一种 Turbo 码译码的矩阵算法 [1]

张忠培 周 亮*

(清华大学微波与数字通信国家重点实验室 北京 100084)
*(电子科技大学国防通信抗干扰实验室 成都 610054)

摘 要 在 Bahl 矩阵算法的基础上，提出了 Turbo 码译码的矩阵算法，使 Turbo 码的复杂迭代运算简化为适用于大规模集成电路的矩阵运算，运算速度得以提高，数据存量变小，译码过程简单明了，特别适用于约束长度较小的 Turbo 码译码。讨论了第三代移动通信 Turbo 编码的状态转移图及矩阵译码过程。

关键词 MAP 算法，Turbo 码，迭代译码，矩阵算法
中图号 TN911.31

1 引 言

Turbo 码自 1993 年提出以后 [1]，对它的译码采用改进 Bahl 算法，即基于码元的最大后验概率译码算法 (MAP) 算法 [2]。然后，又提出了 MAX-LOG-MAP 算法，软判决输出维特比译码算法 (SOVA) 等简化算法，但简化算法在误码率为 10^{-4} 时，有 1dB 左右译码增益损失 [3]。在第三代移动通信建议 [4](3GPP) 中，信道编码的 Turbo 码要求工作在 10^{-3}-10^{-6} 的误码率范围，且要求编译码增益较高，对 Turbo 码的译码实现问题提出了新的要求，并且 Turbo 码由 8 状态 (15, 13)RS 码并行级联而成，状态数越少，其纠错能力相应越低，要取得较高编译码增益使用简化算法有一定困难。本文旨在不损失译码增益的前提下，使译码减化。根据文献 [5]Bahl 算法的矩阵实现方法给出了 Turbo 译码的矩阵算法，并以 3GPP 中的 Turbo 码为例，证明了矩阵译码算法的正确性，与 MAP 算法在实现复杂度进行了比较，为 3GPP 中 Turbo 码的译码实现，给出了理论依据。

2 Turbo 码 MAP 译码算法

为了由 MAP 算法推得矩阵算法，将 MAP 算法简述如下 [2]：Turbo 码的编、译码原理如图 1，接收信息为 X_k ，Y_k ，而 Y_k 由 Y_{1k} 与 Y_{2k} 两路校验位合成，对无记忆离散高斯信道和二进制调制，在时间 k 时：$X_k=(2x_k-1)+n_k$ ，$Y_k=(2y_k-1)+q_k$ ，其中，x_k、y_k 为编码后所得信号，n_k、q_k 是具有方差为 σ^2 的独立高斯噪声信号。

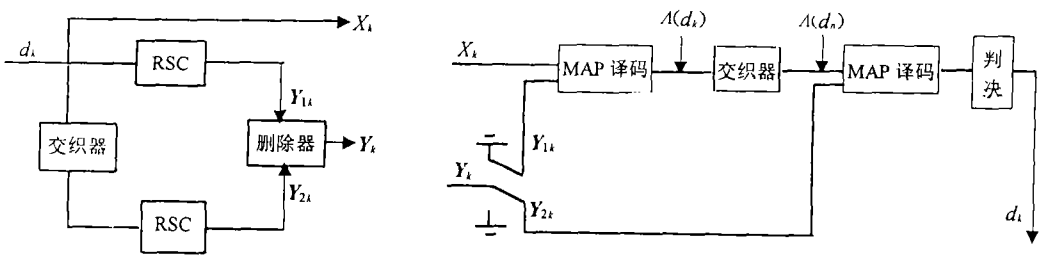

图 1 Turbo 码编码、译码原理图

对于约束长度为 v 的 RS 码，在 k 时的编码状态为：$S_k=(a_1,a_{k-1},\cdots,a_{k-v+2})$ 状态数 $M=2^{v-1}$，假设输入信息序列 $\{d_k\}$ 由 N 个独立比特 d_k 构成，取 0,1 的概率相等，且初始状态 S_0

