摘　要

随着通信技术的飞速发展，信道编码已经成功地应用于各种通信系统中。随着各种传输方式对可靠性要求的不断提高，信道编码技术作为抗干扰技术的一种重要的手段，在数字通信技术领域和数字传输系统中显示出越来越重要的作用。

本文主要对线性分组码和卷积码进行编译码分析，并用 MATLAB 进行了仿真实现。线性分组码以（7,4）汉明码为例，利用 MATLAB 对其在二进制对称信道(BSC)，高斯白噪声信道(AWGN)中的仿真，通过误码率的曲线图来分析汉明码的性能。卷积码则用 MATLAB 进行仿真和纠错验证,并在不同信噪比、不同判决方式情况下对误码率进行分析。仿真结果显示，汉明码随着信噪比的增加,误码率越来越小，而对于卷积码，信噪比稍高会大大降低其误码率，且采用软判决译码方式误码率更低，效果更佳。

关键字：信道编码　MATLAB　误码率

Abstract

With the rapid development of communication technology, channel coding has been successfully applied to various communication systems. And all kinds of transmission mode of reliability requirements continue to improve, channel coding technology as anti-interference technology as an important means, in the field of digital communication technology and digital transmission system shows more and more important role.

In this paper, the linear block codes and convolutional codes are compiled and analyzed, and the simulation is carried out with MATLAB. Linear block code to (7,4) Hamming code as an example, using MATLAB to the over a binary symmetric channel (BSC), additive white Gaussian noise (AWGN) simulation, the bit error rate curve to analyze performance of Hamming code. The convolutional code is verified by MATLAB for simulation and error correction, and the error rate is analyzed under different SNR and different decision modes.Simulation results show that the Hamming code with increase of the signal to noise ratio, bit error rate is getting smaller and smaller, and for convolutional codes, signal-to-noise ratios were slightly higher will greatly reduce the bit error rate (BER) and using a soft decision decoding error rate is lower, a better effect.

Keywords: channel coding　MATLAB　bit error rate

目录

1 绪 论

本章首先介绍了所选课题研究目的及意义，并对信道编码的国内外研究现状及发展进行综述，最后概述了本文的主要内容及结构安排。

1.1 研究目的及意义

20 世纪 40 年代，香农提出可以通过差错控制编码（又称为信道编码）在信息传输速率不大于信道容量的前提下实现可靠通信。在随后的半个世纪，信道编码技术无论在理论还是实际中都得到飞速发展，现在的绝大多数数字通信系统都使用该技术以增加通信的可靠性。信道编码种类多种多样，主要分为分组码和卷积码两大类，另外还有编码与调制、编码与交错、编码与编码级联等方式的应用，目的在于提高通信系统性能。不同的通信系统对于信道编码方案的选择有不同的要求，影响信道编码的方案主要有数据、信道以及用户需求三方面因素：数据主要表现在数据的结构，信息的特性，对误码率的要求，数据速率以及各种处理实时性要求；信道要求表现在功率和带宽限制以及信道特性上；用户需求主要是系统实现的成本。

而香农在他的论文中正式应用概率理论对通信系统进行研究和分析，将通信系统抽象为图 1-1 所示的基本框图，并且成功地定义了信息量的概念，由此提出了信道及信道容量的概念，同时他还提出了著名的信道编码定理，从而奠定了信息理论的基础。

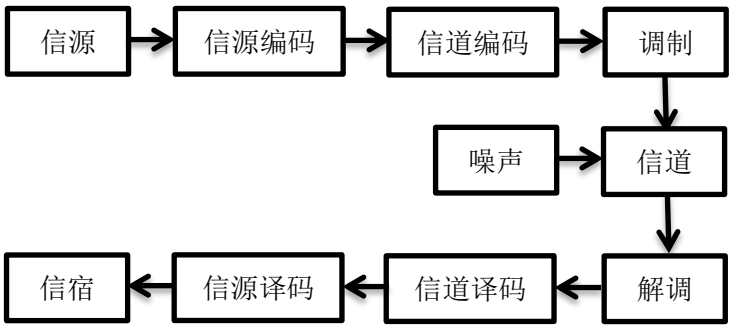

图 1-1 数字传输系统的基本原理

1.2 国内外研究现状及发展前景

1948 年香农发表论文《通信的数学理论》，提出可以使用信道编码的方法提高信息传输的可靠性，开创了信道编码技术的发展历史。半个多世纪以来，随着通信技术的发展，传输系统对可靠性要求的不断提高，信道编码技术在数字通信技术领域中的地位越来越重要。不断有新的信道编码方案被提出，性能离香农限越来越近。

20 世纪 50 年代陆续提出和构造几种编码方案包括汉明码、Go]ay 码和 Reed—Muller 码。这几类码属于分组码，它们使用于 AWGN 信道，而不适用于衰落信道，因为这几种编码只是针对随机错误，而不能纠正突发错误。这些码构造出的码长较短，所有的译码方法都是采用硬判决译码，损失了一定的软判决译码增益随。1959 年 Bose 和 Chaudhuri 提出了一类循环码中的好码 BCH 码，具有较好的编码增益，目前很多系统仍在使用。

1961 年 MIT 的 Gallager 博士在他的博士论文中首次提出著名的 LDPC 码的思想，在当时未引起人们重视，但今天事实证明，LDPC 码是目前最接近香农限的好码之一。1960 年 Reed 和 Solomon 提出 RS 码，由于其与早期的编码相比具有良好频带有效性，广泛应用与卫星通信信道中。1955 年 Elias 提出不同于分组码的卷积编码，由 Wozencraft 和 Reiffen 提出针对卷积编码的序列译码方法，由 Massey 提出了易于实现的门限译码方法，1967 年 Viterbi 提出针对卷积码的最大似然译码方法，实现对约束长度小的卷积码实现软判决译码。

1978 年欧洲的 Ungerboeck 和日本的 Imai 分别独立地提出了具有不展宽频带的编码调制相结合的思想，通过对信号空间的分配，使信号星座点之间的最小欧式距离最大。台湾 L.F.Wei 教授发明了多维 TCM 网格编码，既有很好的功率有效性，又有很好的频带有效性。1996 年 Forney 提出级联编码，通过使用短分量码构造长码，可以获得较高的可靠性和较低的译码复杂度。1993 年，C．Barrou 在 IEEE 国际通信会议上发表了 Turbo 码论文，给出了 Turbo 码结构及其在 BPSK 下几乎可以达到香农限的性能。1995 年 MacKay 和 Neal 重新发现低密度校验码的优越性后，越来越多的研究者将注意力集中在 LDPC 码上，并将其作为未来高速宽带移动通信系统中的信道编码的主要备选方案之一。

目前，国内外信道编码技术广泛应用于各种数字通信系统。比如在数字电视中常用的纠错编码，通常采用两次附加纠错码的前向纠错（FEC）编码。这种纠错码信息不需要储存，不需要反馈，实时性好。所以目前国内外广播系统几乎都采用这种信道编码方式。

在卫星通信系统中，，卷积码作为其传输信道编码中双层级联码的内码，与外码 RS 码一起，可以有效地纠正随机和突发错误，在码率为 10^-5 时，其编码增益可达 6.7dB，大大提高了卫星数据传输质量，而且节省了卫星功率。因此，在卫星通信系统中 RS 码和卷积码及其级联码已成为了最常用的的信道码。

而在移动通信领域，3G 和 4G 的发展也离不开信道编码。比如我国的 WCDMA 和 cdma2000 方案都建议采用除与 IS－95 CDMA 系统类似的卷积编码技术和交织技术之外，采用 Turbo 编码技术。随着 LDPC 码的提出，国内外对 4G 技术的研究又有了一个新层次。

1.3 主要内容和章节安排

信道编码的目的是为了改善通信系统的传输质量，对于它的研究已经涉及到通信领域的方方面面。本文的主要研究内容是运用 MATLAB，对信道编码进行仿真，从而分析它的性能。

第一章主要介绍了本文的研究目的及意义，通过国内外一些相关研究技术的发展，了解目前信道编码在各领域的应用。

第二章信道编码以及其运行环境 MATLAB 的介绍，首先讲述了信道编码的概念与分类，通过介绍几种简单而常见的信道码和信道，进一步了解了本文的主要研究内容，最后对 MATLAB 语言作了简单的介绍。

第三章线性分组码，线性分组码是信道编码里一种常见码字，通过介绍线性分组码的基本原理，奠定了编码的理论基础。随后分别讲解编码译码方法及其 MATLAB 仿真程序，并以线性分组码的一种常见码字汉明码为例，对高斯白噪声信道和二进制对称信道的仿真，得出误码率曲线，分析信噪比对误码率的影响。

第四章卷积码，分析了卷积码编译码原理和算法，并通过 MATLAB 仿真对卷积码性能进行研究，重点比较分析了不同信噪比、不同译码判决方式对 Viterbi 译码性能的影响，并得出相关结论。

第五章总结，对三、四两章的仿真结果作出了综合论述。并写出了自身在毕业设计过程中遇到的一些问题及感悟。

2　信道编码及仿真平台简介

2.1 信道编码的概念及分类

进行信道编码是为了提高信号传输的可靠性，改善通信系统的传输质量，研究信道编码的目标是寻找具体构造编码的理论与方法。从原理上，构造信道码的基本思路是根据一定的规律在待发送的信息码元中人为的加入一定的多余码元，以引入最小的多余度为代价来换取最好的抗干扰性能。信道编码是通过信道编码器和译码器实现的用于提高信道可靠性的理论和方法，是信息论的内容之一。

信道编码大致分为两类：①信道编码定理，从理论上解决理想编码器、译码器的存在性问题，也就是解决信道能传送的最大信息率的可能性和超过这个最大值时的传输问题。②构造性的编码方法以及这些方法能达到的性能界限。编码定理的证明，从离散信道发展到连续信道，从无记忆信道到有记忆信道，从单用户信道到多用户信道，从证明差错概率可接近于零到以指数规律逼近于零，正在不断完善。编码方法，在离散信道中一般用代数码形式，其类型有较大发展，各种界限也不断有人提出，但尚未达到编码定理所启示的限度。在连续信道中常采用正交函数系来代表消息，这在极限情况下可达到编码定理的限度，不是所有信道的编码定理都已被证明。

