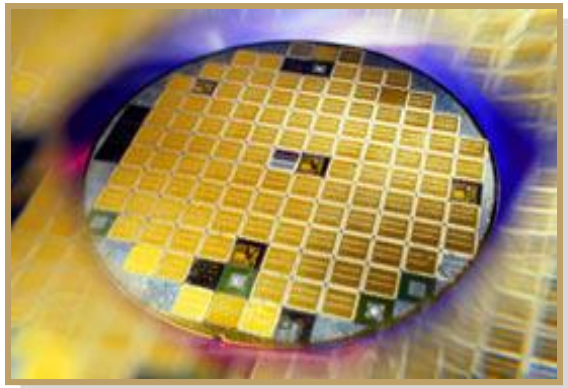
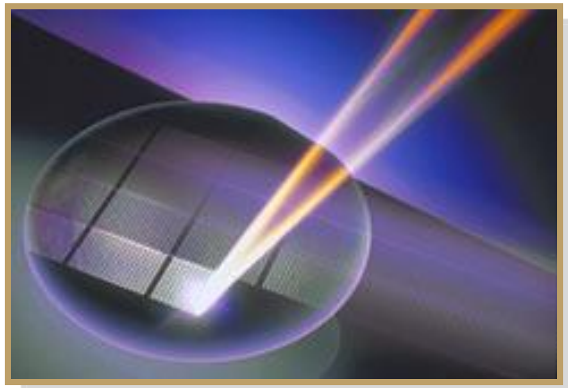
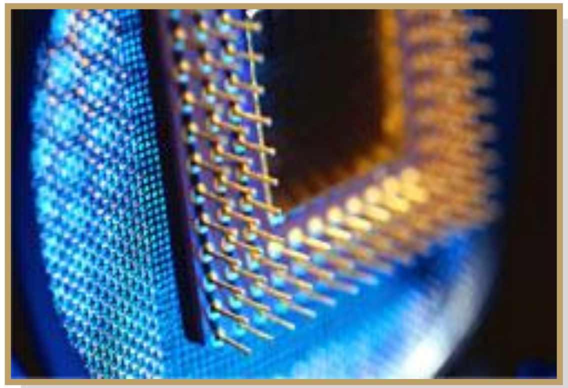
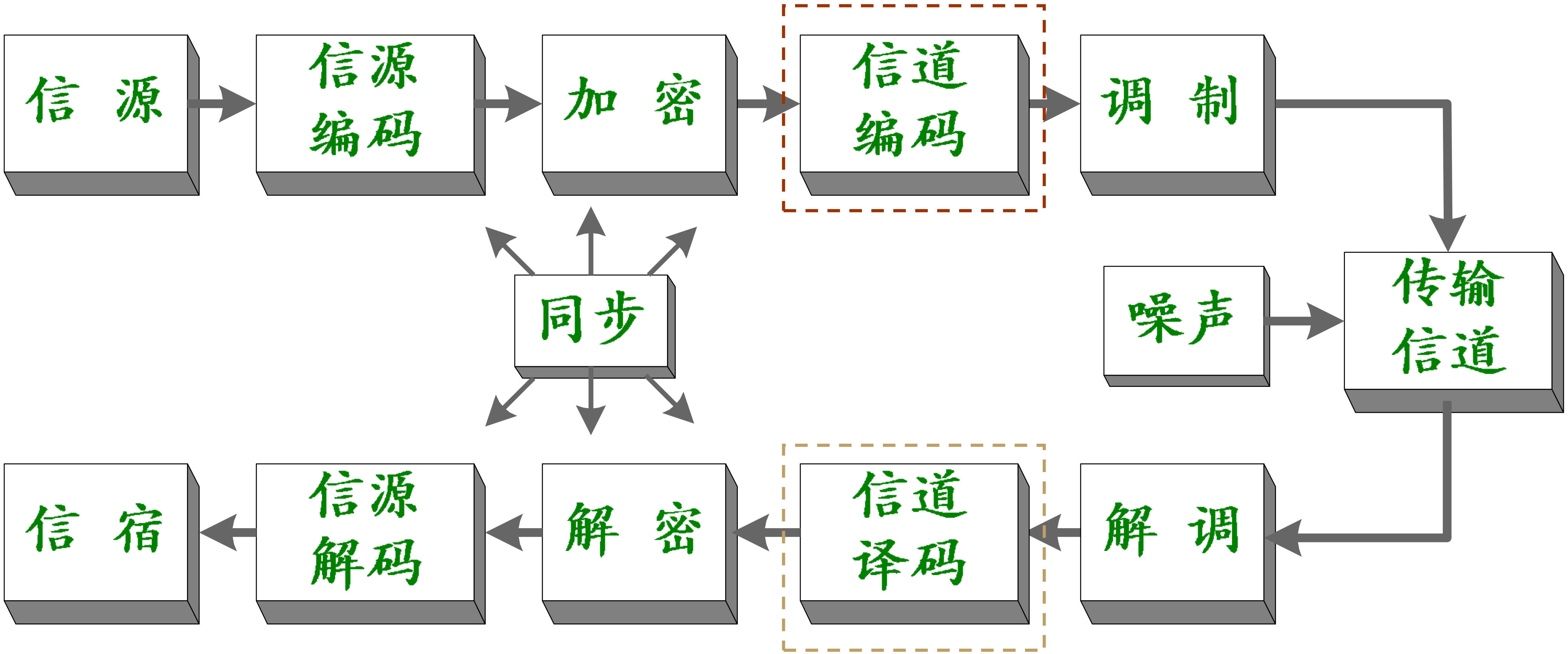
《VLSI数字通信原理与设计》实验课