[1] 1999-11-30 收到，2000-04-07 定稿
中兴通讯研究基金资助

和终止状态 S_N 已回到 0 状态，则 $S_0 = S_N = (0, 0, \cdots, 0)$，对编码输出 $C_1^N = \{C_1 \cdots C_k \cdots C_N\}$，$C_k = (x_k, y_k)$，离散信道 (DMC) 的输出为 $R_1^N = \{R_1 \cdots R_k \cdots R_N\}$，$R_k = (X_k, Y_k)$。为了计算软判决输出，引入 3 个量 α、β、γ，分别由式 (1)、(2)、(3) 经前向和后向递归迭代求出。

$$\alpha_k(m) = \frac{\sum_{m'} \gamma_0(R_k, m', m)\alpha_{k-1}(m') + \sum_{m'} \gamma_1(R_k, m', m)\alpha_{k-1}(m')}{\sum_m \sum_{m'} \gamma_0(R_k, m', m)\alpha_{k-1}(m') + \sum_m \sum_{m'} \gamma_1(R_k, m', m)\alpha_1(m')} \tag{1}$$

$$\beta_k(m) = \frac{\sum_{m'} \gamma_0(R_{k+1}, m', m)\beta_{k=1}(m) + \sum_{m'} \gamma_1(R_{k+1}, m', m)\beta_{k+1}(m)}{\sum_m \sum_{m'} \gamma_0(R_{k+1}, m', m)\alpha_k(m') + \sum_m \sum_{m'} \gamma_1(R_{k+1}, m', m)\beta_{k+1}(m)} \tag{2}$$

$$\begin{aligned}\gamma_i(R_k, m', m) =& p(R_k | d_k = i, s_k = m, s_{k-1} = m') \\ & \times q(d_k = i | s_k = m, s_{k-1} = m')\pi(s_k = m | s_{k-1} = m')\end{aligned} \tag{3}$$

其中 m' 代表状态转换前的状态，m 代表状态转换后的状态，p 为信道转移概率，q，π 为状态转移概率。初始条件：

$$\alpha_0(0) = 1, \quad \alpha_0(m) = 0, \quad \forall m \neq 0; \quad \beta_N(0) = 1, \quad \beta_N(m) = 0, \quad \forall m \neq 0 \tag{4}$$

由于 Turbo 码由 RS 编码并行级联而成，RS 码为递归系统卷积码，$x_k = d_k$，故 $p(x_k | d_k = i, s_k = m, s_{k-1} = m')$ 与状态 S_k，S_{k-1} 是独立的，得到似然软判决 $\Lambda(d_k)$ 为

$$\Lambda(d_k) = \log \frac{p(x_k | d_k = 1)}{p(x_k | d_k = 0)} + \log \frac{\sum_m \sum_{m'} \gamma_1(y_k, m', m)\alpha_{k-1}(m')\beta_k(m)}{\sum_m \sum_{m'} \gamma_0(y_k, m', m)\alpha_{k-1}(m')\beta_k(m)} \tag{5}$$

MAP 算法的基本步骤如下：

(1) 根据 (5) 式初始化 α，β。

(2) 由 (1),(3) 式计算 α，γ。

(3) 当 R_1^N 完全接收时，由 (2) 式计算 β，由 (5) 式计算每一译码比特 d_K。

3 Turbo 码译码的矩阵算法

由 MAP 算法知，在时刻 k，$\gamma_i(m', m)$ 共有 $M \times M$ 种可能的情况，为了简化，将 $i = 0$ 和 $i = 1$ 分别定义为两个矩阵，Γ_k^0 表示 $d_k = 0$ 时对应状态转移矩阵，Γ_k^1 表示 $d_k = 1$ 时对应的状态转移矩阵。

$$\Gamma_k^1 = \begin{bmatrix} \gamma_k^1(R_k, 0, 0) & \gamma_k^1(R_k, 0, 1) & \cdots & \gamma_k^1(R_k, 0, M) \\ \gamma_k^1(R_k, 1, 0) & \gamma_k^1(R_k, 1, 1) & \cdots & \gamma_k^1(R_k, 1, M) \\ \vdots & \vdots & \vdots & \vdots \\ \gamma_k^1(R_k, M, 0) & \gamma_k^1(R_k, M, 1) & \cdots & \gamma_k^1(R_k, M, M) \end{bmatrix} \tag{6}$$

$$\Gamma_k^0 = \begin{bmatrix} \gamma_k^0(R_k, 0, 0) & \gamma_k^0(R_k, 0, 1) & \cdots & \gamma_k^0(R_k, 0, M) \\ \gamma_k^0(R_k, 1, 0) & \gamma_k^0(R_k, 1, 1) & \cdots & \gamma_k^0(R_k, 1, M) \\ \vdots & \vdots & \vdots & \vdots \\ \gamma_k^0(R_k, M, 0) & \gamma_k^0(R_k, M, 1) & \cdots & \gamma_k^0(R_k, M, M) \end{bmatrix} \tag{7}$$

