

学 号: 0121807780613

# 武汉理工大学

## 课 程 设 计

题 目: 基于卷积码差错控制系统实现

学 院: 信息工程学院

专 业: 通信工程

班 级: 通信 1803

学 号: 0121807780613

姓 名: 王胜鹏

指导教师: 胡辑伟

2021 年 06 月 20 日

## 摘要

信道编码是数字通信系统中的重要组成部分，它是保证信号可靠传输的一种重要方式。卷积码以其优越的性能被广泛使用在数字通信系统中。 $(2,1,7)$  卷积编码已经是国际卫星通信的标准。

本课程设计主要应用 SIMULINK 设计出基于卷积码的差错控制系统仿真模型，并通过 MATLAB 对系统进行性能仿真分析。解决对一个卷积码序列进行维特比 (Viterbi) 译码输出，分别从卷积编码软判决和卷积编码硬判决以及无编码三种情况下仿真控制系统的误码率情况，以 BPSK 调制方式为例，并通过搭建 simulink 模块协同 Matlab 脚本编写进行设计与仿真，并进行误码率分析。

采用加性高斯白噪声信道，采用 BPSK 解调并分别用软决策和硬决策下的维特比译码，利用 MATLAB 协同 simulink 对决策方式、卷积编码效率、维特比译码回溯深度自动调参，研究其在不同信噪比下维特比译码的误比特率关系，通过曲线图得出仿真与理论分析一致。

**关键词：**软判决，硬判决，simulink，维特比译码，回溯深度，卷积码

## Abstract

Channel coding is an important part of the digital communication system, which is an important way to ensure the reliable transmission of signals. The convolutional code is widely used in a digital communication system with its superior performance. (2, 1, 7) Convolution Coding is already the standard for international satellite communications.

This course design mainly applies SIMULINK to design the emulation model based on convolutional code-based error control system, and perform performance simulation analysis of the system through MATLAB. Solve a Viterbi decoding output, respectively, from the convolution encoded soft decision and convolution encoding hard decision and the error rate of the control system, in BPSK modulation mode Take an example, and coexue the MATLAB scripting by building a Simulink module to prepare the MATLAB script and perform a bit error rate analysis.

Using an additive Gaussian white noise channel, BPSK demodulation and Viterbi decodes are used to draw simulation and theoretical analysis by drawing different signal-to-noise ratios.

**Keywords:** Soft decision, hard decision, simulink, Viterbi Decodes, Traceback depth, Convolution code

# 目 录

<b>1 絮论 . . . . .</b>	<b>1</b>
<b>2 通信系统设计 . . . . .</b>	<b>2</b>
2.1 信道编码 . . . . .	2
2.2 卷积码 . . . . .	3
2.2.1 卷积码的结构 . . . . .	4
2.2.2 卷积码的图解表示 . . . . .	5
2.2.3 维特比译码 . . . . .	9
2.2.4 软判决 . . . . .	12
2.2.5 硬判决 . . . . .	12
<b>3 Simulink 单元模块设计 . . . . .</b>	<b>14</b>
3.1 整体设计流程 . . . . .	14
3.2 卷积码的差错控制系统仿真模型 . . . . .	14
3.2.1 总体设计框图 . . . . .	14
3.2.2 信源子系统 . . . . .	14
3.2.3 信道 . . . . .	17
3.2.4 信宿子系统 . . . . .	17
3.3 量化判决 . . . . .	18
3.4 matlab 和 simulink 交互式仿真 . . . . .	20
<b>4 卷积编码实现与仿真 . . . . .</b>	<b>21</b>
4.1 BPSK 卷积编码硬判决仿真 . . . . .	21
4.1.1 卷积速率对误码率的影响 . . . . .	21
4.1.2 回溯深度对误码率的影响 . . . . .	23
4.2 BPSK 卷积编码软判决仿真 . . . . .	25
4.2.1 卷积速率对误码率的影响 . . . . .	25
4.2.2 回溯深度对误码率的影响 . . . . .	26
4.3 matlab 仿真 64QAM 下的卷积编码 . . . . .	28
<b>5 心得体会 . . . . .</b>	<b>32</b>

# 1 绪论

卷积码（convolutional code）是由伊莱亚斯 (P.Elias) 于 1954 年首先提出的一种非分组码。通常它更适用于前向纠错，因为对于许多实际情况它的性能优于分组码，而且运算较简单。无论从理论上还是实际上均已证明其性能优于线性分组码。近年来众多有关卷积码研究结果表明，卷积码最有希望实现香农信道编码定理。但卷积码在译码理论及实际应用较为复杂，这些缺点限制了其进一步发展和应用。

维特比译码算法由维特比 (Viterbi) 1964 年提出，算法实质是最大似然译码，但它利用了编码网格图的特殊结构，在网格图中选择一条路径，使相应的译码序列与接收到的序列之间的汉明距 (即量度) 最小的一种最大似然译码方法，从而大大降低了计算的复杂性。

卷积码在通信系统中的广泛应用卷积码是一种性能优越的信道编码，它的编码器和译码器都比较容易实现，同时也具有较强的纠错能力，随着纠错编码理论研究的不断深入，卷积码的实际应用越来越广泛。卷积码是一种向前纠错控制编码。它将连续的信息比特序列映射为连续的编码器输出符号。这种映射是高度结构化的，使得卷积码的译码方法与分组码译码所采用的方法完全不同。可以验证的是在同样复杂度情况下，卷积码的编码增益要大于分组码的编码增益。对于某个特定的应用，采用分组编码还是采用卷积编码哪一种更好则取决于这一应用的具体情况和进行比较时可用的技术。

本次课设应用 SIMULINK 设计出基于卷积码的差错控制系统仿真模型，并通过 MATLAB 对系统进行性能仿真分析。模型包括信源部分、信道部分和信宿部分，信源部分的数据源是随机的二进制序列，随机的二进制序列要经过卷积编码，经过编码的数据要进行调制。信道部分对调制后的信号进行加噪，采用加性高斯白噪声。信宿部分完成信号的解调和译码（维特比译码）。通过此次课设便于了解各模块的功能，各参数的意义；并分析不同信噪比下维特比译码的误比特率关系曲线图，加深对特比译码原理和卷积码译码的理解。

## 2 通信系统设计

通信的目的是传输信息。通信系统的作用就是将信息从信源发送到一个或多个目的地。首先要把消息转变成电信号，然后经过发送设备，将信号送入信道，在接收端利用接收设备对接收信号作相应的处理后，送给信宿再转换为原来的消息，如图2.1所示。

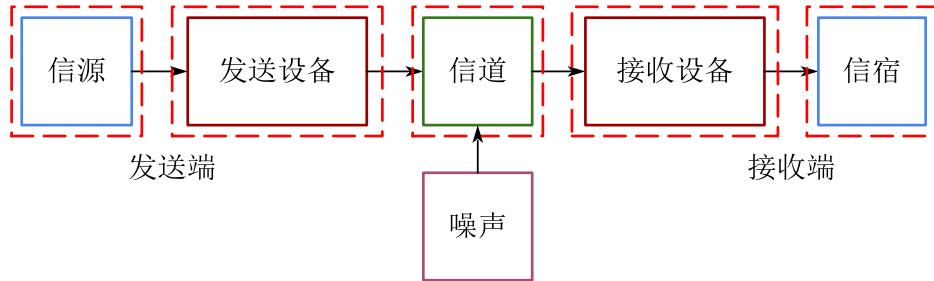


图 2.1 通信系统传输模型

### 2.1 信道编码

信道编码技术是指包括前向纠错、交织编码等的技术，它可使移动通信系统工作在低载干比和高噪声环境。利用数字信号处理技术可以很方便地实现信道自适应均衡、分集和跳频等功能。信道编码与数字信号处理技术将保证移动通信系统在多径和衰落信道条件下正常工作。因此，它们是移动环境下进行通信必不可少的技术。

由于移动通信存在干扰和衰落，在信号传输过程中将出现差错，故对数字信号必须采用纠、检错技术，即纠、检错编码技术，以增强数据在信道中传输时抵御各种干扰的能力，提高系统的可靠性。对要在信道中传送的数字信号进行的纠、检错编码就是信道编码。通常纠错码分为两大类，即分组码和卷积码。在移动通信系统中另一种纠错方法就是信令重发，解码时先存储再逐位判决，如重发五次，三次或三次以上均为1，则判1。信道编码之所以能够检出和校正接收比特流中的差错，是因为加入一些冗余比特，把几个比特上携带的信息扩散到更多的比特上。为此付出的代价是必须传送比该信息所需要的更多的比特。

考虑到移动通信多处于无线通信中，因此无线通信系统中的信道编码可表示为图2.2：

本次课设则是对二进制基带信号通过卷积码信道编码、BPSK 调制、维特比译码、BPSK 解调、误码率判决。

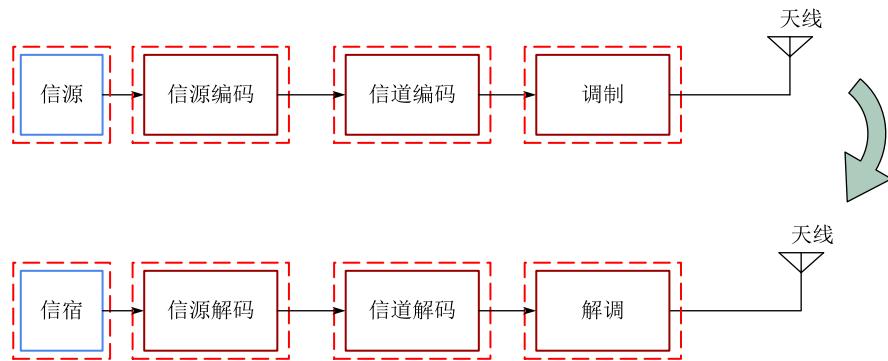


图 2.2 无线通信系统中的信道编码

## 2.2 卷积码

分组码是把  $k$  个信息比特的序列编成  $n$  个比特的码组，每个码组的  $n-k$  个校验位与本码组的  $k$  个信息位有关，而与其他码组无关。为了达到一定的纠错能力和编码效率，分组码的长度一般都比较大。编译码时必须把整个信息码组存储起来，由此产生的译码延时随  $n$  的增加而增加。

卷积码是一个有限记忆系统，如图2.3所示，它也将信息序列分割成长度  $k$  的一个个分组，然后将  $k$  个信息比特编成  $n$  个比特，但  $k$  和  $n$  通常很小，特别适合以串行形式进行传输，时延小。与分组码不同的是在某一分组编码时，不仅参看本时刻的分组而且参看以前的  $N-1$  个分组，编码过程中互相关联的码元个数为  $nN$ 。  $N$  称为约束长度。常把卷积码写成  $(n, k, N-1)$  卷积码。正因为卷积码在编码过程中，充分利用了各级之间的相关性，无论是从理论上还是实际上均已证明其性能要优于分组码。

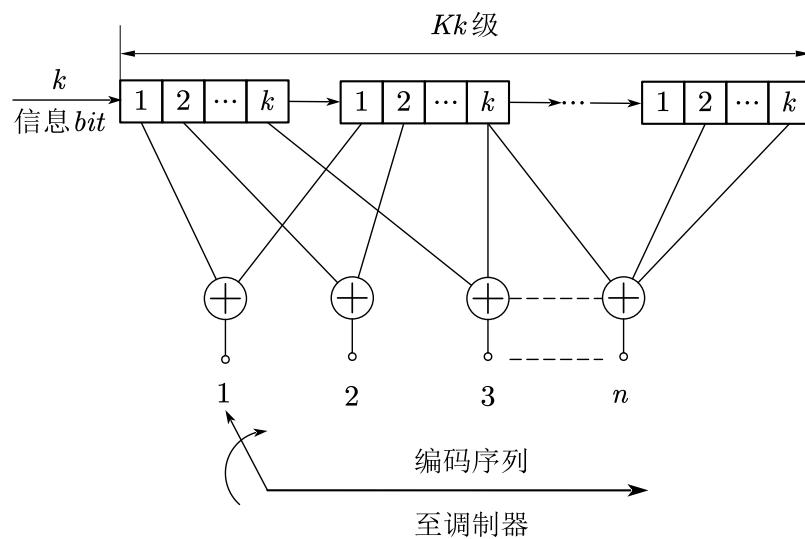


图 2.3 卷积码的结构图

## 2.2.1 卷积码的结构

卷积码是让发送的信息序列通过一个线性的、有限状态的移位寄存器而产生的码。通常，该移寄器由  $K$  级（每级  $k$  比特）和  $n$  个线性的代数函数发生器组成，如图下图所示。二进制数据移位输入到编码器，沿着移存器每次移动  $k$  比特位。每一个  $k$  比特长的输入序列对应一个  $n$  比特长的输出序列。因此其编码效率（码率）定义为  $R_c=k/n$ ，这和分组码编码效率的定义一致，参数  $K$  称为卷积码的约束长度。