2.2 常用信道编译码方法

1948 年，信息论的奠基人 Shannon 在他的开创性论文《通信的数学理论》中，提出了著名的有噪信道编码定理。他指出：对任何信道，只要信息传输速率 R 不大于信道容量 C，　就一定存在这样的编码方法：在采用最大似然译码时，其误码率可以任意小。该定理在理论上给出了对给定信道通过编码所能达到的编码增益的上限，并指出了为达到理论极限应采用的译码方法。在信道编码定理中，香农提出了实现最佳编码的三个基本条件 ：

（1）采用随机编译码方式 ；

（2）编码长度 $L \to \infty$ ，即分组的码组长度无限；

（3）译码采用最佳的最大似然译码算法。

下面介绍常用的前向纠错信道码的编码方法和译码算法。

2.2.1 分组码

简单来说，所谓“分组码”就是指码字的生成只与当前的源数据有关 ，而与前后的数据无关（相应地，与前后数据相关的码称为卷积码）。分组码将连续的数据比特流分割为长度固定的组；各组进一步以 m 比特为单位分割为符号，通常取 3 比特或者 8 比特数据组成一个符号。k 个符号一起组成源字，经过编码后变为长度为 n 的码字，称为 m 比特符号的（n，k）分组码。所谓“线性”，是指编码过程均为线性变换，即可以通过矩阵变换来表示。目前研究的纠错码大都属于线性码。在线性空间中，所有可能的 m 比特源字都可以进行编码变换，而无需关心这 m 比特数据所代表的含义。所谓“系统”，是指码字中包含了源字和变换所得的校验字。

2.2.2 卷积码

卷积码属于非分组码，它是一种小分组（n，k）多码段相关、纠错能力较强的前向纠错码。卷积码不同于（n，k）分组码，它将（n，k）变成很短的分组（n，k），如（2，1）、（3，2）卷积码等。每一个监督元不仅是由本码段（n，k）的 k 位信码所决定，而且与其前 N-1 个码段的信码有关，因此称为卷积码。它适于串行传送，延时较小。

卷积码是应用最多的一类编码，其译码算法包括等效于最大似然译码的 Viterbi 译码算法以及 Fano 译码算法、堆栈式译码算法等。

2.2.3 级联码

级联码实际上是组合码，包括串行级联、并行级联和混合级联等方式，基连的目的主要是在保持译码复杂性增加受控的条件下提高信道纠错能力。例如，将纠突发错误能力提出的 RS 码与纠随机错误性能较好的卷积码级联，可以实现对不同信道错误的纠正。

比较成熟的一类级联码是 RS 码和卷积码组成的串行级联码。另外，Turbo 码作为一类性能优异的并行级联码，已经成为移动通信、卫星和深空通信、宽带接入等系统的主要编码方案之一。

2.3 信道

信道（Channel），通俗地说，是指以传输媒质为基础的信号通路。具体地说，信道是指由有线或无线电线路提供的信号通路。信道的作用是传输信号，它提供一段频带让信号通过，同时又给信号加以限制和损害。

通常，我们将仅指信号传输媒介的信道称为狭义信道。狭义信道按具体媒介的不同类型可分为有线信道和无线信道。在通信原理的分析中，从研究消息传输的观点看，我们所关心的只是通信系统中的基本问题，因而，信道的范围还可以扩大。它除包括传输

媒介外，还可能包括有关的转换器，如馈线、天线、调制器、解调器等等。通常将这种扩大了范围的信道称为广义信道。广义信道通常也可分成两种：调制信道和编码信道。本节介绍的便是在通信系统仿真中常用的编码信道模型。

2.3.1 二进制对称信道（BSC）

二进制对称信道的输入和输出都只有 0 和 1 两种符号，并且发送 0 而接受到 1，以及发送 1 而收到 0（即误码）的概率相同，所以称信道是对称的。此时条件差错概率由 p 表示。二进制对称信道的转移概率如图 2-1 所示。

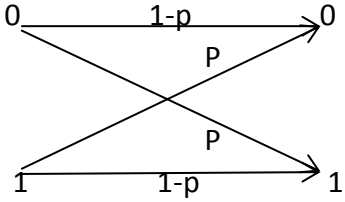

图 2-1　二元对称信道模型

该信道的信道转移概率为

$$p=\begin{bmatrix}1-p & p\\ p & 1-p\end{bmatrix} \tag{2-1}$$

2.3.2 高斯白噪声信道（AWGN）

在信号传输的过程中，它会不可避免地受到各种干扰，这些干扰统称为“噪声”。加性高斯白噪声 AWGN 是最常见的一种噪声。它的信道模型如图 2-2 所示。

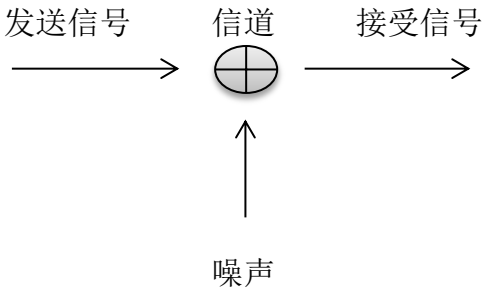

图 2-2　AWGN 信道模型

AWGN 信道模型在通过信道的信号上添加高斯白噪声，通常添加噪声的幅度和程度用信噪比（SNR）表征，信噪比一般定义为信号功率与噪声功率之比（dB 形式），即

$$SNR=10log_{10}\frac{P}{N}(dB) \tag{2-2}$$

其中，P 为信号功率（通常取 1），N 为高斯白噪声的功率。

2.4 MATLAB 简介

MATLAB 是矩阵实验室（Matrix Laboratory）之意，除具备卓越的数值计算能力外，它还提供了专业水平的符号计算，文字处理，可视化建模仿真和实时控制等功能。MATLAB 将高性能的数值计算和可视化集成在一起，并提供了大量的内置函数，从而被广泛地应用于科学计算、控制系统、信息处理等领域的分析、仿真和设计工作，而且利用 MATLAB 产品的开放式结构，可以非常容易地对 MATLAB 的功能进行扩充，从而在不断深化对问题认识的同时，不断完 MATLAB 产品以提高产品自身的竞争能力。

MATLAB 的基本数据单位是矩阵，它的指令表达式与数学，工程中常用的形式十分相似，MATLAB 故用 MATLAB 来解算问题要比用 C、FORTRAN 等语言完相同的事情简捷得多。作为一种数值计算和与图形处理工具软件，其特点是语法结构简明、数值计算高效、图形处理完备、易学易用，它在矩阵代数数值计算、数字信号处理、震动理论、神经网络控制、动态仿真等领域都有广泛的应用。与其他高级语言相比，MATLAB 不但在数学语言的表达与解释方面表现出人机交互的高度一致，而且具有优秀高技术计算环境所不可缺少的如下特征：

（1）高质量、高可靠的数值计算能力；

（2）基于向量、数组和矩阵的高维设计语言；

（3）高级图形和可视化数据处理的能力；

（4）广泛解决各学科各专业领域内复杂问题的能力；

（5）拥有一个强大的非线性系统仿真工具箱——Simulink；

（6）支持科学和工程计算标准的开放式、可交互结构；

（7）跨平台兼容。

3　线性分组码

信道编码是为了保证信息传输的可靠性、提高传输质量而设计的一种编码。其实质是在信息码元中增加一定数量的监督（校验）码元，使它们满足一定的约束关系，在传输过程中如果受到干扰，某些码元发生了错误，就破坏了它们之间的约束关系，收方通过检验这种约束关系发现错误并在纠误能力范围内加以纠正，从而保证通信的可靠性。

本章将着重讲述线性分组码的编码特征及其 MATLAB 仿真。

3.1 线性分组码基本理论

线性分组码在信道编码和译码过程中是按照分组进行的，线性分组码中的监督码是按线性方程生成的。循环码是线性分组码一个最主要子类，绝大多数工程应用的线性分组码都是循环码。循环码具有很多良好理论结构与实际性能，如理论上的固有代数结构，特别是模多项式性质，工程上可采用反馈移位寄存器来进行码字构造等。

3.1.1 线性分组码的定义

分组码的基本思想是对信息序列分段编码。若对包含 k 个信息元的信息组 M

$$M=(m_{k-1},m_{k-2},...,m_1,m_0) \tag{3-1}$$

按照一定的编码规则产生包括 n 个码元的码组 C

$$C=(c_{n-1},c_{n-2},...,c_1,c_0) \tag{3-2}$$

编码规则定义为

$$\begin{aligned} c_0 &= f_0(m_{k-1},m_{k-2},...,m_1,m_0) \\ c_1 &= f_1(m_{k-1},m_{k-2},...,m_1,m_0) \end{aligned} \tag{3-3}$$

以此类推，因此，由定义可知，分组码由一组固定长度称为码字的矢量构成，码字的长度是矢量元素的个数，用 n 表示。码字的元素选自由 q 个元素组成的字符集。当字符集由 0、1 两个元素组成时，该码就是二进制码，此时码字的任一元素称作比特。当码字的元素从由 q 个元素（q>2）组成的字符集选取时，该码为非二进制码。应该指出，当 q 是 2 的幂次，即 $q=2^m$（m 是正整数时），每个 q 进制码元可以用对应的包含 m 比特的二进码制表示，分组长度为 N 的非二进制码可以映射成分组长度为 n=mN 的二进制分组码。

长度 n 的二进制分组码有2^n种可能的码字。从这 2^n 种码字中，可以选择 $M=2^k$ 个码字（k<n）组成一种码。这样，一个 k 比特信息的分组可以映射到长度为 n 的一个码字，该码字是从由 $M=2^k$ 个码字构成的码集中选出来的。这样得到的分组码称为（n，

k）码，定义 k/n=R 为码率。除码率 R 这个参数外，另一个重要参数是码字的重量，即该码字包含的非零元素的个数，用 d 表示。通常，不同的码字具有不同的码重，编码中所有码字重量的集合形成该码的重量分布，码重直接影响编码的纠错性能。

表 3-1 给出了（7，4，3）线性分组码的信息组和码字的对应关系。

表 3-1（7，4，3）线性分组码的码表

信 息 组	码 字	信 息 组	码 字
0000	0000000	1000	1101000
0001	1010001	1001	0111001
0010	1110010	1010	0011010
0011	0100011	1011	1001011
0100	0110100	1100	1011100
0101	1100101	1101	0001101
0110	1000110	1110	0101110
0111	0010111	1111	1111111