实验三：卷积码编解码实验

目录

信道编码与译码



信道编码与译码

信道编码保证信息传输的正确性

在信息中插入冗余，使其获得检错或者纠错的能力。

数字移动通信系统中的信道编码

Turbo码（3/4G），LDPC码（5G数据信道），Polar码（5G控制信道）。

常见信道编码

Hamming码、BCH码、RS码、卷积码、Turbo码、LDPC码、Polar码。

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

信道编码

冗余

$$(a_0, a_1, \cdots, a_i, \cdots a_{k-1}, p_0, p_1, \cdots, p_{r-1})$$

传输或存储中发生错误

$$(a_0, {\color{red}a_1}, \cdots, {\color{red}a_i}, \cdots a_{k-1}, p_0, {\color{red}p_1}, \cdots, p_{r-1})$$

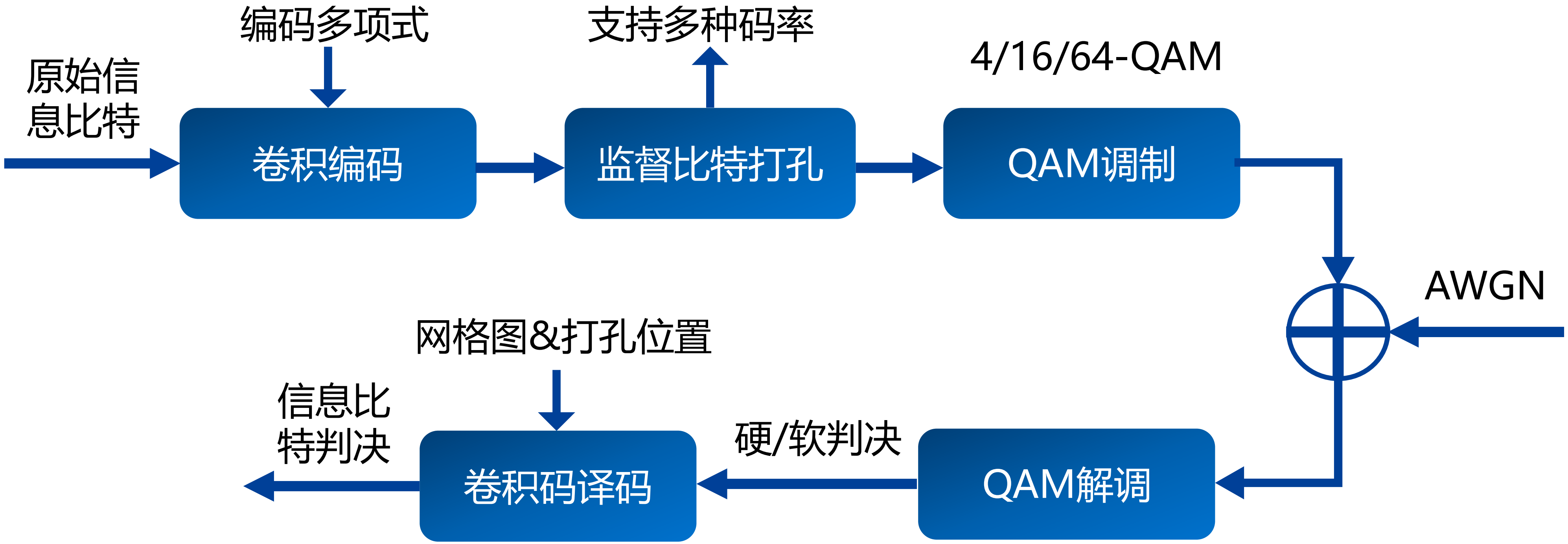
译码恢复原信息

$$(a_0, a_1, \cdots, a_i, \cdots a_{k-1}, p_0, p_1, \cdots, p_{r-1})$$

编解码仿真平台

提供如下蒙特卡洛仿真平台，支持多种译码算法同时仿真对比

其中编码函数和部分译码算法为MATLAB工具箱自带



支持 · 硬判决Viterbi译码
· 软判决Viterbi译码
· 最大后验概率译码

目录

实验目标

- 理解**信道编解码与凿孔**在通信系统中的作用，掌握卷积码**编码原理**以及**最大似然译码**流程
- 编写**(2,1,7)卷积码编码**模块与**Viterbi译码模块(支持凿孔)**
- 完成整个链路的仿真以及相关性能分析
- 实验环境：MATLAB
- ***本次实验至多两人一组，鼓励一人一组。***

