D^3 + D^4}\right]$$

The numbers on the different trajectories are the E_b/N_0 values in dB.

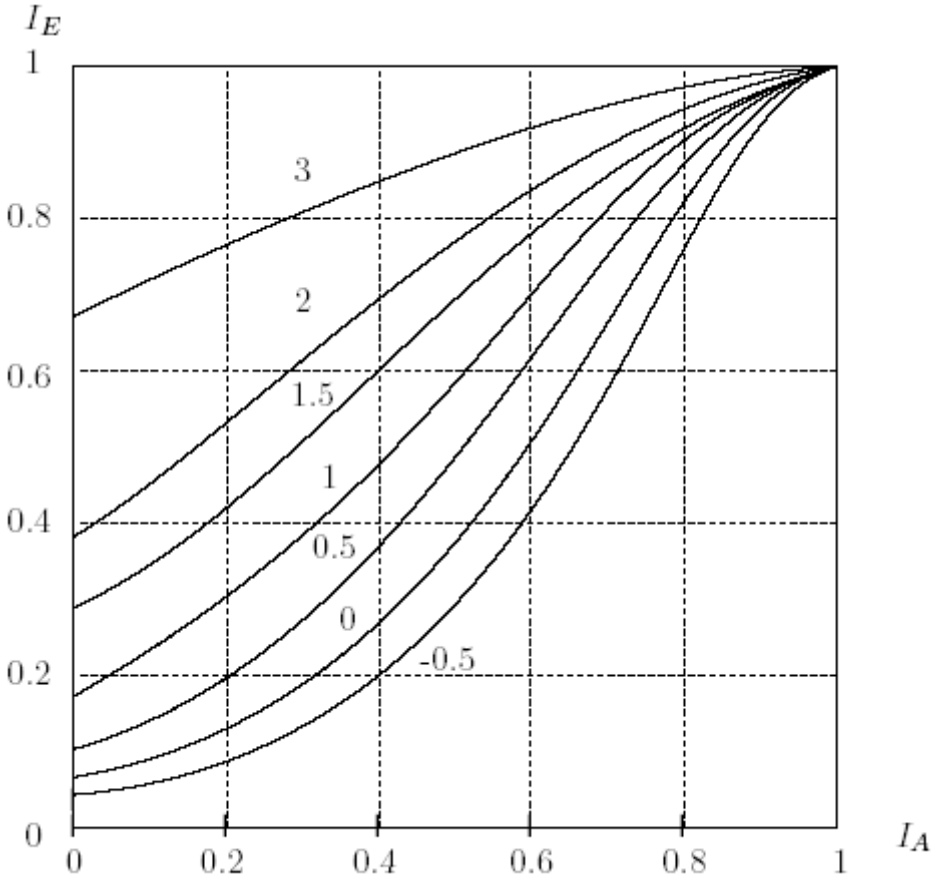


Figure 4.24: EXIT function of a 16-state convolutional code.

4.5.2 EXIT Chart

In a complete turbo decoder this extrinsic information is now exchanged between the decoders, where output extrinsic LLRs become *a pripori* LLRs for the next decoder. In our analysis these iterations are captured by a sequence of applications of component decoder EXIT functions, that is

$$0 \xrightarrow{T_1} I_E^{(1)} = I_A^{(2)} \xrightarrow{T_2} I_E^{(2)} = I_A^{(1)} \xrightarrow{T_1} I_E^{(1)} = I_A^{(2)} \xrightarrow{T_2} \cdots$$

where T_1 is the EXIT function of decoder DEC1, and T_2 of decoder DEC2. The behavior of this exchange can be visualized in the EXIT chart, where the EXIT functions of both component decoders are plotted, however, the transfer curve of decoder #2 is reflected about the $\pi/4$ line to reflect that I_E is the input mutual information, and I_A is the output mutual information. As shown in Figure 4.25, for sufficiently large values of E_b/N_0 (in this case for E_b/N_0 = 0.8dB) these two curves leave open a channel, and they have no intersect point other than (1, 1). Since the iterations start with zero LLRs, i.e., with zero mutual information at

point (0, 0) in the chart, the iterations progress by bouncing from one curve to the other as illustrated in Figure 4.25.