3.1.2 生成矩阵和校验矩阵

设$x_1,x_2\cdots x_k$是 k 个信息码元 X，用行矢量表示码字。这样，输入编码器的 k 位信息可写作：

$$X=[x_1,x_2,...,x_k] \tag{3-4}$$

编码器输出矢量记为：

$$C=[c_1,c_2,...,c_k] \tag{3-5}$$

对于二进制线性分组码，编码运算可以用一组 n 个方程表示如下：

$$C=XG=\begin{bmatrix}x_1 & x_2 & \cdots & x_k\end{bmatrix}\begin{bmatrix}g1\\ g2\\ \vdots\\ gk\end{bmatrix}=\begin{bmatrix}x_1 & x_2 & \cdots & x_k\end{bmatrix}\begin{bmatrix}g_{11} & g_{12} & \cdots & g_{1n}\\ g_{21} & g_{2\,2} & \cdots & g_{n2}\\ \vdots & \vdots & \vdots & \vdots\\ g_{k1} & g_{k2} & \cdots & g_{kn}\end{bmatrix} \tag{3-6}$$

其中 G 称为该码的生成矩阵。（n，k，d）线性分组码的生成矩阵 G 是一个 k×n 的二元矩阵。例如，可以在前述表 3-1（7，4，3）线性分组码中找到 4 个线性无关的行向量（码字）来构成生成矩阵 G

$$G=\begin{bmatrix}1&1&0&1&0&0&0\\0&1&1&0&1&0&0\\1&1&1&0&0&1&0\\1&0&1&0&0&0&1\end{bmatrix} \tag{3-7}$$

根据线性代数知识，矩阵 G 是由 k 个线性无关的行向量构成的，因此一定存在一个由 n-k 个线性无关的行向量组成的矩阵 H 与之正交，即

$$GH^T = 0 \tag{3-8}$$

矩阵 H 是一个（n-k）×n 的二元矩阵，即线性分组码的校验矩阵。一般情况下，线性分组码的校验矩阵 H 可以表示为

$$H = \begin{bmatrix} h_0 \\ h_1 \\ \vdots \\ h_{n-k-1} \end{bmatrix} = \begin{bmatrix} h_{0,n-1} & h_{0,n-2} & \cdots & h_{0,0} \\ h_{1,n-1} & h_{1,n-2} & \cdots & h_{1,0} \\ \vdots & \vdots & \vdots & \vdots \\ h_{n-k-1,n-1} & h_{n-k-1,n-2} & \cdots & h_{n-k-1,0} \end{bmatrix} \tag{3-9}$$

例如，前述（7，4，3）线性分组码的校验矩阵为

$$H = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix} \tag{3-10}$$

实际上，（n，k，d）线性分组码编码的目的就是如何在 n 维线性空间V_n中找到编码要求的、由2^k个向量组成的 k 维子空间。这相当于建立一组线性方程，已知系数和未知系数的个数分别为 k 个和 n-k 个，且使得码字恰好最小距离为 d。

建立的线性方程组为

$$\begin{bmatrix} h_{0,n-1} & h_{0,n-2} & \cdots & h_{0,0} \\ h_{1,n-1} & h_{1,n-2} & \cdots & h_{1,0} \\ \vdots & \vdots & \vdots & \vdots \\ h_{n-k-1,n-1} & h_{n-k-1,n-2} & \cdots & h_{n-k-1,0} \end{bmatrix} \begin{bmatrix} c_{n-1} \\ c_{n-2} \\ \vdots \\ c_0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{3-11}$$

写成矩阵和行向量的乘积形式，有

$$CH^T = 0 \tag{3-12}$$

根据生成矩阵 G 的定义，对于系统码，矩阵 G 的 k 列可以组成一个 k 维的单位矩阵，如果这个单位矩阵出现在矩阵 G 的最左边 k 列（也可以是最右边的 k 列），则码字 C 的钱前 k 位就是信息元。由此，矩阵 G 可以写成如下分块矩阵的形式：

$$G = [I_k \ \mathrm{P}] \tag{3-13}$$

其中，P 是 k×（n-k）维矩阵。根据校验矩阵 H 和生成矩阵 G 的正交关系，有

$$H = [-P^T \ I_{n-k}] \tag{3-14}$$

其中，$-P^T$是（n-k）×k 维矩阵，对于二元码，有

$$H = [P^T \ I_{n-k}] \tag{3-15}$$

3.1.3 纠错能力

对于线性分组码，其纠错能力与码字最小距离直接相关。一般情况下，有如下结论：任一（n，k，d）线性分组码，若要求在码字内，

（1）检测 e 个随机错误，则要求码的最小距离 $d \geq e+1$；

（2）纠正 t 个随机错误，则要求码的最小距离 $d \geq 2t+1$；

（3）纠正 t 个同时检测 e 个随机错误（ $e \geq t$ ），则要求码的最小距离 $d \geq t+e+1$；

（4）纠正 t 个随机错误同时纠正 e 个删除，则要求码的最小距离 $d \geq 2t+e+1$。

3.2 编码方法及 MATLAB 仿真

对于线性分组码的编码，可通过生成矩阵 G 来进行。假设信息组 $M=(m_{k-1},m_{k-2},...,m_1,m_0)$，码组为 $C=(c_{n-1},c_{n-2},...,c_1,c_0)$，则 M、C 与 G 之间的关系为

$$C=MG \tag{3-16}$$

对于系统码，有

$$C=[M\ MP] \tag{3-17}$$

线性分组码的编码程序如下：

```
function code = Block_encoder(n,k,msg,G）
%输入
%n:码字长度
%k：信息组长度
%msg：编码输入信息组
%输出
%code：编码输出码字
[M,N] = size(G);
if N~=n
    disp('Parameter of Code Length is error.\n');
end
if M~=k
    disp('Parameter of Info. Length is error.\n');
end
%计算编码输出
code = rem(msg*G,2);
```

3.3 译码算法及 MATLAB 仿真

3.3.1 伴随式译码

伴随式 S 定义为

$$S = RH^T \tag{3-18}$$

式中，S 是 n-k 维列向量。根据发送码字 C 与校验矩阵 H 之间的关系以及发送码字 C 与接收码字 R 之间的关系，有

$$S = RH^T = (C+E)H^T = CH^T + EH^T = EH^T \tag{3-19}$$

由此可以看出，伴随式的值仅与信道错误图样有关，与发送码字无关。如果在信道传输过程中无错误发生，即 E=0，则 S=0；否则 S≠0。伴随式译码算法的基本思想就是根据伴随式 S 的值来估计错误图样 E。如果将校验矩阵 H 写成列向量的形式

$$H = [h_{n-1}\ h_{n-2}\ \cdots\ h_0] \tag{3-20}$$

则有

$$S = E \bullet H^T = \sum_{i=0}^{n-1} e_i h_i \tag{3-21}$$

综上所述，伴随式译码的主要步骤如下：

（1）根据接受码字，利用公式 $S = RH^T$ 计算伴随式 S；

（2）根据伴随式 S，在码字纠错能力范围内得到错误图样 E 的估计；

（3）估计发送码字 C=R+E。

3.3.2 标准阵列译码

由于（n，k，d）线性分组码的 2^k 个码字构成了 n 维线性空间的一个 k 维子空间，即一个子群。以子群为基础可以将整个 n 维线性空间的所有 2^n 个元素划分成个 C_1 陪集，如表 3-2 所示

表 3-2　标准阵列译码表

码字	C_1	C_2	…	C_i	…	C_{2^k}
禁用码字	E_2	C_2+E_2	…	C_i+E_2	…	$C_{2^k}+E_2$
	E_3	C_2+E_3	…	C_i+E_3	…	$C_{2^k}+E_3$
	…	…	…	…	…	…
	$E_{2^{n-k}}$	$C_2+E_{2^{n-k}}$	…	$C_i+E_{2^{n-k}}$	…	$C_{2^k}+E_{2^{n-k}}$

在标准阵列译码表，2^k 个码字放置在第 1 行，该子群的恒等元素 C_1（全 0 码字）放在最左边，然后在禁用码字集合中选择一个 E_2 放置在下面一行，并计算其与所有码字的模 2 和，放置在第 2 行，构成码空间的一个陪集。类似的选择 E_3、E_4、…、$E_{2^{n-k}}$ 并构成个 2^{n-k} 陪集。其中第 1 列的向量称为陪集首。

标准阵列译码的关键问题是如何确定陪集首。一般原则是保证译码器能够纠正出现可能性最大的错误图样，即重量最小的错误图样，所以选择重量最小的 n 重向量作为陪集首。

线性分组码的译码程序如下：

```
function est_code=Block_decoder(n,k,rec,H,d)
%输入
%n:码字长度
%k:信息组长度
%rec:译码器接受的硬判决码字
%H:分组码的校验矩阵
%d:分组码的最小距离
%输出
%est_code:译码输出码字
[M,N]=size(H);
if N~=n
    disp('Parameter of Code Length is error.\n');
end
if M~=n-k
    disp('Parameter of Parity Length is error.\n');
end
t=fix((d-1)/2);              %初始化并计算不同个数 H 矩阵列向量的所有组合的个数
num=zeros(1,t);
for idx=1:t
   num(idx)=factorical(n)/factorical(n-idx)/factorical(idx);
end
maxnum=max(num);                    %计算最大组合数
out=zeros(t,maxnum,t);              %分配空间
```

```
out(1,1:num(1),1=1:n);              %单个 H 列向量所以
out=compiund(out,n,num,t,1);        %2 到 t 个 H 列向量组合的所有可能索引集合
Hcom=zeros(1,n-k);                  %初始化 H 组合列向量
E=zeros(1,N);                       %初始化错误图样
S=rem(rec*H',2);                    %计算伴随式
if find(S)                          %伴随式不为零，表示码字中有错
    for  err=1:t                            %在分组码的纠错范围内，查找与伴随式值匹配的 H 列向量或器组合
        for idx=1:num(err)                  %逐个检查 H 列向量组合
            Hcom=zeros(1,n-k);
            for j=1:err                     %计算 H 组合列向量
                Hcom=rem(Hcom+H(:,out(err,idx,j))',2);
            end
            if(sum(rem(S+Hcom,2))==0)       %找到匹配的 H 列向量组合
                E(out(err,idx,1:err))=1;    %估计错误图样
                est_code=rem(rec+E,2);      %纠错
                return;
            end
        end
    end
else                                        %无错，直接输出
    est_code=rec;
end
```

