第3章 线性分组码

目前，几乎所有得到实际应用的纠错码都是线性的。线性分组码是整个纠错码中很重要的一类码，也是讨论各类码的基础。它概念清楚，易于理解，而且能方便地引出各类码中广为采用的一些基本参数和名称的定义，因此本章的内容将涉及整个纠错码的基本知识，重点学习线性分组码的构成理论及其编码译码方法。

3.1 线性分组码的定义

将信源的输出序列分成长为 k 的段 $\boldsymbol{u}=(u_{k-1},u_{k-2},\cdots,u_1,u_0)$，序列中的每一分量都是一随机变量。

为了能够纠错，信道编码器（纠错码编码器）按一定的规则将长为 k 的段 $\boldsymbol{u}=(u_{k-1},u_{k-2},\cdots,u_1,u_0)$ 编为长为 n 的码字（码符号序列）$\boldsymbol{c}=(c_{n-1},c_{n-2},\cdots,c_1,c_0)$（$n>k$）。码字共有 n 位，其中 k 位为信息位，$(n-k)$ 位为校验位（监督位）。假设共有 M 个消息序列，则对应的 M 个码字的集合 $\{\boldsymbol{c}_1,\boldsymbol{c}_2,\cdots,\boldsymbol{c}_M\}$ 称为一个 (n,k) 分组码，记为 C。

注：以上文中的 $\boldsymbol{u}=(u_{k-1},u_{k-2},\cdots,u_1,u_0)$ 和 $\boldsymbol{c}=(c_{n-1},c_{n-2},\cdots,c_1,c_0)$，下标采用的是倒序的形式，如果采用顺序的形式如 $\boldsymbol{u}=(u_0,u_1,\cdots,u_{k-2},u_{k-1})$ 和 $\boldsymbol{c}=(c_0,c_1,\cdots,c_{n-2},c_{n-1})$ 也是可以的，一般根据需要和方便性来选择使用。

在分组码中，若 $\boldsymbol{u}$ 与 $\boldsymbol{c}$ 的对应关系是线性的（可以用一次线性方程来描述），则称为线性分组码。

对二进制而言，假设 $\boldsymbol{c}_i$ 和 $\boldsymbol{c}_j$ 是 (n,k) 二进制分组码中的两个码字（或码矢量），当且仅当 $\boldsymbol{c}_i+\boldsymbol{c}_j$ 也是一个码字时，这个码才是线性的；特别地，当 $\boldsymbol{c}_i=\boldsymbol{c}_j$ 时，$\boldsymbol{c}_i+\boldsymbol{c}_j=\boldsymbol{0}$。由此可见，二元分组码是线性分组码的充分条件是两个码字的模 2 和也是码字；或者说，一个线性分组码，它的子集以外的矢量不能由该子集内的码字相加产生。

根据上面的定义，分组码的线性只与选用的码字有关，而与消息序列怎样映射到码字无关。

例 3-1 有一个 $(5,2)$ 分组码 $\boldsymbol{C}=\{00000,01011,10101,11110\}$，假设消息序列与码字的映射为 00→00000，01→01011，10→10101，11→11110，容易验证它是线性的；如果映射关系变为 00→11110，01→10101，10→01011，11→00000，容易验证它仍是线性的。

同时也容易验证第一种映射关系满足如下关系：如果消息序列 $\boldsymbol{u}_1$ 映射为码字 $\boldsymbol{c}_1$，$\boldsymbol{u}_2$ 映射为码字 $\boldsymbol{c}_2$，则 $\boldsymbol{u}_1+\boldsymbol{u}_2$ 映射为码字 $\boldsymbol{c}_1+\boldsymbol{c}_2$。而第二种映射关系尽管是线性的，却不满足

上述的映射关系。

满足 $\boldsymbol{u}_1+\boldsymbol{u}_2$ 映射为码字 $\boldsymbol{c}_1+\boldsymbol{c}_2$ 的关系为线性分组码的特殊关系。一般来说，在线性分组码中，只要全零的消息序列映射为全零的码字，都能满足这种特殊关系。

3.2　生成矩阵和校验矩阵

3.2.1　生成矩阵

根据线性分组码的定义，可以得出如下所述的一种构成线性分组码的方法。

在(n,k)线性分组码中，取 k 个消息序列(序列中只含有一位“1”符号)分别为

$$\begin{cases}\boldsymbol{U}_1=(1000\cdots00)\\ \boldsymbol{U}_2=(0100\cdots00)\\ \boldsymbol{U}_3=(0010\cdots00)\\ \quad\vdots\\ \boldsymbol{U}_{k-1}=(0000\cdots10)\\ \boldsymbol{U}_k=(0000\cdots01)\end{cases}\tag{3-1}$$

这 k 个消息序列的长度都是 k 位，其对应的码字分别为 $\boldsymbol{g}_1,\boldsymbol{g}_2,\boldsymbol{g}_3,\cdots,\boldsymbol{g}_{k-1},\boldsymbol{g}_k$，均是长度为 n 的二进制序列，如 $\boldsymbol{g}=(g_{11},g_{12},\cdots,g_{1n})$，这样，对于任意的消息序列 $\boldsymbol{u}=(u_1,u_2,u_3,\cdots,u_{k-1},u_k)$，都可以用行矩阵表示为

$$\boldsymbol{u}=\sum_{i=1}^{k}u_i\boldsymbol{U}_i\tag{3-2}$$

对应的码字为

$$\boldsymbol{C}=\sum_{i=1}^{k}u_i\boldsymbol{g}_i=(c_1,c_2,\cdots,c_n)\tag{3-3}$$

定义

$$\boldsymbol{G}=\begin{bmatrix}\boldsymbol{g}_1\\ \boldsymbol{g}_2\\ \vdots\\ \boldsymbol{g}_k\end{bmatrix}=\begin{bmatrix}g_{11}&\cdots&g_{1n}\\ g_{21}&\cdots&g_{2n}\\ \vdots&\ddots&\vdots\\ g_{k1}&\cdots&g_{kn}\end{bmatrix}\tag{3-4}$$

为该分组码的生成矩阵，则有

$$\boldsymbol{C}=\boldsymbol{u}\boldsymbol{G}\tag{3-5}$$

对于某(7,3)线性分组码，$\boldsymbol{u}=(u_2,u_1,u_0)$，$\boldsymbol{C}=(c_6,c_5,c_4,c_3,c_2,c_1,c_0)$，若 $c_6=u_2$，$c_5=u_1$，$c_4=u_0$，按以下的编码规则(校验方程)可得到 4 个校验元 c_0、c_1、c_2 和 c_3。

$$\begin{cases}c_3=u_2+u_0\\ c_2=u_2+u_1+u_0\\ c_1=u_2+u_1\\ c_0=u_1+u_0\end{cases}$$

由此可计算得到该(7,3)线性分组码的8个码字，8个信息组与8个码字的对应关系见表3-1。由此方程可见，信息元与校验元满足线性关系。

表3-1 (7,3)码字与信息组的对应关系

信息组	码字	信息组	码字
000	0000000	100	1001110
001	0011101	101	1010011
010	0100111	110	1101001
011	0111010	111	1110100

从表中的8个码字中，挑选$k=3$个信息组(100)、(010)和(001)，其对应的线性无关的码字分别为(1001110)、(0100111)和(0011101)组成$\boldsymbol{G}$的行，得

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

若信息组为$\boldsymbol{u}=(011)$，则相应的码字为

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

由此可见，生成矩阵$\boldsymbol{G}$提供了一种简明而有效地表示线性分组码的方法。$k\times n$阶矩阵可以生成2^k个码字。因此，我们只需要一个生成矩阵而不需要含2^k个码字的查询表。这对大码的储存空间是极大的节省。

实际上，码的生成矩阵还可以由其编码方程直接得出。例如，上述(7,3)线性分组码，$c_6=u_2$，$c_5=u_1$和$c_4=u_0$是三个信息元，按以下的规则(校验方程)可得到4个校验元c_0、c_1、c_2和c_3。

$$\begin{cases}c_3=u_2+u_0\\c_2=u_2+u_1+u_0\\c_1=u_2+u_1\\c_0=u_1+u_0\end{cases}$$

可将编码方程改为

$$\begin{cases}c_6=u_2\\c_5=u_1\\c_4=u_0\\c_3=u_2+u_0\\c_2=u_2+u_1+u_0\\c_1=u_2+u_1\\c_0=u_1+u_0\end{cases}$$

写成矩阵形式为

$$\begin{aligned}\begin{bmatrix}c_6 & c_5 & c_4 & c_3 & c_2 & c_1 & c_0\end{bmatrix} &= \begin{bmatrix}c_6\\c_5\\c_4\\c_3\\c_2\\c_1\\c_0\end{bmatrix}^{\mathrm{T}} = \begin{bmatrix}u_2 & & \\ & u_1 & \\ & & u_0\\ u_2 & & +u_0\\ u_2 & +u_1 & +u_0\\ u_2 & +u_1 & \\ & u_1 & +u_0\end{bmatrix}^{\mathrm{T}}\\ &= \begin{bmatrix}u_2 & u_1 & u_0\end{bmatrix}\begin{bmatrix}1&0&0&1&1&1&0\\0&1&0&0&1&1&1\\0&0&1&1&1&0&1\end{bmatrix}\\ &= \begin{bmatrix}u_2 & u_1 & u_0\end{bmatrix}\cdot\boldsymbol{G}=\boldsymbol{u}\cdot\boldsymbol{G}\end{aligned}$$

因此生成矩阵为

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

例 3-2　线性分组码为 $n=7,k=4$，记为$(7,4)$码。

信源符号 4 位一组为 $\boldsymbol{u}=(u_3,u_2,u_1,u_0)$，$u_i\in\{0,1\}$，$i=0,1,2,3$；

码符号 7 位一组为 $\boldsymbol{c}=(c_6,c_5,c_4,c_3,c_2,c_1,c_0)$，$c_j\in\{0,1\}$，$j=0,1,2,3,4,5,6$；

若码符号与信源符号的关系为

$$\begin{cases}c_6=u_3\\c_5=u_2\\c_4=u_1\\c_3=u_1+u_2+u_3\\c_2=u_0\\c_1=u_0+u_2+u_3\\c_0=u_0+u_1+u_3\end{cases}$$

式中，c_6、c_5、c_4、c_2 为信息位，c_3、c_1、c_0 为校验位，码符号位是信息位的线性组合，用矩阵表示如下：

$$\begin{bmatrix}c_6 & c_5 & c_4 & c_3 & c_2 & c_1 & c_0\end{bmatrix}=\begin{bmatrix}u_3 & u_2 & u_1 & u_0\end{bmatrix}\begin{bmatrix}1&0&0&1&0&1&1\\0&1&0&1&0&1&0\\0&0&1&1&0&0&1\\0&0&0&0&1&1&1\end{bmatrix}$$

$\boldsymbol{G}=\begin{bmatrix}1&0&0&1&0&1&1\\0&1&0&1&0&1&0\\0&0&1&1&0&0&1\\0&0&0&0&1&1&1\end{bmatrix}$，即为生成矩阵。

表 3-2 列出了按上式生成的 16 个码字。

表 3-2 (7,4)线性分组码

信息组	码字	信息组	码字
0000	0000000	1000	1001011
0001	0000111	1001	1001100
0010	0011001	1010	1010010
0011	0011110	1011	1010101
0100	0101010	1100	1100001
0101	0101101	1101	1100110
0110	0110011	1110	1111000
0111	0110100	1111	1111111

例 3-3 考虑一个生成矩阵 $\boldsymbol{G}=\begin{bmatrix}1 & 0 & 1\\ 0 & 1 & 0\end{bmatrix}$，信息字为[00]、[01]、[10]、[11]，则有

$$\boldsymbol{c}_1=[0\quad 0]\begin{bmatrix}1 & 0 & 1\\ 0 & 1 & 0\end{bmatrix}=[0\quad 0\quad 0]$$

$$\boldsymbol{c}_2=[0\quad 1]\begin{bmatrix}1 & 0 & 1\\ 0 & 1 & 0\end{bmatrix}=[0\quad 1\quad 0]$$

$$\boldsymbol{c}_3=[1\quad 0]\begin{bmatrix}1 & 0 & 1\\ 0 & 1 & 0\end{bmatrix}=[1\quad 0\quad 1]$$

$$\boldsymbol{c}_4=[1\quad 1]\begin{bmatrix}1 & 0 & 1\\ 0 & 1 & 0\end{bmatrix}=[1\quad 1\quad 1]$$

因此，这个生成矩阵生成的码为 $C=\{000,010,101,111\}$。

3.2.2 校验矩阵

对于前述(7,3)线性分组码，为了更好地说明信息元与校验元的关系，将校验方程重新表述为

$$\begin{cases}c_3=u_2+u_0=c_6+c_4\\ c_2=u_2+u_1+u_0=c_6+c_5+c_4\\ c_1=u_2+u_1=c_6+c_5\\ c_0=u_1+u_0=c_5+c_4\end{cases}$$

变换为

$$\begin{cases}1\cdot c_6+0\cdot c_5+1\cdot c_4+1\cdot c_3+0\cdot c_2+0\cdot c_1+0\cdot c_0=0\\ 1\cdot c_6+1\cdot c_5+1\cdot c_4+0\cdot c_3+1\cdot c_2+0\cdot c_1+0\cdot c_0=0\\ 1\cdot c_6+1\cdot c_5+0\cdot c_4+0\cdot c_3+0\cdot c_2+1\cdot c_1+0\cdot c_0=0\\ 0\cdot c_6+1\cdot c_5+1\cdot c_4+0\cdot c_3+0\cdot c_2+0\cdot c_1+1\cdot c_0=0\end{cases}$$

再用矩阵表示这些线性方程

$$\begin{bmatrix}1&0&1&1&0&0&0\\1&1&1&0&1&0&0\\1&1&0&0&0&1&0\\0&1&1&0&0&0&1\end{bmatrix}\begin{bmatrix}c_6\\c_5\\c_4\\c_3\\c_2\\c_1\\c_0\end{bmatrix}=\begin{bmatrix}0\\0\\0\\0\end{bmatrix}=\mathbf{0}^{\mathrm{T}}$$

或

$$\begin{bmatrix}c_6&c_5&c_4&c_3&c_2&c_1&c_0\end{bmatrix}\begin{bmatrix}1&1&1&0\\0&1&1&1\\1&1&0&1\\1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}=\begin{bmatrix}0&0&0&0\end{bmatrix}=\mathbf{0}$$

将上面的 4 行 7 列系数矩阵用 $\boldsymbol{H}$ 表示

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

由此可见，若 $\boldsymbol{H}$ 已知，便可由信息元求出校验元，编码问题迎刃而解；或者说，要解决编码问题，只要找到 $\boldsymbol{H}$ 即可。由于(n,k)码的所有码字均按 $\boldsymbol{H}$ 所确定的规则求出，故称 $\boldsymbol{H}$ 为它的校验矩阵。一般而言，对于(n,k)线性码有 $r=n-k$ 个校验元，则必须有 r 个校验方程，如果 r 个校验方程线性独立，此时的校验矩阵称为一致校验矩阵。

(n,k)线性码的 $\boldsymbol{H}$ 矩阵由 r 行和 n 列组成，可表示为

$$\boldsymbol{H}=\begin{bmatrix}h_{11}&h_{12}&\cdots&h_{1n}\\h_{21}&h_{22}&\cdots&h_{2n}\\\vdots&\vdots&\ddots&\vdots\\h_{r1}&h_{r2}&\cdots&h_{rn}\end{bmatrix}$$

这里 h_{ij} 中，i 代表行号，j 代表列号。因此 $\boldsymbol{H}$ 是一个 r 行 n 列矩阵。由 $\boldsymbol{H}$ 矩阵可建立码的 r 个线性方程，由上可知：

$$\begin{bmatrix}h_{11}&h_{12}&\cdots&h_{1n}\\h_{21}&h_{22}&\cdots&h_{2n}\\\vdots&\vdots&\ddots&\vdots\\h_{r1}&h_{r2}&\cdots&h_{rn}\end{bmatrix}\begin{bmatrix}c_{n-1}\\c_{n-2}\\\vdots\\c_1\\c_0\end{bmatrix}=\mathbf{0}^{\mathrm{T}}$$

简写为

$$\boldsymbol{H}\boldsymbol{c}^{\mathrm{T}}=\boldsymbol{0}^{\mathrm{T}}$$

或

$$\boldsymbol{c}\boldsymbol{H}^{\mathrm{T}}=\boldsymbol{0} \tag{3-6}$$

这里 $\boldsymbol{c}=[c_{n-1},c_{n-2},\cdots,c_1,c_0]$，$\boldsymbol{c}^{\mathrm{T}}$ 是 $\boldsymbol{c}$ 的转置，$\boldsymbol{0}$ 是一个全为 0 的 r 重矩阵。

综上所述，将 $\boldsymbol{H}$ 矩阵的特点归纳如下：

(1) $\boldsymbol{H}$ 矩阵的每一行代表一个线性方程的系数，它表示求一个校验元的线性方程；

(2) $\boldsymbol{H}$ 矩阵的每一列代表此码元与哪几个校验方程有关；

(3) 由此 $\boldsymbol{H}$ 矩阵得到的(n,k)分组码的每一码字 $\boldsymbol{c}_i(i=1,2,\cdots,2^n)$都必须满足由 $\boldsymbol{H}$ 矩阵行所确定的线性方程；

(4) (n,k)码须有 $r=n-k$ 个校验元，故须有 r 个独立的线性方程。因此，$\boldsymbol{H}$ 矩阵必须有 r 行，且各行之间线性无关；

(5) 考虑到生成矩阵 $\boldsymbol{G}$ 中的每一行及其线性组合都是(n,k)码，故有 $\boldsymbol{H}\boldsymbol{G}^{\mathrm{T}}=\boldsymbol{0}^{\mathrm{T}}$ 或 $\boldsymbol{G}\boldsymbol{H}^{\mathrm{T}}=\boldsymbol{0}$，即生成矩阵 $\boldsymbol{G}$ 的行与校验矩阵 $\boldsymbol{H}$ 的行相互正交。由于 $\boldsymbol{G}$ 是 $k\times n$ 阶矩阵，故 $\boldsymbol{H}$ 是$(n-k)\times n$ 阶矩阵，$\boldsymbol{0}$ 是一个 $k\times(n-k)$阶的全零矩阵。

注：生成矩阵和校验矩阵是同一个编码方法的不同描述形式。

3.3 系统线性分组码

根据对信息元的处理方法不同，可以将纠错码分为分组码与卷积码。

分组码是把信源输出的信息序列，以 k 个码元划分为一段，通过编码器把这段 k 个信息元按一定规则产生 r 个校验(监督)元，输出码长为 $n=k+r$ 的一个码组。这种编码中每一码组的校验元仅与本组的信息元有关，而与别组无关。分组码用(n,k)表示，n 表示码长，k 表示信息位，如下：

k 个信息码元序列	r 个监督码元
r 个监督码元	k 个信息码元序列

通常称具有上图结构的分组码为系统线性分组码。由图知消息部分可以写在左半边也可以写在右半边，二者的纠错或检错能力是一样的。

系统线性分组码是分组码的一种，其构成应该和分组码一样，但由式 $\boldsymbol{c}=\boldsymbol{u}\boldsymbol{G}$ 得到的码字是否是上图的结构，将取决于生成矩阵 $\boldsymbol{G}$ 的结构。

在一般情况下，生成矩阵 $\boldsymbol{G}$ 是 $k\times n$ 矩阵，可通过初等变换将 $\boldsymbol{G}$ 变成如下形式：

$$\boldsymbol{G}=\begin{bmatrix}1&0&\cdots&0&p_{11}&p_{12}&\cdots&p_{1(n-k)}\\0&1&\cdots&0&p_{21}&p_{22}&\cdots&p_{2(n-k)}\\\vdots&\vdots&\ddots&\vdots&\vdots&\vdots&\ddots&\vdots\\0&0&\cdots&1&p_{k1}&p_{k2}&\cdots&p_{k(n-k)}\end{bmatrix}=[\boldsymbol{I}_k\ \vdots\ \boldsymbol{P}] \tag{3-7}$$

$\boldsymbol{I}_k$ 是 $k\times k$ 单位方阵，$\boldsymbol{P}$ 是 $k\times(n-k)$矩阵$(n>k)$，称这种形式的 $\boldsymbol{G}$ 为标准生成矩阵，因为初等变换不改变矩阵的秩，因此 $[\boldsymbol{I}_k\ \vdots\ \boldsymbol{P}]$仍由 k 个线性无关的行矢量组成。

这样得到的线性分组码的矩阵表示为

$$\boldsymbol{c}=\boldsymbol{u}\boldsymbol{G}=\boldsymbol{u}[\boldsymbol{I}_k \vdots \boldsymbol{P}]=(c_{n-1},c_{n-2},\cdots,c_1,c_0) \tag{3-8}$$

前面 k 位 $c_{n-1},c_{n-2},\cdots,c_{n-k}$ 是信息位，后面$(n-k)$位 $c_{n-k-1},c_{n-k-2},\cdots,c_1,c_0$ 是校验位，这样的编码即为线性系统分组码。

因为 $\boldsymbol{G}\boldsymbol{H}^{\mathrm{T}}=[\boldsymbol{I}_k \vdots \boldsymbol{P}]\begin{bmatrix}-\boldsymbol{P}^{\mathrm{T}}\\ \boldsymbol{I}_{n-k}\end{bmatrix}=\boldsymbol{0}$，这时校验矩阵相应地变成

$$\boldsymbol{H}=[-\boldsymbol{P}^{\mathrm{T}} \vdots \boldsymbol{I}_{n-k}] \tag{3-9}$$