目录

卷积码（Convolutional Code）

卷积码属纠错码，通过移位寄存器与模2加法进行编码

卷积码于1955年由Peter Elias提出；

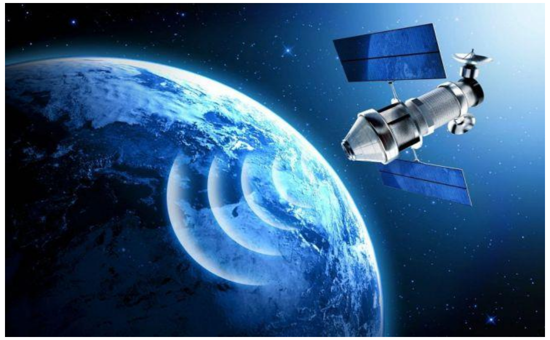
1967年Andrew Viterbi提出了著名的**Viterbi译码算法**；

1974 年，Bahl, Cocke, Jelinek, Raviv给出了卷积码的最大后验概率译码方案——BCJR算法

Peter Elias

卷积码由于码长、码率的灵活性以及高效的编解码而用途广泛

全世界每秒Viterbi译码器恢复的二进制比特数是10^{15}

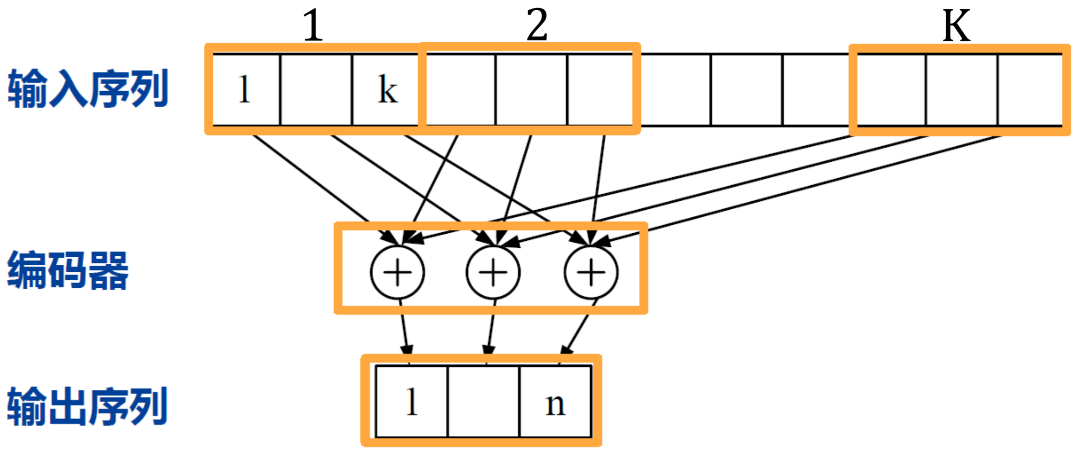


Andrew Viterbi

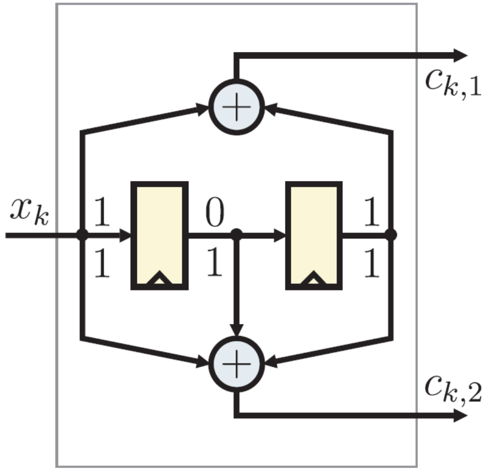
卷积编码的基本概念

(n, k, K)卷积编码：

通过将输入信息比特序列与编码器做模2卷积运算，将k位信息比特编成n位编码比特。此n比特不仅与当前k位信息比特有关，还与前面$(K-1)$段信息比特有关



(a) (n, k, K)卷积编码过程



