LDPC 编译码原理及其仿真实现

谷林海

摘要：本文从学习理解和仿真实现的角度，总结 LDPC 编译码原理，适合学习 LEPC 码的初学者。如需 LDPC 码的 MATLAB 代码，可联系：ilynchpin@lynchpin.com.cn

1 研究意义

1948 年 C.E.Shannon 发表的著名的《通信的数学理论》一文，为信道编码技术的发展指明了方向。Shannon 在著名的有噪声信道编码定理中，给出了在数字通信系统中实现可靠通信的方法以及在特定信道上实现可靠通信的信息传输速率上限。Shannon 在他的证明中引用了三个基本条件：

（1） 采用随机的编译码方法；

（2） 构造码长的渐进好码或 Shannon 码；

（3） 译码采用最佳的最大似然译码算法。

50 多年来构造好码的思想基本上是按照 Shannon 所引用的基本条件的后两条条为主线进行研究的。

Shannon 提出的有噪声信道编码定理作为现代通信理论的基础，给出了在有噪声信道上实现可靠通信的理论极限。虽然该定理指出了可以通过差错控制码在信息传输速率不大于信道容量的前提下实现可靠通信，但却没有给出具体实现差错控制编码的方法。根据 Shannon 有噪声信道编码定理，在信道传输速率 R 不超过信道容量 C 的前提下，只有在码组长度无限的码集合中随机地选择编码码字并且在接收端采用最大似然译码算法时，才能使误码率接近为零。但是最大似然译码的复杂性随着编码长度的增加而加大，当编码长度趋于无穷大时，最大似然译码是不可能实现的。

1993 年于瑞士日内瓦召开的国际通信会议上，C.Berrou、A.Glavieux 和

P.Thitimajshima 首次提出了一种新型的编码方案——Turbo 码，由于他们很好地应用了 Shannon 信道编码定理中的随机性编、译码条件，从而获得了几乎接近 Shannon 理论极限的译码性能。虽然 Turbo 码标志着人们构造其性能接近 Shannon 限的好码的开始，但 Turbo 码仍未将随机化思想真正贯穿于其编译码的始终，而且它也有许多缺点：（1）译码时延大，这就使 Turbo 码在某些对时延要求高的通信系统（如数字电话）中的应用受到限制；（2）计算量大，为达到高码率需要很大的交织器，这就增加了时延；（3）有错误平层效应。

在深入研究 Turbo 码原理的过程中，1996 年 Mackay、Spielman 和 Wiberg 几乎同时发现:Gallager 早在 1962 年提出的低密度校验码（LDPC，也称 Gallager 码）也是一个好码，具有更低的线性译码复杂度。Gallager 提出 LDPC 码后一直没有得到编码界的重视，只有 1981 年 Tanner 从图论的角度研究过 LDPC 码。

自从 Mackay“再发现”LDPC 码后，人们进一步研究表明：基于非规则双向图的 LDPC 长码的性能优于 Turbo 码，而且这样码的性能可以非常接近 Shannon 限。相对于 Turbo 码，LDPC 码具有以下优点：（1）不需要深度交织以获得好的五马性能；（2）具有更好的分组误码性能；（3）误码平台处的误码率大大降低；（4）译码不基于网格。由于 LDPC 码不仅具有良好的距离特性、小的译码错误概率和较低的译码复杂度，而且适当码长（比如大于 200）时，不存在错误平台，其码率容易调整。实验结果中的错误几乎均为可检测错误。所以 LDPC 码无论在理论上还是实际上都具有及其重要的价值。LDPC 的重新发现时继 Turbo 码后在纠错码领域又一重大进展。

2 LDPC 码技术

2.1 LDPC 码相关知识

2.1.1 线性分组码

线性分组码是把信息划成 k 个码元为一段（信息组），通过编码器变成长度为 n 个码元的一组，这 n 个码元的一组称为码字（码组）。在二进制情况下信息组共有 2^k 个，通过编码器之后，相应的码字也有 2^k 个，称这 2^k 个码字集合为线

性分组码，用（n，k）表示，n 表示码长，k 表示信息位，码率 $R=k/n$。二元线性分组码必须满足如下两个条件：

（1）码字集合中的任意两个码字经过模 2 加之后得到的结果仍然是码字集合中的一个码字。

（2）码字集合中包含有全零码字。

从数学角度讲，可以把一个（n，k）线性分组码看成二元 n 维线性空间上的 k 维子空间。因此，（n，k）线性分组码可以通过由 k 个线性无关的二元 n 维矢量集合 $\{g_0, g_1, \cdots g_{k-1}\}$ 来得到，得到的码字实际上是这些 n 维矢量根据信息序列分组中各个比特的取值而得到的线性组合。

2.1.2 生成矩阵和校验矩阵

线性分组码的编码过程可以描述为一个信息矢量 m 和一个矩阵相乘的结果：

$$C = m \bullet G \quad \text{式（3-1）}$$

其中，G 是有 k 个 n 维矢量 $\{g_0, g_1, \cdots g_{k-1}\}$ 构成的矩阵，其中 $g_0, g_1, \cdots g_{k-1}$ 是 n 维行向量；m 是信息序列分组 $\{m_0, m_1, \cdots, m_{k-1}\}$；C 是编码后得到的 n 维编码输出 $\{c_0, c_1, \cdots, c_{n-1}\}$，其中矢量与矩阵的乘法是在二元域 GF(2)上进行的。

