密级：_____

中国科学院研究生院

硕士学位论文

水声通信中 Turbo 均衡技术的研究

作者姓名：唐怀东

指导教师：朱敏　研究员　中国科学院声学研究所

武岩波　副研究员　中国科学院声学研究所

学位类别：工学硕士

学科专业：信号与信息处理

培养单位：中国科学院声学研究所

2013 年 5 月

Research of Turbo Equalization for Underwater Acoustic Communication System

By

Tang Huaidong

A Dissertation Submitted to

Graduate University of Chinese Academy of Sciences

In partial fulfillment of the requirement

For the degree of

Master of Signal and Information Processing

Institute of Acoustics, Chinese Academy of Sciences

May, 2013

中国科学院声学研究所

学位论文原创性声明和使用授权说明

原创性声明

本人郑重声明：本论文的所有工作，是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不含任何其他个人或集体已经发表或撰写过的作品或成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本声明的法律结果由本人承担。

作者签名：

日期：　　年　　月　　日

学位论文使用授权说明

本人完全了解中国科学院研究生院关于收集、保存、使用学位论文的规定，即：

- 按照中国科学院研究生院要求提交学位论文的印刷本和电子版本；
- 中国科学院研究生院与中国科学院声学研究所有权保存学位论文的印刷本和电子版，并提供目录检索与阅览服务；
- 中国科学院研究生院与中国科学院声学研究所可以采用影印、缩印、数字化或其它复制手段保存论文；

（保密论文在解密后遵守此规定）

作者签名:　　导师签名：

日期：　　年　　月　　日

摘　要

在低信噪比的水声通信系统中，采取卷积码和 RS 码来提高信息传送的可靠性。对于卷积码，采用序贯译码方式，来简化译码算法，提高译码效率。而费诺算法是序贯译码方式中空间开支小，运算量小，实现比较简单，现实应用比较广泛的一种算法。而对于 RS 码，采用 BM 算法进行译码。

通过 matlab 仿真，确定译码参数，并在 PC/104 上运行，测试效率。

通过测试得到的数据，可以得到几个结论。卷积码序贯译码在约束长度为 24 的情况下，误码率可以很好的满足要求，与书上的理论相一致，而运行时间有一定的延迟，因此系统需要缓冲器来存储译码数据；而 $RS(15,9,7)$ 采用 BM 译码算法，误码率有很大的改善，而且运行时间很短，因此系统不需要缓冲器来存储译码数据。

关键词

低信噪比　水声通信　卷积码　RS 码

ABSTRACT

In the low signal to noise ratio of acoustic communication system, convolutional code and RS code are adopted to improve the reliability of information transmission.For convolutional code, using sequential decoding method to simplify the decoding algorithm and to improve the efficiency. The fano algorithm is one of the sequential decoding methods which is small space spending, relatively simple to achieve and used widely in reality. So, finally taking it as the decoding algorithm. For RS code, BM algorithm is almost the most widely one among all decding algorithms.

Determining the coding parameters by matlab simulation and test decoding efficiency in PC/104.

Through test data, several conclusions can be obtained. With sequential decoding algorithm, the error of convolutional code which constraint length is 24 can be very good to meet the requirements and be consistent with the theory of the book, also the running time is with delay. Therefore, the system needs to a buffer to store the decoding data; The $RS(15, 9, 7)$with BM decoding algorithm, the error rate has been greatly improved and the running time is very short. So the system does not need to buffer the decoding data.

Keywords

Low SNR Underwater acoustic communication Convolutional code RS code

目录

第一章　引言

1.1　选题的背景与意义

经历了农业时代，工业时代，信息时代，人类发现自己研究的内容始终在陆地上，而神秘且广袤的海洋却涉足不深，因此，有人提出 21 世纪是海洋的世纪，这说明对海洋的研究已经开始进入人们的视线，而且也越来越深入。

海洋占据地球表面的 70%，蕴涵了丰富的资源，能源，是维持人类社会可持续发展的后勤保障和空间保障，对人类的发展和社会的进步起着至关重要的作用。而且，近些年来，明显感觉到各海洋国家对海洋探测，资源开发的重视，不论是人力财力的投入。

这些都说明对海洋的信息获取和处理的重要性，因此，要构建一个能实时监控海洋的系统。

那么，通信必然成为这个系统非常重要的环节。迄今为止，人类的研究和探测表明，在所有传输载体中，适合海洋通信的只有声波。因为在浑浊、含盐以及各种矿物质比例比较高的海水中，无论我们在陆地通信中常用的光波还是无线电波，它们的传播衰减都是非常大的，因此，在海水中的传输距离非常有限，根本无法满足人类对于海洋活动传播距离的要求，而相比之下，声波却可以在海水中传输的很好，满足要求。这也是为什么，在海洋通信中，我们一直研究水声通信技术，而不是其他的，通过水声通信，测试仪器可以把探测的数据直接传输到监视端，从而实现实时监测的目的。

水声通信的介质是海水，介于海水的时变性，随机性，复杂性，为了保障通信质量，提高可靠性，我们需要对传输信息进行信道编码，降低误码率。因此，信道编码必是一种研究方向。

1.2　水声通信技术发展概况

水声通信是一项在水下收发信息的技术，这个领域涉及到水声学、通信技术、信号处理、电子技术和计算机技术等多种学科。

人类关于水下发送、接收信息的想法可以追溯到几百年前。早在 1490 年，意

大利的达·芬奇在他的摘记中就有“将长管的一段插入水中，将管的开口放在耳旁就听到远处的航船”的记载。而最具现代意义的真正的水声通信出现在第二次世界大战中，主要用于军事上。例如 45 年美国开发的用于潜艇间通信的水声电话，采用单边带[1]调制技术。现在随着硬件的高速发展，尤其是半导体技术的进步，使得过去很多被认为过于复杂，运算量过于繁重的算法得到逐步的应用；同时，对新的算法发展和研究提供了硬件保障。

水声通信系统的工作原理是这样的：首先将文字、语音、图像等信息，通过电发送机转换成电信号，并由编码器将这些信息进行数字化处理后，换能器又将这些电信号转换成声信号。声信号通过水这一介质，将信息传递到接收换能器。这时声信号又转换为电信号，解码器将数字信息破译后，电接收机再将信息复原为文字、语音、图像。

换能器是一种能将声能和电能相互转换的仪器。有了它，人们就可以在空气中、水中、固体中任意发射和接收不同频率、不同强度的声信号。

最先，水声通信机使用的是模拟信号，可是海洋中的波浪、鱼类、舰船等产生噪声，使海洋中的声场极为混乱，声波在海水中传递时产生“多径干扰信号”这一较大的难题，导致接收到的信号模糊不清。

但是，近代由于数字通信的产生，陆地上的信号干扰被成功解决，水声领域的专家也开始了在该领域进行探索。

因为海水成分很复杂，所以声波传递时被吸收了一部分，而且频率越高吸收就越厉害，对于频率越低的声波，被海水吸收反而越少。专家测得结果，声波频率在 4KHZ 左右为远距离传递的最佳频率。

水下通信非常困难，主要是由于通道的多径效应、时变效应、可用频带窄、信号衰减严重，特别是在长距离传输中。水下通信相比于有线、无线通信来说速率非常低，因为水下通信采用的是声波而非无线电波。常见的水声通信方式是采用扩频通信技术，如 CDMA 等。

目前，水声通信技术发展的已经较为成熟，国外很多机构都已研制出水声通信 Modem，通信方式主要有：OFDM，扩频以及其它的一些调制方式。此外，现在水声通信技术已发展到网络化的阶段，将无线电中的网络技术 (AdHoc) 应用到水

[1]SSB,single sideban

声通信网络中，可以在海洋里实现全方位、立体化通信 (可以与 AUV、UUV 等无人设备结合使用)，但目前只有少数国家实验成功。

1.3 信道编码的发展概况

数字信号在传输中往往由于各种原因，使得在传送的数据流中产生误码，从而使接收端产生图象跳跃、不连续、出现马赛克等现象。所以通过信道编码这一环节，对数据码流进行相应的处理，使系统具有一定的纠错能力和抗干扰能力，可极大地避免码流传送中误码的发生。误码的处理技术有纠错、交织、线性内插等。

提高数据传输效率，降低误码率是信道编码的任务。信道编码的本质是增加通信的可靠性。但信道编码会使有用的信息数据传输减少，信道编码的过程是在源数据码流中加插一些码元，从而达到在接收端进行判错和纠错的目的，这就是我们常常说的开销。这就好象我们运送一批玻璃杯一样，为了保证运送途中不出现打烂玻璃杯的情况，我们通常都用一些泡沫或海棉等物将玻璃杯包装起来，这种包装使玻璃杯所占的容积变大，原来一部车能装 5000 各玻璃杯的，包装后就只能装 4000 个了，显然包装的代价使运送玻璃杯的有效个数减少了。同样，在带宽固定的信道中，总的传送码率也是固定的，由于信道编码增加了数据量，其结果只能是以降低传送有用信息码率为代价了。将有用比特数除以总比特数就等于编码效率了，不同的编码方式，其编码效率有所不同。

1. **线性分组码** 线性分组码中的线性是指码组中码元间的约束关系式线性的，而分组则是对编码方法而言，即编码时将每 k 个信息位分为一组进行独立处理，变成长度为 $n(n > k)$ 的二进制码组。

   循环码是线性分组码中最主要、最有用的一类，目前对它的研究和应用也很多。它是 1957 年由 Prange 首先提出并进行研究的。循环码最引人注目的特点是：首先它可以用线性反馈移位寄存器很容易地实现其编码和伴随式计算，其次由于循环码有很多固有的代数结构，从而可以找到各种简单实用的译码方法。由于循环码具有很好的良好性质，所以它在理论和实践中都是很重要。在循环码中 BCH 码是其中最主要的一大类。汉明码、R-M 码、

Golay 码、R-S 码等均可变换成或者纳入循环码内，1970 年发现的 Goppa 码类中有一子类也属于循环码。

2. **卷积码** 卷积码是 1955 年由 Elias 等人提出的，是一种非常有前途的编码方法。卷积码将 k 个信息比特编成 n 个比特，但 n 和 k 通常很小，特别适合以串行形式进行传输，时延小。与分组码不同，卷积码编码后的 n 个码元不仅与当前段的 k 个信息有关，还与前面的 $N-1$ 段信息有关，编码过程中相互关联的码元个数为 nN。卷积码的纠错性能随 N 的增加而增大，而差错率随 N 的增加而指数下降。在编码器复杂性相通的情况下，卷积码的性能优于分组码。

3. **Turbo 码** 1993 年法国人 Berrou 等在 ICC 国际会议上提出了一种采用重复迭代 (Turbo) 译码方法的并行级联码，并采用软输入/输出译码器，可以获得接近 Shannon 极限的性能，至少在大的交织器和 $BER \cong 10^{-5}$ 条件下，可以达到这种性能。Turbo 码的优良性能，受到移动通信领域的广泛重视，特别是在第三代移动通信体制中，非实时的数据通信广泛采用 Turbo 码。

1.4 论文的研究内容及章节安排

本论文主要研究低信噪比下的水声通信的信道编码技术。主要包括卷积码编码和序贯译码算法，RS 码编码和 BM 译码算法，通过 matlab 仿真确定参数和分析误码率曲线，最后在 PC/104 上运行。

论文由六章组成，各章内容安排如下：

- **第一章 绪论** 主要介绍论文研究内容的现实意义已经国内外水声通信技术和信道编码技术发展概况。

- **第二章 信道编码代数基础** 包括信道编码的基础代数知识，主要包括近世代数里面的群、域、矢量空间、矩阵等。

- **第三章 卷积码的编码与序贯译码算法** 主要是卷积码序贯译码算法理论基础分析，以及 C 语言实现流程，译码算法以费诺算法为主要研究对象。

- **第四章 RS 码的编码与 BM 硬判决算法** 主要是 RS 码译码算法理论基础分析，以及 C 语言实现流程，译码算法以 BM 算法为主要研究对象。
- **第五章 PC/104 运行分析** 主要是把编好的程序在 PC/104 上运行，观察译码效率，和误码率情况。
- **第六章 结论与展望** 介绍论文的研究结论和未来的研究工作

第二章　信道编码代数基础

2.1　群[1]

令 **G** 是一个集合，现规定 **G** 上的二元运算 "$*$" 的规则：

对 **G** 中的每一对元素 a 和 b，在 **G** 中指定一个唯一确定的第三个元素 $c = a * b$。当这样的二元运算 "$*$" 定义在 **G** 上时，我们就称在 "$*$" 运算下 **G** 是封闭的。若对 **G** 中的任意元素 a, b, c 有

$$a * (b * c) = (a * b) * c \tag{2-1}$$

则称 **G** 上的二元运算 "$*$" 是结合的。

定义 2.1：设 **G** 是非空集合，并在 **G** 上定义了一种运算 "$*$"，如果满足以下条件就称做群：

1．**满足封闭性**　若 a 和 b 为集合 **G** 中的任意元素，即 $a \in \mathbf{G}, b \in \mathbf{G}$, 恒有

$$a * b = c \in \mathbf{G} \tag{2-2}$$

2．**满足结合律**　对任意 $a \in \mathbf{G}, b \in \mathbf{G}, c \in \mathbf{G}$,

$$a * (b * c) = (a * b) * c \tag{2-3}$$

3．**存在恒等元**　对任意 $a \in \mathbf{G}$，有

$$a * a^{-1} = e \tag{2-4}$$

其中，$a^{-1} \in \mathbf{G}$，且称为 a 的逆元素。

例如，整数中，任意两个整数相加还是一个整数，以此满足封闭性；显然也满足结合律；任意一个非零整数 Z 的逆元素是 $-Z$，$Z + (-Z) = 0$, 所以恒等元是 0；则整数在实数相加下是一个群。但整数在实数乘法下就不能构成一个群。

若群 **G** 中，$a \in \mathbf{G}, b \in \mathbf{G}$，有

$$a * b = b * a \tag{2-5}$$

则称群为可交换群或阿贝尔群。

定理 2.1:群 **G** 的恒等元是唯一的,每个元素的逆元素也是唯一的。

定理 2.2:令 **H** 是 **G** 的非空子集。若 **H** 在 **G** 的群运算下是封闭的且满足群的所有条件,就称 **H** 为 **G** 的子群。

例如,偶数是整数的一个非空子集。同样可以证明偶数在实数加法下也是一个群,所以偶数时整数的一个子群。

定理 2.3:群中元素的个数称为群的阶。

2.2 域[1]

域就是一个集合,在其中可以进行加、减、乘、除而不会超出该集合。加法和乘法都必须满足交换律、结合律和分配律,正式定义:

定义 2.2:令 **F** 是一个集合,其上定义了两个二元运算,称作加法 "+" 和乘法 "·"。满足下述条件时,就称集合 **F** 和两个运算 "+" 和 "·" 是域:

1. **F** 关于加法运算构成阿贝尔群;

2. **F** 中的非零元素在乘法下构成阿贝尔群,其恒等元素以 1 表示;

3. 对加法和乘法分配律成立

$$a \cdot (b + c) = a \cdot b + a \cdot c \tag{2-6}$$

根据定义可得出,一个域至少由两个元素即加法恒等元素和乘法恒等元素组成。域中元素的个数叫做域的阶。有限个元素的域叫做有限域。在域中,元素 a 的加法逆元素用 $-a$ 表示,乘法逆元素用 a^{-1} 表示。域的一些基本性质可以从域的定义导出。

