第 5 章

CHAPTER 5

信道编码理论

第 4 章描述了信源编码理论，包括无失真信源编码和限失真信源编码。但是一般信道中总存在噪声和干扰，这些干扰和噪声会造成信息损失。那么，如何保证在有噪信道中可靠地进行信息传输？怎样才能使消息通过传输后发生的错误最少？传输可达的最大信息传输率又是多少？这些问题属于通信的可靠性问题，是本章要研究的主要内容。

信道编码包括线路编码和纠错编码。由于信号在信道传输时受到干扰，信号码元波形将发生变形，传输到接收端时可能发生错误判断，因而需采用线路编码使之更适合于信道特性或满足接收端对恢复信号的要求，减少信息的损失。另外，若少量错误已经发生，则可通过纠错编码恢复信息。本章主要研究纠错编码，其过程是通过有选择地在数据中引入冗余，以最小冗余的代价获得最佳抗干扰性能，纠正数据传输过程中出现的错误。通常，我们将用于检测差错的编码称为检错码，将可以检测和校正错误的编码称为纠错码。

5.1 最佳译码准则

由于信道中存在噪声，因而在有噪信道中传输消息将发生错误。为了减少错误的发生和提高信息传输的可靠性，首先需要分析这些错误与哪些因素有关，有没有办法控制这些错误的发生，以及能控制到什么程度。

很明显，传输信息错误的多少与信道的特性有关，如信噪比较高的信道产生的错误较少。除此之外，错误的产生还与编译码过程有关。图 5-1 给出了通信系统的编译码过程。

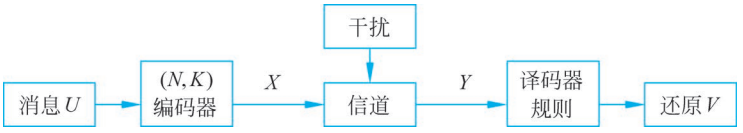


图 5-1 编译码过程

设有噪离散信道的输入符号集为 $X=\{a_1,a_2,\cdots,a_n\}$，输出符号集为 $Y=\{b_1,b_2,\cdots,b_m\}$，信道的传递概率为 $p(Y|X)=\{p(b_j|a_i),i=1,2,\cdots,n;\ j=1,2,\cdots,m\}$。由于噪声的干扰，信道输入符号 $a_i(i=1,2,\cdots,n)$时，信道的输出是 $a_i(i=1,2,\cdots,n)$或者是该符号的变形 $b_j(j=1,2,\cdots,m)$。因此，为了达到通信的目的，必须制定译码规则，也就是设计函数 $F(b_j)=a_i(i=1,2,\cdots,n;\ j=1,2,\cdots,m)$，使它的每个输出符号有且仅有唯一的输入符号与之相对应，该函数被称为译码函数。由于每个输出符号都可译成输入符号中的任何一个，

所以共有 n^m 种译码规则。

如果确定了译码规则，那么，当信道输出符号是 $b_j(j=1,2,\cdots,m)$ 时，则一定可以按译码规则译成相应的符号 $F(b_j)=a_i(i=1,2,\cdots,n;\ j=1,2,\cdots,m)$。当信道的输入符号是 a_i 时，显然是正确译码；如果不是 a_i，则出现译码错误。由此，在信道输出端收到符号 $b_j(j=1,2,\cdots,m)$ 后，正确译码的概率 p_j^{r} 就应该是在信道输出端出现 $b_j(j=1,2,\cdots,m)$ 的前提下，推测信道输入符号为 a_i 的后验概率，即

$$p_j^{\mathrm{r}}=p\{\{X=F(b_j)=a_i\}\mid\{Y=b_j\}\} \tag{5.1.1}$$

同时，在信道输出端收到某符号 $b_j(j=1,2,\cdots,m)$ 后，错误译码的概率是 p_j^{e} 就应该是在信道输出端出现 $b_j(j=1,2,\cdots,m)$ 的前提下，推测信道输入符号是除了 a_i 以外的其他任何可能的输入符号的后验概率，即

$$p_j^{\mathrm{e}}=p\{\{X=\varepsilon\}\mid\{Y=b_j\}\} \tag{5.1.2}$$

其中，ε 表示除了 $F(b_j)=a_i$ 以外的其他可能的输入，显然，式(5.1.2)可改写为

$$p_j^{\mathrm{e}}=1-p_j^{\mathrm{r}}=1-p\{\{F(b_j)=a_i\}\mid b_j\} \tag{5.1.3}$$

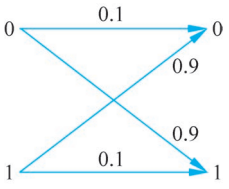


图 5-2 二元对称信道

【例 5-1】 设有一个二元对称信道，如图 5-2 所示，其输入符号为等概率分布，求译码差错概率的大小。

解：针对译码规则一：接收到符号 0，译成发送的符号为 0；接收到符号 1，译成发送的符号为 1。因此，当发送符号 0 时，正确译码的概率为 0.1，译错的概率为 0.9；当发送符号 1 时，正确译码的概率为 0.1，译错的概率为 0.9。那么，在输入符号是等概率的情况下，平均错误概率为

$$p_{\mathrm{e}}=\frac{1}{2}(p_0+p_1)=\frac{1}{2}(0.9+0.9)=0.9$$

针对译码规则二：接收到符号 1，译成发送的符号为 0；接收到符号 0，译成发送的符号为 1。因此当发送符号 0 时，正确译码的概率为 0.9，译错的概率为 0.1；当发送符号 1 时，正确译码的概率为 0.9，译错的概率为 0.1。那么，在输入符号是等概率的情况下，平均错误概率为

$$p_{\mathrm{e}}=\frac{1}{2}(p_0+p_1)=\frac{1}{2}(0.1+0.1)=0.1$$

由此可见，错误概率既与信道的统计特性有关，也与译码规则有关。在例 5-1 中，输入符号有 0、1 两种，输出符号也有 0、1 两种，所以可以构成 $n^m=2^2=4$ 种译码规则，分别如下。

(1) $\begin{cases}F(0)=0\\F(1)=1\end{cases}$；

(2) $\begin{cases}F(0)=1\\F(1)=0\end{cases}$；

(3) $\begin{cases}F(0)=0\\F(1)=0\end{cases}$；

(4) $\begin{cases}F(0)=1\\F(1)=1\end{cases}$。

译码规则(1)表示信道输出端出现 0，则译为 0；信道输出端出现 1，则译为 1。

译码规则(2)表示信道输出端出现0,则译为1;信道输出端出现1,则译为0。

译码规则(3)表示不论信道输出端出现0还是1,都译为0。

译码规则(4)表示不论信道输出端出现0还是1,都译为1。

很明显,不论采用哪一种译码规则,都有可能将输入端的0码在输出端译成1,或者将1译成0。那么,关键问题是选择哪种译码规则可以使平均错误概率最小。

对于通信系统而言,不仅要描述个别的输出符号的正确或错误译码概率,更需要描述在信道的输出端,每收到一个符号的平均正确译码或平均错误译码的概率,即在信道输出端每收到一个符号时,发生正确译码或错误译码的可能性的大小。

在信道输出端,每收到一个Y符号的平均正确译码概率p_r,应该是式(5.1.1)所示的后验概率$p_j^r(j=1,2,\cdots,m)$在信道输出随机变量Y概率空间中的统计平均值,即

$$p_r=\sum_{j=1}^{m}p(b_j)p_j^r=\sum_{j=1}^{m}p(b_j)p\{\{F(b_j)=a_i\}\mid b_j\} \tag{5.1.4}$$

同样,在信道输出端,每收到一个Y符号的平均错误译码概率p_e,应该是式(5.1.3)所示的后验概率$p_j^e(j=1,2,\cdots,m)$在信道输出随机变量Y概率空间中的统计平均值,即

$$\begin{aligned}p_e&=\sum_{j=1}^{m}p(b_j)p_j^e=\sum_{j=1}^{m}p(b_j)\{1-p[(F(b_j)=a_i)\mid b_j]\}\\&=\sum_{j=1}^{m}p(b_j)-\sum_{j=1}^{m}p(b_j)p[F(b_j)=a_i\mid b_j]\\&=1-\sum_{j=1}^{m}p(b_j)p[F(b_j)=a_i\mid b_j]\end{aligned} \tag{5.1.5}$$

平均错误译码概率p_e表示在信道输出端平均每收到一个符号时,产生错误译码的可能性的大小;p_e越大,表示在信道输出端平均每收到一个符号产生错误译码的可能性越大,也就是通信的可靠性越差。所以,p_e可以作为衡量通信可靠性的指标。

综上,平均错误译码概率p_e取决于变量Y的概率空间、信道的后验概率分布和预先设定的译码规则。因此,对于给定信道,其信道的转移概率是确定的,当信道的输入符号(即信源的输出符号的概率分布)确定后,不同的译码规则就有不同的平均错误译码概率p_e,合适的译码规则可以降低平均错误译码概率,因此,制定合适的译码规则成为提高通信可靠性的一种有效手段。

那么,按什么准则来选择译码规则,可使其平均错误译码概率p_e达到最小呢?若给定信道的传递概率$p(Y|X)=\{p(b_j|a_i),i=1,2,\cdots,n;\ j=1,2,\cdots,m\}$,且有$0\leqslant p(b_j|a_i)\leqslant 1$,$(i=1,2,\cdots,n;\ j=1,2,\cdots,m)$;$\sum_{j=1}^{m}p(b_j\mid a_i)=1,i=1,2,\cdots,n$。可将给定的$n\times m$个传输概率$p(b_j|a_i)(i=1,2,\cdots,n;\ j=1,2,\cdots,m)$排列成信道矩阵如下:

$$\boldsymbol{P}=\begin{array}{c}\\a_1\\a_2\\\vdots\\a_n\end{array}\begin{array}{c}\begin{array}{cccc}b_1&b_2&\cdots&b_m\end{array}\\\begin{bmatrix}p(b_1\mid a_1)&p(b_2\mid a_1)&\cdots&p(b_m\mid a_1)\\p(b_1\mid a_2)&p(b_2\mid a_2)&\cdots&p(b_m\mid a_2)\\\vdots&\vdots&\ddots&\vdots\\p(b_1\mid a_n)&p(b_2\mid a_n)&\cdots&p(b_m\mid a_n)\end{bmatrix}\end{array} \tag{5.1.6}$$

可得

$$p(a_i \mid b_j)=\frac{p(a_i)p(b_j \mid a_i)}{p(b_j)}=\frac{p(a_i)p(b_j \mid a_i)}{\sum_{i=1}^{n}p(a_i)p(b_j \mid a_i)} \tag{5.1.7}$$

由给定的信道输入 X 的概率分布 $p(X)=\{p(a_1),p(a_2),\cdots,p(a_n)\}$和信道的传递概率 $p(Y|X)=\{p(b_j|a_i),i=1,2,\cdots,n;\ j=1,2,\cdots,m\}$,可以求出信道输出随机变量 Y 的 m 个概率分量为

$$p(b_j)=\sum_{i=1}^{n}p(a_i)p(b_j \mid a_i),\quad j=1,2,\cdots,m \tag{5.1.8}$$

式(5.1.8)表明,对于给定信源和给定信道,后验概率和信道输出随机变量 Y 的概率分布也都是确定的。

现在,我们来看平均错误译码概率,问题是如何选择译码规则 $F(b_j)=a_i,j=1,2,\cdots,m;\ a_i\in\{a_1,a_2,\cdots,a_n\}$,使平均错误译码概率达到最小值。不同的译码规则,不会引起 $p(b_j)(j=1,2,\cdots,m)$和 $p(a_i|b_j)(i=1,2,\cdots,n;\ j=1,2,\cdots,m)$的变化。显然,要使 p_e 最小,必须使式(5.1.9)达到最大,即

$$\max\sum_{j=1}^{m}p(b_j)p\{F(b_j)=a_i \mid b_j\},\quad i=1,2,\cdots,n \tag{5.1.9}$$

因此,必须有

$$\max p\{F(b_j)=a_i \mid b_j\},\quad j=1,2,\cdots,m;\ i=1,2,\cdots,n \tag{5.1.10}$$

它是每一个 $b_j(j=1,2,\cdots,m)$所对应的 n 个后验概率 $p(a_1|b_j),p(a_2|b_j),\cdots,p(a_n|b_j)(j=1,2,\cdots,m)$中最大的一个值。如果把最大值所对应的信道输入符号记作 a^*,则有

$$p(a^* \mid b_j)\geqslant p(a_i \mid b_j),\quad i=1,2,\cdots,n;\ j=1,2,\cdots,m \tag{5.1.11}$$

这就是说,要使平均错误译码概率达到最小,则有

$$p\{F(b_j)=a_i \mid b_j\}=p(a^* \mid b_j)\geqslant p(a_i \mid b_j),\quad i=1,2,\cdots,n;\ j=1,2,\cdots,m \tag{5.1.12}$$

意味着,必须选择译码规则

$$F(b_j)=a^*,\quad j=1,2,\cdots,m \tag{5.1.13}$$

而信源符号满足

$$p(a^* \mid b_j)\geqslant p(a_i \mid b_j),\quad i=1,2,\cdots,n;\ j=1,2,\cdots,m \tag{5.1.14}$$

这就是最大后验概率译码准则。该准则指出:若 $p(a^*|b_j)\geqslant p(a_i|b_j)(i=1,2,\cdots,n;\ j=1,2,\cdots,m)$是信道输出端收到符号 $b_j(j=1,2,\cdots,m)$后,推测信道输入符号 $a_i(i=1,2,\cdots,n)$的 n 个后验概率 $p(a_i|b_j)(i=1,2,\cdots,n;\ j=1,2,\cdots,m)$中最大的一个,即 $p(a^*|b_j)\geqslant p(a_i|b_j)(i=1,2,\cdots,n;\ j=1,2,\cdots,m)$,则把接收符号 $b_j(j=1,2,\cdots,m)$译成 a^*,即 $F(b_j)=a^*,j=1,2,\cdots,m$。用最大后验概率译码准则选择译码函数 $F(b_j)=a^*(j=1,2,\cdots,m)$构成的译码规则,一定可以使平均错误译码概率 p_e 达到最小值,这是因为

$$\begin{aligned}p_e&=\sum_{j=1}^{m}p(b_j)\{1-p[\{F(b_j)=a_i\} \mid b_j]\}\\&\geqslant\sum_{j=1}^{m}p(b_j)\{1-p[\{F(b_j)=a^*\} \mid b_j]\}\\&=\sum_{j=1}^{m}p(b_j)\{1-p(a^* \mid b_j)\}\end{aligned}$$

$$=\sum_{j=1}^{m}p(b_j)-\sum_{j=1}^{m}p(b_j)p(a^*\mid b_j)$$

$$=1-\sum_{j=1}^{m}p(a^*b_j)=\sum_{i=1}^{n}\sum_{j=1}^{m}p(a_ib_j)-\sum_{j=1}^{m}p(a^*b_j)$$

$$=\sum_{j=1}^{m}\sum_{i=1,a_i\neq a^*}^{n}p(a_ib_j)=p_{\mathrm{emin}} \tag{5.1.15}$$

由此可见,最大后验概率译码规则使平均错误译码概率达到最小值。因此,当信源和信道给定,采用最大后验概率译码准则时,可使通信的可靠性最大。

通常,最大后验概率准则中后验概率的计算存在困难,且不方便。但是,由式(5.1.7)可得

$$p(a^*\mid b_j)=\frac{p(a^*)p(b_j\mid a^*)}{p(b_j)},\quad j=1,2,\cdots,m \tag{5.1.16}$$

考虑到 $p(a_i\mid b_j)=\dfrac{p(a_i)p(b_j\mid a_i)}{p(b_j)}$,则在信道输入符号(信源输出符号)$a_i(i=1,2,\cdots,n)$的先验概率为

$$p(a_1)=p(a_2)=\cdots=p(a^*)=\cdots=p(a_n)=\frac{1}{n} \tag{5.1.17}$$

它们与信道的每个输出符号 $b_j(j=1,2,\cdots,m)$的 n 个后验概率 $p(a_i\mid b_j)(i=1,2,\cdots,n;\ j=1,2,\cdots,m)$的大小相比,可以转换为与信道的每个输出符号 $b_j(j=1,2,\cdots,m)$相关的 n 个前向概率(传递概率)$p(b_j\mid a_i)$之间的大小比较,即由

$$p(a^*\mid b_j)=\frac{p(a^*)p(b_j\mid a^*)}{p(b_j)}\geqslant p(a_i\mid b_j)=\frac{p(a_i)p(b_j\mid a_i)}{p(b_j)} \tag{5.1.18}$$

则有

$$p(b_j\mid a^*)\geqslant p(b_j\mid a_i),\quad i=1,2,\cdots,n;\ j=1,2,\cdots,m \tag{5.1.19}$$

所以,在信道输入符号 $a_i(i=1,2,\cdots,n)$的先验概率等概的条件下,最大后验概率准则等价于最大似然译码准则。显然,最大似然译码准则是在信道输入符号先验概率等概的特定条件下的最大后验概率准则,其平均错误译码概率同样可以达到最小值。

由此可见,在信道输入符号先验概率等概的特定条件下,采用最大似然准则所得到的最小平均错误译码概率取决于等概信道输入所含符号数 n 和信道的传递特性 $p(b_j\mid a_i)$,$i=1,2,\cdots,n;\ j=1,2,\cdots,m$。当信道的输入符号数 n 固定不变时,最小平均错误译码概率随信道传递特性 $p(b_j\mid a_i)(i=1,2,\cdots,n;\ j=1,2,\cdots,m)$的变动而变动,这就为我们提供了一个继续降低最小平均错误译码概率并提高通信可靠性的方法,即在信道的输入符号先验概率等概的条件下,如果信道输入符号数固定不变,则可以通过改变信道的传输特性,使最小平均错误译码概率 p_{emin} 进一步下降。实际上,这就是抗干扰信道编码的基本理论依据。

对于无记忆信道情形,码字的似然概率应等于组成该码字的各码元的似然概率的乘积(即联合条件概率等于各符号概率的乘积)。因此,码字的最大似然概率也就是各码元似然概率乘积的最大化,即

$$\max p(\boldsymbol{r}\mid \boldsymbol{C}_i)=\max\prod_{j=1}^{N}p(r_j\mid C_{ij}),\quad i=1,2,\cdots,M \tag{5.1.20}$$

其中,$\boldsymbol{r}$ 是信道输出码字矢量,$\boldsymbol{C}_i$ 是信道第 i 个输入码字矢量,N 为码字矢量的长度,M 是码集中码字矢量的总个数。

为了使乘法运算简化为加法，取似然概率的对数，称为对数似然概率。由于对数函数的单调性，似然概率最大时，对数似然概率也最大化。于是，码字对数似然概率最大化可等效于对数似然概率之和的最大化，即

$$\max \log p(\boldsymbol{r} \mid \boldsymbol{C}_i)=\max \sum_{j=1}^{N} \log p(r_j \mid C_{ij}), \quad i=1,2,\cdots,M \tag{5.1.21}$$

式中的对数可以 e 为底(自然对数)，也可以 2 或 10 为底，M 是码集中码字矢量的总个数。

下面作为一个特例，证明二进制对称信道(BSC)的最大似然译码准则可以简化为最小汉明距离译码准则。

对于 BSC 信道，当逐位比较发送码和接收码时，仅存在相同或不同两种可能，且两种可能发生的概率与信道的错误转移概率有关，为

$$p(r_j \mid C_{ij})=\begin{cases} p, & C_{ij} \neq r_j \\ 1-p, & C_{ij}=r_j \end{cases} \tag{5.1.22}$$

如果 $\boldsymbol{r}$ 中有 d 个码元与 $\boldsymbol{C}_i$ 的码元不同，$\boldsymbol{r}$ 与 $\boldsymbol{C}_i$ 的汉明距离为 d。显然，d 代表在 BSC 信道传输过程中码元差错的个数，即

$$d=\operatorname{dis}(\boldsymbol{r},\boldsymbol{C}_i)=W(\boldsymbol{r} \oplus \boldsymbol{C}_i)=\sum_{j=1}^{N} r_j \oplus C_{ij}, \quad i=1,2,\cdots,M \tag{5.1.23}$$

M 是码集中码字矢量的总个数。此时的似然概率为

$$\begin{aligned} p(\boldsymbol{r} \mid \boldsymbol{C}_i) &= \prod_{j=1}^{N} p(r_j \mid C_{ij}) = p^d (1-p)^{N-d} \\ &= \left(\frac{p}{1-p}\right)^d (1-p)^N, \quad i=1,2,\cdots,M \end{aligned}$$

其中，M 是码集中码字矢量的总个数，$(1-p)^N$ 是常数。由于 $0 \leqslant p \leqslant \frac{1}{2}$，导致 $\frac{p}{1-p} \leqslant 1$，$d$ 越大时，似然概率越小。因此，最大似然函数 $\max p(\boldsymbol{r} \mid \boldsymbol{C}_i)$ 的问题转换为求最小汉明距离 $d_{\min}$ 的问题。

汉明距离译码是一种硬判决译码。只要在接收端将发送码与接收码的各码元逐一比较，选择汉明距离最小的码字作为译码估值，就可以获得最大似然概率译码。由于 BSC 信道是对称的，因此，当发送的码字独立且等概率时，汉明距离译码是一种最佳译码。


视频讲解

5.2 信道编码的基本概念

图 5-3 是简化的通信系统模型，其中信道可以是包括调制、解调和传输媒质在内的有线(含光纤)和无线的数字信道，也可以是包括网桥、路由器、网关、电缆(或光缆)和低层协议等在内的计算机通信网的数字信道。

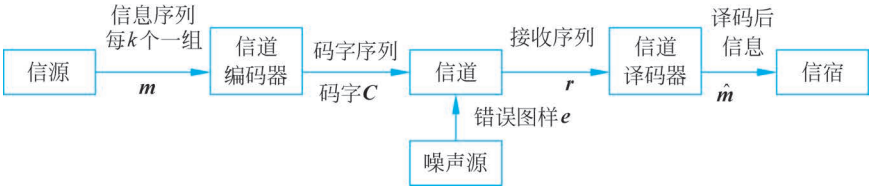


图 5-3 简化的通信系统模型

信道编码器将输入的信息序列，每 k 个信息符号分成一组，记作 $\boldsymbol{m}$，称为信息组，其中的每一位称为信息元 m_i，$i=1,2,\cdots,k$。通过信道编码器，产生一个码字 $\boldsymbol{C}$，长度为 N，通常将 $R=k/N$ 称为码率。码字 $\boldsymbol{C}$ 在有噪声的信道中传输，接收到 $\boldsymbol{r}$ 序列，通过信道译码器，获得译码后的信息 $\hat{\boldsymbol{m}}$。

5.2.1 错误图样

首先了解在有噪信道中产生差错的特点，数据在信道中传输时受到各种干扰，这些干扰是使数据产生差错的主要原因。差错有两种基本形式：一是随机错误，即数据序列中前后码元之间是否发生错误彼此无关，将产生随机错误的信道称为随机信道，以高斯白噪声为主体的信道就属于这类信道，如太空信道、卫星信道和同轴电缆信道等；二是突发性错误，即错误之间有相关性，错误成串出现，将产生突发性错误的信道称为突发差错信道，实际的衰落信道、码间干扰信道均属于这类信道，如短波信道和移动通信信道等。

对于二进制数字通信系统，设信道输入的发送序列为 $\boldsymbol{C}=0000\cdots$，由于干扰，信道输出的接收序列为 $\boldsymbol{r}=0010\cdots$，表明接收序列的第 3 位发生错误 $\boldsymbol{e}$，即 $\boldsymbol{e}=\boldsymbol{r}-\boldsymbol{C}$，称为错误图样。由于二进制数字通信系统中加法运算与减法运算等价，因此，$\boldsymbol{e}=\boldsymbol{r}\oplus\boldsymbol{C}$ 或者 $\boldsymbol{r}=\boldsymbol{C}\oplus\boldsymbol{e}$。若发送序列为 $00100000\cdots$，而接收序列为 $10111000\cdots$，此时错误图样为 $\boldsymbol{e}=0010000\cdots\oplus 10111000\cdots=10011000\cdots$，这种错误称为突发错误。突发错误的长度 l 等于第 1 个错误与最后一个错误之间的长度，该例中突发错误的长度为 5。当然，如果获得错误图样 $\boldsymbol{e}$，由接收序列 $\boldsymbol{r}$ 可计算出发送的信息序列 $\hat{\boldsymbol{C}}=\boldsymbol{e}\oplus\boldsymbol{r}$。

对于二进制对称信道，若信道的错误传递概率为 ε，正确传递概率将为 $1-\varepsilon$。N 长的码字在这样的信道中传输时，发生一位随机错误的差错图样的概率为 $p(e_1)=(1-\varepsilon)^{N-1}\varepsilon$，发生两位随机错误的差错图样的概率为 $p(e_2)=(1-\varepsilon)^{N-2}\varepsilon^2$，以此类推，发生 k 位随机错误的错误图样的概率为 $p(e_k)=(1-\varepsilon)^{N-k}\varepsilon^k$。由于信道错误传递概率一般总是远小于 1，因此有 $p(e_N)<\cdots<p(e_k)<\cdots<p(e_2)<p(e_1)$，即发生一位随机错误、两位随机错误的概率将大于发生多位随机错误的概率。因此，无记忆信道中总是优先纠正发生较少位的随机错误。

5.2.2 矢量空间和码矢量

信道编码的任务就是通过构造出以最小冗余度为代价换取最大抗干扰性能的好码。码字可被看作一个矢量或一个矩阵，于是可以采用矢量空间的概念研究码的本质特性。

定义 5.1：对于 n 重有序元素的集合 $\boldsymbol{V}$，$\boldsymbol{V}=\{\boldsymbol{V}_i\}$，$\boldsymbol{V}_i=(v_{i_1},v_{i_2},\cdots,v_{i_n})$，$v_{i_j}\in F$，$j=1,\cdots,n$，$F$ 表示码元所在的域，对于二进制码，F 代表二元域，常用 GF(2)表示。若集合 $\boldsymbol{V}$ 满足以下条件。

(1) $\boldsymbol{V}$ 中的矢量元素在矢量加运算下构成加群。

(2) $\boldsymbol{V}$ 中的矢量元素与数域 F 元素的标乘封闭在 $\boldsymbol{V}$ 中。

(3) 分配律和结合律成立。

则称集合 $\boldsymbol{V}$ 是数域 F 上的 n 维矢量空间，或称 n 维线性空间。n 维矢量也称为 n 重(n-tuples)矢量。

另外，GF(q)域，又称伽罗瓦域(Galois Field)，是由符合上述运算规则的 δ 个元素构成

的 q 元有限域，其域元素为 $\{0,1,\cdots,q-1\}$。与 q 元域对应的加、乘运算为模 q 运算。如果 $q=p^m$（p 是素数，m 是任意正整数），则有可能将 GF(q)域扩展成 GF(q^m)，称为 GF(q)的扩域。扩域中元素的乘、加运算也是基于模 q 运算。值得注意的是，域里面的乘法和加法不一定是我们平常使用的乘法和加法。

对于域 F 上的若干矢量 $\boldsymbol{V}_1,\boldsymbol{V}_2,\cdots,\boldsymbol{V}_i$ 及 $\boldsymbol{V}_k\in\boldsymbol{V}$，若满足 $\boldsymbol{V}_k=a_1\boldsymbol{V}_1+a_2\boldsymbol{V}_2+\cdots+a_i\boldsymbol{V}_i$（$a_i\in F$），则称 $\boldsymbol{V}_k$ 是 $\boldsymbol{V}_1,\boldsymbol{V}_2,\cdots,\boldsymbol{V}_i$ 的线性组合。