描述卷积码的方法之一是给出它的生成矩阵，正如我们在处理分组码时的做法一样。一般来说，卷积码的生成矩阵是半（单边）无限矩阵，这是因为输入序列本身的长度是半无限的。另一种描述生成矩阵的方法是用一组  $n$  个矢量来表示，每个矢量对应  $n$  个模 2 加法器中的一个，这与生成矩阵在功能上是等效的。每个矢量有  $Kk$  维，包含了编码器和模 2 加法器之间连接关系的信息。某矢量第  $i$  个元素如果是“1”，表示相应的移存器第  $i$  级和模 2 加法器相连；反之，如果在该位置上为 0，则表明这一级移存器和模 2 加法器不相连。

具体地，一个约束长度  $K=3, k=1$  以及  $n=3$  的二进制卷积编码器，移位寄存器初始是全零状态。假设第 1 个输入比特是 1，那么 3 比特输出序列为 111。如果第 2 个输入比特是 0，则输出序列是 001。如果第 3 个输入比特是 1，输出序列将是 100，如此类推。现若将 3 比特输出序列从上到下按 1、2 和 3 编号，并给每个对应的函数生成矢量也作类似的编号，那么，由于只有第 1 级与第 1 个函数生成器相连（不需要模 2 加法器），因此第 1 个函数生成矢量是：

$$\mathbf{g}_1 = [1 \ 0 \ 0] \quad (2.1)$$

第二个函数生成矢量和第一级、第三级相连，所以：

$$\mathbf{g}_2 = [1 \ 0 \ 1] \quad (2.2)$$

最后，第三个函数生成矢量为：

$$\mathbf{g}_3 = [1 \ 1 \ 1] \quad (2.3)$$

这种码的生成矢量如用八进制方式表示为 (4, 5, 7) 则更为方便。可以得出这样一个结论：当  $k=1$  时，需要用  $n$  个生成矢量来表示，每个矢量是  $K$  维。

显然， $\mathbf{g}_1, \mathbf{g}_2$  和  $\mathbf{g}_3$  是从编码器输入到三个输出的脉冲响应。如果输入到编码器的是一个信息序列  $\mathbf{u}$ ，则三个输出是：

$$\begin{cases} \mathbf{c}^{(1)} = \mathbf{u} * \mathbf{g}_1 \\ \mathbf{c}^{(2)} = \mathbf{u} * \mathbf{g}_2 \\ \mathbf{c}^{(3)} = \mathbf{u} * \mathbf{g}_3 \end{cases} \quad (2.4)$$

这里的 \* 表示卷积运算。对应的编码序列  $\mathbf{c}$ ,  $\mathbf{c}^{(1)}$ ,  $\mathbf{c}^{(2)}$ ,  $\mathbf{c}^{(3)}$  的交织的结果:

$$\mathbf{c} = \left( \mathbf{c}_1^{(1)}, \mathbf{c}_1^{(2)}, \mathbf{c}_1^{(3)}, \mathbf{c}_2^{(1)}, \mathbf{c}_2^{(2)}, \mathbf{c}_2^{(3)}, \dots \right) \quad (2.5)$$

卷积运算相当于在变换域中作乘法。定义  $\mathbf{u}$  的 D 变换为:

$$\mathbf{u}(D) = \sum_{i=0}^{\infty} u_i D^i \quad (2.6)$$

三个脉冲响应的转移函数:  $\mathbf{g}_1$ ,  $\mathbf{g}_2$ ,  $\mathbf{g}_3$  是:

$$\begin{cases} \mathbf{g}_1(D) = 1 \\ \mathbf{g}_2(D) = 1 + D^2 \\ \mathbf{g}_3(D) = 1 + D + D^2 \end{cases} \quad (2.7)$$

则输出变换为:

$$\begin{cases} \mathbf{c}^{(1)}(D) = \mathbf{u}(D) \mathbf{g}_1(D) \\ \mathbf{c}^{(2)}(D) = \mathbf{u}(D) \mathbf{g}_2(D) \\ \mathbf{c}^{(3)}(D) = \mathbf{u}(D) \mathbf{g}_3(D) \end{cases} \quad (2.8)$$

转换成编码输出  $\mathbf{c}$  是:

$$\mathbf{c}(D) = \mathbf{c}^{(1)}(D^3) + D\mathbf{c}^{(2)}(D^3) + D^2\mathbf{c}^{(3)}(D^3) \quad (2.9)$$

## 2.2.2 卷积码的图解表示

卷积码的表示方法有图解表示法和解析表示法两种: 解析法, 它可以用数学公式直接表达, 包括离散卷积法、生成矩阵法、码生成多项式法; 图解表示法, 包括树状图、网络图和状态图三种。一般情况下, 解析表示法比较适合于描述编码过程, 而图形法比较适合于描述译码。一个编码效率为  $1/n$ , 约束度为  $K$  的卷积编码器的连接矢量和多项式分别定义为:

$$\mathbf{g}_i = (\mathbf{g}_{i1}, \mathbf{g}_{i2}, \mathbf{g}_{i3}, \dots, \mathbf{g}_{ij}, \dots, \mathbf{g}_{iK}) \quad (2.10)$$

$$\mathbf{g}_i(x) = \mathbf{g}_{i1} + \mathbf{g}_{i2}x + \mathbf{g}_{i3}x^2 + \dots + \mathbf{g}_{ij}x^{j-1} + \dots + \mathbf{g}_{iK}x^{K-1} \quad (2.11)$$

其中连接矢量  $\mathbf{g}_i$  定义为:

$$g_{ij} = \begin{cases} 0, & \text{第 } i \text{ 个模 } 2 \text{ 和加法器与编码寄存器第 } j \text{ 级不相连} \\ 1, & \text{第 } i \text{ 个模 } 2 \text{ 和加法器与编码寄存器第 } j \text{ 级相连时} \end{cases} \quad (2.12)$$

通过引入连接矢量使得卷积码的格子结构(后一节会提到)便于表达,同时连接矢量的存在使得卷积码具有脉冲响应,这也使得对于任意输入的序列对应的输出序列都可以用脉冲序列响应和输入序列做卷积,这也是卷积码的由来。

下面以(2,1,3)卷积码为例表示,设其连接矢量为:

$$\begin{cases} \mathbf{g}_1 = 1 + x + x^2 \\ \mathbf{g}_2 = 1 + x^2 \end{cases} \quad (2.13)$$

其编码器结构如图2.4:

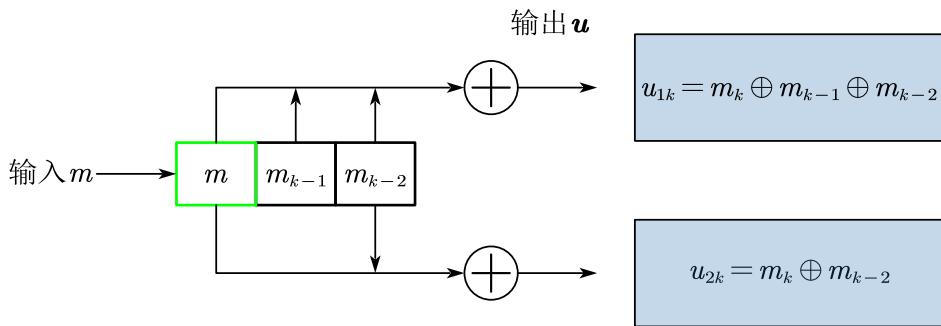


图 2.4 编码器结构图

在初始时刻,假设寄存器状态为(0,0,0)。在t1时刻,输入1,则寄存器状态变为(1,0,0),相当于把初始寄存器状态(0,0,0)中从左往右第三个0“挤”掉了。在t2时刻,输入0,则寄存器状态变为(0,1,0),相当于把t1时刻寄存器状态(1,0,0)中从做往右第三个0“挤”掉了。

在t3时刻,输入1,则寄存器状态变为(1,0,1),相当于把t2时刻寄存器状态(0,1,0)中从做往右第三个0“挤”掉了。接着需要两个冲洗比特(连续输入两个0),用以清空寄存器。在t4时刻,输入0,则寄存器状态从t3时刻的(1,0,1)变为(0,1,0),相当于把t3时刻寄存器状态(1,0,1)中从做往右第三个1“挤”掉了。在t5时刻,输入0,则寄存器状态从t4时刻的(0,1,0)变为(0,0,1),相当于把t4时刻寄存器状态(0,1,0)中从做往右第三个0“挤”掉了。

其实,输入两个就达到了清空寄存器的目的了,试想一下,如果在t6时刻输入一个1,这时候寄存器状态就从(0,0,1)变为(1,0,0),t5时刻的寄存器状态中的1被“挤”掉了,也就是这个1被舍弃掉了。换言之,这个1是没用的,所以两个冲洗比特足矣,只需要把寄存器状态变为(0,0,X)的形式即可,其中,X可为0或1。不管X为0还是1,都对下一个输入进来的時候,X都会被舍弃掉。卷积码过程图解如图2.5:

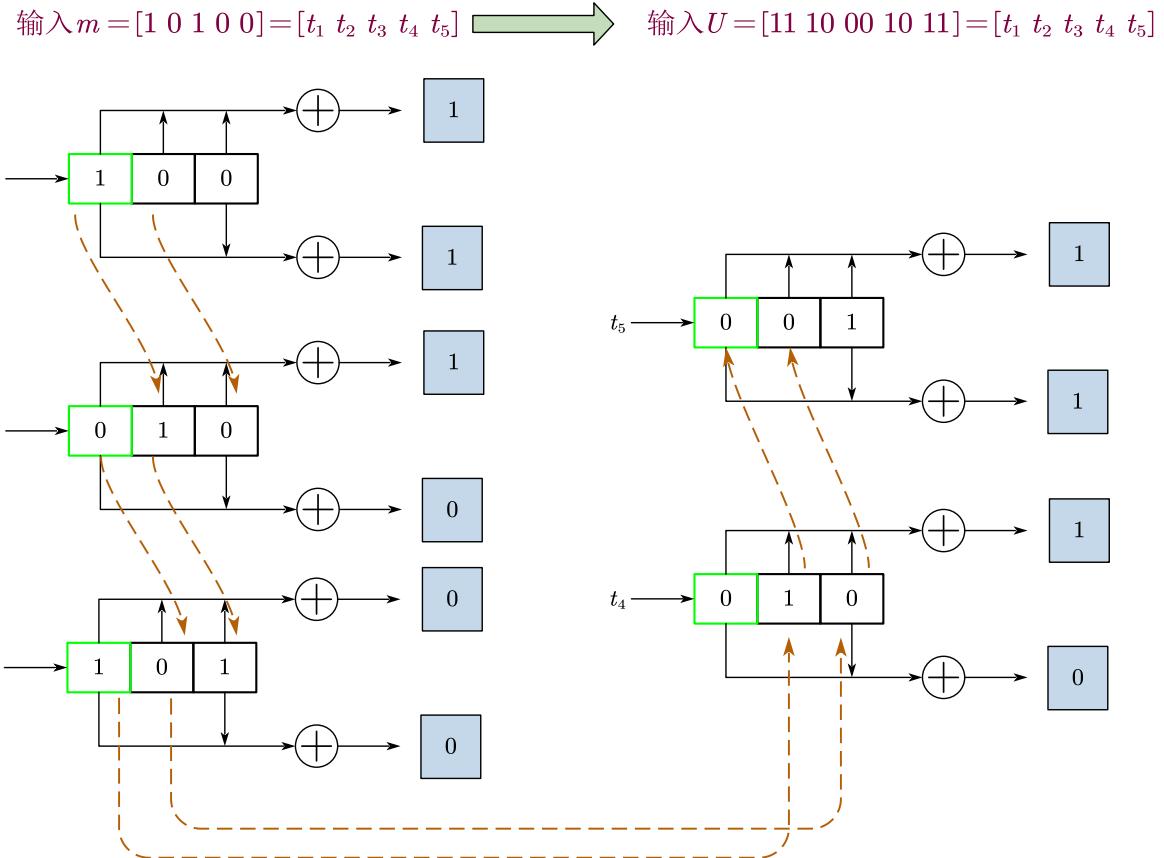


图 2.5 卷积码过程图解

### 状态图表示法:

状态图 (Statechart Diagram) 是描述一个实体基于事件反应的动态行为，显示了该实体如何根据当前所处的状态对不同的事件做出反应。通常我们创建一个 UML 状态图是为了以下的研究目的：研究类、角色、子系统、或组件的复杂行为。用于显示状态机（它指定对象所在的状态序列）、使对象达到这些状态的事件和条件、以及达到这些状态时所发生的操作。