式中，$\boldsymbol{I}_{n-k}$ 是$(n-k)\times(n-k)$单位方阵，$-\boldsymbol{P}^{\mathrm{T}}$ 是矩阵 $\boldsymbol{P}$ 的逆元转置矩阵，对于模 2 加运算，0 的逆元为 0，1 的逆元为 1，故有$-\boldsymbol{P}^{\mathrm{T}}=\boldsymbol{P}^{\mathrm{T}}$，因此

$$\boldsymbol{H}=[\boldsymbol{P}^{\mathrm{T}} \vdots \boldsymbol{I}_{n-k}] \tag{3-10}$$

常用的系统码有两种形式：信息组被排在码字的最左边 k 位，或信息组被排在码字的最右边 k 位。

一般来说，系统码的编译码相对非系统码要简单一些，但两者的纠错能力完全等价，因此一般总希望线性分组码采用系统码形式。

例 3-4　二元(6,3)线性分组码，下面给出的 $\boldsymbol{G}_1$ 和 $\boldsymbol{G}_2$ 都可以作为它的生成矩阵，即

$$\boldsymbol{G}_1=\begin{bmatrix}1&0&1&0&1&1\\1&1&0&1&0&1\\1&1&1&0&0&0\end{bmatrix},\quad \boldsymbol{G}_2=\begin{bmatrix}1&0&0&1&1&0\\0&1&0&0&1&1\\0&0&1&1&0&1\end{bmatrix}$$

表 3-3 给出了分别由 $\boldsymbol{G}_1$ 和 $\boldsymbol{G}_2$ 所生成的线性码。

表 3-3　由不同生成矩阵生成的线性分组码

信息组	由 $\boldsymbol{G}_1$ 生成的(6,3)码	由 $\boldsymbol{G}_2$ 生成的(6,3)码
000	000000	000000
001	111000	001101
010	110101	010011
011	001101	011110
100	101011	100110
101	010011	101011
110	011110	110101
111	100110	111000

从表 3-3 可以看出，由 $\boldsymbol{G}_1$ 和 $\boldsymbol{G}_2$ 所生成的线性码对应同一个三维子空间 V_6^3，也就是从 2^6 个码组中选出 2^3 个码组，而且这 2^3 个码组相同，只不过信息组与码组之间有着不同的映射关系。

观察由 $\boldsymbol{G}_2$ 所生成的线性码，由

$$\begin{aligned}\boldsymbol{c}&=\boldsymbol{u}\boldsymbol{G}_2\\&=[u_2\quad u_1\quad u_0]\begin{bmatrix}1&0&0&1&1&0\\0&1&0&0&1&1\\0&0&1&1&0&1\end{bmatrix}\\&=[u_2\quad u_1\quad u_0\quad u_2+u_0\quad u_2+u_1\quad u_1+u_0]\\&=[c_5\quad c_4\quad c_3\quad c_2\quad c_1\quad c_0]\end{aligned}$$

得

$$\begin{cases} c_5 = u_2 \\ c_4 = u_1 \\ c_3 = u_0 \\ c_2 = u_2 + u_0 \\ c_1 = u_2 + u_1 \\ c_0 = u_1 + u_0 \end{cases}$$

码组的前3位是信息位，后3位是校验位，因此这样的码是系统码。

例 3-5 一个(7,4)线性分组码的生成矩阵为

$$\boldsymbol{G}=\begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$，可知矩阵 $\boldsymbol{P}=\begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{bmatrix}$，它的转置 $\boldsymbol{P}^{\mathrm{T}}=\begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{bmatrix}$。

可以得到校验矩阵为

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

例 3-6 对于例3-2给出的(7,4)线性分组码，生成矩阵

$$\boldsymbol{G}=\begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix} \xrightarrow{\text{初等变换}} \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}=[\boldsymbol{I}_k \vdots \boldsymbol{P}]$$

则校验矩阵为

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

生成的码字如表3-4所示。表中的码字前4位是信息位，后3位是校验位。

表 3-4 (7,4)系统码

信息组	码字	信息组	码字
0000	0000000	1000	1000111
0001	0001011	1001	1001100
0010	0010101	1010	1010010
0011	0011110	1011	1011001
0100	0100110	1100	1100001
0101	0101101	1101	1101010
0110	0110011	1110	1110100
0111	0111000	1111	1111111

3.4 对偶码

设原码有 k 位信息位，其生成矩阵为 $\boldsymbol{G}$，校验矩阵为 $\boldsymbol{H}$，对应的线性码为 $\boldsymbol{C}$。

若用 $\boldsymbol{H}$ 作为生成矩阵，生成另一码 $\boldsymbol{C}^{\perp}$，则对应的校验矩阵为 $\boldsymbol{G}$，称 $\boldsymbol{C}^{\perp}$ 为 $\boldsymbol{C}$ 的对偶码，$\boldsymbol{C}^{\perp}$ 有 $(n-k)$ 位信息位，k 位校验位。

因为 $\boldsymbol{C}^{\perp}=\boldsymbol{uH}$，$\boldsymbol{C}=\boldsymbol{uG}$，$\boldsymbol{C}(\boldsymbol{C}^{\perp})^{\mathrm{T}}=\boldsymbol{uG}(\boldsymbol{uH})^{\mathrm{T}}=\boldsymbol{u}(\boldsymbol{GH}^{\mathrm{T}})\boldsymbol{u}^{\mathrm{T}}=0$，说明互为对偶码的码矢内积为 $\boldsymbol{0}$，即两码矢正交。

例 3-7 给定生成矩阵 $\boldsymbol{G}_{(7,3)}=\begin{bmatrix}1&0&0&1&1&1&0\\0&1&0&0&1&1&1\\0&0&1&1&1&0&1\end{bmatrix}$，求由 $\boldsymbol{G}_{(7,3)}$ 生成的原码 $\boldsymbol{C}$ 和它的对偶码 $\boldsymbol{C}^{\perp}$。

设 $u=(u_2,u_1,u_0)$，由 $\boldsymbol{G}_{(7,3)}$ 生成(7,3)线性分组码：

$$\boldsymbol{C}=\boldsymbol{uG}_{(7,3)}=\begin{bmatrix}u_2&u_1&u_0\end{bmatrix}\cdot\begin{bmatrix}1&0&0&1&1&1&0\\0&1&0&0&1&1&1\\0&0&1&1&1&0&1\end{bmatrix}$$

$$=\begin{bmatrix}u_2&u_1&u_0&u_2\oplus u_0&u_2\oplus u_1\oplus u_0&u_2\oplus u_1&u_1\oplus u_0\end{bmatrix}$$

具体码见表 3-5。

表 3-5 由 $\boldsymbol{G}_{(7,3)}$ 生成(7,3)线性分组码

信息位 u_2,u_1,u_0	码矢 $\boldsymbol{C}$	信息位 u_2,u_1,u_0	码矢 $\boldsymbol{C}$
000	0000000	100	1001110
001	0011101	101	1010011
010	0100111	110	1101001
011	0111010	111	1110100

而 $\boldsymbol{G}_{(7,3)}$ 就是其对偶码 $\boldsymbol{C}^{\perp}_{(7,4)}$ 的校验矩阵 $\boldsymbol{H}^{\perp}_{(7,4)}$，先将 $\boldsymbol{G}_{(7,3)}$ 通过初等变换化为校验矩阵的标准形式：

$$\boldsymbol{G}_{(7,3)}\xrightarrow{\text{初等变换}}\begin{bmatrix}1&1&1&0&1&0&0\\0&1&1&1&0&1&0\\1&1&0&1&0&0&1\end{bmatrix}=[\boldsymbol{P}^{\mathrm{T}}\vdots\boldsymbol{I}_3]=\boldsymbol{H}^{\perp}_{(7,4)}$$

根据 $\boldsymbol{H}^{\perp}_{(7,4)}$ 可得对偶码的生成矩阵 $\boldsymbol{G}^{\perp}_{(7,4)}$：

$$\boldsymbol{G}^{\perp}_{(7,4)}=[\boldsymbol{I}_4\vdots\boldsymbol{P}^{\mathrm{T}}]=\begin{bmatrix}1&0&0&0&1&0&1\\0&1&0&0&1&1&1\\0&0&1&0&1&1&0\\0&0&0&1&0&1&1\end{bmatrix}$$

设 $\boldsymbol{u}=(u_3,u_2,u_1,u_0)$，由 $\boldsymbol{G}^{\perp}_{(7,4)}$ 生成对偶码：

$$\boldsymbol{C}^{\perp}=\boldsymbol{u}\cdot\boldsymbol{G}^{\perp}_{(7,4)}=\begin{bmatrix}u_3&u_2&u_1&u_0\end{bmatrix}\cdot\begin{bmatrix}1&0&0&0&1&0&1\\0&1&0&0&1&1&1\\0&0&1&0&1&1&0\\0&0&0&1&0&1&1\end{bmatrix}$$

$$=[u_3 \quad u_2 \quad u_1 \quad u_0 \quad u_3 \oplus u_2 \oplus u_1 \quad u_2 \oplus u_1 \oplus u_0 \quad u_3 \oplus u_2 \oplus u_0]$$

具体码见表 3-6。

表 3-6 由 $G_{(7,4)}^{\perp}$ 生成(7,4)线性分组码

信息位 u_3,u_2,u_1,u_0	码矢 $\boldsymbol{C}$	信息位 u_3,u_2,u_1,u_0	码矢 $\boldsymbol{C}$
0000	0000000	1000	1000101
0001	0001011	1001	1001110
0010	0010110	1010	1010011
0011	0011101	1011	1011000
0100	0100111	1100	1100010
0101	0101100	1101	1101001
0110	0110001	1110	1110100
0111	0111010	1111	1111111

可以验证，表 3-5 中的任何一个码字与表 3-6 中的任何一个码字的内积为 **0**。

若一个码的对偶码就是它自己，则称该码为自对偶码。自对偶码必是$(2m,m)$形式的分组码。例如(2,1)重复码就是一个自对偶码。

3.5 编码的实现

生成矩阵 $\boldsymbol{G}$ 的行是线性独立的，因此，行的线性组合可以用于生成 $\boldsymbol{c}$ 中的码字。生成矩阵将是秩为 k 的 $k\times n$ 阶矩阵，它完整地描述了编码的过程，有了生成矩阵 $\boldsymbol{G}$，编码器的结构就很容易确定，式 $\boldsymbol{c}=\boldsymbol{uG}$ 事实上给出了编码的实现方法。

当已知(n,k)线性分组码的生成矩阵 $\boldsymbol{G}$ 或校验矩阵 $\boldsymbol{H}$ 时，编码问题是容易实现的。

设(n,k)系统码的生成矩阵为

$$\boldsymbol{G}=\begin{bmatrix} 1 & 0 & \cdots & 0 & p_{1,n-k-1} & p_{1,n-k-2} & \cdots & p_{1,0} \\ 0 & 1 & \cdots & 0 & p_{2,n-k-1} & p_{2,n-k-2} & \cdots & p_{2,0} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & p_{k,n-k-1} & p_{k,n-k-2} & \cdots & p_{k0} \end{bmatrix}=[\boldsymbol{I}_k \vdots \boldsymbol{P}]$$

若 $\boldsymbol{u}=(u_{k-1},u_{k-2},\cdots,u_0)$，由于系统码的前 k 位是信息位，则信息位可表示为 $\boldsymbol{u}=(u_{n-1},u_{n-2},\cdots,u_{n-k})$，相应的码字是

$$\boldsymbol{c}=(c_{n-1},c_{n-2},\cdots,c_1,c_0)=\boldsymbol{uG}$$

于是有，

$$\begin{cases} c_j=u_j & (n-k\leqslant j\leqslant n-1) \\ c_j=u_{n-1}p_{1,j}+u_{n-2}p_{2,j}+\cdots+u_{n-k}p_{k,j} & (0\leqslant j\leqslant n-k) \end{cases} \tag{3-11}$$

编码实现方法，如果采用硬件实现，电路如图 3-1 所示，电路由移位寄存器、模 2 乘法器和模 2 加法器等器件组成。在图中，“→□→”表示移位寄存器单元，“→⊕→”表示模 2 加法器，“→○→”及近旁的 $\boldsymbol{P}$ 矩阵元素 p_{ij} 表示模 2 乘法器，对于二元域，$p_{ij}=1$ 表示该处连通，$p_{ij}=0$ 表示该处断开。

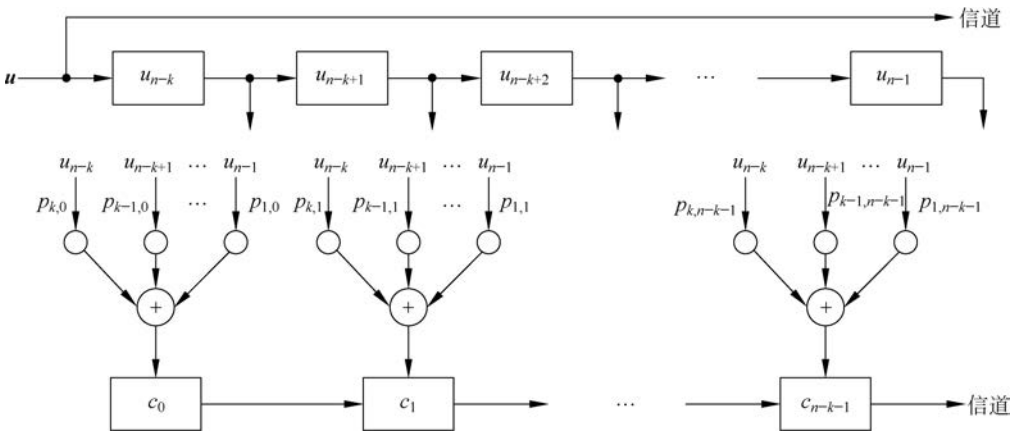


图 3-1　(n,k)线性分组码编码电路

对于例 3-7 给出的(7,3)线性分组码,则有,

$$\boldsymbol{c}=\boldsymbol{u}\boldsymbol{G}_{(7,3)}=\begin{bmatrix}u_6 & u_5 & u_4\end{bmatrix}\cdot\begin{bmatrix}1&0&0&1&1&1&0\\0&1&0&0&1&1&1\\0&0&1&1&1&0&1\end{bmatrix}$$

$$=\begin{bmatrix}u_6 & u_5 & u_4 & u_6\oplus u_4 & u_6\oplus u_5\oplus u_4 & u_6\oplus u_5 & u_5\oplus u_4\end{bmatrix}$$

根据图 3-1 的电路,可画出例 3-7 给出的(7,3)线性分组码的编码器电路,如图 3-2 所示。

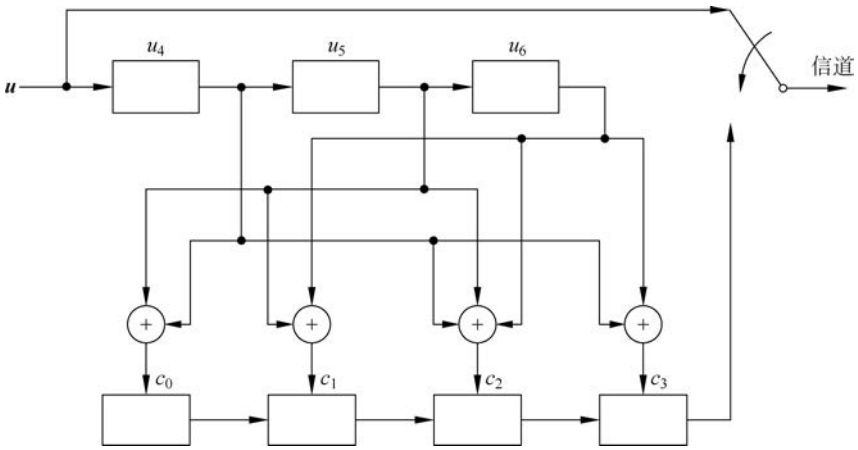


图 3-2　(7, 3)线性分组码编码电路

在图 3-2 中,3 位信息 $\boldsymbol{u}$ 在依次进入移位寄存器的同时也依次进入信道,当 3 位信息位输入结束后,停止信息输入,同时右边的开关切换到下方,此时图下方的 4 个模 2 加法器也已得到了 c_3,c_2,c_1 和 c_0,将这 4 个值存入移位寄存器,然后再依次送入信道,就得到编码 $\boldsymbol{c}$。

3.6 线性分组码的译码

3.6.1 信息传输系统模型

为了方便研究，将信息传输系统模型简化成图 3-3 所示的简化模型。

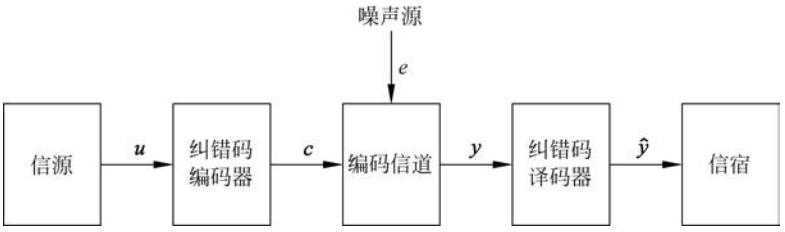


图 3-3 简化的信息传输系统模型

这里信源是指原来的信源和信源编码器，它的输出是二进制信息序列 $\boldsymbol{u}$，纠错编码器按一定的编码规则将 $\boldsymbol{u}$ 编成码字 $\boldsymbol{c}$ 送入信道。信道是包括调制器、传输媒质和解调器在内的数字信道（也称编码信道），它的输入是码字 $\boldsymbol{c}$，输出是接收码字 $\boldsymbol{y}$，纠错译码器根据编码规则对接收码字进行译码，输出估值 $\hat{\boldsymbol{y}}$，信宿包括信源译码器和用户，它的输入是经过纠错的估值序列 $\hat{\boldsymbol{y}}$。该模型突出了以控制差错为目的的纠错码编码器和译码器，因此也称为差错控制系统。

在译码过程中有两个重要的概念是错误图样和伴随式。

1. 错误图样

定义：$\boldsymbol{e}=\boldsymbol{y}-\boldsymbol{c}$，称 $\boldsymbol{e}$ 为错误图样，在二进制中模 2 加和模 2 减是相同的，因此有 $\boldsymbol{e}=\boldsymbol{y}+\boldsymbol{c}$ 和 $\boldsymbol{y}=\boldsymbol{e}+\boldsymbol{c}$。

注：从发送信息的角度看，错图图样即是噪声（或干扰）图样，因此有 $\boldsymbol{y}=\boldsymbol{c}+\boldsymbol{e}$；从接收信息的角度看，错图图样即是差错图样 $\boldsymbol{e}=\boldsymbol{y}-\boldsymbol{c}=\boldsymbol{y}+\boldsymbol{c}$。

2. 伴随式

定义：$\boldsymbol{s}=\boldsymbol{H}\boldsymbol{y}^{\mathrm{T}}$（或 $\boldsymbol{s}=\boldsymbol{y}\boldsymbol{H}^{\mathrm{T}}$），称 $\boldsymbol{s}$ 为 $\boldsymbol{y}$ 的伴随式。

$\boldsymbol{s}=\boldsymbol{H}\boldsymbol{y}^{\mathrm{T}}=\boldsymbol{H}(\boldsymbol{e}+\boldsymbol{c})^{\mathrm{T}}=\boldsymbol{H}\boldsymbol{e}^{\mathrm{T}}+\boldsymbol{H}\boldsymbol{c}^{\mathrm{T}}=\boldsymbol{H}\boldsymbol{e}^{\mathrm{T}}$

若 $\boldsymbol{s}=\boldsymbol{0}$，根据 $\boldsymbol{H}\boldsymbol{c}^{\mathrm{T}}=0$，则 $\boldsymbol{y}$ 是选用码字，认为传输无误；若 $\boldsymbol{s}\neq\boldsymbol{0}$，则 $\boldsymbol{y}$ 不是选用码字，说明在传输过程中产生了误码。

从物理意义上看，伴随式 $\boldsymbol{s}$ 并不反映发送的码字是什么，而只是反映信道对码字造成了怎样的影响。

错误图样 $\boldsymbol{e}$ 是 n 重矢量，共有 2^n 个可能的组合，而伴随式 $\boldsymbol{s}$ 是 $(n-k)$ 重矢量，只有 2^{n-k} 个可能的组合，因此不同的错误图样可能有相同的伴随式。

3.6.2 标准阵列

1. 标准阵列定义

线性分组码的 n 位码符号由 k 位信息位加上($n-k$)位校验位组成,这 n 位码符号取自符号集$\{a_1,a_2,\cdots,a_q\}$,在整个 n 维空间 $\boldsymbol{V}_n$ 中共有 q^n 个矢量(一般 $q=2$)。

线性分组码对应 k 维子空间 $\boldsymbol{V}_n^k$,在 k 维子空间中,共有 2^k 个矢量,这 2^k 个矢量就是选用码矢或许用码矢,其余(2^n-2^k)个矢量称为禁用矢量。

下面将 2^n 个矢量排成标准阵列:

第 1 步:从全零码矢开始,把所有选用码矢$\{c_1,c_2,\cdots,c_{q^k}\}$依次写成一行。

第 2 步:选一个在第 1 行中没列出的且重量最轻的禁用矢量 $\boldsymbol{y}_1$ 写在第 2 行的第 1 列,其余各列都用 $\boldsymbol{y}_1$ 和第一行对应码矢相加(模 2 加)。

第 3 步:再选一个第 1 行、第 2 行没列出的且重量最轻的禁用矢量 $\boldsymbol{y}_2$,其余各列都用 $\boldsymbol{y}_2$ 和第一行对应码矢相加(模 2 加)。

