文章编号:1004-5694(2001)04-0026-04

Turbo 码译码中的 BCJR 算法

朱联祥，李元彬，周围
(重庆邮电学院·重庆 400065)

摘　要:BCJR 算法是在 Turbo 码的译码中广泛使用的一种重要算法。对 BCJR 算法进行了详细的推导，并简要讨论了其在 Turbo 码译码中的一些实现问题。实践及理论研究证明，BCJR 算法对于 Turbo 码译码性能的提高具有相当重要的意义。

关键词:Turbo 码;BCJR 算法;回归系统卷积码
中图分类号:TN911.22　　**文献标识码**:A

The BCJR Algorithm in Turbo Codes Decoding

ZHU Lian-xiang, LI Yuan-bin, ZHOU Wei
(*Communications and Information Institute, CUPT, Chongqing 400065, China*)

Abstract: The BCJR algorithm is an important and popular algorithm used in turbo codes decoding. In this paper, a thorough derivation of the algorithm is given and also some implementation aspects of it in turbo codes decoding are discussed. The theory and practice prove that BCJR algorithm plays an important role in improving the performance of Turbo code decoding.

Key words: turbo codes; BCJR algorithm; recursive systematic convolutional code

0　引　言

自从 1993 年 C. Berrou 等人提出 Turbo 码的概念以来，Turbo 码就以其接近 Shannon 限的差错控制性能而受到了人们的普遍关注。Turbo 码之所以能够取得如此优异的性能，主要得益于其独特的编译码结构：通过两个或者多个成员码通过交织器的并行级联，从而可以由简单的短码构造出长的"随机码"，接近 Shannon 信道编码的随机性假设；使用软输入软输出(SISO)的译码算法，充分利用了解调器的软输出信息，在性能上较之于硬判决有很大改善；采用类似于涡轮发动机的迭代译码结构，通过简单成员译码器译码输出信息的重复利用实现近似最优的序列译码，降低了译码实现的复杂性。

在 Turbo 码中，通常使用回归系统卷积码(RSC)作为其成员码。由于 Turbo 码所采用的独特迭代译码结构，因此 RSC 码的软输入软输出译码成为影响其性能及系统实现的一个关键因素。BCJR 算法是一种计算经过加性白高斯噪声(AWGN)信道传输的卷积码编码序列后验概率的最优算法。本文在对该算法进行详细推导的基础上，讨论了其在 Turbo 码译码中的一些实现问题。

1　回归系统卷积码(RSC 码)

设 Turbo 码的成员码为一码率 $R=1/2$ 的回归系统卷积码，在图 1 中，给出了一个生成多项式为 $(1,7/5)_8$ 的 RSC 编码器的一个示例。编码器的输入序列记为 $u=(u_1,u_2,\cdots,u_N)$，由 RSC 编码器产生

收稿日期:2001-06-05
作者简介:朱联祥(1971-)，男，陕西户县人，讲师，重庆大学博士生，研究方向为 CDMA 扩频通信、信道编码。

的校验序列为 $c_1^N=(c_1,c_2,\cdots,c_N)$，其中 $u_n,c_n\in\{0,1\}$。在采用二元双极性调制的情况下，u_n 及 c_n 被转

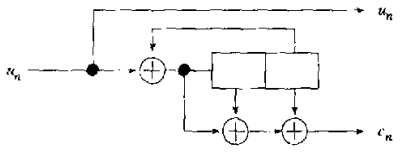

图1 生成多项式为$(1,7/5)_8$的RSC编码器

Fig. 1 An $(1,7/5)_8$ RSC encoder

换成取值为±1的符号后经 AWGN 信道传输，接收到的系统及校验序列分别为 $x_1^N=(x_1,x_2,\cdots,x_N)$，$y_1^N=(y_1,y_2,\cdots,y_N)$，其中：