又由 MAP 算法，在时刻 k，$\alpha_k(m)$，$\beta_k(m)$ 都有 M 种情况，都有 M 个值，作为一 M 维向量或一 $1 \times M$ 级矩阵的元素，可得到 M 维向量或一 $1 \times M$ 级矩阵如下：

$$\alpha_k = [\alpha_k(0) \quad \alpha_k(1) \quad \cdots \quad \alpha_k(M)] \qquad \beta_k = [\beta_k(0) \quad \beta_k(1) \quad \cdots \quad \beta_k(M)]$$

对于边界条件，在 $k=0,N$ 时，$\alpha_0=(1,0,\cdots,0)$，$\beta_N=(1,0,\cdots,0)$。由 (1) 式中的求和，

$$\sum_{m'}\sum_{i=0}^{1}\gamma_i(R_k,m',m)\alpha_{k-1}(m')=\sum_{m'}\gamma_0(R_k,m',m)\alpha_{k-1}(m')+\sum_{m'}\gamma_1(R_k,m',m)\alpha_{k-1}(m')$$

其中，

$$\begin{aligned}\sum_{m'}\gamma_0(R_k,m',m)\alpha_{k-1}(m')=&\gamma_0(R_k,0,m)\alpha_{k-1}(0)+\gamma_0(R_k,1,m)\alpha_{k-1}(1)+\cdots\\&+\gamma_0(R_k,M,M)\alpha_{k-1}(M)\\=&[\gamma_0(R_k,0,m)\ \ \gamma_0(R_k,1,m)\ \ \cdots\ \ \gamma_0(R_k,M,M)]\\&\bullet[\alpha_{k-1}(0)\ \ \alpha_{k-1}(1)\ \ \cdots\ \ \alpha_{k-1}(M)]^T\end{aligned}\tag{8}$$

由 (8) 式，对所有状态 m，可定义一矩阵。$A_{\alpha k}$ 表示 (1) 式中分子的 γ,α 之乘积。

$$\begin{aligned}&A_{\alpha k}=[A_{\alpha k}(0)\ \ A_{\alpha k}(1)\ \ \cdots\ \ A_{\alpha k}(M)]\\&A_{\alpha k}=\Gamma_k^0\bullet\alpha_{k-1}^T+\Gamma_k^1\bullet\alpha_{k-1}^T\end{aligned}\tag{9}$$

则可得 α_k 的矩阵表示，I 为单位矩阵：

$$\alpha_k=A_{\alpha k}/(I\bullet A_{\alpha k})\tag{10}$$

同理将 (2) 式变成矩阵形式，定义 β,γ 之积求和的矩阵为 $A_{\beta k}$：

$$A_{\beta k+1}=\Gamma_{k+1}^0\bullet\beta_{k+1}^T+\Gamma_{k+1}^1\bullet\beta_{k+1}^T\ \ \text{且}\ \ A_{\beta k+1}=[A_{\beta k+1}(0)\ \ A_{\beta k+1}(1)\ \ \cdots\ \ A_{\beta k+1}(M)]$$

则得到 β_k 的矩阵表示：

$$\beta_k=A_{\beta k+1}/(I\bullet A_{\alpha k+1})\tag{11}$$

将 (6),(7) 式及 α_k,β_k 的矩阵定义代入 (5) 式，得到软判决输出信号 d_k 的矩阵表达式：

$$\Lambda(d_k)=\log\frac{p(x_k|d_k=1)}{p(y_k|d_k=0)}+\log\frac{(\Gamma_k^1\bullet\alpha_{k-1})^T\bullet\beta_k}{(\Gamma_k^0\bullet\alpha_{k-1})^T\bullet\beta_k}\tag{12}$$