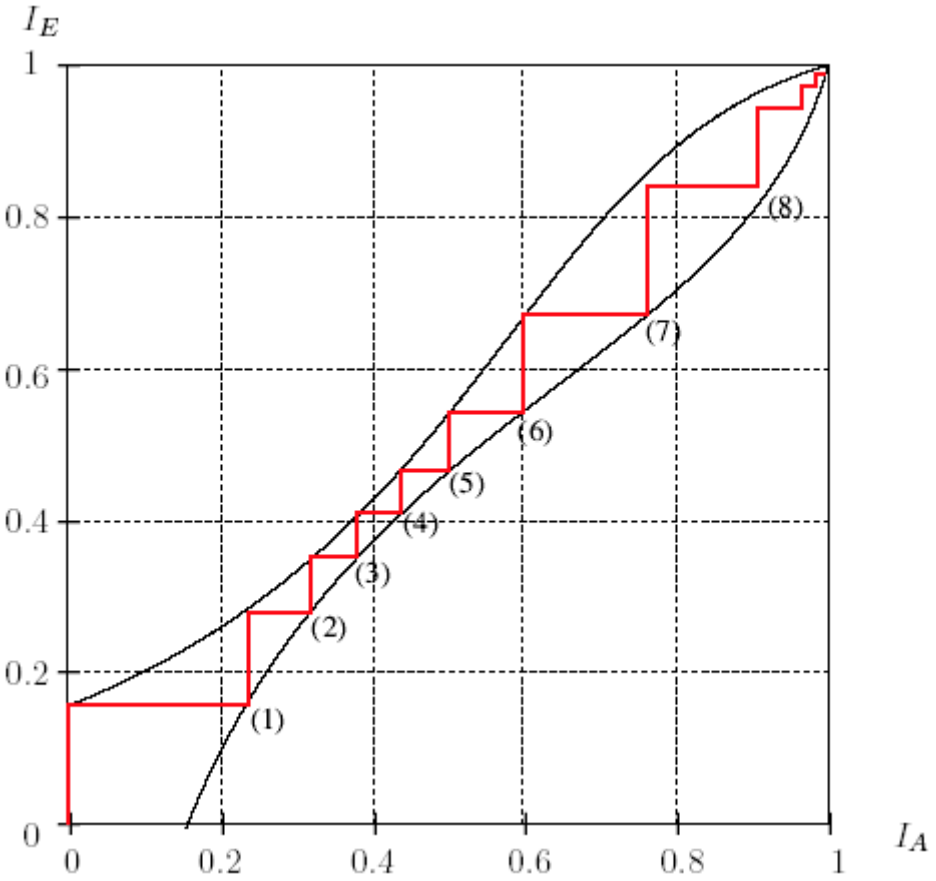


Figure 4.25: EXIT chart combining the EXIT functions of the two decoders involved in a turbo decoder for E_b/N_0 = 0.8dB.

The trajectory plotted in Figure 4.25 is a measured characteristic of a single decoding cycle of the turbo decoder, and the accuracy with respect to the prediction of the EXIT chart is impressive, testifying to the robustness of the method.

4.5.3 Application Examples of the Exit Chart

As an example of using the EXIT chart to predict the onset of the turbo cliff behavior, we reduce the signal-to-noise ratio E_b/N_0 for the codes discussed until the open channel closes at E_b/N_0 = 0.53dB, which represents the *pinch-off* signal-to-noise ratio of this, the original turbo code combination of decoders. Fig. 4.26 shows the dramatic turbo cliff effect through simulations.

