**第 13 次课 学时 2**

|  |  |
| --- | --- |
| **授课章节内容** | 第八章 有噪信道编码 |
|  | 8.3 线性分组码 |
| **教学目标** | 教学目标5 |
| **支撑毕业要求** | 毕业要求2-1 |

**教学要求：**

知识目标

* 掌握线性分组码概念。
* 掌握生成矩阵与监督矩阵

1. 能力目标

* 具备给定纠错能力条件下设计线性分组码的能力。

1. 素质目标

* 激发学生对电子信息相关专业课程的学习热情，通过对线性分码的学习，扩展提高通信可靠性途径更加广阔的思维。

**教学重点与难点**：

* 生成矩阵与监督矩阵

**教学过程设计：**

课前预习，观看视频8.6，通过线性方程组引入线性分分组码的生成矩阵与监督矩阵

| **讲授与指导内容** | | **讲课、互动内容设计** |
| --- | --- | --- |
| 8.3 线性分组码  奇偶校验码属于线性分组码，它可以由前面提到的码形式表示。编码器将一个*k*比特信息分组（信息矢量）转变成一个更长的由给定元素符号集组成的*n*比特编码分组（编码矢量）。当这个符号集包含2个元素（0和1）时，与二进制数字相应，称为二进制编码。如不加说明，下面所指的线性分组码均指二进制编码。  *k*比特信息形成个不同的信息序列，称为*k*元组（*k*比特序列），同样，*n*比特可以形成个不同的序列，称为*n*元组。编码过程就是将每个*k*元组（共个）映射到个*n*元组中的一个。分组码是一一对应的编码，即个*k*元组唯一映射到个*n*元组，映射可以通过一个查询表实现。  8.3.1 矢量空间  所有的二进制*n*元组组成的集合，称为0和1的二进制域的矢量空间（vector space）。这个二进制域有两个运算“加”和“乘”，所有运算的结果都在包含这两个元素的同一集合中。算数运算加与乘都由代数域中的规定定义。例如，在一个二进制域中，加与乘的规则如下：    加操作用符号表示，它是模2运算。二进制*n*元组的“和”通常是模2加，但是，为了简单起见，一般用普通的“+”号。  8.3.2 矢量子空间  矢量空间的一个子集*S*如果满足以下两个条件，则称为子空间（subspace）：   1. 全0矢量在*S*中。 2. *S*中任意两个矢量的和也在*S*中（即满足封闭性）   这些性质是线性分组码（linear block code）的基本性质。假设和是二进制分组码中的两个码字（或码矢量），当且仅当也是一个码矢量时，这个码才是线性的。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。  例如，矢量空间总共由下面的个4元组组成：    由形成的一个子集如下：    容易证明，任意两个矢量的和只能得到子集中的其他成员。一个由个*n*元组组成的集合称为线性分组码，当且仅当它是所有*n*元组的矢量空间的一个子集。图8.4用一个简单的几何结构类比线性分组码的结构。假设矢量空间包含个个*n*元组，矢量空间中的个*n*元组的子集组成一个子空间。这个矢量或点“稀疏分布”在大量的个点中，表示合法和许用码字（legitimate or allowable codeword）。信息被编码为个许用码矢量中的一个，然后发送。由于信道中存在噪声，接收到的可能是一个受干扰的码字。如果这个受干扰的码字与有效的码字相差不是很远，译码器可以将信息正确译码。选择一个特定编码的基本目标与选择一个调制波形集是相似的，可以根据图8.4表述如下：   1. 用尽可能多的码字构造空间，以争取高的编码效率。也就是说，希望使用很少的冗余度（附加带宽）。 2. 希望码字间的距离越远越好，这样即使矢量在传输中受到干扰，它们仍然能以很高的概率被正确译码。   {DG3BEV$MBX7~)`S(DY4BSM  图8.4 线性分组码结构  8.3.3 （6,3）线性分组码举例  考察这样一个（6,3）码，它有个信息矢量，共有8个码字；在空间中有个6元组。  