第 4 步:如此重复,直至把所有 2^n 个矢量列完。

把这样列出的表格称为标准阵列。取 $q=2$,则 2^n 个矢量列出的标准阵列如表 3-7 所示。

表 3-7 线性分组码的标准阵列

许用码字	$c_1(e_0)$(陪集首)	c_2	c_3	…	c_{2^k}
禁用码字	$y_1(e_1)$	y_1+c_2	y_1+c_3	…	$y_1+c_{2^k}$
	$y_2(e_2)$	y_2+c_2	y_2+c_3	…	$y_2+c_{2^k}$
	⋮	⋮	⋮	⋮	⋮
	$y_{2^{n-k}-1}(e_{2^{n-k}-1})$	$y_{2^{n-k}-1}+c_2$	$y_{2^{n-k}-1}+c_3$	…	$y_{2^{n-k}-1}+c_{2^k}$

例 3-8 二元(5,3)码,生成矩阵 $\boldsymbol{G}=\begin{bmatrix}1&0&0&1&1\\0&1&0&1&0\\0&0&1&1&1\end{bmatrix}$,信源有 8 个消息待发,对应信源编码器的 8 个输出序列,即{000,001,010,011,100,101,110,111}。根据 $\boldsymbol{c}=\boldsymbol{uG}$ 编码,得到 8 个码矢,即

{00000,00111,01010,01101,10011,10100,11001,11110}

按上述方式将 $2^5=32$ 个 5 重矢量排成标准阵列,如表 3-8 所示。第一行为 8 个码矢,其余各行为本行的第一个元素与第一行元素的模 2 加。

表 3-8 将 32 个 5 重矢量排成标准阵列

00000	00111	01010	01101	10011	10100	11001	11110
00001	00110	01011	01100	10010	10101	11000	11111
00010	00101	01000	01111	10001	10110	11011	11100
00100	00011	01110	01001	10111	10000	11101	11010

码字 $\boldsymbol{c}=(c_4,c_3,c_2,c_1,c_0)$，通过有噪信道传输，信道输出 $\boldsymbol{y}=(y_4,y_3,y_2,y_1,y_0)$，由于信道干扰，$\boldsymbol{y}$ 序列中的某些码元可能与 $\boldsymbol{c}$ 序列中对应码元的值不同，即传输产生了错误。在二进制序列中，错误无非是 1 错成 0 或 0 错成 1，因此，信道中的干扰可用二进制序列 $\boldsymbol{e}=(e_4,e_3,e_2,e_1,e_0)$表示，有错的各位 e_i 取值为 1，无错的各位 e_i 取值为 0，则有 $\boldsymbol{y}=\boldsymbol{c}+\boldsymbol{e}$，$\boldsymbol{e}$ 即为信道的错误图样。

显然，当 $\boldsymbol{e}=\boldsymbol{0}$ 时，$\boldsymbol{y}=\boldsymbol{c}$，表示接收序列 $\boldsymbol{y}$ 无错；当 $\boldsymbol{e}\neq\boldsymbol{0}$ 时，$\boldsymbol{y}\neq\boldsymbol{c}$，表示接收序列 $\boldsymbol{y}$ 有错。当 $\boldsymbol{c}$ 序列长为 n 时，信道可能产生的错误图样 $\boldsymbol{e}$ 的数目共有 2^n 种。在此例中 $n=5$，信道可能产生的错误图样 $\boldsymbol{e}$ 的数目共有 $2^5=32$ 种，即表 3-8 列出的 32 个 5 重矢量都是可能的错误图样。从另一个角度来说，表 3-8 列出的 32 个 5 重矢量也都是可能的信道输出矢量 $\boldsymbol{y}$。

译码器的任务就是要从收到的 $\boldsymbol{y}$ 中得到 $\hat{\boldsymbol{c}}$，或者说由 $\boldsymbol{y}$ 中解出错误图样 $\boldsymbol{e}$，然后得到 $\hat{\boldsymbol{c}}=\boldsymbol{y}+\boldsymbol{e}$。这里 $\hat{\boldsymbol{c}}$ 是对码字 $\boldsymbol{c}$ 的估值，若 $\hat{\boldsymbol{c}}=\boldsymbol{c}$，则译码正确，否则译码错误。

2. 陪集分解

由标准阵列的构成法，有以下结论：

(1) 第 1 行是 2^k 个选用码矢，以后每行都是第 1 行的陪集，每行的第一个元素称为陪集首，分别记为 $e_0,e_1,e_2,\cdots,e_{2^{n-k}-1}$，如表 3-7 所示；一个陪集中具有最小重量的向量作为陪集首，如果有多于一个向量具有最小重量，则从中随机选择一个作为陪集首。

(2) 陪集首是前面列出的元素中没有出现过的，从而该陪集中的元素也是前面没有出现过的。

(3) 如果选的陪集首是该行中的另一个元素，则该行中的元素还是原来的 2^k 个元素，只不过排列顺序变了，这说明该行中的每个元素都可以作为陪集首。

(4) 根据(1)和(2)，最后把所有 2^n 个元素全列完(2^{n-k} 行$\times 2^k$ 列$=2^n$ 个元素)。

(5) 同一陪集中所有元素的伴随式相同，不同陪集的元素的伴随式不同。

上述(1)、(2)、(3)、(4)条是不言而喻的，下面证明第(5)条。

证明：取第 i 个陪集中的第 j 个元素 $\boldsymbol{e}_i+\boldsymbol{c}_j$，根据伴随式的定义，它的伴随式 $\boldsymbol{s}=\boldsymbol{H}(\boldsymbol{e}_i+\boldsymbol{c}_j)^{\mathrm{T}}=\boldsymbol{H}\boldsymbol{e}_i^{\mathrm{T}}$，可见，$\boldsymbol{s}$ 仅与第 i 个陪集首有关，而与 j 的取值无关，故同一陪集的元素的伴随式相同。

下面证明不同陪集的元素的伴随式不同，用反证法。

反证法 设第 i 个陪集与第 k 个陪集的伴随式相同，即 $\boldsymbol{H}\boldsymbol{e}_i^{\mathrm{T}}=\boldsymbol{H}\boldsymbol{e}_k^{\mathrm{T}}$，则 $\boldsymbol{H}(\boldsymbol{e}_i+\boldsymbol{e}_k)^{\mathrm{T}}=\boldsymbol{0}$。该式说明 $\boldsymbol{e}_i+\boldsymbol{e}_k$ 是一个码矢，即 $\boldsymbol{e}_i+\boldsymbol{e}_k=\boldsymbol{c}\in C$，则 $\boldsymbol{e}_i=\boldsymbol{e}_k+\boldsymbol{c}$。

该式表明第 k 个陪集中有一个元素 $\boldsymbol{e}_k+\boldsymbol{c}$ 与第 i 个陪集首相同，根据标准阵列的排法，这是不可能的，故不同陪集的元素的伴随式不同。

例 3-9 设 C 为一个二元(3,2)码，其生成矩阵如下：

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

即 $C=\{000,010,101,111\}$。C 的陪集为

$$000+C=\{000,010,101,111\}$$
$$001+C=\{001,011,100,110\}$$

注意所有 8 个向量都被这两个陪集覆盖了。随便列出一种情况，则

$$100+C=\{100,110,001,011\}$$

可以看到这 4 个向量已在 8 个向量之中了。

例 3-10 考虑某(4,2)码 $C=\{0000,1011,0101,1110\}$。相应的标准阵列为

码字→0000,1011,0101,1110
1000,0011,1101,0110
0100,1111,0001,1010
0010,1001,0111,1100
↑
陪集首

例 3-11 在例 3-8 的表 3-8 所列的标准阵列中，每行的第一个元素为该行的陪集首，有 $\boldsymbol{e}_0=(00000)$，$\boldsymbol{e}_1=(00001)$，$\boldsymbol{e}_2=(00010)$，$\boldsymbol{e}_3=(00100)$。

由矩阵 $\boldsymbol{G}$ 可得 $\boldsymbol{H}=\begin{bmatrix}1&1&1&1&0\\1&0&1&0&1\end{bmatrix}$，根据伴随式的定义，$\boldsymbol{s}=\boldsymbol{H}\boldsymbol{e}_i^{\mathrm{T}}$，计算出 4 个伴随式，即 $\boldsymbol{s}_0=(00)$，$\boldsymbol{s}_1=(01)$，$\boldsymbol{s}_2=(10)$，$\boldsymbol{s}_3=(11)$。

从 $\boldsymbol{s}=\boldsymbol{H}\boldsymbol{y}^{\mathrm{T}}=\boldsymbol{H}(\boldsymbol{e}_i+\boldsymbol{c}_j)^{\mathrm{T}}=\boldsymbol{H}\boldsymbol{e}_i^{\mathrm{T}}$ 来看，若 $\boldsymbol{e}=\boldsymbol{0}$，则 $\boldsymbol{s}=\boldsymbol{0}$；若 $\boldsymbol{e}\neq\boldsymbol{0}$，则 $\boldsymbol{s}\neq\boldsymbol{0}$，伴随式 $\boldsymbol{s}$ 只由错误图样 $\boldsymbol{e}$ 决定，即伴随式 $\boldsymbol{s}$ 是否全为零矢量可以作为判断一个码字传送是否出错的依据。$\boldsymbol{s}\neq\boldsymbol{0}$ 时，译码器要做的就是如何从伴随式 $\boldsymbol{s}$ 中找到错误图样 $\boldsymbol{e}$，从而译出发送的码字 $\hat{\boldsymbol{c}}=\boldsymbol{y}+\boldsymbol{e}$。

3.6.3 译码及纠错能力

1. 用标准阵列译码

由标准阵列的构成可知，第一行为码矢，从第二行开始，每一列与本列的第一行元素都只相差一个陪集首。在标准阵列中出现的 2^n 个矢量都是可能的错误图案，而陪集首选的是本行中重量最轻的矢量，所以 $\hat{\boldsymbol{y}}=\boldsymbol{y}+\boldsymbol{e}$ 就是最小距离译码，即最小错误概率译码。

接收到 $\boldsymbol{y}$ 后，到标准阵列中去找(因为 2^n 个矢量全部列在其中，总可以找到)，如果接收到的码字是个合法码字，那么可以下结论说没有错误发生(这个结论可能是错的，因为噪声也可能把一个合法码字改变成另一个合法码字，但这种错误发生概率很低)。如果接收到的码字是一个禁用码字，则推测发生了错误。译码器则声明陪集首就是错误图样 $\boldsymbol{e}$，然后译码为 $\boldsymbol{y}+\boldsymbol{e}$，这就是在 $\boldsymbol{y}$ 同一列中第一行的那个码字。因此，标准阵列译码就是把接收到的码字译为包含该码字的列的第一行的那个码字。

例 3-12 考虑码 $C=\{0000,1011,0101,1110\}$，若接收到的码字为 $\boldsymbol{y}=(1101)$，因为它不是一个合法的码字，则推测发生了错误。因此要估计 4 个可能的码字中哪一个是实际被传送的。标准阵列为

码字→0000,1011,0101,1110
1000,0011,1101,0110
0100,1111,0001,1010
0010,1001,0111,1100

可以发现 1101 在第三列，该列第一行的码字为 0101，于是估计码字为 0101。进一步可观察到 $d(1101,0000)=3$，$d(1101,1011)=2$，$d(1101,0101)=1$，$d(1101,1110)=2$。

错误图样 $\boldsymbol{e}=(1000)$，即陪集首。说明满足最大似然译码方法。

例 3-13 仍考虑例 3-8 所述的二元(5,3)码，假设接收矢量为 $\boldsymbol{y}=(10110)$，在表 3-8 列出的标准阵列中，找到 $\boldsymbol{y}$ 位于第 6 列，相应地译成 $\hat{\boldsymbol{y}}=(10100)$。错误图样是 $\boldsymbol{y}=(10110)$所在行的陪集首 $\boldsymbol{e}_2=(00010)$。

具有较大分组长度的码是可取的，因为大的码在码率方面更接近香农限。但当取越来越大的码长时(大的 k 值和 n 值)，采用标准阵列方法越来越不实际，因为大小为 $2^{n-k}\times 2^k$ 的标准阵列将大到不可操作。编码理论的基本目标之一就是要设计有效的译码方法。那么，能不能缩减标准阵列呢？

2. 伴随式与错误位数

伴随式译码与标准阵列的译码相似，伴随式译码的一般步骤如下：

(1) 计算接收矢量的伴随式。

(2) 由伴随式找到对应于它的陪集首(即寻找与该伴随式相同的陪集首)。

(3) 纠正错误 $\hat{\boldsymbol{y}}=\boldsymbol{y}+\boldsymbol{e}$，$\hat{\boldsymbol{y}}$ 即为译码输出的码字。

设(n,k)码的一致校验矩阵为

$$\boldsymbol{H}=\begin{bmatrix} h_{0,n-1} & h_{0,n-2} & \cdots & h_{0,0} \\ h_{1,n-1} & h_{1,n-2} & \cdots & h_{1,0} \\ \vdots & \vdots & \ddots & \vdots \\ h_{n-k-1,n-1} & h_{n-k-1,n-2} & \cdots & h_{n-k-1,0} \end{bmatrix}=[\boldsymbol{h}_{n-1} \quad \boldsymbol{h}_{n-2} \quad \cdots \quad \boldsymbol{h}_0] \tag{3-12}$$

式中，$\boldsymbol{h}_{n-j}(j=0,1,\cdots,n)$是 $\boldsymbol{H}$ 矩阵的第 j 列，它是一个 $n-k$ 重列矢量。

设码字传送发生 t 位错误，为不失一般性，设码字的第 $j_1,j_2,\cdots,j_t$ 位有错，则错误图样可表示成

$$\boldsymbol{e}=(0\cdots e_{j_1}\cdots 0\cdots e_{j_2}\cdots e_{j_t}\cdots 0\cdots 0) \tag{3-13}$$

在二进制情况下，$e_{j_1},e_{j_2},\cdots,e_{j_t}$ 为 1，那么伴随式

$$\begin{aligned} \boldsymbol{s}&=\boldsymbol{e}\boldsymbol{H}^{\mathrm{T}} \\ &=(0\cdots e_{j_1}\cdots 0\cdots e_{j_2}\cdots e_{j_t}\cdots 0\cdots 0)\cdot\begin{bmatrix}\boldsymbol{h}_{n-1}\\ \boldsymbol{h}_{n-2}\\ \vdots\\ \boldsymbol{h}_0\end{bmatrix} \\ &=e_{j_1}\cdot\boldsymbol{h}_{n-j_1}+e_{j_2}\cdot\boldsymbol{h}_{n-j_2}+\cdots+e_{j_t}\cdot\boldsymbol{h}_{n-j_t} \\ &=\boldsymbol{h}_{n-j_1}+\boldsymbol{h}_{n-j_2}+\cdots+\boldsymbol{h}_{n-j_t} \end{aligned} \tag{3-14}$$

上式说明，$\boldsymbol{s}$ 是 $\boldsymbol{H}$ 矩阵中对应于 $e_{j_1},e_{j_2},\cdots,e_{j_t}$ 为 1 的那几列 $\boldsymbol{h}_{n-j}$ 的线性组合，因 $\boldsymbol{h}_{n-j}$ 是 $n-k$ 重列矢量，则 $\boldsymbol{s}$ 也是一个 $n-k$ 重列矢量。

值得注意的是，若 $\boldsymbol{e}$ 本身就是一个码字，计算得 $\boldsymbol{s}$ 等于 $\boldsymbol{0}$。此时的错误不能被发现，也无法纠正，称之为不可检错误图样。

3. 译码表译码

在标准阵列中，每个陪集全部 2^k 个矢量都有相同的伴随式，而不同陪集的矢量有不同的伴随式。这就表明陪集首和伴随式有一一对应的关系，而这些陪集首实际上也就代表着可纠正的错误图样。根据这一关系可把上述标准阵列表进行简化，得到一个简化译码表。

将标准阵列译码和伴随式译码结合起来简化成更为实用的译码表，译码表保留了标准阵列中的 2^{n-k} 个可纠正错误图样 $\boldsymbol{e}_j$（陪集首）与其伴随式 $\boldsymbol{s}=\boldsymbol{H}\boldsymbol{e}_j^{\mathrm{T}}$ 之间的一一对应关系，译码器存储该表后，在译码时就可以查表实现从伴随式到错误图样的转换。表 3-9 就是表 3-8 所列(5,3)线性分组码标准阵列的译码表。

表 3-9 (5,3)线性分组码标准阵列的译码表

伴随式 $\boldsymbol{s}$	错误图样(陪集首)$\boldsymbol{e}$	伴随式 $\boldsymbol{s}$	错误图样(陪集首)$\boldsymbol{e}$
00	00000	10	00010
01	00001	11	00100

例 3-14 仍考虑例 3-8 所述的二元(5,3)码，假设接收矢量为 $\boldsymbol{y}=(10110)$，计算伴随式

$$\boldsymbol{s}=\boldsymbol{H}\boldsymbol{y}^{\mathrm{T}}=\begin{bmatrix}1&1&1&1&0\\1&0&1&0&1\end{bmatrix}\begin{bmatrix}1\\0\\1\\1\\0\end{bmatrix}=\begin{bmatrix}1\\0\end{bmatrix}$$

从表 3-9 中找到对应陪集首 $\boldsymbol{e}_2$ 为(00010)，则

$$\hat{\boldsymbol{y}}=\boldsymbol{y}+\boldsymbol{e}=10110\oplus 00010=10100$$

从表 3-8 可知 $\hat{\boldsymbol{y}}$ 就是 $\boldsymbol{y}$ 所在列的第一个矢量。

用译码表译码，译码正确的概率与陪集首的选择有关。根据最大后验概率译码准则，重量最轻的错误图样产生的可能性最大，所以应该优先选择重量小的 n 重矢量作为陪集首。这样构造的译码表，使得 $\boldsymbol{e}_j+\boldsymbol{c}_i$ 与 $\boldsymbol{c}_i$ 之间的距离最小，从而使译码器能以更大的正确概率译码，这就是最小距离译码。

4. 纠错能力分析

线性分组码的纠错能力 t 和码字的最小 d_0 有关，一般 t 是由通信系统提出的，那么寻找满足纠正 t 个错误码元的码字就是编码技术的任务，为此还需进一步研究 d_0 和码字结构的关系。线性分组码码字的结构是由其生成矩阵决定的，当然也可由一致校验矩阵决定。实际上，所谓检验就是利用 $\boldsymbol{H}$ 矩阵去鉴别接收矢量 $\boldsymbol{y}$ 的结构。若已知 $\boldsymbol{H}$ 矩阵，该码的结构也就知道了。那么从研究码纠错能力的角度来看，d_0 与 $\boldsymbol{H}$ 有什么关系呢？

首先，我们来看一个利用伴随式对码字译码的例子。

例 3-15 已知(7,3)线性分组码的一致校验矩阵为

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

对两个码字进行译码 $\boldsymbol{c}_1=(1110100)$，$\boldsymbol{c}_2=(0111010)$。我们假设有以下几种传输可能：

(1) 发送码字在传输中没有发生错误。

即 $\boldsymbol{e}=(0000000)$，此时伴随式为

$$\boldsymbol{s}_1^{\mathrm{T}}=\boldsymbol{H}\boldsymbol{y}_1^{\mathrm{T}}=\boldsymbol{H}\boldsymbol{c}_1^{\mathrm{T}}=\boldsymbol{0}^{\mathrm{T}}$$

$$\boldsymbol{s}_2^{\mathrm{T}}=\boldsymbol{H}\boldsymbol{y}_2^{\mathrm{T}}=\boldsymbol{H}\boldsymbol{c}_2^{\mathrm{T}}=\boldsymbol{0}^{\mathrm{T}}$$

(2) 发送码字在传输中发生一位错误。

若 $\boldsymbol{e}=(1000000)$，即传输中码字的第 1 位出错，则 $\boldsymbol{y}_1=(0110100)$，$\boldsymbol{y}_2=(1111010)$，则可根据接收矢量分别计算伴随式，得

$$\boldsymbol{s}_1^{\mathrm{T}}=\boldsymbol{H}\boldsymbol{y}_1^{\mathrm{T}}=\begin{bmatrix}1&0&1&1&0&0&0\\1&1&1&0&1&0&0\\1&1&0&0&0&1&0\\0&1&1&0&0&0&1\end{bmatrix}\begin{bmatrix}0\\1\\1\\0\\1\\0\\0\end{bmatrix}=\begin{bmatrix}1\\1\\1\\0\end{bmatrix}$$

$$\boldsymbol{s}_2^{\mathrm{T}}=\boldsymbol{H}\boldsymbol{y}_2^{\mathrm{T}}=\begin{bmatrix}1&0&1&1&0&0&0\\1&1&1&0&1&0&0\\1&1&0&0&0&1&0\\0&1&1&0&0&0&1\end{bmatrix}\begin{bmatrix}1\\1\\1\\1\\0\\1\\0\end{bmatrix}=\begin{bmatrix}1\\1\\1\\0\end{bmatrix}$$

若 $\boldsymbol{e}=(0100000)$，即传输中码字的第 2 位出错，则 $\boldsymbol{y}_1=(1010100)$，$\boldsymbol{y}_2=(0011010)$，则同样计算出伴随式，得

$$\boldsymbol{s}_1^{\mathrm{T}}=\boldsymbol{H}\boldsymbol{y}_1^{\mathrm{T}}=\begin{bmatrix}0\\1\\1\\1\end{bmatrix},\quad \boldsymbol{s}_2^{\mathrm{T}}=\boldsymbol{H}\boldsymbol{y}_2^{\mathrm{T}}=\begin{bmatrix}0\\1\\1\\1\end{bmatrix}$$

这说明，$\boldsymbol{s}$ 的确仅与 $\boldsymbol{e}$ 有关，而与发送码字无关。此外，对于该(7,3)码，若发生一位错误，计算得到的 $\boldsymbol{s}^{\mathrm{T}}$ 正好与 $\boldsymbol{H}$ 中的某一列矢量相同。也就是说，如果 $\boldsymbol{s}^{\mathrm{T}}$ 正好与 $\boldsymbol{H}$ 中的第 i 列相同，则接收码字的第 i 位出错。对于第一对接收码字，$\boldsymbol{s}^{\mathrm{T}}$ 均为 $\boldsymbol{H}$ 矩阵的第一列，因此有 $\boldsymbol{e}=(1000000)$；对于第二对接收码字，$\boldsymbol{s}^{\mathrm{T}}$ 均为 $\boldsymbol{H}$ 矩阵的第二列，因此有 $\boldsymbol{e}=(0100000)$。这正好与假设的错误图样一致。