根据式（3-1），码字 C 可以表示为：

$$C = m_0 \bullet g_0 + m_1 \bullet g_1 + \cdots + m_{k-1} \bullet g_{k-1} \quad \text{式（3-2）}$$

而矩阵 G 称为编码生成矩阵，形式为：

$$G = \begin{bmatrix} g_0 \\ g_1 \\ \vdots \\ g_{k-1} \end{bmatrix} = \begin{bmatrix} g_{0,0} & g_{0,1} & \cdots & g_{0,n-1} \\ g_{1,0} & g_{1,1} & \cdots & g_{1,n-1} \\ \vdots & \vdots & \ddots & \vdots \\ g_{k-1,0} & g_{k-1,1} & \cdots & g_{k-1,n-1} \end{bmatrix} \quad \text{式（3-3）}$$

和每个线性分组码相联系的还有一种有用的矩阵，对于任意有 k 个线性独立行的 $k \times n$ 矩阵 G，存在有一个具有 n-k 行线性独立的 $(n-k) \times n$ 阶矩阵 H，它使得 G 的行空间中的任意向量都和 H 的行正交，且与 H 的行正交的任意向量都在 G

的行空间中。因此可以用另一种方法来描述有 G 生成的（n，k）线性码：一个 n 维向量 C 是 G 生成的码字中的码字，其充要条件为：

$$C \bullet H^T = O^T \qquad \text{式（3-4）}$$

此时 H 称为一致校验矩阵。一般情况下，一个（n，k）码的 H 矩阵可表示为：

$$H = \begin{bmatrix} h_{1,n-1} & h_{1,n-2} & \cdots & h_{1,0} \\ h_{2,n-1} & h_{2,n-2} & \cdots & h_{2,0} \\ \vdots & \vdots & \ddots & \vdots \\ h_{n-k,n-1} & h_{n-k,n-2} & \cdots & h_{n-k,0} \end{bmatrix} \qquad \text{式（3-5）}$$

则式（3-4）表示成

$$[c_0, c_1, \cdots, c_{n-1}] \begin{bmatrix} h_{1,n-1} & h_{2,n-1} & \cdots & h_{n-k,n-1} \\ h_{1,n-2} & h_{2,n-2} & \cdots & h_{n-k,n-2} \\ \vdots & \vdots & \ddots & \vdots \\ h_{1,0} & h_{2,0} & \cdots & h_{n-k,0} \end{bmatrix} = 0 \qquad \text{式（3-6）}$$

因 G 中的每一行及其线性组合均为（n，k）码的一个码字，所以由式（3-4）可知

$$G \bullet H^T = O \qquad \text{式（3-7）}$$

2.1.3 系统码

对于线性分组码的码字，希望具有图 3-1 所示的系统结构。其码字划分成两部分，即消息部分和冗余校验部分。信息部分由 k 个未变化的信息数字组成，冗余校验部分有 n-k 为一致校验数字组成，是信息数字的线性组合。有这种结构的线性分组码称为线性系统分组码。

图 3-1 码字的系统形式

因此系统的生成矩阵为：

$$G = [I_k \quad P] \qquad \text{式（3-8）}$$

其中，P 是$k\times(n-k)$阶矩阵；I_k是 k 阶单位阵。如果信息位不在码字的前 k 位，而在码字的后 k 位，则 G 矩阵的I_k单位阵在 P 矩阵的右边。

若（n，k）线性码生成矩阵为式（3-8）的系统形式，则一致校验矩阵 H 可取如下形式

$$H=[-P^T \quad I_{n-k}] \qquad \text{式（3-9）}$$

其中$-P^T$是一个$(n-k)\times k$阶矩阵，它是 P 矩阵的转置，“-”号表示P^T矩阵中的每一元素是 P 矩阵中对应元素的逆元，在二进制情况下，仍是该元素自己。由此得到的 H 矩阵满足

$$G\bullet H^T=[I_k \quad P]\begin{bmatrix}-P\\ I_{n-k}\end{bmatrix}=O \qquad \text{式（3-10）}$$

2.1.4 LDPC 码的描述和图模型表达

LDPC 码是一类线性分组码，由它的校验矩阵来定义，设码长为 N，信息位为 K，校验位为 M=N-K，码率为 R=K/N，则码校验矩阵 H 是一个$M\times N$的矩阵。