3.4 汉明码及其 MATLAB 仿真

3.4.1 汉明码定义

汉明码是一类最常用的线性分组码，能纠正单个随机错误。汉明码的定义即若一个（n，k）线性分组码码长满足

$$\frac{q^{n-k}-1}{q-1} \tag{3-22}$$

则称它为汉明码。特别的，当$q=2$，$n=2^{n-k}-1$时可得到最常用的二进制汉明码。

当校验矩阵 H 是由任意次序排列的$2^{n-k}-1$个互不相同的非全 0 的 n-k 维列向量组成时，它所构成的线性分组码称为 GF（2）上的（n，k）汉明码。GF（2）是有限域的一种，二元域。它可由一个不小于 3 的正整数 m 定义，其码长为$n=2^m-1$，信息组长度为$k=\mathrm{n}-\mathrm{r}=2^m-1-m$，检验元个数为$\mathrm{r}=\mathrm{n}-\mathrm{k}=\mathrm{m}$，最小距离 d=3。

汉明码的检验矩阵 H 由所有不全为 0 的 m 维列向量构成。任意改变检验矩阵 H 中的各列位置都不会影响汉明码的纠错能力。

例如，m=3 时得到的（7，4，3）汉明码的生成矩阵 G 和校验矩阵 H 分别为

$$G=\begin{bmatrix}1&0&0&0&0&1&1\\0&1&0&0&1&0&1\\0&0&1&0&1&1&0\\0&0&0&1&1&1&1\end{bmatrix}\qquad H=\begin{bmatrix}0&0&0&1&1&1&1\\0&1&1&0&0&1&1\\1&0&1&0&1&0&1\end{bmatrix}\qquad (3\text{-}23)$$

3.4.2 汉明码对高斯白噪声信道的仿真

下面以汉明码为例，从产生信源、编码、调制、信道（噪声）、解调、译码这完整的通信系统模块，来分析在高斯白噪声信道中，随着信噪比的变化，信道误码率的变化。MATLAB 程序参见附录 1。

仿真结果分析：

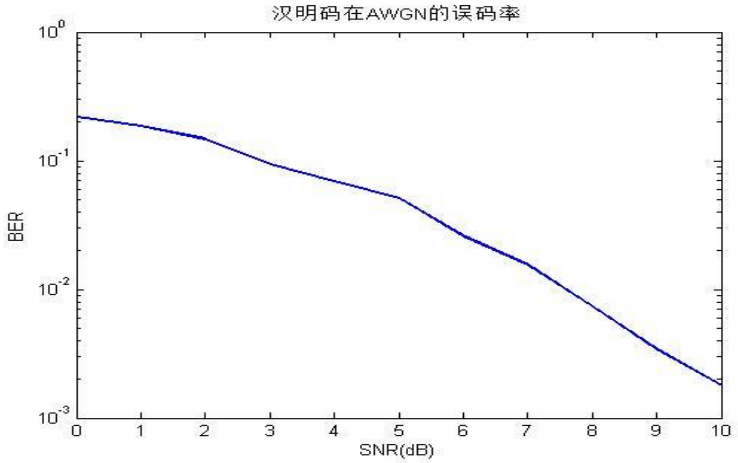

图 3-1　汉明码对 AWGN 信道仿真的误码率曲线图

对图 3-1 的观察，我们可以看到，在高斯白噪声信道里当信噪比越来越大时，误码率越来越低，到一定程度，信道的误码率会为 0。仿真结果和理论结果基本一致。

3.4.3 汉明码对二进制对称信道的仿真

误码率 BER（bit error ratio）衡量数据在规定时间内数据传输精确性的指标，是衡量一个数字系统可靠性的主要的判断依据，理论上误码率越低越好。而信噪比 SNR（signal

noise ratio）通常指的是基带信号中有用信号功率与噪声功率的比值。在传输过程中，无论什么信道，随着信噪比的增加，误码率越来越低，这已经在上节高斯白噪声信道中得到验证。

对于二进制对称信道，必定是随着信噪比的增加，误码率越来越低，这里便不在赘述。我们都知道二进制对称信道有个差错概率 p，下面则研究一下 p 对汉明码在二进制对称信道中误码率的影响。MATLAB 程序参加附录 2。

仿真结果分析：

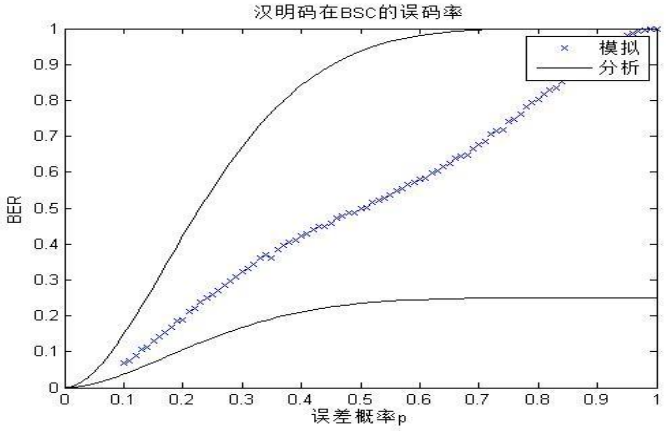

图 3-2　汉明码对 BSC 信道编码后误码率的曲线图

如上图 3-2 中，可以看到蓝色的曲线是模拟后的误码率曲线，黑色的曲线是分析的误码率曲线。上面黑色曲线是概率误差 p 最大值的误码率，下面黑色的曲线是概率误差 p 最小值的误码率。从图中可以看出概率误差最大，BSC 信道的误码率就越高。

4　卷积码

卷积码是 1955 年由 Elias（1923～2001）首次提出的，随后 Wrozencran 和 Reiffen 提出了序列译码方法，对具有较大约束长度的卷积码非常有效。1963 年，Massey 提出了门限译码，这使得卷积码大量应用于卫星和无线信道的数字传输中。

1967 年，Viterbi 提出了最大似然译码算法，易于实现具有较小约束长度的卷积码的软判决译码。Viterbi 算法配合序列译码的软判决，使卷积码在 20 世纪 70 年代广泛应用在深空和卫星通信系统。1974 年，Balll、Cocke、Jelinek 以及 Raviv 针对具有不等先验概率信息比特的卷积码，提出了最大后验概率译码算法。

由于卷积码充分利用了各码组之间的相关性，无论理论上还是实际中均已证明其性能不差于甚至优于分组码。但是与有严格代数结构的分组码不同，卷积码至今尚未找到可以把纠错性能与码的构成有规律地联系起来的严密的数学手段。目前大都采用计算机来搜索好码。因此，对卷积码的研究还在发展中。

4.1 卷积码基本理论

4.1.1 卷积码的基本概念

卷积码一般用（n，k，m）或 N（n，k）来表示，其中

n：分组长度；

K：分组中信息位数目；

N=m+1：编码约束长度；

卷积码的编码器可以看作是一个由 k 个输入端和 n 个输出端组成的时序网络，时序网络是有记忆的，即卷积码编码器某时刻的输出不仅与该时刻输入编码器的信息有关，而且与以前若干时刻输入编码器的信息有关。卷积码的编码器框图 4-1 所示。

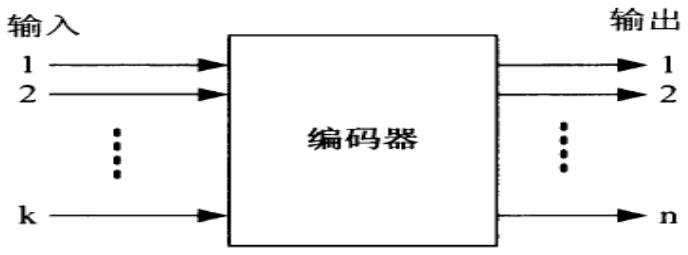

图 4-1　卷积码编码器

4.1.2 卷积码编码原理

由 4.1.1 节可知，卷积码一般表示为（n，k，m）的形式，即将 k 个信息比特编码为

n 个比特的码组，N 为编码约束长度，说明编码过程中相互约束的码段个数。卷积码编码后的 n 个码元不仅与当前组的 k 个信息比特有关，还与前 N-1 个输入组的信息比特有关。编码过程中相互关联的码元有 N*n 个。R=k/n 是编码效率。编码效率和约束长度是衡量卷积码的两个重要参数。典型的卷积码一般选 n，k 较小，但 N 值可取较大（大于 10），以获得简单而高性能的卷积码。卷积码的编码描述方式有很多种：冲激响应描述法、生成矩阵描述法、多项式乘积描述法、状态图描述，树图描述，网格图描述等。下面介绍两种常见的描述方法，状态图法和网格法。

状态图法：

由于卷积码编码器在下一时刻的输出取决于编码器的当前状态和下一时刻的输入，而编码器当前状态取决于编码器当前各移位寄存器的存储内容。称编码器当前各移位寄存器存储内容（0 或 1）为编码器在该时刻的状态（此状态代表记忆以前的输入信息）。随着信息序列的不断输入，编码器不断从一个状态转移到另外一个状态，并且输出相应的编码序列。编码器的总可能状态数为 2mk 个。对（2，1，2）码的编码器来说，n=2，k=1，N=3，m=2。共有四个可能状态，其状态图如图 4-2 所示：

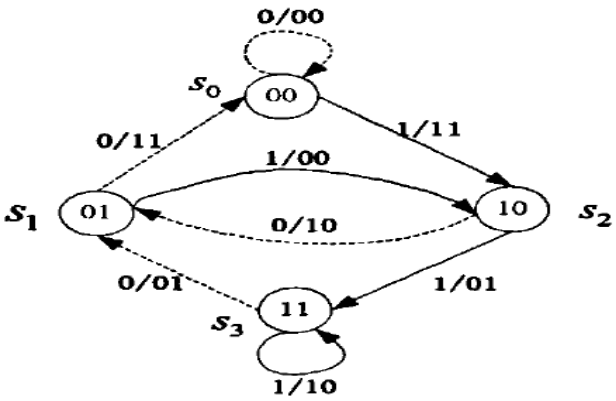

图 4-2　卷积码状态转移图

图中四个圆圈表示状态，状态间的连线与箭头表示转移方向，连线上的数字表示是状态发生转移的到来比特，斜杠后的数字由一个状态到另一个状态转移时的输出码字。如当前状态为 11，输入信息为 0，则转移到 01 状态并输出 01 码字，若输入信息为 1，则依然为 11 状态，并输出 10 码字。