(3) 发送码字在传输中发生两位错误。

若 $\boldsymbol{e}_1=(1100000)$，则 $\boldsymbol{y}_1=(0010100)$，若 $\boldsymbol{e}_2=(0010100)$，则 $\boldsymbol{y}_2=(0101110)$，根据接收

矢量分别计算伴随式，得

$$\boldsymbol{s}_1^{\mathrm{T}}=\boldsymbol{H}\boldsymbol{y}_1^{\mathrm{T}}=\begin{bmatrix}1\\0\\0\\1\end{bmatrix},\quad \boldsymbol{s}_2^{\mathrm{T}}=\boldsymbol{H}\boldsymbol{y}_2^{\mathrm{T}}=\begin{bmatrix}1\\0\\0\\1\end{bmatrix}$$

由于 $\boldsymbol{s}\neq\boldsymbol{0}$，说明传送的码字有错，但 $\boldsymbol{s}^{\mathrm{T}}$ 与 $\boldsymbol{H}$ 中任何一列均不相同，说明是不可纠的错误，即无法由 $\boldsymbol{s}$ 得到 $\boldsymbol{e}$。因为 $\boldsymbol{e}_1=(1100000)$，$\boldsymbol{e}_2=(0010100)$，表明这两个接收码字都有两位错误，但各自的错误位置不同。然而，$\boldsymbol{s}_1^{\mathrm{T}}$ 和 $\boldsymbol{s}_2^{\mathrm{T}}$ 却相同，这也说明无法由 $\boldsymbol{s}^{\mathrm{T}}$ 确定 $\boldsymbol{e}$。

虽然无法由 $\boldsymbol{s}^{\mathrm{T}}$ 确定 $\boldsymbol{e}$，但是 $\boldsymbol{s}^{\mathrm{T}}$ 与 $\boldsymbol{e}$ 是有关系的。$\boldsymbol{e}_1=(1100000)$，表明接收码字第一位和第二位出错，而 $\boldsymbol{s}_1^{\mathrm{T}}$ 正好是 $\boldsymbol{H}$ 中第一列与第二列之和。同样，$\boldsymbol{s}_2^{\mathrm{T}}$ 正好是 $\boldsymbol{H}$ 中第三列与第五列之和。这即表明 $\boldsymbol{s}^{\mathrm{T}}$ 与 $\boldsymbol{e}$ 有关系，同时也验证了式(3-14)。

(4) 发送码字在传输中发生三位错误。

若 $\boldsymbol{y}_1=(0000100)$，可判知 $\boldsymbol{e}=(1110000)$，计算伴随式，得

$$\boldsymbol{s}_1^{\mathrm{T}}=\boldsymbol{H}\boldsymbol{y}_1^{\mathrm{T}}=\begin{bmatrix}0\\1\\0\\0\end{bmatrix}$$

由于 $\boldsymbol{s}\neq\boldsymbol{0}$，说明传送的码字有错，但此时 $\boldsymbol{s}_1^{\mathrm{T}}$ 与 $\boldsymbol{H}$ 中第五列相等，是否说明是第五位出错呢？显然不是。因为该编码的最小距离为 4，由纠错性能与码距的关系可知，该码的纠错能力为纠正一位错误，检测两位或三位错误。同时，也可以分析出 $\boldsymbol{s}_1^{\mathrm{T}}$ 是 $\boldsymbol{H}$ 中第一、第二、第三列之和。

综上所述，一个(n,k)码要能纠正所有单个错误，则由所有单个错误的错误图样确定的 $\boldsymbol{s}$ 均不相同且不等于 $\boldsymbol{0}$。那么，一个(n,k)码怎样才能纠正小于或等于 t 个错误呢？这就必须要求所有小于或等于 t 个错误的所有可能组合的错误图样，都必须有不同的伴随式与之对应。

任一(n,k)线性分组码若要纠正小于或等于 t 个错误，其充要条件是 $\boldsymbol{H}$ 矩阵中任何 $2t$ 列线性无关。

例 3-16　以表 3-4 给出的(7,4)线性分组码为例。已知该码的校验矩阵 $\boldsymbol{H}$ 为

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

(1) 若传送时发生一位错误。

设 $\boldsymbol{e}_1=(1000000)$，计算伴随式得

$$\boldsymbol{s}_1=\boldsymbol{H}\boldsymbol{e}_1^{\mathrm{T}}=\begin{bmatrix}1\\1\\1\end{bmatrix}$$

$\boldsymbol{s}_1$ 正好就是矩阵 $\boldsymbol{H}$ 的第一列，在一般情况下，若传送时发生一位错码，如果错的是第 j 列，则伴随式 $\boldsymbol{s}$ 恰好就等于矩阵 $\boldsymbol{H}$ 的第 j 列。

(2) 若传送时发生两位错误。

设 $\boldsymbol{e}_2=(0001001)$，第四位和第七位出错，计算伴随式得

$$\boldsymbol{s}_2=\boldsymbol{H}\boldsymbol{e}_2^{\mathrm{T}}=\begin{bmatrix}0\\1\\0\end{bmatrix}$$

又设 $\boldsymbol{e}_3=(1000001)$，第一位和第七位出错，计算伴随式得

$$\boldsymbol{s}_3=\boldsymbol{H}\boldsymbol{e}_3^{\mathrm{T}}=\begin{bmatrix}1\\1\\0\end{bmatrix}$$

经观察可以发现，$\boldsymbol{s}_2$ 是 $\boldsymbol{H}$ 矩阵中第四列与第七列之和，$\boldsymbol{s}_3$ 是 $\boldsymbol{H}$ 矩阵中第一列与第七列之和。

(3) 若传送时发生三位错误。

$\boldsymbol{e}_4=(0111000)$，第二位、第三位和第四位出错，计算伴随式得

$$\boldsymbol{s}_4=\boldsymbol{H}\boldsymbol{e}_4^{\mathrm{T}}=\begin{bmatrix}0\\0\\0\end{bmatrix}$$

这种情况说明伴随式 $\boldsymbol{s}_4=\boldsymbol{0}$，表明无错，但实际情况是发生了错误，这说明三位(及其以上)的错误检测不出来，注意观察可以发现，$\boldsymbol{s}_4$ 是 $\boldsymbol{H}$ 矩阵中第二位、第三位和第四位之和。验证了伴随式 $\boldsymbol{s}$ 是 $\boldsymbol{H}$ 矩阵中码字出错位置所对应的列矢量的线性组合。

5. 译码后的错误概率

假设有 M 个码字(长度为 n)，它们被使用的概率相同。假设译码用标准阵列。记 α_i 为重量 i 的陪集首的个数。我们假定信道为二元对称信道(BSC)，符号错误概率为 p。如果错误图样 $\boldsymbol{e}$ 不是一个陪集首，则译码出错。因此正确译码概率为


译码后的错误概率

$$P_c=\sum_{i=0}^{n}\alpha_i p^i(1-p)^{n-i} \tag{3-15}$$

故错误概率为

$$P_e=1-P_c=1-\sum_{i=0}^{n}\alpha_i p^i(1-p)^{n-i} \tag{3-16}$$

例 3-17 考虑码 $C=\{0000,1011,0101,1110\}$，相应的标准阵列为

码字→0000，1011，0101，1110
1000，0011，1101，0110
0100，1111，0001，1010
0010，1001，0111，1100
↑
陪集首

陪集首为 0000、1000、0100、0010，可知 $\alpha_0=1$(只有一个重量等于零的陪集首)，$\alpha_1=3$(有 3 个重量等于 1 的陪集首)，而且所有其余的 $\alpha_i=0(i\geqslant 2)$。因此，在编码的情况下，错误概率为

$$P_e=1-P_c=1-\sum_{i=0}^{n}\alpha_i p^i(1-p)^{n-i}=1-[\alpha_0 p^0(1-p)^4+\alpha_1 p^1(1-p)^3]$$

$$=1-[p^0(1-p)^4+3p^1(1-p)^3]$$

该码有 4 个码字，且可用来每次发送两比特。如果不采用编码，$n=2$，陪集首为 00，$\alpha_0=1$（只有一个重量等于零的陪集首），而所有其余的 $\alpha_i=0(i\geqslant 1)$。

因此，无编码情况下的错误概率为

$$P_e=1-P_c=1-\sum_{i=0}^{n}\alpha_i p^i(1-p)^{n-i}=1-(1-p)^2$$

若 $p=0.01$，则在编码的情况下，码字出错的概率为 $P_e=0.01030$，而对于无编码的情况 $P_e=0.0199$。因此编码几乎把码字出错概率减小了一半。如图 3-4 对有编码和无编码情况下的 P_e 进行了比较。可以看出，当 $p=0.5$ 时，编码和无编码的误码率相同，此时任何编码方法都不能降低误码率；当 $p<0.5$ 时编码性能优于无编码性能，当 $p>0.5$ 时编码性能反而劣于有编码性能，出现这种情况的原因与 1.5 节的译码规则有关。注意编码带来的改进是以信息传输速率的减小为代价的，由于我们对任意两个信息比特都要发送两个校验比特，因此信息传输速率被降低为原来的一半。

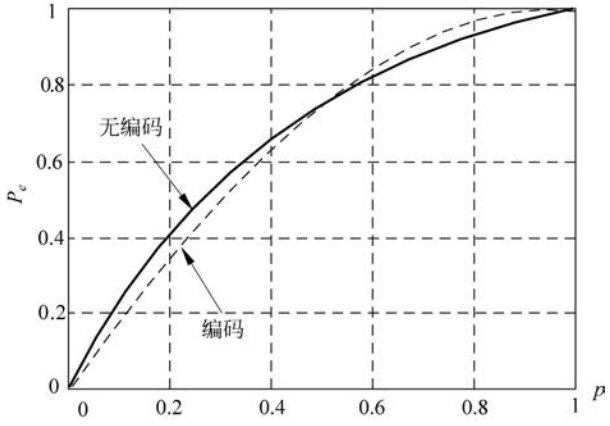


图 3-4　消息有编码和无编码的误码率比较

例 3-18　考虑一个符号出错概率为 $p=10^{-7}$ 的 BSC 信道。假设 10bit 长的码字未经编码就被传输了。设发送端的比特速率为 10^7bit/s，这表明每秒可以发送 10^6 个码字。一个码字不能被正确接收的概率为

$$\binom{10}{1}(1-p)^9 p+\binom{10}{2}(1-p)^8 p^2+\binom{10}{3}(1-p)^7 p^3\cdots\approx\binom{10}{1}(1-p)^9 p\approx 10^{-6}$$

因此一秒将有 $10^{-6}\times 10^6=1$ 个码字出错！这意味着每秒都有一个错误而且它未被检测到。

下面我们在未编码的码字中加进一个奇偶校验比特使它们变为 11bit。该奇偶校验比特使所有码字为偶校验，这样可以确保单个错误被检测到。仅当两个或更多比特有错时译码后的码字才会有错，即至少有两个比特有错。单个比特有错的情况可以以概率 1 被检测出来。因此码字错误概率将为

$$1-(1-p)^{11}-\binom{11}{1}(1-p)^{10}p\approx 1-(1-11p)-11(1-10p)p=110p^2=11\times 10^{-13}$$

新的码率为每秒 $10^7/11$ 个码字，因为每个码字有 11bit 而比特速率与以前相同。因此 1 秒将有 $(10^7/11)\times(11\times10^{-13})=10^{-6}$ 个码字出错。这表明编码后每 10^6 秒即大约 11.6 天将有一个错误码字未被检测而被错误地接收！

可以看出，仅仅把字长从 10bit(未编码)增加到 11bit(编码)，就能使码字出错概率惊人地减小。

3.7 汉明码

汉明码是 1951 年由汉明提出的能纠正单个错误的线性分组码。它性能良好，既具有较高的可靠性，又具有较高的传输效率，而且编译码电路较为简单，易于工程实现，因此汉明码在发现后不久，就得到了广泛的应用。

3.7.1 汉明码的构造

汉明码是一个能纠正单个错误，且信息传输率(码率 $R=k/n$)最大的线性分组码。我们已经知道，具有纠正单个错误能力的线性分组码的最小距离应为 3，即要求其 $\boldsymbol{H}$ 矩阵中至少任意两列线性无关。要做到这一点，只要 $\boldsymbol{H}$ 矩阵满足“两无”——无相同的列，无全零列就可以了。

一个(n,k)线性分组码的 $\boldsymbol{H}$ 矩阵是一个 $(n-k)\times n=r\times n$ 阶矩阵，这里 $r=n-k$ 是校验元的数目。显然，r 个校验元能组成 2^r 列互不相同的 r 重矢量，其中非全零矢量有 2^r-1 个。如果用这 2^r-1 个非全零矢量作为 $\boldsymbol{H}$ 矩阵的全部列，即令 $\boldsymbol{H}$ 矩阵的列数 $n=2^r-1$，则此 $\boldsymbol{H}$ 矩阵的各列均不相同，且无全零列，由此可构造一个能纠正单个错误的(n,k)线性分组码。

同时，2^r-1 是 n 所能取的最大值，因为如果 $n>2^r-1$，那么 $\boldsymbol{H}$ 矩阵的 n 列中必会出现相同的两列，这样就不能满足对 $\boldsymbol{H}$ 矩阵的要求。而由于 $n=2^r-1$ 是 n 所能取的最大值，也就意味着码率 R 取得了最大值，即

$$R=\frac{k}{n}=\frac{n-r}{n}=1-\frac{r}{n}=1-\frac{r}{2^r-1} \tag{3-17}$$

这样设计出来的码是符合要求的，这样的码就是汉明码。

定义 3-1 若 $\boldsymbol{H}$ 矩阵的列是由非全零且互不相同的所有二进制 r 重矢量组成，则由此得到的线性分组码，称为 GF(2)上的$(2^r-1,2^r-1-r)$汉明码。

表 3-10 列出了几种 r 取不同值时汉明码的$(n=2^r-1,k=2^r-1-r)$值。

表 3-10 几种汉明码的(n,k)值

r	$n=2^r-1$	$k=2^r-1-r$
1	1	0
2	3	1
3	7	4
4	15	11
5	31	26
6	63	57

例 3-19 二元(7,4)汉明码的生成矩阵为

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

相应的校验矩阵为

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

观察到该校验矩阵的列由(100),(010),(101),(110),(111),(011)和(001) 构成,这 7 个是所有长为 3 的非零二元向量,很容易可以得到一个系统汉明码。该校验矩阵 $\boldsymbol{H}$ 可以被安排成如下的系统型:

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

于是该二元汉明码的生成矩阵的系统型为

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

例 3-20 取 $r=3$,构造 GF(2)上的(7,4)汉明码。

当 $r=3$ 时,有 7 个非全零的三重矢量

(100),(010),(101),(110),(111),(011),(001)

构成矩阵

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

由此得到一个能纠正单个错误的(7,4)汉明码。若码字传输中左边第一位出错,则相应的伴随式 $\boldsymbol{s}=(001)$就是 $\boldsymbol{H}$ 矩阵的第一列,也正好是“1”的二进制表示。同理可知,无论哪一位出错,它对应的伴随式就是该位的二进制表示,故译码十分方便,特别适用于计算机内部运算和记忆系统中的纠错。

如果要得到系统码形式的 $\boldsymbol{H}$ 矩阵,只需对上述矩阵进行初等变换交换列即可,则

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

相应地,生成矩阵 $\boldsymbol{G}$ 为

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

由此构成的(7,4)汉明码如表 3-4 所示。

3.7.2 汉明限与完备码

一个二进制(n,k)线性分组码，若要纠正t个错误，则应使小于或等于t个错误所组成的所有错误图样，都必须有不同的伴随式与之对应，即不等式成立

$$2^{n-k} \geqslant \binom{n}{0}+\binom{n}{1}+\cdots+\binom{n}{t}=\sum_{i=0}^{t}\binom{n}{i} \tag{3-18}$$

式中，2^{n-k}为全部$r=n-k$重矢量数目，即伴随式数目；$\sum_{i=0}^{t}\binom{n}{i}$为所有错误个数小于或等于$t$的错误图样数。

式(3-18)称为汉明限。该限是构造任何二进制码所必须满足的，也就是构造码的必要条件。

如果某一(n,k)线性分组码能使式(3-18)等号成立，即错误图样总数正好等于伴随式数目，则称这种码为完备码。完备码相当于在标准阵列中，能将重量小于等于t的所有错误图样作为陪集首，而大于t的错误图样都不作为陪集首，其校验元得到充分的利用。显然无论r取何值，汉明码都是可纠正$t=1$位错误的完备码。

如果一个(n,k)线性分组码，除了能将重量小于或等于t的所有错误图样作为陪集首外，还有部分(但不是全部)重量大于t的错误图样作为陪集首，则称这种码为准完备码。

表3-8列出的二元(5,3)码，就不是一个完备码，由它的生成矩阵

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

可得到校验矩阵

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

由此算出$l=1$(l是$\boldsymbol{H}$中线性无关的列数)，$d=2$(d是码的最小距离，$d=l+1$)，$t=\left[\frac{d-1}{2}\right]=\left[\frac{2-1}{2}\right]=0$(纠错能力的计算方法)，纠错能力为$t=0$，除全零码外，标准阵列中还有部分重量等于1的矢量作为陪集首。

例 3-21 (7,4)系统码生成矩阵为

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

校验矩阵为

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

由此算出$l=2$，$d=3$，$t=\left[\frac{d-1}{2}\right]=\left[\frac{3-1}{2}\right]=1$，列出它的标准阵列如表3-11所示。

从表3-11可看出，上述(7,4)码是完备汉明码。

表 3-11 （7，4）系统码的标准阵列

0000000	0001011	0010101	001110	0100110	0101101	0110011	0111000	1000111	1001100	1010010	1011001	1100001	1101010	1110100	1111111
0000001	0001010	0010100	0011111	0100111	0101100	0110010	0111001	1000110	1001101	1010011	1011000	1100000	1101011	1110101	1111110
0000010	0001001	0010111	0011100	0100100	0101111	0110001	0111010	1000101	1001110	1010000	1011011	1100011	1101000	1110110	1111101
0000100	0001111	0010001	0011010	0100010	0101001	0110111	0111100	1000011	1001000	1010110	1011101	1100101	1101110	1110000	1111011
0001000	0000011	0011101	0010110	0101110	0100101	0111011	0110000	1001111	1000100	1011010	1010001	1101001	1100010	1111100	1110111
0010000	0011011	0000101	0001110	0110110	0111101	0100011	0101000	1010111	1011100	1000010	1001001	1110001	1111010	1100100	1101111
0100000	0101011	0110101	0111110	0000110	0001101	0100011	0011000	1100111	1101100	1110010	1111001	1000001	1001010	1010100	1011111
1000000	1001011	1010101	1011110	1100110	1101101	1110011	1111000	0000111	0001100	0010010	0011001	0100001	0101010	0110100	0111111

3.8 线性分组码的实现与仿真

利用 MATLAB 来实现线性分组码的生成矩阵、校验矩阵、编码、译码，可以进一步加深对线性分组码的概念以及实现原理的理解；通过编程和 Simulink 对通信过程进行仿真，可以对通信系统的组成及各组成模块之间的关系有较好的掌握；同时通过仿真可以理解纠错编码在整个通信过程中的重要性。

3.8.1 生成矩阵与校验矩阵

1. 从码元符号与信息符号的关系得到生成矩阵

在进行编码时，有时候并不知道编码的生成矩阵，而只知道码元符号与信源符号之间的线性关系，由这些线性关系可以求出生成矩阵。

对于例 3-2，已知码元符号与信源符号的关系，获得生成矩阵的方法如下：

程序 3_1（program3_1.m）

```
%由已知的线性关系得出生成矩阵
%信源符号 u=(u3,u2,u1,u0),码符号 c=(c6,c5,c4,c3,c2,c1,c0)
%码符号与信源符号的关系为 c6=u3,c5=u2,c4=u1,c3=u3+u2+u1,c2=u0,c1=u3+u2+u0,
%c0=u3+u1+u0
%将码符号的每一位用 4 位信源符号来表示
c6=[1 0 0 0]; %c6=u3,只与 u3 有关,对应系数为 1,与 u2,u1,u0 无关,则对应系数为 0
c5=[0 1 0 0]; %c5=u2
c4=[0 0 1 0]; %c4=u1
c3=[1 1 1 0]; %c3=u3+u2+u1
c2=[0 0 0 1]; %c2=u0
c1=[1 1 0 1]; %c1=u3+u2+u0
c0=[1 0 1 1]; %c0=u3+u1+u0
G=[(c6)',(c5)',(c4)',(c3)',(c2)'(c1)',(c0)'];
```

运行该程序后，在 MATLAB 命令窗口（COMMAND WINDOW）输入 G，结果如下：

```
G =
    1    0    0    1    0    1    1
    0    1    0    1    0    1    0
    0    0    1    1    0    0    1
    0    0    0    0    1    1    1
```