卷积编码是有限状态机器件。只有有限个状态机制，状态提供了有关过去序列过程以及一组将来可能输出序列的限制，即下一状态总是受到前一状态的限制。状态图中，实线表示输入为 0 的状态转变，虚线表示输入为 1 的状态转变，线上的数字表示当由此状态转换成下一状态时编码器的输出。例如， $a$  到  $b$  的虚线表示：当输入为 1 时，由  $a$  状态 00 变换成  $b$  状态 10，编码输出 11；如果输入为 0，则由  $a$  状态 00 变换成  $a$  状态 00，编码输出 00。卷积码状态图如图2.6所示：

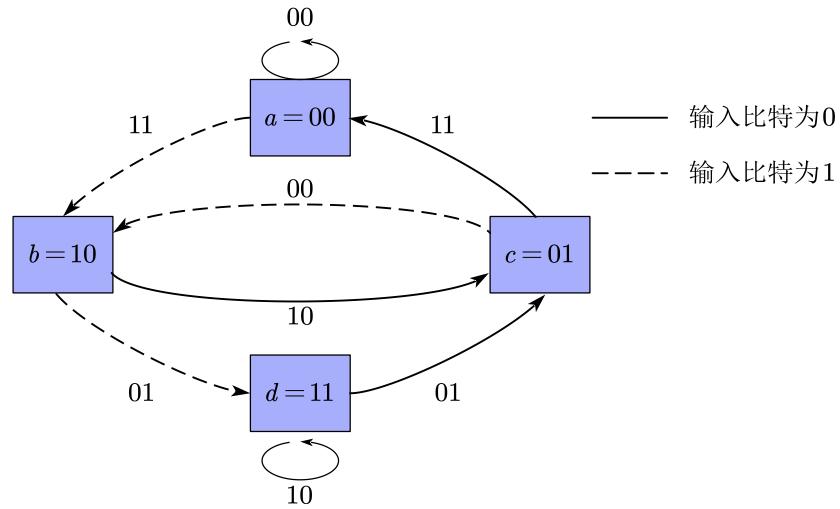


图 2.6 卷积码状态图

状态图完全描述了编码器的特性，但没有表示时间过程。树图在状态图的基础上增加了时间尺度，如图2.7。寄存器状态： $a=00$   $b=10$   $c=01$   $d=11$ ：

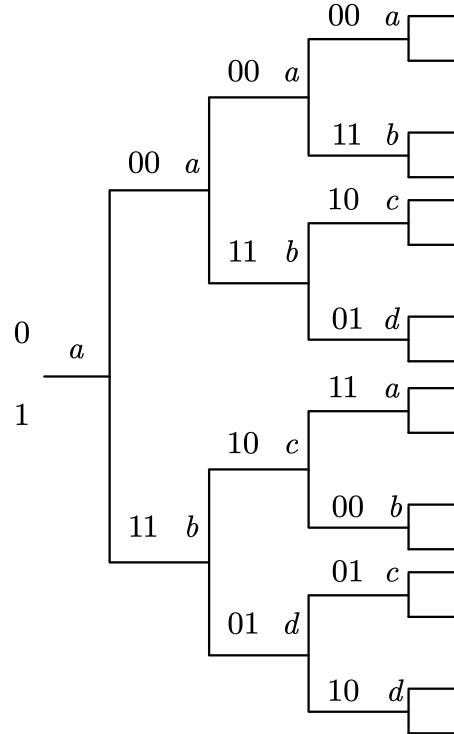


图 2.7 卷积码树图

如图2.8所示。该图设接收到的序列长度为 8，所以画 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  $2^L$  条。这里节点  $a=00$ ,  $b=10$ ,  $c=01$ ,  $d=11$ 。

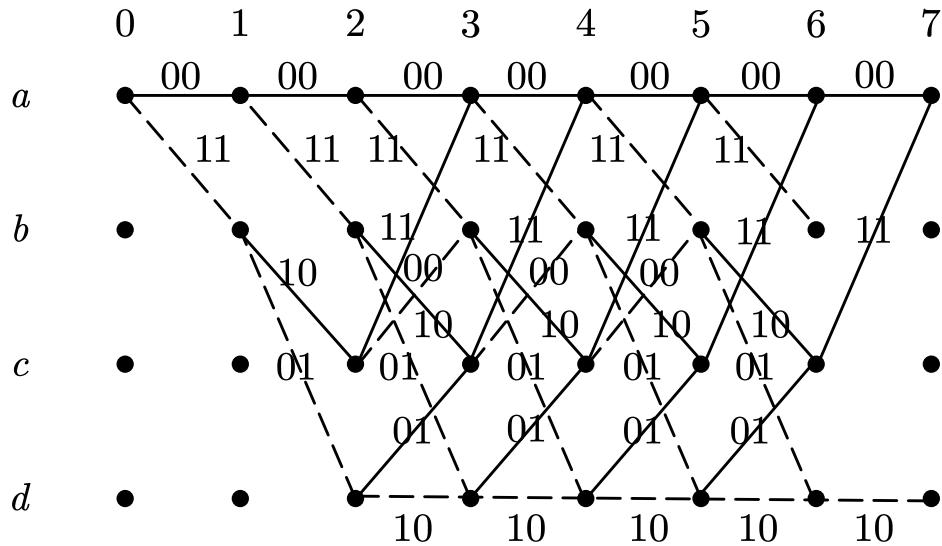


图 2.8 卷积码格图

### 2.2.3 维特比译码

维特比译码是采用最大似然法，对回溯长度的所有前一组幸存路径进行比较，取出距离较小的路径作为下一组幸存路径，码长采用概率译码的基本思想是：把已接收序列与所有可能的发送序列做比较，选择其中码距最小的一个序列作为发送序列。如果发送  $L$  组信息比特，那么对于  $(n, k)$  卷积码来说，可能发送的序列有  $2^k L$  个，计算机或译码器需存储这些序列并进行比较，以找到码距最小的那个序列。当传信率和信息组数  $L$  较大时，使得译码器难以实现。维特比算法则对上述概率译码做了简化，以至成为了一种实用化的概率算法。它并不是在网格图上一次比较所有可能的  $2^k L$  条路径(序列)，而是接收一段，计算和比较一段，选择一段最大似然可能的码段，从而达到整个码序列是一个最大似然值得序列。

设输入编码器的信息序列为  $(11011000)$ ，则输出的序列为  $Y=(1101010001011100)$ 。若收到的序列  $R=(0101011001011100)$ ，对照网格图来说明维特比译码的方法。

首先选择接收序列的前 6 位序列  $R_1=(010101)$  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  $R_1$  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  $R_1$  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  $R_1$  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)。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。幸存路径格子分析图如图2.9所示:

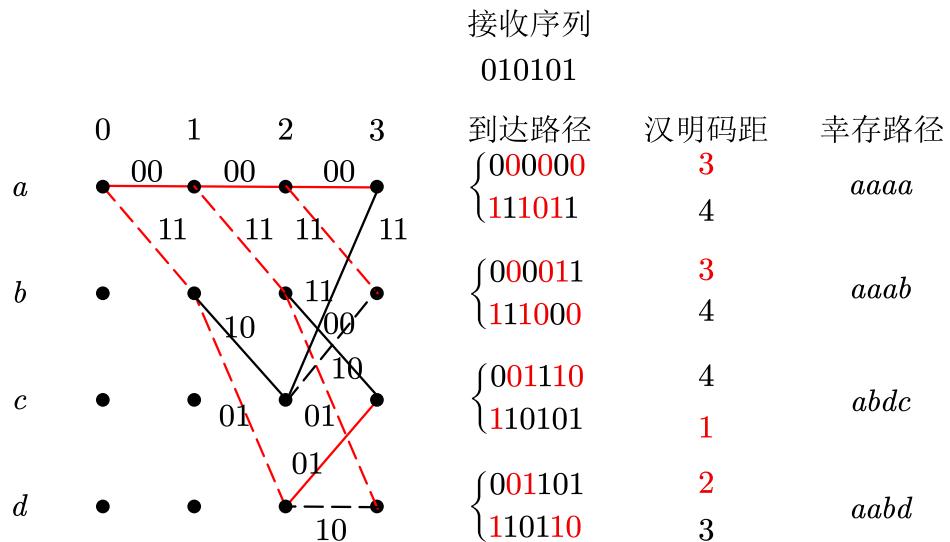


图 2.9 幸存路径分析图

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,此时不会影响最终的译码结果。在码的终止A时刻a状态,得到一条幸存路径。如图2.10所示。由此可看到译码器。

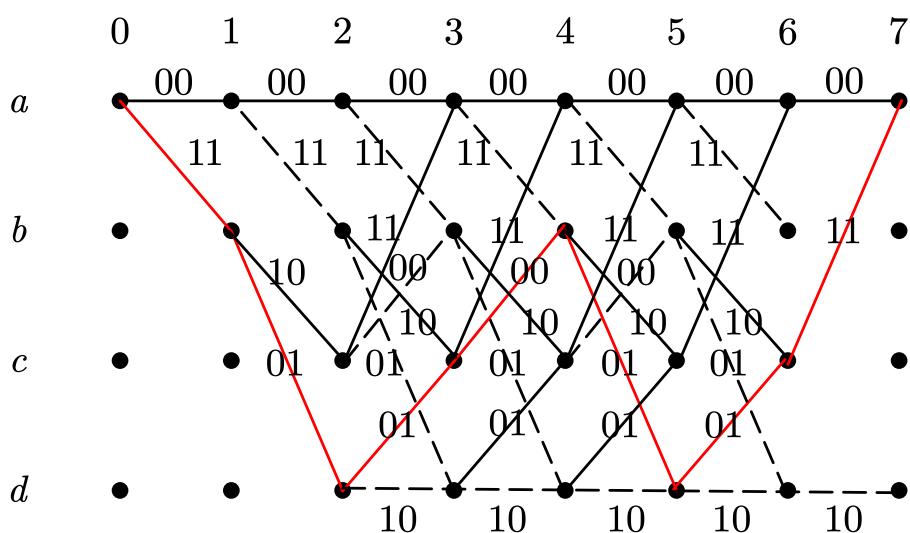


图 2.10 终止时刻幸存路径

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。初始时, 状态 00 代价为 0, 其它  $2^{(k-1)} - 1$  个状态代价为正无穷 ( $\infty$ )。

算法的主循环由两个主要步骤组成: 首先计算下一时刻监督比特序列的分支度量, 然后计算该时刻各状态的路径度量。路径度量的计算可以被认为是一个“加比选”过程:

- 将分支度量与上一时刻状态的路径度量相加。
- 每一状态比较达到该状态的所有路径 (每时刻每个状态只有两条这样的路径进行比较, 因为只有两条来自前一时刻状态的分支)。
- 每一状态删除其余到达路径, 保留最小度量的路径 (称为幸存路径), 该路径对应于错误最少的路径。