矩阵算法计算过程如下：

(1) 根据边界条件初始化向量 $\alpha_0=[1,0,\cdots,0]$ 和 $\beta_N=[1,0,\cdots,0]$；

(2) 译码器收到 Y_k，根据 RS 码的状态转移图，由 (6),(7) 式得到 γ_k^0，γ_k^1，由 (10) 式计算 α_k；

(3) 当 N 个值全部接收以后，根据 (11) 式计算 β_k，由 (12) 式软判决输出 d_k 值。

计算流程图如图 2：

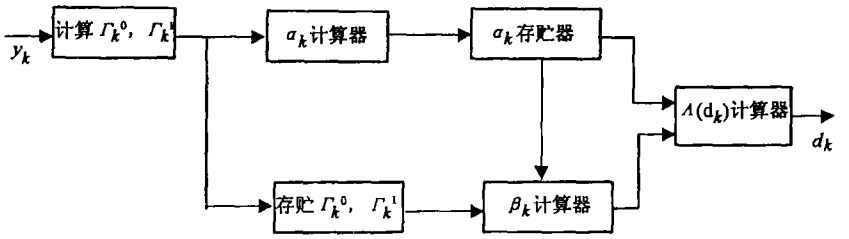

图 2　Turbo 码译码的矩阵算法流程图

矩阵算法与 MAP 算法比较：

(1) 从计算形式上看，MAP 算法以单个数为计算单位，并反复进行累加，而矩阵算法以 K 维向量为单位进行计算，从而使计算速度提高，并减少了数据传输和存贮时间。

(2) 由于矩阵算法使运算公式由矩阵形式给出，便于集成电路实现，将多次相乘累加变成矩阵相乘，利于并行实现。

(3) 由于 MAP 算法自身特点，用矩阵算法，同样存在要求系统存贮量大，并且递归计处算 $\alpha\beta$ 时，由于分母的存在，不能利用 Γ 累乘来计算 β，达到减少 β 计算时间的目的。

4 译码实例

在第三代移动通信建议里，信道编码采用的是两个 8 状态 RS 码构成 Turbo 码，其 RS 编码如图 3，它的编码多项式为：$G(D)=[1, n(D)/d(D)]$，$d(D)=1+D^2+D^3$，$n(D)=1+D+D^3$。用八进制可表示为 (15, 13)RS 码，其状态网格图如图 4。

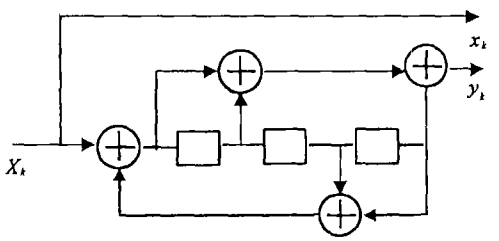

图 3　(15, 13)RS 编码图

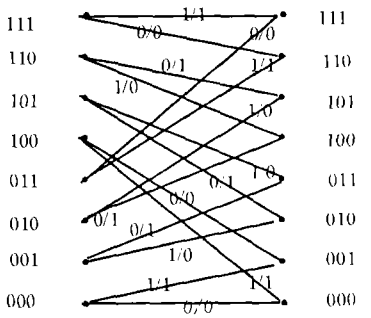

图 4　(15, 13)RSC 码状态转移网格图

图 4 中，数字标注“Y/X”表示输入为 X 值，输出为 Y，因为 RS 码为系统码，故只标出了一个输出量。从图 4 可以看出，RS 码与非递归系统卷积码在网格图上有很大区别，特别是状态不容易回到零状态，因此，在实际应用中，网格终止是在所有的信息比特编码完成后，让寄存器的数据反馈回输入端，让寄存器的状态回零。设 DMC 的转移概率 $P_e=0.1$，输入序列 d_k 为 (10110000)，编码输出为 (11, 01, 10, 11, 00, 00, 00, 00)，接收序列 R=(11, 01, 00, 11, 01, 00, 10, 00)，由文献 [2] 中 γ 的表达式 $\gamma_i(R_k, m', m)=p(R_k|d_k=i, S_k=m, S_{k-1}=m')\cdot q(d_k=i|S_k=m, S_{k-1}=m')\cdot\pi(S_k=m|S_{k-1}=m')$，$R_k=(x_k, y_k)$，$x_k$ 和 y_k 是相互独立的变量。根据图 4 得到在 k 时刻的 Γ_i 矩阵：

$$\Gamma_i=\begin{bmatrix} A_{00} & A_{11} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & A_{01} & A_{10} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & A_{10} & A_{01} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & A_{00} & A_{11} \\ A_{11} & A_{00} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & A_{10} & A_{01} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & A_{10} & A_{01} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & A_{00} & A_{11} \end{bmatrix}$$

式中 $A_{00}=\frac{1}{2}R(y_k|00)$, $A_{11}=\frac{1}{2}R(y_k|11)$, $A_{10}=\frac{1}{2}R(y_k|10)$, $A_{01}=\frac{1}{2}R(y_k|01)$ 。