对于域 F 上的若干矢量 $\boldsymbol{V}_1,\boldsymbol{V}_2,\cdots,\boldsymbol{V}_i\in\boldsymbol{V}$，若满足 $a_1\boldsymbol{V}_1+a_2\boldsymbol{V}_2+\cdots+a_i\boldsymbol{V}_i=\boldsymbol{0}$，其中 $a_i\in F$ 不全为零，则称 $\boldsymbol{V}_1,\boldsymbol{V}_2,\cdots,\boldsymbol{V}_i$ 是线性相关的；如果上述条件不成立，则称这些矢量是线性无关或线性独立的。

定义 5.2：如果存在一组线性无关的矢量 $\boldsymbol{V}_1,\boldsymbol{V}_2,\cdots,\boldsymbol{V}_K$，这些矢量的线性组合的集合可构成一个矢量空间，则称这组矢量为该矢量空间的基底。n 维矢量空间应有 n 个基底，也可以说，n 个基底张成 n 维矢量空间。

以 n 个线性无关的矢量为基底，它们的全部线性组合可构成一个 n 维 n 重矢量空间 $\boldsymbol{S}$，这里 n 重指 n 个 GF(q)域元素的有序排列。如果从 n 个基底中选出 k（$k<n$）个基底，则它们的所有线性组合也构成一个集合，这个集合是 $\boldsymbol{S}$ 的一个 k 维子集，称为 k 维 n 重子空间，记作 $\boldsymbol{S}_C$。

【例 5-2】 二元域 GF(2)上三维矢量空间 $\boldsymbol{V}$ 的 3 个自然基底为(100)，(010)，(001)，求它的一维三重子空间和二维三重子空间。

解：由于三维矢量空间有 3 个基底，任取其中一个基底，可以张成一维三重子空间。例如，取基底(100)，则可张成一个子空间，含有 $2^1=2$ 个元素，即 $\boldsymbol{V}_1=\{(000),(100)\}$。以(010)(001)为基底，可以张成一个二维三重子空间，共有 $2^2=4$ 个元素，即 $\boldsymbol{V}_1=\{(000),(001),(010),(011)\}$。

【例 5-3】 求二元扩域 GF(2^3)上的域元素。

解：扩域 GF(2^3)上的域元素可表示成多项式 $f_2x^2+f_1x+f_0$，$f_2,f_1,f_0\in$GF(2)，其可对应到三维矢量(f_2,f_1,f_0)，其自然基底是(100)，(010)和(001)。由于每个系数只有 $\{0,1\}$两种选择，自然基底只能组合出 8 个矢量，构成 GF(2)上的三维矢量空间。

每个矢量可对应一个码字，所以也可把矢量说成码矢、码字，或一个 n 重矢量。如果两矢量的内积为 0，则称为矢量正交。如果一矢量空间中的任一矢量都与另一矢量空间正交，则两个空间称为对偶空间，其中一个空间是另一个空间的零空间（又称零化空间）。

n 维 n 重空间 $\boldsymbol{S}$ 中含有 2^n（二进制分组码）个长度为 n 的矢量，其中至少存在一组 n 个矢量 $\boldsymbol{B}_1,\boldsymbol{B}_2,\cdots,\boldsymbol{B}_n$ 可用作基底，n 维 n 重空间的所有矢量都是这些基底的线性组合，$\alpha_1\boldsymbol{B}_1+\alpha_2\boldsymbol{B}_2+\cdots+\alpha_n\boldsymbol{B}_n$，$\alpha_i\in\{0,1\}$，$i=1,2,\cdots,n$。若 n 个基底相互正交，将 n 个基底分成 k 个和 $n-k$ 个两组，则可分别张成 k 维 n 重和($n-k$)维 n 重的两个对偶空间，若将 k 维 n 重空间作为码空间 $\boldsymbol{C}$，则($n-k$)维 n 重空间可用作检验空间 $\boldsymbol{H}$。纠错编码过程就是将 k 维 k 重空间通过一定的映射法则，映射到 n 维 n 重空间的一个 k 维 n 重子空间上，其过程如图 5-4 所示，它的核心是：

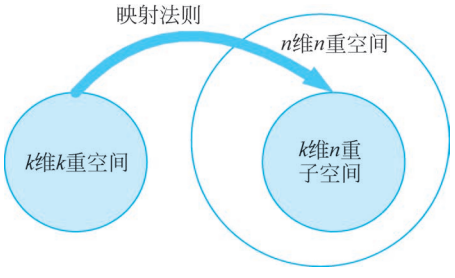


图 5-4 信道编码方法

①确定码空间，选择 k 个正交基底 $\boldsymbol{B}_1,\boldsymbol{B}_2,\cdots,\boldsymbol{B}_k$ 张成码空间；②确定映射法则。

分组码由一组固定长度为 N 的码矢量构成。每个矢量元素就是一个码元，其值选自GF(q)域中的元素。长度为 N 的二进制分组码有 2^N 种可能的组合，选择其中的 2^k 种($k<N$)构成一个许用码的码集 $\boldsymbol{C}$ 称为分组码。因此，分组码的编码就是将 k 位的信息组一一对应地映射到许用码码集，不同的编码算法对应不同的映射方法，这样的分组码又称为(N,k)分组码，并定义 $k/N=R$ 为码率。

假设 C_i 和 C_j 是某(N,k)分组码的两个码字，a_1 和 a_2 是码元字符集里的任意两个元素，那么，当且仅当 $a_1C_i+a_2C_j$ 也是码字时，该码称为线性码。因为 $\forall a_1=0,a_2=0,a_1C_i+a_2C_j=0$。因此，线性码必须包含全零码字。

5.2.3 码距与纠检错能力

假设一个码字 $\boldsymbol{C}_i=(C_{i1},C_{i2},\cdots,C_{iN})(i=1,2,\cdots,M,M$ 为码集中码字总个数)可以被看作一个 N 重矢量，每个码字与 N 维矢量空间 2^N 中的一个点对应(二进制码字)，全部码字所对应的点的集合构成矢量空间里的一个子集，即 k 维 N 重子空间。若 C_i 和 C_j 是码集上的任意两个有效码字，则它们之间的汉明距离定义为两码字中不相同元素的个数，即 $D(C_i,C_j)=\sum_{k=1}^{N}C_{i_k}\oplus C_{j_k}$ ($i,j=1,2,\cdots,M,M$ 为码集中码字总个数)，所有码集中汉明距离的最小值称为最小码距，记作 $d_{\min}$。当传输没有差错时，接收到的 N 重矢量对应到该子集相应的有效码字(许用码集的有效点)上。但当传输过程出现差错时，接收到的 N 重矢量会有两种可能性：一种是不再对应到该子集，而是对应到另外一个空间；另一种仍然是对应到该子集，却对应到该子集另外的有效点上。通过前一种情况能够判断出该码字出现了差错，而后一种情况将无法做出判断。若码集的最小距离为 $d_{\min}$，那么一个能使码字矢量空间点偏移 $d_{\min}$ 的差错组合(称为重量为 $d_{\min}$ 的错误图样)，会将码字矢量的空间点位置从子集的一个有效点偏移到另一个有效点上，从而产生一个“不可检测的差错”。但是，如果差错数小于 $d_{\min}$，就不可能将码字矢量空间点从子集的一个有效点偏移到子集的另一个有效点上，也就可以检测出差错的存在。因此，对于(N,k)分组码来说，最多能够检测出 $d_{\min}-1$ 个错误。

码的纠错能力同样也取决于码的最小距离。为了确定(N,k)分组码的纠错能力，一种简单的办法是将 2^k 个码字看作位于 n 维空间上的点。如果以子集中每个有效码字为球心，以汉明距离 t 为半径作 2^k 个球体，那么它们之中任意一对球体两两不相交(包括不相切)的 t 的最大取值为 $t=\left\lfloor\frac{d_{\min}-1}{2}\right\rfloor$，其中 $\lfloor\cdot\rfloor$ 表示向下取整。那么，在每一个球内(含有与该码字之间的距离小于 t 的所有可能)的接收码字，被译成位于球心的那个有效码字。最小距离为 $d_{\min}$ 的分组码的纠错能力为 $t=\left\lfloor\frac{d_{\min}-1}{2}\right\rfloor$，且纠错能力总是小于检错能力。

如上所述，分组码如果单独考虑检错或单独考虑纠错，则可以检测 $d_{\min}-1$ 个错误或纠正 $t=\left\lfloor\frac{d_{\min}-1}{2}\right\rfloor$ 个错误。然而，如果将两种能力放在一起考虑，情况就有所变化。能纠正 t 个错误显然能够检测 t 个错误，但如果想检测 $d_{\min}-1$ 个错误就必须对纠错能力有所抑制。例如，假设两个码字 A、B 的码距为 7，即 $d_{\min}$ 为 7，若码字 A 发生 3 个错误时能够纠正过

来，若发生 4 个错误，按照前面的方法应译成码字 B。这是因为译码器认为接收点是由码字 B 发生 3 个错误而来的，而不认为是码字 A 发生了 4 个错误所导致。换言之，此时译码器的检错能力仅是 3。如果要有 4 个错误的检测能力，只能把球的半径从 3 减到 2。这样，错误数不大于 2 的接收码可以纠正，错误数为 3 个或 4 个时可以检测，而 4 个以上的错误可能是接收点落在别的球内，将变成不可检测错误。同理，保持 $d_{\min}=7$ 不变，也可以检测 5 个错误、纠正 1 个错误。一般性的结论是：如果最小距离为 $d_{\min}$ 的码字能同时检测 e 个错误，纠正 t 个错误，则 $d_{\min}\geqslant e+t+1$，$e\geqslant t$。

对于线性码而言，码距特性就是码的重量特性。如果有 2^k 个码字，就存在 $2^k(2^k-1)/2$ 个距离，这些距离是大小不一的，码的总体性能取决于这些距离的分布特性，而纠错能力取决于其中的最小值 $d_{\min}$。正如各符号等概率时熵最大一样，当所有的码距相等时，码的性能将是最好的。


视频讲解

5.3 离散信道编码定理

1948 年，香农在他的论文中给出了有关信息传输的基本定理，称为有噪信道编码定理。尽管他没有给出定理的严格证明过程，但他引入的随机编码方法在后来的信道编码定理的严格证明中一直被采用，因而，下面首先介绍随机编码的概念。

按照分组码的编码方法，编码器要对每一消息 $m(m=1,2,\cdots,M)$ 给以相应长为 N 的码字 $C_m=C_{m1}C_{m2}\cdots C_{mN}$。随机编码的思想就是按照信道输入符号的概率分布完全随机地选取码字母表中的字母作为码字的符号，从码字 C_1 的第 1 个码符号 C_{11} 开始直到码字 C_M 的最后一个码符号 C_{MN} 为止，以这种方式得到全部 $M=2^{NR}$ 的有效码字，组成一个码集 $C=\{C_1,C_2,\cdots,C_M\}$。于是，随机编码产生某一特定码字 C 的概率 $p(C)$ 是

$$p(C)=\prod_{m=1}^{M}\prod_{l=1}^{N}p(C_{ml}) \tag{5.3.1}$$

所有可能产生的码的总数应是 $|A|^{MN}$，其中 $|A|$ 表示码字母表 A 的大小。例如，当 $|A|=2$，$N=16$，$M=2^8$ 时，码集中码字的总数为 $2^{16\times 2^8}=2^{4096}\approx 10^{1233}$，这是一个很大的数。当然，在这些码集中有一部分是无法使用的，例如，码集中有若干码字相同，但由于码集中码字数为 $M=2^{NR}$ 只占全部可能序列数 2^N 的很小的一部分，因此同一码集中码字相同的概率很小，大部分码集中的码字各不相同。

5.3.1 有噪信道编码定理

定理 5.1：（有噪信道编码定理）设离散无记忆信道的容量为 $C_{容量}$，信道的信息传输率为 R。只要 $R<C_{容量}$，总存在码长为 N、码字数 $M=2^{NR}$ 的分组码和相应的译码规则，使译码的平均错误概率 $\overline{P}_e<\varepsilon$，$\varepsilon$ 是大于零的任意小量。

证明：按照随机编码的方法，随机地产生 $M=2^{NR}$ 个码字，记为 $C_m=C_{m1}C_{m2}\cdots C_{mN}$，$m=1,2,\cdots,M$，其概率分布为 $p(C_m)=\prod_{l=1}^{N}p(C_{ml})$。若将这 M 个码字排在一起，构成如下矩阵：

$$\boldsymbol{C}=\begin{bmatrix}\boldsymbol{C}_1\\ \boldsymbol{C}_2\\ \vdots\\ \boldsymbol{C}_M\end{bmatrix}=\begin{bmatrix}C_{11} & C_{12} & \cdots & C_{1N}\\ C_{21} & C_{22} & & C_{2N}\\ & & \ddots & \\ \vdots & & & \vdots\\ C_{M1} & C_{M1} & \cdots & C_{MN}\end{bmatrix} \tag{5.3.2}$$

从而产生一个特定码字 $\boldsymbol{C}$ 的概率是 $p(\boldsymbol{C})=\prod_{m=1}^{M}\prod_{l=1}^{N}p(C_{ml})$。令输入消息等概率分布，即 $p(C_m)=\frac{1}{M}$，$m=1,2,\cdots,M$，则特定码字 $\boldsymbol{C}$ 的平均错误概率是

$$p_e(\boldsymbol{C})=\sum_{m=1}^{M}p(C_m)p(e\mid C_m)=\frac{1}{M}\sum_{m=1}^{M}p(e\mid C_m) \tag{5.3.3}$$

在全体码集$\{\boldsymbol{C}\}$上对 $p_e(\boldsymbol{C})$取平均，得到全码集$\{\boldsymbol{C}\}$上的平均错误概率为

$$\bar{p}_e=\sum_{|\boldsymbol{C}|}p(\boldsymbol{C})p_e(\boldsymbol{C})=\frac{1}{M}\sum_{m=1}^{M}\sum_{|\boldsymbol{C}|}p(\boldsymbol{C})p(e\mid C_m) \tag{5.3.4}$$

其中，$|\boldsymbol{C}|$表示全码集中码总数。在随机编码中，不同 C_m 产生的方法完全一样，因此 $p(e|C_m)$ 在对码集$\{\boldsymbol{C}\}$取平均后将得到一个与 m 无关的值，即 $\sum_{|\boldsymbol{C}|}p(\boldsymbol{C})p(e\mid C_m)$ 的值与 m 无关，且设 $m=1$，得到

$$\bar{p}_e=\sum_{|\boldsymbol{C}|}p(\boldsymbol{C})p(e\mid C_1) \tag{5.3.5}$$

设 y 是发送 C_1 码字时经信道输出后接收到的序列，定义 y 与 C_m 构成联合典型序列事件 E_m，即 $E_m=\{(C_m,y)\in T_{XY}(N,\varepsilon)\}$，$m=1,2,\cdots,M$，$T_{XY}(N,\varepsilon)$为输入输出联合典型序列。$X$ 表示输入，Y 表示输出。按照联合典型译码法则，译码错误将使 y 不与 C_1 成为联合典型，或 y 与 C_1 以外的其他码字产生联合典型序列的概率，故

$$\bar{p}_e=p(E_1^C\cup E_2\cup\cdots\cup E_M)\leqslant p(E_1^C)+\sum_{m=2}^{M}p(E_m) \tag{5.3.6}$$

由于 y 是对应输入 C_1 时的信道输出，所以它与 $C_2,C_3,\cdots,C_M$ 相互独立，故可得

$$\begin{aligned}\bar{p}_e&\leqslant\varepsilon_1+\sum_{m=2}^{M}2^{-N[I(X;Y)-3\varepsilon_1]}\\&=\varepsilon_1+(M-1)2^{-N[I(X;Y)-3\varepsilon_1]}\\&\leqslant\varepsilon_1+2^{NR}2^{-N[I(X;Y)-3\varepsilon_1]}\\&=\varepsilon_1+2^{-N[I(X;Y)-R-3\varepsilon_1]}\end{aligned} \tag{5.3.7}$$

当 $I(X;Y)-R-3\varepsilon_1>0$，即 $I(X;Y)>R$ 时，存在充分大的 N，使 $2^{-N[I(X;Y)-R-3\varepsilon_1]}<\varepsilon_1$，从而

$$\bar{p}_e<\varepsilon_1+\varepsilon_1=2\varepsilon_1 \tag{5.3.8}$$

如果取输入分布 $p(a_k)$是达到信道容量 $C_{容量}$ 的分布，则此时 $I(X;Y)=C_{容量}$，于是当 $R<C_{容量}$时，对任意的 $\varepsilon=\frac{\varepsilon_1}{2}>0$，总存在充分大的 N，使 $\bar{p}_e<\varepsilon$。由于$\{\boldsymbol{C}\}$是随机码集合，因此至少存在一个码 $\boldsymbol{C}\in\{\boldsymbol{C}\}$，使其平均错误概率满足 $\bar{p}_e<\varepsilon$。

香农只是证明了满足这种特性码的存在性，还不能按照他的证明方法得到很好的编码方法。由于随机编码所得到的码集很大，通过搜索得到好码的方法在实际中很难实现；而

且即使找到其中的好码，这种好的码字也是毫无结构的，这意味着译码时只能用查表的方式，在 N 很大的情况下，这一译码表所需要的存储量是难以接受的。因此，真正实用的信道编码还需要通过各种数学工具来构造，使码具有很好的结构以便于更好地译码。

5.3.2 有噪信道编码逆定理

我们知道平均错误概率 $\bar{p}_e$ 与译码规则有关，而译码规则又由信道特性决定。由于信道中存在噪声，导致信道输出端出现错误，所以接收到输出符号后，对发送的是什么符号仍存在不确定性。因此，平均错误概率 $\bar{p}_e$ 与条件熵 $H(X|Y)$ 之间有一定关系，这就是费诺(Fano)不等式，该不等式也是有噪信道编码逆定理证明的基础。

设离散无记忆信道的输入随机变量 X 取值于符号集 $A=\{a_1,a_2,\cdots,a_r\}$，输出随机变量 Y 取值于符号集 $B=\{b_1,b_2,\cdots,b_s\}$，$\bar{p}_e$ 为某一译码规则的平均错误概率，则

$$H(X\mid Y)\leqslant H(\bar{p}_e,1-\bar{p}_e)+\bar{p}_e\log(r-1) \tag{5.3.9}$$

称为费诺不等式，其中 $H(\cdot)$ 为信息熵。此处，$\bar{p}_e$ 可以同 p_e。

证明：由式(5.1.15)，得

$$p_e=\sum_{j=1}^{s}\sum_{k=1,a_k\neq a^*}^{r}p(a_kb_j)$$

$$1-p_e=\sum_{j=1}^{s}p(a^*b_j)$$

因此，有

$$\begin{aligned}H(p_e,1-p_e)+p_e\log(r-1)&=p_e\log\frac{1}{p_e}+(1-p_e)\log\frac{1}{1-p_e}+p_e\log(r-1)\\&=\sum_{j=1}^{s}\sum_{k=1,a_k\neq a^*}^{r}p(a_kb_j)\log\frac{r-1}{p_e}+\sum_{j=1}^{s}p(a^*b_j)\log\frac{1}{1-p_e}\end{aligned}$$

而

$$\begin{aligned}H(X\mid Y)&=\sum_{k=1}^{r}\sum_{j=1}^{s}p(a_kb_j)\log\frac{1}{p(a_k\mid b_j)}\\&=\sum_{j=1}^{s}\sum_{k=1,a_k\neq a^*}^{r}p(a_kb_j)\log\frac{1}{p(a_k\mid b_j)}+\sum_{j=1}^{s}p(a^*b_j)\log\frac{1}{p(a^*\mid b_j)}\end{aligned}$$

则

$$\begin{aligned}&H(X\mid Y)-H(p_e,1-p_e)-p_e\log(r-1)\\&=\sum_{j=1}^{s}\sum_{k=1,a_k\neq a^*}^{r}p(a_kb_j)\log\frac{p_e}{(r-1)p(a_k\mid b_j)}+\sum_{j=1}^{s}p(a^*b_j)\log\frac{1-p_e}{p(a^*\mid b_j)}\\&\leqslant\sum_{j=1}^{s}\sum_{k=1,a_k\neq a^*}^{r}p(a_kb_j)\left[\frac{p_e}{(r-1)p(a_k\mid b_j)}-1\right]\log\mathrm{e}+\sum_{j=1}^{s}p(a^*b_j)\left[\frac{1-p_e}{p(a^*\mid b_j)}-1\right]\log\mathrm{e}\\&=\left[\frac{p_e}{r-1}\sum_{j=1}^{s}\sum_{k=1,a_k\neq a^*}^{r}p(b_j)-\sum_{j=1}^{s}\sum_{k=1,a_k\neq a^*}^{r}p(a_k,b_j)+(1-p_e)\sum_{j=1}^{s}p(b_j)-\right.\\&\quad\left.\sum_{j=1}^{s}p(a^*b_j)\right]\log\mathrm{e}\end{aligned}$$

$$=\left[\frac{p_e}{r-1}(r-1)-p_e+(1-p_e)-(1-p_e)\right]\log e$$
$$=0$$

其中用到了不等式 $\ln x \leqslant x-1$。由此证得费诺不等式：$H(X|Y)\leqslant H(p_e,1-p_e)+p_e\log(r-1)$。

下面给出有噪信道编码逆定理。

定理 5.2：（有噪信道编码逆定理）设离散无记忆信道的容量为 $C_{容量}$，R 是消息传输率，则当 $R>C_{容量}$ 时，无论码长 N 有多长，必存在某一常数 $\delta>0$，使在码字等概率分布下平均错误概率 $\bar{p}_e(N)\geqslant\delta$。

证明：对于任意 $\varepsilon>0$，设 $M=2^{N(R+\varepsilon)}$，$X=X_1X_2\cdots X_N$ 为输入随机序列，因为码字等概率分布，故 $H(X)=\log M$，因此 $I(X;Y)=H(X)-H(X|Y)=\log M-H(X|Y)$。又由于 $I(X;Y)\leqslant NC_{容量}$，故 $\log M-H(X|Y)\leqslant NC_{容量}$。由费诺不等式，得

$$\begin{aligned}\log M-NC_{容量}&\leqslant H(X|Y)\\&\leqslant H(\bar{p}_e(N),1-\bar{p}_e(N))+\bar{p}_e(N)\log(M-1)\\&\leqslant 1+\bar{p}_e(N)\log(M-1)\\&<1+\bar{p}_e(N)\log M\end{aligned}$$

于是

$$\frac{\log M-NC_{容量}-1}{\log M}<\bar{p}_e(N)$$

即

$$\begin{aligned}\bar{p}_e(N)&>1-\frac{NC_{容量}+1}{\log M}>1-\frac{NC_{容量}+1}{N(R+\varepsilon)}\\&>1-\frac{NC_{容量}+1}{N(C_{容量}+\varepsilon)}=\frac{N\varepsilon-1}{N(C_{容量}+\varepsilon)}\\&=\frac{\varepsilon-\frac{1}{N}}{C_{容量}+\varepsilon}\end{aligned}$$

由于 $\dfrac{\varepsilon}{C_{容量}+\varepsilon}>0$，故存在 $\delta_1>0$ 和 $N_0>0$，当 $N>N_0$ 时，有 $\bar{p}_e(N)\geqslant\delta_1>0$。又若对某一个固定的 N，有 $\bar{p}_e(N)=0$，则 $\log M-NC_{容量}<0$，即 $M=2^{N(R+\varepsilon)}\leqslant 2^{NC_{容量}}$ 与 $R>C_{容量}$ 矛盾，从而对任给 N，有 $\bar{p}_e(N)>0$。因此，只需取 $\delta=\min\{\delta_1,\bar{p}_e(1),\bar{p}_e(2),\cdots,\bar{p}_e(N_0)\}$，则对所有的 N，都有 $\bar{p}_e(N)>\delta$。当选用码字数（或消息数）$M=2^{N(C_{容量}+\varepsilon)}$ 时，信道的消息传输率为 $R=\dfrac{\log M}{N}=C_{容量}+\varepsilon$。这表明，信道消息传输率 R 已超过了信道容量，这是不可能的，既要使消息传输率大于信道容量而又无错误地传输消息是不可能的。

5.4 信道编码方法

在通信中，由于码字序列是一种随机序列，接收端无法预知码元的取值，因而也无法识别其中有无错误。因此，可以在发送端的信息序列中增加一些差错控制元，由它们来校验是

否出错，称为监督码元。这些监督码元和信息序列之间有确定的关系。信息序列和监督码元之间的关系不同，形成码的类型也不同，大体上分为两类：分组码和卷积码。其中，分组码是把信息序列以每 k 个码元分组，编码器将每个信息组按照一定规律产生 r 个多余的码元(有时也称为校验元)，形成一个长为 $n=k+r$ 的码字。当分组码的信息序列与监督码元之间是线性关系时，即可用信息的线性方程组表示监督码元，这种分组码就称为线性分组码，如汉明码和循环码。卷积码则不仅考虑信息序列与监督码元之间的关联，还需考虑信息序列间的相关信息。码长越长，性能越好，但是译码复杂度越大。卷积码给出了一种利用分组间前后相关信息等效增加码长的方法。


视频讲解

5.4.1 线性分组码

线性分组码建立在代数群论基础上，各许用码字的集合构成了代数学中的群，它们具有以下主要性质。

(1) 任意两个许用码字之和(对于二进制码这个和的含义是模二加)仍为一个许用码字，即线性分组码具有封闭性。

(2) 码字间的最小码距等于非零码的最小码重。

对于长度为 n 的二进制线性分组码，它有 2^n 种可能的码组。从 2^n 种码组中，可以选择 $M=2^k$ 个码组($k<n$)组成一种码集。这样，一个 k 比特信息的线性分组码可以映射到一个长度为 n 的码组上。该码组是从 $M=2^k$ 个码组构成的码集中选出来的，这样剩下的码组就可以对这个分组码进行检错或纠错。

1. 线性分组码的编码

线性分组码的码空间 $\boldsymbol{C}$ 是由 k 个线性无关的基底张成的 k 维 n 重子空间，码空间的所有元素都可以写成 k 个基底的线性组合，即

$$\boldsymbol{C}=m_{k-1}\boldsymbol{g}_{k-1}+\cdots+m_1\boldsymbol{g}_1+m_0\boldsymbol{g}_0 \tag{5.4.1}$$

因而，线性分组码的关键是基底、子空间和映射规则。若对于每个信息矢量 $\boldsymbol{m}=[m_{k-1},\cdots,m_1,m_0]$，其对应的码字为 $\boldsymbol{C}=[C_{n-1},\cdots,C_1,C_0]$，那么式(5.4.1)可以写成如下矩阵形式：

$$[C_{n-1}\quad\cdots\quad C_1\quad C_0]=[m_{k-1}\quad\cdots\quad m_1\quad m_0]\begin{bmatrix} g_{(k-1)(n-1)} & \cdots & g_{(k-1)1} & g_{(k-1)0} \\ \vdots & \ddots & \vdots & \vdots \\ g_{1(n-1)} & \cdots & g_{11} & g_{10} \\ g_{0(n-1)} & \cdots & g_{01} & g_{00} \end{bmatrix} \tag{5.4.2}$$