性质 1: 对域中的每个元素 $a \cdot 0 = 0 \cdot a = 0$

性质 2: 对域中任意两个非零元素 a 和 b，$a \cdot b \neq 0$

性质 3: $a \cdot b = 0$ 且 $a \neq 0$，这意味着 $b = 0$

性质 4: 对域中任意两个元素 a 和 b，$-(a \cdot b) = (-a) \cdot b = a \cdot (-b)$

性质 5: 对 $a \neq 0, a \cdot b = a \cdot c$，可推出 $b = c$

一个重要的概念 **--素域**

令 p 为素数，不难证明，整数集 $\{0, 1, 2, \cdots, p-1\}$ 在模 p 加法下是阿贝尔群，非零集合 $\{1, 2, \cdots, p-1\}$ 在模 p 乘法下也构成阿贝尔群，由于模 p 加法和模 p 乘法是可分配的，所以集合 $\{0, 1, 2, \cdots, p-1\}$ 是阶为 p 的域。由于这个域由素数 p 构成，故称为素域并以 $GF(p)$ 表示。对于 $p = 2$，我们得到二元域 $GF(2)$。例如 $p = 7$，模 7 加法和模 7 乘法由表2-1和表2-2给出。整数集合 $\{0, 1, 2, 3, 4, 5, 6\}$ 在模 7 加法和模 7 乘法下是一个有 7 个元素的域，以 $GF(7)$ 表示。

表 2-1　模 7 加法

模 7 加	0	1	2	3	4	5	6
0	0	1	2	3	4	5	6
1	1	2	3	4	5	6	0
2	2	3	4	5	6	0	1
3	3	4	5	6	0	1	2
4	4	5	6	0	1	2	3
5	5	6	0	1	2	3	4
6	6	0	1	2	3	4	5

表 2-2　模 7 乘法

模 7 乘	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6
2	0	2	4	6	1	3	5
3	0	3	6	2	5	1	4
4	0	4	1	5	2	6	3
5	0	5	3	1	6	4	2
6	0	6	5	4	3	2	1

对于任何素数 p 都存在一个有 p 个元素的有限域。对于任何正整数 m，可以将素域 $GF(p)$ 扩展成有 p^m 个元素的域，称它为 $GF(p)$ 的扩域，并以 $GF(p^m)$ 表示。而且以证明，任意有限域的阶是素数的幂次。有限域以其发现者命名为伽罗华域。大部分代数编码理论，码的构造和译码都是围绕有限域建立的。

研究 q 个元素的有限域 $GF(q)$。因为在加法下域是封闭的，两个元素之和也是域中的元素，即必存在两个正整数 m 和 $n, m < n$，使

$$\sum_{i=1}^{k} 1 = k \neq 0 \, , \sum_{i=1}^{p} 1 = 0 \tag{2-7}$$

定理 2.4：有限域的特征 λ 是素数

定理 2.5：令 a 是有限域 $GF(q)$ 中的非零元素，则 $a^{q-1} = 1$。

定理 2.6：令 a 是有限域 $GF(q)$ 中的非零元素，令 n 是 a 的阶，则 $q-1$ 能被 n 除尽。

在有限域 $GF(q)$ 中，若非零元素 a 的阶是 $q-1$，就称 a 是本原的。所以，本原元素的幂次生成 $GF(q)$ 的所有非零元素，每个有限域都有本原元素。若在群中存在一个这样的元素，其各次幂构成整个群，就称该群是循环的。若 a 为循环群中的本原元素，使 $a^n = e$ 的最小整数 n 为循环群的级。

有限循环群的性质

性质 1： 若 $a \in \mathbf{G}$ 是 n 级元素，则 $a^m = e$ 的充要条件是 m 可以被 n 除尽。

性质 2： 若 a 是 n 级元素，则元素 a^k 的级为 $\frac{n}{(k,n)}$[1]。

2.3 伽罗华域[2]

在数字通信系统中用到的伽罗华域通常是二元域 $GF(2)$ 及其扩域 $GF(2^m)$，所以这里仅限于讨论二元域及扩域。

定义 2.3：不能分解因式的多项式称为既约多项式。若 $GF(2)$ 上的 m 次多项式 $f(x)$ 不能被 $GF(2)$ 上任何次数小于 m，但大于零的多项式除尽，就称它是 $GF(2)$ 上的既约多项式。

[1] (k,n) 表示 k 除以 n 后的余数

例 2.1 :$x^3+x+1=0$,由模 m 加法可得 $x^3=x+1, x=x^3+1$,若 α 是多项式的根,则 α 的所有幂次为:

表 2-3 $GF(2^3)$ 所有幂次

幂表达式	多项式表达式	矢量表达式
0	0	000
1	1	001
α^1	α^1	010
α^2	α^2	100
α^3	$\alpha+1$	011
α^4	$\alpha\cdot\alpha^3=\alpha\cdot(\alpha+1)=\alpha^2+\alpha$	110
α^5	$\alpha\cdot\alpha^4=\alpha\cdot(\alpha^2+\alpha)=\alpha^2+\alpha+1$	111
α^6	$\alpha\cdot\alpha^5=\alpha\cdot(\alpha^2+\alpha+1)=\alpha^2+1$	101
α^7	$\alpha\cdot\alpha^6=\alpha\cdot(\alpha^2+1)=1$	001

以上分析可知,α 的所有幂次共有七个非零元素,再加上 0 元素,构成了含有八个元素的域 $GF(2^3)$,它是 $GF(2)$ 的三次扩域。可见,x^3+x+1 是既约多项式。那么,是不是凡是二元域上的既约多项式的根都能构成 m 次扩域 $GF(2^m)$。为了说明这个问题,再看一个例子。

例 2.2 :如:$x^4+x^3+x^2+x+1$

$\alpha^0=1\Rightarrow 0001$

$\alpha^1=\alpha^1\Rightarrow 0010$

$\alpha^2=\alpha^2\Rightarrow 0100$

$\alpha^3=\alpha^3\Rightarrow 1000$

$\alpha^4=\alpha^3+\alpha^2+\alpha+1\Rightarrow 1111$

$\alpha^5=1\Rightarrow 0001$

$\alpha^6=\alpha\Rightarrow 0010$

我们发现,α 的所有幂次仅有五个元素,加上零元素共有 6 个,而不是 2^4 个,所以 $x^4+x^3+x^2+x+1$ 不能构成 $GF(2^4)$ 扩域。

此外，$2^{15}=1$ 表明 α 还是 $x^{15}+1$ 的根，因此，既约多项式 x^4+x+1 能够被多项式 $x^{15}+1$ 整除，而不能被其它次数小于 15 的多项式整除。

归纳：$GF(2^m)$ 上的 m 次既约多项式有两大类：一类是能被 x^n+1 整除，但不能被 x^s+1 整除，其中 $n=2^{m-1}, s<n$。它的根是 $GF(2^m)$ 扩域的本原元素，这一类多项式称为本原多项式。另一类既约多项式技能被 x^n+1 整除，又能被 x^s+1 整除，称为非本原多项式。只有本原多项式才能构成 $GF(2^m)$ 域。

伽罗华域上运算与实数域上的运算不同，其主要区别[2] 是：

1. 域上加减运算等同 (对加法特征为 2 的有限域)，加减运算以域元素的矢量形式进行，加减运算为域元素的矢量表示对应位模 2 加；
2. 域上元素间的乘除运算以本原元素的形式进行，乘除运算为本原元素的幂次模 (2^m-1) 加减；
3. 加法恒元乘任何数等于加法恒元；任何数与加法恒元相加，其值不变；
4. 任何数与乘法恒元相乘，其值不变；
5. 域元素的指数运算以域元素的指数形式进行，运算规则与实数域上相同，但结果需进行模 2^m-1 运算。

由上面分析可知，要实现域上运算，至少需要域元素的两种表示形式，即域元素的指数表示形式和矢量表示形式。

因为通过指数形式和矢量形式才能实现域上运算，而一个码符号取自 $GF(2^m)$ 的差错控制系统，故必须选择其中一种表示方法作为系统数据处理的基本形式，域上运算实现的另一表示形式则通过基本形式转换来得到。

为了提高域运算处理速度，一般采用查表的方法实现基本形式到域运算实现所需的另一形式之间的转换，即系统为了完成域上运算，形成了域元素矢量形式 $\leftrightarrow$ 指数形式转换对照表[2], 这就是二表法。

$GF(2^m)$ 上共有元素 2^m 个：加法恒元 0、乘法恒元 1 及 $\alpha^i(i=1,2,\cdots,2^m-2)$。由于加法恒元 0 无法用指数形式表示，该算法就是选用域元素的矢量形式 (二进

[2]可通过迭代法生成

制 m) 作为数据处理的基本形式。此时，根据域运算原理，两域元素相加就是两域元素对应位模 2 加；两非零元素相乘 (除) 则要利用域元素矢量形式 $\leftrightarrow$ 指数形式转换表，转换为指数形式进行，最后再将结果转换为矢量形式存放。同理，域上指数运算也需要查表转换进行。

2.4 矢量空间[3]

定义 2.4：设 $\mathbf{F}$ 是一个域，$\mathbf{V}$ 是一个非空集合。设在 $\mathbf{V}$ 中定义了一个二元加法运算 "$+$"，即对任意 $\mathbf{u},\mathbf{v}\in\mathbf{V}$，都有 $\mathbf{u}+\mathbf{v}\in\mathbf{V}$；同时还定义了一个 $\mathbf{F}$ 中的元素乘以 $\mathbf{V}$ 中元素的乘法运算 "$\cdot$"，即对任意 $a\in\mathbf{F}$ 和任意 $\mathbf{v}\in V$，都有 $a\cdot\mathbf{v}\in\mathbf{V}$。如果下述运算规则成立，则称 V 是域 F 上的一个向量空间。

1．$\mathbf{V}$ 关于加法运算 "$+$" 是一个交换群[3]

2．对任意 $a\in\mathbf{F}$ 和任意 $\mathbf{u},\mathbf{v}\in\mathbf{V}$，都有

$$a\cdot(\mathbf{u}+\mathbf{v})=a\cdot\mathbf{u}+a\cdot\mathbf{v} \tag{2-8}$$

3．对任意 $a,b\in\mathbf{F}$ 和任意 $\mathbf{v}\in\mathbf{V}$，都有

$$(a+b)\cdot\mathbf{v}=a\cdot\mathbf{v}+b\cdot\mathbf{v} \tag{2-9}$$

4．对任意 $a,b\in\mathbf{F}$ 和任意 $\mathbf{v}\in\mathbf{V}$，都有

$$a\cdot(b\cdot\mathbf{v})=(ab)\cdot\mathbf{v} \tag{2-10}$$

5．对任意 $\mathbf{v}\in\mathbf{V}$，都有

$$1\cdot\mathbf{v}=\mathbf{v} \tag{2-11}$$

其中，1 是域 $\mathbf{F}$ 的乘法单元。

通常，我们将 $\mathbf{V}$ 中的元素称为向量，并将 $\mathbf{V}$ 的加法群的零元素称为零向量，记为 $\mathbf{0}$。域 $\mathbf{F}$ 中的元素称为纯量。

[3]这个群通常称为 $\mathbf{V}$ 的加法群

下面介绍非常有用的，在编码理论中起着中心作用的 $GF(2)$ 上的矢量空间。研究 n 个分量的有序序列。

$$(a_0, a_1, \cdots a_{n-1})$$

其中每个分量 a_i 是二元域 $GF(2)$ 上中的元素[4]。这个序列一般称为 $GF(2)$ 上的 n 重。由于每个 a_i 有两种选择，我们可以构造 2^n 个不同的 n 重。令 v_n 表示 2^n 个不同 n 重集合。现在，我们在 v_n 上定义一个加法：对 v_n 中任何 $\mathbf{u} = (u_0, u_1, \cdots u_{n-1})$ 和 $\mathbf{v} = (v_0, v_1, \cdots v_{n-1})$，

$$\mathbf{u} + \mathbf{v} = (u_0 + v_0, u_1 + v_1, \cdots u_{n-1} + v_{n-1}) \tag{2-12}$$

其中 $u_i + v_i$ 按模 2 进行相加。显然 $\mathbf{u}+\mathbf{v}$ 也是 $GF(2)$ 上的 n 重。因此 v_n 在2-12式所定义的加法下是封闭的。容易证明，v_n 在2-12式定义的加法下是可交换群。

下面定义以 $GF(2)$ 中元素 a 数称 v_n 中一个 n 重 $\mathbf{v}$ 为：

$$a \cdot (v_0, v_1, \cdots v_{n-1}) = (a \cdot v_0, a \cdots v_1, \cdots a \cdot v_{n-1}) \tag{2-13}$$

其中 $a \cdot v_i$ 按模 2 乘法进行。显然，$a \cdot (v_0, v_1, \cdot v_{n-1})$ 也是 v_n 中的 n 重。若 $a = 1$，则：

$$1 \cdot (v_0, v_1, \cdots v_{n-1}) = (1 \cdot v_0, 1 \cdot v_1, \cdots 1 \cdot vn - 1) = (v_0, v_1, \cdots v_{n-1}) \tag{2-14}$$

容易证明，式2-12和2-13定义的矢量加法和数乘分别满足分配律和结合律。所以，$GF(2)$ 上所有 n 重的集合 v_n 形成 $GF(2)$ 上的一个矢量空间。

V 是域 **F** 上的一个矢量空间，可能会碰到 **V** 的一个子集 **S** 也是 **F** 上的一个矢量空间，这种子集称作是 **V** 的子空间。

定理 2.7：令 **S** 是域 **F** 上矢量空间 **V** 的一个非空子集，则当 **S** 满足下述条件时它是 **V** 的一个子空间：

1．对 **S** 中的任意两个矢量 **u** 和 **v**，$\mathbf{u}+\mathbf{v}$ 也是 **S** 中的矢量。

2．对 **F** 中的任意元素 a 和 **S** 中的任意矢量 **v**，$a \cdot \mathbf{u}$ 也在 **S** 中。

[4]两元素即 $a_i = 0$ 或 $a_i = 1$

定理 2.8:令 $v_1, v_2, \cdots v_k$ 是域 **F** 上矢量空间 **V** 的 k 个矢量,则 $v_1, v_2, \cdots v_k$ 的所有线性组合构成 **V** 的一个子空间。

我们称矢量集合张成一矢量空间 **V**,若 **V** 中每个矢量都是该集合中矢量的线性组合,在任何矢量空间或子空间中,都至少存在一个线性独立的矢量的集合 **B**,它张成该空间。这个集合称作矢量空间的基底[5](或基). 矢量空间基底中矢量的数目称作矢量空间的维数。

研究 $GF(2)$ 上所有 n 重构成的矢量空间 $\mathbf{V_n}$,我们作下述 n个n 重:

$$\left.\begin{array}{ll} e_0 & = (1,0,0,0,\cdots 0) \\ e_1 & = (0,1,0,0,\cdots 0) \\ & \vdots \\ e_{n-1} & = (0,0,0,0,\cdots 1) \end{array}\right\} \text{"1" 每次右移一位} \tag{2-15}$$

其中 n 重 e_i 仅在第 i 位上有一个非零分量,则 $\mathbf{V_n}$ 中每个 n 重 $(a_0, a_1, \cdots a_{n-1})$ 可以表示成 $(e_0, e_1, \cdots e_{n-1})$ 的线性组合:

$$(a_0, a_1, a_2, \cdots a_{n-1}) = a_0 e_0 + a_1 e_1 + a_2 e_2 + \cdots + a_{n_1} e_{n-1} \tag{2-16}$$