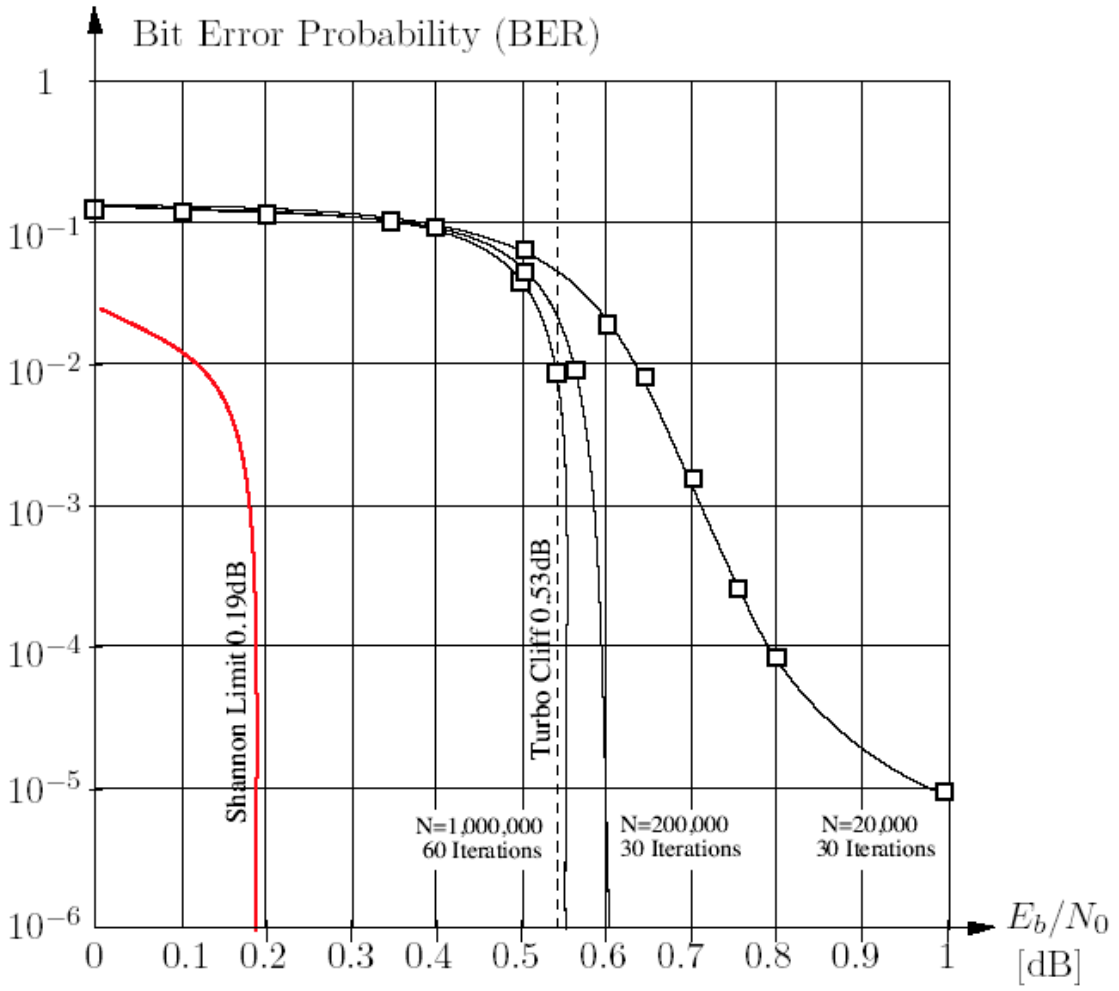


Figure 4.26. Simulations of the turbo code using 16-state component decoders and different sized interleavers related to theoretical prediction of the turbo cliff.

In [8], ten Brink presented the following table of pinch-off limits for PCCC of rare-1/2 using the listed constituent codes: (注意：基于高斯分布假设得到的译码门限只是一个近似值)

ν	C_1/C_2	(3,2)	(7,5)	(13,15)	(23,37)	(67,45)	(147,117)
1	(3,2)	>2dB					
2	(7,5)	1.49dB	0.69dB				
3	(13,15)	1.14dB	0.62dB	0.62dB			
4	(23,37)	1.08dB	0.65dB	0.64dB	0.68dB		
5	(67,45)	0.86dB	0.62dB	0.66dB	0.70dB	0.77dB	
6	(147,117)	0.84dB	0.63dB	0.67dB	0.72dB	0.81dB	0.84dB

Appendix

Calculation of Weight Enumerators for Convolutional Codes

In this appendix, we describe a precise and efficient method to evaluate transfer functions for convolutional codes. The method is called the transfer matrix method and is applied to obtain weight enumerators. The material presented in this section is modified from the [McEliece98].