即 $\boldsymbol{C}=\boldsymbol{mG}$，其中 $\boldsymbol{G}$ 为该码的生成矩阵，由 k 个基底矢量组成，即 $\boldsymbol{G}=\begin{bmatrix}\boldsymbol{g}_{k-1}\\ \vdots \\ \boldsymbol{g}_1 \\ \boldsymbol{g}_0\end{bmatrix}$。如果要保证 (n,k) 线性分组码能够构成 k 维 n 重子空间，$\boldsymbol{G}$ 的 k 行矢量必须是线性无关的。因此，k 个基底 $\boldsymbol{g}_i$ 也是码字。

原则上，(n,k) 线性分组码的任何生成矩阵都可以通过行运算或者列置换简化成“系统矩阵”形式：

$$\boldsymbol{G}=(\boldsymbol{I}_k \quad \vdots \quad \boldsymbol{P})=\left[\begin{array}{cccc:ccc} 1 & 0 & \cdots & 0 & p_{(k-1)(n-k-1)} & \cdots & p_{(k-1)1} & p_{(k-1)0} \\ 0 & 1 & \cdots & 0 & \vdots & \ddots & \vdots & \vdots \\ \vdots & \vdots & \ddots & \vdots & p_{1(n-k-1)} & \cdots & P_{11} & P_{10} \\ 0 & 0 & \cdots & 1 & p_{0(n-k-1)} & \cdots & P_{01} & P_{00} \end{array}\right] \tag{5.4.3}$$

这样，在生成码字时，码字的前 k 位一定与信息码元相同，而其余的 $n-k$ 位冗余比特称为一致校验位，是前 k 个信息位的线性组合，这样的码称为系统码。每个(n,k)线性组合码都可以和一个系统的(n,k)线性码等效。不具备“系统”特性的码也称为非系统码。非系统码与系统码并无本质的区别，系统码不改变码集，只改变映射规则。

另一方面，与任何一个(n,k)线性码的码空间 $\boldsymbol{C}$ 相对应，一定存在一个对偶空间 $\boldsymbol{H}$。将 $\boldsymbol{H}$ 空间的 $n-k$ 个基底排列起来可构成一个$(n-k)\times n$ 矩阵，称为校验矩阵 $\boldsymbol{H}$。

由于校验矩阵是$(n,n-k)$对偶码的生成矩阵，它的每一行是一个基底，也是一个码字。因此，(n,k)码的任一个码字均正交于校验矩阵的任一行矢量，即

$$\begin{aligned} &\boldsymbol{C}\boldsymbol{H}^{\mathrm{T}}=\boldsymbol{0} \quad (1\times(n-k)\text{ 全零行矢量}) \\ \Rightarrow &\boldsymbol{G}\boldsymbol{H}^{\mathrm{T}}=\boldsymbol{0} \quad (k\times(n-k)\text{ 全零行矢量}) \end{aligned} \tag{5.4.4}$$

如果(n,k)线性码是系统码，$\boldsymbol{G}=[\boldsymbol{I}_k \quad \vdots \quad \boldsymbol{P}]$，根据式(5.4.4)和 GF(2)域的特性，可得到校验矩阵为 $\boldsymbol{H}=[-\boldsymbol{P}^{\mathrm{T}} \quad \vdots \quad \boldsymbol{I}_{n-k}]$，负号在二进制情况下可省略，因为模 2 减法和模 2 加法是等同的。

【例 5-4】 考虑一(7,4)线性码，其生成矩阵 $\boldsymbol{G}=\left[\begin{array}{ccccccc} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{array}\right]$。

(1) 对于信息序列 $\boldsymbol{m}=[1 \quad 0 \quad 1 \quad 1]$，求其对应码字。

(2) 根据给定的系统码生成矩阵，画出编码器原理图。

(3) 确定 $\boldsymbol{r}=[1 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 1]$是否为码字。

解：(1) 由编码公式得

$$[C_6 \quad C_5 \quad C_4 \quad \cdots \quad C_0]=[m_3 \quad m_2 \quad m_1 \quad m_0]\boldsymbol{G}$$

得

$$C_2=0, \quad C_1=0, \quad C_0=0$$

所以

$$\boldsymbol{C}=[1 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0]$$

(2) 二进制(n,k)的编码器可用 k 级移存器和连接列移存器适当位置的 $n-k$ 个模 2 加法器组成，加法器生成校验位后按顺序暂存在另一个 $n-k$ 的移存器中。原理图如图 5-5 所示。

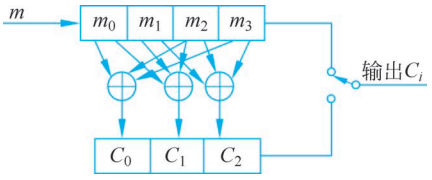


图 5-5 编码器原理图

(3) 判断 $\boldsymbol{rH}^{\mathrm{T}}$ 是否为 0。

由生成矩阵可得到校验矩阵为

$$\boldsymbol{H}=[\boldsymbol{P}^{\mathrm{T}} \quad \vdots \quad \boldsymbol{I}_{n-k}]=\begin{bmatrix}1&1&1&0&1&0&0\\0&1&1&1&0&1&0\\1&1&0&1&0&0&1\end{bmatrix}$$

于是计算 $\boldsymbol{rH}^{\mathrm{T}}$,得

$$[1\quad 0\quad 0\quad 1\quad 1\quad 0\quad 1]\begin{bmatrix}1&0&1\\1&1&1\\1&1&0\\0&1&1\\1&0&0\\0&1&0\\0&0&1\end{bmatrix}=[0\quad 1\quad 1]\neq 0$$

所以,$\boldsymbol{r}=[1\quad 0\quad 0\quad 1\quad 1\quad 0\quad 1]$不是码字。

校验矩阵除了校验码字的作用外,还可用来计算最小码距。这是因为,校验矩阵 $\boldsymbol{H}$ 可以写成 n 个列矢量的组合,即

$$\boldsymbol{H}=\begin{bmatrix}h_{(n-k-1)(n-1)}&\cdots&h_{(n-k-1)1}&h_{(n-k-1)0}\\\vdots&\ddots&\vdots&\vdots\\h_{1(n-1)}&\cdots&h_{11}&h_{10}\\h_{0(n-1)}&\cdots&h_{01}&h_{00}\end{bmatrix}=[\boldsymbol{h}_{n-1}\quad\cdots\quad\boldsymbol{h}_1\quad\boldsymbol{h}_0] \tag{5.4.5}$$

其中,$\boldsymbol{h}_j(j=n-1,\cdots,1,0)$是$(n-k)\times 1$ 列矢量。

并且有

$$\begin{aligned}\boldsymbol{CH}^{\mathrm{T}}&=[C_{n-1}\quad\cdots\quad C_1\quad C_0][\boldsymbol{h}_{n-1}\quad\cdots\quad\boldsymbol{h}_1\quad\boldsymbol{h}_0]^{\mathrm{T}}\\&=C_{n-1}\boldsymbol{h}_{n-1}^{\mathrm{T}}+\cdots+C_1\boldsymbol{h}_1^{\mathrm{T}}+C_0\boldsymbol{h}_0^{\mathrm{T}}=0\end{aligned} \tag{5.4.6}$$

其中,$\boldsymbol{CH}^{\mathrm{T}}$ 代表 n 个 $1\times(n-k)$矢量 $\boldsymbol{h}_j^{\mathrm{T}}$ 的线性组合。

由前面的分析可知,分组码的最小距离等于 $d_{\min}$,说明码集里重量最小的码字有 $d_{\min}$ 个非零码元。若将该最小重量的码字代入式(5.4.6),那么将有 $d_{\min}$ 个 $\boldsymbol{h}_j^{\mathrm{T}}$ 项。由此可以断言:$\boldsymbol{H}$ 矩阵的列矢量至少要有 $d_{\min}$ 个才能线性相关,$d_{\min}-1$ 个列矢量必定是线性无关的。另外,$\boldsymbol{H}$ 是$(n-k)\times n$ 矩阵,其秩至多是 $n-k$,即最多有 $n-k$ 个列矢量线性无关,综上可以得到 $d_{\min}-1\leqslant n-k$,即

$$d_{\min}\leqslant n-k+1 \tag{5.4.7}$$

那么,如何用校验矩阵计算最小码距呢?现通过定理 5.3 给出。

定理 5.3:设 $\boldsymbol{H}$ 是线性码 $\boldsymbol{C}$ 的校验矩阵,则码 $\boldsymbol{C}$ 的最小码间距离(或最小非零码字重量)等于 $\boldsymbol{H}$ 中和为 0 的最小列数。

证明略。

【例 5-5】 已知某线性分组码的校验矩阵为 $\boldsymbol{H}=\begin{bmatrix}1&0&0&1&0&1&1\\0&1&0&1&1&1&0\\0&0&1&0&1&1&1\end{bmatrix}$,求其最小

码距。

解：根据式(5.4.7)，有

$$d_{\min} \leqslant n-k+1=4$$

再根据定理 5.3，$\boldsymbol{H}$ 中的所有列均不为 0，而且没有两列是相同的，码 $\boldsymbol{C}$ 的最小距离至少为 3，例如，0 列、2 列、6 列之和为 0，因此 $d_{\min}=3$。

定义 5.3：极大最小距离码(Maximized Minimum Distance Code，MDC)：如果(n,k)线性码的 $d_{\min}$ 达到了 $n-k+1$，则称该(n,k)线性分组码为极大最小距离码。

显然，当 n、k 确定之后，MDC 码达到了纠错能力的极限，是给定条件下纠错能力最强的码，所以也是纠错码的设计目标。然而在二进制码中，除了将一位信息位重复 n 次的$(n,1)$码外，不存在其他的二进制的 MDC 码。但如果是非二进制码，则是存在极大最小距离码的，如 RS(Reed-Solomon)码就是 MDC 码。

2. 线性分组码的译码

码字 $\boldsymbol{C}=[C_{n-1},C_{n-2},\cdots,C_1,C_0]$在传输过程中受到各种干扰，接收端的接收码 $\boldsymbol{R}=[r_{n-1},r_{n-2},\cdots,r_1,r_0]$不一定等于发送码，两者间的差异为差错图样，即错误图样。差错图样 $\boldsymbol{E}=[e_{n-1},e_{n-2},\cdots,e_1,e_0]=\boldsymbol{R}-\boldsymbol{C}=[r_{n-1}-c_{n-1},\cdots,r_0-c_0]$。由于二进制模 2 加与模 2 减是等同的，因而有 $\boldsymbol{E}=\boldsymbol{R}+\boldsymbol{C}$ 或 $\boldsymbol{R}=\boldsymbol{E}+\boldsymbol{C}$。

同时，由于 $\boldsymbol{C}\boldsymbol{H}^{\mathrm{T}}=0$，所以 $\boldsymbol{R}\boldsymbol{H}^{\mathrm{T}}=(\boldsymbol{C}+\boldsymbol{E})\boldsymbol{H}^{\mathrm{T}}=\boldsymbol{C}\boldsymbol{H}^{\mathrm{T}}+\boldsymbol{E}\boldsymbol{H}^{\mathrm{T}}=\boldsymbol{E}\boldsymbol{H}^{\mathrm{T}}$。当无误码时，即 $\boldsymbol{E}=\boldsymbol{0}$，所以 $\boldsymbol{R}\boldsymbol{H}^{\mathrm{T}}=0$；反之，当有误码时，$\boldsymbol{E}\neq\boldsymbol{0}$，可以得到 $\boldsymbol{R}\boldsymbol{H}^{\mathrm{T}}=\boldsymbol{E}\boldsymbol{H}^{\mathrm{T}}\neq 0$。因此，在 $\boldsymbol{H}^{\mathrm{T}}$ 固定的前提下，$\boldsymbol{R}\boldsymbol{H}^{\mathrm{T}}$ 与 $\boldsymbol{E}$ 有关，而与发送码 $\boldsymbol{C}$ 无关。于是，我们可以获得线性分组码的译码方法，如图 5-6 所示。先利用收到的码字 $\boldsymbol{R}$ 和已知的校验矩阵 $\boldsymbol{H}$ 计算出 $\boldsymbol{R}\boldsymbol{H}^{\mathrm{T}}$；再利用 $\boldsymbol{R}\boldsymbol{H}^{\mathrm{T}}$ 计算出差错图样 $\boldsymbol{E}$；由 $\boldsymbol{E}$ 和收到的码字 $\boldsymbol{R}$ 可以得到发送码的估计 $\hat{\boldsymbol{C}}$。

图 5-6 译码过程示意图

定义 5.4：伴随式 $\boldsymbol{S}=[s_{n-k-1}\quad\cdots\quad s_1\quad s_0]$是接收码 $\boldsymbol{R}$ 与校验矩阵 $\boldsymbol{H}$ 的转置 $\boldsymbol{H}^{\mathrm{T}}$ 的乘积，$\boldsymbol{S}=\boldsymbol{R}\boldsymbol{H}^{\mathrm{T}}=(\boldsymbol{E}+\boldsymbol{C})\boldsymbol{H}^{\mathrm{T}}=\boldsymbol{E}\boldsymbol{H}^{\mathrm{T}}+\boldsymbol{C}\boldsymbol{H}^{\mathrm{T}}=\boldsymbol{E}\boldsymbol{H}^{\mathrm{T}}$。

其中

$$\begin{aligned}\boldsymbol{S}&=[s_{n-k-1}\quad\cdots\quad s_1\quad s_0]=\boldsymbol{E}\boldsymbol{H}^{\mathrm{T}}\\&=[e_{n-1}\quad\cdots\quad e_1\quad e_0]\begin{bmatrix}h_{(n-k-1)(n-1)}&\cdots&h_0(n-1)\\\vdots&\ddots&\vdots\\h_{(n-k-1)0}&\cdots&h_{00}\end{bmatrix}\end{aligned}$$

为了求解该方程组，可以进一步展开如下：

$$\begin{aligned}s_{n-k-1}&=e_{n-1}h_{(n-k-1)(n-1)}+\cdots+e_0h_{(n-k-1)0}\\&\vdots\\s_1&=e_{n-1}h_{1(n-1)}+\cdots+e_0h_{10}\\s_0&=e_{n-1}h_{0(n-1)}+\cdots+e_0h_{00}\end{aligned}$$

这是一线性欠定方程组，其中变元个数为 n，方程数为 $n-k$。在二元域中，少一个方程将导

致两个解，少两个方程将导致 4 个解，以此类推，共少了 $n-(n-k)=k$ 个方程，于是每个未知数有 2^k 个解。因此，对于每一个确定的 $\boldsymbol{S}$，错误图样 $\boldsymbol{E}$ 有 2^k 个解，那么取哪一个作为解呢？这里采用**概率译码**的处理方法，即把所有 2^k 个解的重量(错误图案 $\boldsymbol{E}$ 中 1 的个数)做比较，选择其中错误图样重量最小者作为 $\boldsymbol{E}$ 的估值。它的理论依据如下：对于 BSC 信道，若错误概率为 p，则长度 n 的码中错 1 位的概率为 $p(1-p)^{n-1}$，错两位的概率为 $p^2(1-p)^{n-2}$，以此类推，错 n 位的概率为 p^n。因为 $p<<1$，必有 $p(1-p)^{n-1}>p^2(1-p)^{n-2}>\cdots>p^n$，所以 $\boldsymbol{S}$ 对应最小重量错误图案 $\boldsymbol{E}$ 的可能性最大。

由于 $\boldsymbol{E}=\boldsymbol{R}+\boldsymbol{C}$，$\boldsymbol{E}$ 的重量最小等同于 $\boldsymbol{R}$、$\boldsymbol{C}$ 间的汉明距离最小，所以二进制的概率译码体现最小距离译码法则，也是最大似然译码法则。

上述的概率译码，每接收一个码字需要解一次线性方程，运算量非常大。由于伴随式的数目是有限的，共 2^{n-k} 个。如果 $n-k$ 不太大，可以换一种思路来考虑问题。预先对不同 $\boldsymbol{S}$ 取值的错误图样 $\boldsymbol{E}$ 求解方程组，再按最大概率译码从 2^k 个解中取出一个最可能的错误图样 $\boldsymbol{E}$。将 $\boldsymbol{S}$ 取值与错误图样 $\boldsymbol{E}$ 相对应，再把发码与错误图样的组合作为可能接收到的码字列表出来，译码时就不必去解方程，而是在上述表格中查找发送码的码矢量，这个表称为标准阵列译码表。

在标准阵列译码表中，将没有任何差错时的接收码 $\boldsymbol{R}$ 放在第 1 行，差错图案为全零，伴随式也为全零。由于对于 (n,k) 线性分组码有 2^k 个码字，所有码表有 2^k 列；由于 (n,k) 线性分组码有 2^{n-k} 个伴随式，所以从第 2 行至 $n+1$ 行为差错重量是 1 的图样，以及由该差错图样导致的接收码，接下来 C_n^2 行为差错重量为 2 的图案，以此类推，直到全部 2^{n-k} 个伴随式有解为止。

【例 5-6】 某(5,2)系统线性码的生成矩阵是 $\boldsymbol{G}=\begin{bmatrix}1&0&1&1&1\\0&1&1&0&1\end{bmatrix}$，设接收的码是 $\boldsymbol{R}=10101$，试构造出标准译码阵列表，并译码发送码的估值 $\hat{\boldsymbol{C}}$。

解：由于该码是(5,2)系统线性码，所有码字的信息序列应该是 $\boldsymbol{m}=00,01,10,11$，这样，通过生成矩阵获得的有效码字是

$$\boldsymbol{C}_1=00000,\quad \boldsymbol{C}_2=10111,\quad \boldsymbol{C}_3=01101,\quad \boldsymbol{C}_4=11010$$

由生成矩阵，可得校验矩阵为

$$\boldsymbol{H}=[\boldsymbol{P}^{\mathrm{T}}\quad\vdots\quad\boldsymbol{I}_3]=\begin{bmatrix}1&1&1&0&0\\1&0&0&1&0\\1&1&0&0&1\end{bmatrix}$$

因此伴随式有 $2^{n-k}=2^{5-2}=8$ 种。差错图样中除了代表无差错的全零图样外，有一个差错的图样有 5 种，两个差错的图样有 $C_5^2=10$ 种，以此类推。根据概率译码准则，8 种伴随式将对应于差错重量最轻的图样，所以应选一种 0 个差错的图样、5 种一个差错图样和两种两个差错的图样。由此，对于 5 种一个差错的图样，可通过 $\boldsymbol{S}=\boldsymbol{E}\boldsymbol{H}^{\mathrm{T}}$ 求出其对应的伴随式；再通过求解剩余伴随式方程组，获得两种两个差错的图样。值得注意的是，对于伴随式 011，将有 2^2 个解 00011、10100、01110 和 11001，重量为 2 的差错图样有两种，可以任意选择其中一种为伴随式 011 的解。因为对于该码来说，它的纠错能力 t 仅为 1，两个差错是纠正不了的，随机选择一种重量为 2 的差错图样不会对译码结果带来影响。对应的标准阵列译码如表 5-1 所示。

表 5-1 (5,2)线性码的标准阵列译码

$S_1=000$	$E_1+C_1=00000$	$C_2=10111$	$C_3=01101$	$C_4=11010$
$S_2=111$	$E_2=10000$	00111	11101	01010
$S_3=101$	$E_3=01000$	11111	00101	10010
$S_4=100$	$E_4=00100$	10011	01001	11110
$S_5=010$	$E_5=00010$	**10101**	01111	11000
$S_6=001$	$E_6=00001$	10110	01100	11011
$S_7=011$	$E_7=00011$	10100	01110	11001
$S_8=110$	$E_8=10001$	10001	01011	11100

对于接收码 10101,直接查表可得它的子集头是 10111,因此译码输出为 10111。

实际上,获得发送码估值 $\hat{\boldsymbol{C}}$ 的方案有 3 种:(1)直接搜索码表;(2)先求伴随式找到行数;(3)先求伴随式,在表中查出对应的差错图案,由 $\boldsymbol{C}=\boldsymbol{R}+\boldsymbol{E}_5$ 得到结果。

任意一个二元(n,k)线性分组码都有 2^{n-k} 个伴随式,假如该码的纠错能力为 t,则对于任何一个重量小于 t 的差错图案,都应有一个伴随式与之对应,所以伴随式的数目应满足条件

$$2^{n-k}\geqslant\binom{n}{0}+\binom{n}{1}+\cdots+\binom{n}{t}=\sum_{i=0}^{t}\binom{n}{i} \tag{5.4.8}$$

上式称为汉明限(Hamming Bound),任何一种纠错能力为 t 的线性码都应满足以上条件。

定义 5.5:若某二元(n,k)线性分组码使式(5.4.8)成立,即 $2^{n-k}=\sum_{i=0}^{t}\binom{n}{i}$,则将$(n,k)$线性分组码称为完备码。

迄今为止,完备码并不多见。$t=1$ 的汉明码,$t=3$ 的格雷码,长度 n 为奇数、由两个码字组成、满足 $d_{\min}=n$ 的任何二进制码,以及三进制的 $t=3$ 的(11,6)码都属于完备码。

汉明码不是指一个码,而是指一类码,它是纠错能力 $t=1$(既有二进制,又有非二进制)的完备的线性分组码的一类码的统称。二进制码长为 n 和信息位为 k 的汉明码服从如下规律:$(n,k)=(2^m-1,2^m-1-m)$,其中,$m=n-k$。当 $m=3,4,5,6,7,8$ 时,对应的汉明码分别为(7,4)、(15,11)、(31,26)、(63,57)、(127,120)和(255,247)。值得注意的是,汉明码是一种完备码。

除此之外,汉明码的校验矩阵 $\boldsymbol{H}$ 具有特殊的性质。一个(n,k)的线性码校验矩阵有 $n-k$ 行 n 列。而二进制 $n-k$ 个码元能组合的列矢量的总数(全零矢量除外)是 $2^{n-k}-1$,恰好与二进制汉明码校验矩阵的列数 $n=2^m-1$ 相等。因此,对于汉明码,只要排列所有 $n-k$ 个码元组成的所有列矢量(全零矢量除外),可构成一个汉明码校验矩阵,再通过初等变换可以获得系统形式校验矩阵 $\boldsymbol{H}$,从而得到生成矩阵。

格雷码是二进制(23,12)线性码,其最小距离 $d_{\min}=7$,纠错能力 $t=3$。由于满足 $2^{23-12}=2048=1+\binom{23}{1}+\binom{23}{2}+\binom{23}{3}$,因此它也是完备码。

定义 5.6:如果给每个码字添加奇偶校验位 $C_{i(n+1)}$ 来对码字的所有比特进行校验,使其满足 $C_{i1}+C_{i2}+\cdots+C_{i(n+1)}$ 等于 0 或 1,就构成了一个二进制$(n+1,k)$线性码,称为扩展码。

在偶校验的情况下,若原来码字中 1 的个数为偶数,则添加的校验位为 0;若原码 1 的个数为奇数,则添加的检验位为 1。于是,扩展码校验矩阵 $\boldsymbol{H}_e$ 可表示为

$$\boldsymbol{H}_{\mathrm{e}}=\left[\begin{array}{c:c}\multicolumn{1}{c:}{\multirow{2}{*}{$\boldsymbol{H}$}} & \begin{matrix}0\\0\\\vdots\\0\end{matrix}\\ \hdashline \begin{matrix}1 & 1 & \cdots & 1\end{matrix} & 1\end{array}\right] \tag{5.4.9}$$

其中，$\boldsymbol{r}_{\mathrm{e}}\boldsymbol{H}_{\mathrm{e}}^{\mathrm{T}}=0$。这时，偶校验后最小距离将增加 1，检错能力也增加 1。例如，在(23,12)格雷码上添加一位奇偶位即可获得(24,12)扩展格雷码，其最小码距 $d_{\min}=8$。

若把一定数量的信息设为零，(n,k)系统码可以缩短。把码的前 e 位设为 0，从而将码(n,k)缩短成码$(n-e,k-e)$。

定义 5.7：在生成矩阵一定的条件下，由于信息组中的 0 和 1 结构对称、奇偶对称，因此码字的第 1 位 $m_1=0$ 的概率将为 0.5。把第 1 位为 1 的所有码字去掉，剩下另一半第 1 位为 0 的码字舍去第 1 位后组成的新的$(n-1,k-1)$系统码，称为缩短码。

缩短码与原码具有相同的最小码距 $d_{\min}$。若(n,k)线性码的编码运算为 $\boldsymbol{C}=\boldsymbol{mG}$，由于缩短码信息组的前一位是 0，因此缩短码的编码运算为

$$\boldsymbol{C}_{\mathrm{s}}=\boldsymbol{m}_{\mathrm{s}}\boldsymbol{G}_{\mathrm{s}} \tag{5.4.10}$$

其中，$\boldsymbol{G}_{\mathrm{s}}$ 是 $\boldsymbol{G}(k\times n)$去掉最左边 l 列及最上面 l 行后剩下的$(n-l)\times(k-l)$矩阵。

另一方面，原码的校验矩阵 $\boldsymbol{H}_{(n-k)\times n}$ 缩短后为$(n-k)\times(n-l)$的 $\boldsymbol{H}_{\mathrm{s}}$，可见校验矩阵 $\boldsymbol{H}$ 与缩短后的校验矩阵 $\boldsymbol{H}_{\mathrm{s}}$ 的行数是一致的，将 $\boldsymbol{H}$ 最左边的 l 列去掉得 $\boldsymbol{H}_{\mathrm{s}}$。由于$(k-i)/(n-i)<k/n$，因此，缩短码的码率总是小于原码。

【例 5-7】 构造一个 $m=3$ 的二元(7,4)汉明码。

解：

$$\boldsymbol{H}=\begin{bmatrix}0&0&0&1&1&1&1\\0&1&1&0&0&1&1\\1&0&1&0&1&0&1\end{bmatrix}\Rightarrow\begin{bmatrix}1&1&1&0&1&0&0\\0&1&1&1&0&1&0\\1&1&0&1&0&0&1\end{bmatrix}$$

$$\boldsymbol{G}=[\boldsymbol{I}_4\quad\vdots\quad\boldsymbol{P}]=\begin{bmatrix}1&0&0&0&1&0&1\\0&1&0&0&1&1&1\\0&0&1&0&1&1&0\\0&0&0&1&0&1&1\end{bmatrix}$$

扩展汉明码：给(n,k)汉明码添加一位奇偶校验位，可得到一个 $d_{\min}=4$ 的$(n+1,k)$扩展汉明码 $\boldsymbol{H}_{\mathrm{e}}=\begin{bmatrix}1&1&1&0&1&0&0&0\\0&1&1&1&0&1&0&0\\1&1&0&1&0&0&1&0\\1&1&1&1&1&1&1&1\end{bmatrix}\Rightarrow\begin{bmatrix}1&1&1&0&1&0&0&0\\0&1&1&1&0&1&0&0\\1&1&0&1&0&0&1&0\\1&0&1&1&0&0&0&1\end{bmatrix}$；反之，在生成矩阵 $\boldsymbol{G}$ 中删除 l 行 l 列，或等效在 $\boldsymbol{H}$ 中删除 l 列，可以得到缩短汉明码$(n-l,k-l)$，例如(6,3)缩短码，其校验矩阵为

$$\boldsymbol{H}_{\mathrm{s}}=\begin{bmatrix}1&1&0&1&0&0\\1&1&1&0&1&0\\1&0&1&0&0&1\end{bmatrix}$$

$$\boldsymbol{G}_{\mathrm{s}}=\begin{bmatrix}1&0&0&1&1&1\\0&1&0&1&1&0\\0&0&1&0&1&1\end{bmatrix}$$