$$x_n=(2u_n-1)+w_{x,n} \tag{1}$$

$$y_n=(2c_n-1)+w_{y,n} \tag{2}$$

其中，$w_{x,n}$和 $w_{y,n}$为独立同分布的零均值高斯随机变量，方差为 σ^2。

送入译码器的除了接收到的码序列 x_1^N 和 y_1^N 之外，还有每一传输比特的先验信息 λ_n。该先验信息表示为"1"码和"0"码出现的先验概率比值的对数，即：

$$\lambda_n=\ln\frac{P_r(u_n=1)}{P_r(u_n=0)} \tag{3}$$

这样，译码器的完整输入信息可表示为 R_1^N，并有：

$$R_1^N=(x_1^N,y_1^N,\lambda_1^N)$$

2 BCJR 算法

译码器的主要目标是根据接收到的码序列及先验信息计算信息位 u_n 分别为 1 和 0 的后验概率，即计算 $P_r(u_n=1|R_1^N)$和 $P_r(u_n=0|R_1^N)$。这一概率可以通过对网格图中的状态转移概率求和而得到。图 2

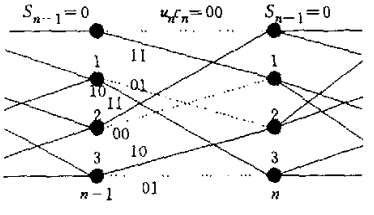

图2 生成多项式为$(1,7/5)_8$的RSC码的网格图的一段

Fig. 2 A trellis section of $(1,7/5)_8$ RSC code

给出了图 1 所示的 4 状态 RSC 编码器网格图的一段，其中与 0 码和 1 码对应的网格转移分别用虚线和实线给出。为了计算 $P_r(u_n=1|R_1^N)$和 $P_r(u_n=0|R_1^N)$，可以将网格图中对应于 $u_n=1$ 和 $u_n=0$ 的转移概率相加。n 时刻编码器状态记为 $S_n\in\{0,1,\cdots,2^v-1\}$，其中 v 为编码器的级数，则

$$P_r(u_n=i|R_1^N)=\sum_{s'=0}^{2^v-1}\sum_{s=0}^{2^v-1}P_r(u_n=i,S_{n-1}=s',S_n=s|R_1^N)\quad u_n=0,1 \tag{4}$$

为了计算上述状态转移概率，可定义联合概率密度函数为：

$$\sigma_n(s',s)=p(u_n=i,S_{n-1}=s',S_n=s,R_1^N) \tag{5}$$

从而有：

$$P_r(u_n=i|R_1^N)=\sum_{s'=0}^{2^v-1}\sum_{s=0}^{2^v-1}\sigma_n(s',s)/p(R_1^N) \tag{6}$$

概率密度函数 $\sigma_n(s',s)$可以通过分解成为 $\tilde{\alpha}_{n-1}(s')$，$\tilde{\gamma}_n(R_n,s',s)$和 $\tilde{\beta}_n(s)$三个部分后以递归的方式计算得到。首先，

$$\begin{aligned}\sigma_n(s',s)&=p(u_n=i,S_{n-1}=s',S_n=s,R_1^N)\cdot p(R_{n+1}^N|u_n=i,S_{n-1}=s',S_n=s,R_1^n)=\\&p(S_{n-1}=s',R_1^{n-1})\cdot p(u_n=i,S_n=s,R_n|S_{n-1}=s',R_1^{n-1})\cdot p(R_{n+1}^N|u_n=i,S_{n-1}=s',S_n=s,R_1^n)\end{aligned} \tag{7}$$

由于在 S_{n-1}已知的情况下信息比特 u_n，编码器状态 S_n 及译码器输入 R_n 均与 R_1^{n-1} 无关，所以有

$$p(u_n=i,S_n=s,R_n|S_{n-1}=s',R_1^{n-1})=p(u_n=i,S_n=s,R_n|S_{n-1}=s')$$

类似地，

$$p(R_{n+1}^N|u_n=i,S_{n-1}=s',S_n=s,R_1^n)=p(R_{n+1}^N|S_n=s)$$

从而有

$$\begin{aligned}\sigma_n(s',s)&=p(S_{n-1}=s',R_1^{n-1})\cdot p(u_n=i,S_n=s,R_n|S_{n-1}=s')\cdot p(R_{n+1}^N|S_n=s)=\\&\tilde{\alpha}_{n-1}(s')\tilde{\gamma}_n(R_n,s',s)\tilde{\beta}_n(s)\end{aligned} \tag{8}$$