二元 LDPC 码的校验矩阵 H 矩阵需要满足以下四个条件：

（1） H 矩阵的每行有ρ个“1”；

（2） H 矩阵的每列有γ个“1”；

（3） H 矩阵的任意两行（或两列）间共同为“1”的个数不超过 1；

（4） 与码长和 H 矩阵的行数相比较，ρ和γ很小，即矩阵中很少一部分元素非零，其他大部分元素都是零，LDPC 码的校验矩阵是稀疏矩阵。

满足以上四个条件的 H 校验矩阵对应的 LDPC 码一般表述为(N,γ,ρ)，码率则为$R\geq 1-\gamma/\rho$。LDPC 码的校验矩阵对应可用一个双向图表示，图的下边有 N 个节点，每个节点表示码字的信息位，称为信息节点$\{x_j, j=1,2,\cdots,N\}$，是码字的比特位，对应于校验矩阵各列，信息节点也称为变量节点；上边有 M 个节点，每个节点表示码字的一个校验集，称为校验节点$\{z_i, i=1,2,\cdots,M\}$，代表校验方程，

对应于校验矩阵各行；与校验矩阵中“1”元素相对应的左右两节点之间存在连接边。将这条边两端的节点称为相邻节点，每个节点相连的边数称为该节点的度数，每个信息节点与γ个校验节点相连，称该变量节点的度数为γ，每个校验节点与ρ个信息节点相连，称该校验节点的度数为ρ。(10, 2, 4)LDPC 码的校验矩阵和双向图如表 3-1 和图 3-2 所示：

表 3-1 （10，2, 4）LDPC 码校验矩阵

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
z_1	1	1	1	1	0	0	0	0	0	0
z_2	1	0	0	0	1	1	1	0	0	0
z_3	0	1	0	0	1	0	0	1	1	0
z_4	0	0	1	0	0	1	0	1	0	1
z_5	0	0	0	1	0	0	1	0	1	1

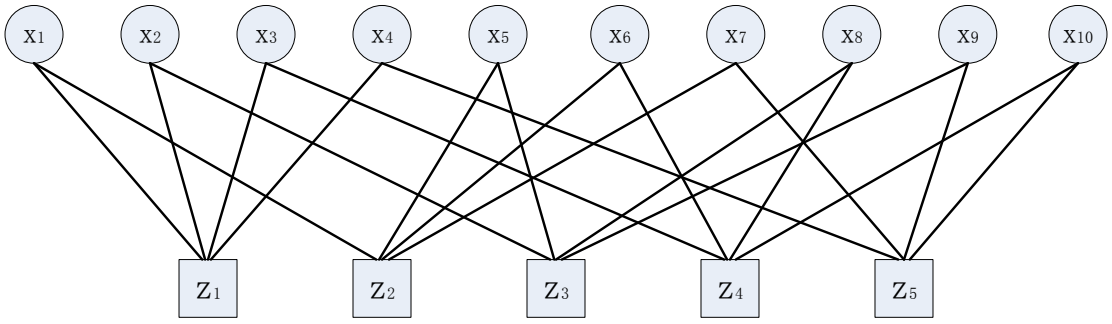

图 3-2 （10,2,4）LDPC 码的双向图

一般情况下校验矩阵是随机构造的，因而是非系统化的。在编码时，对校验矩阵 H 进行高斯消去，可得

$$H = [I \quad P] \qquad \text{式（3-11）}$$

由式（3-11）得生成矩阵

$$G = [-P^T \quad I] \qquad \text{式（3-12）}$$

若 LDPC 校验矩阵的每行（列）中“1”的个数是相等的，则是规则的 LDPC 码；如果检验矩阵的每行（列）中“1”的个数不相等，则称这种 LDPC 码为不规

则 LDPC 码，不规则 LDPC 码一般表述为（N,K）。

2.2 LDPC 码的编码

对 LDPC 码编码方法的研究主要分两步进行，首先是奇偶校验矩阵的构造，然后是基于奇偶校验矩阵的编码算法。

2.2.1 校验矩阵的构造

LDPC 码校验矩阵的构造方法分为两大类：一类是随机构造方法，其校验矩阵与生成矩阵不规则，使编码复杂度高，主要包括 Gallager 构造法、半随机构造法和 PEG 构造法等；另一类是结构化构造法，它是由几何、代数和组合设计等方法构造，其校验矩阵具有某种特殊的结构，因而硬件实现极其简单，主要包括准循环构造法。

1. Gallager 构造方法

为了构造 Gallager LDPC 码的奇偶校验矩阵 H，首先需要构造一个 $k\times k\rho$ 的子矩阵 H_1，其列重为 1，行重为 ρ。然后需要找到 $\gamma-1$ 中置换方式，用来对 H_1 进行列置换，生成 $\gamma-1$ 个 $k\times k\rho$ 子矩阵 $H_2,H_3,\cdots,H_\gamma$。通过这 γ 个子矩阵，尅采用如下方式构成奇偶校验矩阵 H:

$$H=\begin{bmatrix}H_1\\H_2\\\vdots\\H_\gamma\end{bmatrix}$$

H 的列重和行重分别为 γ 和 ρ。对于很大的 k，H 是一个稀疏矩阵。$\gamma-1$ 个置换的选择必须满足由奇偶校验矩阵 H 生成的码具有很好地最小距离性能，同时其泰纳图中不包含长度很短的环，尤其是不包含长度为 4 的环。