2. 从码元符号与信息符号的关系得到校验矩阵

在进行编码时，有时候并不知道编码的校验矩阵，而只知道校验位与信源符号之间的线性关系，由这些线性关系可以求出校验矩阵。

对于 3.2.2 节内容，已知信息元与校验元的关系，获得校验矩阵的方法如下：

程序 3_2(program3_2. m)

```
%由已知的线性关系得出校验矩阵
%信源符号 u = (u2,u1,u0),码符号 c = (c6,c5,c4,c3,c2,c1,c0)
%码符号与信源符号的关系为 c6 = u2,c5 = u1,c4 = u0,c3 = u2 + u0,c2 = u2 + u1 + u0,c1 = u2 + u1,
%c0 = u1 + u0
%则信息元与校验元的关系为 c3 = c6 + c4,c2 = c6 + c5 + c4,c1 = c6 + c5,c0 = c5 + c4
%因二进制数自身相加为 0,则有 c6 + c4 + c3 = 0,c6 + c5 + c4 + c2 = 0,c6 + c5 + c1 = 0,c5 + c4 + c0 = 0
%将等式用码符号的 7 位符号来表示
%c6 + c4 + c3 = 0,与 c6,c4,c3 有关,对应系数为 1,与 c5,c2,c1,c0 无关,则对应系数为 0
%1 × c6 + 0 × c5 + 1 × c4 + 1 × c3 + 0 × c2 + 0 × c1 + 0 × c0 = 0
%[1 0 1 1 0 0 0]·[c6 c5 c4 c3 c2 c1 c0]' = 0
r3 = [1 0 1 1 0 0 0];
%c6 + c5 + c4 + c2 = 0,则 1 × c6 + 1 × c5 + 1 × c4 + 0 × c3 + 1 × c2 + 0 × c1 + 0 × c0 = 0
%[1 1 1 0 1 0 0]·[c6 c5 c4 c3 c2 c1 c0]' = 0
r2 = [1 1 1 0 1 0 0];
%c6 + c5 + c1 = 0,则 1 × c6 + 1 × c5 + 0 × c4 + 0 × c3 + 0 × c2 + 1 × c1 + 0 × c0 = 0
%[1 1 0 0 0 1 0]·[c6 c5 c4 c3 c2 c1 c0]' = 0
r1 = [1 1 0 0 0 1 0];
%c5 + c4 + c0 = 0,则 0 × c6 + 1 × c5 + 1 × c4 + 0 × c3 + 0 × c2 + 0 × c1 + 1 × c0 = 0
%[0 1 1 0 0 0 1]·[c6 c5 c4 c3 c2 c1 c0]' = 0
r0 = [0 1 1 0 0 0 1];
H = [r3;r2;r1;r0];
```

运行该程序后,在 MATLAB 命令窗口(COMMAND WINDOW)输入 H,结果如下:

```
H =
     1     0     1     1     0     0     0
     1     1     1     0     1     0     0
     1     1     0     0     0     1     0
     0     1     1     0     0     0     1
```

3. 利用 hammgen()函数产生生成矩阵和校验矩阵

MATLAB 有大量的库函数,很方便编程。可以利用 MATLAB 的 hammgen()函数产生生成矩阵和校验矩阵。

hammgen 函数的功能:产生汉明码生成矩阵和校验矩阵。

语法:$\boldsymbol{H}$=hammgen(r);

$[\boldsymbol{H},\boldsymbol{G}]$=hammgen(…);

$[\boldsymbol{H},\boldsymbol{G},n,k]$=hammgen(…);

说明:用 $\boldsymbol{H}$=hammgen(r)产生 $r\times n$ 汉明校验矩阵(r 为校验位数,n 是码长)。

用$[\boldsymbol{H},\boldsymbol{G}]$= hammgen(…)产生生成矩阵 $\boldsymbol{G}$ 和校验矩阵 $\boldsymbol{H}$。

用$[\boldsymbol{H},\boldsymbol{G},n,k]$= hammgen($r$)产生生成矩阵 $\boldsymbol{G}$ 和校验矩阵 $\boldsymbol{H}$,码长 n 以及信息位长度 k。

例如,若 r=3,$[\boldsymbol{H},\boldsymbol{G},n,k]$= hammgen(3)。

代码$[H,G,n,k]$=hammgen(3)运行结果为

```
H =
    1    0    0    1    0    1    1
    0    1    0    1    1    1    0
    0    0    1    0    1    1    1
G =
    1    1    0    1    0    0    0
    0    1    1    0    1    0    0
    1    1    1    0    0    1    0
    1    0    1    0    0    0    1
n =
    7
k =
    4
```

4. 生成矩阵与校验矩阵的相互转换

如果知道生成矩阵，根据生成矩阵与校验矩阵之间的关系可以求出校验矩阵。同样，如果知道校验矩阵，根据生成矩阵与校验矩阵之间的关系可以求出生成矩阵。

(1) 标准形式的生成矩阵转化为校验矩阵。

实现方法如下：

```
%标准形式的生成矩阵转化为校验矩阵
G=[1 0 0 0 1 0 1; 0 1 0 0 1 1 1; 0 0 1 0 0 1 0; 0 0 0 1 0 1 0]; %标准形式的生成矩阵
H=gen2par(G); %该函数用来将标准形式的生成矩阵转换为校验矩阵
```

运行结果如下：

```
H =
    1    1    0    0    1    0    0
    0    1    1    1    0    1    0
    1    1    0    0    0    0    1
```

(2) 标准形式的校验矩阵转化为生成矩阵。

实现方法如下：

```
%标准形式的校验矩阵转化为生成矩阵
H=[1 0 1 1 0 0 0; 1 1 1 0 1 0 0; 1 1 0 0 0 1 0; 0 1 1 0 0 0 1]; %标准形式的校验矩阵
G=gen2par(H); %该函数用来将标准形式的校验矩阵转换为生成矩阵
```

运行结果如下：

```
G =
    1    0    0    1    1    1    0
    0    1    0    0    1    1    1
    0    0    1    1    1    0    1
```

3.8.2 编码

利用 MATLAB 实现线性分组码编码的方法比较多，我们对其中的几种方法加以介绍。

1. 利用 encode 函数来实现编码

语法：code=encode(msg,n,k)；

code＝encode(msg,n,k,method,opt)；

[code,added]＝encode(…)；

说明：这个函数可完成三种主要的差错控制编码，汉明码、线性分组码、循环码。

code＝encode(msg，n，k)对二进制信息 msg 进行汉明编码。信息位为 k bit，码字长为 n bit。生成矩阵是系统默认的 msg 可用一个矢量形式或 k 列矩阵的形式表达。汉明码是一种可纠正单个错误的线性分组码。

code＝encode(msg,n,k,method，opt)是使用这一函数的通用形式。式中，msg 是信息；method 是编码方式；n 是码字长度；k 是信息位长度；opt 是有些编码方式需要的参数，具体含义如表 3-12 所示。

表 3-12　encode 函数的参数用法

method	含　义	opt
'hamming'	汉明编码	可用来指定一个原始多项式，如省略，则使用默认多项式
'linear'	线性分组码	opt 必须指定一个校验矩阵
'cyclic'	循环码	必须指定一个生成多项式

在所有编码的方式中，信息位必须以二进制矢量方式或列矩阵的形式表示。

在[code,added]＝encode(…)中，输出函数为使输入信息位数达到 k 时所需添加的列数，添加的信息位是“0”。

```
code = encode([1 0 0 1],7,4); % 产生(7,4)汉明码,生成矩阵是系统默认的,这里输入的 k = 4
```

运行结果如下：

```
code =
     0     1     1     1     0     0     1
[code,added] = encode([1 0 0],7,4);    % 这里输入的 k = 3,编码时添加了 1 位"0"
```

运行结果如下：

```
code =
     1     1     0     1     0     0     0
added =
     1
```

下面以线性分组码为例，看看如何使用 encode 函数。

已知(7,4)线性分组码，其生成矩阵为 $\boldsymbol{G}=\begin{bmatrix}1&0&0&1&0&1&1\\0&1&0&1&0&1&0\\0&0&1&1&0&0&1\\0&0&0&0&1&1&1\end{bmatrix}$，根据不同的情况分别求编码。

(1) 如果信息序列已知，如 $\boldsymbol{m}=[1\ 0\ 0\ 1]$，采用线性分组码编码，则编码的语句如下：

```
msg = [1 0 0 1]; % 已知的信息序列
code = encode(msg,7,4,'linear', [1 0 0 1 0 1 1; 0 1 0 1 0 1 0; 0 0 1 1 0 0 1; 0 0 0 0 1 1 1]);
 % 已知生成矩阵 G,调用 encode 函数进行(7,4)线性分组码的编码
```

运行结果如下：

```
code =
    1    0    0    1    1    0    0
```

(2) 如果信息序列是随机产生的，则编码的语句如下：

```
msg = randi([0,1],1,4); % 随机产生长度为 4 的信息序列
code = encode(msg,7,4,'linear', [1 0 0 1 0 1 1; 0 1 0 1 0 1 0; 0 0 1 1 0 0 1; 0 0 0 0 1 1 1]);
% 已知生成矩阵 G,调用 encode 函数进行(7,4)线性分组码的编码
```

运行结果如下：

```
msg =
    0    0    1    0(每次运行产生的信息序列不同)
code =
    0    0    1    1    0    0    1
```

(3) 如果要随机产生长串的信息序列，则编码的语句如下：

```
msg = randi([0,1],1,1000); % 随机产生长度为 1000 的信息序列
code = encode(msg,7,4,'linear', [1 0 0 1 0 1 1; 0 1 0 1 0 1 0; 0 0 1 1 0 0 1; 0 0 0 0 1 1 1]);
% 已知生成矩阵 G,调用 encode 函数进行(7,4)线性分组码的编码
```

注：如果产生的信息序列的长度不是 4 的整数倍，则编码时会自动在序列的末尾补零。

2. 利用生成矩阵来实现编码

利用 encode 函数编码，编码过程隐含在 encode 函数内部，为了加深对编码原理的理解，下面根据编码原理的编码步骤来实现译码。

例 3-22 如果生成矩阵为 $\boldsymbol{G}=\begin{bmatrix}1&0&0&1&0&1&1\\0&1&0&1&0&1&0\\0&0&1&1&0&0&1\\0&0&0&0&1&1&1\end{bmatrix}$，根据不同情况分别求编码结果。

(1) 信息序列 $\boldsymbol{m}=[1\ 0\ 1\ 1]$，求编码后的序列 c。则编码的语句如下：

```
G = [1 0 0 1 0 1 1; 0 1 0 1 0 1 0; 0 0 1 1 0 0 1; 0 0 0 0 1 1 1];      % 生成矩阵
m = [1 0 1 1];                                                          % 信息码字
c = rem(m * G,2);                                                       % 生成码字
disp(c)                                                                 % 显示编码结果
```

运行结果如下：

```
1    0    1    0    1    0    1
```

(2) 如果自己设置信息序列，并显示编码结果，则语句如下：

```
G = [1 0 0 1 0 1 1; 0 1 0 1 0 1 0; 0 0 1 1 0 0 1; 0 0 0 0 1 1 1];      % 生成矩阵
m = input('请输入信息码字: ');                                          % 从命令窗口输入参数的函数
c = rem(m * G,2);                                                       % m 与 G 相乘后进行模 2 运算
fprintf('编码为: c = ')                                                 % 输出字符
```

```
disp(c);
```

注：输入的序列必须是矩阵形式的，如[1 0 1 0]，而不能为 1010 或 1 0 1 0。

运行结果如下：

```
请输入信息码字：[1 0 1 0]
编码为 c =     1     0     1     0     0     1     0
```

(3) 如果要得到所有的编码序列，并显示编码结果，则语句如下：

```
G = [1 0 0 1 0 1 1; 0 1 0 1 0 1 0; 0 0 1 1 0 0 1; 0 0 0 0 1 1 1]; % 生成矩阵
% 利用循环语句产生所有可能的信息序列
for i = 0: 1: 15
      a = dec2bin(i,4); % 将十进制的整数转换为二进制序列
      c = mod(a * G,2);
      disp(a);
      disp('对应的码字为：');
      disp(c);
end
```

运行结果如下：

```
0000 对应的码字为

0     0     0     0     0     0     0

0001 对应的码字为

0     0     0     0     1     1     1

0010 对应的码字为

0     0     1     1     0     0     1

0011 对应的码字为

0     0     1     1     1     1     0

0100 对应的码字为

0     1     0     1     0     1     0

0101 对应的码字为

0     1     0     1     1     0     1

0110 对应的码字为

0     1     1     0     0     1     1

0111 对应的码字为

0     1     1     0     1     0     0

1000 对应的码字为

1     0     0     1     0     1     1
```

1001 对应的码字为

```
1    0    0    1    1    0    0
```

1010 对应的码字为

```
1    0    1    0    0    1    0
```

1011 对应的码字为

```
1    0    1    0    1    0    1
```

1100 对应的码字为

```
1    1    0    0    0    0    1
```

1101 对应的码字为

```
1    1    0    0    1    1    0
```

1110 对应的码字为

```
1    1    1    1    0    0    0
```

1111 对应的码字为

```
1    1    1    1    1    1    1
```

也可以用下面的方法得到所有的码字。

(7,4)分组码，已知校验矩阵，由校验矩阵求出生成矩阵，列出所有的信息码字，求出所有的编码码字。

```
H = [1 1 1 0 1 0 0; 0 1 1 1 0 1 0; 1 1 0 1 0 0 1]; % 已知校验矩阵
G = gen2par(H);                                    % 调用 MATLAB 函数求与 H 对应的生成矩阵
msg = [0 0 0 0; 0 0 0 1; 0 0 1 0; 0 0 1 1; 0 1 0 0; 0 1 0 1; 0 1 1 0; 0 1 1 1; …
      1 0 0 0; 1 0 0 1; 1 0 1 0; 1 0 1 1; 1 1 0 0; 1 1 0 1; 1 1 1 0; 1 1 1 1]; % 列出所有信息序列
C = rem(msg * G,2);
```

运行结果如下：

```
C =
    0    0    0    0    0    0    0
    0    0    0    1    0    1    1
    0    0    1    0    1    1    0
    0    0    1    1    1    0    1
    0    1    0    0    1    1    1
    0    1    0    1    1    0    0
    0    1    1    0    0    0    1
    0    1    1    1    0    1    0
    1    0    0    0    1    0    1
    1    0    0    1    1    1    0
    1    0    1    0    0    1    1
    1    0    1    1    0    0    0
```

```
1    1    0    0    0    1    0
1    1    0    1    0    0    1
1    1    1    0    1    0    0
1    1    1    1    1    1    1
```

3. 利用自定义函数来实现编码

为了方便编码，在已知生成矩阵的情况下可以利用自定义函数来实现编码。

对于(7,4)汉明码，若 $\boldsymbol{G}=\begin{bmatrix}1&0&0&0&1&0&1\\0&1&0&0&1&1&1\\0&0&1&0&1&1&0\\0&0&0&1&0&1&1\end{bmatrix}$，求编码码字。

方法一　如果给定 k 位信息，求出相应的 n 位码字，对应(7,4)汉明编码，实现的方法如下：

自定义函数：hmencode74_1.m，即

```
function bitcoded = hmencode74_1(m)
对输入的任意 4 位信息序列进行汉明编码,并输出编码序列
G = [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];    % (7,4)汉明码的生成矩阵
bitcoded = mod(msg * G,2);                                           % 编码的码字
disp('编码后序列为: ');
disp(bitcoded);
```

程序 3_3(program3_3.m)

```
% 该程序调用 hmencode74_1.m 函数文件,得到编码结果
msg = [1 0 0 1];                    % 给定的 4 位信息
bitcoded = hmencode74_1(msg);       % 调用编码函数 hmencode74_1,得到编码码字
```

运行结果如下：

```
编码后序列为:
    1    0    0    1    1    1    0
```

对于其他的编码方式改变相应的参数和生成矩阵就可以得到相应的编码结果。

方法二　如果给定或随机产生的长串输入信息序列，要求出相应的编码序列，实现方法如下：

自定义的函数：hmencode74_2.m，即

```
function bitcoded = hmencode74_2(msg,G,k)
A = vec2mat(msg,k);                 % 输入的序列转换为矩阵,矩阵的列为 k
U = A * G;
U = mod(U,2);
bitcoded = reshape(U',1,[]);        % 重新构造为行矩阵输出
disp(bitcoded);                     % 显示编码结果
```

程序 3_4(program3_4.m)

```
% 该程序调用 hmencode74_2.m 函数文件,得到编码序列
msg = randi([0,1],1,40);            % 随机产生的信息序列
```

```
G= [1 0 0 1 0 1 1; 0 1 0 1 0 1 0; 0 0 1 1 0 0 1; 0 0 0 0 1 1 1];    %生成矩阵
k=4;                                  %信息元长度
bitcoded=hmencode74_2 (msg,G,k);  %调用编码函数 hmencode74_2,得到编码序列
```

运行结果如下：

```
Columns 1 through 27
0    1    0    1    1    0    1    1    1    0    0    0    0    1    0    1    0    1
1    0    1    0    0    0    0    0    0
Columns 28 through 54
0    1    1    1    1    0    0    0    1    1    0    0    1    1    0    0    0    0
0    0    0    0    0    0    0    0    0
Columns 55 through 70
0    0    1    1    0    0    0    0    1    0    1    0    1    0    1    0
```

注：每次运行结果均不相同，因为序列是随机产生的。

方法一和方法二的区别有：方法一只能对单个长度为 k 的信息求编码码字。方法二不限信息的长度，对输入的任意长串序列进行编码。方法二将生成矩阵、信息长度和编码长度参数都放在程序 3_4 中，这样更具有通用性，在不用修改函数文件的情况下，只要改变程序 3_4 中相应的参数，就可以进行其他方式的编码；如果将生成矩阵、信息长度和编码长度参数都放在 encode74_2.m 函数文件中，要进行其他的编码，则必须修改函数文件。

3.8.3 译码

1. 利用库函数(decode)来实现译码

语法：msg=decode(code,n,k);

msg=decode(code,n,k,method,opt1,opt2,opt3,opt4);

[msg,err]=decode(…);

[msg,err,ccode]=decode(…);

[msg,err,ccode,cerr]=decode(…);

说明：这个函数对接收到的码字进行译码，恢复出原始的信息，译码参数和方式必须和编码时采用的严格相同。

msg=decode(code,n,k)是对码长为 n，信息位长度为 k 的汉明码进行译码。

msg=decode(code,n,k,method,opt1,opt2,opt3,opt4)对接收到的码字，按 method 指定的方式进行译码。opt1,opt2,opt3 和 opt4 是可选项参数，它们的用法如表 3-13 所示。

表 3-13 decode 函数的参数用法

method	含　义	opt
'hamming'	汉明译码	opt1 可用来指定一个原始多项式，也可省略不用，opt2 不用
'linear'	线性分组码译码	opt1 必须指定一个校验矩阵，opt2 用来指定一个检错逻辑电路，如果省略，则默认单个错纠正逻辑
'cyclic'	循环码译码	opt1 是必须指定的生成多项式，可使用 cycpoly 函数选择一个合适的循环多项式，opt2 用来指定一个检错逻辑电路，如果省略，则默认单个错纠正逻辑

译码器输出 msg 与码字 code 的格式匹配，当码字是一个 n 列矩阵时，输出 msg 信息以 k 列矩阵表示。当 decode 函数输入的码字与 encode 函数的格式不一样时，这个函数停止工作。

用[msg,err]=decode(…)可输出译码过程检测出的错误数。当 err 为复数时，它表示纠错逻辑对差错控制无能为力了。

[msg,err,ccode]=decode(…)输出纠正的码字。

[msg,err,ccode,cerr]=decode(…)输出 ccode 每行的错误数。在'convol'方式下，cerr 表示输出译码判决的计算尺度，而不是纠错个数。

下面以线性分组码为例，看看如何使用 decode 函数。

已知(7,4)线性分组码，生成矩阵为 $\boldsymbol{G}=\begin{bmatrix}1&0&0&0&1&0&1\\0&1&0&0&1&1&1\\0&0&1&0&0&1&0\\0&0&0&1&0&1&0\end{bmatrix}$，在不同的情况下分别求译码结果。

(1) 如果接收到的序列为已知，若 $\boldsymbol{r}$=[1 0 0 1 0 1 1]，进行线性分组码译码，则译码的语句如下：

```
r = [1 0 0 1 0 1 1]; % 接收到的序列
msg = decode(r,7,4,'linear', [1 0 0 0 1 0 1; 0 1 0 0 1 1 1; 0 0 1 0 0 1 0; 0 0 0 1 0 1 0]); % 线性分
% 组码译码,生成矩阵必须是标准形式的
```

运行结果如下：

```
msg =
    1    0    0    1
```

若 $\boldsymbol{r}$=[1 0 0 1 0 1 0]，则译码的语句如下：

```
r = [1 0 0 1 0 1 0]; % 接收到的序列
msg = decode(r,7,4,'linear', [1 0 0 0 1 0 1; 0 1 0 0 1 1 1; 0 0 1 0 0 1 0; 0 0 0 1 0 1 0]); % 线性分
% 组码译码,生成矩阵必须是标准形式的
```

运行结果如下：

```
msg =
    0    0    0    1
```

(2) 如果接收到的码字是随机产生的，则译码的语句如下：

```
r = randi([0,1],1,7); % 随机产生长度为 7 的接收序列
msg = decode(r,7,4,'linear', [1 0 0 0 1 0 1; 0 1 0 0 1 1 1; 0 0 1 0 0 1 0; 0 0 0 1 0 1 0]); % 线性分
% 组码译码,生成矩阵必须是标准形式的
```

运行结果如下：

```
r =
    1    1    1    1    1    0    1(每次运行产生的码不同)
msg =
    1    0    1    1
```

（3）如果接收到的是随机产生的长串序列，则译码的语句如下：

```
r = randi([0,1],1,1000); % 随机产生长度为 1000 的信息序列
msg = decode(r,7,4,'linear', [1 0 0 0 1 0 1; 0 1 0 0 1 1 1; 0 0 1 0 0 1 0; 0 0 0 1 0 1 0]); % 线性分
% 组码译码,生成矩阵必须是标准形式的
```