式(8)中

$$\tilde{\alpha}_{n-1}(s')=p(S_{n-1}=s',R_1^{n-1}) \tag{9}$$

$$\tilde{\gamma}_n(R_n,s',s)=p(u_n=i,S_n=s,R_n|S_{n-1}=s') \tag{10}$$

$$\tilde{\beta}_n(s)=p(R_{n+1}^N|S_n=s) \tag{11}$$

对于 $\tilde{\alpha}$ 及 $\tilde{\beta}$ 的计算可分别以递归的方式来实现，具体如下：

$$\tilde{\alpha}_n(s) = p(S_n = s, R_1^n) =$$

$$\sum_{s'=0}^{2^{v}-1}\sum_{i=0}^{1} p(u_n = i, S_{n-1} = s', S_n = s, R_1^n) =$$

$$\sum_{s'=0}^{2^{v}-1}\sum_{i=0}^{1} p(S_{n-1} = s', R_1^{n-1}) \cdot p(u_n = i, S_n = s,$$

$$R_n | S_{n-1} = s', R_1^{n-1}) = \sum_{s'=0}^{2^{v}-1}\sum_{i=0}^{1} p(S_{n-1} = s', R_1^{n-1}) \cdot$$

$$p(u_n = i, S_n = s, R_n | S_{n-1} = s') =$$

$$\sum_{s'=0}^{2^{v}-1}\sum_{i=0}^{1} \tilde{\alpha}_{n-1}(s')\tilde{\gamma}_n(R_n, s', s) \tag{12}$$

$$\tilde{\beta}_n(s') = p(R_{n+1}^N | S_n = s') =$$

$$\sum_{s=0}^{2^{v}-1}\sum_{i=0}^{1} p(u_{n+1} = i, S_{n+1} = s, R_{n+2}^N | S_n = s') =$$

$$\sum_{s=0}^{2^{v}-1}\sum_{i=0}^{1} p(R_{n+2}^N | u_{n+1} = i, S_{n+1} = s, R_{n+1}, S_n = s') \cdot$$

$$p(u_{n+1} = i, S_{n+1} = s, R_{n+1}, S_n = s') / p_r(S_n = s') =$$

$$\sum_{s=0}^{2^{v}-1}\sum_{i=0}^{1} p(R_{n+2}^N | S_{n+1} = s) p(u_{n+1} = i, S_{n+1} = s,$$

$$R_{n+1} | S_n = s') = \sum_{s=0}^{2^{v}-1}\sum_{i=0}^{1} \tilde{\beta}_{n+1}(s)\tilde{\gamma}_{n+1}(R_{n+1}, s', s) \tag{13}$$

上述两式计算及 $\tilde{\alpha}\tilde{\beta}$ 的过程分别被称为正向递归和反向递归。

接下来的是 $\tilde{\gamma}_n(R_n, s', s)$ 的计算，根据(10)式 $\tilde{\gamma}_n(R_n, s', s)$ 可以被表示为

$$\tilde{\gamma}_n(R_n, s', s) = p(u_n = i, S_n = s, R_n | S_{n-1} = s') =$$

$$p(R_n | u_n = i, S_n = s, S_{n-1} = s') \cdot$$

$$P_r(S_n = s | u_n = i, S_{n-1} = s') \cdot P_r(u_n = i) \tag{14}$$

在给定网格转移下，假定译码器的输入相互独立，并用 $(x_n, y_n, \tilde{\Lambda}_n)$ 代替 R_n，可得

$$\tilde{\gamma}_n(R_n, s', s) = p(x_n, y_n, \tilde{\Lambda}_n | u_n = i, S_n = s,$$