采用这种方法，校验矩阵的构造简单，而且容易控制校验矩阵的行重和列重，只需先根据码长和行重构造一个子矩阵，就可以利用此子矩阵和行列置换得到所需的校验矩阵。由于随机构造的原因，在码长较短时，构造的校验矩阵所对应的 Tanner 图中可能会出现较短的环，影响译码性能。

2.半随机构造方法

校验矩阵 H 的半随机构造方法可以概括为以下几个步骤

(1) 首先生成一个全 0 矩阵，然后随机地往每列插入 ρ 个 1；

(2) 调整行重，使行重尽量保持一致；

(3) 调整列中 1 的位置，使得相邻两列 1 的位置在行上不重叠；

(4) 消除矩阵中的短循环。

这种算法的运算量比较大，并且构造出的矩阵不一定是规则的，还需要进一步处理使行重尽可能达到一致。与 Gallager 构造方法类似，半随机构造方法在码长较短时，出现长度为 4 的环的概率较大，所以必须反复地扫描校验矩阵 H，直到所有的长度为 4 的环都被消除。

3.PEG 构造方法

PEG(Progressive Edge-Growth)构造方法是一种构造校验矩阵的贪婪算法，它每次向 Tanner 图中添加一条边，并尽可能保持大的环长(growth)。假设在 Tanner 图上前 i–1 个变量节点的边已经构造出来，在构造第 i 个变量节点的边时，要使新添加的边经过第 i 个变量节点的最小环长最大化。

PEG 方法不仅适用于规则 LDPC 码的构造，也适用于非规则 LDPC 码的构造。在 Tanner 图中，由于短环的存在会降低译码性能，而利用 PEG 方法构造的 Tanner 图使得最小环长最大化，即最大程度地消除短环，所以利用 PEG 方法构造的 LDPC 码通常比利用 Gallager、半随机方法等构造的 LDPC 码性能优异，但这种方法要对节点的度分布进行优化，导致复杂度增加。

4.准循环构造法

准循环 LDPC 码的校验矩阵由一些零矩阵和循环置换单位子矩阵构成。对于准循环 LDPC 码只需存储循环置位单位子矩阵第一行中“1”元素的位置以及循环置换单位子矩阵在校验矩阵中的位置，所需要的存储量大为减少。而且采用代数或组合理论构造的准循环(Quasi-Cyclic)LDPC 码具有循环码的一些特性，由于准循环码构造特殊，在编码时可以采用移位寄存器来实现输入信息与生成矩阵的乘法运算，极大地降低了编码复杂度。

循环码是线性分组码的一个重要子类，利用它的固有代数结构，可以找到简单实用的译码算法。准循环 LDPC 码(QC-LDPC)是 LDPC 码中一个非常重要的子类，同时又是循环码的推广。QC-LDPC 码性能优异，编码复杂度低，已成为 LDPC 码研究领域的一个热门方向。我国的数字电视地面广播标准(DTTB)中的前向纠错编码中就采用了 QC-LDPC 码作为其内码。

2.2.2 编码算法

在传统编码过程中，一般生成矩阵是必需的。尽管 LDPC 码的奇偶校验矩阵是非常稀疏的，但其生成矩阵的稀疏性却无法保证，这样就会导致编码的运算和存储复杂性大大增加；而且如果通过行列变换的方式将稀疏奇偶校验矩阵 H 转换成生成矩阵 G，再根据 G 来进行编码，运算复杂度为$O(n^2)$，将不具有使用性。这样就出现了一些新的编码方法，不再产生生成矩阵，直接利用校验矩阵进行编码以获得低的编码复杂度。

LU 分解编码算法：

首先将$m\times n$校验矩阵 H 写成如下形式：

$$H=[A\ \ B]$$

经过高斯消元后得到如下形式：

$$H_1=\begin{bmatrix} I & H_2 \end{bmatrix}$$

其中I的尺寸为$m\times m$，H_2的尺寸为$m\times k$。设编码后的码字向量为 c，它的长度为 n，把它写成如下形式：

$$c=[s\ \ p]$$

其中 s 是信息码向量，长度为 k，p 为校验码向量，长度为 m。根据校验等式有：

$$H\bullet c^T=0$$

将 H 消元后的矩阵带入上式得：

$$[I\quad H_2]\begin{bmatrix} s^T \\ p^T \end{bmatrix}=0$$

展开该方程，并考虑在 GF(2)上运算有：

$$s^T=H_2\bullet p^T$$

如果校验矩阵 H 是非奇异，则H_2满秩，所以有：

$$p=s\bullet H_2^{-T}$$

2.3 LDPC 码译码

Gallager 在其博士论文中最初提出了两种译码算法：硬判决位翻转译码算法和软判决概率译码算法。硬判决位翻转译码算法的复杂度低，但不能达到 LDPC 码的最佳性能，适用于对译码性能要求不高的场合；软判决概率译码（BP）算法具有非常优异的译码性能，但其译码复杂度高。硬判决：解调器首先对调制器输入符号做最佳判决，然后将硬判决结果送给译码器，译码器再将编码器输入消息做最佳判决，纠正解调器可以发生的错误判决。（经过解调器对符号的硬判决，丢失了很多有利于译码的信息）软判决：解调器对输出不进行判决，送到译码器的是判决符号可能的概率值或未量化输出，而非硬判决值，则译码器就可以利用这些信息与编码信息综合做出判决，从而提高系统性能。