所以 $e_0, e_1, \cdots e_{n-1}$ 张成 $GF(2)$ 所有 n 重的矢量空间。从上述方程可以看出 $e_0, e_1, \cdots e_{n-1}$ 是线性独立的。因此,它们构成 v_n 的基底,且 v_n 的维数是 n,若 $k < n$ 且 $v_1, v_2, \cdots v_k$ 是 v_n 中的 k 个线性独立矢量,则 $v_1, v_2, \cdots v_k$ 的所有型为

$$\mathbf{u} = c_1 v_1 + c_2 v_2 + \cdots + c_k v_k \tag{2-17}$$

的线性组合构成 v_n 的一个 k 为子空间 **S**。由于每个 c_i 有两个可能的取值 0 或 1,故有 2^k 个可能不相同的 $v_1, v_2, \cdots v_k$ 的线性组合。因此,**S** 由 2^k 个矢量组成且为 v_n 的一个 k 为子空间。

令 $\mathbf{u} = (u_0, u_1, \cdots u_{n-1})$ 和 $\mathbf{v} = (v_0, v_1, \cdots v_{n-1})$ 是 v_n 中的两个 n 重。这里定义 **u**和**v** 的内积 (或点积) 为

$$\mathbf{u} \cdot \mathbf{v} = u_0 \cdot v_0 + u_1 \cdot v_1 + \cdots + u_{n-1} \cdot v_{n-1} \tag{2-18}$$

其中,$u_i \cdot v_i$和$u_i \cdot v_i + u_{i+1} \cdot v_{i+1}$ 按模 2 乘法和加法进行。因而内积 $\mathbf{u} \cdot \mathbf{v}$ 是 $GF(2)$ 中的标量。若 $\mathbf{u} \cdot \mathbf{v} = 0$,就称 **u**和**v** 彼此正交。

[5]注意:任意两个基底中矢量的数目相同

内积[6]有如下性质：

1. $\mathbf{u}\cdot\mathbf{v}=\mathbf{v}\cdot\mathbf{u}$
2. $\mathbf{u}\cdot(\mathbf{v}+\mathbf{w})=\mathbf{u}\cdot\mathbf{v}+\mathbf{u}\cdot\mathbf{w}$
3. $(\mathbf{au})\cdot\mathbf{v}=\mathbf{a}(\mathbf{u}\cdot\mathbf{v})$

令 **S** 是 v_n 的 k 维子空间，并令 $\mathbf{S_d}$ 是 v_n 中这样的矢量集合：对 **S** 中的任意 **u** 和 $\mathbf{S_d}$ 中的任意 **v** 有 $\mathbf{u}\cdot\mathbf{v}=0$。集合 $\mathbf{S_d}$ 至少包含全零 n 重，$0=(0,0,\cdots 0)$，因为对 **S** 中的任一 **u** 有 $0\cdot\mathbf{u}=0$。因此 $\mathbf{S_d}$ 是非空的。对 $GF(2)$ 中任一元素 a 和 $\mathbf{S_d}$ 中任一 **v**，

$$a\cdot\mathbf{v}=\begin{cases}0\text{ 若 }a=0\\ \mathbf{v}\text{ 若 }a=1\end{cases}\tag{2-19}$$

所以 $a\cdot\mathbf{v}$ 也是 $\mathbf{S_d}$ 中。令 **v**和**w** 是 $\mathbf{S_d}$ 中任意两个矢量。对 **S** 中任意 $\mathbf{u},\mathbf{u}\cdot(\mathbf{v}+\mathbf{w})=\mathbf{u}\cdot\mathbf{v}+\mathbf{u}\cdot\mathbf{w}=0+0=0$。这就是说，若 **v**和**w** 正交于 **u**，则矢量 $\mathbf{v}+\mathbf{w}$ 也与 **u** 正交。因而，$\mathbf{v}+\mathbf{w}$ 也是 $\mathbf{S_d}$ 中一个矢量。因此 $\mathbf{S_d}$ 也是 v_n 的一个子空间。这个子空间 $\mathbf{S_d}$ 称作是 **S** 的零化 (或对偶) 空间。反之，**S** 也是 $\mathbf{S_d}$ 的零化空间。$\mathbf{S_d}$ 的维数由定理2.9给定。

定理 2.9：令 **S** 是 $GF(2)$ 上所有 n 重的矢量空间中的一个 k 维子空间，则其零化空间 $\mathbf{S_d}$ 的维数为 $n-k$。换句话说，$dim(s)+dim(s_d)=n$

矢量空间是伽罗华域中矢量表示法的基础，也是后来将介绍的 RS 译码当中十分重要的概念，在对伽罗华域当中加法的运算起着至关重要的作用。

2.5 矩阵[4]

$GF(2)$ 上 (或任意其他域上)$k\times n$ 阶矩阵是一个有 k 行和 n 列的长方阵：

$$\mathbf{G}=\begin{bmatrix}g_{00} & g_{01} & g_{02} & \cdots & g_{0n}\\ g_{10} & g_{11} & g_{12} & \cdots & g_{1n}\\ \vdots & & & \vdots & \\ g_{(k-1)0} & g_{(k-1)1} & g_{(k-1)2} & \cdots & g_{(k-1)n}\end{bmatrix}\tag{2-20}$$

其中每个元素 $g_{ij},0\le i<k$和$0\le j<n$，都是取自二元域 $GF(2)$ 中的元素。第一个下标 i 表示行，第二个下标 j 表示列。有时以符合 $[g_{ij}]$ 来简化表示矩阵2-20式。

[6]内积的概念可以推广到任何伽罗华域

从2-20式中可看出，**G** 的每一行都是 $GF(2)$ 的一个 n 重，每一列都是 $GF(2)$ 上的一个 k 重。矩阵 **G** 也可用它的 k 行 $g_0, g_1, \cdots g_{k-1}$ 表示如下：

$$\mathbf{G} = \begin{bmatrix} g_0 \\ g_1 \\ \vdots \\ g_{k-1} \end{bmatrix} \tag{2-21}$$

若 **G** 的 k 行 $(k \leq n)$ 是线性独立的，则这些行的 2^k 个线性组合形成 $GF(2)$ 上所有 n 重的矢量空间 v_n 的 k 维子空间。这个子空间称为 **G** 的行空间，我们可以交换 **G** 中任意两行或将一行加到另一行。这些称作是初等行运算。对 **G** 进行初等行运算我们得到 $GF(2)$ 上的另一个矩阵 **G**′。但是，**G**和**G**′ 都给出同一行空间。

令 **S** 是 $GF(2)$ 上一个 $k \times n$ 阶矩阵 **G** 的行空间，其 k 行 $g_0, g_1, \cdots g_{n-1}$ 线性独立。令 $\mathbf{S_d}$ 是 **S** 的零化空间，则 $\mathbf{S_d}$ 的维数是 $n-k$。令 $h_0, h_1, \cdots h_{n-k-1}$ 是 S_d 中 $n-k$ 个线性独立矢量。显然，这些矢量张成 $\mathbf{S_d}$。我们可以用 $h_0, h_1, \cdots h_{n-k-1}$ 作为行构成一个 $(n-k) \times n$ 阶矩阵 **H**，

$$\mathbf{H} = \begin{bmatrix} h_0 \\ h_1 \\ \vdots \\ h_{n-k-1} \end{bmatrix} = \begin{bmatrix} h_{00} & h_{01} & \cdots & h_{0,n-1} \\ h_{10} & h_{11} & \cdots & h_{1,n-1} \\ \vdots & \vdots & & \vdots \\ h_{n-k-1,0} & h_{n-k-1,1} & \cdots & h_{n-k-1,n-1} \end{bmatrix} \tag{2-22}$$

H 的行空间是 $\mathbf{S_d}$。由于 **G** 的每一行 g_i 是 **S** 中的矢量，且 **H** 的每一行 h_i 是 $\mathbf{S_d}$ 中的矢量，则 g_i和h_i 的内积必是零 (即$g_i \cdot h_i = 0$)。由于 **G** 的行空间 **S** 是 **H** 的行空间 S_d 的零化空间，故我们称 **S** 是 **H** 的零化空间。综上结果可得到：

定理 2.10：对 $GF(2)$ 上有 k 个线性独立的任意 $k \times n$ 阶矩阵，都存在一个有 $(n-k)$ 个线性独立行的 $(n-k) \times n$ 阶矩阵，使得对 **G** 中任一行 g_i 和 H 中任意行 h_j 有 $g_i \cdot h_j = 0$。**G** 的行空间是 **H** 的零化空间，反之亦然。

若两个矩阵有相同的行数和相同的列数，它们就可以相加。两个 $k \times n$ 阶矩阵 $\mathbf{A} = [a_{ij}]$ 和 $\mathbf{B} = [b_{ij}]$ 相加，我们简单地将它们的相应元素 a_{ij}和b_{ij} 相加如下：

$$[a_{ij}] + [b_{ij}] = [a_{ij} + b_{ij}] \tag{2-23}$$

因此，得到的矩阵也是 $k \times n$ 阶矩阵。两个矩阵只要第一个矩阵的列数等于第二个矩阵的行数就可以相乘。$k \times n$ 阶矩阵 $\mathbf{A} = [a_{ij}]$ 乘以 $n \times l$ 阶矩阵 $\mathbf{B} = [b_{ij}]$ 的

积

$$\mathbf{C} = \mathbf{A} \times \mathbf{B} = [c_{ij}] \tag{2-24}$$

是 $k \times l$ 阶矩阵，其元素 c_{ij} 等于 **A** 中第 i 行 a_i 和 **B** 中第 j 列 b_i 的内积，即

$$c_{ij} = a_i \cdot b_j = \sum_{i=0}^{n-1} a_{ij} \cdot b_{ij} \tag{2-25}$$

令 **G** 是 $GF(2)$ 上一个 $k \times n$ 阶矩阵，用 $\mathbf{G^T}$ 表示 **G** 的转置，它是一个 $n \times k$ 阶矩阵。它的各行是 **G** 的各列，而它的各列是 **G** 的各行。一个 $k \times k$ 阶矩阵，若在主对角线都为 1 而其它处为 0。就称作是恒等矩阵。这一矩阵通常用 $1k$ 表示。矩阵 **G** 的子阵是由除去 **G** 的给定行或列所得到的矩阵。

矩阵的介绍是为了分析 RS 译码的理论知识的基础，它可以直观，深入的解释 BM 算法以及 RS 译码的全过程，是我们作理论分析的必备知识。

2.6 本章小节

本章从近世代数的角度，介绍了信道编码的基本知识，其中包括群、域、矢量空间、矩阵，着重介绍了伽罗华域及其上的运算准则，为下面 RS 码的编译码提供理论依据。

第 三 章　卷积码的编码与序贯译码算法

3.1　卷积码与水声通信

现实中已经开始运用卷积码在水声通信中，而且可以很好的解决误码率问题，但是基于以下分析我们还是要继续研究卷积码的编译码。

由于先前的卷积码采用的是 $(4,1,7)$，译码方式自然采用的是维特比译码算法，误码率曲线没什么说的，译码算法效率也是非常出色，但是有个致命的问题，就是码率 $R=\frac{1}{4}$，这对于原本带宽就非常狭窄的水声通信系统来说无疑是雪上加霜，因此我们开始关注于卷积码的另外一种译码算法就是序贯译码算法。

序贯译码和维特比译码比较图[8] 可知，在约束长度 $K=41$，码率 $R=\frac{1}{2}$ 的情况下的序贯译码的误码率要比约束长度 $K=7$，码率 $R=\frac{1}{3}$ 的维特比译码，软硬判决都好，因此选择序贯译码，可以提高码率。

至于约束长度问题，由于序贯译码的运算量和运行时间与约束长度无关，因此可以选择很大的约束长度，而维特比译码却不可以，这在后面章节会详细介绍。

3.2　卷积码概述

卷积码最早是 1955 年由伊利亚斯 (P.Elias) 提出来的。它是一种非分组码，卷积码更适合用于前向纠错法。

在一个二进制分组码 (n,k) 当中，包含 k 个信息位，码组长度为 n，每个码组的 $(n-k)$ 个校验位仅与本码组的 k 个信息位有关，而与其它码组无关。为了达到一定的纠错能力和编码效率 $(R_e=\frac{k}{n})$，分组码的码组长度 n 通常都比较大。编译码时必须把整个信息码组存储起来，由此产生的延时随着 n 的增加而线性增加。

与分组码不同，卷积码中编码后的 n 个码元不仅与当前段的 k 个信息有关，而且也与前码 $(N-1)$ 段的信息有关，编码过程中相互关联的码元为 nN 个。因此，这 N 时间内的码元数目 nN 通常被称为这个码的约束长度。卷积码的纠错能力随着 N 的增加而增大，在编码器复杂程度相同的情况下，卷积码的性能优于分组码。另一点不同的是：分组码有严格的代数结构，但卷积码至今尚未找到如此

严密的数学手段，把纠错性能与码的结构十分规律地联系起来，目前大都采用计算机来搜索好码。

3.3 卷积码的编码[5]

卷积码的编码器是由一个有 k 个输入位、n 个输出位，且具有 m 级移位寄存器所构成的有限状态的有记忆系统，通常称为时序网络。卷积码编码原理图如图3-1 表3-1给出了卷积码一些参数及其说明。

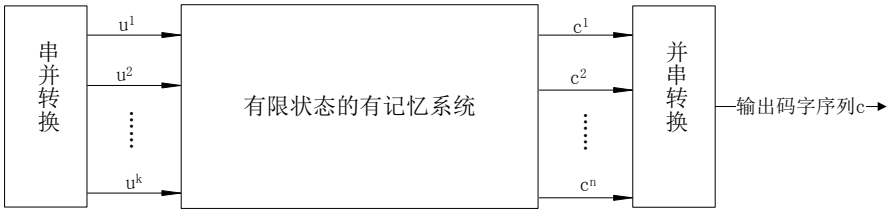

图 3-1 卷积码编码器原理图

表 3-1 卷积码编码器参数

卷积编码器的记忆参数[3]

- k 和 n 分别表示编码器输入和输出位。编码效率为 $R = \frac{k}{n}$
- 当前 n 位的输出是当前 k 位和前 $k \times m$ 位输入的线性组合，这里的 m 称为卷积码的记忆长度。
- 二进制卷积码通常用 3 个参数写成 (n, k, m)。
- 编码器的记忆长度 $m = \max_l v_l$。
- 最小约束长度 $v_{min} = \min_l v_l$。
- 总约束长度 $v = \sum_{l=1}^{k} v_l$。

描述这类时序网络的方法很多，大致分为两大类型：解析表示法与图形表示法。在解析法中又可分为离散卷积法、生成矩阵法、码多项式法等；在图形表示法中也可分为状态图法、树图法、网格图法等。由于这个设计着重于实现卷积码的编码与序贯译码，所以只介绍码多项式法、状态图法以及网格图法。下面，引用具体实例对这三种表示方法加以说明。图3-2给出一个二元 (2,1,3) 卷积码的编码器结构。由图可见，它是由 $k=1$ 即一个输入位，$n=2$ 即两个输出位，$K=3, m=2$