5.4.2 循环码


视频讲解

循环码是线性分组码的一个子集，它满足如下循环移位特性：码集 $\boldsymbol{C}$ 中任何一个码字的循环移位仍是码字。

由于循环码的 k 个基底可以由同一个基底循环 k 次得到，因此用一个基底就足以表示一个码的特性。于是，可以把码字 $\boldsymbol{C}=[c_{n-1}c_{n-2}\cdots c_1c_0]$ 与一个不大于 $n-1$ 次的多项式联系起来，称为码多项式 $C(x)$，其定义为

$$C(x)=c_{n-1}x^{n-1}+c_{n-2}x^{n-2}+\cdots+c_1x+c_0 \tag{5.4.11}$$

其中，$c_i\in\{0,1\}$，$i=0,1,\cdots,n-1$。

根据循环码的定义，码循环移一位后可表示为 $[c_{n-1},c_{n-2},\cdots,c_1,c_0]\to[c_{n-2},\cdots,c_1,c_0,c_{n-1}]$，其对应的码多项式应为 $C_1(x)=c_{n-2}x^{n-1}+c_{n-3}x^{n-2}+\cdots+c_0x+c_{n-1}$。比较循环移位的前后，可用如下的多项式运算来表示循环移 1 位：$C_1(x)=xC_0(x) \bmod (x^n+1)$。

于是，以此类推，可以得到移 2 位至移 $n-1$ 位的码多项式分别如下。

移 2 位：$C_2(x)=xC_1(x)=x^2C_0(x) \bmod (x^n+1)$；

⋮

移 $n-1$ 位：$C_{n-1}(x)=xC_{n-2}(x)=x^{n-1}C_0(x) \bmod (x^n+1)$。

码字 $C_0(x)$ 移 n 位后又回到码字 $C_0(x)$，一个码字的移位最多能得到 n 个码字，再根据码空间的封闭性，得到码字的线性组合仍是码字，即

$$\begin{aligned}C(x)&=a_0C_0(x)+a_1xC_0(x)+\cdots+a_{n-1}x^{n-1}C_0(x)\\&=(a_0+a_1x+\cdots+a_{n-1}x^{n-1})C_0(x)\\&=A(x)C_0(x) \bmod (x^n+1)\end{aligned} \tag{5.4.12}$$

其中，$C_0(x)$ 是一个码多项式，$A(x)$ 是次数不大于 $n-1$ 的任意多项式，$a_i\in\{0,1\}$，$i=0,1,\cdots,n-1$（对于二进制编码）。作为特殊情形，若选择 $C_0(x)$ 是 $n-k$ 次多项式，$A(x)$ 是 $k-1$ 次任意多项式。那么，在 $C_0(x)$ 不变情况下，$A(x)$ 系数有 2^k 种组合，它恰好对应于 2^k 个码字。此时 $C_0(x)$ 起到了生成多项式的作用，且 $C_0(x)$ 是 $C(x)$ 的一个因式。

从近世代数的观点来看，GF(2)域次数小于 n 的多项式在模 2 加、模 (x^n+1) 乘法运算下构成了一个交换环。多项式交换环的一个主理想子环一定可以产生一个循环码，且所有码多项式都可以由其中一个元素的倍式组成，这个元素称为该主理想子环的生成元。从交换环的性质中，可以找到构造 (n,k) 循环码的步骤。

(1) 对 x^n+1 做因式分解，找出其 $(n-k)$ 次因式。

(2) 以该 $(n-k)$ 次因式作为生成多项式 $g(x)$，与不高于 $(k-1)$ 次的信息多项式 $m(x)$ 相乘，即得到码多项式 $C(x)=m(x)g(x)$，其中，$C(x)$ 的次数不高于 $(k-1)+(n-k)=n-1$。

可以验证所得码的循环性。令 $C_1(x)=xC(x)=xm(x)g(x) \bmod (x^n+1)$，由于 $g(x)$ 本身也是码多项式，而 $xm(x)$ 是不高于 k 次的多项式，由式(5.4.12)得到 $C_1(x)$ 一定是码字，即码字的循环移位也是码字，所以是循环码。

【例 5-8】 构造 $n=7$ 的循环码。

解：$x^7+1=(x+1)(x^3+x^2+1)(x^3+x+1)$。

一次因式 1 种：$x+1$；

三次因式 2 种：(x^3+x^2+1)，(x^3+x+1)；

四次因式 2 种：$(x^3+x^2+1)(x+1)=x^4+x^2+x+1$，

$(x^3+x+1)(x+1)=x^4+x^3+x^2+1$；

六次因式 1 种：$(x^3+x^2+1)(x^3+x+1)=x^6+x^5+x^4+x^3+x^2+x+1$。

以 $g(x)=x^3+x+1$ 为生成多项式可以生成(7,4)循环码。当信息位为[0110]时，得到信息多项式为 $m(x)=x^2+x$，所以对应的码多项式为 $C(x)=m(x)g(x)=(x^2+x)(x^3+x+1)=x^5+x^4+x^3+x\rightarrow[0111010]$。

多项式 x^n+1 因式分解为 $g(x)h(x)$，去除生成多项式 $g(x)$ 外，剩下的因式组成 $h(x)$，称为校验多项式。因为任何码多项式 $C(x)$ 与 $h(x)$ 的模 (x^n+1) 的乘积一定等于 0，而非码字与 $h(x)$ 的模 (x^n+1) 的乘积必不为 0。

$$C(x)h(x)=m(x)g(x)h(x)=m(x)(x^n+1) \bmod (x^n+1)=0 \tag{5.4.13}$$

在 $x^n+1=g(x)h(x)$ 分解中，$g(x)$ 和 $h(x)$ 处于同等地位。由 $g(x)$ 生成的 (n,k) 循环码和由 $h(x)$ 生成的 $(n,n-k)$ 循环码互为对偶码。$(n,n-k)$ 对偶码构成 (n,k) 循环码的零空间。

以上方法获得的循环码并非是系统码。如果希望所获得的循环码同时是系统码，可通过系统码的定义形式直接获得系统循环码的构造方法。

对于系统码而言，其码字的前 k 位应是信息位，后 $n-k$ 位为校验位。因此，其码字多项式应为

$$C(x)=x^{n-k}m(x)+r(x) \tag{5.4.14}$$

其中，$r(x)$ 是校验多项式，它的最高次应为 $n-k-1$。对上式两边同取模 $g(x)$，左边 $C(x) \bmod g(x)=m(x)g(x) \bmod g(x)=0$；右边也必是 0，即 $x^{n-k}m(x)+r(x) \bmod g(x)=x^{n-k}m(x) \bmod g(x)+r(x) \bmod g(x)=0$，因此可得 $x^{n-k}m(x) \bmod g(x)=r(x) \bmod g(x)$。

所以，校验多项式应为

$$r(x)=x^{n-k}m(x) \bmod g(x) \tag{5.4.15}$$

于是获得了直接产生系统循环码的方法，其步骤如下。

(1) 将消息多项式 $m(x)$ 乘以 x^{n-k}。

(2) 将 $x^{n-k}m(x)$ 除以 $g(x)$ 得校验多项式 $r(x)$。

(3) 将 $r(x)$ 加在 $x^{n-k}m(x)$ 后面得到系统循环码多项式。

【例 5-9】 (7,4)循环码的生成多项式为 $g(x)=x^3+x+1$，求：

(1) 该循环码系统形式的生成矩阵。

(2) [1001]的系统循环码字。

解：

(1) $\boldsymbol{G}=\begin{bmatrix} x^3g(x) \\ x^2g(x) \\ xg(x) \\ g(x) \end{bmatrix}=\begin{bmatrix} 1&0&1&1&0&0&0 \\ 0&1&0&1&1&0&0 \\ 0&0&1&0&1&1&0 \\ 0&0&0&1&0&1&1 \end{bmatrix} \xrightarrow{\text{系统化}} \begin{bmatrix} 1&0&0&0&1&0&1 \\ 0&1&0&0&1&1&1 \\ 0&0&1&0&1&1&0 \\ 0&0&0&1&0&1&1 \end{bmatrix}$

(2) $m(x)=m_3x^3+m_2x^2+m_1x+m_0=x^3+1$

$x^{n-k}m(x)=x^6+x^3$

$r(x)=x^{n-k}m(x) \bmod g(x)=(x^6+x^3) \bmod (x^3+x+1)=x^2+x$

$C(x)=x^{n-k}m(x)+r(x)=x^6+x^3+x^2+x\Rightarrow[1001110]$

循环码是线性分组码的子集。从(n,k)循环码生成多项式也可获得生成矩阵的方法：取$g(x)$本身加上移位$k-1$次所得的$k-1$个码字作为k个基底。

若循环码生成多项式为$g(x)=x^{n-k}+g_{n-k-1}x^{n-k-1}+\cdots+g_2x^2+g_1x+1$，则生成矩阵是

$$\boldsymbol{G}=\begin{bmatrix}x^{k-1}g(x)\\x^{k-2}g(x)\\\vdots\\xg(x)\\g(x)\end{bmatrix}=\begin{bmatrix}1&g_{n-k-1}&\cdots&g_2&g_1&1&0&0&\cdots\\\vdots&1&g_{n-k-1}&\vdots&g_2&g_1&1&0&\cdots\\\vdots&\vdots&\ddots&\ddots&\ddots&\ddots&\ddots&\ddots&0\\0&0&\cdots&1&g_{n-k-1}&\cdots&g_2&g_1&1\end{bmatrix}\tag{5.4.16}$$

除此之外，也可以直接获得系统生成矩阵，其思路与求系统循环码类似。对于系统生成矩阵$\boldsymbol{G}=[\boldsymbol{I}_k\quad\vdots\quad\boldsymbol{P}]$，它的第$l$行可写成码多项式$x^{n-l}+r_l(x)$，每行多项式$x^{n-l}+r_l(x)$也一定是循环码的一个码字，即$x^{n-l}+r_l(x)=\theta_l(x)g(x)$，且各行的码字不相关。因此，如果能计算获取$r_l(x)$，则可直接获得系统生成矩阵。

当对$x^{n-l}+r_l(x)=\theta_l(x)g(x)$两边同除以生成多项式$g(x)$时，可以得到

$$r_l(x)=x^{n-l}\bmod g(x)\tag{5.4.17}$$

其中，$l=1,2,\cdots,k-1$。

【例 5-10】 计算$(7,4)$系统循环汉明码最小重量的可纠错误图样和对应的伴随式。

解：对于$(7,4)$系统循环汉明码，其最小重量的可纠错误图样码多项式为$0,1,x,x^2,x^3,x^4,x^5,x^6$。由$s(x)=E(x)\bmod g(x)$，可得8个对应的伴随式：$0,1,x,x^2,x+1,x^2+x,x^2+x+1,x^2+1$。


视频讲解

5.4.3 卷积码

前面研究过的分组码是将输入序列分割成一定长度的信息组后各自孤立地进行编译码，分组与分组之间没有考虑任何联系。从信息论的角度，它忽略了各分组间的相关性，必将损失一部分信息。

当码长n有限时，能否将有限个分组间的相关信息添加到码字中从而等效地增加码长？译码时能否利用前后码字的相关性将前面的译码信息反馈到后面译码中，作为译码参考？这些想法导致了卷积码的产生。

卷积码于1955年由麻省理工学院Elias首次提出，随后Wozencraft和Renffen提出了序贯译码方法(对具有较大约束长度的卷积码非常有效)。1963年，Massey提出了一种效率不高、但易于实现的译码方法，称为门限译码，这使得卷积码被大量应用于卫星和无线信道的数字传输中。到了1967年，Viterbi又提出了最大似然概率译码方法，它可应用于较小约束长度的卷积码。再配合序贯译码的软判决，卷积码在深空和卫星通信系统中也得到广泛应用。1974年，Bahl、Cocke、Jelinek和Raviv提出最大后验概率译码方法，称为BCJR算法，它可对不等先验概率的信息比特进行卷积码译码。卷积码是一种常见的信道编码方法，下面对卷积码的编译码方法进行简要阐述。

1. 卷积码的编码

卷积码可看作一个有限记忆系统，它将信息序列分割成长度为k的一个个分组，与分组码不同的是，在某一个分组编码时，卷积码的码字不仅与本时刻的分组信息相关，而且与本时刻以前的L个分组的信息也相关，其中，$L+1$称为卷积码的约束长度。于是，卷积码常

用 3 个参量描述，如(n,k,L)，其中，n 代表卷积码输出码字的长度，k 为每次输入卷积码编码器的信息组长度，$L+1$ 为约束长度。因此，卷积码中相互关联的码元长度为 $n\times(L+1)$，它的纠错性能随 L 的增加而增大，译码的差错率将随着 n 的增加而指数下降。在相同编码效率情况下，卷积码的性能明显优于分组码，至少不低于分组码。如图 5-7 所示是卷积码编码器示意图，信息序列被分割成一个个分组，通过卷积码编码器进行线性组合，最后获得输出码字。

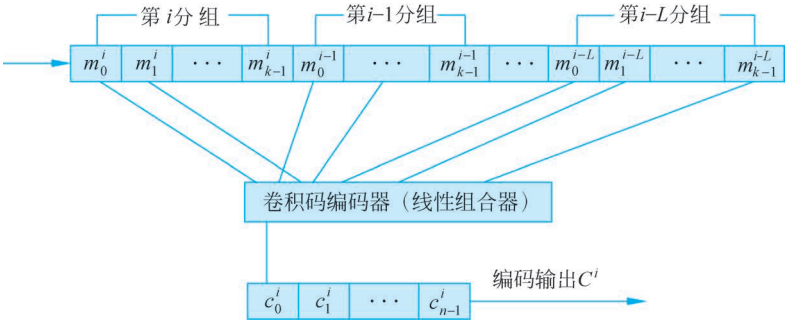


图 5-7 卷积码编码器示意图

卷积码编码的一般过程可用图 5-8 表示。卷积码编码器将信息序列通过串并转换后，存入 k 个长度为 $L+1$ 的移存器中，构成输入信息阵列；编码器按照一定的规则对信息阵列中数据进行组合，获得输出码字信息，并通过串并转换后得到编码输出。由于卷积码码字中每个元素都是 $k\times(L+1)$ 个数据的线性组合，编出当前 i 时刻的码元(码字母)C_j^i，$j=0,1,\cdots,n-1$，需要有 $k\times(L+1)$ 个系数来描述组合规则，且每一个码字需用 $n\times k\times(L+1)$ 个系数才能描述。

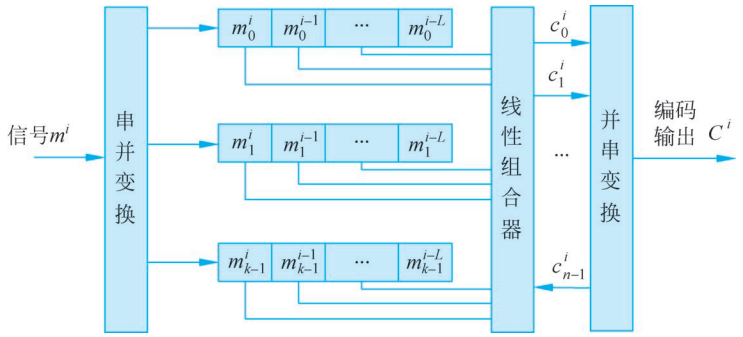


图 5-8 卷积码编码过程的一般示意图

与分组码不同，卷积码在任意给定单元时刻其输出的 n 个码元中，每一个码元不仅与此时刻输入的 k 个信息元有关，而且还与前连续 L 个时刻输入的信息元有关。但是，卷积码也可以用生成矩阵形式描述。下面以(3,2,1)卷积码为例，说明卷积码的编码过程。

【例 5-11】 二进制(3,2,1)卷积码编码器的结构如图 5-9 所示，若本时刻 $i=1$ 的输入信息是 $\boldsymbol{m}^1=[m_0^1,m_1^1]=01$，上一时刻 $\boldsymbol{m}^0=[m_0^0,m_1^0]=10$(用上标正整数 1 表示 1 个时延)，试用生成矩阵表示该编码器，并计算输出码字。

解：假设用 g_{kj}^l 表示记忆序列第 k 行($k=0,1$)、第 l 列($l=0,1$)对第 j 个($j=0,1,2$)码元的影响。令参与组合者(有连线接到模 2 加法器)相应的系数 $g_{kj}^l=1$，否则 $g_{kj}^l=0$。

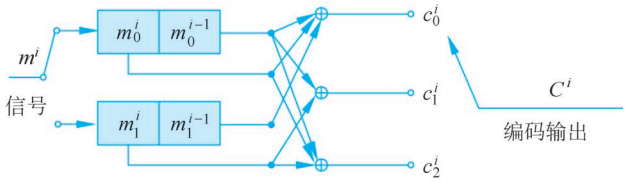


图 5-9 二进制(3,2,1)卷积码编码器

由图中接线可知,共需 $n\times k\times(L+1)=3\times2\times(1+1)=12$ 个系数,分别如下:

$$g_{00}^0=1,\quad g_{01}^0=0,\quad g_{02}^0=1,\quad g_{00}^1=1,\quad g_{01}^1=1,\quad g_{02}^1=1$$

$$g_{10}^0=0,\quad g_{11}^0=1,\quad g_{12}^0=1,\quad g_{10}^1=1,\quad g_{11}^1=0,\quad g_{12}^1=0$$

本时刻输入信息 $\boldsymbol{m}^1=[m_0^1,m_1^1]=01$ 与上时刻输入信息 $\boldsymbol{m}^0=[m_0^0,m_1^0]=10$,用矩阵 $\boldsymbol{G}^0=\begin{bmatrix}g_{00}^0 & g_{01}^0 & g_{02}^0\\ g_{10}^0 & g_{11}^0 & g_{12}^0\end{bmatrix}=\begin{bmatrix}1 & 0 & 1\\ 0 & 1 & 1\end{bmatrix}$,$\boldsymbol{G}^1=\begin{bmatrix}g_{00}^1 & g_{01}^1 & g_{02}^1\\ g_{10}^1 & g_{11}^1 & g_{12}^1\end{bmatrix}=\begin{bmatrix}1 & 1 & 1\\ 1 & 0 & 0\end{bmatrix}$ 分别描述本时刻和上一时刻输入对码字的影响,从而得出本时刻编码的输出为

$$\begin{aligned}\boldsymbol{C}^1&=[C_0^1,C_1^1,C_2^1]=\boldsymbol{m}^1\boldsymbol{G}^0+\boldsymbol{m}^0\boldsymbol{G}^1\\&=[01]\begin{bmatrix}1 & 0 & 1\\ 0 & 1 & 1\end{bmatrix}+[10]\begin{bmatrix}1 & 1 & 1\\ 1 & 0 & 0\end{bmatrix}=011+111=100\end{aligned}$$

上例中,系数矩阵 $\boldsymbol{G}^0$ 和 $\boldsymbol{G}^1$ 的设定具有一般性。对于(n,k,L)卷积码,把 i 以前的第 l 个信息比特组 $\boldsymbol{m}^l=(m_0^{i-l},m_1^{i-l},\cdots,m_{k-1}^{i-l})$对线性组合的影响用一个 $k\times n$ 的生成子矩阵 $\boldsymbol{G}^l$ 来表示,则

$$\boldsymbol{G}^l=\begin{bmatrix}g_{00}^l & g_{01}^l & \cdots & g_{0(n-1)}^l\\ \vdots & \vdots & \ddots & \vdots\\ g_{(k-1)0}^l & g_{(k-1)0}^l & \cdots & g_{(k-1)(n-1)}^l\end{bmatrix}\tag{5.4.18}$$

其中,用 g_{kj}^l 表示记忆序列第 k 行$(k=0,1,\cdots,k-1)$、第 l 列$(l=0,1,\cdots,L)$对第 j 个$(j=0,1,2,\cdots,n-1)$输出码元的影响,且 $g_{kj}^l\in\{0,1\}$。设编码器初始状态为 0(记忆阵列全体清零),则随着一个个 k 比特信息组$(m^0,m^1,\cdots,m^L,m^{L+1},\cdots)$的输入,编码器源源不断地输出码字$(C^0,C^1,\cdots,C^L,C^{L+1},\cdots)$,即

在时刻 $i=0$ 时,$\boldsymbol{C}^0=\boldsymbol{m}^0\boldsymbol{G}^0$;

在时刻 $i=1$ 时,$\boldsymbol{C}^1=\boldsymbol{m}^1\boldsymbol{G}^0+\boldsymbol{m}^0\boldsymbol{G}^1$;

在时刻 $i=2$ 时,$\boldsymbol{C}^2=\boldsymbol{m}^2\boldsymbol{G}^0+\boldsymbol{m}^1\boldsymbol{G}^1+\boldsymbol{m}^0\boldsymbol{G}^2$;

以此类推,在时刻 $i=L$ 时,$\boldsymbol{C}^L=\boldsymbol{m}^L\boldsymbol{G}^0+\boldsymbol{m}^{L-1}\boldsymbol{G}^1+\cdots+\boldsymbol{m}^0\boldsymbol{G}^L$,其等效矩阵形式为

$$\begin{aligned}\boldsymbol{C}&=[\boldsymbol{C}^0,\boldsymbol{C}^1,\cdots,\boldsymbol{C}^{L+1}]=\boldsymbol{m}\boldsymbol{G}\\&=[\boldsymbol{m}^0,\boldsymbol{m}^1,\boldsymbol{m}^2,\cdots]\begin{bmatrix}\boldsymbol{G}^0 & \boldsymbol{G}^1 & \cdots & \boldsymbol{G}^L & \cdots & \cdots\\ 0 & \boldsymbol{G}^0 & \boldsymbol{G}^1 & \cdots & \boldsymbol{G}^L & \cdots\\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots\end{bmatrix}\end{aligned}\tag{5.4.19}$$

其中,$\boldsymbol{G}$ 定义为卷积码的生成矩阵,它是个半无限矩阵。时刻 i 的输出码字为

$$\boldsymbol{C}^i=\sum_{l=0}^{L}\boldsymbol{m}^{i-l}\boldsymbol{G}^l\tag{5.4.20}$$

即码字为无限长输入序列与有限长矩阵元素的卷积运算获得，这也就是**卷积码**名称的由来。

其中，$L+1$ 个子矩阵 $\boldsymbol{G}^l(l=0,1,\cdots,L)$ 实质上是 $\boldsymbol{G}$ 在时间轴上的展开。考虑子矩阵 $\boldsymbol{G}^l$ 和 $\boldsymbol{G}^{l+1}$ 同一位置上的两系数 g_{kj}^l 和 g_{kj}^{l+1} 都表示记忆阵列第 k 行对第 j 输出码元的影响，两者之间仅差一列(即一个时延 D)，因此，也可以用多项式形式来表示卷积码的编码过程，其中，D 为时间延迟。第 k 行输入到第 j 个输出之间的转移函数 $g_{kj}(D)$ 表示为

$$g_{kj}(D)=g_{kj}^0+g_{kj}^1D+g_{kj}^2D^2+\cdots+g_{kj}^LD^L=\sum_{l=0}^{L}g_{kj}^lD^l \tag{5.4.21}$$

由转移函数构成的矩阵 $\boldsymbol{G}(D)=\{g_{kj}(D)\}$ 称为转移函数矩阵，它是 k 行 n 列矩阵。一旦卷积码器结构图给定，转移函数矩阵 $\boldsymbol{G}(D)$ 也就确定。同样，当转移函数矩阵 $\boldsymbol{G}(D)$ 给定，卷积码器的结构图也是确定的。转移函数矩阵同样可以描述卷积码的编码。

【例 5-12】 某二元(3,1,2)卷积码的转移函数矩阵为 $\boldsymbol{G}(D)=[1\quad 1+D\quad 1+D+D^2]$，试画出编码器结构图。

解：根据转移函数矩阵定义，得到

$g_{00}(D)=g_{00}^0+g_{00}^1D+g_{00}^2D^2=1$，所以 $g_{00}^0=1\quad g_{00}^1=0\quad g_{00}^2=0$

$g_{01}(D)=g_{01}^0+g_{01}^1D+g_{01}^2D^2=1+D$，所以 $g_{01}^0=1\quad g_{01}^1=1\quad g_{01}^2=0$

$g_{02}(D)=g_{02}^0+g_{02}^1D+g_{02}^2D^2=1+D+D^2$，所以 $g_{02}^0=1\quad g_{02}^1=1\quad g_{02}^2=1$

根据 g_{kj}^l 值，得到卷积码的编码器结构图如图 5-10 所示。

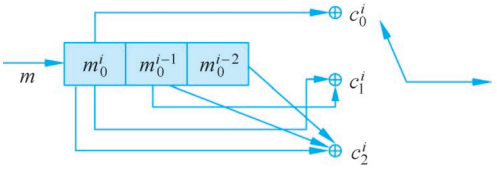


图 5-10　二进制(3,1,2)卷积码编码器

由于编码器是一个线性系统，$\boldsymbol{m}^i(D)$ 表示第 i 个输入序列，$\boldsymbol{C}^j(D)$ 表示第 j 个输出序列，转移函数 $g_{kj}(D)$ 可以表示从输入 i 到输出 j 的转移。对于有 k 个输入、n 个输出的线性系统，共有 kn 个转移函数，可用 $k\times n$ 矩阵表示，转移函数生成矩阵表示为

$$\boldsymbol{G}(D)=\begin{bmatrix} g_{00}(D) & g_{01}(D) & \cdots & g_{0(n-1)}(D) \\ \vdots & \vdots & \ddots & \vdots \\ g_{(k-1)0}(D) & g_{(k-1)0}(D) & \cdots & g_{(k-1)(n-1)}(D) \end{bmatrix} \tag{5.4.22}$$

其中，$g_{ij}(D)$ 由式(5.4.21)所得。

基于转移函数生成矩阵，(n,k,L) 编码器的编码过程也可用多项式形式表示。因此，要首先获取信息多项式，对于信息序列 $\boldsymbol{m}=[m_0,m_1,\cdots,m_k,m_{k+1},\cdots,m_{2k},m_{2k+1},\cdots]$，其信息多项式为

$$m(D)=m_0+m_1D+\cdots+m_kD^k+m_{k+1}D^{k+1}+\cdots+m_{2k}D^{2k}+m_{2k+1}D^{2k+1}+\cdots$$

串/并变换后得到 k 个子多项式 $\boldsymbol{m}^i(D)$，$i=0,\cdots,k-1$：

$$\begin{aligned} m^0(D)&=m_0+m_kD+m_{2k}D^2+\cdots \\ m^1(D)&=m_1+m_{k+1}D+m_{2k+1}D^2+\cdots \\ m^2(D)&=m_2+m_{k+2}D+m_{2k+2}D^2+\cdots \\ &\vdots \end{aligned}$$

卷积编码器可视作一个 k 端口入、n 端口出的线性多端口网络，基于转移函数矩阵的编码过程如图 5-11 所示，其中，$\boldsymbol{G}(D)$是 $k\times n$ 矩阵，每矩阵元素是 L 次多项式，$m(D)$、$\boldsymbol{C}(D)$则是无限高次。