软判决概率译码算法是一种基于置信传播(Belief Propagation)的迭代译码算法，简称 BP 算法，也称和积算法(sum-product algorithm，SPA)，BP 算法可以用基于 Tanner 图的消息传递(Message Passing，MP)算法来实现。BP 译码的每一次迭代过程都包含两个方面：校验节点和变量节点的处理，在每次的迭代中，每一个的校验节点都从与之相邻的变量节点处收到消息，处理后，再重新回到原来的变量节点；然后再对所有的变量节点进行同样的处理；最后对变量节点收集来的数据进行判决。

MP 算法中所用到的符号表示的含义：

H_{ij} 表示 LDPC 码的校验矩阵 H 中第 i 行第 j 列的元素；

$N(m)$ 表示与校验节点 s_m 相连的所有变量节点集合，即 $N_m \equiv \{x_n : H_{mn} = 1\}$；

$M(n)$ 表示与变量节点 x_n 相连的所有校验节点集合，即 $M(n) \equiv \{s_m : H_{mn} = 1\}$；

$N(m) \backslash n$ 表示变量节点集合 $N(m)$ 中去掉变量节点 x_n；

$M(n) \backslash m$ 表示校验节点集合 $M(n)$ 中去掉校验节点 s_m。

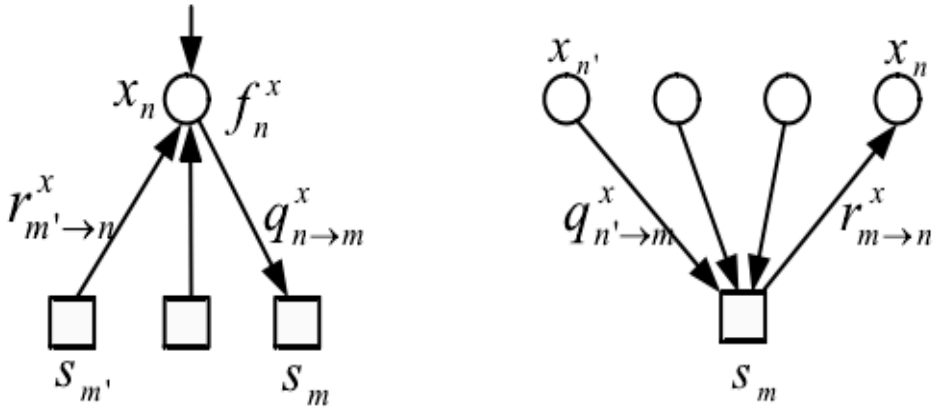

图 3-3 译码过程中消息传递关系图

图3-3 说明了在译码过程中 $q_{n\to m}^x$ 与 $r_{m\to n}^x$ 的消息传递关系。$f_n^x = P(x_n = x \mid y)$ 表示根据接收信号值 y 和信道特性得出的比特 x_n 取值为 x 的概率，其中 $x \in \{0,1\}$，所以有 $f_n^0 + f_n^1 = 1$， f_n^x 可以看成是变量节点 x_n 的先验概率。$q_{n\to m}^x$ 表示基于校验节点集合 $M(n)\backslash m$ 得出的第 n 个变量节点 $x_n = x$ 的概率，其中 $x \in \{0,1\}$，所以有 $q_{n\to m}^o + q_{n\to m}^1 = 1$，$q_{n\to m}^x$ 可看成是变量节点 x_n 向校验节点 s_m 传递的消息。$r_{m\to n}^x$ 表示在码字中已知 $x_n = x$ 并给定其他比特的一组概率分布 $\{q_{n'\to m}^x : n' \in N(m)\backslash n\}$ 时，第 m 个校验方程成立的条件概率。$r_{m\to n}^x$ 可看成是校验节点 s_m 向变量节点 x_n 传递的消息。图 3-4 说明了概率域 BP 译码算法的流程图。