Consider the (2, 1, 3) convolutional encoder shown in Fig. 4.A1. The state transition diagram is depicted in Fig. 4.A2.

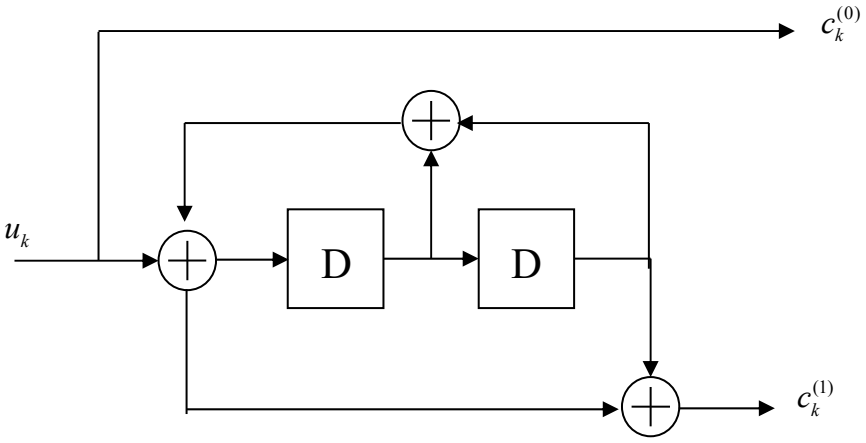


Fig. 4.A1

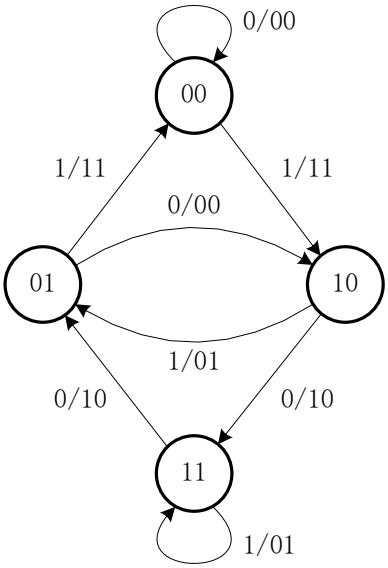


Fig. 4.A2

In Fig. 4.A1, each transition between states is labeled by the input information bit and the corresponding output coded bit. It is convenient to replace each edge label with a monomial $x^i y^j z^N$, where i is the input weight, j is the output weight, and N is the length of the path (which is always equal to 1 in this case). Then the information in the state diagram can be summarized by the state transition matrix

$$A(x,y) = \begin{array}{c} \\ 00 \\ 10 \\ 01 \\ 11 \end{array} \begin{array}{c} \begin{array}{cccc} 00 & 10 & 01 & 11 \end{array} \\ \begin{bmatrix} 1 & xy^2 & 0 & 0 \\ 0 & 0 & xy & y \\ xy^2 & 1 & 0 & 0 \\ 0 & 0 & y & xy \end{bmatrix} \end{array}$$

Denote by $w_{N,i,j}$ the number of paths of length N, input weight i and output weight j, starting and ending in state-0, with the proviso that the last m edges in the path are "termination" edges and have neither length nor input weight. The corresponding transfer function (generating function) is defined by

$$W(x,y,z) = \sum_{N,i,j} w_{N,i,j} x^i y^j z^N \tag{A1}$$

Definition 4.A1: Let V_0 be the finite vertex set at time 0, and V_N the finite vertex set at time N. If $u \in V_0$ and $v \in V_N$, we define the flow from u to v, denoted by $\phi_T(u,v)$, to be the sum of the weights of all paths in trellis T from u to v.