注：如果产生的接收序列的长度不是 7 的整数倍，则译码时会自动在序列的末尾补零。

2. 利用校验矩阵实现译码

利用 decode 函数译码，译码过程隐含在 decode 函数内部，为了加深对译码原理的理解，下面根据译码原理的译码步骤来实现译码。

已知校验矩阵 $\boldsymbol{H}=\begin{bmatrix}1&1&1&0&1&0&0\\1&1&0&1&0&1&0\\1&0&1&1&0&0&1\end{bmatrix}$，求接收到码字的译码。

（1）利用错误图样译码。

程序 3_5（program3_5.m）

```
% (7,4)线性码的译码
% r 为接收到的码元,H 为监督矩阵
% S 为校正子,code 为纠错后的码元
r = input('请输入接收到的码元(矩阵形式 7 位): ');          % 从命令窗口输入接收码字
H = [ 1 1 1 0 1 0 0; 1 1 0 1 0 1 0; 1 0 1 1 0 0 1];          % 校验矩阵
S0 = rem([0 0 0 0 0 0 0] * H',2);                            % 求错误图样的校正子
S1 = rem([0 0 0 0 0 0 1] * H',2);
S2 = rem([0 0 0 0 0 1 0] * H',2);
S3 = rem([0 0 0 0 1 0 0] * H',2);
S4 = rem([0 0 0 1 0 0 0] * H',2);
S5 = rem([0 0 1 0 0 0 0] * H',2);
S6 = rem([0 1 0 0 0 0 0] * H',2);
S7 = rem([1 0 0 0 0 0 0] * H',2);
S = rem(r * H',2);                                           % 模 2 加
if S == S0
    code = bitxor(r,[0 0 0 0 0 0 0]);                        % 由接收码字和对应的错误图样求码字
end
if S == S1
    code = bitxor(r,[0 0 0 0 0 0 1]);
end
if S == S2
      y1 = bitxor(r,[0 0 0 0 0 1 0]);
end
if S == S3
    code = bitxor(r,[0 0 0 0 1 0 0]);
end
if S == S4
    code = bitxor(r,[0 0 0 1 0 0 0]);
end
if S == S5
      y1 = bitxor(r,[0 0 1 0 0 0 0]);
end
```

```
if S == S6
    code = bitxor(r,[0 1 0 0 0 0 0]);
end
if S == S7
    code = bitxor(r,[1 0 0 0 0 0 0]);
end
disp('纠错后的码元为: ');
disp(code);
u = zeros(1,4);
u = [code(: ,1),code(: ,2),code(: ,3),code(: ,4)]; % 因是系统码,因此原信息码是编码的前 4 位
disp('原信息码元为: ');
disp(u);
```

运行结果如下：

```
请输入接收到的码元(矩阵形式 7 位): [1 0 1 0 1 0 1]
纠错后的码元为:
    0    0    1    0    1    0    1

原信息码元为:
    0    0    1    0
```

(2) 利用校正子和校验矩阵的关系译码。

程序 3_6(program3_6. m)

```
% 对接收到的序列进行译码,得到译码后的码字
H = [1 1 1 0 1 0 0; 0 1 1 1 0 1 0; 1 1 0 1 0 0 1];          % 校验矩阵
r = input('请输入接收到的码元(矩阵形式 7 位): ');          % 接收到的码字
S = mod(r * (H'),2);                                        % S 为校正子
E = [1 1 1 1 1 1 1];                                        % 初始化错误图样
% ------------------------------------
for i = 1: 7 % 用该 for 循环取出 H 中的每一列,然后与 S 相加
  T = H(: ,[i]);
   % ------------------------------------
  B1 = S + T';
  B = mod(B1,2);                                            % S 与 H 的第 i 列之和
    if (all(B(: ) == 0)) % 若 S 与 H 的第 i 列之和 B 为零矩阵,则表示 r 中第 i 个码字有误
      fprintf('r 序列中错误码位是第: ');
      disp(i)
    else % 如果 S 与 H 的第 i 列之和 B 不为零矩阵,则表示 r 中第 i 个码字无误。
      E(1,i) = 0;                                           % 错误图样第 i 列为 0
    end;
end;
% ------------------------------------
C = mod((r + E),2);                                         % 求出纠错后码字
fprintf('纠错后的码字应该为: C = ');
disp(C);
% ------------------------------------
% 显示编码前的原码字
u = zeros(1,4);
u = [C(: ,1),C(: ,2),C(: ,3),C(: ,4)]; % 因为是系统码,因此原信息码是编码的前 4 位
```

```
disp('原信息码元为: ');
disp(u);
```

运行结果如下：

```
请输入接收到的码元(矩阵形式 7 位): [1 0 1 0 1 0 1]
r 序列中错误码位是第:     3
纠错后的码字应该为: C =     1     0     0     0     1     0     1
原信息码元为:
    1     0     0     0
```

注：以上两种译码方法对于同样的接收码字[1 0 1 0 1 0 1]，得到的信息码字分别是[0 0 1 0]和[1 0 0 0]，不同的原因是两种编码使用的生成矩阵不同(校验矩阵不同)。

3.8.4 通信过程的编程仿真

纠错编码的目的是提高通信的可靠性，而衡量可靠性的一个重要指标是误码率，误码率越小，系统的可靠性越高。下面就利用 MATLAB 对通信过程进行仿真，以此来说明线性分组码对通信系统性能的改善。

1. 采用 encode、decode 函数进行编译码

程序 3_7(program3_7.m)

```
% (7,4)汉明编码性能(纠错后还原为信息序列,与原信息序列进行比较)
bits = 100000;                                  % 符号数
msg = randi([0,1],bits,1);                      % 随机产生的信息序列
% ------------------------------------
SNR = 0: 1: 12;                                 % 信噪比
L = length(SNR);
BER1 = zeros(1,L);
BER2 = zeros(1,L);
% ------------------------------------
modbit1 = pskmod(msg,2);                        % PSK 调制
% ------------------------------------
for k = 1: L % 未编码的序列,调制后经过高斯白噪声信道,再解调制,求误码
    y1 = awgn(modbit1,SNR(k),'measured');       % 在传输序列中加入 AWGN 噪声
    demmsg1 = pskdemod(y1,2);                   % PSK 解调
    recode = reshape(demmsg1',1,[]);            % 重新构造为行矩阵
    error1 = (recode~ = msg');
    errornum = sum(error1);
    BER1(k) = errornum/length(msg);
end
% ------------------------------------
code = encode(msg,7,4,'hamming');               % (7,4)汉明编码
modbit2 = pskmod(code,2);                       % PSK 调制
% ------------------------------------
for k = 1: L % 编码的序列,调制后经过高斯白噪声信道,再解调制,再纠错后求误码
    y2 = awgn(modbit2,SNR(k),'measured');       % 在传输序列中加入 AWGN 噪声
    demmsg2 = pskdemod(y2,2);                   % PSK 解调
    recode = reshape(demmsg2',1,[]);
    bitdecoded = decode(recode,7,4,'hamming');  % 汉明译码
    % ------------------------------------
```

```
    %计算误码率
    error2 = (bitdecoded~ = msg');
    errorbits = sum(error2);
    BER2(k) = errorbits/length(msg);
end
% ------------------------------------
semilogy(SNR,(BER1),'b- * ')                    %画图
hold on
semilogy(SNR,(BER2),'r-o')
grid on
legend('未编码','(7,4)汉明编码');
xlabel('SNR/dB');
ylabel('BER');
title('(7,4)汉明编码性能');
```

运行结果如图 3-5 所示。

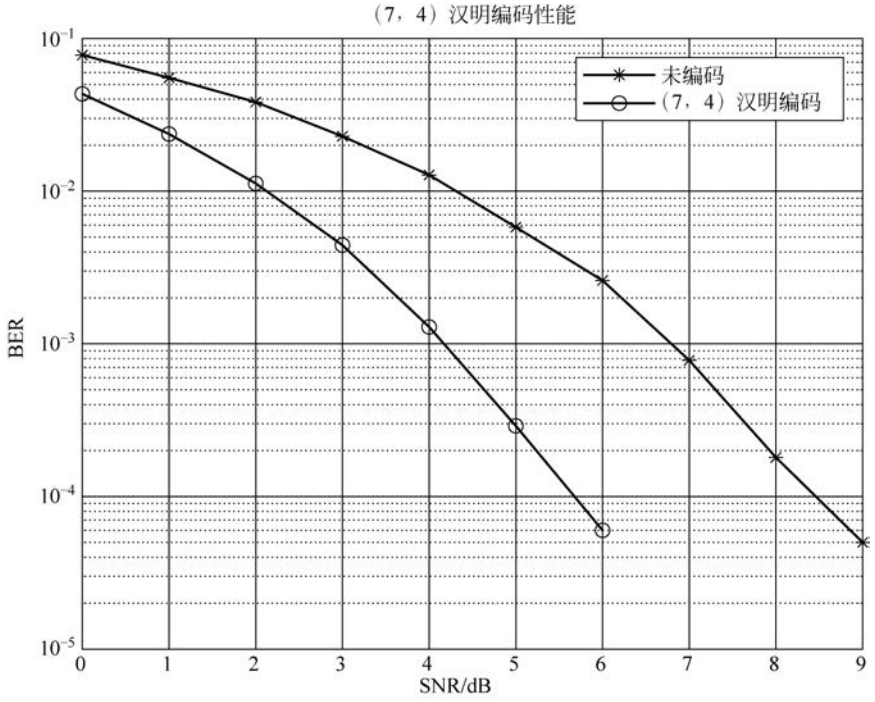


图 3-5　(7,4)汉明编码性能

由图 3-5 可见，采用(7,4)汉明编码后，通信系统的误码率大大降低。误码率的大小与多种因素有关，如编码方法、信道噪声、调制方式等，改变编码过程中相应的参数，则误码率也跟着发生改变。

2. 编程实现编译码

程序 3_8(program3_8.m)

```
%(7,4)汉明编码性能(包含编码和译码的过程)
```

```
bits = 100000;                                          % 符号数
msg = randi([0,1],bits,1);                              % 随机产生的信息序列
% ------------------------------------
G = [1 1 1 1 0 0 0; 1 0 1 0 1 0 0; 0 1 1 0 0 1 0; 1 1 0 0 0 0 1];   % 生成矩阵
H = [1 0 0 1 1 0 1; 0 1 0 1 0 1 1; 0 0 1 1 1 1 0];      % 监督矩阵
Et =  [0 0 0 0 0 0 0;
       0 0 0 0 0 0 1;
       0 0 0 0 0 1 0;
       0 0 0 0 1 0 0;
       0 0 0 1 0 0 0;
       0 0 1 0 0 0 0;
       0 1 0 0 0 0 0;
       1 0 0 0 0 0 0];                                  % 错误图样
Sm = Et * H';                                           % 对应的伴随式
% ------------------------------------
SNR = 0: 1: 12;                                         % 信噪比
L = length(SNR)
BER1 = zeros(1,L);
BER2 = zeros(1,L);
% ------------------------------------
modbit1 = pskmod(msg,2);                                % 调制
for k = 1: L % 未编码的序列,调制后经过高斯白噪声信道,再解调制,求误码
    y1 = awgn(modbit1,SNR(k),'measured');               % 在传输序列中加入 AWGN 噪声
    demmsg1 = pskdemod(y1,2);                           % 解调
    recode = reshape(demmsg1',1,[]);
    error1 = (recode~ = msg');
    errornum = sum(error1);
    BER1(k) = errornum/length(msg);
end
% ------------------------------------
% 编码
A = vec2mat(msg,4);                                     % 序列转换为矩阵
U = A * G;
U = mod(U,2);
bitcoded = reshape(U',1,[]);
% ------------------------------------
modbit2 = pskmod(bitcoded,2);                           % 调制
% ------------------------------------
for k = 1: L % 编码的序列,调制后经过高斯白噪声信道,再解调制,再纠错后求误码
    y2 = awgn(modbit2,SNR(k),'measured');               % 在传输序列中加入 AWGN 噪声
    demmsg2 = pskdemod(y2,2);                           % 解调
    recode = reshape(demmsg2',1,[]);
    % ------------------------------------
    % 译码
        row = length(recode)/7;                         % 行数
        E = zeros(row,7);                               % 错误图样
        RM = zeros(row,7);                              % 纠错之后的矩阵
        R = vec2mat(recode,7);
        S = R * H';                                     % 伴随矩阵
        S = mod(S,2);
        RU = zeros(row,4);
        for i = 1: row
            for j = 1: 2^(7 - 4)                        % 查表纠错
                if(S(i,: ) == Sm(j,: ))
```

```
                    E(i,:) = Et(j,:);
                    RM(i,:) = R(i,:) + E(i,:);
                    RM(i,:) = mod(RM(i,:),2);
% 根据生成矩阵知码的后 4 位是信息码
                    RU(i,:) = [RM(i,4),RM(i,5),RM(i,6),RM(i,7)];      % 得到原信息码
                    break;
                end
            end
        end
        bitdecoded = reshape(RU',1,[]);                               % 转换为比特流
    % ------------------------------------
    % 计算误码率
    error2 = (bitdecoded~ = msg');
    errorbits = sum(error2);
    BER2(k) = errorbits/length(msg);
end
% ------------------------------------
% 画图
semilogy(SNR,(BER1),'b- * ')
hold on
semilogy(SNR,(BER2),'r-o')
grid on
legend('未编码',(7,4)编码)
xlabel('SNR/dB');
ylabel('BER');
title('(7,4)编码性能');
```

运行结果如图 3-6 所示。

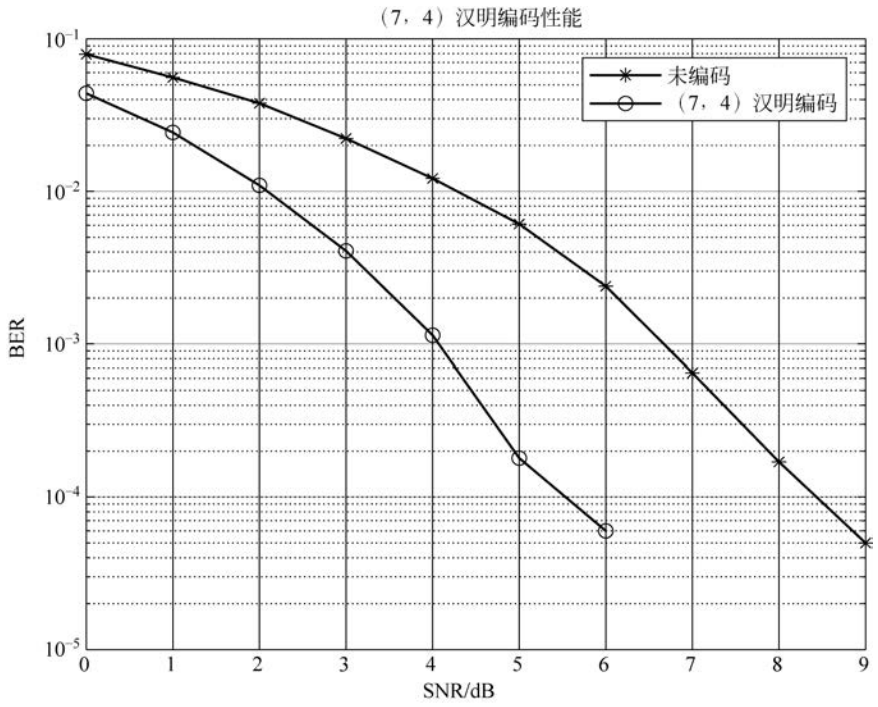


图 3-6 (7,4)汉明编码性能

比较图 3-5 和图 3-6 可知，这两个仿真图基本上是一模一样的，因为二者仿真条件相同，采用的都是(7,4)码，因此结果一致。不同之处是程序 3_7 隐藏了编译码的具体过程，由编译码函数实现，而程序 3_8 将具体的编译码过程写在了程序中。