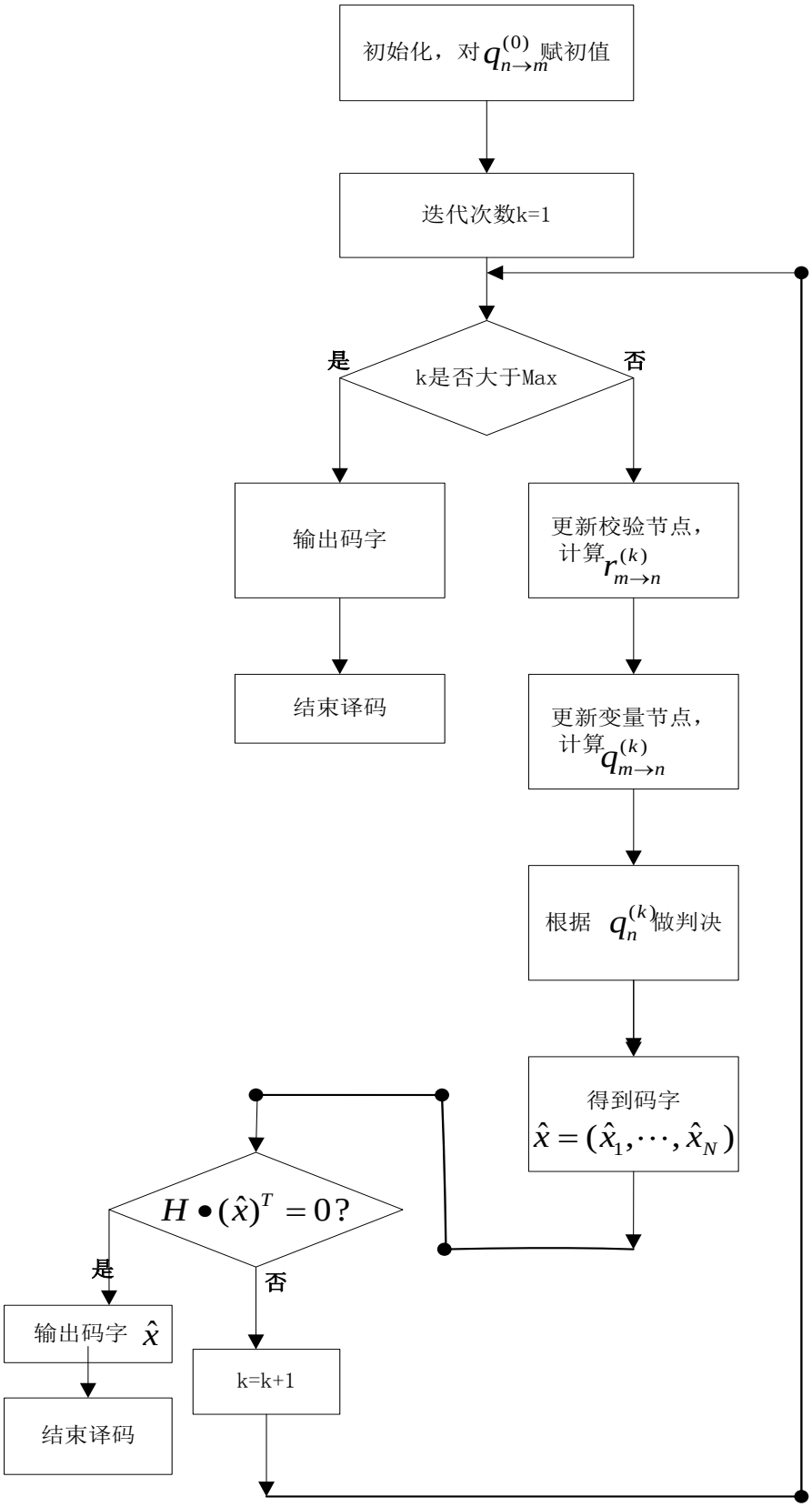

图 3-4 概率域 BP 译码算法的流程图

2.3.1 BP 译码算法

BP 算法的步骤如下：

（1） 初始化：

$$q_{n\to m}^{0} = f_n^{0} \qquad \text{式（3-13）}$$

$$q_{n\to m}^{1} = f_n^{1} \qquad \text{式（3-14）}$$

（2）校验节点更新

设 $\delta q_{n\to m} = q_{n\to m}^{0} - q_{n\to m}^{1}$，对每个校验节点 $m = 1,\cdots,M$ 和 $n \in N(m)$，计算

$$\delta r_{m\to n} = \prod_{n'\in N(m)\backslash n} \delta q_{n'\to m} \qquad \text{式（3-15）}$$

$$r_{m\to n}^{0} = \frac{1}{2}\left(1+\delta r_{m\to n}\right) \qquad \text{式（3-16）}$$

$$r_{m\to n}^{1} = \frac{1}{2}\left(1-\delta r_{m\to n}\right) \qquad \text{式（3-17）}$$

（3）变量节点更新

对每个变量节点 $n = 1,\cdots,N$ 和 $m \in M(n)$，计算

$$q_{n\to m}^{0} = \alpha_{n\to m} f_n^{0} \prod_{m'\in M(n)\backslash m} r_{m'\to n}^{0} \qquad \text{式（3-18）}$$

$$q_{n\to m}^{1} = \alpha_{n\to m} f_n^{1} \prod_{m'\in M(n)\backslash m} r_{m'\to n}^{1} \qquad \text{式（3-19）}$$

其中，$\alpha_{n\to m}$ 是归一化系数，使得 $q_{n\to m}^{0} + q_{n\to m}^{1} = 1$。

（4）计算第 n 为比特 x_n 取值为 0 或 1 的伪后验概率 q_n^{0} 和 q_n^{1}：

$$q_n^{0} = \alpha_n f_n^{0} \prod_{m\in M(n)} r_{m\to n}^{0} \qquad \text{式（3-20）}$$

$$q_n^{1} = \alpha_n f_n^{1} \prod_{m\in M(n)} r_{m\to n}^{1} \qquad \text{式（3-21）}$$