Theorem 4.A1: The flow $\phi_T(u,v)$ is equal to the (u, v)th entry of the product matrix

$$A = A_1 A_2 \cdots A_N$$

It can be shown that $W(x, y, z)$ is the $(0^m, 0^m)$ entry in the matrix

$$\left(\mathbf{I} + \mathbf{A}(x,y,z) + \mathbf{A}(x,y,z)^2 + \cdots\right)\mathbf{A}(1,y,1)^m \tag{A2}$$

The factor $\mathbf{A}(1,y,1)^m$ takes care of the termination edges. Since $\mathbf{I} + \mathbf{A} + \mathbf{A}^2 + \cdots = (\mathbf{I} - \mathbf{A})^{-1}$, it follows from (A2) that

$$W(x,y,z) = \left[\left(\mathbf{I} - \mathbf{A}(x,y)\right)^{-1} \cdot \mathbf{A}(1,y)^m\right]_{0^m,0^m} \tag{A3}$$

To find the xy-weight enumerators, we rewrite (A1) as

$$W(x,y,z) = \sum_{N \geq 0} w_N(x,y) z^N \tag{A4}$$

By omitting the termination factor $\mathbf{A}(1, y, 1)^m$ in (A3), we have

$$W(x, y, z) = 1 - \frac{1}{\left[\left(\mathbf{I} - z\mathbf{A}(x, y)\right)^{-1}\right]_{0,0}} \tag{A5}$$

For the encoder of Fig. 4.A1, by using *mathematica*, we find that

$$W(x, y, z) = \frac{x^3 y^5 z^3 + (x^2 y^6 - x^4 y^6) z^4}{1 - xyz - xyz^2 + (-y^2 + x^2 y^2) z^3} \tag{A6}$$

If we multiply both sides of Eq. (6) by the denominator of the right-hand side, and take the coefficient of *w(x, y)* of both sides of the resulting equation, we obtain the following recursion determining $w(x, y)$ for the code of Fig. 4.A1, for $N >= 0$:

$$W_N(x, y) = xyW_{N-1}(x, y) + xyW_{N-2}(x, y) + (y^2 - x^2 y^2) W_{N-3}(x, y)$$

We can obtain the first few terms of the expansion of $W(x, y, z)$ in powers of z using long division:

$$W(x, y, z) = x^3 y^5 z^3 + x^2 y^6 z^4 + (x^4 y^6 + x^3 y^7) z^5 + (\text{terms of order } z^6 \text{ and higher})$$

Historical Notes

(from IEEE Spectrum, 2004)

It happened a decade ago at the 1993 IEEE International Conference on Communications in Geneva, Switzerland. Two French electrical engineers, Claude Berrou and Alain Glavieux, made a flabbergasting claim: they had invented a digital coding scheme that could provide virtually error-free communications at data rates and transmitting-power efficiencies well beyond what most experts thought possible.

The scheme, the authors claimed, could double data throughput for a given transmitting power or, alternatively, achieve a specified communications data rate with half the transmitting energy—a tremendous gain that would be worth a fortune to communications companies.

Few veteran communications engineers believed the results. The Frenchmen, both professors in the electronics department at the Ecole Nationale Supérieure des Télécommunications de Bretagne in Brest, France, were then unknown in the information-theory community. They must have gone astray in their calculations, some reasoned. The claims were so preposterous that many experts didn't even bother to read the paper.

Unbelievable as it seemed, it soon proved true, as other researchers began to replicate the results. Coding experts then realized the significance of that work. Berrou and Glavieux were right, and their error-correction coding scheme, which has since been dubbed turbo codes, has revolutionized error-correction coding.

At the heart of turbo coding is this iterative process, in which each component decoder takes advantage of the work of the other at a previous decoding step. After a certain number of iterations, typically four to 10, both decoders begin to agree on all bits. That means the decoders are not lost anymore in a universe of codewords; they have overcome the complexity barrier.

"It's a divide-and-conquer solution," says Robert J. McEliece, a professor of electrical engineering at the California Institute of Technology, in Pasadena, and an IEEE Fellow. "It broke the problem into two smaller pieces, solved the pieces, and then put the pieces back together."