$$S_{n-1} = s') \cdot P_r(S_n = s | u_n = i, S_{n-1} = s') \cdot$$

$$P_r(u_n = i) = p(x_n | u_n = i, S_n = s, S_{n-1} = s') \cdot$$

$$p(y_n | u_n = i, S_n = s, S_{n-1} = s') \cdot$$

$$p(\tilde{\Lambda}_n | u_n = i, S_n = s, S_{n-1} = s') \cdot$$

$$P_r(S_n = s | u_n = i, S_{n-1} = s') \cdot P_r(u_n = i) =$$

$$p(x_n | u_n = i) \cdot p(y_n | u_n = i, S_{n-1} = s') \cdot$$

$$p(\tilde{\Lambda}_n | u_n = i) \cdot P_r(u_n = i) \cdot$$

$$P_r(S_n = s | u_n = i, S_{n-1} = s') \tag{15}$$

其中样值 x_n、y_n 分别是均值为 $(2u_n - 1)$ 和 $(2c_n - 1)$ 的高斯随机变量，其方差为 σ^2，即：

$$p(x_n | u_n = i) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x_n - (2i-1))^2}{2\sigma^2}} \tag{16}$$

$$p(y_n | u_n = i, S_{n-1} = s') = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(y_n - (2c_n - 1))^2}{2\sigma^2}} \tag{17}$$

又因为 $P_r(u_n = i | \tilde{\Lambda}_n)$ 是信息位 $u_n = i$ 的先验概率，而 $\tilde{\Lambda}_n = \ln \frac{P_r(u_n = 1)}{P_r(u_n = 0)}$，故有

$$P_r(u_n = 1 | \tilde{\Lambda}_n) = \frac{e^{\tilde{\Lambda}_n}}{1 + e^{\tilde{\Lambda}_n}} \tag{18}$$

$$P_r(u_n = 0 | \tilde{\Lambda}_n) = \frac{1}{1 + e^{\tilde{\Lambda}_n}} \tag{19}$$

另外，在式(15)中概率密度函数 $p(\tilde{\Lambda}_n)$ 未知，然而在计算如下所示的对数似然比 Λ_n 时，$p(\tilde{\Lambda}_n)$ 对于分子和分母均为常数。同样地，尽管从理论上讲计算 $\tilde{\alpha}$ 及 $\tilde{\beta}$ 时需要 $p(\tilde{\Lambda}_n)$，然而由于对于给定的一段网格(即确定的 n)，$p(\tilde{\Lambda}_n)$ 对所有的转移均相同，所以，对于 $\tilde{\alpha}_n(s)$ 和 $\tilde{\beta}_n(s)$，$s \in \{0, \cdots, 2^{v-1}\}$，$p(\tilde{\Lambda}_n)$ 均表现为一常数因子。另外，根据是否存在以输入信息 i 相关联的状态 s' 到 s 之间的网格转移，式(15)中的概率 $P_r(S_n = s | u_n = i, S_{n-1} = s')$ 要么为 1，要么为 0。

从而对数后验似然比 Λ_n 可表示为：

$$\Lambda_n = \ln \frac{P_r(u_n = 1 | R_1^N)}{P_r(u_n = 0 | R_1^N)} =$$

$$\ln \frac{\sum_{s'}\sum_{s} \tilde{\alpha}_{n-1}(s')\tilde{\gamma}_n(R_n, s', s)\tilde{\beta}_n(s)}{\sum_{s'}\sum_{s} \tilde{\alpha}_{n-1}(s')\tilde{\gamma}_n(R_n, s', s)\tilde{\beta}_n(s)} =$$

$$\ln \frac{\sum_{s'}\sum_{s} (\tilde{\alpha}_{n-1}(s')p(x_n | u_n = 1)}{\sum_{s'}\sum_{s} (\tilde{\alpha}_{n-1}(s')p(x_n | u_n = 0)} \rightarrow$$