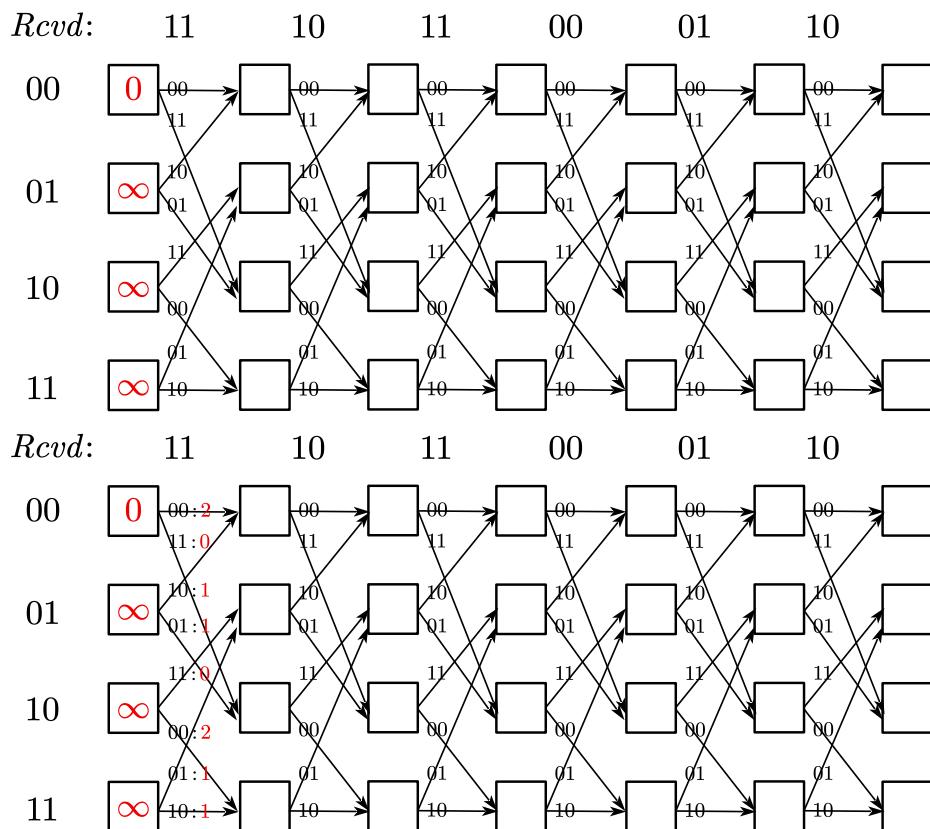


图 2.11 维特比译码幸存路径

图2.11显示了维特比译码器从一个时刻到下一个时刻的译码过程。这个例子显示接收到的位序列为 11 10 11 00 01 10, 以及接收器如何处理它。第四部分显示了具有相同

路径度量四个不同状态，在这个阶段，通向这四个状态的任一路径都是最可能发送的比特序列（它们都具有度量为 2 的汉明距离）。最下面的部分只显示幸存路径的情况，幸存路径是有可能成为最大似然路径的路径。还有很多其他路径可以被删除，因为它们不可能有状态回溯回来。维特比译码器之所以能被实际应用，是因为幸存路径的数量远远小于网格中所有可能路径的总数。

#### 2.2.4 软判决

Viterbi 译码分硬判决和软判决两种，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（BSC）。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（DMC）的译码方法。

软判决译码（有时也称为“软输入维特比译码”）建立在这一观察上，它在译码之前不会数字化输入样本。相反，它使用具备连续性的模拟样本（经采样量化）作为译码器的输入。例如，如果期望的发射监督位为 0 并且接收到的电压为 0.3V，那么我们可以使用 0.3（或 0.32 或某些此类函数）作为该位的值，而不是数字化它。

#### 2.2.5 硬判决

对于数字电路，硬判决的实现是通过截取解量化信号的符号位，可以认为是一级量化，而软判决可认为多级量化，包括高位符号位在内，还含有信道信息的有效位。软判决避免了解调后误判影响，直接送入译码器进行译码处理。一般而言，硬判决译码较软判决译码简单而易于实现，但判决译码由于充分利用了信道输出信号的信息，在性能上要增加 2~3dB。目前，通用的量化电平为 8 电平（3bit 量化）和 16 电平（4bit 量化），再高的话，只能增加译码器复杂度，几乎没有性能的提高。总的来说，软判决是用欧式距离做，硬判决用汉明距离。

硬判决译码通过将接收到的电压信号与阈值进行比较，数字化后将其传递给译码器进行译码。结果，我们失去了部分信息：电压为 0.500001 时数字化结果的可信度肯定比电压为 0.999999 时低得多，但两者都被视为“1”，且译码器现在以相同的方式对待它

们，尽管 0.999999 比其他值更可能是“1”。

### 3 Simulink 单元模块设计

#### 3.1 整体设计流程

纯 matlab 进行软决策、硬决策、无编码三种情况下的 matlab 算法流程图如3.1所示：

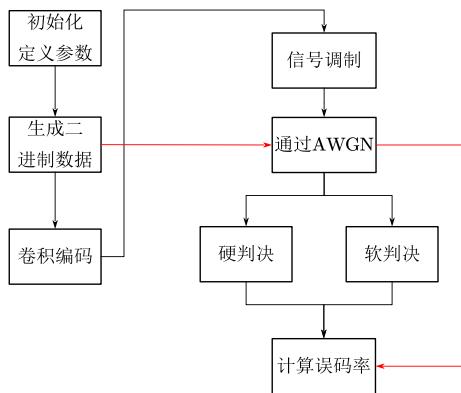


图 3.1 整体设计流程

#### 3.2 卷积码的差错控制系统仿真模型

##### 3.2.1 总体设计框图

卷积码的差错控制系统由信源子模块，信道和信宿子模块组成，信源子模块发出随机二进制信号，经过卷积编码后经过加性高斯白噪声信道，传到信宿模块，信宿模块完成解码并计算误码率的功能。总体设计框图如下3.2所示：软判决如图3.2：

硬判决如图3.3：

##### 3.2.2 信源子系统

信源模块由伯努利二进制序列发生器、卷积码编码器以及二进制相位调制模块组成，伯努利二进制序列发生器产生的随机二进制序列经过卷积编码器编码以及二进制相位调制后送入信道，另外将未经过调制的信号直接送入信宿。电路图如图3.4所示。模块各部分参数设置如下图：

(1)Bernoulli Binary Generator 伯努利发生器的参数设置：

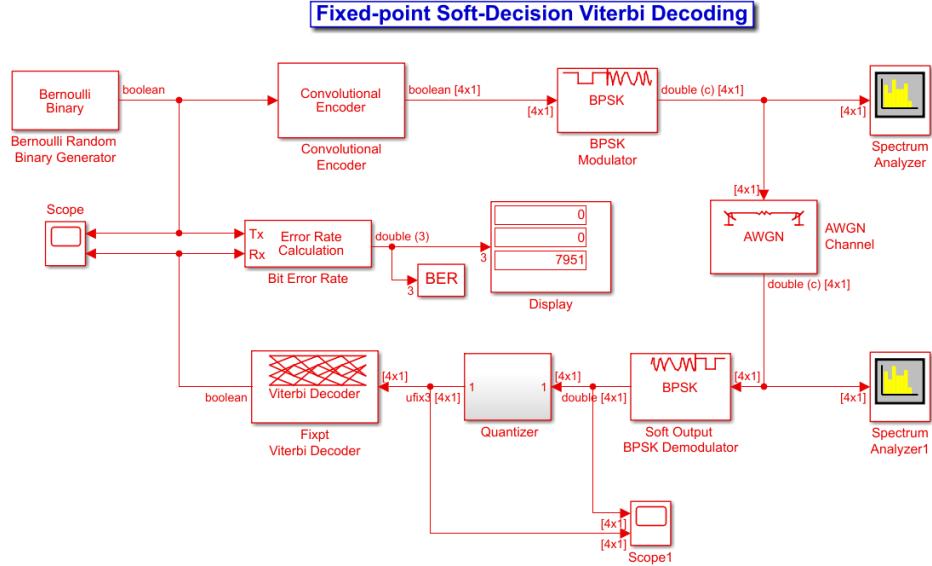


图 3.2 软判决总体设计框图

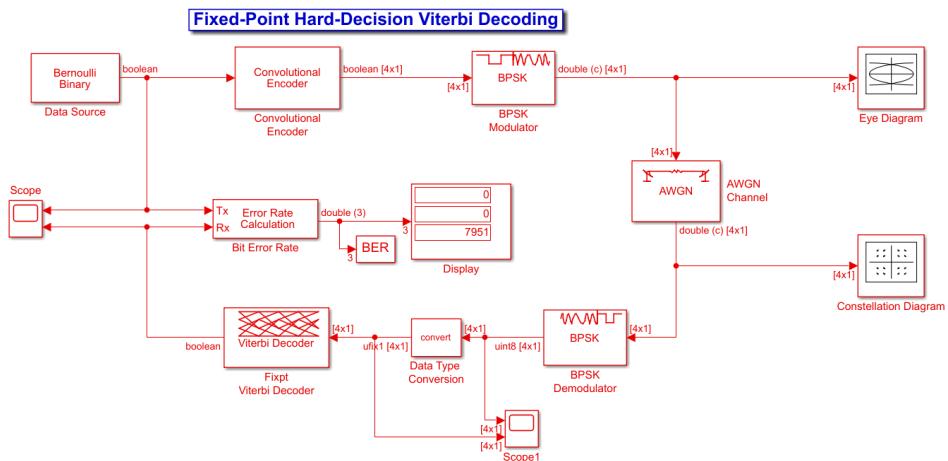


图 3.3 硬判决总体设计框图

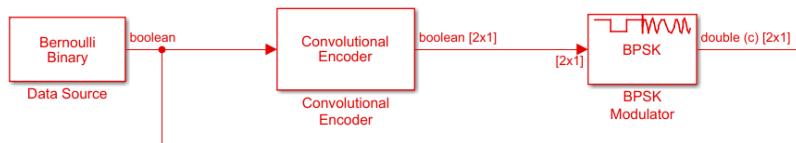


图 3.4 信源子系统图

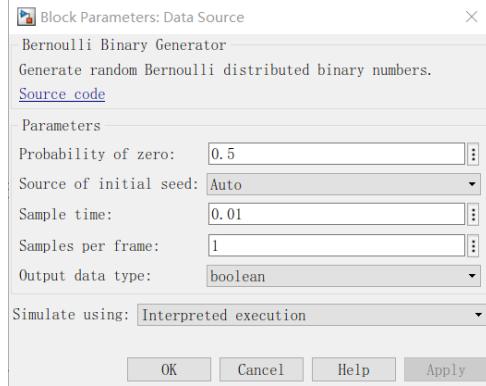


图 3.5 伯努利序列图

(2)Convolutional Encoder 卷积码编码器的参数设置:

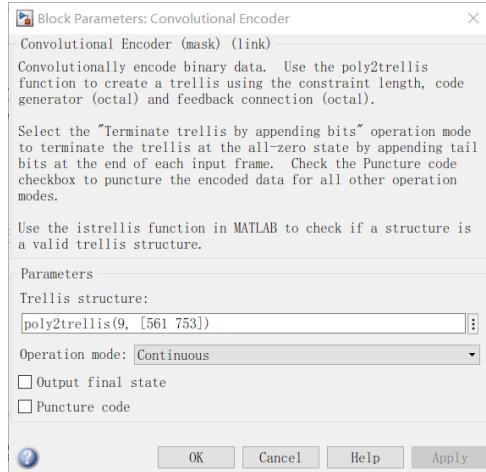


图 3.6 卷积码编码器图

(3)BPSK Modulator Baseband (二进制相位调制模块) 的参数设置:

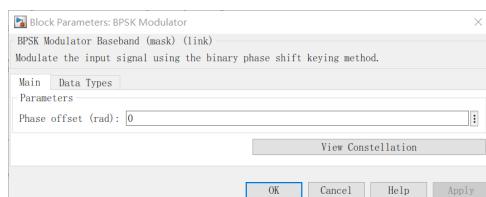


图 3.7 BPSK 相位调制模块

### 3.2.3 信道

信道是加性高斯白噪声信道，用于对传输的信号添加加性高斯白噪声。信道设计，参数设置如下图 AWGN Channel 加性高斯白噪声模块：

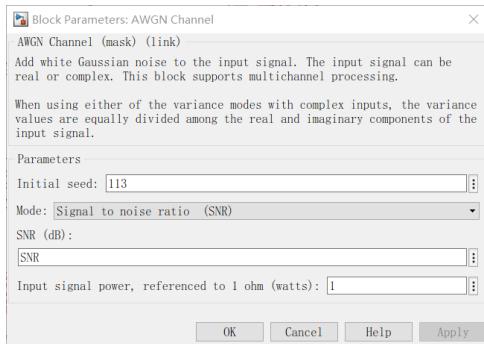


图 3.8 AWGN 信道模块

### 3.2.4 信宿子系统

信宿模块由二进制相位解调模块、维特比译码、误比特率统计模块器、数值显示模块、选择器组成。在接收到二进制相位调制信号后，首先由 BPSK Demodulator Baseband(二进制相位解调模块)对信号进行量化，得到硬判决量化信号，然后通过 Viterbi Decoder(维特比译码器)对软判决信号译码。译码输出信号和信源模块产生的原始信号输入到 Error Rate Calculator(误比特率统计模块)中，统计得到的数据一方面通过 Display(显示模块)显示出来，另一方面通过一个 Selector(选择器)把其中的第一个元素(编码信号的误比特率)保存到 BER 中。模块各部分参数设置如下图：