What led Berrou and Glavieux to their important breakthrough was not some esoteric theorem but the struggle to solve real-world problems in telecommunications. In the late 1980s, when they began to work on coding schemes, they were surprised that an important concept in electronics—feedback—was not used in digital receivers.

In amplifiers, a sample of the output signal is routinely fed back to the input to ensure stable performance. Berrou and Glavieux wondered, why shouldn't it work for coding as well?

The first thing Berrou and Glavieux did after confirming that their results were correct was to patent the invention in France, Europe, and the United States. At the time, France Télécom was the major sponsor of their work, so the French company took possession of the turbo code patents. It was France Télécom that asked Berrou to come up with a commercial name for the invention. He found the name when one day, watching a car race on TV, he noticed that the newly invented code used the output of the decoders to improve the decoding process, much as a turbocharger uses its exhaust to force air into the engine and boost combustion. Voilà: "turbo codes"!

It was the insight and naiveté typical of outsiders that helped Berrou and Glavieux realize what the coding theory community was missing. "Turbo codes are the result of an empirical, painstaking construction of a global coding/decoding scheme, using existing bricks that had never been put together in this way before," they wrote a few years ago.

Berrou says their work is proof that it is not always necessary to know about theoretical limits to be able to reach them. "To recall a famous joke, at least in France," he says, "the simpleton didn't know the task was impossible, so he did it."

中译本（科技日报：Turbo 码启示录）：

Turbo 码结束了一场持续了 40 年的探求。这是划时代的，因为这是革命性的进展。今天如果你不能接近香农极限就会问：“出了什么毛病？”任何人都能接近香农极限，但今天我们谈的是你的编码比别的编码快多少以及你离香农极限差 0.1 分贝还是 0.001 分贝。

是直觉和质朴帮助伯劳和格莱维欧克斯认识到被编码理论界忽视了的东西。几年以前他们写道：“Turbo 码是经验、苦干的结果，是从全局考虑构建的编解码方案，所使用的各种技术元素都是已有的，只是以前从未以这种方式整合而已。”

伯劳指出他们的工作证明并非总是需要知道理论的极限才能达到这些极限。这使人们想起一个法国著名的笑话：傻瓜不知道这件事是没法做到的，因此他就做到了。

References

[1] C. Berrou and A.Glavieux, “Near optimum error correcting coding and decoding: Turbo-codes,” *IEEE Trans. Commun.*, vol. 44, no.10, pp.1261-1271, Oct. 1996.

[2] J. Hagenauer, E. Offer and L. Papke, “Iterative decoding of binary block and convolutional codes,” *IEEE Trans. Inform. Theory*, vol.42, no.2, pp.429-445, Mar. 1996.

[3] S. Benedetto and G. Montorsi, “Unveiling turbo codes: Some results on parallel concatenated coding schemes,” *IEEE Trans. Inform. Theory*, vol.42, no.2, pp.409-428, Mar. 1996.

[4] S. Shamai and I. Sason, “Variations on the Gallager bounds, connections and applications,” *IEEE Trans. Inform. Theory*, vol.48, no.12, pp.3029, Dec. 2002.

[5] I. Sason and S. Shamai, “On improved bounds on the decoding error probability of block codes over interleaved fading channels, with applications to turbo-like codes,” *IEEE Trans. Inform. Theory*, vol.47, no.6, pp.2275-2299, Sep. 2001.

[6] T. M. Duman and M. Salehi, “New performance bounds for turbo codes,” *IEEE Trans. Commun.*, vol.46, no.6, pp.717, June 1998.

[7] C. B. Schlegel and L. C. Perez, *Trellis and Turbo Coding*. IEEE Press, NJ, 2004.

[8] S. ten Brink, “Convergence behavior of iteratively decoded parallel concatenated codes”, *IEEE Trans. Commun.*, vol.49, no.10, pp.1727-1737, Oct. 2001.

[9] *Proceedings of The IEEE*, Special issue on Turbo-information processing: Algorithms, implementations and applications, vol.95, no.6, June 2007.

[10] W. E. Ryan and Shu Lin, *Channel Codes: Classical and Modern*. Cambridge University Press, 2009.