可以验证，表8.2所示的8个码字形成的一个子空间（包括全0矢量，任意两个码字和形成子空间的另一个码字）。所以，这些码字代表一个线性分组码。现在存在这样一个问题，怎样为（6,3）码指定“码字到信息”间的映射关系？对于给定的码，映射关系并非是唯一的，当然选择也不是任意的。  表8.2 信息和码字间的映射关系   |  |  | | --- | --- | | 信息矢量 | 码字 | | 000 | 000000 | | 100 | 110100 | | 010 | 011010 | | 110 | 101110 | | 001 | 101001 | | 101 | 011101 | | 011 | 110011 | | 111 | 000111 |   8.3.4生成矩阵  当*k*很大时，编码器查询表（table look-up）的实现比较复杂。对一个（127,92）码，有个或个码矢量，如果编码过程由一个简单的查询表组成，容纳如此之多码字所需要的存储器的容量将是巨大的。幸运的是，可以按需随时产生码子而非存储码字来减少复杂性。  由于形成线性分组码的码字集是*n*维二进制矢量空间的一个*k*维子空间（）,因而总能够找到一个*n*元组集合，它包含的矢量个数少于个，但可以用它产生子空间的个码字。产生的矢量集称为扩展子空间。用于扩展子空间的最小线性无关（linary independent）集称为子空间的基（basis），该集合中包含的矢量个数称为该子空间的维数。任意*k*个线性独立*n*元组的一个线性组合，也即，个码字中的任意一个都可以描述为  *U*=*m*1*V*1+ *m*2*V*2+…+ *mkVk*  其中，是消息数字，。  通常可以按如下方法定义一个生成矩阵：  （8.14）  码字矢量通常用行矢量描述。这样，*k*个消息比特的序列*m*表示为一个行矢量：    码字*U*是*m*与***G***的乘积，用矩阵形式表示为  （8.15）  采用以下的规则实现矩阵的相乘：    其中，***A***是矩阵，***B***是矩阵，结果***C***是矩阵。对于前面介绍的例子，可以将生矩阵定义为  （8.16）  其中，是三个线性无关矢量（是8个码矢量的一个子集）。用它们可以产生所有的码矢量。注意，任意两个生成矢量的和不能够产生任何其他的生成矢量（与封闭性相反）。现根据式（8.16）的生成矩阵，表8.2中的第4个消息矢量110生成码字，即    这样，对应于一个消息矢量的码矢量是矩阵***G***的行矢量的线性组合，既然这个码字完全由***G***确定，那么编码器只要存储***G***的*ｋ*个行而非个矢量。对于此例，注意，用式（8.16）的生成矩阵代替表8.2中维原始码字矩阵，带来了系统复杂性的降低。  8.3.5系统线性分组码  系统线性分组码是从*k*维消息矢量到*n*维码字的映射，其中的*k*位与消息位对应，余下的（*n*-*k*）比特是监督比特。系统线性分组码的生成矩阵具有如下形式：  （8.17）  其中，***P***是生成矩阵的监督阵列，或1，是的单位矩阵（主对角线元素为1，其余为零）。注意，对于系统生成矩阵，可以进一步减少复杂性，因为不需要存储单位矩阵。联合式（8.16）和式（8.17），每个码字可以表示为    其中    对于给定消息*k*元组：    和码字矢量*n*元组：    系统编码矢量可以表示为  （8.18）  其中  （8.19）  系统码字有时将消息比特写在左半边，校验比特在右半边。这样并不会影响检错或纠错的性能，后面不再考虑这种情形。  对于（6,3）码，码字可以描述为  （8.20）  式（8.20）提供了一些有关线性分组码的结构信息。可以看到，冗余位可由很多方法产生。第一个检验比特是第一个消息比特和第三消息比特的和；第二个检验比特是第一个消息比特和第二消息比特的和；第三个检验比特是第二个消息比特和第三消息比特的和。