#### (1)DPSK Demodulator Baseband (二进制相位解调器)

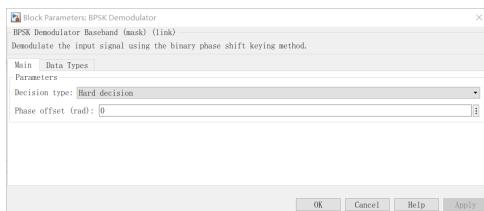


图 3.9 二进制相位解调器模块

#### (2)Viterbi Decoder(维特比译码器)

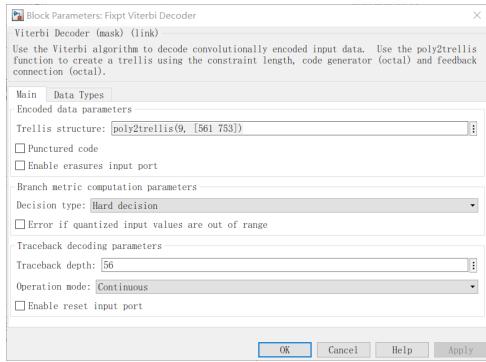


图 3.10 维特比译码器模块

### 3.3 量化判决

Viterbi 译码分硬判决和软判决两种，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（BSC）。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”。由此得到不同的判决方式：

软判决：

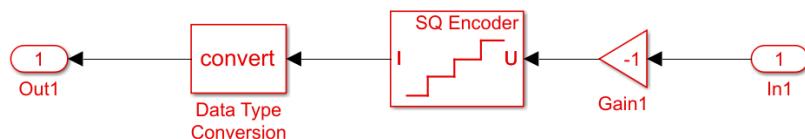


图 3.11 软判决子系统

其信息转化图为：

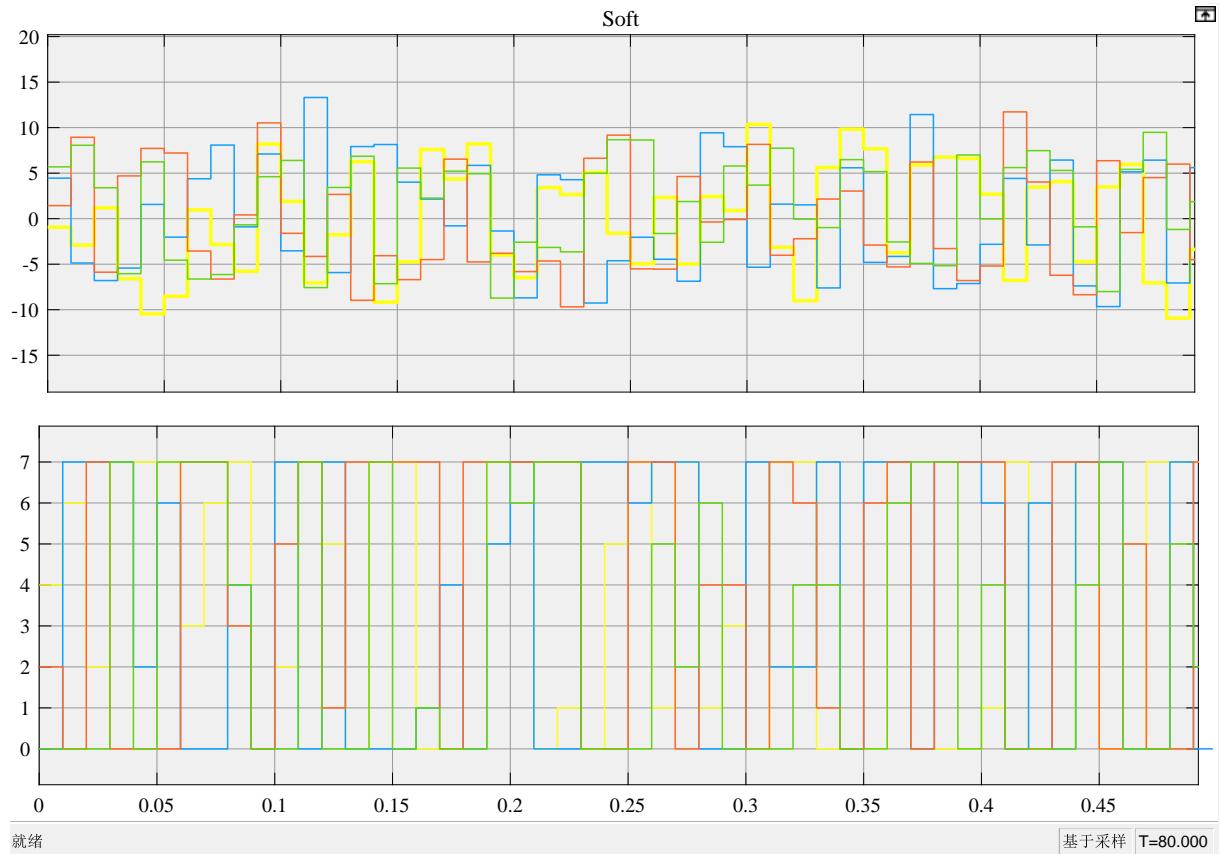


图 3.12 软判决子系统

从 tu3.12可以看出，软决策其实就是将解调出的模拟量进行多电平量化的数字量，从而保留“软信息”，这从理论上也减低了误码的概率。

**硬判决：**

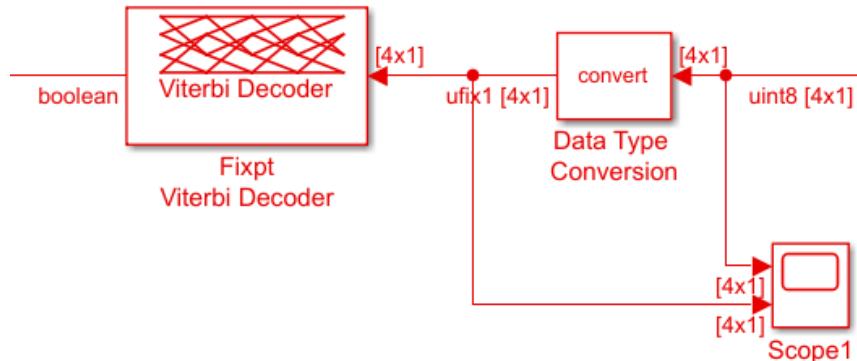


图 3.13 硬判决模块

其信息转化图为：

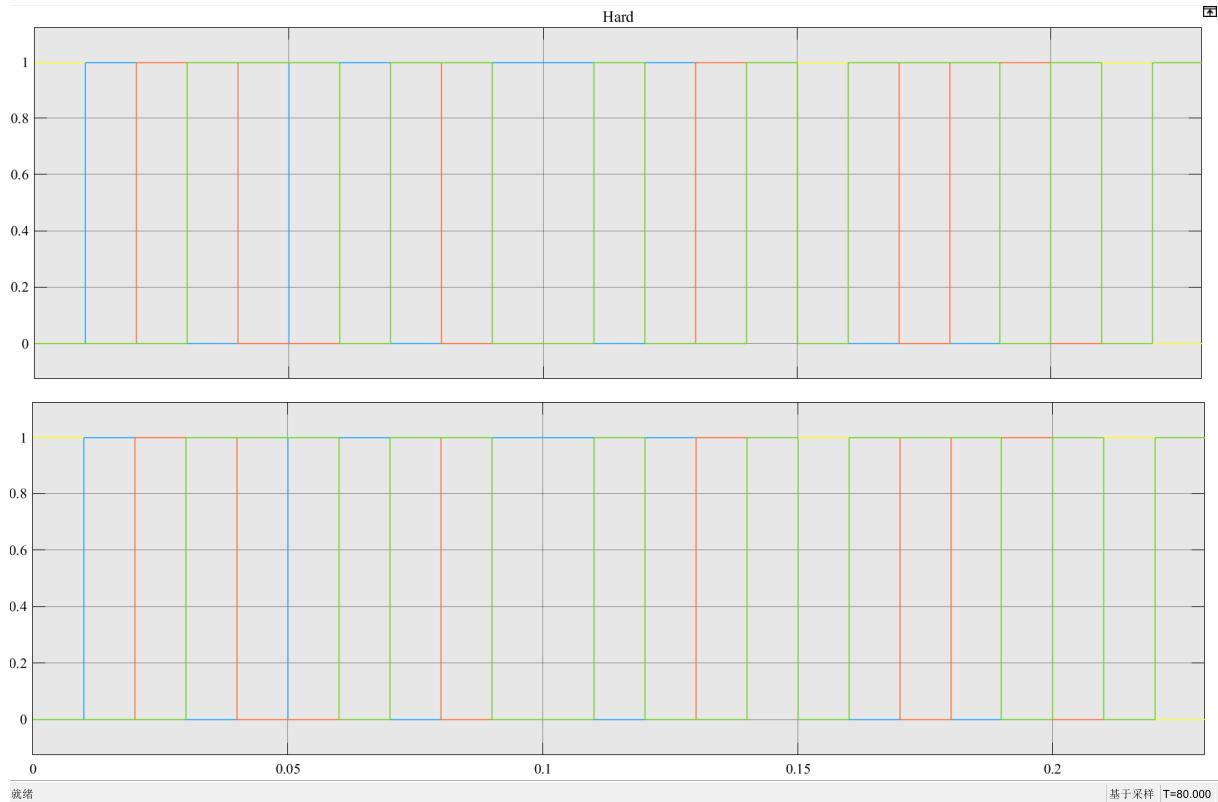


图 3.14 硬判决子系统

从 tu3.14可以看出，硬决策其实就是将解调出的信号进行 0-1 电平的量化，未能保留“软信息”，这从理论上也减低了误码的概率。电压为 0.500001 时数字化结果的可信度肯定比电压为 0.999999 时低得多，但两者都被视为“1”，且译码器现在以相同的方式对待它们，尽管 0.999999 比其他值更可能是“1”。从而理论上从性能上应若干于软判决。

### 3.4 matlab 和 simulink 交互式仿真

虽然 simulink 属于 GUI 式设计，方便简单仿真，但是由于本次课设需要研究卷积速率（或编码效率）、回溯深度、软判决硬判决等影响时，我们需要对 simulink 进行多次调参并且仿真设计，为了实现自动化过程，需要 matlab 命令窗口进行调参对比，方法是通过将关键参数如 AWGN 的信道信噪比、卷积码的格子结构形式等要调参的参数进行 baseplace，和 matlab 的命令窗口变量关联起来进而写 m 脚本。对输出误码率采用 to workplace 模块导出到 matlab 进行记录、保存与绘图。

## 4 卷积编码实现与仿真

### 4.1 BPSK 卷积编码硬判决仿真

BPSK 卷积编码硬判决仿真的 simulink 的总体模块设计模型图如下：

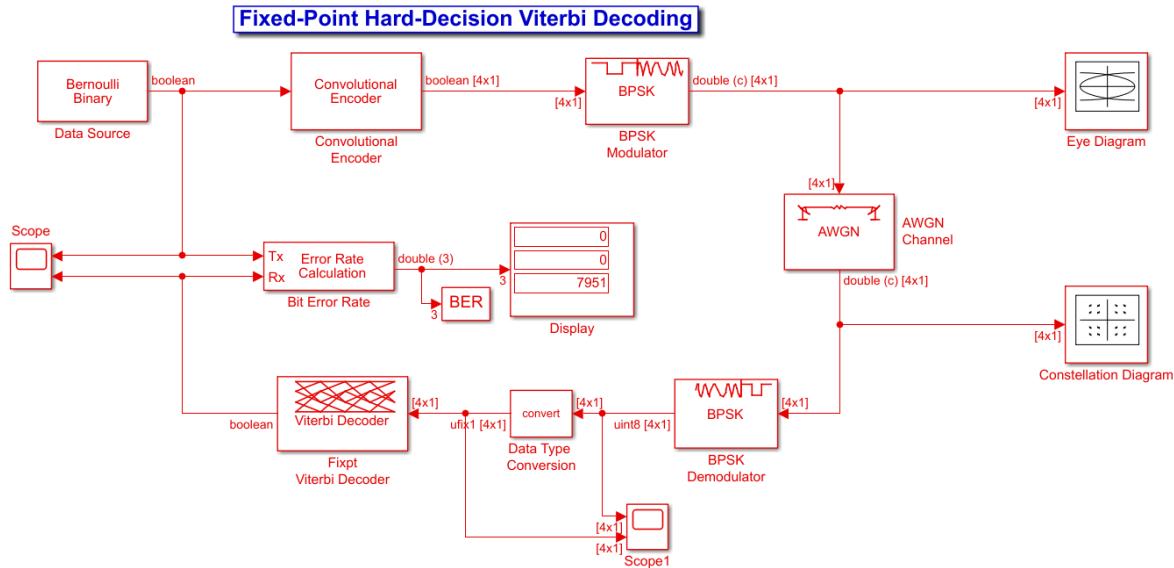


图 4.1 维特比硬判决总体模型设计图

#### 4.1.1 卷积速率对误码率的影响

通过改变卷积码的格子结构来控制寄存器的数目 K 和输出序列 N，从而改变编码率（或编码效率），重点函数为 `poly2trellis`，用作卷积码多项式到网格描述的转换，根据需求设置不同的多项式来控制卷积码的格子结构，从而控制卷积码的码率。