$$\frac{p(y_n | u_n = 1, S_{n-1} = s')P_r(u_n = 1 | \tilde{\Lambda}_n)p(\tilde{\Lambda}_n)}{p(y_n | u_n = 0, S_{n-1} = s')P_r(u_n = 0 | \tilde{\Lambda}_n)p(\tilde{\Lambda}_n)} \rightarrow$$

$$\frac{P_r(S_n = s | u_n = 1, S_{n-1} = s')\tilde{\beta}_n(s))}{P_r(S_n = s | u_n = 0, S_{n-1} = s')\tilde{\beta}_n(s))} =$$

$$\ln \frac{p(x_n | u_n = 1)p(u_n = 1 | \tilde{\Lambda}_n)}{p(x_n | u_n = 0)p(u_n = 0 | \tilde{\Lambda}_n)} \rightarrow$$

$$\frac{\sum_{s}\sum_{s'}\tilde{\alpha}_{n-1}(s')p(y_n|u_n=1,S_{n-1}=s')}{\sum_{s}\sum_{s'}\tilde{\alpha}_{n-1}(s')p(y_n|u_n=0,S_{n-1}=s')}\cdot\frac{P_r(S_n=s|u_n=1,S_{n-1}=s')\tilde{\beta}_n(s))}{P_r(S_n=s|u_n=0,S_{n-1}=s')\tilde{\beta}_n(s))}=$$

$$\ln\frac{p(x_n|u_n=1)}{p(x_n|u_n=0)}+\ln\frac{p(u_n=1|\tilde{\Lambda}_n)}{p(u_n=0|\tilde{\Lambda}_n)}+$$

$$\ln\frac{p(u_n=1|x_1^{n-1},x_{n+1}^N,y_1^N,\tilde{\Lambda}_1^{n-1},\tilde{\Lambda}_{n+1}^N)}{p(u_n=0|x_1^{n-1},x_{n+1}^N,y_1^N,\tilde{\Lambda}_1^{n-1},\tilde{\Lambda}_{n+1}^N)}=$$

$$\frac{2}{\sigma^2}x_n+\tilde{\Lambda}_n+L_{e_n} \tag{20}$$

其中 L_{e_n} 是译码器的外信息输出。Λ_n 的这一形式对于迭代译码是非常有用的，因为此时只有外信息 L_{e_n} 被送到下一级的译码过程。由于上式中的第 1、2 两项容易由译码器的输入求出，因此可直接得到译码器的外信息输出：

$$L_{e_n}=\Lambda_n-\frac{2}{\sigma^2}x_n-\tilde{\Lambda}_n=$$

$$\ln\frac{\sum_{s}\sum_{s'}\tilde{\alpha}_{n-1}(s')\tilde{\gamma}_n^1(R_n,s',s)\tilde{\beta}_n(s)}{\sum_{s}\sum_{s'}\tilde{\alpha}_{n-1}(s')\tilde{\gamma}_n^0(R_n,s',s)\tilde{\beta}_n(s)}-\frac{2}{\sigma^2}x_n-\tilde{\Lambda}_n \tag{21}$$

3 Turbo 码译码中 BCJR 算法的实现

上面计算对数似然比的式子中，对分子分母中的所有求和项均相同的因子可以消去。首先我们看 $\tilde{\gamma}_n^1(R_n,s',s)$：

$$\tilde{\gamma}_n(R_n,s',s)=\frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{(x_n-(2i-1))^2}{2\sigma^2}}\frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{(y_n-(2c_n-1))^2}{2\sigma^2}}\cdot$$

$$\frac{e^{i\tilde{\Lambda}_n}}{1+e^{\tilde{\Lambda}_n}}p(\tilde{\Lambda}_n)P_r(S_n=s|u_n=i,S_{n-1}=s')=$$

$$\frac{p(\tilde{\Lambda}_n)}{2\pi\sigma^2(1+e^{\tilde{\Lambda}_n})}e^{-\frac{x_n^2+(2i-1)^2+y_n^2+2(2c_n-1)^2}{2\sigma^2}}\cdot$$

$$e^{\frac{x_n(2i-1)+y_n(2c_n-1)}{\sigma^2}+i\tilde{\Lambda}_n}P_r(S_n=s|u_n=i,S_{n-1}=s')$$