由直觉可知，与单校验比特检测或简单的重复信息码元相比，这种结构可以提供更强大的检测和纠错能力。  8.3.6监督矩阵  现在定义一个矩阵***H***，称为监督矩阵（parity-check matrix），它可以有助于将接收矢量进行译码。对应每一个的生成矩阵***G***，存在一个的矩阵***H***，***G***的行与***H***的行正交，即。***W***是***H***的转置矩阵，0是一个的全0矩阵。的行是***H***的列，的列是***H***的行。为了实现系统码的正交性要求，***H***可如下表示：  　　　　　　　　　　　　　　　　（8.21）  因此，  （8.22）  容易证明，乘积满足下式：    其中，监督比特由式（6.19）定义，这样，一旦监督矩阵***H***是根据正交性要求生成的，就可以利用它来检验接收矢量是否是码字集合中的有效码组。当且仅当时，***U***才是由***G***生成的码字。  8.3.7伴随式检验  设是一个接收矢量（个*n*元组中的一个），由传输的（个*n*元组中的一个）产生。可以将写成  （8.23）  其中，是错误矢量，是由信道引入的错误图样，在个*n*元组空间中存在个非零的潜在错误图样。的伴随式（也称校正子）定义为  （8.24）  伴随式是对进行监督校验的结果，它用来确定是否是一个有效码字。如果是有效码字，则***S***的值将为0；如果包含可检测到的错误，伴随式将有非零值；如果包含可纠正的错误，伴随式将有特殊的非零值来指示特定的错误图样。译码器根据要求实现FEC或ARQ，采取措施定位错误并加以纠正（FEC）或请求一个重传（ARQ）。联合式（8.23）和式（8.24），的伴随式表示为  （8.25）  然而，对所有的码矢量都有，所有  （8.26）  前面从式（8.23）开始到式（8.26）的推导，说明不论是受干扰码字矢量还是错误图样，伴随式检验都能得到相同的伴随式。线性分组码的一个重要性质是（也是译码的基础），错误图样与伴随式之间是一一对应的。  注意监督矩阵所必须的两个性质：   1. ***H***中没有全0列，否则的话，相应码字位置上的错误就无法影响伴随式，因而无法检测。 2. ***H***中的所有列是唯一的，如果***H***有两列相同，那么对应这两列发生的错码位置将无法识别。   【**例8.4**】 伴随式检验  假设发送码字*U*=101110，接收矢量为***r***=001110，即最左边一比特接收错误。试求伴随式矢量并证明它等于。  解：    下面证明这个受干扰码字矢量的伴随式与错误图样的伴随式相等：  8.3.8纠错  前面已经检测了单个错误，并说明无论是错误码字，还是导致其错误的错误图样，伴随式检验都能产生相同的伴随式。这样，不仅可以检测错误，而且可以同时纠正错误，因为可纠正的错误图样与伴随式之间呈现一一对应的关系。现在用一个矩阵表示所有可能的个*n*元组的接收矢量，这个矩阵称为标准阵（standard array），第一行以全0码字开始，包括了所有码字，而第一列包括了所有可纠正的错误图样。回忆一下线性码的基本性质，即全0矢量必须是码字集的一个成员。每行称为一个陪集（coset），由第一列的错误图样（称为陪集首）及其干扰的码字组成。（*n*，*k*）码的标准阵形式如下：  （8.27）  注意，码字（全0码字）起两个作用：既是其中一个码字，也是错误图样，代表没有错误，即***r*=*U***。该矩阵包括了空间中所有的个*n*元组。每个*n*元组只出现在一个位置（没有一个缺少，也没有一个重复）。每个陪集包含个*n*元组，因此共有个陪集。  译码机制要求将每个有错的矢量用与此矢量同列的最顶端有效码字代替。假设一个码字通过一个噪声信道，接收矢量为，如果错误图样是一个陪集首，其中，则接收矢量将被正确译码为码字。如果错误图样不是一个陪集首（coset leader），那么将会导致译码错误。   1. 