`poly2trellis` (`ConstraintLength`,`CodeGenerator`) 返回与速率  $K/N$  馈送编码器的转换相对应的格子结构描述。k 是对编码器的输入比特流的数量，n 是输出连接的数量。`ConstraintLength` 指定对编码器的输入比特流的延迟。`CodeGenerator` 指定对编码器的输入比特流的输出连接。

如图4.2为码率 2/3 前馈卷积码创建一个格子结构，并显示格子的下一个状态的一部分。该图显示了具有两个输入流，三个输出流和七个移位寄存器的速率 2/3 编码器。其代码使用规则为： `trellis = poly2trellis([5 4],[23 35 0; 0 5 13])`

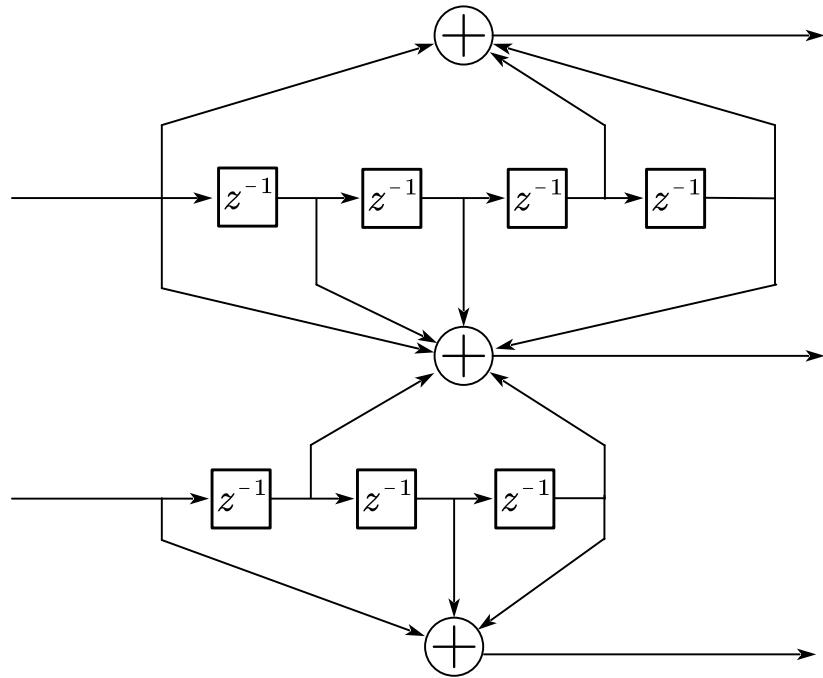


图 4.2 码率 2/3 前馈卷积码格子结构图

本次课设我采用 1/4、1/3、1/2 三种码率进行仿真得到维特比译码的硬判决如下：

```

1 clear y
2 x=-8:2;%x表示信噪比
3 %卷积方式分别取1/4、1/3卷积和1/2卷积
4 A=[poly2trellis(9, [557 663 711]),poly2trellis(7, [171 133]),poly2trellis(11, [3237 1734 416 266])];
5 %不同卷积方式、信噪比情况下重复运行c, 检验不同情况下硬判决译码的性能
6 for j=1:3
7     STRUCTURE=A(j);
8     %信道的信噪比依次取x中的元素
9     for i=1:length(x)
10        %信道的信噪比依次取x中的元素
11        SNR=x(i);
12        %运行仿真程序, 得到的误比特率保存在工作区变量BER中
13        %    sim('cm_viterbi_harddec_fixpt');
14        sim('cm_viterbi_softdec_fixpt');
15        %计算BitErrorRate的均值, 作为本次仿真的误比特率
16        y(j,i)=BER(1);
17    end
18 end

```

最终得到不同卷积速率或编码效率随信噪比的变化，其误码率的变化如图4.3：

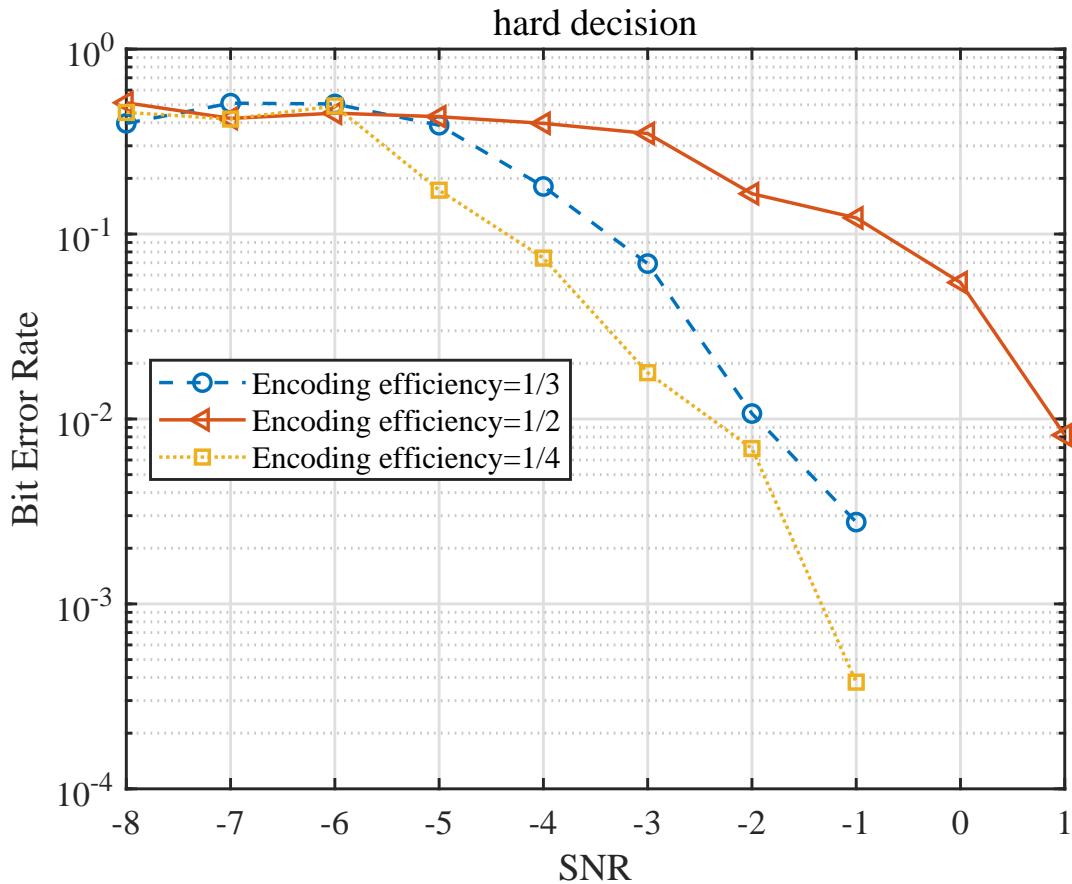


图 4.3 不同卷积速率下的误码率曲线

首先误码率总体随着信噪比的增大而减小，这是因为信噪比越大，其中噪声对信息判决带来的干扰也会越小，从而误比特率也会越低，误比特率曲线符合理论实际，另外可以看出维特比译码的误码率随设置的码率的增加而降低，这一结果是显然的，因为编码效率和纠错能力本身是一种矛盾，因此仿真符合理论实际。

#### 4.1.2 回溯深度对误码率的影响

由于维特比算法采用的是最大似然法，通过回溯的方式寻找出各组到达路径的相对最优路径，从而保留它们为幸存路径。因此回溯深度对误码率也有一定影响。Simulink 中设置回溯深度可有 MATLAB 控制，将 `traceback depth` 设置为变量 `TRACEDEPTH`，通过设置参数 `TRACEDEPTH=16、36、56、76` 进行仿真，得到误码率随信噪比的曲线如图4.4：

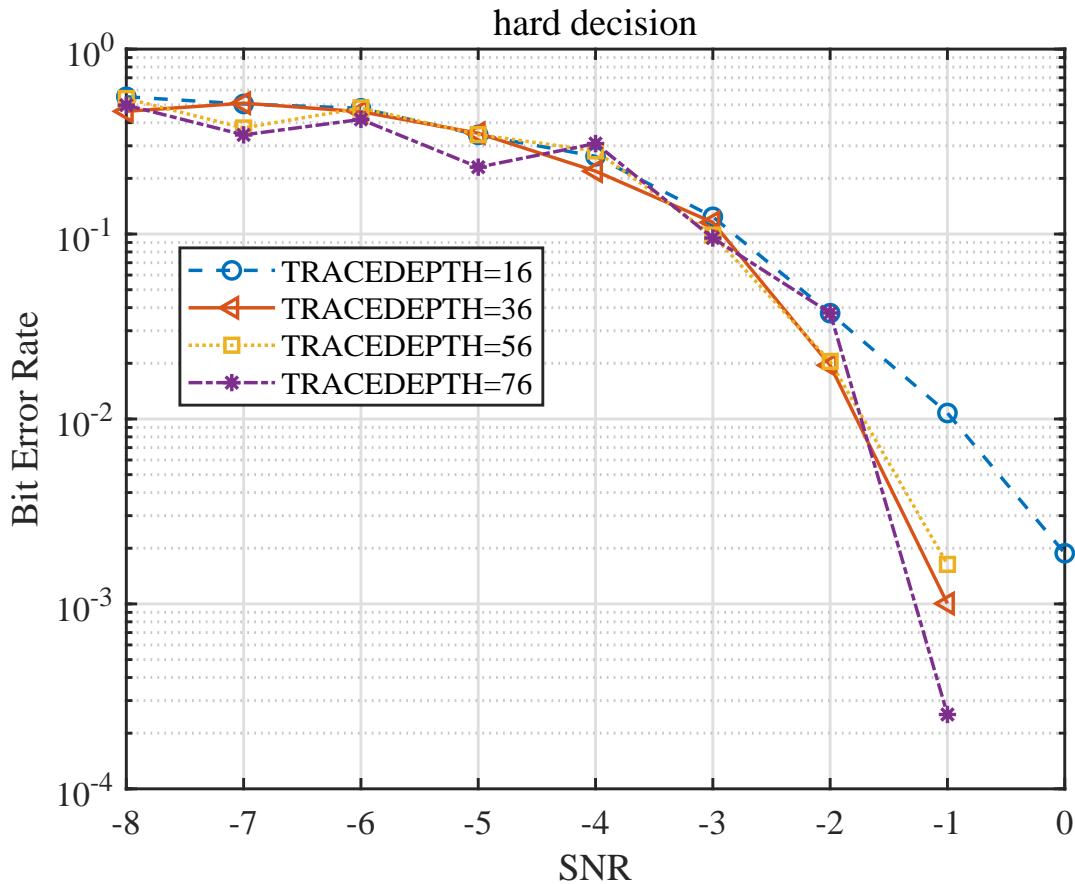


图 4.4 不同回溯深度下的误码率曲线

从图4.4中可以看出，当回溯深度较小时，更易误码，当回溯深度大至一定程度时，回溯深度对误码率的影响不大，其实主要的误码还是源于信噪比太小，导致有用信号微弱，即使牺牲编码效率来提高卷积码的纠错能力，也难以弥补过大比例噪声对判决带来的错误影响，收到的噪声影响大于回溯深度对误码率的影响。