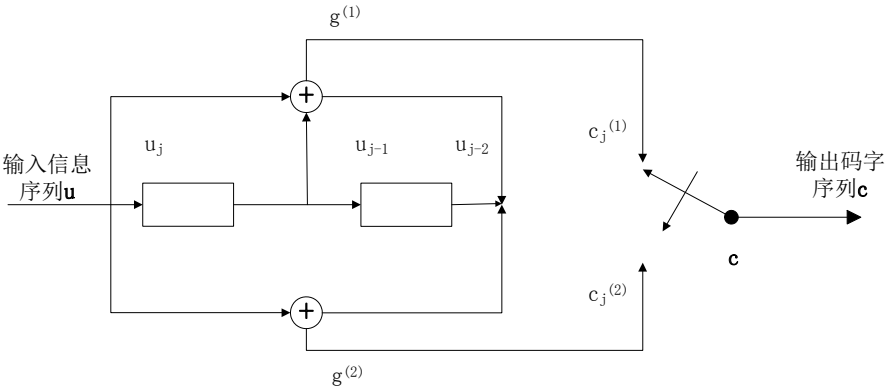

图 3-2 编码效率为 1/2，约束长度 $K=3$ 的 (2,1,3) 卷积码编码器

即两级移位寄存器所组成的有限状态的有记忆系统。

3.3.1 多项式法

由上图可以看出，该卷积码的生成多项式为

$$\begin{aligned}\mathbf{g}^{(1)} &= (111) = 1 + x + x^2 \\ \mathbf{g}^{(2)} &= (101) = 1 + x^2\end{aligned} \tag{3-1}$$

输入信息序列也可以表达为多项式 (在这里，最左边的比特对应于多项式的最低次项)

$$\mathbf{u} = (10111) = 1 + x^2 + x^3 + x^4 \tag{3-2}$$

则卷积码可以用下列码多项式表达

$$\begin{aligned}
\mathbf{c}^{(1)}&=(1+x^2+x^3+x^4)(1+x+x^2)\\
&=1+x^2+x^3+x^4+x+x^3+x^4+x^6+x^2+x^4+x^5+x^6\\
&=1+x+x^4+x^6\\
&=(1100101)\\
\mathbf{c}^{(2)}&=(1+x^2+x^3+x^4)(1+x^2)\\
&=1+x^2+x^3+x^4+x^2+x^4+x^5+x^6\\
&=1+x^3+x^5+x^6\\
&=(1001011)
\end{aligned} \tag{3-3}$$

所以最终生成的卷积码为 $\mathbf{c}=(11100001100100)$。

3.3.2 状态图法

卷积编码器的状态图描述了编码器的运行。在讨论卷积码的距离特性和译码算法时，状态图将非常有用。

由于卷积码编码器在下一时刻的输出取决于编码器当前的状态及下一时刻的输入。而编码器当前状态取决于编码器在当前各移位寄存器所存储的内容，称编码器的各移位寄存器在任一时刻的存数 (0 或 1) 为编码器在该时刻的一个状态[1]。随着信息序列的不断输入，编码器就不断从一个状态转移到另一个状态，并输出相应的码序列。编码器的总可能状态数是 2^{mk} 个。

取图3-2所示的卷积码编码器，其状态图如3-3所示。图3-3中，4 个圆圈中的数

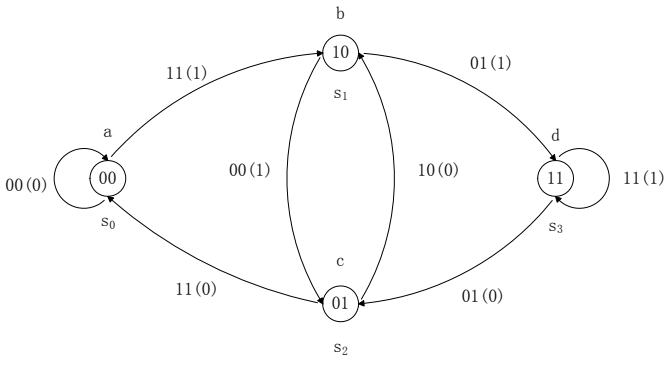

图 3-3 (2,1,3) 卷积码状态图

字表示状态，状态之间的连线与箭头表示转移方向，称作分支，分支上的数字表示

[1]此状态表示记忆着以前的输入信息

由一个状态到另一个状态转移时的输出码字，而括号中数字表示相应的输入信息数字。

3.3.3 网格图

网格图的概念是由 Forney 提出的。图3-4给出了图3-2编码器的网格图。该图的纵坐标表示所有状态，横坐标表示时间，节点表示编码器的状态。这类网格图描述法在卷积码的维特比译码和序贯译码中都有很重要的作用，它综合了状态图和树图的优点，即网格图既有状态图结构简单，又具有树图的时序关系清晰特点。图3-4中实线表示输入为 0 时所走的分支，虚线表示输入为 1 时所走的分支。

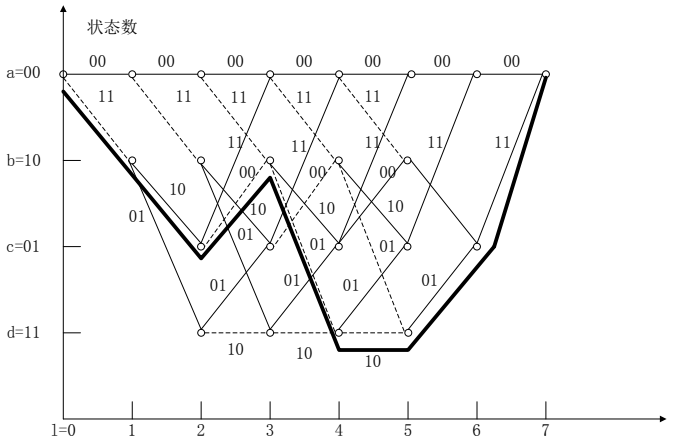

图 3-4 (2,1,3) 卷积码网格图表示法

从网格图中可以看出卷积码编码需要考虑的一个问题，就是码的终结。

理论上卷积码的码序列是无限长的。但是在实际应用中通常都是有限长的码序列。表3-2列举了 3 种获得有限长码序列的方法。结合表3-2与实际编译码时候的需要，本设计选择的是终结与咬尾方式的结合，也就是，通过增加一些尾序列使得初始状态和结束状态相同，从而降低最后若干比特的差错概率。

3.4 卷积码的译码

卷积码的译码基本上可划分为两大类：代数译码和概率译码。在分组码中一般用代数译码，在 RS 译码方式中我们将介绍，而这里则侧重介绍概率译码，而且概率译码也是实际中最常采用的卷积码译码方法。

表 3-2 码的终结方法

码的终结方法[3]

截断 (Truncation)：在一定数量的比特之后停止编码，不采取任何措施。这会在序列的最后若干比特上导致交大差错概率。

终结 (Termination)：在码序列后面附加一些尾序列，以保证编码器进入预定的结束状态。这样，序列最后若干比特的差错概率较低。

咬尾 (Tail-biting)：选用一种起始状态，能保证起始和结束首尾状态相同。这样，序列最后若干比特的差错概率与前面一样。

1967 年，维特比 (Viterbi) 引入了一种卷积码的译码算法，这就是著名的维特比算法。后来 Omura 证明维特比译码等价于求通过一个加权图的最短路径问题的动态规划解。最后 Forney 指出它事实上就是卷积码的最大似然译码算法，即译码器所选择的输出总是能给出对数似然函数值为最大的码字。

早在维特比译码算法提出之前，已有其它译码卷积码算法，最早的是由 Wozencraft 提出，后经 Fano 修改的序贯译码算法 (sequential decoding algorithm[9])。序贯译码的工作原理是，对发送码字序列进行假定，计算这些假定序列和接收信号之间的距离度量，只要度量值表明该选择是可能的，就继续译码；否则就向后返回，并改变假设序列，直到通过系统的试验 -纠正搜索后找到可能的假设序列。

维特比算法的主要缺陷是，尽管差错概率随着约束长度按指数级减少，但是编码状态数以及相应的译码器复杂性，都随着约束长度呈指数级增长。另一方面，维特比算法的计算复杂性与信道特性无关 (和硬判决相比，软判决需要的计算量仅略微增加)。序贯译码渐地达到了和最大似然译码一样的差错概率，但不需要搜索所有的可能状态。实际上，顺序译码所要搜索的状态数，本质上和约束长度无关，因此，可以使用较大 $(K = 41)$ 的约束长度。这是提供低差错率的重要因素。序贯译码的主要缺陷是，它搜索的状态量度的数量是一个随机变量。对于序

贯译码，不良假设和回溯搜索的个数是信道 SNR[2]的函数。当 SNR 较低时的假设次数要比 SNR 较高时多。由于存在这种计算负荷的变化，必须使用缓冲器来存储到达的序列。

由于是水声通信，因此对于运算量的要求比较严，所有维特比译码算法显然就不适合要求，而序贯译码采用约束长度较大时，译码效率不次于维特比译码算法，甚至比它还要好，误码率还要低，而且不需要很多的运算量，至于缓冲区，对于一个水声通信来说不成问题。

综合上面的考虑，选择序贯译码[10] 来作为卷积码译码方式。

3.5 费诺 (Fano) 算法

现有的序贯译码算法有两大类：费诺算法和堆栈算法[11]。

堆栈算法的最大优点是保存了所有搜索路径从而使搜索效率最高，代价是维持一个过于庞大的堆栈，而费诺算法则正相反，省去了存储器，但是会增加很多计算量。

结合考虑两种方式、使用的卷积码特点以及实际应用情况，选择的是费诺译码算法。

3.5.1 费诺算法原理

费诺算法介绍[12] 前面已经说过费诺度量最大的特点就是没有对搜索过的路径进行存储，省去了庞大的堆栈。搜索的前进和回退是有门限值 T 来控制，算法只沿一条路径搜索，只存储这条路径的度量值，只要被搜索的路径度量值增加，译码器就一直沿这条路径搜索下去，直到度量值明显下降时为止，译码器回退搜索由前面节点分离出去的分支路径，这种回退控制是由改变门限值 T 来实现的，当向前搜索时，如果获得足够大的度量值增量就收紧 (提高) 门限值。当回退搜索时，则放松 (降低) 门限值，这样保持没有一个节点以两次同样的门限值被搜索到[13]。

费诺度量[14] 1963 年，通过大量的仿真模拟，费诺度量被发现，并随后被费诺用在序贯译码中[15]，当做典型的路径度量值。

[2]Sound Noice Ratio 信噪比

对于在译码树 l 层中的任意路径 $v_{(ln-1)}$，费诺度量定义如下：

$$M(\mathbf{v}_{(ln-1)}|\mathbf{r}_{(ln-1)}) = \sum_{j=0}^{ln-1} M(v_j|r_j). \tag{3-4}$$

其中 $\mathbf{r} = (r_0, r_1, \cdots, r_{N-1})$ 是接收的信号矢量，并且

$$M(v_j|r_j) = \log_2 \left[\frac{Pr(r_j|v_j)}{Pr(r_j)} \right] - R \tag{3-5}$$

是每个比特位的度量值。

由于本设计只针对于硬判决译码器，因此在错误概率为 p 的 BSC[3]信道中，$Pr(r_j = 0|v_j = 1) = Pr(r_j = 1|v_j = 0) = p$当$0 \le j \le N-1$，当 $0 < p < 1/2$，$v_{(ln-1)}$ 的费诺度量为

$$M(\mathbf{v}_{(ln-1)}|\mathbf{r}_{(ln-1)}) = \sum_{j=0}^{ln-1} \log_2 Pr(r_j|v_j) + \ln(1-R) \tag{3-6}$$

其中 $R = \frac{k}{n}$ 为码率，

$$\log_2 Pr(r_j|v_j) = \begin{cases} \log_2(1-p), & \text{当} r_j = v_j; \\ \log_2(p), & \text{当} r_j \neq v_j; \end{cases} \tag{3-7}$$

因此，式3-5结合3-7并在错误概率为 p 的 BSC 信道中，可以推导出单个比特位的度量值为：

$$M(v_j|r_j) = \begin{cases} \log_2(1-p) + (1-R), & \text{当} r_j = v_j; \\ \log_2(p) + (1-R), & \text{当} r_j \neq v_j; \end{cases} \tag{3-8}$$

而路径的费诺度量则是从根节点到当前节点比特位度量的和，因此直接套用式3-4即可。而表3-3是关于费诺度量的一些说明。

费诺准则[9] 上面介绍了费诺度量，这就涉及到一个问题，什么时候前进，什么时候后退，什么时候选择分支，这需要一个准则，如下表3-4: 表3-4中，F 表示前进，L 表示分支，B 表示后退，T 是判决门限，L_i 表示从根节点到第 i 节点的费诺度量，Δ 是步进长度。

其中 Δ 大小的选择很关键，如果 Δ 太小，那么在搜索最优路径时，计算路径度量的负担会很重；如果 Δ 太大，在一次调整中，动态判决门限值 T 变化太大，将会增加运算量和错误概率。因此，实验仿真表明，在费诺度量没有整数化之前 Δ 取值在 2 和 8 之间，如果整数化之后，相应的 Δ 也要乘以对应的比例。

[3]Binary Symmetric Channel 二进制对称信道

表 3-3 费诺度量的说明

费诺度量的说明[3]

- Massey 证明在任何译码阶段，用在堆栈中最大费诺度量来扩展译码路径能够最小化扩展路径不属于最优译码路径的可能，而且基于上面的分析，序贯译码采用费诺度量作为路径度量
- 费诺度量一个显著特性就是与码率有关，引入码率这个参数可以降低序贯译码算法的复杂度。
- 当码率增大时，错误路径的数量也会增加，因此，当前被检测路径为正确路径的可能性就越小。所以，在高码率情况下，长路径是最优路径的一部分的可能性也会减弱。
- 从费诺度量的计算公式上来看，其不是一个整数，由于全路径的费诺度量都加上一个常量，并不改变译码结果，为了计算方便，我们把度量值整数化。

3.5.2 费诺算法实现

图3-5所示为费诺译码算法整个流程：结合图3-5，总结费诺算法的详细步骤：

1. 初始化当前节点 v_c 为根节点，当前度量 M_c 为 0，前一节点 v_p 为空，前一度量 M_p 为 $-\infty$。
2. 在当前节点 v_c 的所有分支中，选路径度量最大的 v_s，是的 M_s 为其路径度量。
3. 判断 $M_s \geq T$ 是否成立，如果否，转到第 7 步，如果是继续。
4. 路径前进一个节点，前一节点 v_p 用当前节点 v_c 代替，前一度量 M_p 由当前度量 M_c 代替；相应的，当前节点 v_c 为 v_s，当前度量值 M_c 为 M_s。

表 3-4 费诺准则

Conditions			Action	
Rule	Previous	Comparisons[c]	Final threshold	Move
1	F or L	$L_{k-1} < T + \Delta, L_k \geq T$	Raise(if possible)	F
2	F or L	$L_{k-1} \geq T + \Delta, L_k \geq T$	No change	F
3	F or L	any$L_{k-1}, L_k < T$	No change	B
4	B	$L_{k-1} < T$, any L_k	Lower	F
5	B	$L_{k-1} \geq T$, any L_k	No change	L or B

[c] By convention set $L_0 = 0$ and $L_{-1} = -\infty$

5. 判断 V_c 是否是最终码，如果是，那么结束搜索并输出 V_c，如果不是，那么判断 $M_p < T + \Delta$ 是否成立，如果否，跳到第 2 步，如果是，那么继续。