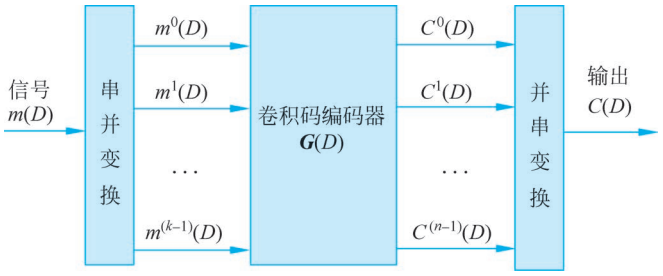


图 5-11 卷积码的转移函数矩阵

定义输入、输出多项式矩阵 $\boldsymbol{m}_P(D)$、$\boldsymbol{C}_P(D)$为

$$m(D)\xrightarrow{\text{串/并}}\boldsymbol{m}_P(D)=[m^0(D),\ m^1(D),\cdots,m^{k-1}(D)]$$

$$C(D)\xleftarrow{\text{串/并}}\boldsymbol{C}_P(D)=[C^0(D),\ C^1(D),\cdots,C^{n-1}(D)]$$

这里，$\boldsymbol{m}_P(D)$、$\boldsymbol{C}_P(D)$的下标 P 表示“并行”。虽然 $m(D)$和 $\boldsymbol{m}_P(D)$数量上有对应关系，然而在数学上有不同含义，$m(D)$、$C(D)$是多项式，而 $\boldsymbol{m}_P(D)$、$\boldsymbol{C}_P(D)$分别是$(1\times k)$、$(1\times n)$的矩阵，且输入、输出多项式矩阵间的关系为

$$\boldsymbol{C}_P(D)=\boldsymbol{m}_P(D)\cdot\boldsymbol{G}(D) \tag{5.4.23}$$

同时，第 j 个支路的输出是所有输入支路对它影响的总和，即

$$C^j(D)=\sum_{i=0}^{k-1}m^i(D)g_{ij}(D) \tag{5.4.24}$$

最后，利用并/串变换公式将 n 个输出多项式合并成一个多项式

$$C(D)=\sum_{j=0}^{n-1}D^jC^j(D^n) \tag{5.4.25}$$

【例 5-13】 某二元(2,1,3)卷积码编码器，其转移函数矩阵为 $\boldsymbol{G}(D)=[1+D+D^3\quad 1+D+D^2+D^3]$，若输入的比特信息为 10111，即输入序列的 $m(D)=1+D^2+D^3+D^4$，求输出码字序列。

解：二元(2,1,3)卷积码的转移函数矩阵为 $\boldsymbol{G}(D)=[1+D+D^3\quad 1+D+D^2+D^3]$，则输出码字为

$$\begin{aligned}\boldsymbol{C}_p(D)&=\boldsymbol{m}_p(D)\cdot\boldsymbol{G}(D)=(1+D^2+D^3+D^4)[1+D+D^3\quad 1+D+D^2+D^3]\\&=[1+D+D^2+D^3+D^6+D^7\quad 1+D+D^3+D^4+D^5+D^7]\end{aligned}$$

那么，经过并/串变换后的输出为

$$\begin{aligned}C(D)=\sum_{j=0}^{n-1}D^jC^j(D^n)&=1+(D^2)+(D^2)^2+(D^2)^3+(D^2)^6+(D^2)^7+\\&D[1+(D^2)+(D^2)^3+(D^2)^4+(D^2)^5+(D^2)^7]\\&=1+D+D^2+D^3+D^4+D^6+D^7+D^9+D^{11}+D^{12}+D^{14}+D^{15}\end{aligned}$$

即输出码字为 1111101101011011。

【例 5-14】 某二元(3,2,1)卷积码如图 5-9 所示,当输入序列 $\boldsymbol{m}=110110$ 时,试计算输出码字。

解:针对图 5-9 所示卷积码编码器,其转移函数矩阵可表示为

$$\boldsymbol{G}(D)=\begin{bmatrix}1+D & D & 1+D\\ D & 1 & 1\end{bmatrix}$$

对于输入序列 $\boldsymbol{m}=110110$,则 $m^0(D)=1+D^2$,$m^1(D)=1+D$,于是

$$\boldsymbol{C}_p(D)=\boldsymbol{m}_p(D)\cdot\boldsymbol{G}(D)=\begin{bmatrix}1+D^2 & 1+D\end{bmatrix}\begin{bmatrix}1+D & D & 1+D\\ D & 1 & 1\end{bmatrix}$$

$$=\begin{bmatrix}1+D^3 & 1+D^3 & D^2+D^3\end{bmatrix}$$

$$C(D)=\sum_{j=0}^{n-1}D^jC^j(D^n)=C^0(D^3)+DC^1(D^3)+D^2C^2(D^3)$$

$$=1+D+D^8+D^9+D^{10}+D^{11}$$

即输入码字序列为 $\boldsymbol{c}=110000001111$。


视频讲解

另一方面,状态图和网格图给出了卷积码内存结构的描述,又称为卷积码的图形表示。由图 5-8 卷积码结构一般结构示意图可知,卷积编码器在 i 时刻编出的码字不仅取决于本时刻的输入信息组 $\boldsymbol{m}^i$,而且取决于 i 之前存储在记忆阵列的 L 个信息组,即取决于记忆阵列的内容。我们把卷积码编码器的记忆阵列中任一时刻所存储的信息称为卷积码编码器的一个状态。对于(n,k,L)卷积码,共有 $2^{(k*L)}$ 个状态,每次输入 k 比特只有 2^k 种状态变化,所以,每个状态只能转移到全部状态的某个子集(2^k 个状态)中去。当然,每个状态也只能由全部状态的某个子集(2^k 个状态)转移而来。下面,以图 5-12 所示的(2,1,2)卷积码编码器为例,讨论卷积码的图形表示法。

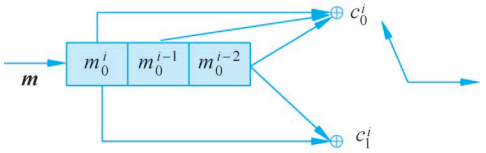


图 5-12 (2,1,2)卷积码编码器

图 5-12 所示的(2,1,2)卷积码包含两个记忆单元(移位寄存器)和两个模 2 加法器。2 级移位寄存器共有 2^2 种不同状态,定义为 $S_0(00)$、$S_1(01)$、$S_2(10)$和 $S_3(11)$共 4 种状态。在每个时刻,输入的 1 比特信息,当前状态将转为 4 种状态中的任何一种。例如,若当前移存器状态为 00,当输入为 1 时,移位寄存器的状态变为 10,输出为 $c_0=1\oplus0\oplus0=1$,$c_1=1\oplus0=1$,即输出 11。表 5-2 描述了该编码器所有状态间转移情况。

表 5-2 卷积码编码器的状态间转移情况

输 入	当前状态	下个状态	输 出 1	输 出 2
0	00	00	0	0
0	01	00	1	1
0	10	01	1	0
0	11	01	0	1
1	00	10	1	1
1	01	10	0	0
1	10	11	0	1
1	11	11	1	0

状态表类似查找表，即根据当前的输入和当前的状态，可以从表中查得输出信息。比表更为简单和直观的方法是状态流图。下面，进一步研究编码器(2,1,2)的状态流图，又称状态图，如图 5-13 所示图中状态转移信息用“输入输出 1 输出 2”表示。

设输入信息序列 $\boldsymbol{m}=10111000\cdots$，按照输入数据可以得到以下不同情况。

(1) 首先对所有移位寄存器进行归 0 处理，此时寄存器的初始化状态为 00。

(2) 输入信息 1，移位寄存器的状态改为 10；输出两个支路分别为 $c_0^1=1\oplus0\oplus0=1$，$c_0^2=1\oplus0=1$，最后输出码组 $\boldsymbol{c}_0=[c_0^1c_0^2]=11$。

(3) 输入的第 2 个信息比特为 0，则寄存器状态转为 01，输出码组 $c_1^1=1$，$c_1^2=0$，输出码组 $\boldsymbol{c}_1=[c_1^1c_1^2]=10$。

(4) 输入的第 3 个信息比特为 1，则寄存器状态转为 10，输出码组 $c_2^1=0$，$c_2^2=0$，故 $\boldsymbol{c}_2=00$。

(5) 输入的第 4 个信息比特为 1，则寄存器状态转为 11，输出码组 $c_3^1=0$，$c_3^2=1$，故 $\boldsymbol{c}_3=01$。

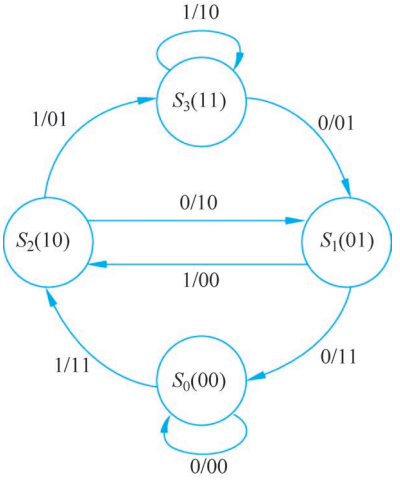


图 5-13 编码器(2,1,2)的状态流图

(6) 输入的第 5 个信息比特为 1 ，则寄存器状态转为 11，输出码组 $c_4^1=1$，$c_4^2=0$，故 $\boldsymbol{c}_4=10$。

(7) 输入的第 6 个信息比特为 0，则寄存器状态转为 01，输出码组 $c_5^1=0$，$c_5^2=1$，故 $\boldsymbol{c}_5=01$。

(8) 输入的第 7 个信息比特为 0，则寄存器状态转为 00，输出码组 $c_6^1=1$，$c_6^2=1$，故 $\boldsymbol{c}_6=11$。

(9) 输入的第 8 个信息比特为 0，则寄存器状态转为 00，输出码组 $c_7^1=0$，$c_7^2=0$，故 $\boldsymbol{c}_3=00$。

以此类推，就可以得到所有情况编码情况，一个完整的(2,1,2)卷积码编码器状态转移图如图 5-13 所示，图中圆圈代表状态，共有 $S_0(00)$、$S_1(01)$、$S_2(10)$和 $S_3(11)$共 4 种状态，箭头代表转移，与箭头对应的标注，如 0/10 表示输入信息 0 时输出码字 10。每个状态都有两个箭头发出，对应于输入是 0 或 1 的两种情况下的转移路径。假如输入信息序列是 $\boldsymbol{m}=10111000\cdots$，从状态流图可以容易地找到输入/输出和状态的转移。可从状态 S_0 出发，根据输入找到相应箭头，随箭头在状态流图上移动，得到编码输出 $\boldsymbol{C}=[11\quad10\quad00\quad01\quad10\quad01\quad11\quad00\quad\cdots]$，具体过程如图 5-14 所示。

$$\xrightarrow{S_0}\ 1/11\ \xrightarrow{S_2}\ 0/10\ \xrightarrow{S_1}\ 1/00\ \xrightarrow{S_2}\ 1/01\ \xrightarrow{S_3}\ 1/10\ \xrightarrow{S_3}\ 0/01\ \xrightarrow{S_1}\ 0/11\ \xrightarrow{S_0}\ 0/00\cdots\cdots$$

图 5-14 卷积码状态及输入输出关系图

由此可见，状态图可为利用信号流图的数学工具奠定基础，但是状态图缺少时间轴，不能记录下状态转移的轨迹。网格图(也称为格栅图、格子图、篱笆图)弥补了这一个缺点，它以状态为纵轴，以时间(单位为码字周期 T)为横轴，将状态转移沿时间轴展开，从而使编码过程一目了然。网格图有助于发现卷积码的性能特征，在卷积码的概率译码中，特别是维特

比(Viterbi)译码算法中特别有用,是借助计算机分析研究卷积码的最得力工具之一。

网格图分成两部分:一部分是对编码器的描述,告诉人们从本时刻的各状态可以转移到下一时刻的哪些状态,伴随转移的输入信息/输出码字是什么;另一部分是对编码过程的记录,一根半无限的水平线(纵轴上的常数)标志某一个状态,一个箭头代表一次转移,每隔时间 T(相当于移存器中的一位时延 D)转移一次,转移的轨迹称为路径。两部分可以合画在一起,也可单独画,如在描述卷积编码器本身而并不涉及具体编码时,只需第一部分网格图就够了。下面仍以(2,1,2)卷积码为例,当节点级数为 $j=L+1=2+1=3$ 时,状态 $S_0(00)$、$S_1(01)$、$S_2(10)$和 $S_3(11)$将呈现重复,利用这一重复,就可得到纵深宽带(或称高度)为 $2^{km}=2^{1\times 2}=4$ 的格状图,如图 5-15 所示,图中实线表示输入 1 时所走的支路,虚线表示输入 0 时所走的分支,从第 3 级节点开始,从同一状态出发所延伸的结构是完全一样的,所以网格图能更为简洁地表示卷积码。

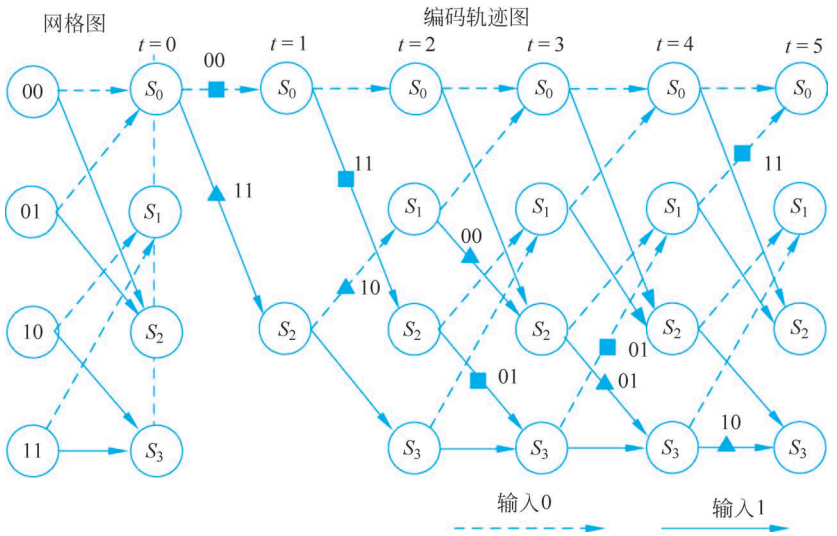


图 5-15 编码器(2,1,2)的网格图

任意给定的一个输入信息序列,在网格图中都存在一条特定的路径,如 $\boldsymbol{m}=[10111]$,则其输出码字序列为 $\boldsymbol{C}=[11\quad 10\quad 00\quad 01\quad 10]$,图 5-15 中带三角的线显示每位信息输入时的状态转移和码字输出过程。又如,当输入信息序列为 $\boldsymbol{m}=[01100]$时,输出码字则为 $\boldsymbol{C}=[00\quad 11\quad 01\quad 01\quad 11]$,由图 5-15 中带正方块的线显示每位信息输入时的状态转移和码字输出过程。因此,不同信息序列将由网格图中不相重合的路径区分。

卷积码的 4 种描述方法,生成矩阵、转移函数矩阵、状态流图和网格图从不同侧面描述卷积码,其中,生成矩阵和转移函数矩阵属同一大类,它们沿用了分组码的描述方法,建立了代数与编码器的关联,特点是物理意义清楚,代数量(多项式系数,矩阵元素)与编码电路连接线之间的对应关系十分明确,非常利于用 VHDL 等硬件描述语言来表达,以及用 FPGA、DSP 等来物理实现。状态流图和网格图属于图形表示法,状态流图可借助信号流图等图论工具或理论来分析卷积码的特性,网格图则特别适合用于计算机的穷举搜索,它使状态能在时域展开,所得的状态轨迹是研究差错事件、卷积码距离特性以及维特比最大似然序列译码最得力的工具。

2. 卷积码的译码

卷积码的性能取决于卷积码的距离特性和译码算法，其中，距离特性是卷积码自身本质的属性，它决定了该卷积码潜在的纠错能力，而译码方法只是研究如何将这种潜在的纠错能力转化为现实的纠错能力。

表述距离特性的最好方法是利用网格图。对于二元码，序列距离 $d(\boldsymbol{C}',\boldsymbol{C}'')$ 指的是汉明距离，即 $d(\boldsymbol{C}',\boldsymbol{C}'')=w(\boldsymbol{C}'\oplus\boldsymbol{C}'')$，其中，$w(\cdot)$ 为码的重量。当然，序列距离与序列的长度相关，同一序列对在不同长度时它们间的距离也不相同。为此，我们将长度 l 的任意两序列对间的最小距离定义为 l 阶列距离，又称为列距离函数，用 $d_c(l)$ 来表示，其定义为

$$d_c(l)=\min\{d([\boldsymbol{C}']_l,[\boldsymbol{C}'']_l):[\boldsymbol{m}']_0\neq[\boldsymbol{m}'']_0\} \tag{5.4.26}$$

其中，$[m']_0\neq[m'']_0$ 表示两个不同信息序列，且在网格图上从零时刻其轨迹开始分叉。由于早期卷积码译码方法与约束长度 $(L+1)$ 有关，于是把 $(L+1)$ 阶列距离称为最小距离：

$$d_{\min}=\min\{w([\boldsymbol{C}]_{L+1}):[\boldsymbol{m}]_0\neq 0\} \tag{5.4.27}$$

而把由零状态零时刻分叉的无限长的两个序列之间的最小距离定义为自由距离：

$$d_f=\min\{w([\boldsymbol{C}]_\infty):[\boldsymbol{m}]_0\neq 0\} \tag{5.4.28}$$

自由距离 d_f 在网格图上就是零时刻从零状态与全零路径分叉 $(\boldsymbol{C}\neq\boldsymbol{0})$，经若干分支后又回到全零路径（与全零序列距离不再继续增大）的所有路径中，重量最轻（与全零序列距离最近）的那条路径的重量。列距离、最小距离和自由距离三者之间的关系如下：

$$d_{\min}=d_c(l)\,|_{l=L+1}$$
$$d_f=\lim_{l\to\infty}d_c(l) \tag{5.4.29}$$

【例 5-15】 二进制(3,1,2)卷积码网格图如图 5-16 所示，试求该码 1～6 阶列距离、最小距离和自由距离。

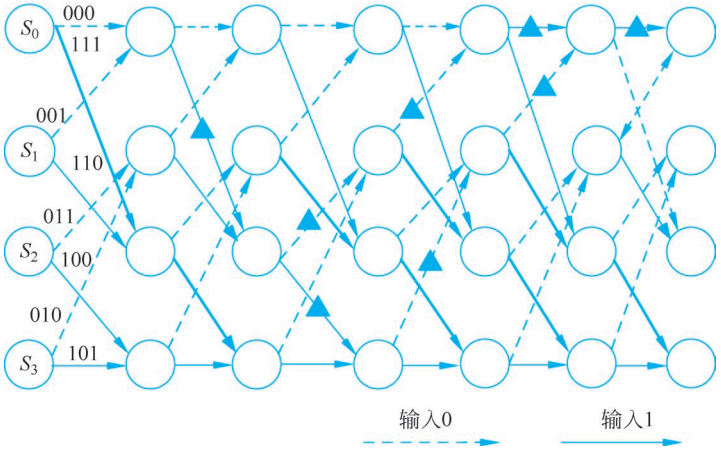


图 5-16 二进制(3,1,2)卷积码网格图

解：

	最轻码字序列	列距离函数 $d_c(l)$
$l=1$	$S_0\to S_2$	$d_c(1)=3$
$l=2$	$S_0\to S_2\to S_3$	$d_c(2)=4$

$l=3$　　$S_0\to S_2\to S_3\to S_1$　　$d_c(3)=5$（最小距离 $d_{\min}$）

$l=4$　　$S_0\to S_2\to S_3\to S_1\to S_0$　　$d_c(4)=6$

或　$S_0\to S_2\to S_1\to S_0\to S_0$

$l=5$　　$S_0\to S_2\to S_1\to S_0\to S_0\to S_0$　　$d_c(5)=6$

$l=6$　　$S_0\to S_2\to S_1\to S_0\to S_0\to S_0\to S_0$　　$d_c(6)=6$

由此可见

$$d_{\min}=d_c(l)\mid_{l=L+1}=d_c(3)=5$$

$$d_f=6$$

对于如上简单的卷积码，我们可以直接从网格图中找出 d_f。但是，随着限制长度的增加，状态数将以指数级增长，网格图将变得非常复杂和庞大，直接找出 d_f 往往变得不可能了。因此，各种计算方法应运而生。最常见的是采用解析法，借助信号流图来求自由距离。当状态数较大时，只能依靠计算机的搜索法来寻找 d_f，而当状态数非常大时，如超过 2^{20}，那么用现有计算机搜索也难以胜任。下面对如何用信号流图来计算自由距离进行说明。

信号流图由美国麻省理工学院的 Mason 于 20 世纪 50 年代首先提出。信号流图中的网络是由一些定向线段将一些节点连接起来组成的，包括节点和支路：节点即变量或信号，其值等于所有进入该节点的信号之和，有输入节点（也称源点）、输出节点（也称汇点）和混和节点；支路即连续两个节点的定向线段。信号在分支上只能沿着其上的箭头单向传递。原则上，可以利用信号流图计算任一以支路为基础的线性增益量。若干支路的串联构成一条路径，路径增益则是组成此路径的各支路增益的乘积（其指数是各支路增益指数之和）。两个节点之间可能有不止一条的路径，所以路径增益称为两节点之间的生成函数，用 $T(D)$ 表示。

卷积码的信号流图与卷积码的状态流图之间具有拓扑等效性。将信号流图法用于自由距离计算时，状态对应于节点，有趣的是不同转移间的距离（当与全零转移相比时就是重量）。因此，以各转移所对应的码字重量 W_j 为指数，定义支路增益为 D^{W_j} 和路径增益为 $D^{\sum_j W_j}$。由于自由距离是由零状态出发又回到零状态的最轻序列的重量，所以可以将零状态拆成两个节点，一个源点和一个汇点，这样，沿着任一条由源点到汇点的路径都有一个路径增益，其中指数最小者就是最轻路径。

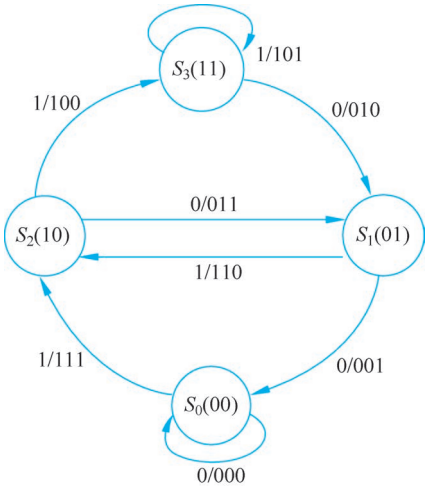
从生成函数 $T(D)$ 的角度看，如果将所有指数相同的路径增益合并，各路径增益的可表示为

$$T(D)=\sum_{d=0}^{+\infty}A_dD^d \tag{5.4.30}$$

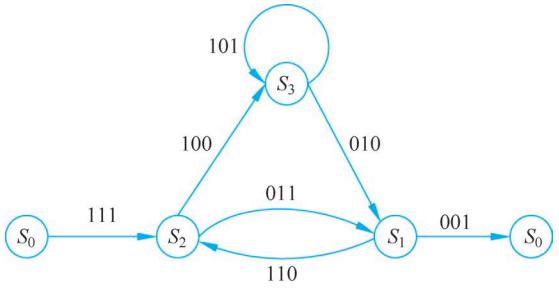
其中，系数 A_d 是路径增益指数为 d 的不同路径的条数。因此，$T(D)$ 中最低次非零项的指数就是最轻序列的重量，即自由距离 d_f。

【例 5-16】 二进制(3,1,2)卷积码结构如图 5-10 所示。试给出其状态流图，并用信号流图法求该码自由距离 d_f。

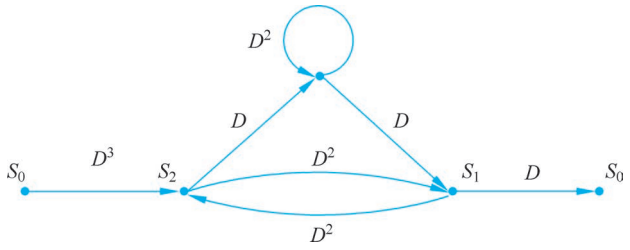
解：根据(3,1,2)卷积码结构，可以画出其状态流图如下：



将卷积码状态图的零状态 S_0 拆成一个源点和一个汇点，状态的自环是全零码，在信号流图中不画出，得到零状态拆分后的状态流图如下：



令各支路的增益为 D^j（这里 j 是与相应转移对应的码字重量，如码字 011 的重量为 2，对应的支路增益是 D^2），可得信号流图如下：



再根据信号流图的一些初等变换规则，如图 5-17 所示，将上图简化，最后得到生成函数，$T(D)=\dfrac{2D^6-D^8}{1-D^2-2D^4+D^6}$。运用多项式除法（长除法），可进一步得到 $T(D)=2D^6+D^8+5D^{10}+\cdots$，自由距离等于其中最轻者即次数最低的第 1 项，即 $d_f=6$。

译码算法是将潜在的纠错能力转化为实际纠错能力的手段，其中较有名的卷积码译码算法称为维特比（Veterbi）译码算法。它采用概率译码的思想，将已接收序列与所有可能的发送序列做比较，计算出各自码距，并选择码间距离最小的序列作为发送码序列的估计。如果接收到 m 组、每个信息组包含 k 个比特，接收到码字序列将与 2^{mk} 条路径结果进行比较，其中码距最小的那一条路径被选择为最有可能的路径，对应的输出码字序列即为发送码序

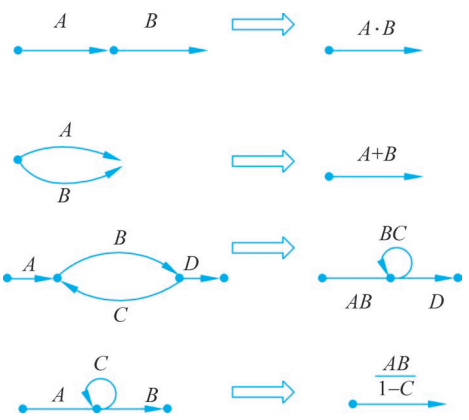


图 5-17 信号流图的一些初等变换规则

列的估计。然而，当 m 较大时，概率译码将难以实现。随后，维特比对概率译码算法进行了简化，它并不是在网格图上一次比较所有可能的 2^{mk} 条路径(序列)，而是接收一段，计算和比较一段，选择出最大似然可能的一段序列作为估计，进而达到整个估计序列为最大似然序列的目的。

由于编码序列经过有噪信道传输之后会受到干扰，使得其值可能不再是 0 或 1 这样的整数，因此在进入译码器前，需要对接收序列中每比特进行处理，卷积码译码可分为硬判决译码(Hard Decision Decoding，HDD)算法和软判决译码(Soft Decision Decoding，SDD)算法两种，前者先将所采样的信号做判别再进行译码，搜寻与接收序列的汉明距离最小的序列作为对编码序列的估计值；后者则直接用被采样信号进行译码运算。下面以硬判决译码算法为例，并结合图格图，详细描述维特比译码过程。