由 Γ_i 矩阵得到 $\Gamma_0\Gamma_1$ 矩阵。利用矩阵算法得到的值如表 1：

表 1 输入序列 d_k 的矩阵译码

时刻 k	$\alpha=(\alpha_{(0)}\alpha_{(1)}\cdots\alpha_{(7)})$	$\beta=(\beta_{(0)}\beta_{(1)}\cdots\beta_{(7)})$	$\Lambda(d_k)$	d_k
0	1,0.0,0			
	0.0,0,0			
1	0.0909,0,0,0,	1.3662,1.3554,1.3411.0.0302	0.9542	1
	0.9091,0,0,0	0.4253,0.0341,0.2774,0.0302		
2	0.0902,0,0.9008,0,	0.0557,0.2552,0.0352,0.0030,	−0.1426	0
	0.0089,0,0,0	0.0355,0.2525.0.0277,0.0030		
3	0.0472,0.4717,0.0047,0,	0.1005,0 0577,0.4817,0.0048	0.1086	1
	0.0047,0.4717,0,0	0.0619.0.0524,0.0052.0.0048		
4	0.3049,0.3021,0.3021,0,	0.1097,0.0603,0.0603,0,0	0.9252	1
	0.0604,0.0003,0.0302,0	0.5538,0.0055,0.0055,0		
5	0.2349,0.2330,0.0424,0.0212,	0.5883,0.0588,0.0588,0,	−1.4848	0
	0.2332,0.2118,0.0023,0.0212	0.0588,0.0588,0,0		
6	0.2347,0.0597,0.2312,0.0040,	0.6473,0.0.0647,0,	−1.6814	0
	0.2331,0.0404,0.0927,0.004	0.0647,0.0.0		
7	0.1631,0.1607,0.1607,0.1309,	0.9552,0,0,0,	−0.5041	0
	0.0563,0.1569,0.0405,0.1309	0.0955,0,0,0		
8	0.1711,0.1689,0.0688,0.0511,	1.0,0,0		
	0.1691,0.1660,0.1538,0.0511	0,0,0.0		

从表 1 的输出 d_k 结果，纠正了传输过程的错误，证明了矩阵算法的正确性。

5 分析及结论

本文推导了 Turbo 码译码的矩阵算法，它不仅是算法上的等效简化运算，而在译码的实现上有着本质的不同。首先，由求和、乘除运算变为向量与矩阵相乘运算，由原来的单处理器运算实现变成可由多处理器并行进行的并行运算，使译码速度提高，延迟减小，并可将 Turbo 码译码多次迭代的流水线结构变成并行的级联流水线结构。其次，由 (15, 13)RS 码的转移矩阵可以看出，该矩阵为稀疏矩阵，而稀疏矩阵在集成电路实现上能进行大量的简化，按稀疏矩阵算法，对 RS 码的状态转移矩阵能进行简化，为 Turbo 码译码实现简化找到了另一新的途径。同时，本文讨论了 3GPP 中信道编码 Turbo 码的状态转移图，用矩阵算法实现了它的译码，证明了本文的矩阵算法的正确性。对 3GPP 中 Turbo 编码的译码实现具有较大的应用价值。

参 考 文 献

[1] C. Berrou, A. Glavieux, P. Thitimajshima, Near Shannon limit error-correcting coding and decoding, Turbo-codes, In ICC'93, Geneva, Switzerland, 1993, 1064–1070.

[2] C. Berrou, A. Glavieux, Near optimum error correcting coding and decoding, Turbo-codes, IEEE Trans. on Commun, 1996, 44(10), 1261–1271.

[3] P. Robertson, P. Hoeher, E. Villebrun, Optimal and sub-optimal maximum *a posteriori* algorithms suitable for Turbo decoding, Europe Trans. on Telecommun, and Related Areas, 1997, 8(2), 119-125.
[4] Third Generation Partnership Project, http://www.3GPP.org
[5] 刘光亮，胡正明，最小化符号错误率译码的矩阵算法，通信学报，1998, 19(8), 1–6.

A MATRIX DECODING ALGORITHM FOR TURBO-CODES

Zhang Zhongpei　　Zhou Liang*

(*State Key Lab on Microwave & Digital Commu., Tsinghua Univ., Beijing* 100084, China)

*(*National Communication Lab. UESTC, Chengdu* 610054, China)

Abstract　A new matrix decoding algorithm for Turbo-codes is derived from Bahl's matrix algorithm. The complex iterating operations are paralleled and well formulated as a set of simple matrix operations which are fit to design efficient VLSI circuits. Thus, the matrix algorithm increases the decoding speed and simplifies the excessive memory accesses, and is specially fit to Turbo-codes with small memories. States transfer and decoding process of Turbo-codes in 3GPP are also discussed in this paper.

Key words　MAP algorithm, Turbo-codes, Iterative decoding, Matrix algorithm

张忠培：　男，1967 年生，讲师，研究方向：信道差错编码，扩频通信.
周　亮：　男，1961 年生，副教授，研究方向：编码理论及应用.