由 $(2i-1)^2=1$ 及 $(2c_n-1)^2=1$，我们定义：

$$C_{\gamma_n}=\frac{p(\tilde{\Lambda}_n)}{2\pi\sigma^2(1+e^{\tilde{\Lambda}_n})}e^{-\frac{x_n^2+y_n^2+2}{2\sigma^2}} \tag{23}$$

及：

$$\gamma_n(R_n,s',s)=e^{\frac{x_n(2i-1)+y_n(2c_n-1)}{\sigma^2}+i\tilde{\Lambda}_n}\cdot$$

$$P_r(S_n=s|u_n=i,S_{n-1}=s') \tag{24}$$

则：
$$\tilde{\gamma}_n(R_n,s',s)=C_{\gamma_n}\gamma_n(R_n,s',s) \tag{25}$$

由于 C_{γ_n} 与 s 及 s' 均无关，因此可以用 $\gamma_n(R_n,s',s)$ 来代替 $\tilde{\gamma}_n(R_n,s',s)$。进而正向递归可以写成：

$$\tilde{\alpha}_n(s)=\sum_{s'=0}^{2^v-1}\sum_{i=0}^{1}\tilde{\alpha}_{n-1}(s')\tilde{\gamma}_n(R_n,s',s)=$$

$$C_{\gamma_n}\sum_{s'=0}^{2^v-1}\sum_{i=0}^{1}\tilde{\alpha}_{n-1}(s')\gamma_n(R_n,s',s) \tag{26}$$

由于 C_{γ_n} 对所有的 $\tilde{\alpha}_n(s),s\in\{0,1,\cdots,2^{v-1}\}$ 均相同，因此可以从递归运算中省去。对于反向递归也有类似的结论。这样，我们可以定义：

$$\alpha_n(s)=\sum_{s'=0}^{2^v-1}\sum_{i=0}^{1}\alpha_{n-1}(s')\gamma_n(R_n,s',s) \tag{27}$$

$$\beta_n(s')=\sum_{s=0}^{2^v-1}\sum_{i=0}^{1}\beta_{n+1}(s)\gamma_{n+1}(R_n,s',s) \tag{28}$$

根据编码器处于不同状态的概率，可以直接对 α 进行初始化。由于编码器是从全零状态开始工作的，因此有：

$$\alpha_0(s)=\begin{cases}1 & s=0\\ 0 & \text{其它}\end{cases} \tag{29}$$

类似地，若编码器结束于全零状态，则有：

$$\beta_N(s)=\begin{cases}1 & s=0\\ 0 & \text{其它}\end{cases} \tag{30}$$

若编码器并不结束于全零状态，则结束于所有状态的概率均等，从而：

$$\beta_N(s)=\frac{1}{2^v}\qquad s=0,\cdots,2^v-1 \tag{31}$$

另外，随着递归过程的进行，α 及 β 的值会越来越小，从而可能导致数值运算的下溢，特别是当数据分组较长时，对此可以采用重新定标的方法来避免。例如，对于每个特定的时刻 n，可以通过重新定标的方法使所有的 α 及 β 之和为 1，即 $\sum_s\alpha_n(s)=1$ 及 $\sum_s\beta_n(s)=1$。