由于在二进制对称信道中最大似然概率译码准则等效于最小汉明距离的译码准则，因此具有最小码距 $d(\boldsymbol{r},\boldsymbol{c})$ 累积值的路径就是对数似然概率 $\log p(\boldsymbol{r}|\boldsymbol{c})$ 的最大路径，其中，$\boldsymbol{r}$ 为信道接收序列，$\boldsymbol{c}$ 为发送到信道的码序列，该路径被称为幸存路径。定义分支量度 BM(Branch Metric)，BM$=d(r_j,c_j)$ 是第 j 时刻接收码 $\boldsymbol{r}_j$ 相对于预测码的相似度。在软判决情况下，BM 一般指欧几里得距离，而在硬判决情况下，BM 为汉明距离。路径度量 PM(Path Metric)与网格中的状态相关联，在硬判决中，对应于网格中从初始状态到当前状态的最可能路径与接收码序列间的汉明距离。维特比算法的核心是接收机可以使用分支度量和先前计算的状态路径度量递推地计算当前状态的路径度量。由于译码过程也建立在网格图中，并且从全零状态开始到全零状态结束。所以，在每个符合输入的分支中，都可以计算出分支度量值。

维特比译码算法步骤可概括如下。

(1) 在 $j=L+1$($L+1$ 为卷积码约束长度)个时刻前，计算每一个状态单个路径分支度量，并存储每一状态下的幸存路径及其度量值。

(2) j 增加 1，即从 $j=L+1$ 开始到 $N-1$ 时刻结束(N 为输入信息序列的长度)，对进入每一个状态的部分路径进行计算，这样的路径应有 2^k 条(k 为卷积码信息位长度)，从中挑选具有最大值的部分路径为幸存路径，删除进入该状态的其他路径，然后幸存路径向前延长一个时间段。若进入某个状态的部分路径中，有两条部分路径值相等，则可以任选其一作为幸存路径。

(3) 重复步骤(2)的计算、比较和判决过程。若接收序列长为$(N+L+1)k$，其中，后 $L+1$ 是人为加入的全 0 值，则译码将至$(N+L+1)$时刻为止。

下面以(2,1,2)卷积码为例说明维特比译码过程，输入数据 $\boldsymbol{m}=[1\ 1\ 0\ 1\ 1]$，输入信道的编码序列为 $\boldsymbol{c}=[11\ 01\ 01\ 00\ 01]$，信道接收端获取的接收码字为 $\boldsymbol{r}=[11\ 01\ 01\ 10\ 01]$。译码参数值分别是 $N=5$，$L=2$，$k=1$。由于系统是有记忆的，j 将从 0 开始，到 $j=N+L+1=5+2+1=8$ 为止。

若假设编码器总是起始于状态 $S_0(00)$，则前 $j=L=2$ 个时刻对应于编码器从状态 S_0 出发，而最后的两个时间段则相当于编码器返回到 S_0 状态。因此，在前两个与后两个时间段内不可能达到所有可能的状态，但在网格图中，其中心部分所有状态都是可以达到的。每一个状态都有 $2^k=2^1=2$ 个分支的离去和进入。在时间段 j，离开每一个状态的虚线分支表示输入是 0，而实线分支表示输入是 1。维特比算法的基本思想是依次在不同时刻 $j=L+1$，$L+2,\cdots,L+N+1$，对图中相应列的每个点（对应于编码器中该时刻的一个状态），按最大似然准则（或最小汉明距准则）比较所有以它为终点的路径，保留一条具有最大似然值（或最小汉明距值）的路径为幸存路径，而将其他路径弃之不用。因此，下一时刻只需对幸存路径延伸出来的路径继续比较，即接收一段，比较一段，保留幸存路径，按此过程一直到最后，即 $N+L+1$ 时刻，所保留下的路径就是所要求的最大似然译码结果。下面以二进制(2,1,2)卷积码为例，讲述上述过程。

(1) 在 $j=0,1,2$ 这 3 个时刻，执行步骤(1)，计算出每个路径的分支度量值，即汉明距离，如图 5-18 所示，并在图中以数字标记该分支路径的最小汉明距离，如接收序列 $\boldsymbol{r}$ 中的第 1 个序列是 11，与第一时刻网络图中的两条路径 00、11 的汉明距分别为 2 和 0，因而这些数字被标注在对应路径的下方或附近。

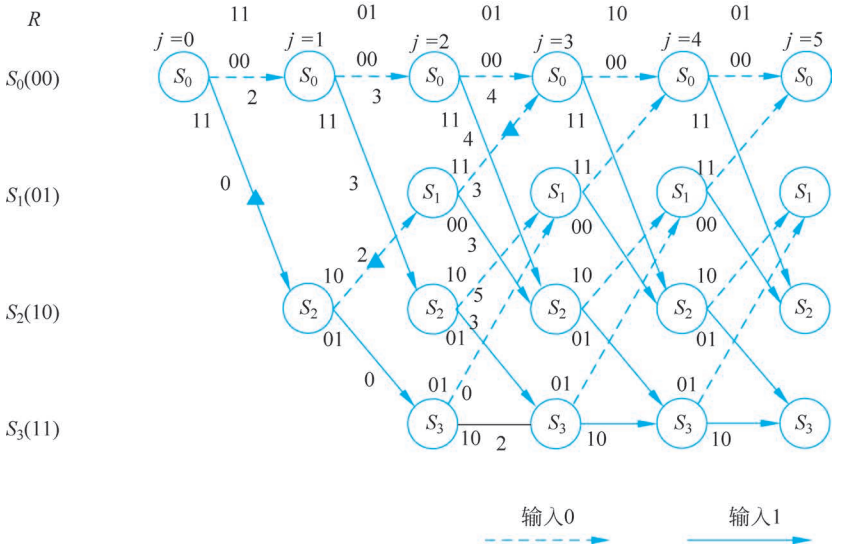


图 5-18 解码步骤(1)

(2) 接下来选择下一时刻的幸存路径。由于在 $j=3$ 时刻进入状态 S_0 的两条路径的 PM 值分别为 3 和 4，所以选择 PM 为 3 的路径为幸存路径，在图中用带三角线段表示，并去除到达 S_0 状态的其他路径。

同理，分别获取进入 S_1、S_2、S_3 状态的幸存路径，如图 5-19 所示。至此，$j=2+1=3$ 时刻的幸存路径选择完成，并将接下来的路径的分支度量值都标注在图中，如图 5-20 所示。

在图 5-20 中，每个节点（状态）进行分支度量值比较，分支度量值较小的被标注在状态节点的上方。按此过程不断进行，获得最后一个接收符号所对应一条分支度量值最小的路径，如图 5-21 所示，即为接收序列的最佳路径。

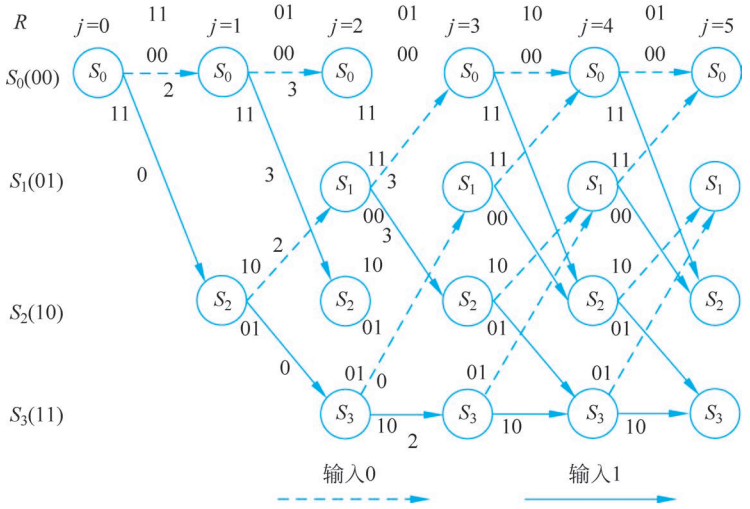


图 5-19　进入 S_0 $S_1S_2S_3$ 的幸存路径选择

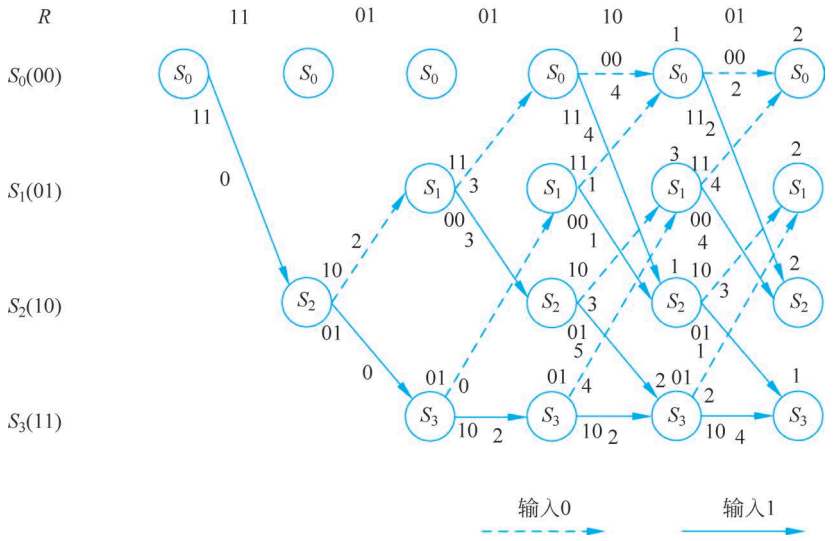


图 5-20　剩余译码路径的分支度量值计算

根据最佳路径，获得对应的输入序列为 11011，且在译码过程中纠正一个比特错误。

维特比译码是按最大似然准则进行的译码，是卷积码的一种最佳译码方法，它的性能由信道质量决定，然而这种译码方法存在着两个主要缺点：一是要等全部接收的数据进入译码器以后才能最后译出结果，所以译码时延长；二是需要存储 2^{kL} 条幸存路径的全部历史数据，所以存储量大。

【例 5-17】　对于图 5-10 所示(3,1,2)卷积码。设发送的码字序列是 $\boldsymbol{c}$=[000,111,011,001,000,000,…]，传输时发生两位差错，接收的码字序列为 $\boldsymbol{r}$=[110,111,011,001,000,000,…]，试用维特比算法进行其译码。

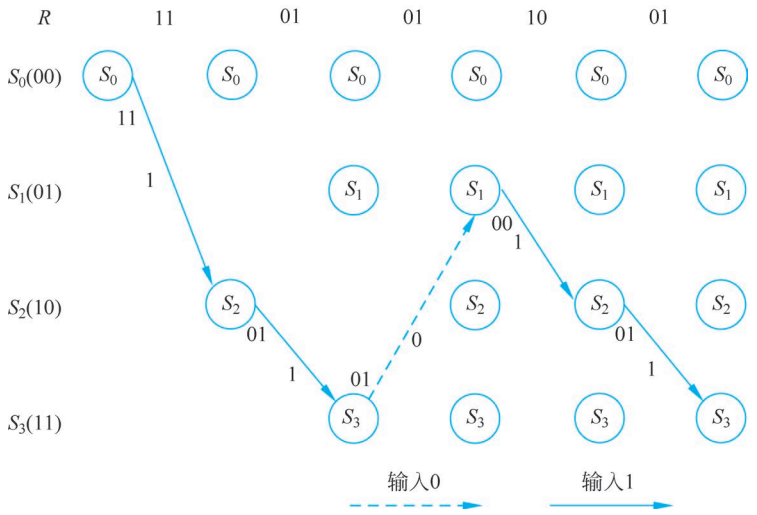
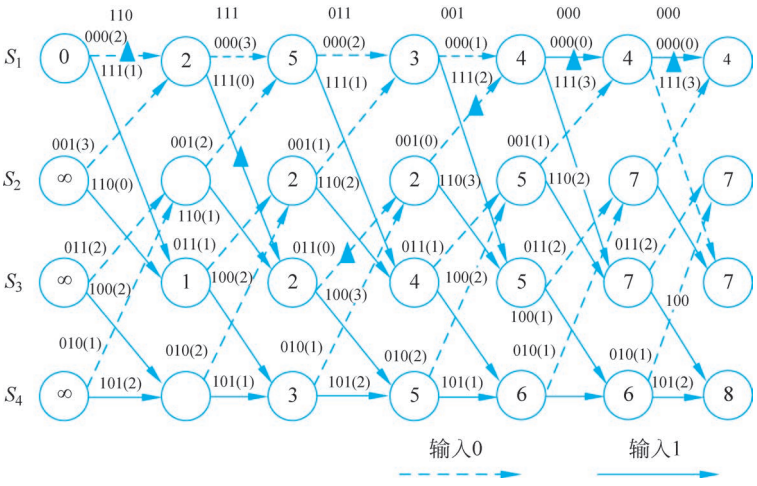
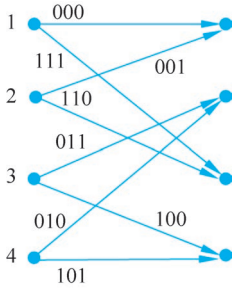


图 5-21 (2,1,2)卷积码对应于接收序列的译码最佳路径

解：利用维特比译码，可以获得其译码过程如下，其中带三角路径是接收序列的最佳路径，具体步骤介绍如下：



(1) 为了便于编程实现，现给出计算过程。首先是网格图，其基本结构如下：



其中，假设 4 个状态用 1、2、3、4 表示。定义 $p(m,n)$ 表示到达 m 状态的第 n 个前状态

(predecessors)是什么,而 $c(m,n)$ 表示第 n 个前状态转移到达 m 状态时对应的码字。例如,$p(4,1)=3$,$p(4,2)=4$ 表示到达 4 状态的第 1、第 2 个前状态分别是状态 3 和 4,对应的码字分别是 $c(4,1)=100$ 和 $c(4,2)=101$,于是,以上网格图可用数组描述如下:

$$\begin{aligned} &p(1,1)=1,c(1,1)=000, \quad p(1,2)=2,c(1,2)=001\\ &p(2,1)=3,c(2,1)=011, \quad p(2,2)=4,c(2,2)=010\\ &p(3,1)=1,c(3,1)=111, \quad p(3,2)=2,c(3,2)=110\\ &p(4,1)=3,c(4,1)=100, \quad p(4,2)=4,c(4,2)=101 \end{aligned}$$

(2) 计算第 j 时刻接收码 r_j 相对于各码字的分支量度 BM。在二进制硬判决情况下,BM 即为汉明距离,因此其计算表达式为

$$BM^j(m,n)=W(c(m,n)\oplus r_j) \tag{5.4.31}$$

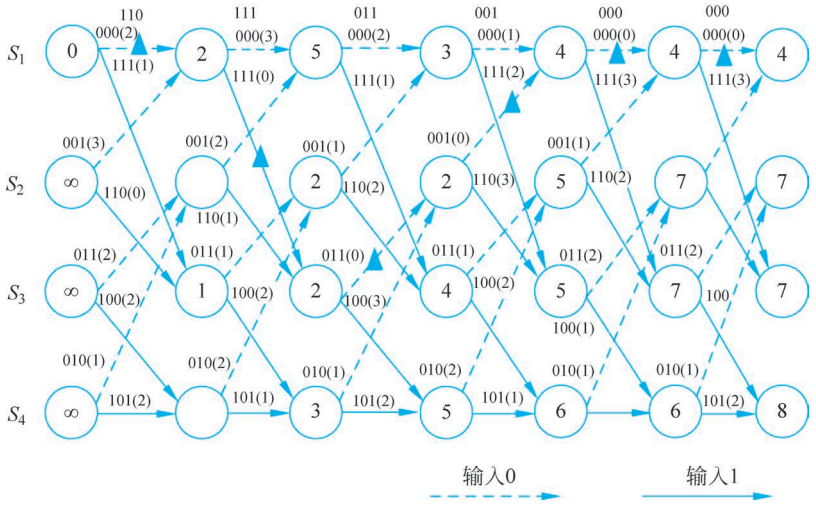
其中,$W(\cdot)$是码字重量,$BM^j(m,n)$表示第 j 时刻接收码 r_j 与到达第 m 状态的第 n 个转移所对应的码字的距离。本题中 $r_1=110$,$r_2=111$,$r_3=011$,$r_4=001$,$r_5=000\cdots$,因此,时刻 3 的 BM 值分别是 $BM^3(1,1)=W(c(1,1)\oplus r_3)=W(000\oplus 011)=2$,同理,可以计算出:$BM^3(1,2)=1$,$BM^3(2,1)=0$,$BM^3(2,2)=1$,$BM^3(3,1)=1$,$BM^3(3,2)=2$,$BM^3(4,1)=3$,$BM^3(4,2)=2$。

(3) 计算第 j 时刻到达状态 m 的路径量度 $PM^j(m)$,它是将上一时刻的路径量度 PM^{j-1} 与本时刻分支量度 BM 累加后选择其中汉明距离最小的一个,即

$$PM^j=\min_m\{PM^{j-1}[p(m,n)]+BM^j(m,n)\} \tag{5.4.32}$$

初始时,除全零状态的 $PM^0(1)=0$ 外,其余状态的 $PM^0(i)$,$i\neq 0$ 均置为∞。因此,时刻 3 中到达状态 1 的路径可以来自状态 1 和状态 2 两处,这两处前时刻的路径量度分别是 $PM^2(1)=5$ 和 $PM^2(2)=2$,本时刻的分支量度分别是 $BM^3(1,1)=2$ 和 $BM^3(1,2)=1$,因此,时刻 3 状态 1 的路径量度 $PM^3(1)=\min\{PM^2[p(1,1)]+BM^3(1,1)$,$PM^2[p(1,2)]+BM^3(1,2)\}=\min\{5+2,2+1\}=3$。以上计算路径量度的过程实际上就是挑选到达状态 1 的最大似然路径的过程。看到有两条路径可达,一条与接收码汉明距离 5+2,另一条的汉明距离 2+1,距离越小则似然度越大,所以取 $PM^3(1)=3$ 隐含了选择路径 $S_1\to S_3\to S_2\to S_1$ 为到达状态 1 的幸存路径,同理,可获得到达状态 2、3、4 的幸存路径。

(4) 译码输出以及更新第 j 时刻、状态 m 对应的留存路径(Survivor)$S^j(m)$。留存路径是与最大似然路径对应的码字序列,每状态一个,长度为 D。留存路径每时刻按以下步骤更新一次:①设到达状态 m 的最大似然路径的前状态是 n,则令 n 状态前时刻的留存路径作为本时刻本状态 m 的留存路径,即 $S^j(m)=S^{j-1}(n)$。②选择具有最小(最似然)PM 那个状态的留存路径最左边的码字作为译码输出。③将各状态留存路径最左边的码字从各移存器移出,再将到达各状态的最大似然路径在时刻 j 所对应的码字从右面移入留存路径 $S^j(m)$。时刻 $j=3$ 到达状态 2 的最大似然路径来自状态 3,而前时刻状态 3 的留存路径是 $S^2(3)=000,000,000,111$(长度 $D=4$)。比较各状态的 $PM^3(i)$,发现状态 2 是最大似然路径,其前时刻在状态 3,于是取 $S^2(3)$最左边的码字 000 作为译码输出。接着,将 $S^2(2)$最左边(最旧)的码字 000 移出,将时刻 3 到达状态 2 的转移所对应的码字 011 从右边移入,得更新后状态 2 的留存路径 $S^3(2)=000,000,111,011$。同理可得 $S^3(1)$、$S^3(3)$、$S^3(4)$。重复步骤②~④,将维特比算法持续下去,获得译码结果如下图中带三角的线段所示。



5.5 信道编码的 MATLAB 计算实现


视频讲解

与信源编码理论一样，信道编码理论仅指出了在传输率小于信道容量的条件下，可以实现可靠性通信的信道编码是存在的，并没有给出具体构造好码的方法。5.4 节从最基本的线性分组码开始，介绍了信道编码的编译码过程。现进一步给出几种常用信道编码方法，以及它们的 MATLAB 实现。

5.5.1 RS 码

RS 码是由 Reed 和 Solomon 提出的一类具有很强纠错能力的多进制循环纠错码，它是循环 BCH 码的一个重要子类。RS 码的译码是按符号进行的，因此 RS 码特别适用于纠正突发错误。例如，在硬盘驱动器、CD 和 DVD 等数据存储系统及消费电子设备中常采用卷积码作为内码、RS 码作为外码的串行级联码；除此之外，RS 码也在众多的数字通信系统（如卫星通信、深空探测、美国数字电视国家标准、数字音频广播和数字视频广播等）中得到应用。

对于定义在 GF(q^m)域上的 RS 码，其分组长度为 $n=q^m-1$，获得码距为 δ 的 RS 码的基本步骤如下。

（1）选取一个次数为 m 的既约生成多项式并构造 GF(q^m)域。

（2）依据 α^i，$i=1,2,\cdots,\delta-1$ 构造生成多项式 $g(x)=\prod_{i=1}^{\delta-1}(x-\alpha^i)$。

（3）依照循环码的编码方法和编码电路进行编码（所有加法运算和乘法运算都在 GF(q^m)上进行）。

现以定义在本原多项式为 $p(x)=x^3+x+1$ 的 GF(2^3)上的 RS 码为例说明构造过程，其中，RS 码的码长 $n=2^3-1=7$，码距 $\delta=5$。

首先，假设 α 为 GF(2^3)的本原元，由于要求设计距离为 $\delta=5$，所以该 RS 码以 $\alpha,\alpha^2,\alpha^3,\alpha^4$ 为根，因此生成多项式为 $g(x)=(x-\alpha)(x-\alpha^2)(x-\alpha^3)(x-\alpha^4)=x^4+\alpha^3x^3+x^2+\alpha x+\alpha^3$。

由于 α 为 $p(x)=x^3+x+1=0$ 的根，即 $\alpha^3+\alpha+1=0$ 或 $\alpha^3=\alpha+1$，那么 GF(2^3)中的元素可按表 5-3 计算。

表 5-3 各元素的计算结果

0	$\mathrm{mod}(\alpha^3+\alpha+1)=0$
α^0	$\mathrm{mod}(\alpha^3+\alpha+1)=\alpha^0=1$
α^1	$\mathrm{mod}(\alpha^3+\alpha+1)=\alpha$
α^2	$\mathrm{mod}(\alpha^3+\alpha+1)=\alpha^2$
α^3	$\mathrm{mod}(\alpha^3+\alpha+1)=\alpha+1$
α^4	$\mathrm{mod}(\alpha^3+\alpha+1)=\alpha^2+\alpha$
α^5	$\mathrm{mod}(\alpha^3+\alpha+1)=\alpha^2+\alpha+1$
α^6	$\mathrm{mod}(\alpha^3+\alpha+1)=\alpha^2+1$
α^7	$\mathrm{mod}(\alpha^3+\alpha+1)=\alpha^2$
α^8	$\mathrm{mod}(\alpha^3+\alpha+1)=\alpha$
…	…

值得注意的是，对于 GF(2^3)域运算，“+”运算与“−”运算等价。由此生成 GF(2^3)上的(7,3)RS 码，其系统码生成多项式矩阵 $\boldsymbol{G}$ 为

$$\boldsymbol{G}=\begin{bmatrix}1&0&0&\alpha^4&1&\alpha^4&\alpha^5\\0&1&0&\alpha^2&1&\alpha^6&\alpha^6\\0&0&1&\alpha^3&1&\alpha&\alpha^3\end{bmatrix}$$

由于该 RS 码是定义在 GF(2^3)上的，因此每个编码信息符号由 3 个信息比特组成，假设输入信息符号组为 $\boldsymbol{m}=[2,4,6]$，转换为 GF(2^3)上的元素 $\boldsymbol{m}=[\alpha,\alpha^2,\alpha^4]$，编码码字为

$$\boldsymbol{C}=\boldsymbol{mG}=\begin{bmatrix}\alpha&\alpha^2&\alpha^4\end{bmatrix}\begin{bmatrix}1&0&0&\alpha^4&1&\alpha^4&\alpha^5\\0&1&0&\alpha^2&1&\alpha^6&\alpha^6\\0&0&1&\alpha^3&1&\alpha&\alpha^3\end{bmatrix}$$

$$=\begin{bmatrix}\alpha&\alpha^2&\alpha^4&0&0&\alpha&\alpha^4\end{bmatrix}$$

即编码输出为 $\boldsymbol{C}=[2\quad 4\quad 6\quad 0\quad 0\quad 2\quad 6]$，转换为二进制为 $\boldsymbol{C}=[010\quad 100\quad 110\quad 000\quad 000\quad 010\quad 110]$。

MATLAB 代码如下：

```
clear all
close all
M = 8;                                          % 调制阶数
bps = log2(M);                                  % 每个符号所含比特数
N = 7;                                          % RS 码长
K = 3;                                          % RS 信息位长度
mod = comm.PSKModulator('ModulationOrder',M,'BitInput',false);
 % 创建调制器(8PSK 调制方式)
demod = comm.PSKDemodulator('ModulationOrder',M,'BitOutput',false);
 % 创建解调器
chan = comm.AWGNChannel('BitsPerSymbol',bps);          % 创建 AWGN 信道
err = comm.ErrorRate;
enc = comm.RSEncoder('BitInput',false,'CodewordLength',N,'MessageLength',K);
```

```
% 创建编码器
dec = comm.RSDecoder('BitInput',false,'CodewordLength',N,'MessageLength',K);
% 创建译码器
ebnoVec = (3:0.5:8)';                                    % 信噪比
errorStats = zeros(length(ebnoVec),3);
for n = 1:length(ebnoVec)
    chan.EbNo = ebnoVec(n);
    reset(err)
    while errorStats(n,2) < 100 && errorStats(n,3) < 1e7
        data = randi([0 2],1500,1);                      % 生成随机数据
        encData = step(enc,data);                        % RS 编码
        modData = step(mod,encData);                     % 调制
        rxSig = step(chan,modData);                      % 通过 AWGN 信道
        rxData = step(demod,rxSig);                      % 解调
        decData = step(dec,rxData);                      % RS 译码
        errorStats(n,:) = step(err,data,decData);        % 错误统计
    end
end
berCurveFit = berfit(ebnoVec,errorStats(:,1));           % 曲线拟合
berNoCoding = berawgn(ebnoVec,'psk',8,'nondiff');        % 未编码
semilogy(ebnoVec,errorStats(:,1),'b*',ebnoVec,berCurveFit,'c-',ebnoVec,berNoCoding,'r')
ylabel('错误率')
xlabel('Eb/N0 (dB)')
legend('RS 编码的仿真数据','拟合曲线','未编码')
grid
```