陪集的伴随式   如果是第*j*个陪集首或错误图样，那么是此陪集中的一个*n*元组。这个*n*元组的伴随式可以写为  （8.28）  陪集是“具有相同特征的数字集”。对于任意给定的行（陪集），成员之间存在什么共同点？从式（8.28）可以清楚地看到，陪集中的每一个元素具有相同的伴随式（syndrome）。编码时每个陪集的伴随式是不同的。伴随式用于估计错误图样。   1. 纠错译码   纠错译码的过程如下：  （1）计算的伴随式。  （2）定位错误图样（陪集首），它的伴随式与相等。  （3）假设错误图样是由信道衰落引起的。  （4）错误接收的矢量或码字表示为。通过减去其中已识别出的错误来恢复正确的码字。  模2运算中，减运算与加运算相同。   1. 错误图样的定位   将个6元组如图8.5所示排列成一个标准阵。有效码字是第一行的8个矢量，可纠正的错误图样是第一列的7个非零陪集首。注意，所有1比特错误图样都可以纠正。同时，除去所有1比特的错误图样，还有一些可纠正的错误，因为并没有用完所有64个6元组。还剩一个未分配的陪集首，因而还可以纠正一个附加的错误图样。可以灵活地将此错误图样定为010001。当且仅当信道引起的错误是其中一个陪集首时，译码才是正确的。  现在，通过计算来确定每个可纠正错误序列的伴随式，即     |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | 000000 | 110100 | 011010 | 101110 | 101001 | 011101 | 110011 | 000111 | | 000001 | 110101 | 011011 | 101111 | 101000 | 011100 | 110010 | 000110 | | 000010 | 110110 | 011000 | 101100 | 101011 | 011111 | 110001 | 000101 | | 000100 | 110000 | 011110 | 101010 | 101101 | 011001 | 110111 | 000011 | | 001000 | 111100 | 010010 | 100110 | 100001 | 010101 | 111011 | 001111 | | 010000 | 100100 | 001010 | 111110 | 111001 | 001101 | 100011 | 010111 | | 100000 | 010100 | 111010 | 001110 | 001001 | 111101 | 010011 | 100111 | | 010001 | 100101 | 001011 | 111111 | 111000 | 001100 | 100010 | 010110 |   图8.5 （6，3）码的标准阵示例  结果列于表8.3。由于表中的每个伴随式都是唯一的，译码器可以确定与*e*对应的错误图样。  表8.3 伴随式查询表   |  |  | | --- | --- | | 错误图样 | 伴随式 | | 000000 | 000 | | 000001 | 101 | | 000010 | 011 | | 000100 | 110 | | 001000 | 001 | | 010000 | 010 | | 100000 | 100 | | 010001 | 111 |  1. 纠错举例   我们得到矢量***r*，**并用计算其伴随式。然后利用前面部分得到的查询表（表8.3），求出相应的错误图样。这个错误图样是对错误的一个估计，记为。译码器然后将加到***r***上得到一个估计的发送码字  （8.29）  如果估计的错误图样与真实的错误图样相同，即，那么估计值等于发送码字*U*。另一方面，如果错误估计不正确，则译码器得到的估计值不是发送的码字，这是一个无法检验的译码错误。  【**例8.5**】 纠错  假设发送码字，接收矢量。