6. 调整判决门限，是的 $T \leq M_c < T + \Delta$，然后继续第二步。

7. 判断 $M_p \geq T$ 是否成立，如果不成立，那么 $T = T - \Delta$，如果成立，那么继续。

8. 后退，并更新 v_c, M_c, v_p, M_p。此过程与前进的时候相反。

9. 如果后退成功，那么继续步骤 2，否则跳到步骤 7。

3.5.3 费诺算法性能仿真

通过 matlab 编程，仿真实现 $K = 24, R = 1/2$ 的卷积码编译码，并画出误码率曲线，并对比在没有信道编码情况下的误码率。

仿真环境为 AWGN 信道，并采用 BPSK 调制方式。图3-6即为误码率曲线图：图中画出误码率曲线图并对比在没有信道编码情况下的误码率，从图中可以出，卷积码编译码的误码率要比没有信道编码的好很多。图中的曲线不很平滑，可以采用差值拟合来平滑曲线。

3.6 本章小结

本章介绍了卷积码的编码与译码算法，其中着重介绍并分析了序贯译码方式，尤其是其中的费诺算法，介绍了基本原理，分析了费诺度量的计算公式及其推导过程，判决门限的作用以及步进长度 Δ 的选取规则，用流程图详细介绍了编程

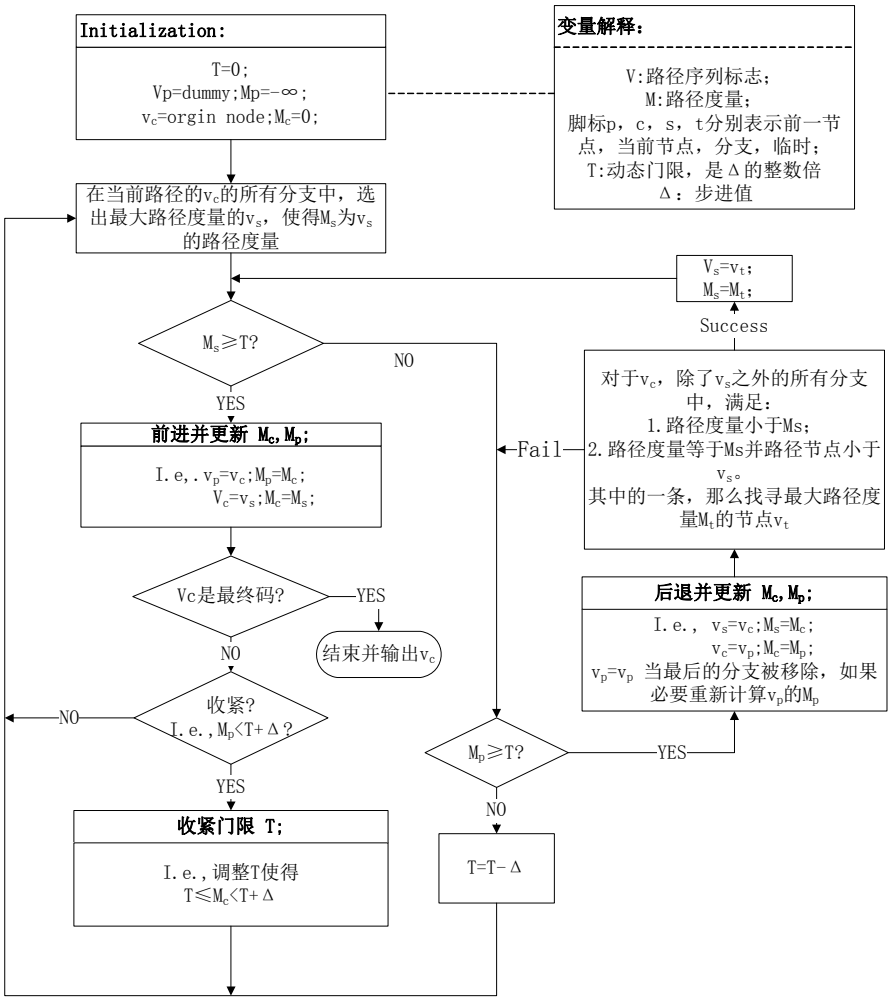

图 3-5 费诺算法流程图

实现费诺算法的步骤。并用 matlab 仿真，测试误码率曲线，以及参数选取是否合理。

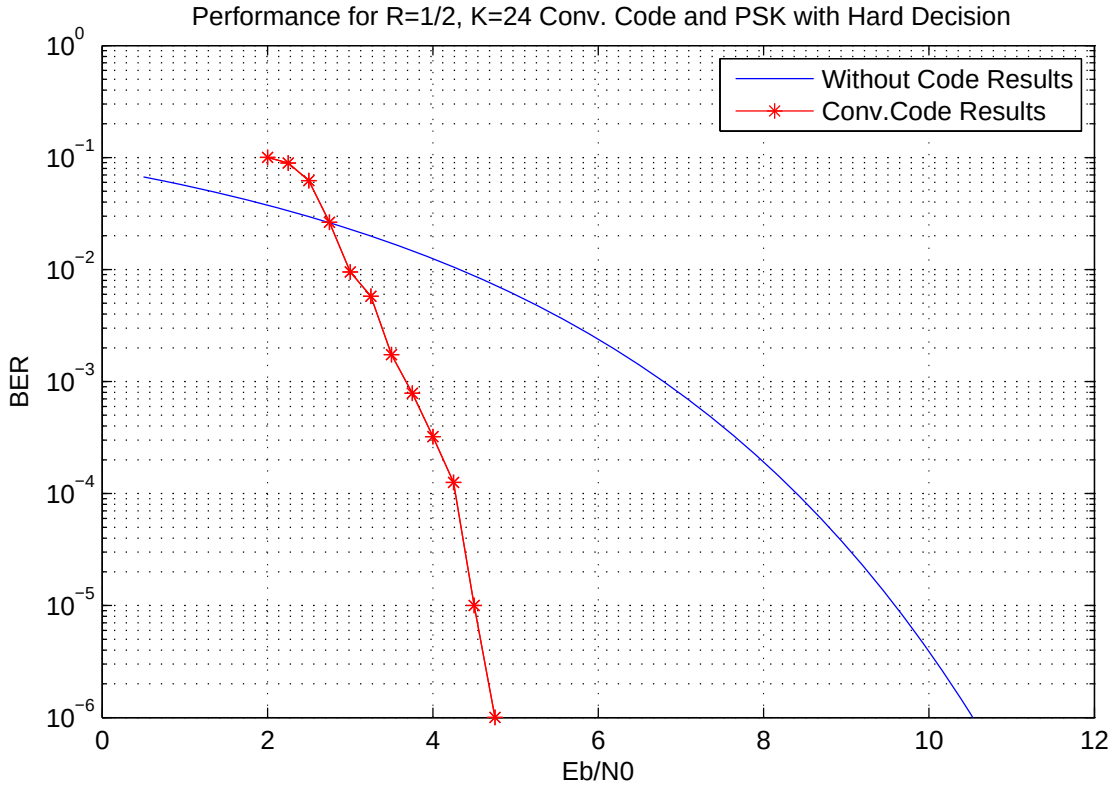
Performance for R=1/2, K=24 Conv. Code and PSK with Hard Decision
Without Code Results
Conv.Code Results
BER
Eb/N0
10^0
10^-1
10^-2
10^-3
10^-4
10^-5
10^-6
0
2
4
6
8
10
12

图 3-6　误码率曲线图

第 四 章　RS 码的编码与 BM 硬判决算法

4.1　RS 码与水声通信

由于前面已经介绍了卷积码，已经在水声通信中使用卷积码，而且理论和实际当中，同等复杂度运算量的卷积码要比 RS 码的误码率性能要好，为什么这里还要研究 RS 码，并且作为一种信道编码方式应用到水声通信中？

在第三章中介绍了卷积码编码中几种码的终结方式，而且本设计也根据通信要求和实际情况选择了终结和咬尾两种码终结方式的结合，这就是意味着，编码的总长度比实际数据码字长度要多一个约束长度 K，对于数据量很大的情况下，约束长度 K 就显得微不足道了，可以忽略不计，并不会导致多少带宽的浪费和码率的下降，而对于命令等几个字节甚至是几个比特数据量的时候，这就显得不合适了，对于序贯译码方式，通常选择约束长度 K 都很大，对于本设计仿真我们采用的是 $K = 24$，就是 3 个字节，可能远比要传输的数据量大，因此造成了很大的带宽浪费。因此采用 RS 码。

4.2　RS 码概述

RS[1]码是由 MIT 林肯实验室的 Irving S. Reed 和 Gustave Solomon[16] 于 1960 年提出的一种基于多项式的纠错编码，Gorenstein 和 Zierler 于 1961 年证明了 RS 码与多进制 BCH 码的关系。Berlekmamp 于 1967 年提出了直到至今还被广泛应用的硬判决译码算法，两年后，Massey 指出了此算法和线性反馈移位寄存器的关系，因此，该硬判决译码算法被称为 BM 算法，一直沿用至今。

RS 码具有很好的纠错特性，它是目前唯一实用的最小距离可分[2]码，能够有效地纠正随即符号错误，突发错误和删除错误。因此，在诞生初期，便被迅速应用。

[1] Reed-Solomon

[2] MDS

4.3 RS 码的编码

RS 码是一种常见的线性分组码，同时它也是循环码。所谓循环码指的是任意码字循环移位后仍然在码字空间内的一类线性分组码[4]。

一个 (n,k) 循环码可以用 $n-k$ 次的生成多项式 $g(x)$ 定义，其码多项式 $(n-1$次$)c(x)$ 是 $g(x)$ 的倍式：$c(x)=f(x)g(x), k-1$ 次多项式 $f(x)$ 称为消息多项式，其 k 个系数就是 k 个消息码元。由第二章的知识可知，如果 $g(x)$ 是 x^n-1 的因子，由 $g(x)$ 生成的码一定是循环码。

通过定义生成多项式，RS 码可以这样定义：在 $GF(q)$ 上，α 为本原域元素，$g(x)$ 定义为：

$$g(x)=(x-\alpha)(x-\alpha^2)\cdots(x-\alpha^{2t}) \tag{4-1}$$

其中 $2t=n-k, n=q-1$

由第二章的知识可知，对 $GF(q)$ 而言，所有的 $q-1$ 个非零元素就是 $x^{q-1}-1$ 的所有根。那么，显然 $g(x)$是x^n-1 的因子，因此，这样构造出来的码是循环码。

由4-1定义的 RS 码最小距离是 $2t+1$，因此通过硬判决能纠正 t 个错误。RS 码的一个特点就是能够通过调整4-1根的数量很方便地调整其纠错能力。

对于能够纠错 t 个错误的 $RS(n,k,d)$ 码，具有如下特征：

1. **码长：**$n=2^m-1$ 符号或 $m(2^m-1)$ 比特；
2. **信息码元数：**$k=n-2t$ 符号或 mk 比特；
3. **监督码元数：**$n-k=2t$ 符号或 $m(n-k)$ 比特；
4. **最小距离：**$d=2t+1=n-k+1$ 符号或 $m(n-k+1)$ 比特

令信息码元多项式为：

$$m(x)=m_0+m_1x+m_2x^2+\cdots+m_{k-1}x^{k-1} \tag{4-2}$$

4.3.1 基于乘法形式的 RS 编码器

公式：$c(x)=m(x)g(x)$

结构图如下：图4-1具体实现步骤如下：

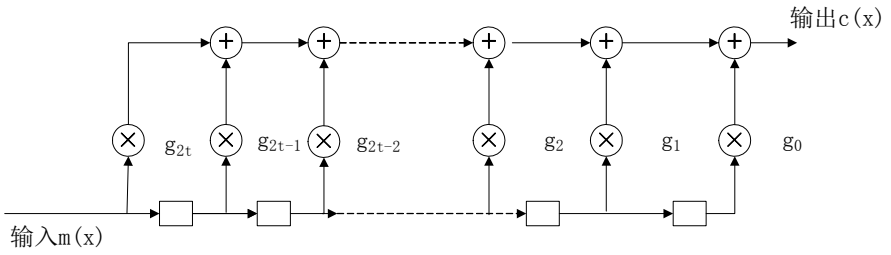

图 4-1 乘法编码器结构图

1. $2t$ 个寄存器全部清 0；

2. $m(x)$ 最高次系数 m_{k-1} 首先送入时，乘法器输出乘积的最高此项 x^{k+2t-1} 的系数 $m_{k-1}g_{2t}$，同时 m_{k-1} 存入寄存器的第一级；

3. $m(x)$ 的第二个系数 m_{k-2} 送入时，m_{k-1} 由第一级进入第二级寄存器，同时与 g_{2t-1} 相乘，$m_{k-2}g_{2t}+m_{k-1}g_{2t-1}$ 就是得到的乘积 x^{k+2t-2} 的系数；

4. 就这样重复进行，直到 $k+2t$ 次移位后，乘法器输出乘积的常数项 m_0g_0

但是结构图4-1编码出来的 RS 码是非系统码，这并不是我们需要的，因此，下面将介绍系统码的编码器结构。

4.3.2 基于除法形式的 RS 编码器

设校验码多项式：

$$h(x)=h_kx^k+h_{k-1}x^{k-1}+\cdots+h_1x+h+0 \tag{4-3}$$

系统码的多项式：

$$C(x)=c_{n-1}x^{n-1}+c_{n-2}x^{n-2}+\cdots+c_{n-k}x^{n-k}+\cdots+c_1x+c_0 \tag{4-4}$$

它的前 k 位系数：$c_{n-1},c_{n-2},\cdots,c_{n-k}$ 是已知的信息位，而后 $n-k$ 位系数：$c_{n-k-1},c_{n-k-2},\cdots,c_1,c_0$ 是要求的校验位。码多项式必是生成多项式 $g(x)$ 的倍式，所以

$$C(x)=m(x)g(x) \tag{4-5}$$

其中 $\partial C(x) \leq n-1, \partial g(x) = n-k, \partial m(x) \leq k-1$ 而

$$h(x)C(x) = m(x)g(x)h(x) = m(x)(x^n - 1) = m(x)x^n - m(x) \tag{4-6}$$

由于中 $\partial C(x) \leq n-1, \partial g(x) = n-k, \partial m(x) \leq k-1$，所以 $m(x)x^n$ 的最低位次数至少为 n 次，因而在 $h(x)C(x)$ 的乘积中 $x^{n-1}, x^{n-2}, \cdots, x^k$ 的次数为 0。

而根据式4-6最左边的乘积可知，x^{n-1} 的系数：

$$c_{n-1-0}h_0 + c_{n-1-1}h_1 + \cdots + c_{n-1-k}h_k \tag{4-7}$$

同理，x^{n-2} 的系数：

$$c_{n-2-0}h_0 + c_{n-2-1}h_1 + \cdots + c_{n-2-k}h_k \tag{4-8}$$

根据上边的分析，

$$\sum_{j=0}^{k} c_{n-i-j}h_j = 0 \qquad i = 0, 1, 2, \cdots, n-k \tag{4-9}$$

由于 $h_k = 1$，所以式4-9可以改写为:

$$c_{n-k-i} = -\sum_{j=0}^{k-1} c_{n-i-j}h_j \qquad i = 1, 2, \cdots, n-k \tag{4-10}$$