网格图法：

网格图可以描述卷积码的状态随时间推移而转移的情况。该图纵坐标表示所有状态，横坐标表示时间。网格图在卷积码的概率译码，特别是 Viterbi 译码中非常重要，它综合了状态图法直观简单和树图法时序关系清晰的特点。

下面以（2，1，3）卷积码为例，如图 4-3 所示该图设输入信息数目 L=5，所以画 L+N=8 个时间单位，图中分别标以 0 至 7。这里设编码器从 a 状态开始运作。该网格图的每一条路径都对应着不同的输入信息序列。由于所有可能输入信息序列共有2^{kL}个，因而网格图中所有可能的路径也为2^{kL}条。这里节点 a=00，b=01，c=10，d=11。

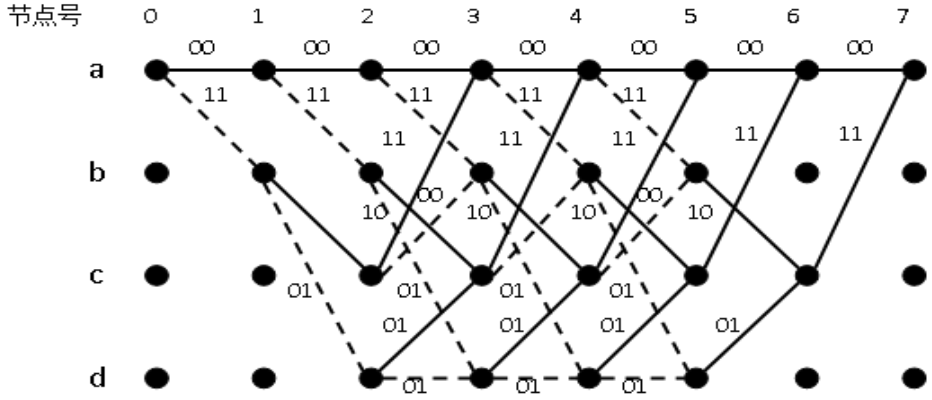

图 4-3　（2，1，3）卷积码网格图

设输入编码器的信息序列为（11011000），则由编码器对应的输出的序列为 Y=（1101010001011100），编码器的状态转移路线为 abdcbdca。若收到的序列 R=（0101011001011100），对照网格图来说明维特比译码的方法。

由于该卷积码的约束长度为位，因此先选择接收序列的前 6 位序列 R1=（010101）同到达第 3 时刻的可能的 8 个码序列（即 8 条路径）进行比较，并计算出码距。该例中到达第 3 时刻 a 点的路径序列是（000000）和（111011），他们与 R1 的距离分别为 3 和 4；到达第 3 时刻 b 点的路径序列是（000011）和（111000），他们与 R1 的距离分别为 3 和 4；到达第 3 时刻 c 点的路径序列是（001110）和（110101），他们与 R1 的距离分别为 4 和 1；到达第 3 时刻 d 点的路径序列是（001101）和（110110），他们与 R1 的距离分别为 2 和 3。上述每个节点都保留码距较小的路径作为幸存路径，所以幸存路径码序列是（000000）、（000011）、（1101001）和（001101），如图 4-4 所示。用于上面类似的方法可以得到第 4、5、6、7 时刻的幸存路径。

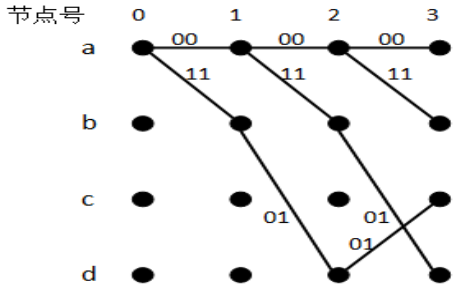

图 4-4　维特比译码第 3 时刻幸存路径

需要指出的是，对于某个节点，如果比较两条路径与接收序列的累计码距值相等时，则可以任意选者一条路径作为幸存路径，吃时不会影响最终的译码结果。在码的终了时刻 a 状态，得到一条幸存路径。如果 4-5 所示。由此可看到译码器输出是 R’=（1101010001011100），即可变换成序列（11011000），恢复了发端原始信息。比较 R’和 R 序列，可以看到在译码过程中已纠正了在码序列第 1 和第 7 位上的差错。当然如果差错出现太频繁，以致超出卷积码的纠错能力，还是会发生纠误的。

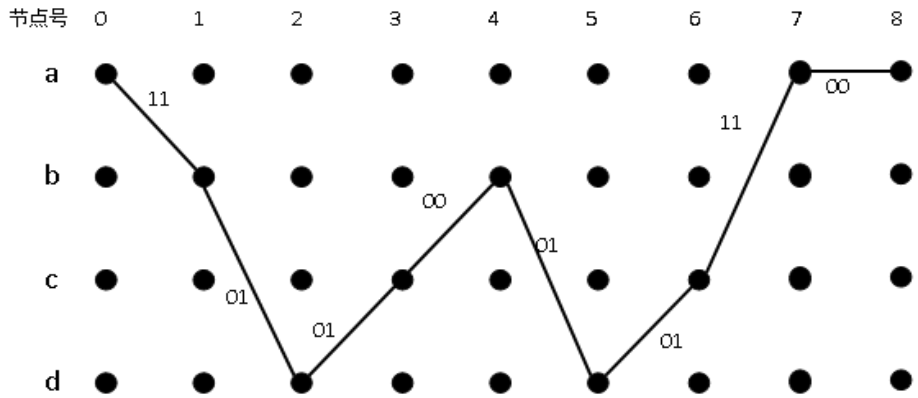

图 4-5　第 8 时刻幸存路径

4.1.3 卷积码译码原理

卷积码的译码可分为代数译码和概率译码两类，其中概率译码又可分为维特比译码和序列译码两种，因此卷积码的译码方法主要有以下 3 种：

1、1963 年由 Massey 提出的门限译码。这是一种利用码代数结构的代数译码，类似于分组码中的大数逻辑译码。它的主要特点是算法简单，易于实现，而且译码延迟是固定的，但不常用。

2、1961 年由 Wozencrar 提出、1963 年由 Fano 改进的序列译码。这是基于码树图结构上的一种准最佳的概率译码，它的特点是平均计算量小，译码速度快，且译码复杂性与码约束度无关。虽然该算法虽然提出较早，但是由于当时硬件条件的限制，并没有大规模应用。

3、1967 年由 Viterbi 提出的 Viterbi 算法。这是基于码的 trellis 图基础上的一种最大似然译码算法，是一种最佳的概率译码方法。由于 Viterbi 译码器的效率高、速度快，在实际中应用最为广泛。

由于本文译码方式为维特比译码，下面重点介绍一下维特比译码原理。

卷积码维特比译码的基本思路是:以接收码流为基础,逐个计算它与其他所有可能出

现的、连续的网格图路径的距离，选出其中可能性最大的一条作为译码估值输出。概率最大在大多数场合可解释为距离最小，这种最小距离译码体现的正是最大似然的准则。卷积码的最大似然译码与分组码的最大似然译码在原理上是一样的，但实现方法上略有不同。主要区别在于：分组码是孤立地求解单个码组的相似度，而卷积码是求码字序列之间的相似度。基于网格图搜索的译码是实现最大似然判决的重要方法和途径。用格图描述时，由于路径的汇聚消除了树状图中的多余度，译码过程中只需考虑整个路径集合中那些使似然函数最大的路径。如果在某一点上发现某条路径已不可能获得最大对数似然函数，就放弃这条路径，然后在剩下的“幸存”路径中重新选择路径。这样一直进行到最后第 L 级（L 为发送序列的长度）。由于这种方法较早地丢弃了那些不可能的路径，从而减轻了译码的工作量，Viterbi 译码正是基于这种想法。

对于（n，k，m）卷积码，其网格图中共 2kL 种状态。由网格图的前 N-1 条连续支路构成的路径互不相交，即最初 2k-1 条路径各不相同，当接收到第 N 条支路时，每条路径都有 2 条支路延伸到第 N 级上，而第 N 级上的每两条支路又都汇聚在一个节点上。在 Viterbi 译码算法中，把汇聚在每个节点上的两条路径的对数似然函数累加值进行比较，然后把具有较大对数似然函数累加值的路径保存下来，而丢弃另一条路径，经挑选后第 N 级只留下 2N 条幸存路径。选出的路径同它们的对数似然函数的累加值将一起被存储起来。由于每个节点引出两条支路，因此以后各级中路径的延伸都增大一倍，但比较它们的似然函数累加值后，丢弃一半，结果留存下来的路径总数保持常数。由此可见，上述译码过程中的基本操作是，“加-比-选”，即每级求出对数似然函数的累加值，然后两两比较后作出选择。有时会出现两条路径的对数似然函数累加值相等的情形，在这种情况下可以任意选择其中一条作为“幸存”路径。

卷积码的编码器从全零状态出发，最后又回到全零状态时所输出的码序列，称为结尾卷积码。因此，当序列发送完毕后，要在网格图的终结处加上（N-1）个已知的信息作为结束信息。在结束信息到来时，由于每一状态中只有与已知发送信息相符的那条支路被延伸，因而在每级比较后，幸存路径减少一半。因此，在接收到（N-1）个已知信息后，在整个网格图中就只有唯一的一条幸存路径保留下来，这就是译码所得的路径。也就是说，在已知接收到的序列的情况下，这条译码路径和发送序列是最相似的。

Viterbi 译码算法有硬判决译码和软判决两种实现方式，两者在结构和译码过程上没有区别，区别在于分支度量的计算方法。硬判决是指解调器根据其判决门限对接收到的信号波形直接进行判决后输出 0 或 1，换句话说，就是解调器供给译码器作为译码用的每个码元只取 0 或 1 两个值，以序列之间的汉明距离作为度量进行译码，适用于二进制

对称信道。而软判决的解调器不进行判决，直接输出模拟量，或是将解调器输出波形进行多电平量化（不是简单的 0、1 两电平量化），然后送往译码器，即编码信道的输出是没有经过判决的“软信息”。软判决译码器以欧几里德距离作为度量进行译码，软判决译码算法的路径度量采用“软距离”而不是汉明距离，最常采用的是欧几里德距离，也就是接收波形与可能的发送波形之间的几何距离，是一种适合于离散无记忆信道的译码方法。实际上，硬判决译码可以看做软判决译码的特殊情况，即采用了 1 比特量化，而软判决译码采用的是多比特量化。