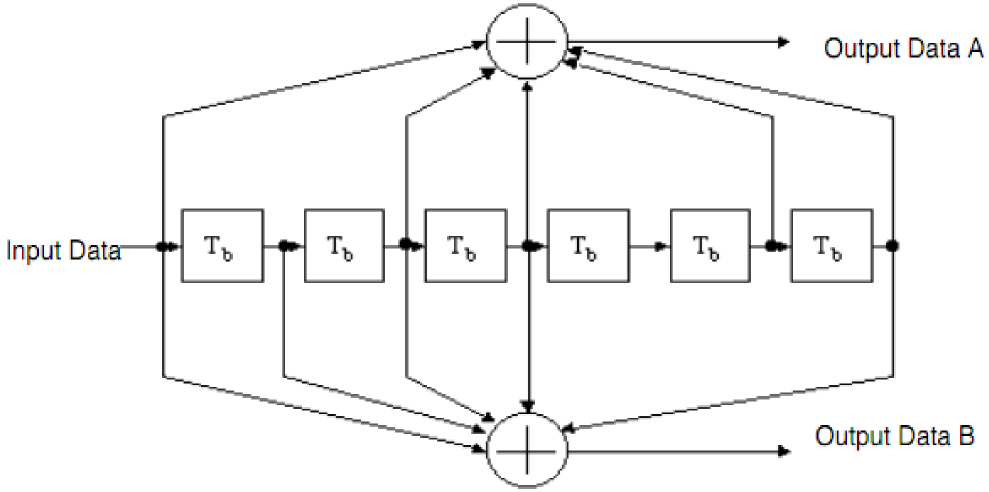
(b) 例子：(2,1,3)卷积编码器电路

(n, k, K)卷积编码参数：

- k ：编码器输入信息比特数
- n ：编码器输出编码比特数
- K：约束长度（每个输出序列约束K个输入序列）
- 码率R ：信息比特占比，$R = k/n$

(2,1,7)卷积编码

- **IEEE 802.11n**协议中规定卷积编码使用的八进制生成多项式为 $[133 \quad 171]_O$ ，即(2,1,7)卷积码：



$$\longrightarrow g_1 = 1011011_b \longrightarrow 133_O$$

$$\longrightarrow g_2 = 1111001_b \longrightarrow 171_O$$

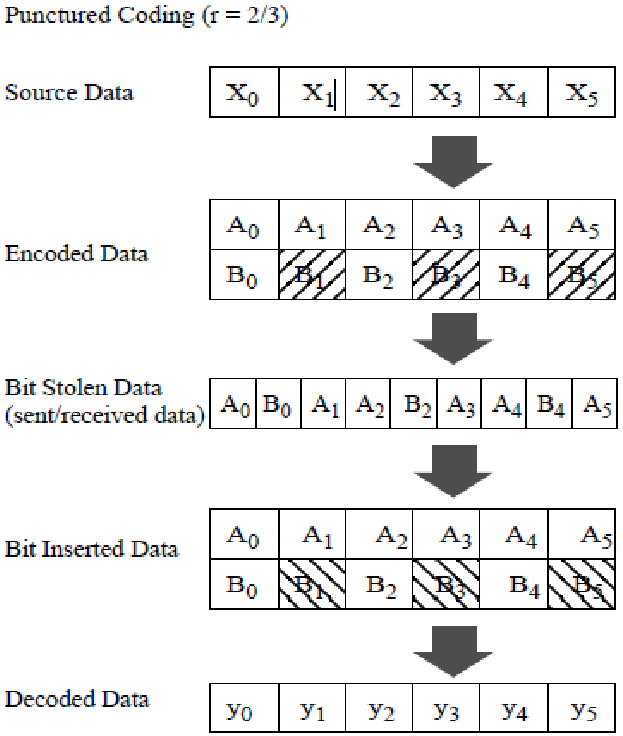
- 该编码方案具有$2^n = 4$种输出状态， $2^k = 2$种输入状态， $2^{K-1} = 64$个寄存器状态

注意：编码前要使得所有寄存器清零；编码最后要在信息序列最后补上(K-1)=6个0，确保信息序列尾部也能完全移除寄存器，否则纠错性能将会下降！(或者，寄存器状态在编码前、后已知)