3.8.5 Simulink 仿真

Simulink 是实现动态系统建模、仿真的一个集成环境。它的存在使 MATLAB 的功能得到进一步的扩展。这种扩展的意义表现在以下 3 方面：

(1) 实现了可视化建模，用户通过简单的鼠标操作就可建立起直观的系统模型，并进行仿真；

(2) 实现了多工作环境间文件互用和数据交换；

(3) 把理论研究和工程实现有机地结合在一起。

Simulink 为用户提供了用方框图进行建模的图形接口，具有直观、方便、灵活的优点。下面就采用 Simulink 对通信系统的性能进行仿真。

1. 二进制对称信道中采用线性分组码或汉明码

(1) 利用 Display 模块显示仿真结果。

在二进制对称信道(BSC)中，采用线性分组码，系统仿真模型(fzm_bsc_1.slx)如图 3-7 所示。

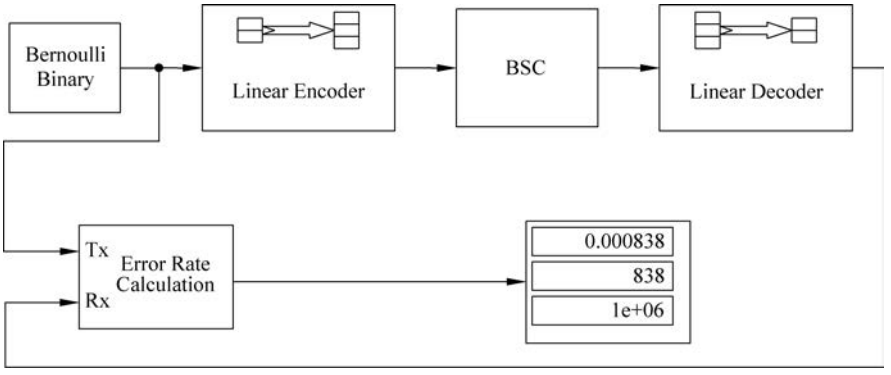


图 3-7 BSC 信道线性分组码系统仿真模型 1


fzm_bsc_1_V

图 3-7 中各模块的参数设置如图 3-8～图 3-13 所示。

图 3-7 中信号源是伯努利随机二进制信号发生器，产生采样时间为 1 的二进制信号，信息中“0”“1”等价出现，传输环境是差错率为 0.01 的二进制对称信道。

在发射端和接收端分别设置线性分组码编码器和线性分组码译码器，线性分组码的生成矩阵为

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

接收端还设置了差错率计算模块，并利用 Display 模块显示仿真结果。

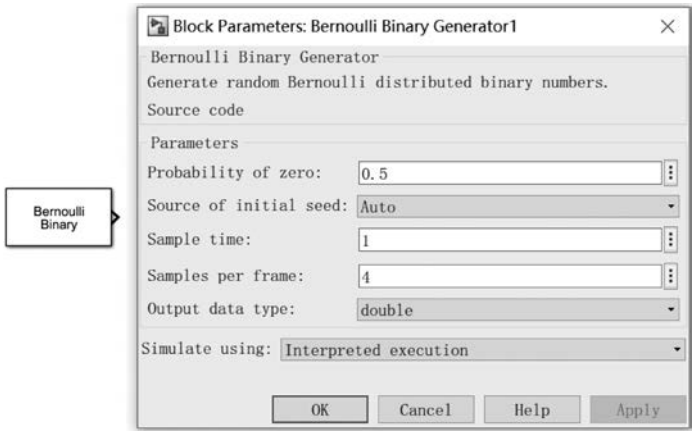


图 3-8　Bernoulli Binary Generator1 模块参数

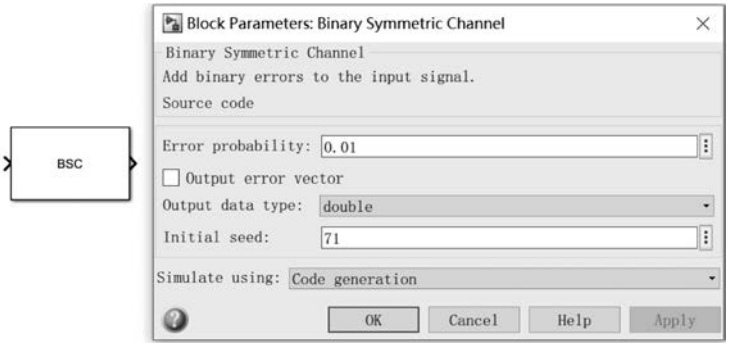


图 3-9　BSC 模块参数

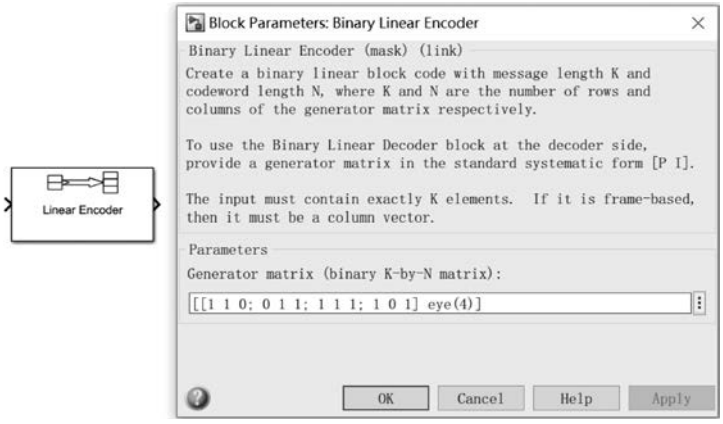


图 3-10　Binary Linear Encoder 模块参数

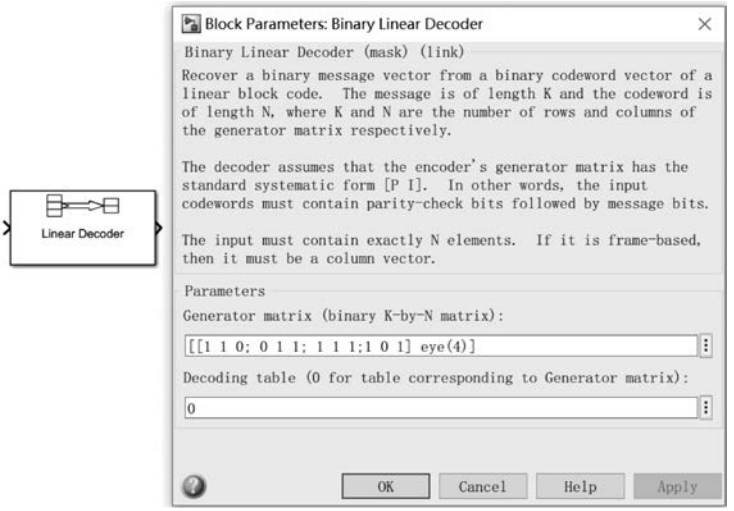


图 3-11 Binary Linear Decoder 模块参数

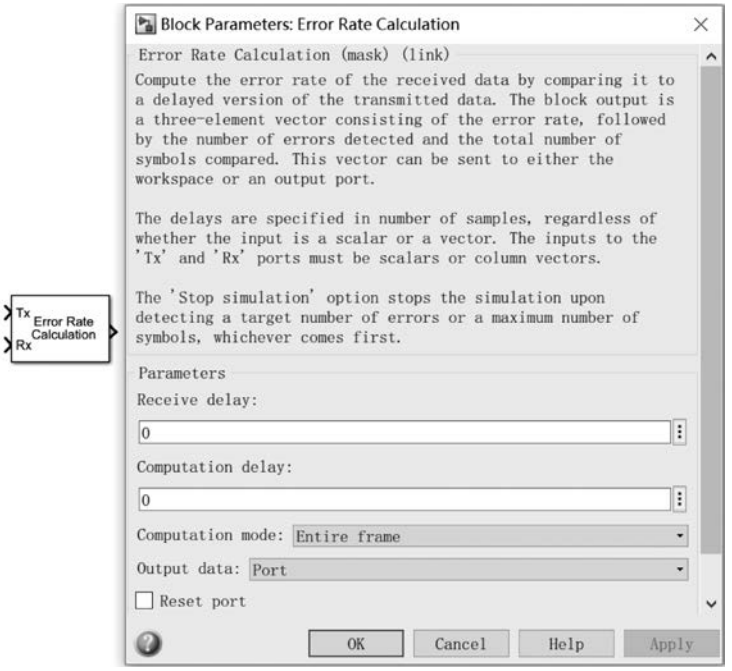


图 3-12 Error Rate Calculation 模块参数

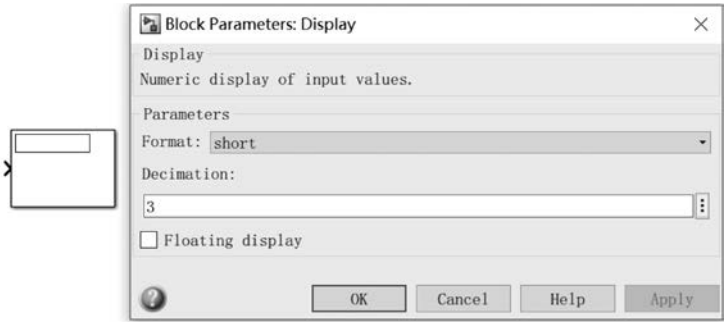


图 3-13 Display 模块参数

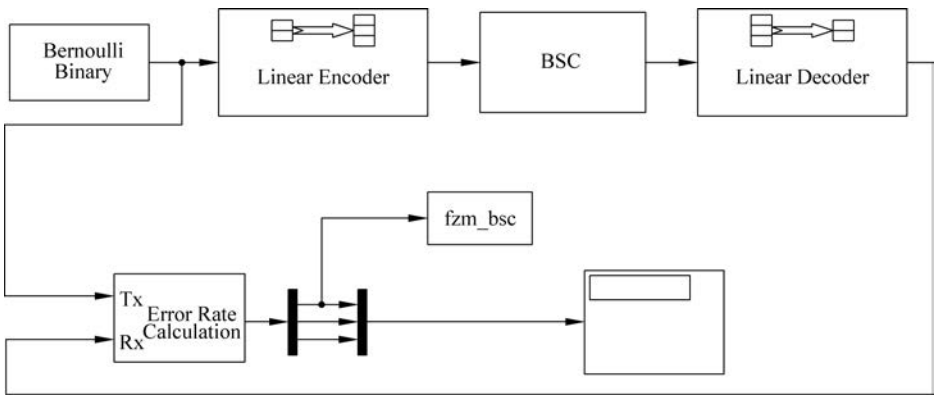
设置仿真结束时间(Simulation stop time)为 1000000,运行图 3-7 的仿真模型,结果由显示器(Display)给出。显示器的第一行显示误码率,第二行显示错误符号数,第三行显示总符号数。如果需要缩短仿真时间,则 Simulation stop time 值可以适当减小,但对于求误码率来说,取值不要小于 10000。另外,改变此参数后仿真结果会有微小变化。

虽然因为信道编码的结果使得传输效率变为 4/7,即发送 7 个码元中仅传递了 4 个码元的有效信息,但使得差错率由 0.01 降为 0.000838。由仿真结果可见,采用线性分组码编码后,通信系统的性能有所改善。

注:图 3-8 和图 3-9 中都有参数 Initial seed,是产生随机数的初始种子。Initial seed 的值可以任意设置,若设置参数改变,产生的随机数跟着改变。对于数量巨大的信息序列来说,不同的序列对仿真的结果几乎没有影响,图 3-8 采用了 Auto 值。图 3-9 中的 Initial seed 设置对仿真结果还是有一定的影响,但影响甚微。

(2) 利用曲线显示仿真结果。

图 3-7 仿真模型中利用 Display 模块显示仿真结果,只能看到当信道差错概率取某个值时的信号误码率。如果想要得到线性分组码的信号误码率与 BSC 信道差错概率之间的关系曲线,则系统仿真模型如图 3-14(fzm_bsc_2.slx)所示。



fzm_bsc_2_V

图 3-14 BSC 信道线性分组码系统仿真模型 2

图 3-14 中主要模块的参数设置如图 3-15 和图 3-16 所示。其余模块参数设置同前。

运行程序 3_9 就可以得到二进制对称信道差错概率与编码后错误率的关系。

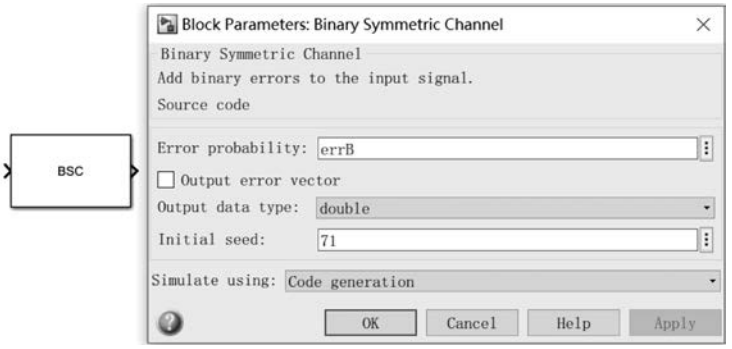


图 3-15　BSC 模块参数

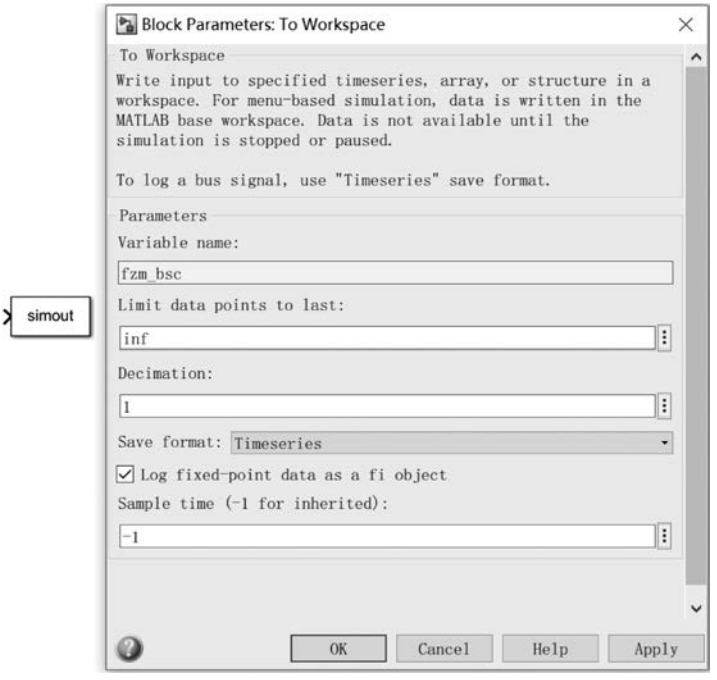


图 3-16　To Workspace 模块参数

程序 3_9（program3_9.m）

```
%二进制对称信道差错概率与编码后错误率的关系
er = 0:0.005:0.05;%二进制对称信道信道差错概率的取值范围
% ------------------------------------------------------------------------
%利用循环语句求错误率
for n = 1:length(er)
    errB = er(n);
    sim('fzm_bsc_2');                    %fzm_bsc_2 是被调用的仿真模型名
    S(n) = [mean(fzm_bsc)]';             % fzm_bsc 是 to Workspace 模块变量名
    EN(n) = [er(n)]';
end
```

```
% ------------------------------------------------------------------------
plot(EN,S,'b- * ');                                    % 画图
grid
xlabel('信道差错概率');
ylabel('编码后错误率');
title('二进制对称信道中分组码性能');
```

运行结果如图 3-17 所示。

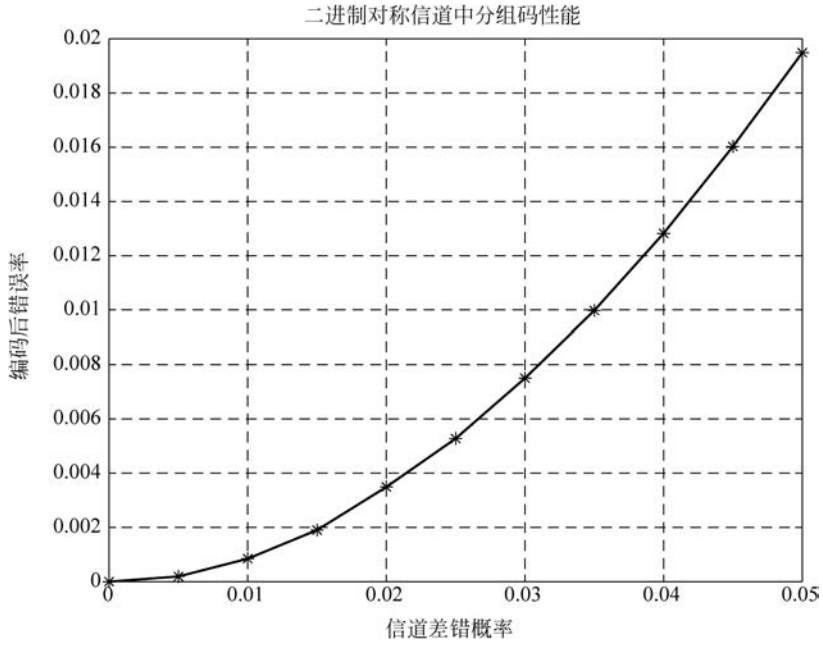


图 3-17　BSC 信道线性分组码性能 1

由图 3-17 可见，在给定信道差错概率的情况下，线性分组码对通信系统的性能有一定的改善。图 3-14 在仿真结束后示波器上显示的数值是信道差错概率取最后一个值时的误码率。

若改变程序 3_9 中二进制对称信道差错概率的取值范围，则运行结果相应地发生改变。

将程序 3_9 中的语句“er＝0:0.005:0.05;”改为“er＝0:0.05:1;”即为程序 3_10 (program3_10.m)(程序略)，运行结果如图 3-18 所示。

由图 3-18 可见，尽管采用了线性分组码，但是当信道的差错概率不同时，系统的性能也不同，这是因为误码性能不但与信道的差错概率有关，与编码方式有关，也与译码方式有关。其实图 3-17 是图 3-18 中信道差错概率在 0～0.05 曲线放大。图 3-18 也说明，当信道差错概率在 0.1～0.9 时，编码几乎不起任何作用。当信道差错概率为 0.5 时，编码后错误率仍是 0.5，此时没有办法判别接收信息的错与对，所以任何编码方式都无能为力。一般情况下，二元对称信道的正确传递概率远大于错误传递概率，仿真结果如图 3-17 所示。在后面其他的仿真程序中信道的差错概率的取值也同程序 3_9。

注：*程序文件和模型文件必须在同一个文件夹中。*

(3) 编码和未编码的性能比较。

为了清楚、直观地表明编码对通信系统性能的改善，我们可以将编码前后的系统性能进

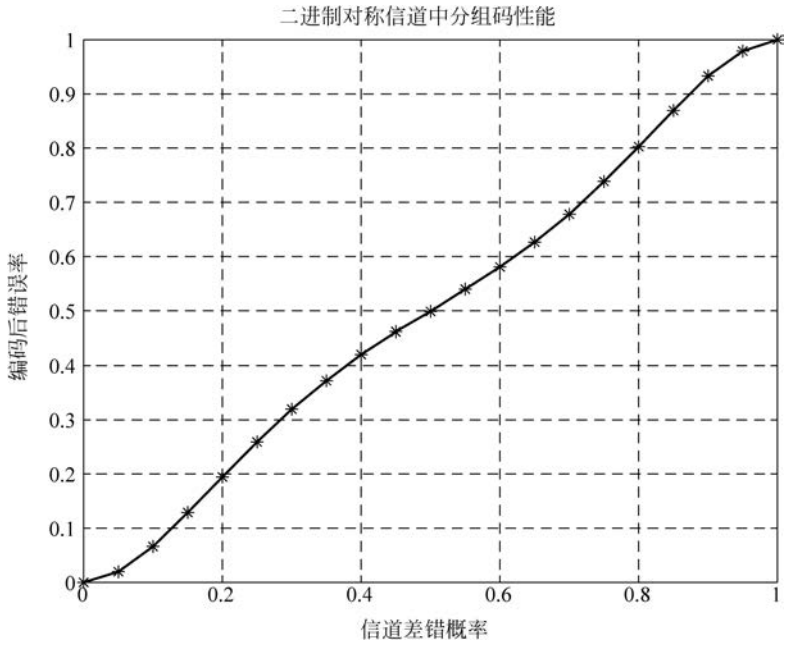


图 3-18　BSC 信道线性分组码性能 2

行比较。系统仿真模型如图 3-19 所示(hm_bsc_bj.slx)。

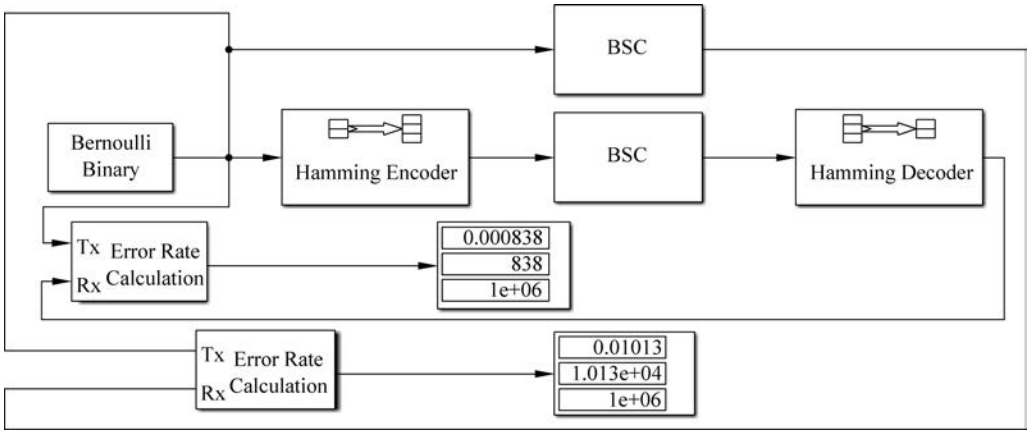


图 3-19　BSC 信道汉明码编码前后比较


hm_bsc_bj_V

在 BSC 信道中,将信道的错误符号概率设为 0.01,采用(7,4)汉明编码。由运行结果图 3-19 可以看出,编码后的错误率为 0.000838;未编码的错误率为 0.01013。编码后误码率大约下降到了一个量级。

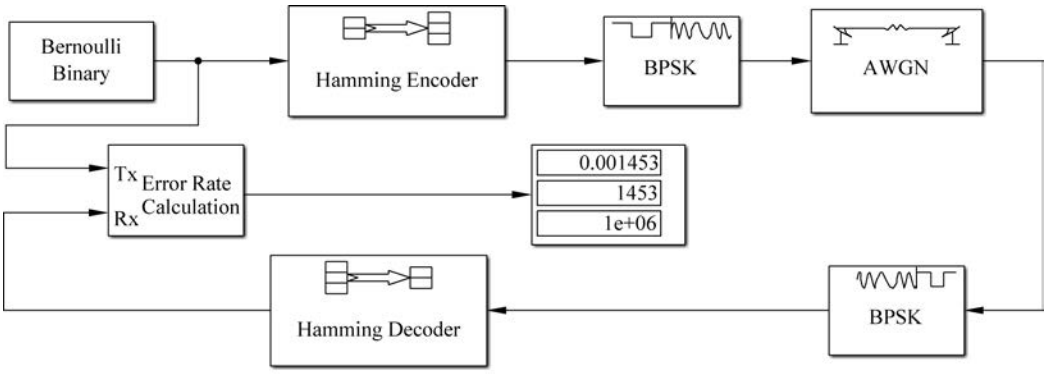
如果采用其他的编码方式,改变相应的模块或只改变模块的参数即可。

2. 高斯白噪声信道中采用汉明码

(1) 利用 Display 模块显示仿真结果。

考虑到实际信道几乎不可能是 BSC 信道,所以下面在仿真中采用跟实际信道模型更接

近的高斯白噪声信道(AWGN)。采用 AWGN 信道时,在通信系统的发送端必须增加调制模块,同时在接收端必须增加解调模块,如图 3-20 所示(hm_awgn_1.slx)。



hm_awgn_1_V

图 3-20 AWGN 汉明码系统仿真模型 1

图 3-20 仿真模型中增加了 BPSK 基带调制和解调两个模块,采用 AWGN 信道。AWGN 信道参数设置如图 3-21 所示。

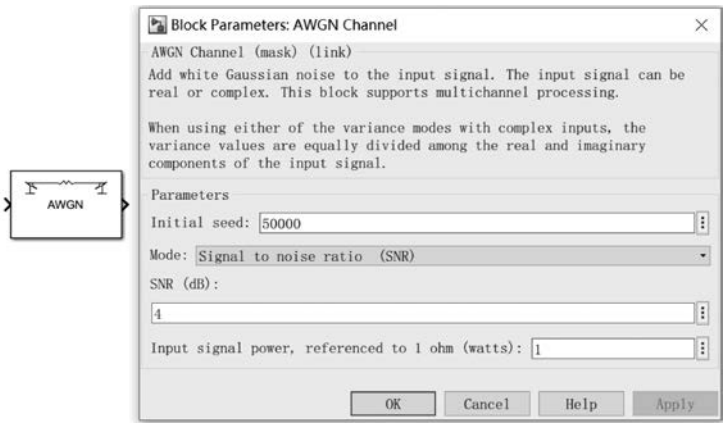


图 3-21 AWGN Channel 模块参数

仿真模型采用(7,4)汉明编码,AWGN 信道的信噪比设置为 4,Initial seed 为 50000 运行后的误码率为 0.001453,见图 3-20。只改变 Initial seed 的设置,也会影响运行结果,但影响甚微。