## 4.2 BPSK 卷积编码软判决仿真

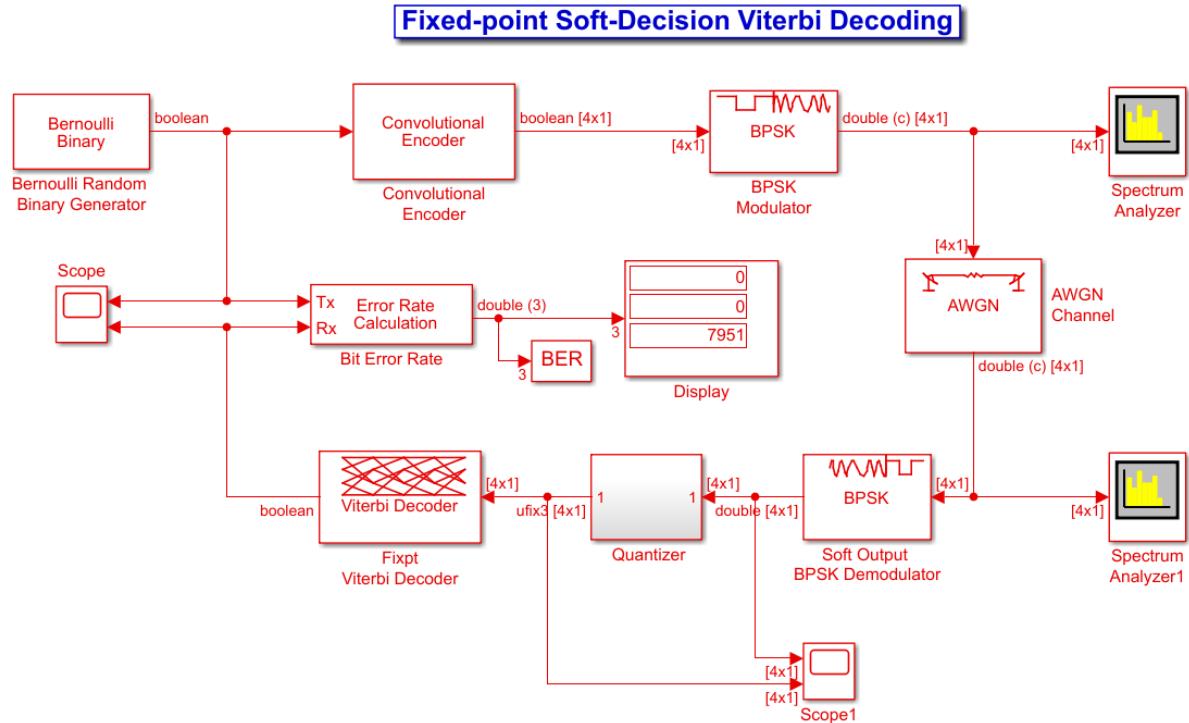


图 4.5 维特比软判决总体模型设计图

### 4.2.1 卷积速率对误码率的影响

同上节可知，采用系统的格子结构图设置方式。`poly2trellis`的设置代码如下：

```

1 clear y
2 x=-8:2;%x表示信噪比
3 %卷积方式分别取1/4、1/3卷积和1/2卷积
4 A=[poly2trellis(9, [557 663 711]),poly2trellis(7, [171 133]),poly2trellis(11, [3237 1734 416 266])];
5 STRUCTURE=A(1);
6 B=[16 36 56 76];
7 %不同卷积方式、信噪比情况下重复运行c, 检验不同情况下硬判决译码的性能
8 for j=1:length(B)
9     TRACEDEPTH=B(j);
10    %信道的信噪比依次取x中的元素
11    for i=1:length(x)
12        %信道的信噪比依次取x中的元素
13        SNR=x(i);
14        %运行仿真程序, 得到的误比特率保存在工作区变量中
15        sim('cm_viterbi_harddec_fixpt');
16        % sim('cm_viterbi_softdec_fixpt');
17        %计算BitErrorRate的均值, 作为本次仿真的误比特率
18        y(j,i)=BER(1);
19    end
20 end

```

最终得到不同卷积速率或编码效率随信噪比的变化，其误码率的变化如图4.6：

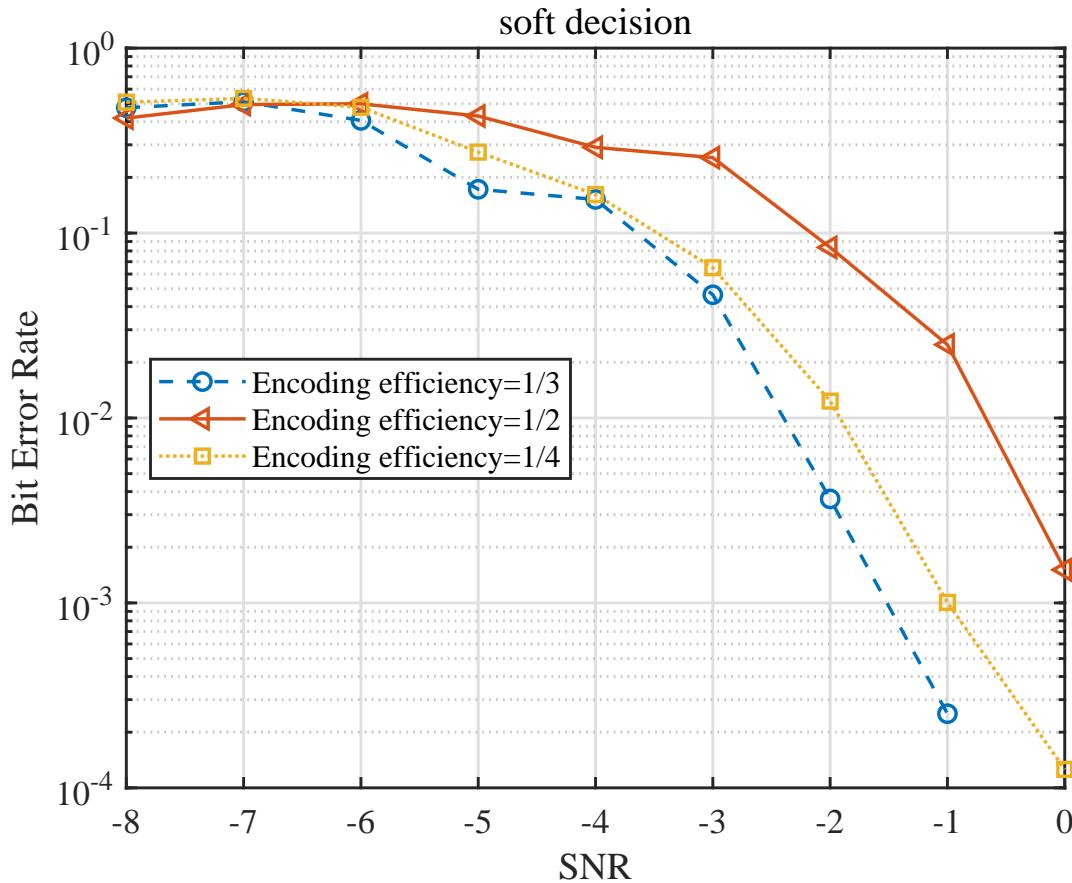


图 4.6 不同卷积速率下的误码率曲线

从图4.6中可以看出，当回溯深度较小时，更易误码，当回溯深度大至一定程度时，回溯深度对误码率的影响不大，其实主要的误码还是源于信噪比太小，导致有用信号微弱，即使牺牲编码效率来提高卷积码的纠错能力，也难以弥补过大比例噪声对判决带来的错误影响，收到的噪声影响大于回溯深度对误码率的影响。规律和硬判决一直，符合理论实际。

#### 4.2.2 回溯深度对误码率的影响

同上一节，维特比软判决下回溯深度对误码率的影响图如图4.7：

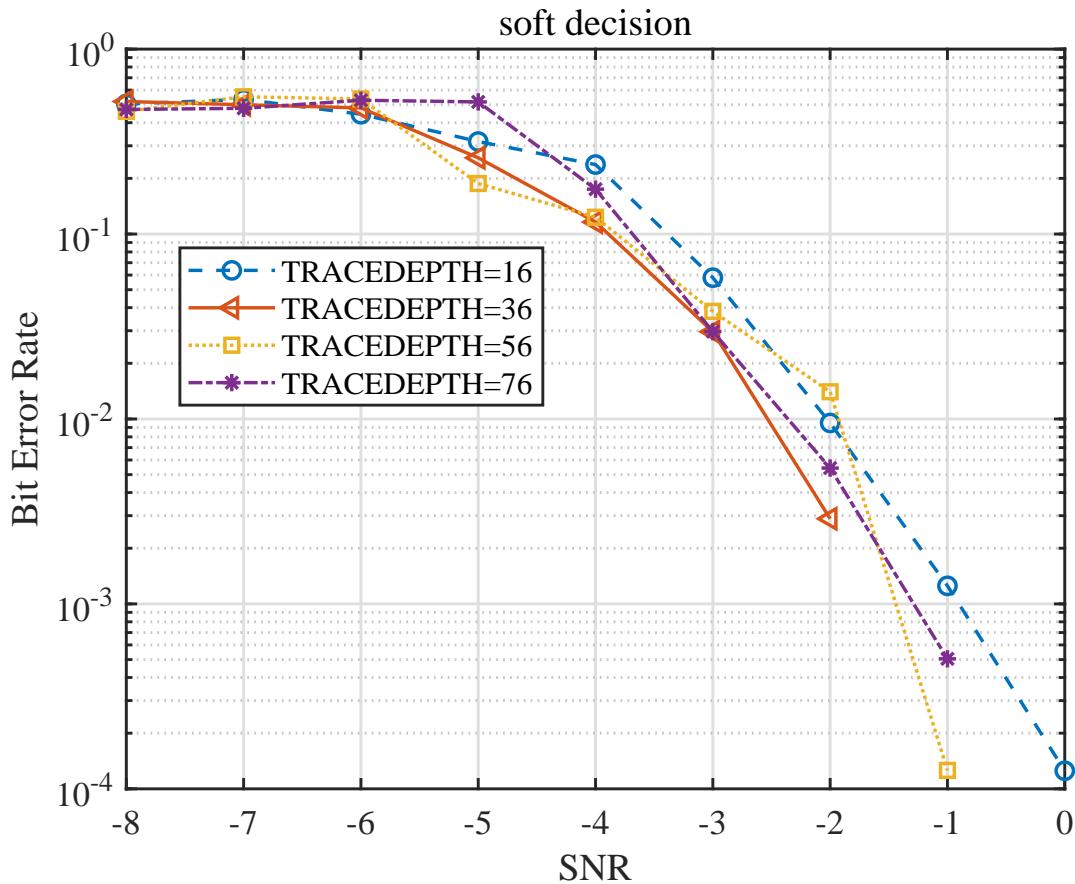


图 4.7 不同回溯深度下的误码率曲线

从图4.7中可以看出，当回溯深度较小时，更易误码，当回溯深度大至一定程度时，回溯深度对误码率的影响不大，其实主要的误码还是源于信噪比太小，导致有用信号微弱，即使牺牲编码效率来提高卷积码的纠错能力，也难以弥补过大比例噪声对判决带来的错误影响，收到的噪声影响大于回溯深度对误码率的影响。

同时维特比译码回溯深度必须和误码计算仪的接收延时保持一致，这样才能抵消由于回溯带来的延时如图4.8，以 50 回溯深度为例，当回溯深度为 50 时，其接入信号和输出信号之间的延时也由此确定。