选择适当的归一化系数 α_n，使得 $q_n^{0} + q_n^{1} = 1$。

（5） 译码判决：

对每一位比特 $x_n (n = 1,2,\cdots N)$，如果 $q_n^{0} > 0.5$ 是，则 $\hat{x}_n = 0$；否则 $\hat{x}_n = 1$，得到码字 $\hat{x} = (\hat{x}_1,\cdots,\hat{x}_N)$。如果 $H \bullet (\hat{x})^{T} = 0$,说明译码成功，则停止译码，并将 $\hat{x}$ 的值作为译码结果输出；否则转步骤（2）继续迭代，如果迭代次数已达到事先设

定的最大迭代次数还未成功译码，则译码结束并宣告译码失败，将最后一次迭代的 $\hat{x}$ 值作为输出。

2.3.2 对数域 BP 译码算法

如果将概率消息用对数似然比（Log-Likelihood Ratio，LLR）表示，则可以由概率域 BP 算法得到对数域 BP 译码算法。在对数域中进行译码有两个优点：一是可以将概率域中大量的乘法运算法转化为对数域中的加法运算，减少了运算时间；二是省去了系数归一化的步骤，提高了计算的稳定性。

定义对数似然比 $L(x)=\ln\frac{p(x=0)}{p(x=1)}$，其中 $p(x=0)$ 和 $p(x=1)$ 分别是随机变量 x 取值为 0 和 1 的概率。由概率域 BP 算法中，令：

$$\lambda_n=\ln\frac{q_n^0}{q_n^1},\quad \lambda_{mn}=\ln\frac{q_{n\to m}^0}{q_{n\to m}^1},\quad \Lambda_{mn}=\ln\frac{r_{n\to m}^0}{r_{n\to m}^1} \qquad \text{式（3-22）}$$

$\lambda_n^{(k)}$，$\lambda_{mn}^{(k)}$，$\Lambda_{mn}^{(k)}$ 分别表示在第 k 次迭代过程中 λ_n，λ_{mn} 和 Λ_{mn} 的值，即 $\lambda_n^{(k)}$ 表示第 n 位输入数据的后验对数似然比，$\lambda_{mn}^{(k)}$ 表示变量节点 n 向校验节点 m 传递的消息，$\Lambda_{mn}^{(k)}$ 表示校验节点 m 向变量节点 n 传递的消息。由概率域 BP 算法可以推导出对数域 BP 算法，对数域的 BP 算法如下：

（1）对 $n=1,2,\cdots,N$，进行初始化：

$$\lambda_{mn}^{(0)}=\lambda_n^0, m\in M(n) \qquad \text{式（3-23）}$$

其中 λ_n^0 是经过信道后接收到的初始对数似然比，$\lambda_n^0=\ln\frac{p(x_i=0\setminus y_i)}{p(x_i=1\setminus y_i)}$。

（2）校验节点更新：

对每个校验节点 m 和 $n\in N(m)$，计算

$$\Lambda_{mn}^n=2\tanh^{-1}\Big(\prod_{n'\in N(m)\setminus n}\tanh\frac{\lambda_{mn'}^{(k-1)}}{2}\Big) \qquad \text{式（3-24）}$$

（3）变量节点更新：

对每个变量节点 n 和 $m\in M(n)$，计算：

$$\lambda_{mn}^{(k)} = \lambda_n^{(0)} + \sum_{m' \in M(n) \backslash m} \Lambda_{m'n}^{(k)} \qquad \text{式（3-25）}$$

对每个变量节点，计算后验对数似然比

$$\lambda_n^{(k)} = \lambda_n^{(0)} + \sum_{m \in M(n)} \Lambda_{mn}^{(k)} \qquad \text{式（3-26）}$$

（4）译码判决

一次迭代完成后，进行译码判决。如果 $\lambda_n^{(k)} \geq 0$ 时， $\hat{x}_n^{(k)} = 1$，否则 $\hat{x}_n^{(k)} = 1$。由此可以得到关于译码码字的一个估计值 $\hat{x}_n^{(k)}$，如果 $H \bullet (\hat{x}^{(k)})^T = 0$，则译码成功，结束译码，并将 $\hat{x}^{(k)}$ 作为有效输出值；否则转步骤（2）继续迭代，指导达到预定的最大迭代次数。

根据函数 $\tanh(x)$ 和 $\tanh^{-1}(x)$ 的定义

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}, \quad \tanh^{-1}(x) = \frac{1}{2} \ln \frac{1+x}{1-x} \qquad \text{式（3-27）}$$

可以将式（3-25）等价变形为：

$$\Lambda_{mn}^{(k)} = \left(\prod_{n' \in N(m) \backslash n} sign(\lambda_{mn'}^{(k-1)}) \right) \bullet \Phi^{-1} \left(\sum_{n' \in N(m) \backslash n} \Phi \left(|\lambda_{mn'}^{(k-1)}| \right) \right) \qquad \text{式（3-28）}$$