式4-10展开为：

$$\begin{array}{rcl} c_{n-k-1} & = & -(c_{n-1}h_0 + c_{n-2}h_1 + \cdots + c_{n-k}h_{k-1}) \\ c_{n-k-2} & = & -(c_{n-2}h_0 + c_{n-3}h_1 + \cdots + c_{n-k-1}h_{k-1}) \\ & & \vdots \\ c_{n-k-(n-k)} & = & c_0 = -(c_k h_0 + c_{k-1}h_1 + \cdots + c_1 h_{k-1}) \end{array} \tag{4-11}$$

由上式看出码字 C 的第一个码元 c_{n-k-1} 可由 k 个信息元 $c_{n-1}, c_{n-2}, \cdots, c_{n-k}$ 与 $h(x)$ 的系数相乘得到，而由 $c_{n-2}, c_{n-3}, \cdots, c_{n-k}, c_{n-k-1}$ 可得到第二个校验元 c_{n-k-2}，再由 $c_{n-3}, \cdots, c_{n-k}$ 信息元和第一、第二校验元 c_{n-k-1}, c_{n-k-2} 可得到第三校验元 c_{n-k-3}。按这样的线性关系递推，一直可求得所有的 $n-k$ 个校验元 $c_{n-k-1}, c_{n-k-2}, \cdots, c_1, c_0$。因此，基于校验多项式 $h(x)$ 构造除法编码器结构图如4-2 由于此种结构图编码出来的码字是系统码，因此，在本设计中，采用的就是这种编码结构。

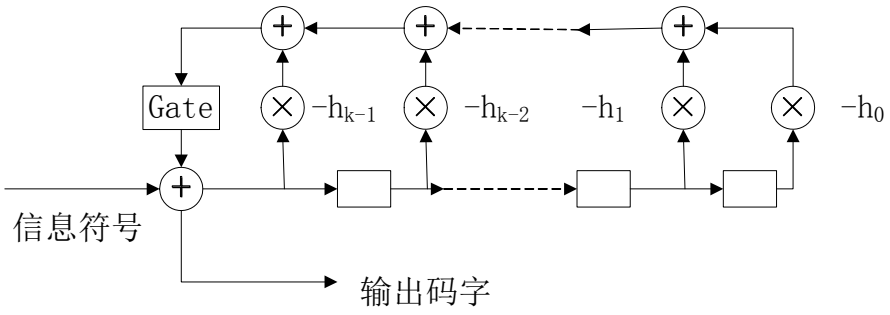

图 4-2 除法编码器结构图

4.4 RS 码的译码

就目前 RS 码的译码算法中，主要有硬判决算法和软判决译码算法，硬判决算法就是上面提到的 Berlekamp-Massey(BM) 算法，其纠错能力为 $t \leq (d_{min}-1)/2, d_{min} = n-k+1$ 为最小汉明距离。

而软判决译码算法有如下几种：

1. **由代数硬判决译码改进的准软判决译码算法**[3]： 由于删除信息可以看作最简单的软信息利用形式，Forney 提出的 GMD 译码算法通过删除最不可靠的码字，多次运行 BM 算法来实现译码，可以看作是 RS 码的准软译码算法；Chase 提出的 Chase 算法通过翻转最不可靠的码字后多次运行 BM 算法来实现译码。这两种算法的原理是很相似的，前者性能低，后者复杂度大。

2. **代数软判决译码算法**[4] 此类算法的代表就是 Koetter 和 Vardy 提出的 KV 算法[17]。KV 算法通过对不同的插值点分配不同的重数，将信道输出的软信息转换成代数约束条件，之后再执行 GS[5]算法[18]，实现了软判决译码。KV 算法可以方便的调节参数，实现复杂度和性能的折中。

3. **自适应的信度传播算法**[6] ABP 算法由 Jiang 等人于 2004 年提出[19]，它利

[3]GMD,Chase 算法

[4]ASD:Algebraic Soft Decision 算法

[5]由 Guruswami 和 Sudan 提出的代数列表译码算法

[6]ABP:Adaptive Belief Propagation 算法

用了在 LDPC 码译码中常用的 BP 算法。BP 算法要求校验矩阵式稀疏的，而 RS 码进行二进制展开后的校验矩阵仍然是高密度的。Jiang 通过在 BP 算法前对矩阵进行一定的处理，使其对应于置信度较低的矩阵式稀疏的，而后进行 BP 迭代，实现译码。

介绍以上方式，只是说明 RS 码应用很广泛，译码方法很成熟，但是针对于水声通信来说，不需要这么复杂的软判决译码算法，而 BM 算法的简单，易实现特点很适合于水声通信。下面就 Berlekmap-Massey 详细讲解。

4.5 Berlekmap-Massey 硬判决算法

4.5.1 Berlekamp-Massey 迭代原理[6]

众所周知分组码最典型的硬判决译码方法就是伴随式 (Syndrome) 译码，译码分为三个步骤：

1. 计算伴随式；
2. 由伴随式求取错误图样
3. 将错误图样与接收码字相加，得到译码输出

在第二步中，又分为求取错误位置和求取错误值，而 BM 算法本质上是求取错误位置多项式，而利用钱搜索计算错误位置多项式的根，而 Forney 算法来计算错误值，这两个算法会在第四小节介绍。下面对 BM 算法的原理做些介绍[20]。

把错误图样 $\vec{e}$ 也看做一个多项式：

$$e(x) = e_0 + e_1 x + e_2 x^2 + \cdots + e_{n-1} x^{n-1} \tag{4-12}$$

假如实际发生的错误为 $v, 0 \leq v \leq t, t$ 为 RS 码纠错能力 $(n - k = 2t)$，那么，错误多项式就可以写成仅有 v 项的形式：

$$e(x) = e_{i_1} x^{i_1} + e_{i_2} x^{i_2} + \cdots + e_{i_v} x^{i_v} \tag{4-13}$$

其对应的错误图样的形式为：$\vec{e} = (0 \cdots e_{i_1} \cdots 0 \cdots e_{i_2} \cdots\cdots 0 \cdots e_{i_v} \cdots 0)$，下标 i_l 在 $0 \sim n-1$ 整数中取值，代表错误的位置，l 在 $1 \sim v$ 取值。

知道，$\vec{V}=\vec{r}H^T=\vec{e}H^T$，代入 RS 码 H 矩阵的形式，则有：

$$\vec{S}=(0\cdots e_{i_1}\cdots 0\cdots e_{i_2}\cdots 0\cdots e_{i_v})\begin{bmatrix}1 & 1 & \cdots & 1\\ \alpha^1 & \alpha^2 & \cdots & \alpha^{2t}\\ (\alpha^1)^2 & (\alpha^2)^2 & \cdots & (\alpha^{2t})^2\\ \vdots & \vdots & \vdots & \vdots\\ (\alpha^1)^{n-1} & (\alpha^2)^{n-1} & \cdots & (\alpha^{2t})^{n-1}\end{bmatrix} \tag{4-14}$$

由矩阵乘法定义可知，$\vec{e}^T$ 和 H 的 $2t$ 个列向量进行求内积运算，$\vec{e}^T$ 与第 j 个列向量求内积求出 $\vec{S}$ 的第 j 个分量：

$$S_j=e_{i_1}\left(\alpha^j\right)^{i_1}+e_{i_2}\left(\alpha^j\right)^{i_2}+\cdots+e_{i_v}\left(\alpha^j\right)^{i_v}=e\left(\alpha^j\right),1\le j\le 2t \tag{4-15}$$

为了表达方便，假设 $Y_l=e_{i_l},1\le l\le v$；设 $X_l=\alpha^{i_l},1\le l\le v$，事实上，$Y_l$ 就是错误图样中某个错误位置上的错误值，而 X_l 就代表着错误位置 i_l，这样，上式4-15可以重写为：

$$S_j=Y_1X_1^j+Y_2X_2^j+\cdots+Y_VX_V^j,1\le j\le 2t \tag{4-16}$$

这可以看作一个方程组。设接收码字为 $r(x)=c(x)+e(x)$，因为 $\alpha^j,1\le j\le 2t$ 是 $c(x)$ 的根：$c(\alpha^j)=0$，由由式4-15，$r(\alpha^j)=e(\alpha^j)=S_j$，接收码字在接收端已知，可以求出 $2t$ 个伴随式 S_j，即完成伴随式译码的第一步。式4-16中未知的是 X_l,Y_l 的值，共 $2v$ 个。$v\le t$，可以通过解该方程组求得错误位置和错误值，但是这是个非线性方程组，求解过程非常复杂，因此采用先求错误位置，再求错误值。

定义次数为 v 错误位置多项式 $\sigma(x)$:

$$\sigma(x)=1+\sigma_1x+\sigma_2x^2+\cdots+\sigma_vx^v=(1-X_1x)(1-X_2x)\cdots(1-X_vx) \tag{4-17}$$

这个定义表明，错误位置 X_l 的逆元 X_l^{-1} 是 $\sigma(x)$ 的根。下面，目标就变成求 $\sigma(x)$ 及其根。

可以证明[21]，$\sigma(x)$ 的系数与伴随式 S_j 之间存在如下关系：

$$\begin{bmatrix}S_1 & S_2 & \cdots & S_v\\ S_2 & S_3 & \cdots & S_{v+1}\\ \vdots & \vdots & \vdots & \vdots\\ S_v & S_{v+1} & \cdots & S_{2v-1}\end{bmatrix}\begin{bmatrix}\sigma_v\\ \sigma_{v-1}\\ \vdots\\ \sigma_1\end{bmatrix}=-\begin{bmatrix}S_{v+1}\\ S_{v+2}\\ \vdots\\ S_{2v}\end{bmatrix} \tag{4-18}$$

而 Berlekamp 提出了一种基于上式4-18迭代求 $\sigma(x)$ 的方法，Massey[22] 发现了这种迭代算法和线性反馈移位寄存器的关系，因此，把这种实用的求解算法称为 BM

算法，下面研究 BM 迭代[23] 求 $\sigma(x)$ 的具体过程，式4-18的关系可以简写成：

$$S_j = -\sum_{i=1}^{v} \sigma(x)_i S_{j-i} = v+1, \cdots, 2v \tag{4-19}$$

第二章讲过，在 $GF(2^m)$ 上的加法求逆和加法等价，因此，可以把负号可以省略。

式4-19可以等效成一个线性反馈移位寄存器 (设 $v=3$)：

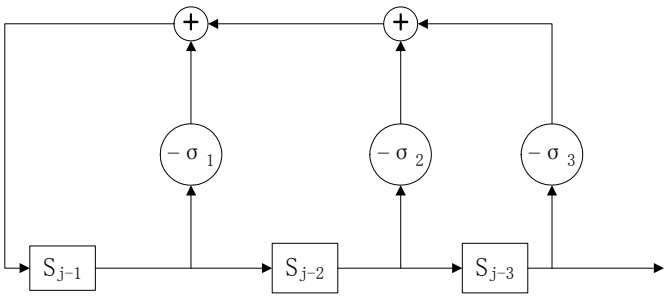

图 4-3　BM 迭代过程移位寄存器图

如果 $\sigma_1, \cdots, \sigma_v$ 已知，将寄存器中置入 $S_1, \cdots, S_v$ 的值，随着移位，在输出处就会产生 $S_{v+1}, \cdots, S_{2v}$ 的值。现在的问题是，$S_1, \cdots, S_{2t}$ 是知道的，$\sigma_1, \cdots, \sigma_v$ 未知。可以采用迭代的方法构造出一个移位寄存器来符合上面的输入输出关系。注意到，$\sigma(x)$ 的系数就是寄存器的抽头权重。

可以用数学归纳法完成对迭代过程的推导。假设，第 $r-1$ 次迭代，寄存器长度 L_{r-1}，此时的寄存器抽头权重为 $\sigma^{r-1}(x)$，它能够保证产生准确的 $S_1, \cdots, S_{r-1}$。下面的问题是怎么产生 $\sigma^r(x)$ 能够使它能产生 $S_1, \cdots, S_r$。当抽头权重为 $\sigma^{r-1}(x)$，前 $r-1$ 次移位，产生了 $S_1, \cdots, S_{r-1}$，第 r 次移位，输出端的值记作 S'_r：

$$S'_r = -\sum_{j=1}^{D(r-1)} \sigma_j^{r-1} S_{r-j} \tag{4-20}$$

其中 $D(r-1) = \deg \sigma^{r-1}(x)$。它可能不等于 S_r，定义它们的差值：

$$\Delta_r = S_r - S'_r = S_r + \sum_{j=1}^{D(r-1)} \sigma_j^{(r-1)} S_{r-j} = \sum_{j=0}^{D(r-1)} \sigma_j^{(r-1)} S_{r-j} \tag{4-21}$$

如果 $\Delta_r = 0$，说明 $\sigma^{r-1}(x)$ 也能够正确地产生 S_r，那么 $\sigma^r(x) = \sigma^{r-1}(x)$，否则，要对抽头系数做个修正：

$$\sigma^{(r)} = \sigma^{(r-1)}(x) + Ax^l\sigma^{(m-1)}(x) \tag{4-22}$$

这里的 $\sigma^{(m-1)}(x)$ 是前面 $r-1$ 次迭代里出现过的一个多项式，l 是一个整数，A 是一个有限域元素，它们的选择我们下面再分析，选择它们的依据就在于它们的值能够让系数修正以后的 $\Delta'_r = 0$。

对这个新的多项式 $\sigma^r(x)$，它的输出和 S_r 的差值为：

$$\Delta'_r = \Delta_r + A\sum_{j=0}^{D(m-1)} S_{r-j-l} \tag{4-23}$$

可以选择前面迭代过程中出现过的一个多项式 $\sigma^{m-1}(x)$，满足 $\Delta_m \neq 0$，令 $l = r-m$，则上式4-23第二项变成了 $A\Delta_m$，再选择 $A = -\Delta_r/\Delta_m$，则有：

$$\Delta'_r = \Delta_r - \frac{\Delta_r}{\Delta_m}\Delta_m = 0 \tag{4-24}$$

这样就满足了 $\Delta'_r = 0$ 的条件，即：$\sigma^r(x)$ 能够产生 S_r。可以证明，$S_1, \cdots, S_r$ 同能也能够由 $\sigma^r(x)$ 产生，事实上，式4-22可以看作把 $\sigma^{m-1}(x)$ 连接在 $\sigma^{r-1}(x)$ 上的辅助寄存器，当 $\sigma^r(x)$ 进行前 $r-1$ 移位时，它始终产生 0，只有在第 r 次移位时，产生一个非零值 Δ_m，经过系统 A 修正后，正好用来补偿 $\sigma^{r-1}(x)$ 的输出，使其输出 S_r，这就是通过修正的方法产生 $\sigma^r(x)$ 的基本思想。

如果只按照上面的说明，这样的 $\sigma^{m-1}(x)$ 不是唯一的，这是不允许的。在 BM 算法中有一个约束条件[24]，就是 $\sigma^r(x)$ 的次数要尽可能低，因为 $\sigma(x)$ 的次数就是错误的个数，在译码时总认为错误少的概率比错误多的概率高。事实上，在陪集首译码中作为陪集首的错误图样也要求重量尽可能轻，也是一样的道理。这些都是为了保证我们的译码是最小距离译码，即选择于接收码字汉明距离最小的码字作为译码输出。为了满足这个条件，在选择 m 的时候，需要选择的是出现在 $\sigma^{r-1}(x)$ 前面，刚刚改变长度的那个 $\sigma^{m-1}(x)$，即 $L_{r-1} = L_M > L_{m-1}$。这样可以保证产生的 $\sigma^r(x)$ 始终次数最低。经过这样的迭代，迭代 $2t$ 次，就可以得到最终的错误位置多项式 $\sigma^r(x)$。