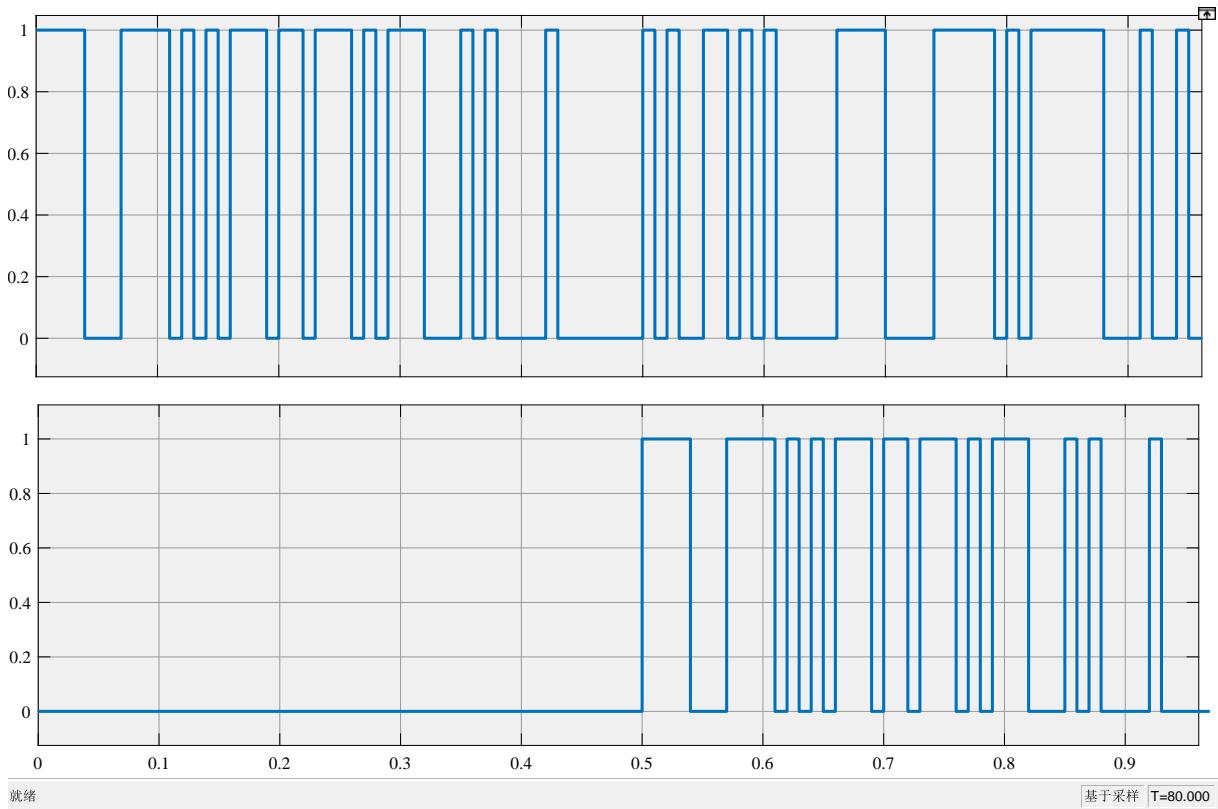


图 4.8 回溯深度与译码延时

### 4.3 matlab 仿真 64QAM 下的卷积编码

估算 AWGN 中硬判决和软判决维特比解码器的误码率 BER 性能。将性能与未编码的 64-QAM 链路进行比较。设置仿真参数。

```

1 clear; close all
2 rng default
3 M = 64;          % 状态空间
4 k = log2(M);     % 每符号的bit数
5 EbNoVec = (4:10)'; % Eb/No的值(dB)
6 numSymPerFrame = 1000; % 每帧QAM的符号数

```

对 BER 结果向量初始化:

```

1 berEstSoft = zeros(size(EbNoVec));
2 berEstHard = zeros(size(EbNoVec));

```

设置卷积码 Grellis 结构、速率 1/2，约束长度 7，回溯深度 32。

```

1 trellis = poly2trellis(7,[171 133]);
2 tbl = 32;

```

```
3 rate = 1/2;
```

主处理循环执行以下步骤:

1. 生成二进制数据
2. 对数据进行卷积编码
3. 将 QAM 调制应用于数据符号。为发送信号指定单位平均功率
4. 将调制后的信号通过 AWGN 信道传送
5. 使用硬判决和近似 LLR 方法解调接收信号。指定接收信号的单位平均功率
6. 维特比采用硬决策和非量化方法对信号进行译码
7. 计算误码率
8. While 循环继续处理数据，直到遇到 100 个错误或  $10^7$  位为止。

```

1 for n = 1:length(EbNoVec)
2 % 将谱能量密度比Eb/No转化为SNR
3 snrdB = EbNoVec(n) + 10*log10(k*rate);
4 % 单位平均信号功率的噪声方差计算。
5 noiseVar = 10.^(-snr dB/10);
6 % 初始化误码数和传输bit数
7 [numErrsSoft,numErrsHard,numBits] = deal(0);
8
9 while numErrsSoft < 100 && numBits < 1e7
10 % 生成二进制数据并转换为符号
11 dataIn = randi([0 1],numSymPerFrame*k,1);
12
13 % 卷积编码
14 dataEnc = convenc(dataIn,trellis);
15
16 % QAM 调制
17 txSig = qammod(dataEnc,M,'InputType','bit','UnitAveragePower',true);
18
19 % 通过 AWGN 信道
20 rxSig = awgn(txSig,snr dB,'measured');
21
22 % 使用硬判决解调噪声信号
23 % 软决策（近似LLR）方法。
24 rxDataHard = qamdemod(rxSig,M,'OutputType','bit','UnitAveragePower',true);
25 rxDataSoft = qamdemod(rxSig,M,'OutputType','approxllr',...
26 'UnitAveragePower',true,'NoiseVariance',noiseVar);
27
28 % 维特比对解调后的数据进行译码
29 dataHard = vitdec(rxDataHard,trellis,tbl,'cont','hard');
30 dataSoft = vitdec(rxDataSoft,trellis,tbl,'cont','unquant');
```

```

32 % 计算帧中的误码数。调整解码延迟，它等于回溯深度。
33 numErrsInFrameHard = biterr(dataIn(1:end-tbl),dataHard(tbl+1:end));
34 numErrsInFrameSoft = biterr(dataIn(1:end-tbl),dataSoft(tbl+1:end));
35
36 % 更新和位计数器
37 numErrsHard = numErrsHard + numErrsInFrameHard;
38 numErrsSoft = numErrsSoft + numErrsInFrameSoft;
39 numBits = numBits + numSymPerFrame*k;
40
41 end
42
43 % 计算误码率
44 berEstSoft(n) = numErrsSoft/numBits;
45 berEstHard(n) = numErrsHard/numBits;
46 end

```

绘制误码率曲线如图4.9所示：

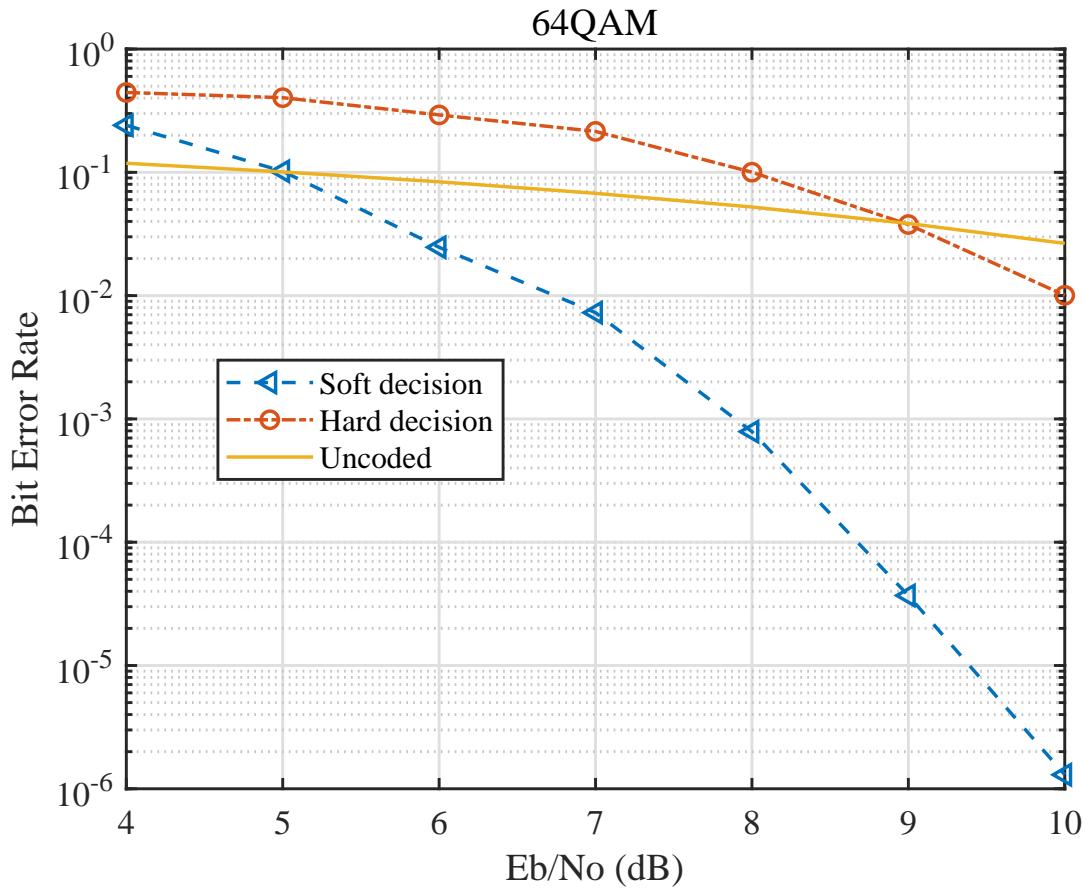


图 4.9 64QAM 调制下的软硬判决误码率曲线

从图中可以看出当  $Eb/No$  较小时，此时的噪声过大，即使采用维特比译码也不能对拯救其大噪声下的判决，误码率总体随着信噪比的增大而减小，这是因为信噪比越大，其中噪声对信息判决带来的干扰也会越小，从而误比特率也会越低，误比特率曲线符

合理论实际。同时对于软决策维特比译码，其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”，性能确实优于硬判决，符合理论。

## 5 心得体会

在这次课程设计中，我学会了如何有效的利用网络资源及图书馆的藏书，找到了几个很不错的专业网站，为以后的查阅专业方面的信息和相互之间的交流打下了坚实的基础。在这次课程设计中，我掌握了通信系统中差错控制拟信号数字化的过程，深刻理解了其中的原理，并且自己动手，用 Matlab 和 simulink 实现了这一过程，我自学学到了很多。在知识点上，我进一步学习了卷积码以及维特比译码算法、对卷积码的理解更加深刻，其实卷积码的内容不止如此有很多其他的类型，比如反馈式系统卷积码、删除卷积码、序列译码算法等，由于时间的关系，我并没有将 John G.Proakis 和 Masoud Salehi 大牛的卷积码内容全部看完，不过真的学到了许多。总结本次实验的结果：

- 卷积码编码能有效降低信息传输的差错，尤其是在信噪比较大时，因为此时有用的信号功率没有因噪声功率过大带来严重误判，从这里也可以看出维特比译码的纠错能力也是有限的。
- 维特比译码是一种最大似然法，对回溯长度的所有前一组幸存路径进行比较，取出距离较小的路径作为下一组幸存路径，它并不是在网格图上一次比较所有可能的  $2kL$  条路径（序列），而是接收一段，计算和比较一段，选择一段最大似然可能的码段，从而达到整个码序列是一个最大似然值得序列。
- 软判决译码（有时也称为“软输入维特比译码”）建立在这一观察上，它在译码之前不会数字化输入样本。相反，它使用具备连续性的模拟样本（经采样量化）作为译码器的输入。硬判决译码通过将接收到的电压信号与阈值进行比较，数字化后将其传递给译码器进行译码。结果，我们失去了部分信息。
- 维特比译码的误码率随码率的增加而降低，因为编码效率和纠错能力本身是一种矛盾，因此仿真符合理论实际。
- 当回溯深度较小时，更易误码，当回溯深度大至一定程度时，回溯深度对误码率的影响不大，其实主要的误码还是源于信噪比太小，导致有用信号微弱，收到的噪声影响大于回溯深度对误码率的影响。

## 参考文献

- [1] John G.Proakis、Masoud Salehi 主编, Digital Communications, 电子工业出版社, 2011 年
- [2] 李平安, 刘泉主编, 宽带移动通信原理及应用, 高等教育出版社, 2016 年
- [3] Mischa 主编, 移动通信, 电子工业出版社, 2006 年
- [4] 王正林.MATLAB/Simulink 与控制系统仿真. 北京: 电子工业出版社, 2012
- [5] 赵书兰.MATLAB 建模与仿真. 北京: 清华大学出版社, 2013
- [6] 丁亦农.Simulink 与信号处理. 北京: 北京航空航天大学出版社, 2010
- [7] 刘学勇. 详解 MATLAB/Simulink 通信系统建模与仿真. 北京: 电子工业出版社, 2011