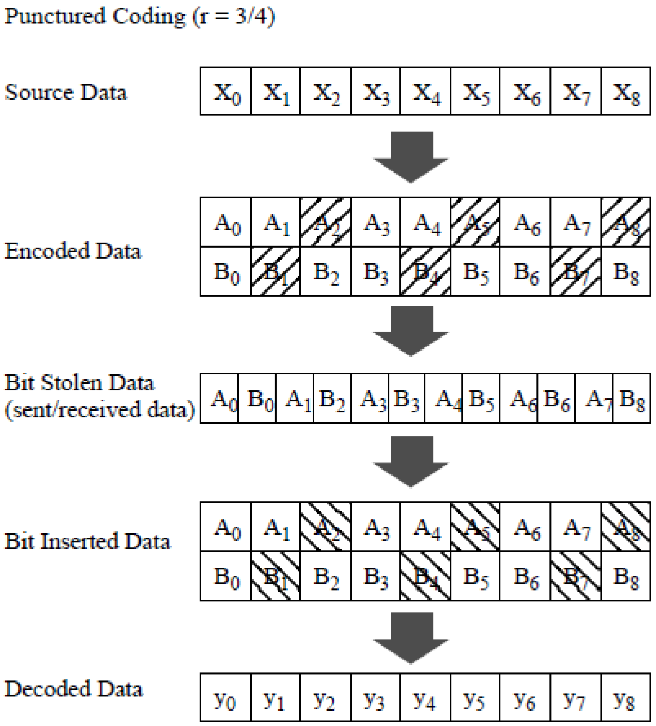
卷积码凿孔（Puncturing）

凿孔：通过减少冗余，牺牲部分纠错性能来提高码率，降低发射功耗

码率为1/2的卷积码，可通过**不发射部分监督比特**来实现高码率传输



凿孔方案1：码率1/2 -> 2/3



凿孔方案2：码率1/2 -> 3/4

为了实现更高的传输效率，部分监督比特(图中阴影部分)在发射前，被“挤掉”了

卷积码凿孔（Puncturing）

为了降低凿孔对性能的影响，凿孔比特位置需精心设计

对1/2码率的卷积码，IEEE 802.11n给出了如下的凿孔(发射)比特方案

码率$1/2 \Rightarrow 2/3$：

对于2个信息比特，编码生成4个比特c_1, c_2, c_3, c_4，发射索引为

1 2 3

处的3个编码比特

码率变化：$\frac{1}{2} \times \frac{4}{3} = \frac{2}{3}$

码率$1/2 \Rightarrow 3/4$：

对于3个信息比特，编码生成6个比特$c_1, c_2, \dots, c_6$，发射索引为

1 2 3 6

处的4个编码比特

码率变化：$\frac{1}{2} \times \frac{6}{4} = \frac{3}{4}$

思考：凿孔带来了部分监督比特的缺失，在使用Viterbi译码时该如何调整译码流程 (如何处理凿孔位置的分支度量)?

目录

卷积码译码

- 卷积码的译码方式可分为硬判决译码和软判决译码 (例如，Viterbi译码)

  硬判决译码：译码输入为接收序列量化后的0，1值

  软判决译码：译码输入为未量化的模拟值(0~1)

- 相较于软判决，硬判决译码在量化时丢失了信息从而导致性能有2~3dB损失

  例如，接收到模拟量0.51和0.99时，软判决译码会利用其不同的可信度，而在硬判决译码中，两者都被视为1并相同对待，从而导致了信息损失

- 给定接收序列，**Viterbi译码利用动态规划的思想借助网格图进行最大似然译码**

  硬判决Viterbi：加、比、选 ⟶ 最小化与接收序列之间的**汉明距离**

  软判决Viterbi：加、比、选 ⟶ 最小化与接收序列之间的**欧式距离**

Viterbi译码：最大似然译码

- Viterbi译码利用动态规划思想在网格图上寻找最小路径，实现最大似然译码

最大似然译码：给定码字$\boldsymbol{c}$经过信道的观测$\boldsymbol{r}$，寻找最有可能(最大化似然函数)的码字$\hat{\boldsymbol{c}}$，使得码字错误概率$\Pr(\hat{\boldsymbol{c}} \neq \boldsymbol{c})$最小化，即寻找

$$\hat{\boldsymbol{c}} = \underset{\boldsymbol{c}}{\operatorname{argmax}}\, p(\boldsymbol{r}|\boldsymbol{c}),\, p\text{为似然函数} \tag{1}$$

- 对于硬判决译码($r_i = 0 \text{ or } 1$)，最大似然译码(1)等价于最小化汉明距离：

$$\hat{\boldsymbol{c}} = \underset{\boldsymbol{c}}{\operatorname{argmin}}\, d_{\mathrm{H}}(\boldsymbol{r}, \boldsymbol{c}) \tag{2}$$

- 对于软判决译码($0 \leq r_i \leq 1$)，最大似然译码(1)等价于最小化欧式距离：

$$\hat{\boldsymbol{c}} = \underset{\boldsymbol{c}}{\operatorname{argmin}}\, d_{\mathrm{E}}(\boldsymbol{r}, \boldsymbol{c}) \tag{3}$$

注：(2)(3)的详细推导见附录

Viterbi译码介绍

- 以(2,1,7)卷积码为例，给出如下定义 (暂不考虑凿孔对译码的影响)

对于长度为L的网格图，任意第$l(1 \leq l \leq L)$个时刻处理两个比特，定义**汉明/欧式距离**

$$\lambda_l = \sum_{j=1}^{2} d_{\mathrm{H}}(r_l^{(j)}, c_l^{(j)}) = r_l^{(1)} \oplus c_l^{(1)} + r_l^{(2)} \oplus c_l^{(2)} \qquad \text{对于硬判决译码}$$

$$\lambda_l = \sum_{j=1}^{2} d_{\mathrm{E}}(r_l^{(j)}, c_l^{(j)}) = \left(r_l^{(1)} - c_l^{(1)}\right)^2 + \left(r_l^{(2)} - c_l^{(2)}\right)^2 \quad \text{对于软判决译码}$$