4.5.2 BM 算法的主要步骤及实现[7]

下面，给出 BM 迭代的流程图4-4。流程图中的 $B(x)$ 相当于前面说的 $\sigma^{m-1}(x)$，两者只相差一个系数，而这个系数实际上是原来系数 A 里面分配了义部分。计算 $T(x)$ 的多项式实际上就是式4-20。具体步骤分析如下：

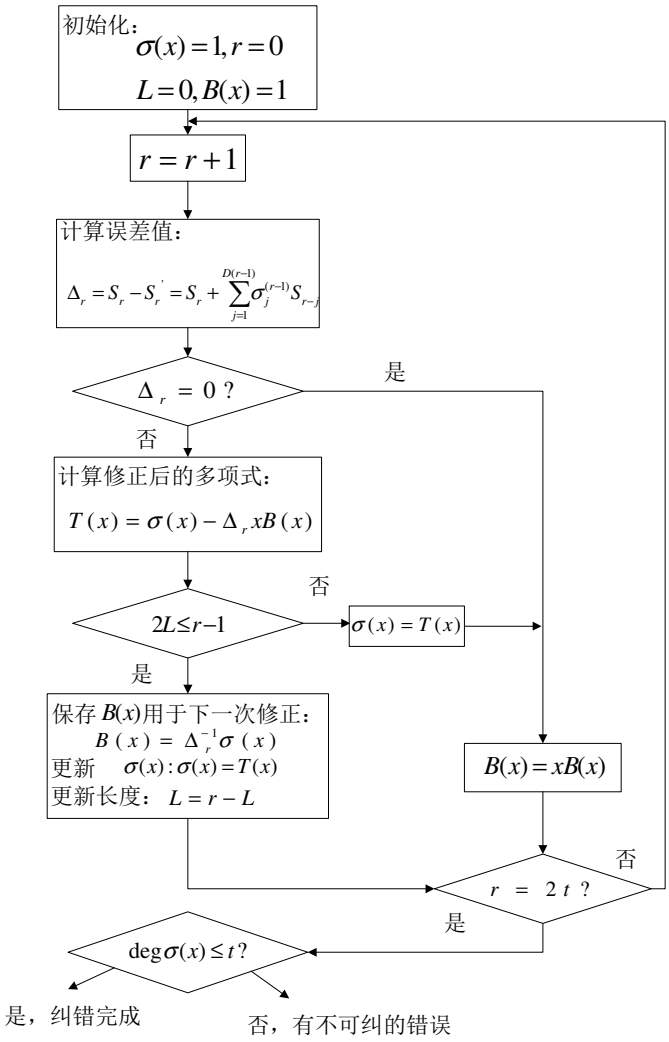

图 4-4 BM 迭代流程图

1. 初始化：$\sigma(x), r=0, L=0, B(x)=1$，然后开始迭代 $r=r+1$；

2. 按 $\Delta_r = S_r + \sum_{j=1}^{D(r-1)} \sigma_j^{(r-1)} S_{r-j}$ 计算 Δ_r，若 $\Delta_r = 0$，则有 $B(x) = xB(x)$，并计算 Δ_{r+1}，再进行下一次迭代；

3. 若 $\Delta_r \neq 0$，计算修正后的多项式 $T(x) = \sigma(x) - \Delta_r x B(x)$，若 $2L \leq r-1$ 不成立，则 $\sigma(x) = T(x), B(x) = xB(x)$，如果成立，继续；

4. 保存 $B(x)$ 用于下一次修正 $B(x) = \Delta_r^{-1}\sigma(x), \sigma(x) = T(x), L = r - L$，判断 $r = 2t$，如果否，再进行下一次迭代；如果是，继续；

5. 如果 $\sigma(x)$ 的最高次数小于等于 t 那么纠错完成，否则，不可纠错。

4.5.3 钱搜索与 Forney 算法

求得 $\sigma(x)$，下一步的问题就是求得多项式的根。工程上使用钱搜索方法来确定错误位置，钱搜索的原理在于有限域的元素是有限域的。可以逐个代入进行验证。这部分比较容易理解，在此做简要说明

$$\begin{aligned}\sigma(x) &= (1 - X_1 x)(1 - X_2 x)\cdots(1 - X_v x) \\ &= \prod_{l=1}^{v}(1 - X_l x) = \sigma_v x^x + \sigma_{v-1}x^{v-1} + \cdots + \sigma_1 x + \sigma_0\end{aligned} \tag{4-25}$$

首先验证位置 x^0 是否有错误，即把 $x = \frac{1}{\alpha^0} = \alpha^n (\alpha^n = 1)$ 代入 $\sigma(x)$ 若结果等于 0，说明第一个码元有错误，否则第一个码元是正确的。

把 $x = \frac{1}{\alpha^1} = \alpha^{n-1}$ 代入 $\sigma(x)$，验证位置 x^1 是否有错误。同理把 $\alpha^{n-2}, \alpha^{n-3}, \cdots, \alpha$ 代入 $\sigma(x)$ 分别验证位置 $x^2, \cdots, x^{n-1}$ 是否错误。

下面还需要求取错误值 Y_l，这里利用的是 Forney 算法。定义错误值多项式：

$$\Omega(x) = S(x)\sigma(x) \mod x^{2t} \tag{4-26}$$

这个式子又可以称作关键方程，其中 $S(x) = 1 + S_1 x + S_2 x^2 + \cdots + S_{2t} x^{2t}$，通过上面的步骤，求得了 $S(x)$ 和 $\sigma(x)$，因此 $\Omega(x)$ 可以很快求出，而 $\Omega(x)$ 还可以写为[21]：

$$\Omega(x) = x\sum_{i=1}^{v} Y_i X_i \prod_{l \neq i}(1 - X_l x) \tag{4-27}$$

将 $\sigma(x)$ 的某个根 X_l^{-1} 代入，就可以求出对应位置的错误值：

$$Y_l = \frac{\Omega(X_l^{-1})}{\prod_{j \neq l}(1 - X_j X_l^{-1})} \tag{4-28}$$

这样，错误图样就被确定，将接收码字和错误图样相加，就得到了译码码字。

4.5.4 BM 算法性能仿真

通过 matlab 仿真信道特性以及调制方式，而 C 语言实现编译码方式，最后得到的数据在 matlab 里编程画出误码率曲线图。

图4-5是 $RS(15,9,7)$ 码在 PSK 调制方式下的误码率仿真图。

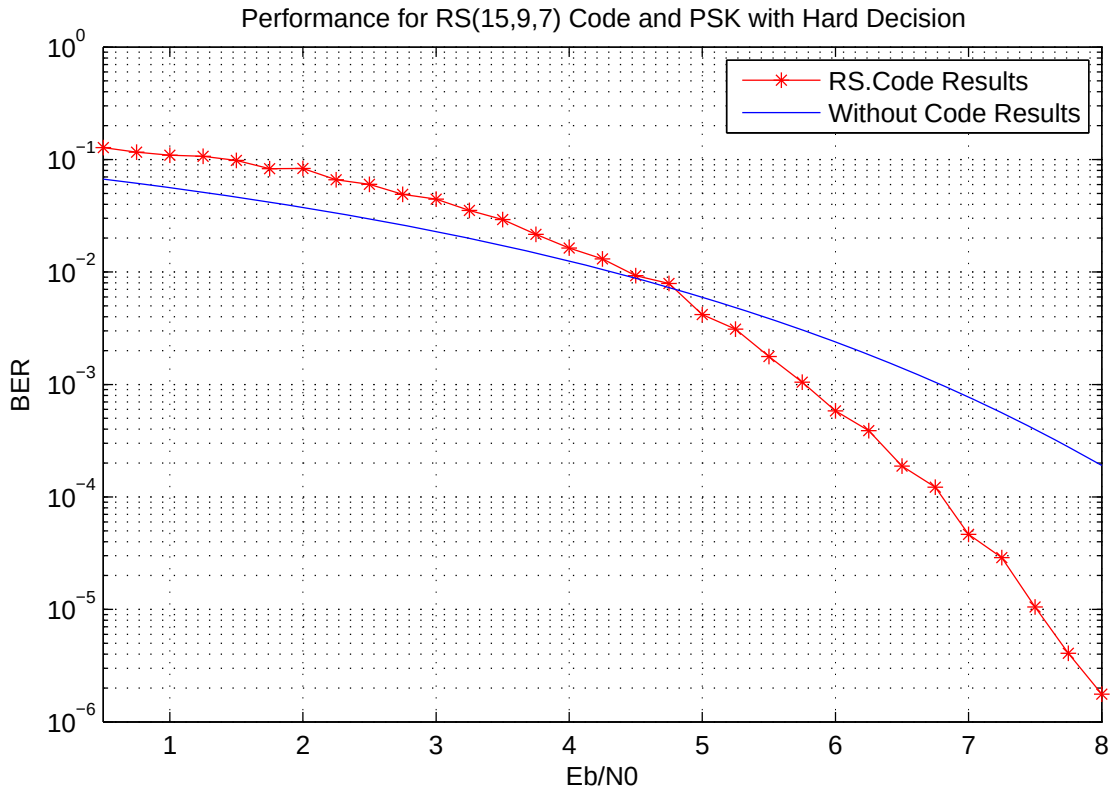

图 4-5 $RS(15,9,7)$ 在 PSK 调制下误码率曲线图

从图4-5可以看出：RS 码 BM 译码算法比未编码的信道误码率要好的多。

E_b/N_0 低于 4.75 的时候，未编码比编码效果好，这是一种误解，因为 RS 码是多进制码，而本设计采用的是四进制，因此，在相同信号幅值和噪声的情况下，未编码码的 E_b/N_0 是 RS 码的四倍，即未编码在 $E_b/N_0=1$ 的误码率应该还 RS 码 $E_b/N_0=4$ 的误码率相比，从图中可以看出，还是 RS 码误码率低。

4.6 本章小结

本章从 RS 码概述，编码器，译码器设计等方面介绍了 RS 码，其中编码器讨论了乘法形式的编码器和除法形式的编码器，而我们采用的是基于 $h(x)$ 除法形式的编码器，因为，这种方式编出来的码字是系统码，然后从理论和具体实现步骤上介绍了 BM 算法，当然，为了译码完整，还简单介绍了用于求错误位置的钱搜索和求错误值的 Forney 算法，最后仿真验证性能。

第 五 章　PC/104 运行分析

5.1　PC/104 简介

PC/104(pc104) 是 ISA(IEEE-996) 标准的延伸。1992 年 PC/104 作为基本文件被采纳，叫做 IEEE-P996.1 兼容 PC 嵌入式模块标准。PC/104 是一种专门为嵌入式控制而定义的工业控制总线。IEEE-P996 是 ISA 工业总线规范，IEEE 协会将它定义 IEEE-P996.1，PC/104 实质上就是一种紧凑型的 IEEE-P996，其信号定义和 PC/AT 基本一致，但电气和机械规范却完全不同，是一种优化的、小型、堆栈式结构的嵌入式控制系统。其小型化的尺寸（90 × 96mm），极低的功耗（典型模块为 1 － 2 瓦）和堆栈的总线形式 (决定了其高可靠性)，受到了众多从事嵌入式产品生产厂商的欢迎，在嵌入式系统中领域逐渐流行开来。

PC/104 的优点

- **大尺寸**

  PC/104 的板卡标准尺寸为 90 × 96mm(比一本新华字典还要小很多，而传统桌面 PC 系统的板卡尺寸为 315 × 122mm)，这样小的尺寸使得 PC/104、PC/104+ 和 PCI-104 模块板成为了嵌入式系统应用的理想产品。

- **开放的高可靠性的工业规范**

  PC/104、PC/104+ 和 PCI-104 产品在电气特性和机械特性上可靠性极高，功耗低，产生热量少。板卡与板卡之间通过自堆栈进行可靠的连接，抗震能力强。全世界有超过 200 家公司使用这些开放的规范来生产和销售各种 PC/104 模块板。

- **模块可自用扩展**

  PC/104 模块具有灵活的可扩展性。它允许工程师互换及匹配各种功能卡，可随系统的需求而升级 CPU 的性能。增加系统的功能和性能只需通过改变相应的模块即可实现。

- **低功耗**

  4mA 的总线驱动电流，即可使模块正常工作，低功耗有利于减少元件数量。

各种插卡广泛采用 VLSI 芯片、低功耗的 ASIC 芯片、门阵列等，其存储采用大容量固态盘（SSD）。

- **堆栈式连接**

  这种结构取消了主板和插槽，可以将所有的 PC/104 模块板利用板上的叠装总线插座连接起来。有效减小整个系统所占的空间。PC104 板的叠装总线插座是针脚插接方式，理论上可以无限扩插 N 多扩展卡，但要看他的承受能里。

5.2 卷积码序贯译码在 PC/104 上运行

在 PC/104 上，测试 $POLY1 = 0xA5048D, POLY2 = 0xDAFB73$ 约束长度为 $K = 24$ 的卷积码的运行效率与误码率，下表就是测试的一些数据，其中测试的数据长度为 $L = 1000000$ 比特，既是 125000 字节，又根据第三章编码中，终结码介绍可知，加上 24 比特，这样可以实现结尾的误码率不受结尾码字的影响。测试环境为 BPSK 调制，AWGN 信道。表5-1就是运行得到的数据。从表5-1，结合参考书

表 5-1　约束长度 $K = 24$ 的卷积码测试数据表

E_b/N_0	2.0	2.25	2.5	2.75	3.0	3.25
比特错误率	1.001×10^{-1}	8.93×10^{-2}	6.24×10^{-2}	2.65×10^{-2}	9.5×10^{-3}	5.8×10^{-3}
E_b/N_0	3.50	3.75	4.0	4.25	4.5	4.75
比特错误率	1.7×10^{-3}	7.8×10^{-4}	3.12×10^{-4}	1.15×10^{-4}	1.0×10^{-5}	1.01×10^{-6}

上给出的实例可以看出，误码率还是符合理论要求的。因此，现在要测试的就是运行的时间，译码运行时间的大小，决定译码时候缓冲区有无以及大小。表5-2将给出在各个 E_b/N_0 下，译码器运行的时间，其中码长为 1000024 比特。从图中可以看出，序贯译码，在 E_b/N_0 很低的时候，运行时间需要很长，这是因为，低 E_b/N_0 情况下，序贯译码需要大量的运算，因此回溯次数会增加。而 E_b/N_0 越高相对需要的时间越少。

从总体考虑，缓冲区还是需要的，尤其是传输码率和数据量很大的时候，至于

表 5-2 约束长度 $K = 24$ 的卷积码运行时间表

E_b/N_0	2.0	2.25	2.5	2.75	3.0	3.25
运行时间 (s)	59.16	50.87	46.39	44.49	40.34	39.50
E_b/N_0	3.50	3.75	4.0	4.25	4.5	4.75
运行时间 (s)	39.99	35.45	33.57	32.90	30.00	28.5

缓冲区大小的确定这要根据数据量的多少，码字传输速率，E_b/N_0 等情况综合考虑。

5.3 RS 码 BM 算法译码在 PC/104 上运行

在 PC/104 上，测试 $RS(15, 9, 7)$ 的运行效率与误码率，下表就是测试的一些数据，错误的帧数为 200，且每帧 36 比特，高于此帧数的时候就停止输入数据，停止编译码。测试环境为 BPSK 调制，AWGN 信道。