除此之外，BCJR 算法的实现复杂性还可以通过在对数域计算来进一步降低。此时可利用近似关系 $\ln(e^{\delta_1}+e^{\delta_2})\approx\max(\delta_1,\delta_2)$ 得到一种次优的 Max-Log-MAP 算法。此时假设：

$$\ln(1+e^{-|\delta_2-\delta_1|})\approx 0 \tag{32}$$

（下转 50 页）

3,5,6 号纤芯。

表 1　光纤自动测试状态一览表

Tab. 1　Chart of optical fiber automatic testing results

光纤序号	状况	设定值	测试值	改变量(dB)
1	正常	15.2	15.4	0.2
2	正常	15.3	15.2	−0.1
3	障碍	15.2	21.2	6.2
4	正常	15.2	15.0	−0.2
5	障碍	15.0	21.0	6.0
6	障碍	15.1	30.4	15.3
7	正常	15.1	15.2	0.1
8	正常	15.2	15.6	0.4
9	正常	15.2	15.6	0.4
10	正常	14.7	15.2	0.7
11	预警	15.2	18.7	3.5
12	未用			
13	未用			
14	正常	15.3	15.1	−0.2
15	正常	15.0	15.2	0.2

表 2　光纤自动测试异常状态一览表

Tab. 2　Chart of irregular state of optical fiber automatic testing

障碍状态			预警状态		
光纤序号	改变量	确认	光纤序号	改变量	确认
6	15.3	OK	11	3.5	OK
3	6.2	OK			
5	6.0	OK			

光缆监测系统在线路维护上是十分迫切和必要的，它对预防障碍的发生，压缩障碍历时，延长光缆使用寿命将起到关键性作用，同时它所带来的隐形效益是不可估量的，对保证全网的安全畅通，提高运行效率，将起到积极作用，也为今后研究光纤、光缆的物理变化及使用寿命等课题提供可靠的依据。随着光通信技术的不断发展，未来的光通信网络将离不开监测系统所带来的潜在效益。借助光缆监测系统构建一个全光监测网必然会随着光通信的发展需求而在我国逐步建立起来。我们有理由相信在通信、计算机、电子技术不断发展的今天，未来的监测系统所发挥的作用将是巨大的。

参 考 文 献

[1] MAIER Frank A. 光时域反射仪的自动测量[J]. 光纤通信技术，1997，(4)：42-47.

[2] 韦乐平. 光通信系统技术的发展与展望[J]. 现代电信科技，2001，(1)：1-6.

[3] 王俊杰. 如何组建我国的光缆自动监测网[J]. 电信技术，1996，(6)：25-27.

（编辑：郭继笃）

（上接 29 页）因为，$\ln(e^{\delta_1}+e^{\delta_2})=\max(\delta_1,\delta_2)+\ln(1+e^{-|\delta_2-\delta_1|})$。

另外，也可以事先计算好 $\ln(1+e^{-|\delta_2-\delta_1|})=f_c(|\delta_2-\delta_1|)$ 的值，并将其存放在一维的查找表格里。通过这种方法，可以达到接近理想 BCJR 算法的精度，而计算复杂性却大大降低。

4　结束语

本文对 BCJR 算法进行了详细的推导，并简单讨论了一些有关该算法在 Turbo 码译码应用中的实现问题。实践及理论研究均表明，BCJR 算法对于 Turbo 码译码性能的提高具有相当重要的意义。

参 考 文 献

[1] BAHL L R, COCKE J, JELINEK F, et al. Optimal decoding of linear codes for minimizing symbol error rate[J]. IEEE Transactions on Information Theory, 1974, 284-286.

[2] BERROU C, GLAVIEUX A. Near optimum error correcting coding and decoding: Turbo Codes[C]. In Proc IEEE International Conference on Communication (ICC), Geneva, Switzerland, 1993, 1064-1070.

[3] JUNG P, NAPHAN M M. Comprehensive comparison of turbo-code decoders[C]. In IEEE Vehicular Technology Conference, Chicago, USA, 1995, 624-628.

[4] ROBERTSON P. Improving decoder and code structure parallel concatenated recursive systematic (turbo) codes[C]. In IEEE International Conference on Universal Personal Communications. San Diego, USA, 1994, 183-187.

（编辑：何先刚）