其中 $sign(x)$ 和 $\Phi（x）=\Phi^{-1}(x) = \ln\left(\frac{e^x+1}{e^x-1}\right)$ 的定义分别为

$$sign(x) = \begin{cases} +1 & x \geq 0 \\ -1 & x < 0 \end{cases} \qquad \text{式（3-29）}$$

$$\Phi（x）=\Phi^{-1}(x) = \ln\left(\frac{e^x+1}{e^x-1}\right) \qquad \text{式（3-30）}$$

可以用式（3-24）对校验节点进行更新，$\Phi（x）$是一个非线性函数，和它的反函数 $\Phi^{-1}(x)$ 相同，可以用查表的方法实现 $\Phi（x）$和 $\Phi^{-1}(x)$ 的计算。

对数域 BP 算法中，另一种校验节点更新的方法：二元函数 $f(x,y) = 2\tanh^{-1}\left(\tanh\frac{x}{2}\tanh\frac{y}{2}\right)$，由于：

$$f(x,y) = 2\tanh^{-1}\left(\tanh\frac{x}{2}\tanh\frac{y}{2}\right) = \ln\frac{1+e^{x+y}}{e^x+e^y} \qquad \text{式（3-31）}$$

根据雅可比对数定理

$$\ln(e^{x}+e^{y})=\max\{x,y\}+\ln(1+e^{-|x-y|}) \quad \text{式（3-32）}$$

对式（3-31）运用两次雅可比对数定理可得：

$$\begin{aligned} f(x,y)&=\max\{0,x+y\}+\ln(1+e^{-|x+y|})-\max\{x,y\}-\ln(1+e^{-|x-y|}) \\ &=sign(x)\,si\,gn(y)\min(|x|,|y|)+\ln(1+e^{-|x+y|})-\ln(1+e^{-|x-y|}) \end{aligned} \quad \text{式（3-33）}$$

在对数域 BP 算法中的式（3-25）可看成是多元函数求解，它可以转化成二元函数，再根据式（3-33）递归地进行计算：

$$\Lambda_{mn}^{(k)}=\underset{n'\in N(m)\backslash n}{g}(\lambda_{mn'}) \quad \text{式（3-34）}$$

其中 $g(a,b)=sign(a)\times si\,gn(b)\times\min(|a|,|b|)+LUT_g(a,b)$ 式（3-35）

且 $LUT_g(a,b)=\ln(1+e^{-|a+b|})-\ln(1+e^{-|a-b|})$ 式（3-36）

根据上面三个式子对校验节点进行更新。

2.3.3 最小和算法

在校验节点的更新步骤中，最下和算法根据式（3-34）进行计算，但在计算时省略了查找表操作 $LUT_g(a,b)$，因此校验节点的更新步骤变为

$$\Lambda_{mn}^{(k)}=\prod_{n'\in N(m)\backslash n}sign(\lambda_{mn'}^{(k-1)})\bullet\min_{n'\in N(m)\backslash n}|\lambda_{mn'}^{(k-1)}| \quad \text{式（3-37）}$$

在对数域 BP 算法中，根据式（3-25）和式（3-26），可以将校验节点更新的计算改写为：

$$\lambda_{mn}^{(k)}=\lambda_{n}^{(k)}-\Lambda_{mn}^{(k)} \quad \text{式（3-38）}$$

利用式（3-38），并在对数域 BP 算法的基础上作些等价变形，可以得到最小和算法的另一种表述：

（1） 初始化：

$$\Lambda_{mn}^{(0)}=0 \quad \text{式（3-39）}$$

（2） 迭代过程

$$\lambda_{n}^{(k)}=\lambda_{n}^{(0)}$$

$$\Lambda_{mn}^{(k)}=\prod_{n'\in N(m)\backslash n}sign(\lambda_{n}^{(k-1)}-\Lambda_{mn'}^{(k)})\bullet\min_{n'\in N(m)\backslash n}|\lambda_{n}^{(k-1)}-\Lambda_{mn'}^{(k)}|$$

$$\lambda_n^{(k)} = \lambda_n^{(k)} + \Lambda_{mn}^{(k)}$$

（3）译码判决：如果 $\lambda_n^{(k)} \geq 0$，$\hat{x}_n^{(k)} = 0$，否则 $\hat{x}_n^{(k)} = 1$。当所有校验方程都满足或达到大迭代次数时停止译码，否则转步骤（2）继续迭代。

小和算法将变量节点更新和校验节点更新两个步骤融合到一块，简化了迭代过程的计算，$\min_{n' \in N(m) \backslash n} |\lambda_n^{(k-1)} - \Lambda_{mn'}^{(k)}|$只有两个值：最小值和次小值，只需存储最小值、次小值和最小值所对应的下标，并存储 $\lambda_n^{(k-1)} - \Lambda_{mn'}^{(k)}$ 的符号位，简化了中间结果的存储。

限于篇幅，MATLAB 代码在此省略，如有需要，联系：ilynchpin@ilycnpin.com.cn

临菲信息技术港

临菲信息技术港 公众号

临菲学堂