为**分支度量 (branch metric)**，定义$\Gamma_L = \sum_{l=1}^{L} \lambda_l$为**路径度量(path metric)**

- Viterbi译码中涉及到的变量如下所示：

$\lambda_l(s', s)$：第$(l-1)$时刻的状态s'到第l时刻的状态s的分支度量，其中$s', s \in \{0,1,\ldots,63\}$

$\Gamma_{l-1}(s')$：第$(l-1)$时刻抵达状态s'处的幸存路径的路径度量

$\Gamma_l(s', s)$：将第$(l-1)$时刻的状态s'延展到第l时刻的状态s，所得路径的路径度量，即

$$\Gamma_l(s', s) = \Gamma_{l-1}(s') + \lambda_l(s', s)$$

注：若考虑调制(例如BPSK)，软判决Viterbi译码中常用双极性$\{-1,+1\}$来代替$\{0,1\}$，此时接收值r_i为$c_i \in \{-1,+1\}$的非量化观测

Viterbi译码流程

加-比-选 迭代寻找最短路径：	
初始化：令$\Gamma_0(0)=0$，其余$\Gamma_0(s')=-\infty, \forall s' \in \{1,\dots,63\}$ (编码前所有寄存器置为零状态)	
for $l=1$ **to** L	
1. 根据卷积码的状态转移图，计算所有可能的分支度量$\lambda_l(s',s)$	
2. 对于第$(l-1)$时刻的每个状态s'，以及其所有可能抵达的第l时刻的状态s，计算路径度量 $\Gamma_l(s',s)=\Gamma_{l-1}(s')+\lambda_l(s',s)$	**//加**
3. 对于第l时刻的每个状态s，比较所有的$\Gamma_l(s',s)$得到幸存路径度量$\Gamma_l(s)$，即 $\Gamma_l(s)=\min_{s'}\Gamma_l(s',s)$	**//比**
并将这些幸存路径(状态和度量)选出，保存在幸存路径矩阵中	**//选**
end for	
回溯译码：	
由于信息序列末尾补零，末尾时刻寄存器强制为零状态，最大似然路径是第L时刻状态为0的幸存路径。从该路径的第L时刻回溯至第1时刻，即可得到最大似然码字	

Viterbi译码补充：凿孔

对凿孔的处理：仍视作对原1/2码率卷积码的译码；在计算分支度量时跳过凿孔比特位置的距离计算

以硬判决译码为例，若第l组比特的第2个编码比特$c_l^{(2)}$被凿孔，分支度量由

$$\lambda_l = \sum_{j=1}^{2} d_{\mathrm{H}}(r_l^{(j)}, c_l^{(j)}) = r_l^{(1)} \oplus c_l^{(1)} + r_l^{(2)} \oplus c_l^{(2)}$$

无凿孔

修改为

$$\lambda_l = \sum_{j=1}^{2} d_{\mathrm{H}}(r_l^{(j)}, c_l^{(j)}) = r_l^{(1)} \oplus c_l^{(1)}$$

$c_l^{(2)}$被凿孔

Viterbi译码补充：滑动窗减小存储开销

- **每个状态的幸存路径**都需要存储对应的**比特路径**及其**度量**，当序列长度较长时，存储开销较大

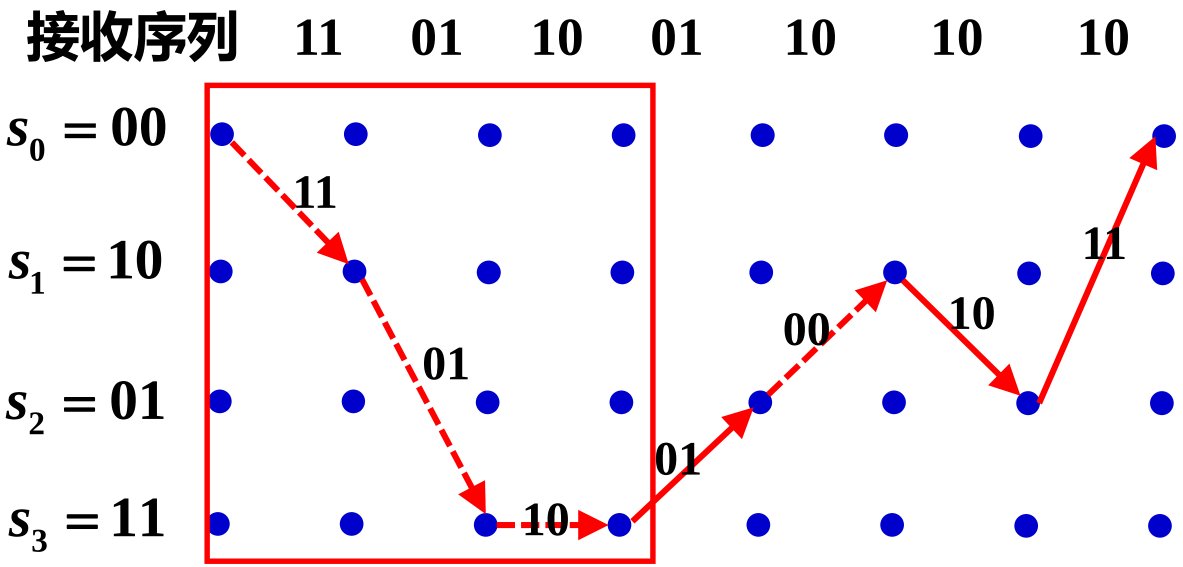
- 有限回溯长度的Viterbi译码：**设置回溯长度$\boldsymbol{TbLen}$**(一般为$(K-1)$的5-10倍)**分段译码**，可以减小内存开销且几乎无损性能