运行结果如图 5-22 所示。

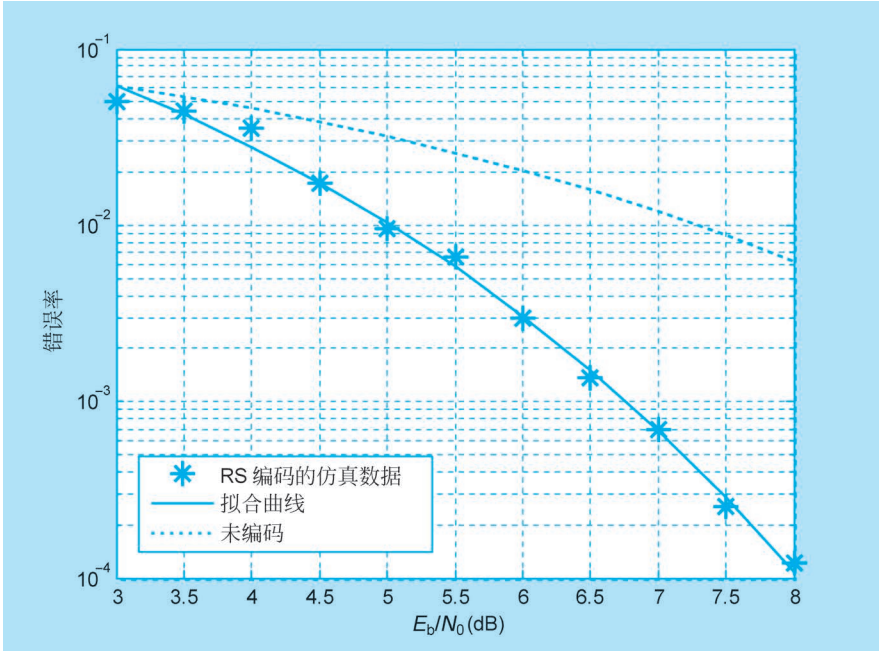


图 5-22 RS 编码结果

5.5.2 Turbo 码

1993 年,法国科学家 C. Berrou 和 A. Glavieux 等于瑞士日内瓦召开的 ICC 会议上提出了 Turbo 码的概念。由于 Turbo 码很好地应用了香农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香农理论极限的译码性能。仿真结果显示:在高斯白噪声信道下,码率为 1/2 的 Turbo 码在达到误比特率(BER)小于或等于 10^{-5} 时,E_b/N_0 只有约 0.7dB(这种情况下达到信道容量的理想 E_b/N_0 值为 0dB),远超其他的编码方式。Turbo 码不仅在信道信噪比很低的环境下性能优越,而且还具有很强的抗衰落能力,因此它在信道条件差的移动通信系统中得到了广泛应用,如在第三代移动通信系统(IMT-2000)中已经将 Turbo 码作为其传输高速数据的信道编码标准。

Turbo 码是一种并行级联卷积码(Parallel Concatenated Convolutional Codes)。编码器通过交织器把两个分量编码器进行并行级联,两个分量编码器分别输出相应的校验位比特;译码器在两个分量译码器之间进行迭代译码,分量译码器之间传递可去掉正反馈的外信息,这样整个译码过程类似涡轮(Turbo)式工作。因此,这个编码方法又被形象地称为 Turbo 码。

图 5-23 所示为 Turbo 码编码器的结构示意图。信息序列 $U=\{u_1,u_2,\cdots,u_N\}$ 经过交织器形成一个新序列 $U'=\{u'_1,u'_2,\cdots,u'_N\}$(长度与内容没有改变,但比特位经过重新排列),$U$ 和 U' 分别传送到两个编码器(编码器 1 与编码器 2),一般情况下,这两个编码器结构相同,生成序列 X^{p1} 和 X^{p2}。为了提高码率,序列 X^{p1} 和 X^{p2} 需要经过删除器,采用删除(Puncturing)技术从这两个校验序列中周期地删除一些校验位,形成校验序列 X^p,X^p 与未编码序列 X' 经过复用调制后,生成了 Turbo 码序列 X。

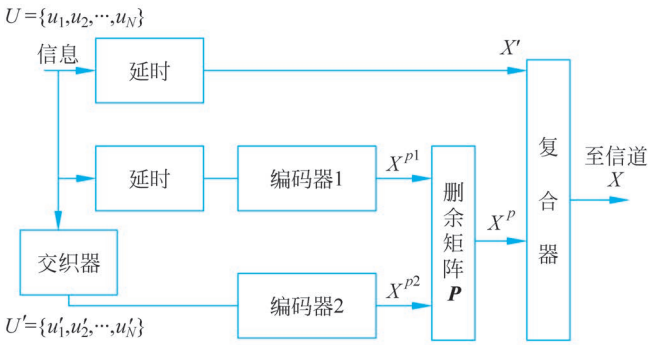


图 5-23 Turbo 码编码器的结构示意图

Turbo 码通过在编码器中引入交织器使得码字具有近似随机的特点,通过分量码的并行级联实现了长码构造,Turbo 码充分考虑了香农信道编码定理证明时假设的条件,从而获得了接近香农理论极限的性能。

Turbo 码的译码算法采用了最大后验概率算法(MAP)。在译码的结构上又进行了改进,再次引入反馈的概念,取得了性能和复杂度之间的折中。同时,Turbo 码采用的是迭代译码,这与经典的代数译码是完全不同的。Turbo 码的译码算法是在 BCJR 算法的基础上改进的,称为 MAP 算法,后来又形成 Log-MAP 算法、Max-Log-MAP 以及软输出维特比译

码(SOVA)算法。

Turbo 码的译码器结构如图 5-24 所示,由两个成员码对应的译码单元和交织器与解交织器组成,一个译码单元的软输出信息将作为另一个译码单元的输入,且将此过程不断迭代以提高译码性能。其中,X_k 是接收到的信息比特序列,Y_{1k} 是编码器 1 的校验比特,Y_{2k} 是编码器 2 的校验比特。

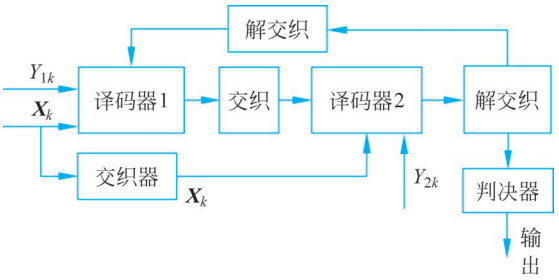


图 5-24 Turbo 码译码器的结构示意图

Turbo 码译码器为串行结构,两个编码器所产生的校验信息通过串并转换被分开,分别送到对应的译码器输入端,即 Y_{1k} 和 Y_{2k}。首先,第 1 级译码器根据接收到的码序列 X_k 和第 1 级编码器的校验位信息 Y_{1k} 得到输出信息(此时译码器 2 的外信息为 0),其输出信息包含两部分:一部分是由译码器根据码字相关性提取出来的外信息;另一部分是来自于接收码序列对应的输出,该输出在传递给下一级译码器之前被减去。外信息在交织之后被送到译码器 2 作为先验信息,译码器 2 根据校验位信息 Y_{2k}、接收码元序列 X_k 和外信息这三者共同做出估计,得到输出信息,同时得到译码器 2 的外信息,该信息在解交织之后,反馈给第 1 级译码器作为先验信息;结合该外信息,译码器 1 重新做出译码估计,得到改善的外信息。而此时的外信息又可以传递到译码器 2,如此往复,形成迭代过程。最终,当达到预先设定的迭代次数或满足迭代结束条件时,译码结束,取输出的符号作为最终硬判决结果。

译码时首先对接收信息进行处理,两个译码器之间外部信息的传递就形成了一个循环迭代的结构。由于外信息的使用,一定信噪比下的误比特率将随着循环次数的增加而降低。同时,外信息与接收码元序列间的相关性也随着译码次数的增加而逐渐增加,外信息所提供的纠错能力也随之减弱,在一定的循环次数之后,译码性能将不再提高。以下是 Turbo 码的 MATLAB 程序实现。

MATLAB 代码如下:

```
clear all
close all
M = 16;                                                    % 调制阶数
EbN0 = -5:-1;                                              % 信噪比范围
frmLen = 500;                                              % 帧长
ber = zeros(size(EbN0));
enc = comm.TurboEncoder('InterleaverIndicesSource','Input port');    % 创建编码器
dec = comm.TurboDecoder('InterleaverIndicesSource','Input port', ...
    'NumIterations',4);                                    % 创建译码器
mod = comm.RectangularQAMModulator('ModulationOrder',M, ...
    'BitInput',true,'NormalizationMethod','Average power');          % 创建 QAM 调码器
demod = comm.RectangularQAMDemodulator('ModulationOrder',M, ...
    'BitOutput',true,'NormalizationMethod','Average power', ...
    'DecisionMethod','Log-likelihood ratio', ...
    'VarianceSource','Input port');                        % 创建 QAM 译码器
chan = comm.AWGNChannel('EbNo',EbNo,'BitsPerSymbol',log2(M));
% 创建 AWGN 信道
```

```
errorRate = comm.ErrorRate;
for k = 1:length(EbNo)
    errorStats = zeros(1,3);
    noiseVar = 10^(-EbNo(k)/10) * (1/log2(M));
    chan.EbN0 = EbN0(k);
    while errorStats(2) < 100 && errorStats(3) < 1e7
        data = randi([0 1],frmLen,1);                          % 创建二进制随机数据
        intrlvrInd = randperm(frmLen);                         % 交织
        encodedData = step(enc,data,intrlvrInd);               % Turbo 编码
        modSignal = step(mod,encodedData);                     % 调制
        receivedSignal = step(chan,modSignal);                 % AWGN 信道
        demodSignal = step(demod,receivedSignal,noiseVar);     % 解调
        receivedBits = step(dec,-demodSignal,intrlvrInd);      % 译码
        errorStats = step(errorRate,data,receivedBits);        % 统计差错
    end
    ber(k) = errorStats(1);
    reset(errorRate)
end
semilogy(EbN0,ber,'r-o')
grid
xlabel('Eb/N0 (dB)')
ylabel('误比特率')
uncodedBER = berawgn(EbNo,'qam',M);                            % 未编码的 BER
hold on
semilogy(EbN0,uncodedBER,'b-')
legend('Turbo 编码','未编码')
```

运行结果如图 5-25 所示。

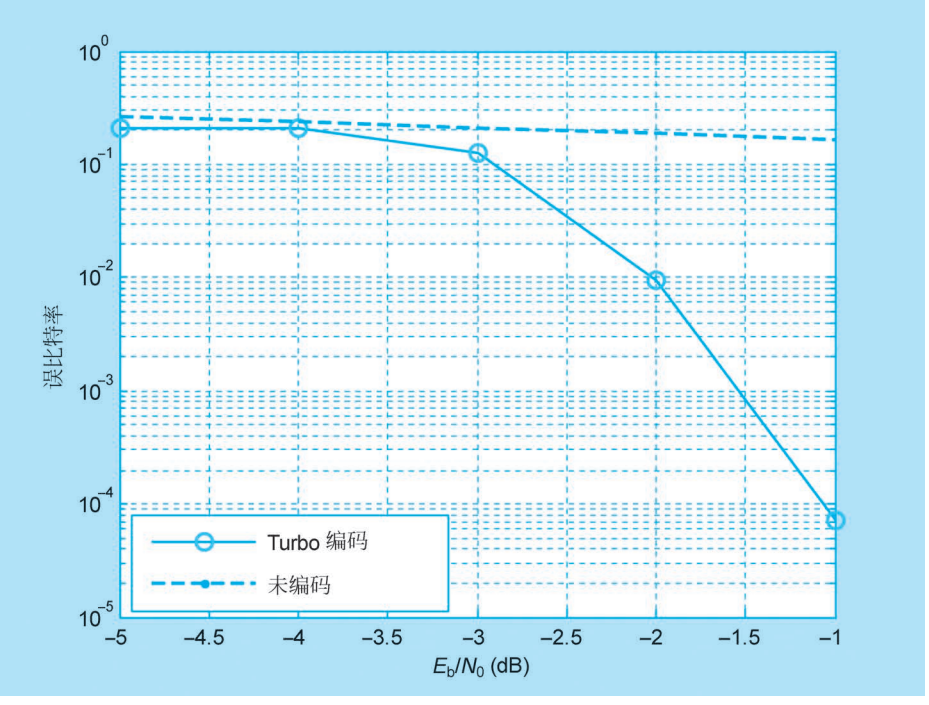


图 5-25 Turbo 编码结果

5.5.3 LDPC 码

低密度奇偶校验(Low-density Parity-check Code,LDPC)码是一种性能接近香农极限的好码,也叫 Gallager 码,现被 3GPP RAN1 会议确定为 5G 中移动宽带业务的长码块编码方案。LDPC 码由 Gallager 于 1962 年首先提出,在很长一段时间里并没受到人们的重视,直到 1996 年,Mackay 和 Neal 等再次提出,LDPC 码的研究才进入一个崭新阶段。随着编码矩阵构造、解码算法优化等关键技术的突破,LDPC 码被各种通信所采纳,如卫星数字广播系统、地面数字视频广播系统和无线接入网络中 IEEE 802.11n 等。

LDPC 码性能优于 Turbo 码,具有较大的灵活性和较低的差错平底特性(Error Floors)。它描述简单,对严格的理论分析具有可验证性,译码复杂度低于 Turbo 码,可实现完全的并行操作且硬件复杂度低,具有高速译码潜力。目前,LDPC 码已在深空通信、光纤通信、卫星数字视频和声频广播、磁/光等存储、移动和固定无线通信、电缆调制/解调器和数字用户线(DSL)中得到广泛应用。

LDPC 码是一类特殊的线性分组码,它的校验矩阵是稀疏矩阵,即校验矩阵中只有很少一部分的 1 元素,绝大多数都是 0 元素。LDPC 码分为规则 LDPC 码和非规则 LDPC 码。对于规则 LDPC 码,Gallager 给出如下的定义:(n,j,k)LDPC 码是长为 n 的码字,在它的奇偶校验矩阵中,每一行和每一列中 1 的个数是固定的,其中每一列有 $j(j\geqslant 3)$个 1,每一行有 $k(k>j)$个 1;列之间 1 的重叠数目小于或等于 1。式(5.5.1)是按定义构造的(12,3,4)LDPC 码的校验矩阵 $\boldsymbol{H}$。在校验矩阵中,每行和每列的 1 元素的个数都是固定的。

$$\boldsymbol{H}=\begin{bmatrix}0&0&1&0&0&1&1&1&0&0&0&0\\1&1&0&0&1&0&0&0&0&0&0&1\\0&0&0&1&0&0&0&0&1&1&1&0\\0&1&0&0&0&1&1&0&0&1&0&0\\1&0&1&0&0&0&0&1&0&0&1&0\\0&0&0&1&1&0&0&0&1&0&0&1\\1&0&0&1&1&0&1&0&0&0&0&0\\0&0&0&0&0&1&0&1&0&0&1&1\\0&1&1&0&0&0&0&0&1&1&0&0\end{bmatrix} \tag{5.5.1}$$

除此之外,LDPC 码还可以用 Tanner 图表示。上式(12,3,4)对应的 Tanner 图如图 5-26 所示。

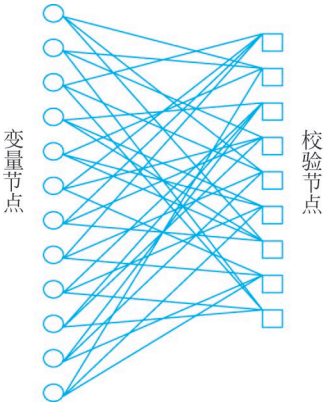


图 5-26 (12,3,4)规则 LDPC 码 Tanner 图

在 Tanner 图中,左边的节点称为变量节点,右边的节点称为校验节点。每个变量节点对应 LDPC 码的一个码元,其个数等于码长 n,即等于校验矩阵的列数。校验节点与校验方程对应,其个数等于校验方程的个数,即校验矩阵的行数。所以,Tanner 图中的线与校验矩阵中的元素 1 是对应的。在 Tanner 图中与某节点相连的边的个数称为该节点的度。规则 LDPC 码中相同类型的节点度数应该是相同的,即所有变量节点的度也相同,所有校验节点的度也相同;而非规则 LDPC 码的变量节点/校验节点的度是不同的。

LDPC码的译码算法主要是基于Tanner图结构的消息传递(Message Passing,MP)算法,该算法的性能随着量化阶数的增加而提高,同时复杂度也随之增加。其中性能最好的是置信传播(Belief Propagation,BP)算法,当编码Tanner图中没有环时,它的性能可等效于最大似然算法。

BP算法可以由初始化过程和迭代过程两个步骤完成,具体如下。

(1) 初始化:在进行迭代过程之前首先要给参数赋初值。令 $p_l^0=p(x_l=0)$(比特 x_l 取0的先验概率),$p_l^1=p(x_l=1)=1-p_l^0$。对于每个满足 $H_{ml}=1$ 的(l,m),变量 q_{ml}^0 和 q_{ml}^1 分别被初始化为 p_l^0 和 p_l^1。

(2) 迭代。

① 对每一个校验约束 m 和对应的每一个 $l\in L(m)$计算两个概率,其中一个是 r_{ml}^0,当 $x_l=0$,且其他比特$\{x_{l'}:l'\neq l\}$对应于相互独立的概率分布时,$\{q_{ml'}^0,q_{ml'}^1\}$校验约束 m 得到满足的概率,即

$$r_{ml}^0=\sum_{\{x_{l'}:\,l'\in L(m)\mid l\}}\left(p\left(\sum_{l'\in L(m)\mid l}x'_l=0\mid x_l=0\right)\times\prod_{l'\in L(m)\mid l}q_{ml'}^{x_{l'}}\right)\tag{5.5.2}$$

另一个概率是 r_{ml}^1,当 $x_l=1$ 时,校验约束得到的概率定义为

$$r_{ml}^1=\sum_{\{x_{l'}:\,l'\in L(m)\mid l\}}\left(p\left(\sum_{l'\in L(m)\mid l}x'_l=1\mid x_l=1\right)\times\prod_{l'\in L(m)\mid l}q_{ml'}^{x_{l'}}\right)\tag{5.5.3}$$

其中,概率内的求和为模2求和,条件概率的取值为0或1,取决于假设的 x'_l 的值是否满足模2求和式,$L(m)\mid l$ 是 $L(m)$集合中去掉 l 后的其他元素。

上面这个复杂的计算可以根据前后向递归算法适当简化。定义

$$\delta_{q_{ml}}=q_{ml}^0-q_{ml}^1\tag{5.5.4}$$

可以得到

$$\delta_{r_{ml}}=r_{ml}^0-r_{ml}^1=\prod_{l'\in L(m)\mid l}\delta_{q_{ml'}}\tag{5.5.5}$$

利用数学归纳法可以得到 r_{ml}^x($x=0$ 或1)的计算式如下:

$$r_{ml}^x=\frac{1+(-1)^x\delta_{r_{ml}}}{2}\tag{5.5.6}$$

经过简化后,r_{ml}^x 的计算量极大减少。

② 利用步骤①计算所得到 r_{ml}^0 的 r_{ml}^1 和更新概率值 q_{ml}^0 和 q_{ml}^1。对于每个 l,计算下式:

$$q_{ml}^0=\alpha_{ml}p_l^0\prod_{m'\in M(l)\mid m}r_{m'l}^0\tag{5.5.7}$$

$$q_{ml}^1=\alpha_{ml}p_l^1\prod_{m'\in M(l)\mid m}r_{m'l}^1\tag{5.5.8}$$

其中,α_{ml} 为归一化系数,使得 $q_{ml}^0+q_{ml}^1=1$。

同时,计算出比特 l 取值为0和1的伪后验概率 q_l^0 和 q_l^1 如下:

$$q_l^0=\alpha_l p_l^0\prod_{m\in M(l)}r_{ml}^0\tag{5.5.9}$$

$$q_l^1=\alpha_l p_l^1\prod_{m\in M(l)}r_{ml}^1\tag{5.5.10}$$

③ 当 $q_l^1>0.5$ 时，令 $\hat{x}=1$，反之令 $\hat{x}=0$。如果校验方程 $\boldsymbol{H}\hat{x}=0 \bmod 2$ 成立，则译码成功并结束，$x=\hat{x}$；否则，若迭代次数少于预先设定的最大迭代次数 T，则重复该迭代过程；如果迭代次数已经超过所设定的最大迭代次数还是没有得到正确的译码，则译码失败。此时，部分译码的 $\hat{x}$ 可以作为其他译码算法的有效译码起点。

BP 算法的复杂度为码长的线性函数，并行实现可极大提高译码速度。BP 算法的译码误码率随信噪比的增加而减少，没有误码率下降急速的差错平底特性的现象。无论是在理论上还是在实际应用中，BP 译码都能够在高斯白噪声环境下达到接近信道限的性能。

MATLAB 代码如下：

```
clear all
close all
enc = comm.LDPCEncoder;                                          % 创建 LDPC 编码器
dec = comm.LDPCDecoder;                                          % 创建 LDPC 译码器
mod = comm.QPSKModulator('BitInput',true);                       % 创建调制器
chan = comm.AWGNChannel('NoiseMethod','Signal to noise ratio (SNR)');
 % AWGN 信道
demod = comm.QPSKDemodulator('BitOutput',true,...
  'DecisionMethod','Approximate log-likelihood ratio', ...
  'VarianceSource','Input port');                                % 创建解调器
err = comm.ErrorRate;
snrVec = [0 0.2 0.4 0.6 0.65 0.7 0.75 0.8];                      % 信噪比范围
ber = zeros(length(snrVec),1);
for k = 1:length(snrVec)
  chan.SNR = snrVec(k);
  noiseVar = 1/10^(snrVec(k)/10);
  errorStats = zeros(1,3);
  while errorStats(2)<=200 && errorStats(3)<5e6
      data = logical(randi([0 1],32400,1));                      % 产生二进制数据
      encData = step(enc,data);                                  % LDPC 编码
      modSig = step(mod,encData);                                % 调制
      rxSig = step(chan,modSig);                                 % AWGN 信道
      demodSig = step(demod,rxSig,noiseVar);                     % 解调
      rxData = step(dec,demodSig);                               % LDPC 译码
      errorStats = step(err,data,rxData);                        % 统计错误
  end
  ber(k) = errorStats(1);
  reset(err)
end
semilogy(snrVec,ber,'-o')
grid
xlabel('SNR (dB)')
ylabel('误比特率')
```

运行结果如图 5-27 所示。

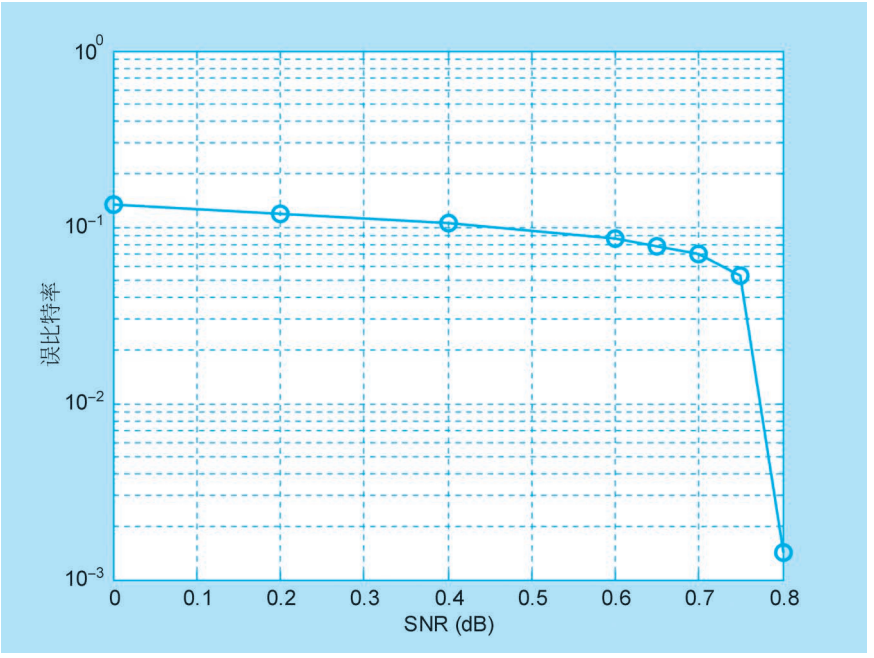


图 5-27 LDPC 编码结果

5.5.4 Polar 码

Polar 码是一种唯一被严格证明能够达到香农限的信道编码方法，现已成为 5G 中的中短码长纠错编码标准。2007 年，E. Arikan 基于信道极化理论首次提出 Polar 码。经过信道极化后，比特信道(bit-channel)被分为好与坏两种，而好比特信道通过 Polar 码的编码被选用于信息传输，极大地提升了信息传输的可靠性。

信道极化分为信道组合和信道分裂两个过程。其中，信道组合就是对二进制离散无记忆信道(B-DMC)W 进行独立复制，通过一定的递归规则得到矢量信道。例如，在 $n=0$ 时，对于 $N=2^n=1$ 信道，则是一个 B-DMC 信道，也就是 $W_1 \triangleq W$；当 $n=1, N=2^n=2$ 时，两个 W_1 的独立信道可组合成一个 W_2 信道，即 $W_2: X^2 \to Y^2$，其组合过程如图 5-28 所示，信道 W_2 的转移概率可表达为 $W_2(y_1, y_2 \mid u_1, u_2) = W(y_1 \mid u_1 \oplus u_2) W(y_2 \mid u_2)$，其中，$\oplus$为模 2 加。

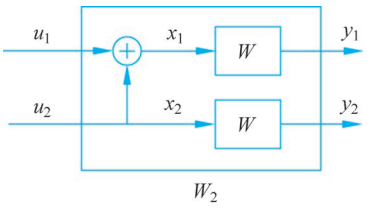


图 5-28 信道 W 到 W_2 的组合图

当 $n=2, N=4$ 时，由两个 W_2 的独立信道可组合成一个 W_4 信道 $X^4 \to Y^4$，如图 5-29 所示，其转移概率为 $W_4(y_1^4 \mid u_1^4) = W_2(y_1^2 \mid u_1 \oplus u_2, u_3 \oplus u_4) W_2(y_3^4 \mid u_2, u_4)$，其中 $y_1^4 = (y_1 y_2 y_3 y_4)$，$u_1^4 = (u_1 u_2 u_3 u_4)$，$\boldsymbol{R}_4$ 是一置换矩阵，将$(v_1 v_2 v_3 v_4)$置换为$(v_1 v_3 v_2 v_4)$。于是，W_4 的输入$(u_1 u_2 u_3 u_4)$与输入$(x_1 x_2 x_3 x_4)$间关系可表示为 $x_1^4 = u_1^4 \boldsymbol{G}_4$，其中 $\boldsymbol{G}_4 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}$。因此，$W_4$ 信道的转移概率可改写成

$$W_4(y_1^4 \mid u_1^4) = W^4(y_1^4 \mid u_1^4 \boldsymbol{G}_4) \tag{5.5.11}$$

该组合过程可以类推，即信道 W_N 可以由两个 $W_{N/2}$ 的独立信道组合得到，并且 W_N 的转移概率为

$$W_N(y_1^N \mid u_1^N) = W^N(y_1^N \mid u_1^N \boldsymbol{G}_N), \quad y_1^N \in Y^N, u_1^N \in X^N \tag{5.5.12}$$

其中，$\boldsymbol{G}_N = \boldsymbol{B}_N \boldsymbol{F}^{\otimes n}$，$\boldsymbol{B}_N$ 为置换矩阵，$\boldsymbol{F} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$。

信道分裂则是将组合后的信道 W_N 映射为 N 个比特信道，第 i 个比特信道的转移概率为

$$W_N^{(i)}(y_1^N, u_1^{i-1} \mid u_i) \triangleq \sum_{u_{i+1}^N \in X^{N-i}} \frac{1}{2^{N-1}} W_N(y_1^N \mid u_1^N) \tag{5.5.13}$$

其中，(y_1^N, u_1^{i-1})是 $W_N^{(i)}$ 的输出，u_i 是 $W_N^{(i)}$ 的输入，$i = 1, 2, \cdots, N$。图 5-30 展示了 $N=4$ 时从信道 W 的 4 个复制到 $W_4^{(i)}$ 的过程。最右端是 4 个独立信道 W，通过蝶形运算得到两个矢量信道$(W_2^{(1)}, W_2^{(2)})$，对两个矢量信道$(W_2^{(1)}, W_2^{(2)})$再一次蝶形运算得到 $W_4^{(i)}$ 的每个比特信道。其中，蝶形结构右边两个结点代表相同且独立的信道，左边即是信道组合后分裂的结果，此过程可以不断进行。