4.2 卷积码编码及 MATLAB 仿真

在程序设计中，我们没有采用 MATLAB 自带的编码函数而是采用了自己的编码函数 codec 对（2，1，3）卷积码编码，其参数 m 为输入信息序列，g1，g2 为两个输出端口的冲激响应序列。

卷积码编码程序：

```
function cod=codec(m,g1,g2)          % g1,g2 为两输出端口的冲激响应序列。
m1=conv(m,g1);                       %端口一输出
m2=conv(m,g2);                       %端口二输出
l=length(m1);
for i=1:l;
cod([2*i-1])=rem(m1([i]),2);         %将端口一编码输出赋给 cod 奇数位置
cod([2*i])=rem(m2([i]),2);           %将端口二编码输出赋给 cod 偶数位置
end
```

在命令窗口试运行编码：

```
clear all
g1=[1 1 1];
g2=[1 0 1];
msg=[1 1 0 1];
cod=codec(msg,g1,g2)
```

输出为：

```
cod =[110101001011]
```

4.3 卷积码译码及 MATLAB 仿真

4.3.1 Viterbi 译码算法解析

Viterbi 算法是通过加-比较-选择来实现的，状态量度的计算方法如下：将前两个状态点上的状态量度和相应分支量度相加，得到的两个可能路径量度作为新的状态量度的候选项，送入逻辑单元中进行比较，将其中似然性最大(距离最小)的一个作为状态的新状态量度存储，同时存储的还有状态新的路径记录，主要算法步骤为：

（1）将接收到的序列分成每段长为n_0的 m 组子序列。

（2）对所研究的码画出深度为 m 级的网格图。对该网格图的最后（L-1）级仅画出对应于全 0 输入序列的路径。

（3）置 s=1，并置初始全 0 状态的度量等于 0。

（4）对网格图中全部连接第 s 级状态到（s+1）级状态的支路求出该接收序列中的第 s 个子序列的距离。

（5）将这些距离加到第 s 级各状态的度量上去，得到对第（s+1）级状态的度量候选者。对于第（s+1）级的每一状态，有2^k个候选度量，其中每一个都对应终止在那个状态的一条支路。

（6）对在第（s+1）级的每一状态，挑选出最小的候选度量，并将对应于这个最小值的支路标以留存支路，同时指定之歌候选度量的最小值作为第（s+1）级状态的度量值。

（7）若 s=m，转到下一步；否则将 s 增加 1 并转到第 4 步。

（8）在第（m+1）级以全 0 状态开始，沿着留存支路通过网格图往会到达初始全 0 状态，这条路径就是最佳路径，并且对应于这条路径的输入比特序列是最大似然解码序列。为了得到有关这个输入比特序列最好的推测，将最后（L-1）k 个 0 从该序列中除掉。

从 Viterbi 算法中可以看到，对于一个长信息序列解码时，解码延时和所需要的存储量都是无法接受的。直到整个序列全被接收，解码才能开始，而且还不得不将总的留存路径存储起来。实际上，不会引起这些问题的次优解倒是很希望的。一种称为路径存储截断的办法是：在每一级编码器在网格图中仅往会搜索 w 级，而不回到网格图的出发点。用这种办法在第（w+1）级解码器对相应于网格图第 1 级的输入比特作出判决，并且未来的接收比特不改变这个判决。这意味着解码延时是k_0w 比特，需要保存的路径只相应于最后 w 级的路径。计算机仿真已经表明，如果 w>5n，由于路径存储截断造成的性能下降可忽略不计。

4.3.2 Viterbi 译码 MATLAB 仿真

Viterbi 译码程序参见附录 3。

将程序在 MATLAB 中仿真，对于（2，1，3）卷积码译码，G=[1 1 1；1 0 1]，此时令信噪比 snr_db=12，输出的是正确结果[1 1 0 1]，由理论分析知，信道编码经过 bpsk 调制，由于信道噪声的干扰，在信号解调后会产生一定的差错，维特比译码虽然有具有纠错功能，但是毕竟是有限的，所以信噪比对 viterbi 译码性能将产生影响，且信噪比越大，维特比译码越准确。为了验证上述推论，将信噪比减小，令 SNR=6，译码输出变为[0 1 1 1]，产生了差错。验证了上述推论。

4.4 信噪比对卷积码译码性能的影响

上述过程只是可以看出信噪比会影响维特比译码的性能，为了进一步得到两者更加直观的联系，我们可以通过编程得到信噪比与误码率之间的函数关系，同时将没有进行卷积码的信号也计算其误码率。

MATLAB 程序参见附录 4。

程序在 MATLAB 中运行，得到仿真图如图 4-6 所示，该图直观的反映了经过卷积码编码和没有经过卷积码编码的误码率之间的区别。从图中可以看出，当信噪比较小时，未编码的误码率反而比采用编码的误码率低，这是因为误比特太多导致接收到的信息几乎无效。但是当信噪比稍高后卷积码编码译码的误码率就大大降低了。

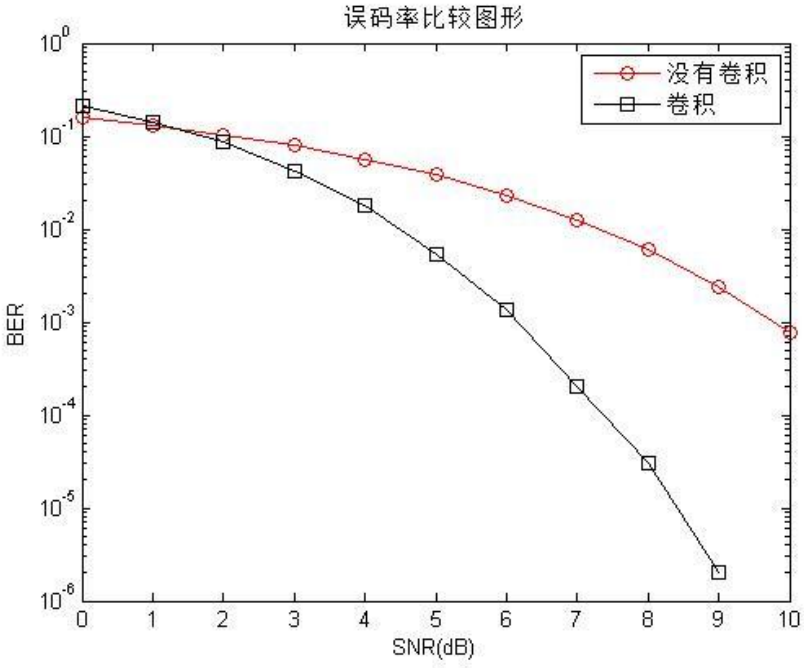

图 4-6　误码率比较图形

4.5 判决方式对卷积码误码性能的影响

由前文可知，Vibiter 译码有硬判决和软判决之分，采用软判决即将信道输出的编码先不立即判决，而是进行量化。判决方式对误码性能的 MATLAB 程序参见附录 5。

再进行维特比软判决译码。从图 4-7 可知，采用软判决的误码率明显低于硬判决，实际上软判决译码算法的路径度量采用“软距离”而不是汉明距离。最常采用的是欧几里德距离，也就是接收波形与可能的发送波形之间的几何距离。在采用软距离的情况下，路径度量的值是模拟量，需要经过一些处理以便于相加和比较。因此，使计算复杂度有所提高。

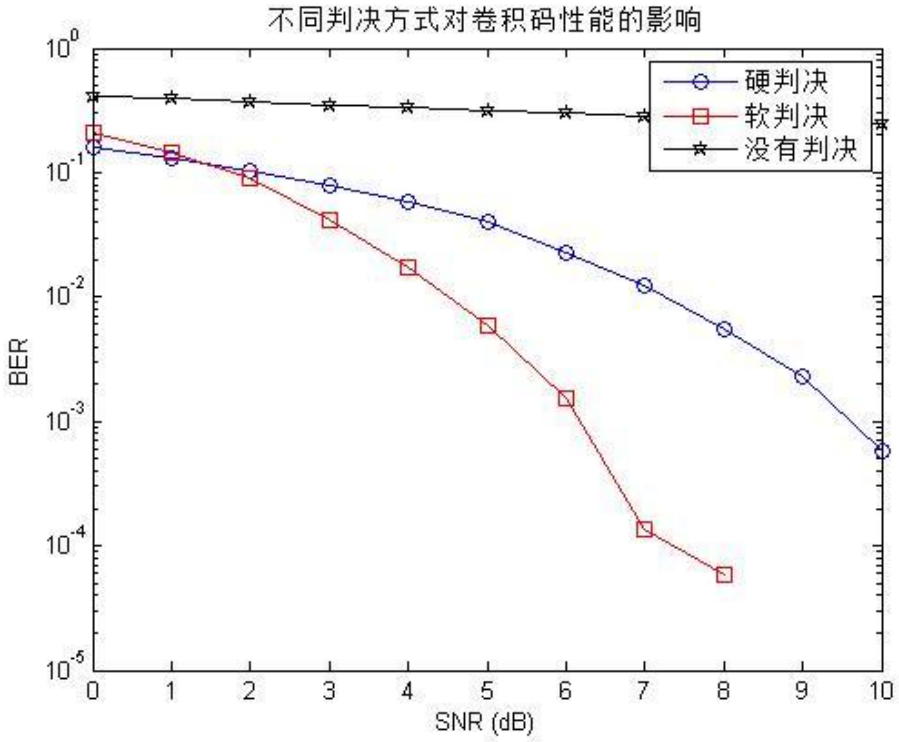

图 4-7　不同判决方式对卷积码性能的影响

5 总 结

5.1 结论

本文采用了 MATLAB 为工具，分别对线性分组码和卷积码进行了 MATLAB 仿真。对于线性分组码，从它的定义、生成矩阵和校验矩阵、纠错能力等方面入手，论述了线性分组码的编译码原理，又运用 M 函数对其编码译码进行了 MATLAB 仿真。以汉明码为例，构建了整个的通信系统，通过产生信号源、汉明码编码、调制、高斯白噪声信道（二进制对称信道）、汉明码译码、解调这一系列过程，分析了在这两种信道中的误码率性能。