表 5-3 $RS(15, 9, 7)$ 测试数据表

E_b/N_0	0.5	0.75	1	1.25	1.5	1.75
比特错误率	1.3×10^{-1}	1.2×10^{-1}	1.0×10^{-1}	1.0×10^{-1}	9.8×10^{-2}	8.3×10^{-2}
字节错误率	6.4×10^{-1}	6.2×10^{-1}	6.0×10^{-1}	5.8×10^{-1}	5.2×10^{-1}	4.7×10^{-1}
帧错误率	9.4×10^{-1}	9.0×10^{-1}	8.9×10^{-1}	8.7×10^{-1}	8.5×10^{-1}	7.6×10^{-1}

E_b/N_0	2	2.25	2.5	2.75	3	3.25
比特错误率	8.3×10^{-2}	6.0×10^{-2}	6.1×10^{-2}	4.9×10^{-2}	4.4×10^{-2}	3.5×10^{-2}
字节错误率	4.6×10^{-1}	3.7×10^{-1}	3.5×10^{-1}	3.0×10^{-1}	2.7×10^{-1}	2.2×10^{-1}
帧错误率	7.5×10^{-1}	6.4×10^{-1}	5.9×10^{-1}	5.2×10^{-1}	4.7×10^{-1}	3.9×10^{-1}

E_b/N_0	3.5	3.75	4	4.25	4.5	4.75
比特错误率	2.9×10^{-2}	2.2×10^{-2}	1.6×10^{-2}	1.3×10^{-2}	9.3×10^{-3}	7.9×10^{-3}
字节错误率	1.8×10^{-1}	1.3×10^{-1}	1.1×10^{-1}	8.6×10^{-2}	6.0×10^{-2}	5.2×10^{-2}
帧错误率	3.2×10^{-1}	2.4×10^{-1}	2.0×10^{-1}	1.6×10^{-1}	1.1×10^{-1}	9.9×10^{-2}

E_b/N_0	5	5.25	5.5	5.75	6	6.25
比特错误率	4.2×10^{-3}	3.1×10^{-3}	1.8×10^{-3}	1.1×10^{-3}	5.8×10^{-4}	3.9×10^{-4}
字节错误率	2.9×10^{-2}	1.9×10^{-2}	1.2×10^{-2}	7.3×10^{-3}	3.9×10^{-3}	2.6×10^{-3}
帧错误率	5.5×10^{-2}	3.7×10^{-2}	2.3×10^{-2}	1.4×10^{-2}	7.5×10^{-3}	5.0×10^{-3}

E_b/N_0	6.5	6.75	7	7.25	7.5	7.75
比特错误率	1.9×10^{-4}	1.2×10^{-4}	4.6×10^{-5}	2.9×10^{-5}	1.1×10^{-5}	4.1×10^{-6}
字节错误率	1.3×10^{-3}	8.1×10^{-4}	3.1×10^{-4}	1.9×10^{-4}	6.8×10^{-5}	2.8×10^{-5}

续下页

续表 5-3　$RS(15,9,7)$ 测试数据表

帧错误率	2.4×10^{-3}	1.6×10^{-3}	6.1×10^{-4}	3.6×10^{-4}	1.4×10^{-4}	5.7×10^{-5}

表5-3测试了 E_b/N_0 从 2 到 7.75 步进为 0.25 的比特错误率，字节错误率和帧错误率，表格很细致的反映了 RS 误码率的走势和未经过信道编码的 BPSK 调制误码率的比较，从而可以看出，RS 还是很好的降低了误码率，满足了我们的要求。

表 5-4　$RS(15,9,7)$ 运行时间表

E_b/N_0	0.5	0.75	1	1.25	1.5	1.75
运行时间 (s)	1.0×10^{-2}	1.0×10^{-2}	1.0×10^{-2}	1.0×10^{-2}	1.0×10^{-2}	2.0×10^{-2}
总比特数	7.7×10^{3}	8.0×10^{3}	8.1×10^{3}	8.4×10^{3}	8.5×10^{3}	9.6×10^{3}

E_b/N_0	2	2.25	2.5	2.75	3	3.25
运行时间 (s)	1.0×10^{-2}	2.0×10^{-2}	2.0×10^{-2}	1.0×10^{-2}	3.0×10^{-2}	2.0×10^{-2}
总比特数	9.7×10^{3}	1.1×10^{4}	1.2×10^{4}	1.4×10^{4}	1.5×10^{4}	1.9×10^{4}

E_b/N_0	3.5	3.75	4	4.25	4.5	4.75
运行时间 (s)	4.0×10^{-2}	3.0×10^{-2}	7.0×10^{-2}	5.0×10^{-2}	9.0×10^{-2}	1.2×10^{-1}
总比特数	2.2×10^{4}	3.0×10^{4}	3.6×10^{4}	4.4×10^{4}	6.5×10^{4}	7.3×10^{4}

E_b/N_0	5	5.25	5.5	5.75	6	6.25
运行时间 (s)	1.7×10^{-1}	2.8×10^{-1}	4.2×10^{-1}	6.7×10^{-1}	1.2×10^{0}	2.1×10^{0}
总比特数	1.3×10^{5}	1.9×10^{5}	3.1×10^{5}	5.1×10^{5}	9.6×10^{5}	1.4×10^{6}

E_b/N_0	6.5	6.75	7	7.25	7.5	7.75
运行时间 (s)	3.4×10^{0}	7.4×10^{0}	1.4×10^{1}	3.4×10^{1}	7.1×10^{1}	1.7×10^{2}
总比特数	2.8×10^{6}	4.5×10^{6}	1.2×10^{7}	2.0×10^{7}	5.3×10^{7}	1.3×10^{8}

从表5-4中可以看出，随着 E_b/N_0 的升高，运行时间一直在增加，这里有个误解，事实上并非如此，由于随着 E_b/N_0 的升高，误码率是下降的，为了精确度的原因，必然需要越多的数据进行测试，才能达到误码率精度的要求，因此随着 E_b/N_0 的升高，测试数据也增多，相应的运行时间也是增多的。

通过表中运行时间和测试数据量的关系可以看出，运行时间和 E_b/N_0 并无关系，这也是理论上支持的。而且从运行时间上看，译码算法还是很高效的，在加上本设计的目的和水声通信的要求，RS 编译码只用于少量数据的传输，因此缓冲区可以省掉。

5.4 本章小结

本小节首先介绍了 PC/104，并说明为什么选择 PC/104 作为我们测试译码算法的平台，其次在给出了序贯译码算法和 BM 译码算法在 PC/104 上运行的结果，包括测试误码率数据和运行时间，以及讨论关于缓冲区的选取和大小的选择。

第六章　结论与展望

6.1　研究工作总结

由于人们对海洋的关注越来越多，人类在海洋中的活动越来越频繁，因此作为海洋技术基础的水声通信，必然也是人们关注的焦点，而信道编码作为水声通信系统可靠性的保障，也必然研究广泛。

本文分别从编码理论、卷积码编译码已经 RS 码编译码三个方面介绍水声通信的信道编码。

1. 编码代数理论，作为信道编码的基础，有必要认真的学习一番，这是我们理解和应用各种信道编码的前提，因此在最初，就介绍了信道编码的相关代数基础。

2. 卷积码的编译码，从原理和实现两个方面介绍了整个编译码过程，着重讨论了序贯译码方式的费诺算法，并仿真和分析性能。

3. RS 码的编译码，也是从原理和实现两个方面介绍了整个编译码过程，着重讨论了 BM 译码算法，并仿真和分析性能。

6.2　下一步工作

6.2.1　费诺算法的进一步改进

在上述章节中提到的费诺算法，我们知道，其运算量很大，尤其是在信噪比很低的时候，因此会有很大的延时，这是系统不能容忍的，而与之对应的另外一种序贯译码算法——多堆栈算法，却能避免大量的运算，只是存储空间的浪费比费诺算法严重的多。

结合这两种算法，使其具有多堆栈算法的效率，又有费诺算法的节省存储空间。

6.2.2 Turbo 码等其他码字的研究

由于水声通信和编码技术的日益发展，其他一些新的码字也渐渐地显示出自己的优势，而 Turbo 无疑是其中的佼佼者，因此，Turbo 码也是下一步工作的重点。

最后还有级联码，因为已经做好了卷积码和 RS 码的编译码，因此把 RS 码作为外码，卷积码作为内码的级联码的仿真也是一个重点。

参考文献

[1] 张禾瑞. *近世代数基础*. 人民教育出版社, 1952.

[2] 向茜，刘钊. 伽罗华域上代数运算的最简实现. *电子科技大学学报*, 29(1):5--8, 2000.

[3] Jurgen Freudenberger Ander Neubauer and Volker Kuhn. *Coding Theory Algorithms,Architectures and Applications*. POSTS& TELECOM PRESS, 2009.

[4] 陈鲁生，沈世镒. *编码理论基础*. 高等教育出版社, 2010.

[5] 周炯槃，庞沁华，续大我，吴伟陵. *通信原理 (合订本)*. 北京邮电大学出版社, 2005.

[6] Jonathan Hong and Martin Vetterli. A heuristic dicussion of probabilistic decoding. *IEEE TRANSACTIONS ON COMMUNICATIONS*, 43(8):64--73, 1963.

[7] Gema M.Diaz-Toca Nadia Ben Atti and Henri Lombardi. The berlekamp-massey algorithm revistited, 2000.

[8] John G.Proakis. *Digital Communications Fourth Edition*. Publishing House of Electronics Industry, 2008.

[9] Jr. George C.Ckark and J.Bibb Cain. *Error-Correction Coding for Digital Communications*. PLENUM PRESS nEW YORK AND LONDON, 1995.

[10] J.B.Anderson and S.Mohan. Sequential coding algorithms:a survey and cost analysis. *IEEE TRANSACTIONS ON COMMUNICATIONS*, 32(2):169--176, 1984.

[11] J.M.Geist. Am empirical comparison of two sequential decoding algorithms. *IEEE TRANSACTIONS ON COMMUNICATIONS*, 19(4):415--419, 1971.

[12] 孙军. 卷积码的序列译码算法及信道模拟. *北京邮电大学*, pages 1--25, 2006.

[13] Jr.G.D.Forney and E.K.Bower. A high-speed sequential decoder:prototype design and test. *IEEE TRANSACTIONS ON COMMUNICATIONS*, 19(5):821--835, 1971.

[14] Yunghsiang S. Han. Sequential decoding of binary convolutional codes, 2005.

[15] ROBERT M. FANO. Simple algorithms for bch decoding. *IEEE TRANSACTIONS ON INFORMATION THEORY*, 43(8):2324--2333, 1995.

[16] G.Solomon I.S.Reed. Search properties of some sequential decoding algorithms. *IEEE TRANSACTIONS ON INFORMATION THEORY*, 19(4):519--526, 1973.

[17] A.Vardy R.Koetter. Algebraic soft-decision decoding of reed-solomon codes. *IEEE Information Theory*, 49(8):2809--2825, 2003.

[18] M.Sudan V.Guruswami. Improved decoding of reed-solomon and algebraic-geometry codes. *IEEE Trans. Information Theory*, 45(8):1757--1767, 1999.

[19] K.R.Narayanan J.Jiang. Iterative soft-input-soft-output decoding of reed-solomon codes by adapting the parity check matrix. *IEEE Information Theory*, 52(8):3746--3756, 2006.

[20] 陈文礼. Rs 纠错编码原理及其实现方法, 2000.

[21] Richard E.Blahut. Theory and practice of error control codes, 1983.

[22] J.L. Massey. Shift register synthesis and bch decoding. *IEEE TRANSACTIONS ON INFORMATION THEORY*, 15(1):122--127, 1969.

[23] E.R.Berlekamp. Algebraic coding theory. *McGraw-Hill*, (7), 1968.

[24] U.Cheng. On the continued fraction and berlekamp's algorithm. *IEEE TRANSACTIONS ON INFORMATION THEORY*, 30(30):541--544, 1984.

个人简历及论文发表

唐怀东，男，1988 年生于江苏省徐州市。2006-2010 年就读于北京邮电大学电信工程学院，获得学士学位；2010-2013 期间就读于中国科学院声学研究所海洋声学技术实验室，获得信号与信息处理硕士学位。

硕士期间的主要工作是水声通信中 Turbo 均衡的研究。研究工作基于国家 863“水声通信网络节点及组网关键技术”课题，目的是设计适用于深海通信的信道估计与均衡算法。在整个研究过程中对已有的水声通信信道估计，相位补偿以及信道均衡算法进行了对比分析，结合课题背景选取适合的算法进行深入的研究，并提出一定的改进方法，最终形成本篇论文。

硕士生期间发表的学术文章：

1. Huaidong Tang, Min Zhu, Yanbo Wu, Lijun Xu, Zeping Xing. Inter-frame Interleaved Bi-SOVA Algorithm for Underwater Acoustic Communication, The second International COnference on Consumer Electronics, Communications and Networks, 2012, 3:1865-1869

2. 唐怀东，朱敏，武岩波. 一种水声通信 Turbo 均衡中的软迭代信道估计算法，电子与信息学报，

3. 唐怀东，朱敏，武岩波. 一种基于先验信息 MMSE 准则的水声 Turbo 均衡与软迭代信道估计算法

致谢

本论文的研究工作是在我的导师朱敏研究员的悉心指导下完成的。朱老师做事严谨、知识丰富且待人和善。作为科研工作者，朱老师有渊博的知识和丰富的经验；作为项目负责人，大到总体框架，小到一个小的电路板的设计都了然于胸；作为一名导师，更是谆谆善诱、诲人不倦。正是在这样的导师的指导下，我才能在技术上和做事上有了较大的提高。在此，对朱老师衷心的说一声“谢谢”！

感谢朱维庆教授。朱维庆教授是实验室的创始人，正是有了老先生的指导，实验室多年来蒸蒸日上，为我们研究生的成长提供了优越的环境。当日选择来中科院读研究生，也是希望能目睹中科院老前辈们的工作风采，以受熏陶，如今愿望成真，老先生的工作热情鼓舞了我追求自己梦想的勇气。

感谢武岩波老师。初到实验室，不知所措，是他，让我慢慢适应环境，他的细心指点、自己的学习体会、学习规划以及未来研究方向，都热情地给我讲述，让我感慨自己知识的狭窄，惊喜于未来知识的广阔。在毕设开始以致结束整个过程，武岩波总是为我解开各种疑惑和问题，并对设计提出非常有意义的建议，在论文撰写的时候，他也给我提出很多结构上的问题，让论文条理清晰。再次感谢武岩波。

感谢水声通信网项目组的徐立军、傅翔、刘烨瑶、杨波、李欣国、魏振坤等，感谢他们对我学习和生活上的指导和照顾，感谢他们陪伴我度过千岛湖实验那段美好的时光。

感谢王季煜师兄和李海莲师姐。他们对学业认真的态度，对问题冷静的分析方法，都是值得我学习的。他们不仅在学习上给予我诸多指点，在生活上也对我颇多照顾。

感谢我的同学崔兴隆、赵二亮、马驰、陈若婷，感谢我的师弟师妹许浩、张威、李丹丹、曹松军、樊艳强，陪伴我度过研究生时光。

最后，我要感谢的是我最亲爱的父母。在我二十多年的成长过程中，你们无时无刻无私地关怀和奉献，是我独在异乡求学的最大精神支柱，也是我可以依偎的最温馨港湾。你们是我永远的牵挂和眷念！谨以此文献给我挚爱的双亲。