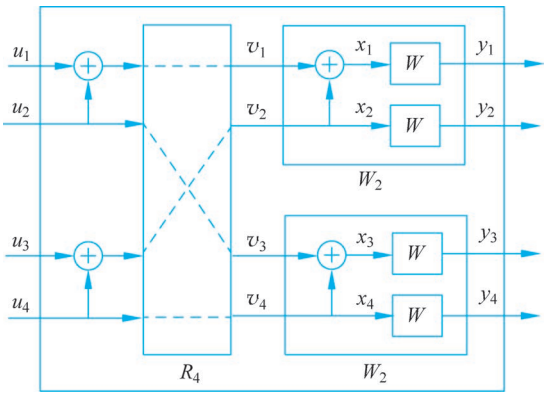


图 5-29 信道 W_2 到 W_4 的组合图

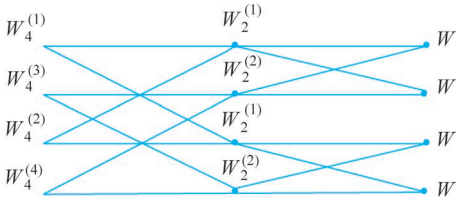


图 5-30 从 W 到 W_4 的变换过程示意图

根据信道极化理论，随着 N 逐渐增大，信道分裂得到的每一个比特信道的信道容量呈现出两极分化的趋势，即有一些比特信道的容量逐渐接近 1，而另一些比特信道的容量逐渐接近 0。因此，在实际的通信过程中，选择那些容量接近 1 的比特信道来传输信息，称为信息比特，而将容量接近 0 的比特信道赋予固定值（通常为全 0），称为冻结比特，冻结比特这部分信息是已知的。信息比特和冻结比特所在位的集合分别被称为信息位和冻结位。

在 Polar 码中，比特信道的好与坏一般可用巴氏（Bhattacharyya）参数来衡量。巴氏参数 $Z(W)$代表的是信道输入为 0 或 1 时，信息被错误判决的概率上界，其定义为

$$Z(W_N^{(i)}) = \sum_{y_1^N \in Y^N} \sum_{u_1^{i-1} \in X^{i-1}} \sqrt{W_N^{(i)}(y_1^N, u_1^{i-1} \mid 0) W_N^{(i)}(y_1^N, u_1^{i-1} \mid 1)} \tag{5.5.14}$$

1）Polar 码的构造

Polar 码的构造就是比特信道选择的过程，即选择那些好的比特信道来传输信息，这里的好是指巴氏参数趋近 0，或信道容量趋近 1。

对于 B-DMC 信道，Polar 码可表现为三元组参数(W,N,K)，其中，W 表示 B-DMC 信道，N 为 Polar 码的码长，K 是 Polar 码信息位的长度，信息位集合为 $A\subset\{1,2,\cdots,N\}$。Polar 码构造的目标是使 $\sum_{i\in A}Z(W_N^{(i)})$ 最小，即计算各比特信道巴氏参数$\{Z(W_N^{(i)}):1\leqslant i\leqslant N\}$，并使信息位集合中的巴氏参数之和为最小，其中，$Z(W_N^{(i)})$为第 i 比特信道的巴氏参数。从实现角度来讲，Polar 码的构造问题转换为判决问题，即在给定了信道下标的索引号 i，$i\in\{1,2,\cdots,N\}$和门限值 γ 的情况下($\gamma\in[0,1]$)，依据 $A_\gamma\triangleq\{i\in\{1,2,\cdots,N\}:Z(W_N^{(i)})<\gamma\}$来判决下标 i 对应的比特信道是否用来传输信息。

总的来说，Polar 码的构造就是 K 个信息位索引号的选取。因此，通常，首先计算出各比特信道巴氏参数的值，然后对计算出的巴氏参数值从小到大排序，最后挑选出前 K 个较小的巴氏参数值的比特信道索引组合成信息位，剩余的比特信道则为冻结位。常见的构造方法有基于 Polar 码的蒙特卡罗构造方法、基于 BEC 的构造方法、密度进化构造方法、Tal-Vardy 方法和高斯近似法等。

作为一种线性分组码，Polar 码也可用生成矩阵来描述。给定长度为 N 的 Polar 码，其中 $N=2^n$，其编码 $x_1^N=u_1^N\boldsymbol{G}_N$，其中，$x_1^N$ 是编码码字，u_1^N 为信息序列，$\boldsymbol{G}_N$ 为 $N\times N$ 的生成矩阵，且 $\boldsymbol{G}_N=\boldsymbol{B}_N\boldsymbol{F}^{\otimes n}$，$\boldsymbol{B}_N$ 为置换矩阵，$\boldsymbol{F}=\begin{bmatrix}1&0\\1&1\end{bmatrix}$。通常情况下，信息位集合用 A 表示，冻结位集合则用 A^c 表示。若 u_A 表示信息位信息，u_A^c 表示冻结位信息，则编码过程可表示为

$$x_1^N=u_A\boldsymbol{G}_N(A)\oplus u_{A^c}\boldsymbol{G}_N(A^c) \tag{5.5.15}$$

其中，$\boldsymbol{G}_N(A)$由矩阵 $\boldsymbol{G}_N$ 中对应于集合 A 中各元素的行构成，$\boldsymbol{G}_N(A^c)$由矩阵 $\boldsymbol{G}_N$ 中对应于集合 A^c 中各元素的行构成，且冻结位的取值对译码性能没有影响，常将置为 0。于是，Polar 码这种线性分组码编码用 4 个参数(N,K,A,u_{A^c})表示，其中 N 为码长，K 为信息位的个数，A 和 A^c 分别是信息位和冻结位，u_{A^c} 是冻结位信息。

2) Polar 的译码

Polar 码的译码是将接收到的 y_1^N 恢复成发送信息序列 u_1^N 的过程，一般来说，译码方法会与其编码方法相适应。针对 Polar 码的结构，Arikan 提出了连续删除(Successive Cancellation，SC)译码和置信传播(Belief Propagation，BP)译码算法，下面简单介绍 SC 译码。

SC 译码是 Polar 码的核心译码算法，该算法是基于比特信道的概率似然比通过硬判决值进行译码，一比特一比特地译码，直到最后一个比特译码结束后才得到译码输出。

对于参数为(N,K,A,u_{A^c})的 Polar 码，当要传输的信息序列与生成矩阵相乘后得到编码，再通过信道传输，在接收端获得接收序列 y_1^N，其中，信息序列包括信息位 u_A 和冻结位 u_{A^c}。由于冻结位在编译码过程中保持不变，是已知的，所以在译码时如果是冻结位则直接译码，其估计值即为冻结位信息，$\hat{u}_{A^c}=u_{A^c}$；如果是信息位，则通过下式判别当前比特的估计值

$$h_i(y_1^N,\hat{u}_1^{i-1})=\begin{cases}0, & \dfrac{W_N^{(i)}(y_1^N,\hat{u}_1^{i-1}\mid 0)}{W_N^{(i)}(y_1^N,\hat{u}_1^{i-1}\mid 1)}\geqslant 1\\ 1, & \text{其他}\end{cases} \tag{5.5.16}$$

在译码过程中，通常把转移概率的比值定义为似然比(Likelihood Ratio，LR)

$$L_N^{(i)}(y_1^N \mid \hat{u}_1^{i-1}) = \frac{W_N^{(i)}(y_1^N, \hat{u}_1^{i-1} \mid 0)}{W_N^{(i)}(y_1^N, \hat{u}_1^{i-1} \mid 1)} \tag{5.5.17}$$

将式(5.5.16)代入式(5.5.17)中，得到当前信息位的估值为

$$\hat{u}_i = \begin{cases} 0, & L_N^{(i)}(y_1^N, \hat{u}_1^{i-1}) \geqslant 1 \\ 1, & \text{其他} \end{cases} \tag{5.5.18}$$

SC 译码算法的计算复杂度主要取决于译码过程中似然比的计算，而对于似然比的计算，Arikan 给出了一种递归迭代计算方法：

$$L_N^{(2i-1)}(y_1^N, \hat{u}_1^{2i-2}) = \frac{L_{N/2}^{(i)}(y_1^{N/2}, \hat{u}_{1,o}^{2i-2} \oplus \hat{u}_{1,e}^{2i-2}) L_{N/2}^{(i)}(y_{N/2+1}^N, \hat{u}_{1,e}^{2i-2}) + 1}{L_{N/2}^{(i)}(y_1^{N/2}, \hat{u}_{1,o}^{2i-2} \oplus \hat{u}_{1,e}^{2i-2}) + L_{N/2}^{(i)}(y_{N/2+1}^N, \hat{u}_{1,e}^{2i-2})} \tag{5.5.19}$$

$$L_N^{(2i)}(y_1^N, \hat{u}_1^{2i-1}) = [L_{N/2}^{(i)}(y_1^{N/2}, \hat{u}_{1,o}^{2i-2} \oplus \hat{u}_{1,e}^{2i-2})]^{1-2\hat{u}_{2i-1}} \cdot L_{N/2}^{(i)}(y_{N/2+1}^N, \hat{u}_{1,e}^{2i-2}) \tag{5.5.20}$$

其中，$L_N^{(2i-1)}$ 表示 N 长序列中第$(2i-1)$比特标号(奇数)的似然比，$L_N^{(2i)}$ 是 N 长序列中第$(2i)$比特标号(偶数)的似然比，$\hat{u}_{1,o}^{2i-2}$ 是 $\hat{u}_1^{2i-2}$ 序列中奇数位构成的序列，$\hat{u}_{1,e}^{2i-2}$ 是 $\hat{u}_1^{2i-2}$ 序列中偶数位构成的序列。从上式可以看出，计算长度为 N 的似然比可以转换为计算两个长度为 $N/2$ 的 LR，即似然对$(L_N^{(2i-1)}(y_1^N, \hat{u}_1^{2i-2}), L_N^{2i}(y_1^N, \hat{u}_1^{2i-1}))$可以转换为计算似然对$(L_{N/2}^{(i)}(y_1^{N/2}, \hat{u}_{1,o}^{2i-2} \oplus \hat{u}_{1,e}^{2i-2}), L_{N/2}^{(i)}(y_{N/2+1}^N, \hat{u}_{1,e}^{2i-2}))$，当然 $N/2$ 的 LR 继续利用式(5.5.19)和式(5.5.20)，由 $N/4$ 的 LR 似然对$(L_{N/4}^{(i)}(y_1^{N/4}, \hat{u}_{1,o}^{2i-2} \oplus \hat{u}_{1,e}^{2i-2}), L_{N/4}^{(i)}(y_{N/4+1}^{N/2}, \hat{u}_{1,e}^{2i-2}))$得到，以此类推，直到码长为 1 时终止。当码长为 1 时，LR 的值由公式 $L^{(i)}(y_i) = W(y_i \mid 0)/W(y_i \mid 1)$ 直接计算得到。

图 5-31 显示了码长为 8 的 Polar 码的 SC 译码的因子图结构，其中，编码器参数为$(8,5,\{4,5,6,7,8\},(0,0,0))$。在该过程中，每个负责计算的节点会向下发起一个 LR 的请求，从最左边到最右边依次进行。在最左边的第 1 列节点对应码长为 8 的 LR 请求，相应的第 2 列对应码长为 4 的 LR 请求，第 3 列对应码长为 2 的 LR 请求，第 4 列对应码长为 1 的 LR 请求。在图中每一个节点都对应两个标签，例如，第 2 列第 2 个节点带有 22 和$(y_1^4, \hat{u}_{1,e}^4 \oplus \hat{u}_{1,o}^4)$两个标签，其中，第 1 个标签指的是译码过程中该节点第 22 个被激活，第 2 个标签指的是要计算且保存的 LR 值，其中，数字 1～32 表明一共需要计算 32 个 LR 值。

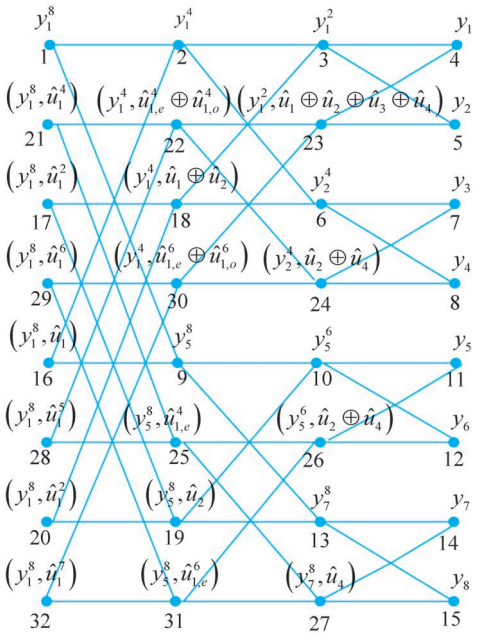


图 5-31 Polar 码的 SC 译码过程

译码时先从第 1 个节点开始激活，首先计算 $L_8^{(1)}(y_1^8)$，但是 $L_8^{(1)}(y_1^8)$ 的值并不能直接得到，而是触发节点 2 和节点 9，在触发这两个节点之后，节点 1 开始等待节点 2 和节点 9 返回的 LR 的值；先激活节点 2 进行 $L_4^{(1)}(y_1^4)$ 的计

算,同样,节点 2 继续传递,触发节点 3,节点 3 再触发节点 4,到节点 4 时已经到了最右边一列,通过给定的信道信息可直接计算出 $L_1^{(1)}(y_1)$,于是可将计算结果返回给触发节点 3 和节点 23。对于节点 3,除了获得节点 4 返回值外,还需要继续向右侧传递,触发节点 5 获得 $L_1^{(1)}(y_2)$的值。对节点 4 和节点 5 的返回信息计算可得到节点 3 的 LR 值 $L_2^{(1)}(y_1^2)$,并将其 LR 值返回给节点 2。同前面的过程一样,节点 2 还需向右依次触发节点 6、节点 7 和节点 8,获得节点 2 的 LR 值 $L_4^{(1)}(y_1^4)$,并将结果返回给节点 1。同理,节点 1 还需要节点 9 的返回值,节点 9 的 LR 值计算过程如上类似。当节点 9 的 LR 值 $L_8^{(1)}(y_5^8)$返回给节点 1 后,节点 1 整合计算出 $L_8^{(1)}(y_1^8)$,得到了第 1 个比特的似然值,由于 u_1 是冻结位,因此直接置为 0,并将其似然值传递给下一个信息比特,即触发节点 16。

同第 1 个比特激活的过程一样,节点 16 触发后向右侧传递,需要节点 2 和节点 9 的值,而这两个节点的 LR 值已经保存,因此直接将已经获得的 $L_4^{(1)}(y_1^4)$和 $L_8^{(1)}(y_5^8)$两个 LR 值计算得到 $L_8^{(2)}(y_1^8,\hat{u}_1)$,由于 u_2 是冻结位,因此也直接置 $u_2=0$,并传递给下一个信息比特。

计算下一个比特时,需激活节点 17,计算节点 17 的值需要节点 18 和节点 19 的 LR 值,而节点 18 和节点 19 的 LR 值在前面的过程中已经被保存下来,因此可以直接计算得到 $L_8^{(3)}(y_1^8,\hat{u}_1^2)$。由于 u_3 是信息比特,因此需要根据得到的 LR 值来估计 u_3,当 $L_8^{(3)}(y_1^8,\hat{u}_1^2)\geqslant 1$ 时,估值为 0,否则估值为 1。得到估计值后继续传递到下一比特的计算,直到依次得到 u_1^8 的所有估值。

通过上面 SC 译码过程可以看到,译码器由蝶形组成,例如,节点 2、3、18、6 构成一个蝶形,以节点 2 和节点 18 为根的计算子树又可以继续分裂后得到以节点 3 和节点 6 为根的译码子树。显然,在一个以 4 个节点构成的蝶形进行译码时,最先被激活的是最左上方的节点,最后被激活的是左下方的节点。

因此,蝶形图在 LR 的计算上有时间约束,任何一个蝶形图,计算左下方的节点需要提供左上方节点的 LR 值,而只有得到右侧节点的 LR 值才能计算出左上方节点的 LR 值。因此,可以说 SC 译码算法是一种串行迭代过程,计算 LR 值的等待时间将造成译码的较大时延,且 SC 译码是一种顺序译码,前面的译码结果会对后续的译码产生影响。

若 $P_e(N,K,A,u_{A^c})$表示 Polar 码译码差错率,则其表达式为

$$P_e(N,K,A,u_{A^c})=\sum_{u_A\in X^K}\frac{1}{2^K}\sum_{y_1^N\in Y^N:\hat{u}_1^N(y_1^N)\neq u_1^N}W_N(y_1^N\mid u_1^N) \tag{5.5.21}$$

可以获得它与信息位巴氏参数之间的关系为

$$P_e(N,K,A,u_{A^c})\leqslant\sum_{i\in A}Z(W_N^{(i)}) \tag{5.5.22}$$

因此,当信息位的选择使得 $\sum_{i\in A}Z(W_N^{(i)})$ 足够小时,如趋于 0,则译码的差错概率也趋于 0。

SC 译码 MATLAB 代码如下:

```
function u_finish = decode(n,y,frozen_positions,frozen_bits)
%(阶数 接收值 冻结位序号 冻结位数字)
global sigma r_value r_flag max_number min_number
N = 2^n;                                          %N 为码长,n 为阶数
r_value(:,n + 1) = exp( - 2 * y'./(sigma^2));     %根据信道接收值和信噪比计算 LR
r_flag(:,n + 1) = ones(N,1);                      %定义因子图的矩阵
```

```
u_finish = [];
t = 1;
for i = 1:1:N
  likelihood_ratio = ww(n,(1:N),i,u_finish);
% (当前阶数 初始接收值序号 信道序号 译完比特位)
  if ismember(i,frozen_positions)                    % 判断是否为冻结位
      u_finish(i) = frozen_bits(t);                  % 冻结位信息发送双方已知,直接赋值
      t = t + 1;
  else if likelihood_ratio > 1          % 如果不是冻结位,根据最左端的节点信息来判决译码结果
          u_finish(i) = 0;              % 如果 if  L_N^(i)(y_1^N,û_1^(i-1))≥1,译码结果为 0
      else u_finish(i) = 1;             % 否则译码结果为 1
      end
  end
end
end

function likelihood_ratio = ww(n,y,i,u)
% 计算似然比(阶数 接收值序号 信道序号 译完比特位)
global r_value r_flag nn max_number min_number
row = y(1) + bit_reverse(i,n) - 1;                   % 对应在大矩阵中的行号
   col = nn + 1 - n;                                 % row 是实际在大矩阵中的列号
   if r_flag(row,col) == 1
     likelihood_ratio = r_value(row,col);
   else
      k = 2^n;
    y1 = y(1:k/2);
    y2 = y(k/2 + 1:k);
    kk = round(i/2);                                 % 向上取整
    uue = u(2:2:2 * kk - 2);
    uuo = u(1:2:2 * kk - 2);
    uu = mod(uue + uuo,2);
    if mod(i,2) == 0
  likelihood_ratio = feven(ww(n - 1,y1,kk,uu),ww(n - 1,y2,kk,uue),u(2 * kk - 1));      % 递归
    if likelihood_ratio > max_number
      likelihood_ratio = max_number;
       end
    else likelihood_ratio = fodd(ww(n - 1,y1,kk,uu),ww(n - 1,y2,kk,uue));
       if likelihood_ratio > max_number
           likelihood_ratio = max_number;
        end
    end
    r_value(row,col) = likelihood_ratio;
    r_flag(row,col) = 1;
      end
  end

function likelihood_ratio = fodd(a,b)                % 奇数节点似然信息计算
  likelihood_ratio = (a * b + 1)/(a + b);
  end
function likelihood_ratio = feven(a,b,c)             % 偶数节点似然信息计算
```

```
    likelihood_ratio = a^(1 - 2 * c) * b;
  end
function num = bit_reverse(i,n)
    num = 1;
    for k = n: - 1:1
      num = num + mod(i - 1,2) * 2^(k - 1);
      i = round(i/2);
    end
end
```

码长 1024、码率 0.5 的 Polar 码在高斯白噪声信道下的运行结果如图 5-32 所示。

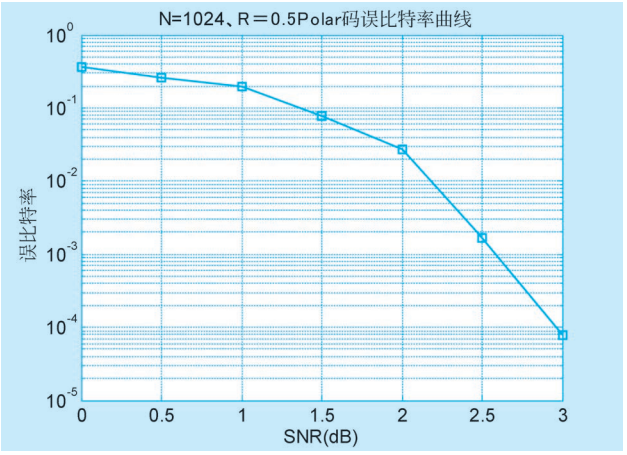


图 5-32 Polar 码编码结果

习题 5

5-1 设一个离散无记忆信道,其信道矩阵为

$$\begin{bmatrix} \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 \\ 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} \end{bmatrix}$$

(1) 计算信道容量 C。

(2) 找出一个长度为 2 的码,其信息传输率为(log 5)/2(5 个码字),如果按最大似然译码准则设计译码器,计算其译码器输出端平均错误译码概率 P_e(设输入码字等概)。

(3) 有没有可能存在一个长度为 2 的码,使得每个码字的平均错误译码概率 $P_e^{(i)}=0(i=$

1,2,3,4,5),即对所有码字有 $p_e=0$? 如果存在,请找出来。

5-2 设离散无记忆信道的输入符号集 $X\in\{0,1\}$,输出符号集 $Y\in\{0,1,2\}$,信道矩阵为$[p_{ji}]=\begin{bmatrix}\frac{1}{2} & \frac{1}{4} & \frac{1}{4}\\ \frac{1}{4} & \frac{1}{2} & \frac{1}{4}\end{bmatrix}$。若某信源输出两个等概消息 s_1 和 s_2,现用信道输入符号集中的符号对 s_1 和 s_2 进行信道编码,以 $\omega_1=00$ 代表 s_1,用 $\omega_2=11$ 代表 s_2,试写出能使平均错误译码概率 $p_e=p_{emin}$ 的译码规则,并计算平均错误译码概率。

5-3 设有一离散无记忆信道,其信道矩阵为

$$\boldsymbol{P}=\begin{bmatrix}1/2 & 1/3 & 1/6\\ 1/6 & 1/2 & 1/3\\ 1/3 & 1/6 & 1/2\end{bmatrix}$$

若 $p(x_1)=1/2$,$p(x_2)=p(x_3)=1/4$,试求最佳译码时的平均错误概率。

5-4 写出构成二元域上 4 维 4 重矢量空间的全部矢量元素,并找出其中一个三维子空间及其相应的对偶子空间。

5-5 一个线性分组码的监督矩阵为

$$\boldsymbol{H}=\begin{bmatrix}1&0&0&1&0&0&1&1&0\\ 1&0&1&0&1&0&0&1&0\\ 0&1&1&1&0&0&0&0&1\\ 1&0&1&0&1&1&1&0&1\end{bmatrix}$$

求其生成矩阵以及码的最小距离 d_{min},并画出该编码器硬件逻辑连接图。

5-6 将例 5-9 的(7,4)系统码缩短为(5,2)码,写出缩短码的生成矩阵、校验矩阵,及所有码字。

5-7 列出例 5-9 的(7,4)汉明码的标准阵列译码表。若接收码 $R=$(0010100,0111000,1110010),由标准阵列译码表判断发送码是什么。

5-8 某线性二进码的生成矩阵为

$$\boldsymbol{G}=\begin{bmatrix}0&0&1&1&1&0&1\\ 0&1&0&0&1&1&1\\ 1&0&0&1&1&1&0\end{bmatrix}$$

(1) 用系统码 $[\boldsymbol{I}\vdots\boldsymbol{P}]$ 的形式表示 $\boldsymbol{G}$。

(2) 计算该码的校验矩阵 $\boldsymbol{H}$。

(3) 列出该码的伴随式表。

(4) 计算该码的最小距离。

(5) 证明:与信息序列 101 相对应的码字正交于 $\boldsymbol{H}$。

5-9 设一个(15,4)循环码的生成多项式为 $g(x)=1+x+x^5+x^6+x^{10}+x^{11}$。试求:

(1) 此码的监督多项式 $h(x)$。

(2) 此码的生成矩阵(非系统码和系统码形式)。

(3) 此码的监督矩阵。

5-10 设计一个(15,11)系统汉明码的生成矩阵 $\boldsymbol{G}$,以及由 $g(x)=1+x+x^4$ 生成的系

统 (15,11)循环汉明码的编码器。

5-11 计算(7,4)系统循环汉明码最小重量的可纠差错图案和对应的伴随式。

5-12 某帧所含信息是(00001101011000101011 00),循环冗余校验码的生成多项式是 CRC-ITU-T 规定的 $g(x)=x^{16}+x^{12}+x^5+1$。问附加在信息位后的 CRC 校验码是什么?

5-13 某卷积码(2,1,3)的转移函数矩阵是 $\boldsymbol{G}(D)=[1+D^2,1+D+D^2+D^3]$。

(1) 画出编码器结构图。

(2) 画出编码器的状态图。

(3) 求该码的自由距离 d_f。

5-14 某(2,1,4)卷积码,其两脉冲冲激响应(当输入信息为 100…时所观察到的输出序列值)为 $g^1=(11101),g^2=(10011)$。

(1) 画出该编码器结构图。

(2) 写出该码的生成矩阵。

(3) 若输入信息序列 $\boldsymbol{m}=[11010001]$,求相应的码序列。

5-15 某码率为 1/2、约束长度 $K=3$ 的二进制卷积码,其编码器如图 5-33 所示。

(1) 画出状态图和网格图。

(2) 求转移函数 $T(D)$,据此指出自由距离。

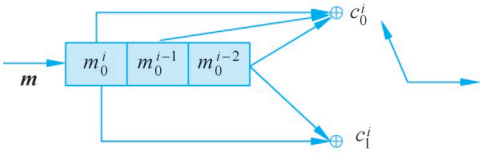


图 5-33 习题 5-15 图

5-16 某卷积码 $\boldsymbol{G}^0=[1\quad 0\quad 0],\boldsymbol{G}^1=[1\quad 0\quad 1],\boldsymbol{G}^2=[1\quad 1\quad 1]$。

(1) 画出该码的编码器。

(2) 画出该码的状态图和网格图。

(3) 求出该码的转移函数和自由距离。

5-17 某卷积码如图 5-34 所示。

(1) 画出该码的状态图。

(2) 求转移函数 $T(D)$。

(3) 求该码的自由距离 d_f,在格栅图上画出相应路径(与全 0 码字相距 d_f 的路径)。

(4) 当利用 BSC 信道进行数据传输时,接收序列为 10 01 00 11 10 00 00,试用维特比算法对接收序列进行译码的最可能原始信息序列。

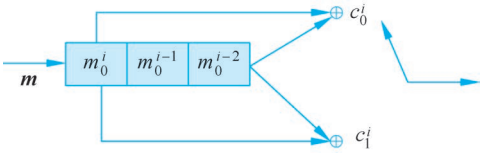


图 5-34 习题 5-17 图