根据表8.3的伴随式查询表，试说明译码器是如何纠错的。  解：的伴随式计算如下：    利用表8.3，与此伴随式对应的错误图样估计为    正确的矢量估计为    因为此例中估计的错误图样就是实际的错误图样，所以由纠错过程得到。  注意，对于受干扰码字的译码，首先要检测，然后纠正一个错误，这一过程可以与医疗过程做个类比。病人（可能受到干扰的码字）进入医疗机构（译码器）；检查的医生做了一个诊断（乘以），以找到症状（伴随式）。想象医生找到了病人*X*光照片上的特定点，一个有经验的医生会立即将症状与疾病肺结核病（错误图样）联系起来。而一个新手医生可能还要参阅一下医科手册（表8.3），将症状（伴随式）与病症（错误图样）联系起来。最后一步是提供合理的治疗来消除疾病，就像式（8.29）所看到的一样。在二进制编码与医疗的类比中，式（8.29）揭示了一种特殊的医学实践方法，即病人通过重新运用原来的疾病而获得治疗。  8.3.9译码器的实现  在码比较短的情况下，如同在前面一节描述的（6,3）码，译码器可以用简单的电路实现。考虑译码器必须采取的步骤：（1）计算伴随式，（2）找出错误图样，（3）对错误图样和接收到的矢量执行模2加运算（除去错误）。在例8.5中，从一个被干扰的矢量开始，可以看到以上这些步骤如何产生正确的码字。现在考虑图8.6中的电路由异或门和门组成，它可以与（6,3）码的单个错误图样产生相同的结果。根据表8.3和式（8.28），可以根据接收到的码字为每个伴随式写出表达式      和    下面用这些伴随式构成图8.6中的电路。异或门进行的是模2算数加的操作，因此使用相同的符号。进入与门的连线结尾的小圆圈代表信号的逻辑取反。  受到干扰的信号同时进入译码器的两个位置。在电路上端，计算伴随式，在电路的下端，伴随式变为它对应的错误图样。通过把错误图样加到接收矢量以产生正确的码字，这样就消除了错误。    图8.6 （6,3）译码器的实现  注意，为了讲解方便，图8.6只强调了代数译码步骤——伴随式的计算、错误图样和正确的结果。实际情况是，一个（）码通常设置为系统码，译码器不需要输出整个码字；它的输出只要仅仅包含数据比特。因此，图8.6通过删除带阴影的门得到简化。对于较长的码，这种实现非常复杂，更常使用的译码技术是通过采用串行方法而不是并行方法简化电路。同样需要强调的是，图8.5只能检测和纠正（6,3）码单个错误的图样。对两个错误图样的控制需要更多的电路。  码字、错误图样、接收矢量和伴随式分别用矢量***U*，*e*，*r***和***S***表示，为了简化起见，表示一个矢量所用的下标通常被省略，准确地说，这些矢量***U*，*e*，*r***，***S*，**每一个都是一个集合，具有如下一般的形式：    考虑下标j和i在表8.2中（6,3）码的取值范围。对于码字，下标表明有个不同的码字，而下标代表每一个码字由*n*=6比特组成。对于一个可纠正的错误图样，下标代表有个陪集首（7个非零可纠正的错误图样），而下标代表每一个错误图样由*n*=6比特组成。而对于接受矢量，下标代表有个可能的*n*元组可能被接收到，而下标代表每一个接收到的*n*元组有6比特。最后，对于伴随式，下标代表有个不同的伴随式矢量，而下标代表每一个伴随式由*n*-*k*=3比特组成。在本章中，下标通常被省略去，而矢量分别记为***U*，*e*，*r***和***S***。 | | 介绍必要的基础知识  空间与子空间的概念是理解线性分组码的关键  理解线性分组码的典型案例  生成矩阵和监督矩阵是信道编码的核心知识点  注意系统码与非系统码的异同  伴随式检验是理解线性分组码纠错原理的核心，需要展开讲解 |
| **小结** | 线性分组码的生成矩阵与监督矩阵；伴随式检验。 | |
| **复习要点** | 掌握线性分组码的设计； | |
| **思考题** | 两种编码方法的区别是什么？ | |
| **作业题** | 8.4 8.7 | |

作者签名：

****