在高斯白噪声信道中，当信噪比越来越大时，误码率越来越低，到一定程度，信道的误码率会为 0。仿真结果和理论结果基本一致。

在二进制对称信道中，通过比较模拟后的误码率曲线和分析的误码率曲线，可以得出概率误差最大，二进制对称信道的误码率就越高。

而对于卷积码，首先引入卷积码的一些基本理论，接着讨论了卷积码的编译码基本原理，对卷积码的表示方法和编译码算法做了简单的介绍，再通过编写卷积码的编码和解码程序，并且用 MATLAB 仿真软件进行仿真和纠错验证，并通过对不同信噪比、不同判决方式情况下误码率的分析，得出以下结论：

当信噪比较小时，未编码的误码率反而比采用编码的误码率低，这是因为误比特太多导致接收到的信息几乎无效。但是当信噪比稍高后卷积码编码译码的误码率就大大降低了。

对于码率一定的卷积码，采用的译码判决方式不同，也会对系统的误码率产生影响，一般来说，软判决译码性能要优于硬判决译码性能，但这是以提高设备复杂性为代价的。

5.2 展望

通过此次的毕业设计，我学到了很多知识，在论文的写作过程中，通过查资料和搜集有关的文献，培养了自学能力和动手能力。并且由原先的被动的接受知识转换为主动的寻求知识，这可以说是学习方法上的一个很大的突破。在以往的传统的学习模式下，我们可能会记住很多的书本知识，但是通过毕业设计，学会了如何将学到的知识转化为自己的东西，学会了怎么更好的处理知识和实践相结合的问题。尤其是对 MATLAB 这一强大语言的学习，不但让我掌握了 MATLAB 的基础理论知识，还提高了我对 MATLAB 的实际操作运用。在整个毕业设计过程中，我亲身感受到实际运用起来真是十分困难，需

要克服很多很多。有些方面还是不能够完美的做出来。所以我要更加锻炼使得理论与实践相结合，为进一步学习 MATLAB 打下坚实的基础；同时由于时间和个人能力等方面的不足，课题研究中还有些地方做的不到位，也有些领域没有研究，值得进一步完善。

信道编码自提出以来就受到人们的重视，并显现出了巨大的优越性，随着通信技术的进步，信道编码也将得到更大的发展，将提出更多形式的新型码字，进一步优化通信系统的性能，为通信行业的发展做出重大贡献。

致　谢

时光匆匆，转眼大学四年的学习生活就要走到尽头。回想大学四年的生活少不了酸甜苦辣，和同学一起度过的欢乐一起讨论学习的时光也将成为记忆。大学四年让我不论在思想、生活、学习上都得到了很大的成长，我要感谢同学和老师在学习和生活中对我的帮助，这四年是充实的四年，在以后的日子里这将是一段美好并值得怀念的记忆。

在此我要非常感谢在我的论文写作过程中给我提供支持和帮助的老师和同学们。特别感谢我的导师贾利琴老师在我的论文撰写过程中，从论文的选题、拟定提纲、构思、初稿的修改直至最后的定稿，倾注了大量的心血，提出了众多宝贵意见和建议，给予了精心的指导，让我的论文得以最终完稿。贾老师的严谨治学态度、深厚的学识和悉心的教导让我受益很多，在此谨表示我深深的谢意。同时还要感谢我的同学，没有他们的支持及帮助，我是无法尽心尽力的进行毕业设计工作的。在和她们的沟通和讨论中，我解决了一个个问题，打消了种种疑惑，战胜了重重困难，从而可以更好地进行毕业设计。事实证明了集体的智慧的强大!尤其是宋兰兰同学和李方方同学，她们给了我许多帮助，让我获益匪浅，在此由衷地感谢她们。

毕业在即，感慨颇多，也希望通过这一次认真的毕业设计来对我本科四年生活做一个完美总结。在毕设的过程，才了解到自身还有很多的不足，有幸能够再次跨进大学校园开始研究生生活，以后必定努力学习专业知识，提高自身技能。

参考文献

[1] 刘东华，向良军. 信道编码与 MATLAB 仿真. 北京:电子工业出版社，2013

[2] 张永光. 信道编码及其识别分析. 北京:电子工业出版社，2010

[3] 杜宇峰. 信道编码分析识别技术研究. 西安电子科技大学硕士论文, 2012

[4] 刘玉君. 信道编码. 郑州:河南科学技术出版社， 2006

[5] 樊昌信，曹丽娜. 通信原理. 北京:国防工业出版社， 2006

[6] 王育民，李晖. 信息论与编码理论. 北京:高等教育出版社，2005

[7] 赵鸿图. 通信原理 MATLAB 仿真教程. 北京:人民邮电出版社， 2010

[8] 王新梅，肖国镇. 纠错码. 西安:西安电子科技大学出版社， 2009

[9] 邓华. Matlab 通信仿真及应用. 北京:人民邮电出版社， 2000

[10] 孙祥，徐流美，吴清. MATLAB 基础教程. 北京:清华大学出版社， 2005

[11] 吴伟陵. 信息处理与编码. 北京:北京邮电大学出版社， 2006

[12] 曹雪虹，张宗橙. 信息论与编码. 北京:清华大学出版社， 2011

[13] 王立宁等. MATLAB 与通信仿真. 北京:人民邮电出版社， 2010

[14] 邵玉斌. MATLAB/simulink 通信系统建模与仿真实例分析（第一版）. 北京:清华大学出版社， 2008

[15] Viterbi A J.Error bounds for convolutional codes and an asymptotically optimum decoding algorithm[J].IEEE Trans.Inform Theory,1967,IT-IS(2):260-269

[16] J un Tan. Gordon L. Stüber. New SISO decoding algo2 rithms[J] . IEEE Trans. Commun , 2003 , 51 (6) : 845～848

[17] 仇佩亮. 信息论与编码. 北京:高等教育出版社， 2008

[18] 张忠培，史治平，王传丹. 现代编码理论与应用. 北京:人民邮电出版社， 2007

[19] C. Berrou, A. Glavieux, and P. Thitimajshima. “Near shannon limit error correctioncoding and decoding: Turbo codes” . in Proc. IEEE Int. Conf. Commun. , pp.1064–1070, 1993

附录 1

```
%%%产生随机信源信号
m=randint(1,10000);
x=reshape(m,length(m)/4,4);
%%%分组编码
G=[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];%设置编码矩阵
X=mod(x*G,2);%进行编码，生成编码后的码组
X=X';
n=length(m)/4*7;
X=reshape(X,1,n);
X=bi2de((reshape(X,2,n/2))','left-msb');%二进制数组转十进制
%%%产生 QPSK 信号
h=modem.pskmod(4);% 产生调制句柄 ,相位偏移默认为 0
y=modulate(h,X);%对信号进行调制
P=[];
%%%%加入高斯白噪声
for SNR=0:10
z=awgn(y,SNR);
g=modem.pskdemod(4);
%%%QPSK 解调
z=demodulate(g,z);
z=de2bi(z,2,'left-msb');
z=z';
z=reshape(z,1,n);
z=reshape(z,7,n/7);
z=z';
%%%解码
X=decode(z,7,4,'linear',G);
%%%计算误码率
[N,p]=biterr(x,X);
P=[P p];
```

```
end
%%%绘制误码率图形
SNR=0:10;
semilogy(SNR,P,'linewidth',2);
title('汉明码在 AWGN 的误码率')
xlabel('SNR(dB)');
ylabel('BER');
axis([0 10 10^-3 1]);
```

附录 2

```
k = 4;                          %定义变量(7,4)汉明码模拟
n = 7;
p_vector = 0.1:0.01:1;          % 向量 P 的值，误码率
N = length(p_vector);           % p_vector 的长度
RUNS = 5000;                    % 运行数
xtable =    [0 0 0 0 0 0 0; ...     % 码字表
             1 1 0 1 0 0 0; ...
             0 1 1 0 1 0 0; ...
             1 0 1 1 1 0 0; ...
             1 1 1 0 0 1 0; ...
             0 0 1 1 0 1 0; ...
             1 0 0 0 1 1 0; ...
             0 1 0 1 1 1 0; ...
             1 0 1 0 0 0 1; ...
             0 1 1 1 0 0 1; ...
             1 1 0 0 1 0 1; ...
             0 0 0 1 1 0 1; ...
             0 1 0 0 0 1 1; ...
             1 0 0 1 0 1 1; ...
             0 0 1 0 1 1 1; ...
             1 1 1 1 1 1 1;];
for (p_i=1:N)
    error = 0;                  % 错误的数
    p=p_vector(p_i);
    for (r=1:RUNS)              %生成 4 位的信息块
    z = unifrnd(0, 1, 1, 4);    %0 和 1 的 4 位串
    w = round(z);               %圆的 Z 值
    m = w(1) + w(2)*2 + w(3)*4 + w(4)*8;    %找到行的索引，二进制转换：
    x = xtable(m + 1, :);
    z = unifrnd(0, 1, 1, 7);    % 0 和 1 的 7 位随机字符串
```

```
zi = find(z <= p);                    %错误的位置


% 误比特率
    e = zeros(1,7);                   % 创建一个 0 的 7 位串
    e(zi) = ones(size(zi));           % 创建一个字符串的大小 Zi
    y = xor(x,e);                     % 异或 X 和 E
        for(q=1:16)
        dH(q) = sum(xor(y, xtable(q,:)));     % 比较接收到的码字矢量
            if(dH(q)<=1)
            wh = xtable(q, 4:7);
            end
        end


% 计算误码率
            dHw = sum(xor(w,wh));
            error = error + dHw;
    end
    BER(p_i) = error/(RUNS*4);        % 误码率计算
    P(p_i)=p;                         % 存到 p
end
Ps=logspace(-4,0,200);
Pb_high = 1 - ((1-Ps).^7 + 7.*(1-Ps).^6.*Ps);
Pb_low = (1 - ((1-Ps).^7 + 7.*(1-Ps).^6.*Ps))/k;
figure(1)
plot(P, BER, 'bx', Ps, Pb_high, 'k-', Ps, Pb_low, 'k-')
legend('模拟','分析')
xlabel('误差概率 p')
ylabel('BER')
title('汉明码在 BSC 的误码率')
```

附录 3

(1)

```
function y=bin2deci(x)
l=length(x);
y=(l-1:-1:0);
y=2.^y;
y=x*y';
```

(2)

```
function y=deci2bin(x,l)
y=zeros(1,l);
i=1;
while x>=0 & i<=l
    y(i)=rem(x,2);
    x=(x-y(i))/2;
    i=i+1;
end
y=y(l:-1:1);
```