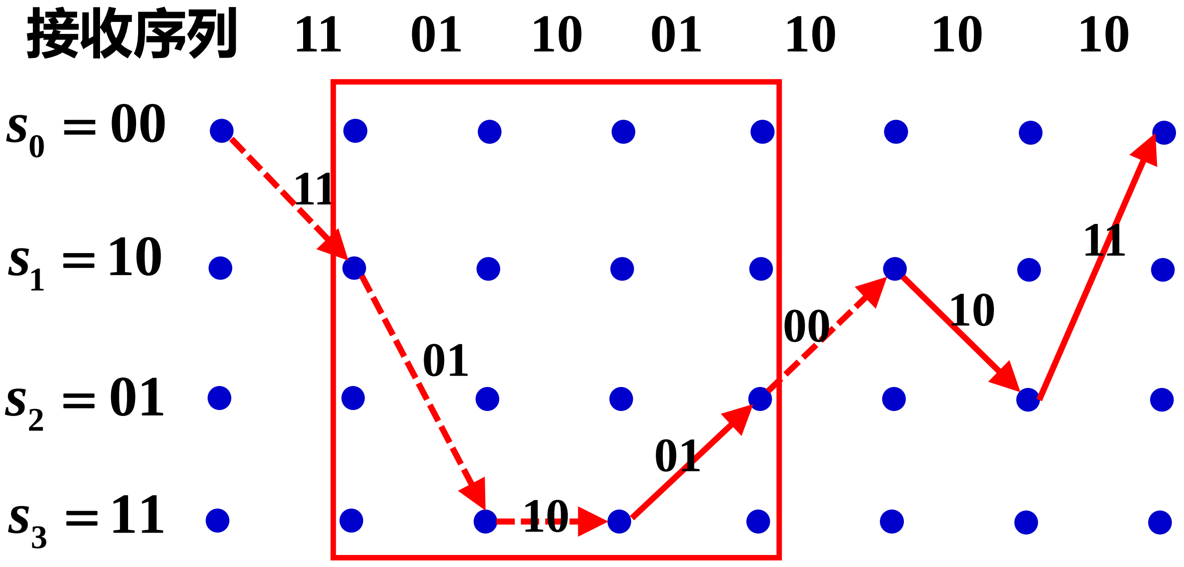
- 有限回溯长度的Viterbi译码**通过滑动窗口实现**，以长度为$TbLen$的滑动窗口在网格上滑动译码，只有**滑动窗口内的比特**将会被存储

  从初始状态开始到滑动窗口存储了2^{K-1}条幸存路径，每条路径存储了$TbLen$比特及其分支度量，此时滑动窗口的存储已满，在进行下一个比特的译码前，需要对第一个比特进行判决，以腾出新的存储空间存放下一个比特

Viterbi译码补充：滑动窗减小存储开销

滑动窗口举例：$TbLen = 3$（这里没有约束长度5倍以上，仅作为示例）





- 从初始状态到滑动窗口被4条幸存路径的比特及其度量填满
- 此时滑动窗口内近存放每条幸存路径的前3个比特

- 处理下一比特之前，滑动窗口需要对第一个比特进行判决并空出其存储位置给下一比特
- 此时滑动窗口内存放每条路径的比特2、3、4

目录

基本函数

函数poly2trellis：由生成多项式与约束长度生成网格图信息

```
trellis = poly2trellis(7, [133 171]); % requires communication toolbox
```