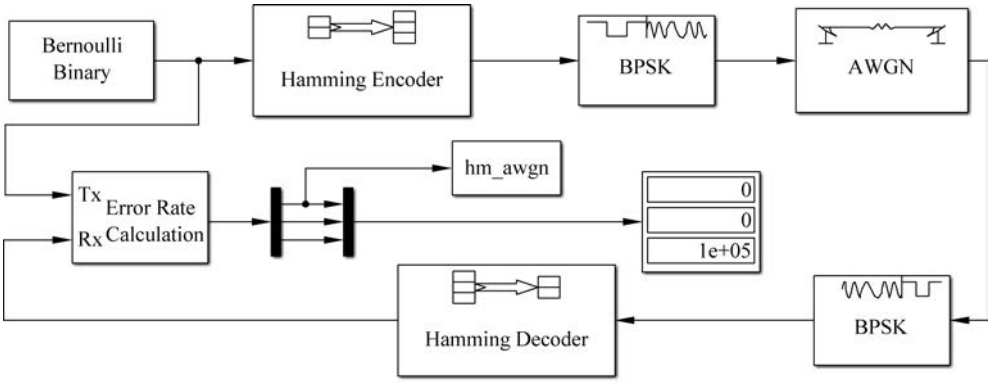
(2) 利用曲线显示仿真结果。

如果想要得到汉明编码的误码率与信道信噪比之间的关系,则仿真模型如图 3-22(hm_awgn_2.slx)所示。

图 3-22 中主要模块的参数设置如图 3-23 和图 3-24 所示。

程序 3_11(program3_11.m)

```
%高斯白噪声信道汉明码性能
SNR = -1:1:8; %信噪比
% ------------------------------------------------------------------------
for n = 1:length(SNR) %误码率计算
    snr = SNR(n);
```



hm_awgn_2_V

图 3-22　AWGN 汉明码系统仿真模型 2

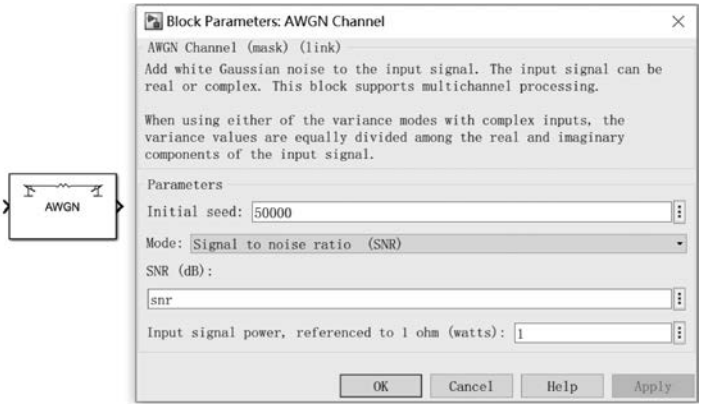


图 3-23　AWGN Channel 模块参数

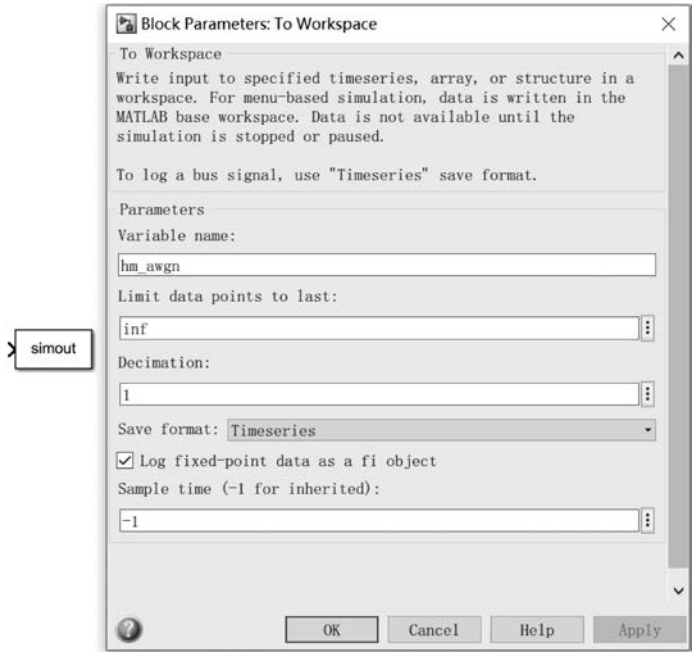


图 3-24　To Workspace 模块参数

```
    sim('hm_awgn_2')
    S2(n) = [mean(hm_awgn)]';
    S3(n) = S2(n) + eps;
    EN(n) = [SNR(n)]';
end
% ------------------------------------------------------------------------
semilogy(EN,S3,'b-*') % 画图
axis([-1,8,1e-5,1]);grid
xlabel('高斯白噪声信道信噪比 SNR(dB)');
ylabel('误码率');
title('高斯白噪声信道汉明码性能');
```

运行程序 program3_11.m，结果如图 3-25 所示。

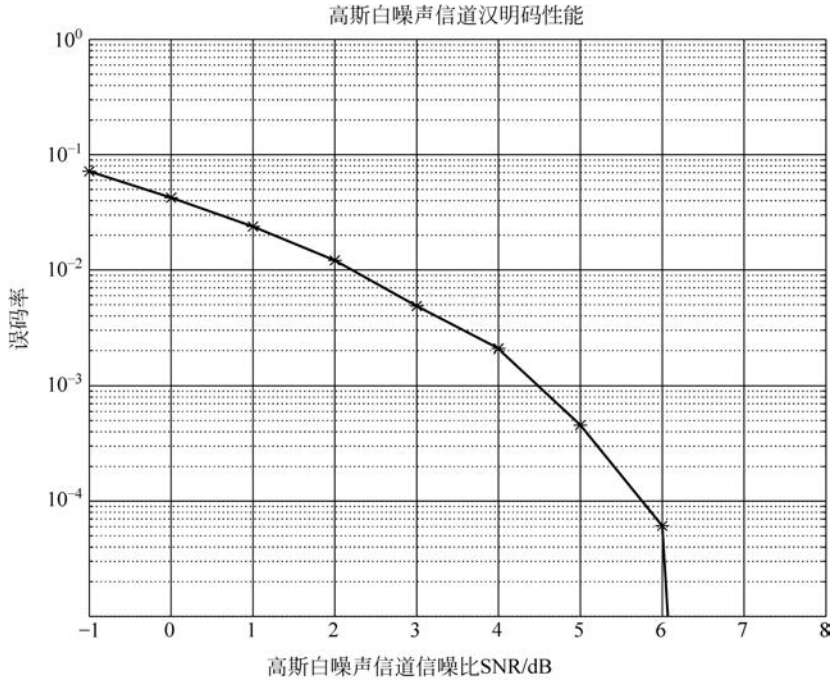


图 3-25　AWGN 信道汉明码性能

由图 3-25 可以看出，误码率随着 AWGN 信道信噪比的增大而降低。仿真图 3-22 中显示的数值是信道信噪比取最后一个值时的误码率。

(3) 编码和未编码的性能比较。

为了清楚、直观地说明高斯白噪声信道中汉明编码对通信系统性能的改善，可以将编码前后的系统性能进行比较。系统仿真模型如图 3-26 所示(hm_awgn_bj_1.slx)。

图 3-26 中，两个 AWGN 信道的信噪比均设置为 4，采用(7,4)汉明编码。由运行结果图 3-26 可以看出，未编码的误码率为 0.01252，而编码后的误码率为 0.001453，采用编码后误码率大约下降了一个量级。

如果要考虑不同信噪比情况下编码对系统性能的改善，仿真模型如图 3-27 所示(hm_awgn_bj_2.slx)。

hm_awgn_bj_1_V

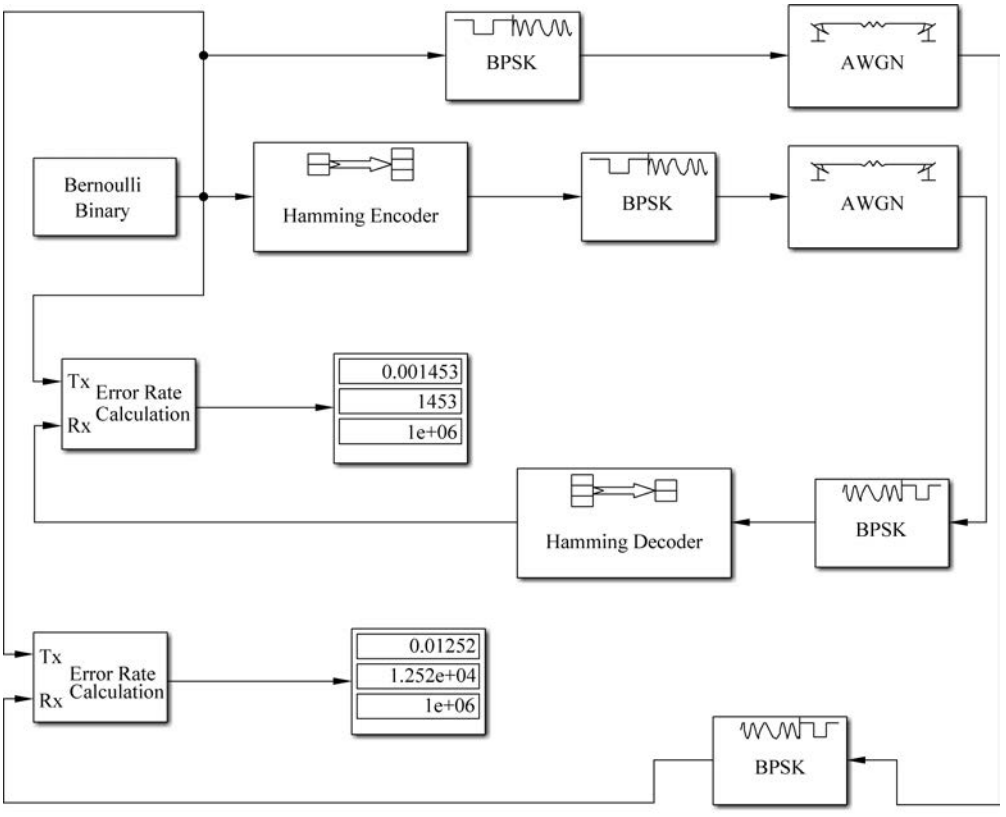


图 3-26　AWGN 信道汉明码编码前后比较 1

hm_awgn_bj_2

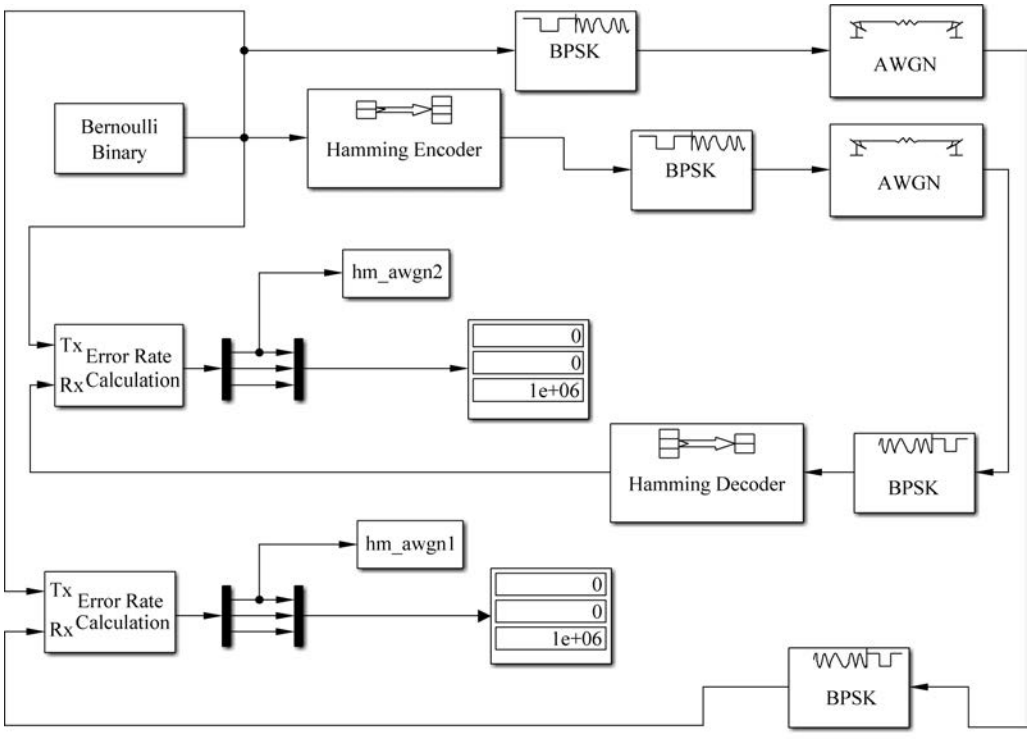


图 3-27　AWGN 信道汉明码编码前后比较 2

程序 3_12（program3_12. m）

```
% 高斯白噪声信道汉明码性能
SNR = -1:1:12; % 信噪比
% ------------------------------------------------------------------------
for n = 1:length(SNR) % 误码率计算
    snr1 = SNR(n);
    snr2 = SNR(n);
    sim('hm_awgn_bj_2')
    S21(n) = [mean(hm_awgn1)]';
    S22(n) = [mean(hm_awgn2)]';
    S31(n) = S21(n) + eps;
    S32(n) = S22(n) + eps;
    EN(n) = [SNR(n)]';
end
% ------------------------------------------------------------------------
semilogy(EN,S31,'b-*',EN,S32,'r-o'); % 画图
axis([-1,12,1e-5,1]);grid
legend('未编码','(7,4)汉明编码');
xlabel('高斯白噪声信道信噪比 SNR(dB)');
ylabel('误码率');
title('高斯白噪声信道汉明码性能');
```

运行程序 program3_12. m，结果如图 3-28 所示。

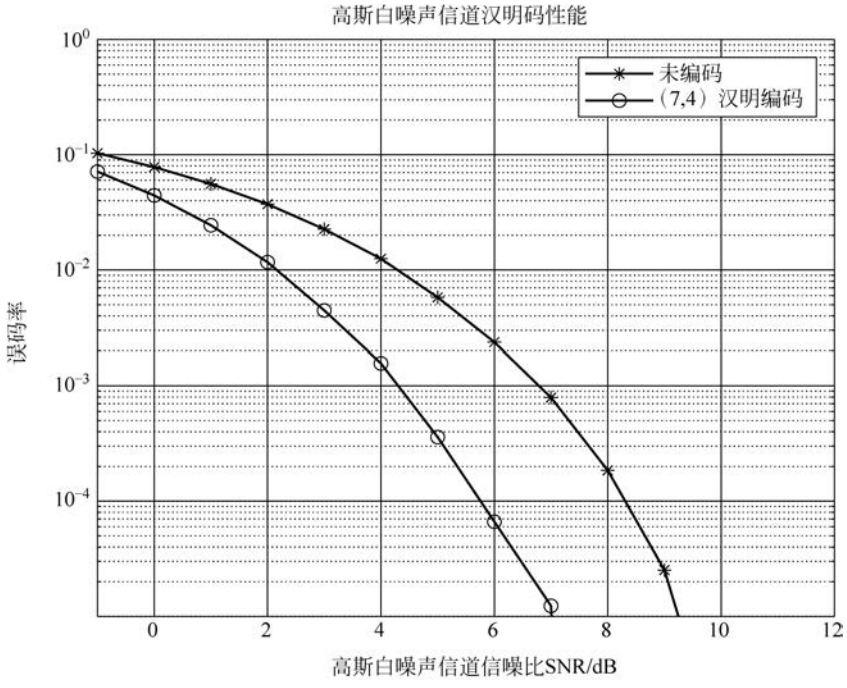


图 3-28　AWGN 信道汉明编码前后比较

由图 3-28 可见，汉明编码对系统的性能有明显的改善。如果采用其他的编码方式，改变相应的模块，或只改变模块的参数即可。

另外，图 3-28 和图 3-5 都是汉明码对高斯白噪声信道的通信性能的改善，两个图也基本上是一致的，只是二者采用的仿真方法不同。

3.8.6 线性分组码电路仿真

对于图 3-2 的(7,3)线性分组码的编码电路，其对应的 Simulink 仿真模型如图 3-29 所示(xxfzmdl.slx)。


xxfzmdl_V

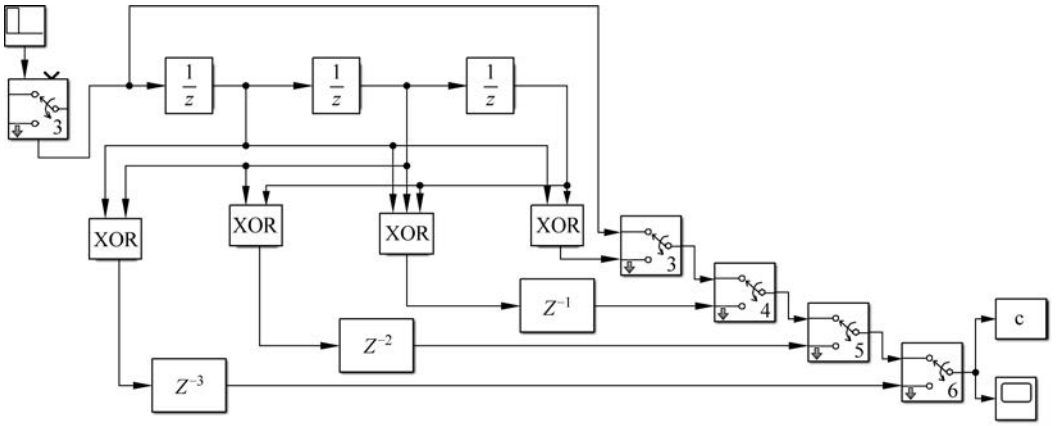


图 3-29 (7,3)线性分组码的编码电路

图 3-29 中采用了多个 N-Sample Switch 开关模块，输入端开关参数设置如图 3-30 所示，其余略。移位寄存器采用 Unit Delay 模块，模 2 加法器采用 Logical Operator 模块，用 Scope 模块和 To Workspace 模块来查看编码序列。

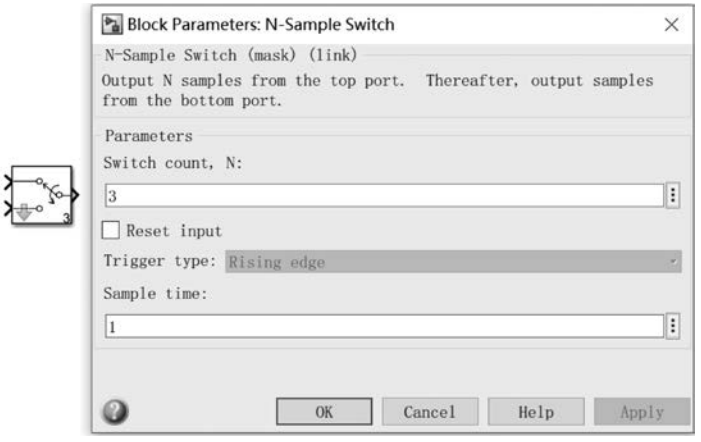


图 3-30 输入端的开关设置

输入信息序列采用 Repeating Sequence Stair 模块，目的是为了得到确定的输入信息，如这里输入信息序列为 100，则参数设置如图 3-31 所示。

运行仿真模型，查看示波器的显示和 $\boldsymbol{c}$，可以看出编码结果为 1001110，改变输入的信息序列，可以得到相应的编码结果，最终的编码结果与表 3-5 一致。

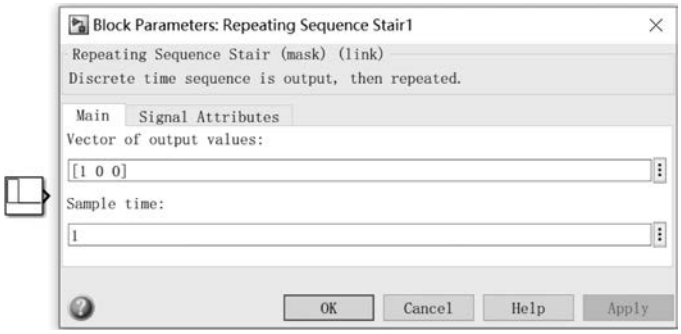


图 3-31　Repeating Sequence Stair1 模块参数

注：此处的仿真模型只是针对前面讲的线性分组码编码电路的原理进行仿真，与实际通信系统的编码还是有区别的。

习题

1. 某分组码的校验矩阵 $\boldsymbol{H}=\begin{bmatrix}0&1&1&1&0&0\\1&0&1&0&1&0\\1&1&0&0&0&1\end{bmatrix}$，求：

(1) $n=?$ $k=?$ 该码的码字有多少？

(2) 该码的生成矩阵。

(3) 矢量 010111 和 100011 是否是码字？

2. 某二元(n,k)系统线性分组码的全部码字为 00000，01011，10110，11101，求：

(1) $n=?$ $k=?$

(2) 码的生成矩阵 $\boldsymbol{G}$ 和校验矩阵 $\boldsymbol{H}$。

3. 已知一个线性分组码的校验矩阵 $\boldsymbol{H}=\begin{bmatrix}0&1&1&1&1&0&0\\1&0&1&1&0&1&0\\1&1&0&1&0&0&1\end{bmatrix}$，试求其生成矩阵 $\boldsymbol{G}$。当输入信息序列为 1001，1100，1101 时，求编码器输出的码字序列。

4. 某(5,2)线性分组码的校验矩阵 $\boldsymbol{H}=\begin{bmatrix}1&1&1&0&0\\1&0&0&1&0\\1&1&0&0&1\end{bmatrix}$，求：

(1) 该码的 $\boldsymbol{G}$ 矩阵。

(2) 码的标准阵列。

(3) 码的简化译码表。

(4) 该码是否是完备码？

5. 试构造 GF(2)上的(15,11)汉明码，求出其系统码形式的 $\boldsymbol{H}$ 矩阵和 $\boldsymbol{G}$ 矩阵。

6. 设一个(7,4)分组码的生成矩阵 $\boldsymbol{G}=\begin{bmatrix}1&0&0&0&1&1&1\\0&1&0&0&1&0&1\\0&0&1&0&0&1&1\\0&0&0&1&1&1&0\end{bmatrix}$，求：

(1) 该码的全部码字。

(2) 码的标准阵列。

(3) 码的简化译码表。

7. 信源的 4 个消息$\{a_1,a_2,a_3,a_4\}$被编成 4 个码长为 5 的二元码字 00000,01101,10111,11010 发送。

(1) 试给出码的生成矩阵 $\boldsymbol{G}$ 和校验矩阵 $\boldsymbol{H}$；

(2) 若上述码字通过固有误码率为 $p<0.5$ 的二进制对称信道传输，请列出其标准矩阵，并计算错误概率 P_e。

8. 设二元(6,3)的生成矩阵 $\boldsymbol{G}=\begin{bmatrix}1&0&0&0&1&1\\0&1&0&0&0&1\\0&0&1&1&1&0\end{bmatrix}$，给出它的校验矩阵 $\boldsymbol{H}$，并计算该码的最小距离。

9. 一个(8,4)系统码的编码规则如下：

$$c_0=u_0,c_1=u_1,c_2=u_2,c_3=u_3,c_4=u_1\oplus u_2\oplus u_3$$

$$c_5=u_0\oplus u_1\oplus u_2,c_6=u_0\oplus u_1\oplus u_3,c_7=u_0\oplus u_2\oplus u_3$$

(1) 写出此码的生成矩阵 $\boldsymbol{G}$ 和校验矩阵 $\boldsymbol{H}$；

(2) 计算该码的最小距离 d。

10. 设有一个长度为 $n=15$ 的汉明码，试问其监督位 r 应该等于多少？其码率等于多少？其最小码距等于多少？试写出其监督位和信息位之间的关系。

11. 利用 MATLAB 编程实现(3,1)汉明编码。

12. 利用 Simulink 搭建仿真模型，得到(15,11)汉明码在 AWGN 信道信噪比为 3 时的误码率。