(3)

```
function distance=metric(x,y)
if x==y
    distance=0;
else
    distance=1;
end
```

(4)

```
function [next_state,memory_contents]=nxt_stat(current_state,input,L,k)
binary_state=deci2bin(current_state,k*(L-1));
binary_input=deci2bin(input,k);
next_state_binary=[binary_input,binary_state(1:(L-2)*k)];
next_state=bin2deci(next_state_binary);
memory_contents=[binary_input,binary_state];
```

(5)

```
function [decoder_output,survivor_state,cumulated_metric]=viterbi(channel,snr_db)
G=[1 1 1;1 0 1];          % G  卷积编码矩阵，如（2,1,3)卷积码生成矩阵[1 1 1;1 0 1]
k=1;                      % k  信息源输入端口数   k=1
channel=[1 1 0 1 0 1 0 0 1 0 1 1 ];   %信源编码
snr_db=6;                 %信噪比，可以通过调节信噪比大小观察 viterbi 译码的性能
%bpsk 调制
channel_output=bpsk(channel,snr_db);          %调用 bpsk 函数,得到信道编码
n=size(G,1);                                  % n  编码输出端口数量,(2,1,3)中 n=2
if rem(size(G,2),k)~=0                        %当 G 列数不是 k 的整数倍时
    error('Size of G and k do not agree')     %发出出错信息
end
if  rem(size(channel_output,2),n)~=0          %当输出量元素个数不是输出端口的整数倍
时
    error('channel output not of the right size')
end
N=size(G,2)/k;                    %得出移位数，即寄存器的个数
M=2^k;
number_of_states=2^(k*(N-1));     %状态数
for j=0:number_of_states-1        %j 表示当前寄存器组的状态因为状态是从零
                                  %开始的,所以循环从 0 到 number_of_states-1
    for m=0:M-1                   %m 为从 k 个输入端的信号组成的状态，总的状
                                      %态数为 2^k，所以循环从 0 到 2^k-1
        %nxt_stat 完成从当前的状态和输入的矢量得出下寄存器组的一个状态
        [next_state,memory_contents]=nxt_stat(j,m,N,k);%调用 nxt_stat 函数
        input(j+1,next_state+1)=m;
        branch_output=rem(memory_contents*G',2);
        nextstate(j+1,m+1)=next_state;
        output(j+1,m+1)=bin2deci(branch_output);
    end
end
```

```
%       state_metric 数组用于记录译码过程在每状态时的汉明距离
%       state_metric 大小为 number_of_states 2，（:,1）当前
%       状态位置的汉明距离，为确定值，而（:,2）为当前状态加输入
%       得到的下一个状态汉明距离，为临时值
state_metric=zeros(number_of_states,2);
depth_of_trellis=length(channel_output)/n;
channel_output_matrix=reshape(channel_output,n,depth_of_trellis);
survivor_state=zeros(number_of_states,depth_of_trellis+1);
for i=1:depth_of_trellis-N+1
    flag=zeros(1,number_of_states);
    if(i<=N)
        step=2^(k*(N-i));
    else
        step=1;
    end
    for j=0:step:number_of_states-1
        for m=0:M-1
            branch_metric=0;
            binary_output=deci2bin(output(j+1,m+1),n);
            for ll=1:n

branch_metric=branch_metric+metric(channel_output_matrix(ll,i),binary_output(ll))
            end
        %选择码间距离较小的那条路径
        %选择方法：
        %当下一个状态没有被访问时就直接赋值，否则，用比它小的将其覆盖
if((state_metric(nextstate(j+1,m+1)+1,2)>state_metric(j+1,1)+branch_metric);
flag(nextstate(j+1,m+1)+1)==0
state_metric(nextstate(j+1,m+1)+1,2)=state_metric(j+1,1)+branch_metric;
                survivor_state(nextstate(j+1,m+1)+1,i+1)=j;
                flag(nextstate(j+1,m+1)+1)=1;
```

```
                end
            end
        end
        state_metric=state_metric(:,2:-1:1);
end
for i=depth_of_trellis-N+2:depth_of_trellis
        flag=zeros(1,number_of_states);
      %状态数从 number_of_states→number_of_states/2→...→2→1
      %程序说明同上，只不过输入矢量只为 0
        last_stop=number_of_states/(2^(k*(i-depth_of_trellis+N-2)));
        for j=0:last_stop-1
            branch_metric=0;
            binary_output=deci2bin(output(j+1,1),n);
            for ll=1:n

branch_metric=branch_metric+metric(channel_output_matrix(ll,i),binary_output(ll))
            end
            if(     (state_metric(nextstate(j+1,1)+1,2)>state_metric(j+1,1)+branch_metric)     |
flag(nextstate(j+1,1)+1)==0 )
                state_metric(nextstate(j+1,1)+1,2)=state_metric(j+1,1)+branch_metric;
                survivor_state(nextstate(j+1,1)+1,i+1)=j;
                flag(nextstate(j+1,1)+1)=1;
            end
        end
        state_metric=state_metric(:,2:-1:1);
end
%从最佳路径中产生解码
%译码过程可从数组 survivor_state 的最后一个位置向前逐级译码
state_sequence=zeros(1,depth_of_trellis+1);
state_sequence(1,depth_of_trellis)=survivor_state(1,depth_of_trellis+1);
for i=1:depth_of_trellis
```

```
state_sequence(1,depth_of_trellis-i+1)=survivor_state((state_sequence(1,depth_of_trellis+2
-i)+1),depth_of_trellis-i+2);
end
decoder_output_matrix=zeros(k,depth_of_trellis-N+1);
for i=1:depth_of_trellis-N+1
    %根据数组 input 的定义来得出从当前状态到下一个状态的输入信号矢量
    dec_output_deci=input(state_sequence(1,i)+1,state_sequence(1,i+1)+1);
    dec_output_bin=deci2bin(dec_output_deci,k);
    %将一次译码存入译码输出矩阵 decoder_output_matrix 相应的位置
    decoder_output_matrix(:,i)=dec_output_bin(k:-1:1)';
end
decoder_output=reshape(decoder_output_matrix,1,k*(depth_of_trellis-N+1));
cumulated_metric=state_metric(1,1);
```

附录 4

```
clear all;
clc;
snr_db = 0:10;
%信息源
msg = randint(1,1e3);
ber0 = zeros(1,length(snr_db));
ber1 = zeros(1,length(snr_db));
% Trellises
trellis = poly2trellis(3,[7 5]); %Define trellis for rate 1/2 code.
for n=1:length(msg)/2;
      for x=1:length(snr_db)
% Code words
code = convenc(msg,trellis); % Encode.
state = 20;
inter = randintrlv(code,state);
% BPSK  调制
s0 = sign(msg-0.5);
s1 = sign(inter-0.5);
% AWGN Channel
add_noise0=awgn(s0,snr_db(x),'measured');
add_noise1=awgn(s1,snr_db(x),'measured');
% Deinterleaver with noise for soft decoding
deinter_noise = randdeintrlv(add_noise1,state);
%  解调
r_0 = 0.5*sign(add_noise0) + 0.5;
r_1 = 0.5*sign(add_noise1) + 0.5;
% Deinterleaver
deinter_1 = randdeintrlv(r_1,state);
% Traceback length
tblen = 5;
```

```
% vitdec 判决
decoded1 = vitdec(deinter_1,trellis,tblen,'cont','hard');    % vitdec 判决
%  比较误码率
[num0,rat0] = biterr(r_0,msg);
[num1,rat1] = biterr(double(decoded1(tblen+1:end)),msg(1:end-tblen));
ber0(n,x) = rat0;
ber1(n,x) = rat1;
        end
end
ber0 = mean(ber0);
ber1 = mean(ber1);
semilogy(snr_db,ber0,'r-o');
hold on
semilogy(snr_db,ber1,'k-s');
xlabel('SNR(dB)');
ylabel('BER');
legend('没有卷积','卷积');
title('误码率比较图形');
```

附录 5

```
clear all;
clc;
cycl = 50;
snr_db = 0:1:10;
% 输入信息
msg = randint(1,1024);
ber0 = zeros(cycl,length(snr_db));
ber1 = zeros(cycl,length(snr_db));
ber2 = zeros(cycl,length(snr_db));
% Trellises
trel = poly2trellis(3,[5 7]); %Define trellis for rate 1/2 code.
for n = 1:cycl
for x = 1:length(snr_db)
% Code words
code = convenc(msg,trel); % Encode.
% Interleaver
state = 20;
inter = randintrlv(code,state);
% BPSK 调制
s0 = sign(msg - 0.5);
s1 = sign(inter-0.5);
s2 = sign(code-0.5);
% AWGN Channel
add_noise0=awgn(s0,snr_db(x),'measured');
add_noise1=awgn(s1,snr_db(x),'measured');
add_noise2=awgn(s2,snr_db(x),'measured');
% Deinterleaver with noise for soft decoding
deinter_noise = randdeintrlv(add_noise1,state);
% 解调
r_0 = 0.5*sign(add_noise0) + 0.5;
```

```
r_1 = 0.5*sign(add_noise1) + 0.5;
r_2 = 0.5*sign(add_noise2) + 0.5;
% Deinterleaver
deinter_1 = randdeintrlv(r_1,state);
% Traceback length
tblen = 5;
% vitdec  硬判决
decoded1 = vitdec(deinter_1,trel,tblen,'cont','hard');
% vitdec  软判决
[y,qcode] = quantiz(deinter_noise,[-.75 -.5 -.25 0 .25 .5 .75],7:-1:0);
decoded2 = vitdec(qcode,trel,tblen,'cont','soft',3);
%  比较误码率
[num0,rat0] = biterr(r_0,msg);
[num1,rat1] = biterr(double(decoded1(tblen+1:end)),msg(1:end-tblen));
[num2,rat2] = biterr(double(decoded2(tblen+1:end)),msg(1:end-tblen));
ber0(n,x) = rat0;
ber1(n,x) = rat1;
ber2(n,x) = rat2;
end
end
ber0 = mean(ber0);
ber1 = mean(ber1);
ber2 = mean(ber2);
semilogy(snr_db,ber0,'b-o',snr_db,ber1,'r-s',snr_db,ber2,'k-p');
xlabel('SNR (dB)');
ylabel('BER');
legend('硬判决','软判决','没有判决');
title('不同判决方式对卷积码性能的影响');
```