其中结构体trellis中的参数可以完全描述该卷积码，包括：

numInputSymbols： $2^k = 2$个输入状态数 (0 1)

numOutputSymbols： $2^n = 4$个输出状态数 (00 01 10 11)

numStates： $2^{K-1} = 64$个寄存器状态数 (000000~111111)

nextStates： numStates×2^k的寄存器状态转移矩阵。1~64行表示000000~111111个当前状态，1~2列表示输入0，1，矩阵元素表示当前状态+输入得到的下一寄存器状态 (0~63)

outputs： numStates×2^k的输出矩阵。行列意义与矩阵nextStates一致，矩阵元素表示当前状态+输入产生的输出(0~3)

基本函数

函数convenc：利用网格图对信息比特序列进行卷积编码

```
coded_bits = convenc(info_bits, trellis); % requires communication toolbox
```

凿孔（等价于发射编码比特序列中，对应索引index处的比特）

```
coded_bits_punctured = coded_bits(index);
```

对于2^Q阶调制，码率为R的系统，比特功率$E_b = E_s/(RQ)$，从而比特信噪比(dB)为

$$\frac{E_b}{N_0}(dB) = \frac{E_s}{N_0}(dB) - 10\log_{10} RQ$$

其中符号功率E_s一般归一化为1

基本函数

硬判决译码

```
info_bits_hat = vitdec(rx_bits, trellis, tblen, 'term', 'hard', puncturing_pattern);
```

info_bits_hat：对信息比特的判决

rx_bits：接收01序列

trellis：poly2trellis函数生成的网格图信息

tblen：回溯长度

% A rate 1/2 code has a TracebackDepth of 5(k-1).
% A rate 2/3 code has a TracebackDepth of 7.5(k-1).
% A rate 3/4 code has a TracebackDepth of 10(k-1). k is constrain length.

'term'：代表编码前、后寄存器都置零

'hard'：代表输入序列为01硬判决

puncturing_pattern：代表打孔比特位置，例如[1 1 1 0] (逻辑数组，表示每4个编码比特凿去第4个)

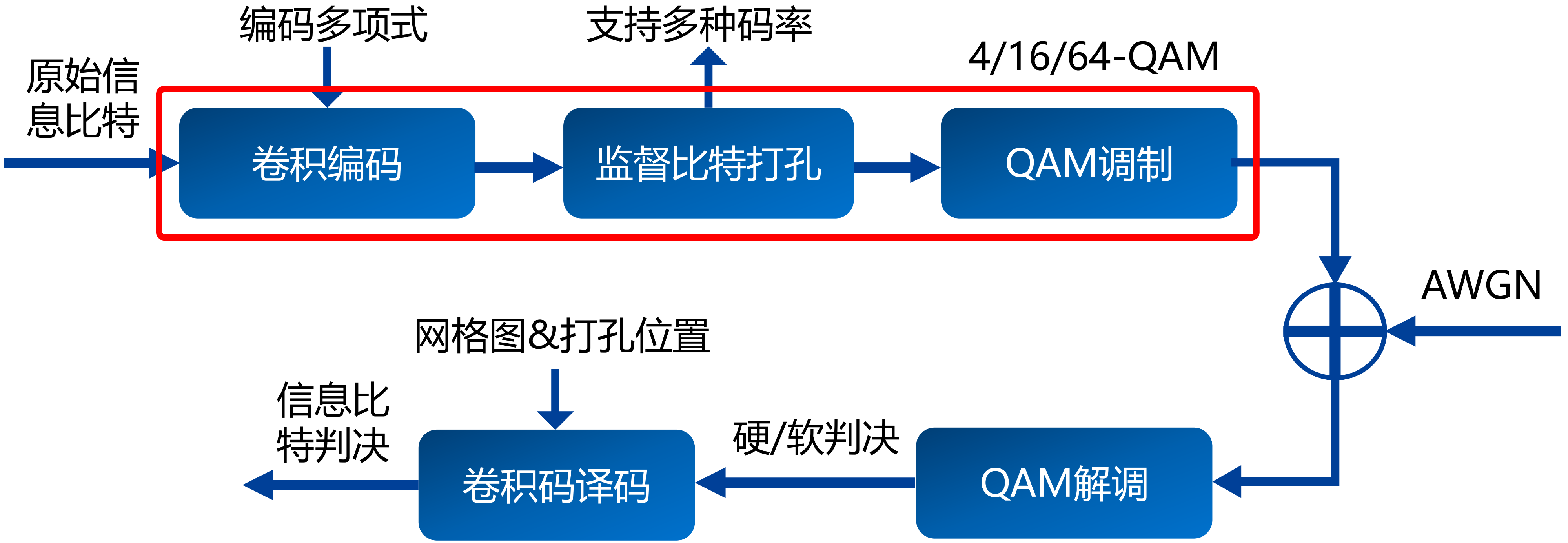
软判决译码

```
info_bits_hat = vitdec(rx_data, trellis, tblen, 'term', 'unquant', puncturing_pattern);
```

rx_data：未量化软输入，这里正值对应逻辑0，负值对应逻辑1

'unquant'：代表输入序列为非量化值

仿真平台介绍



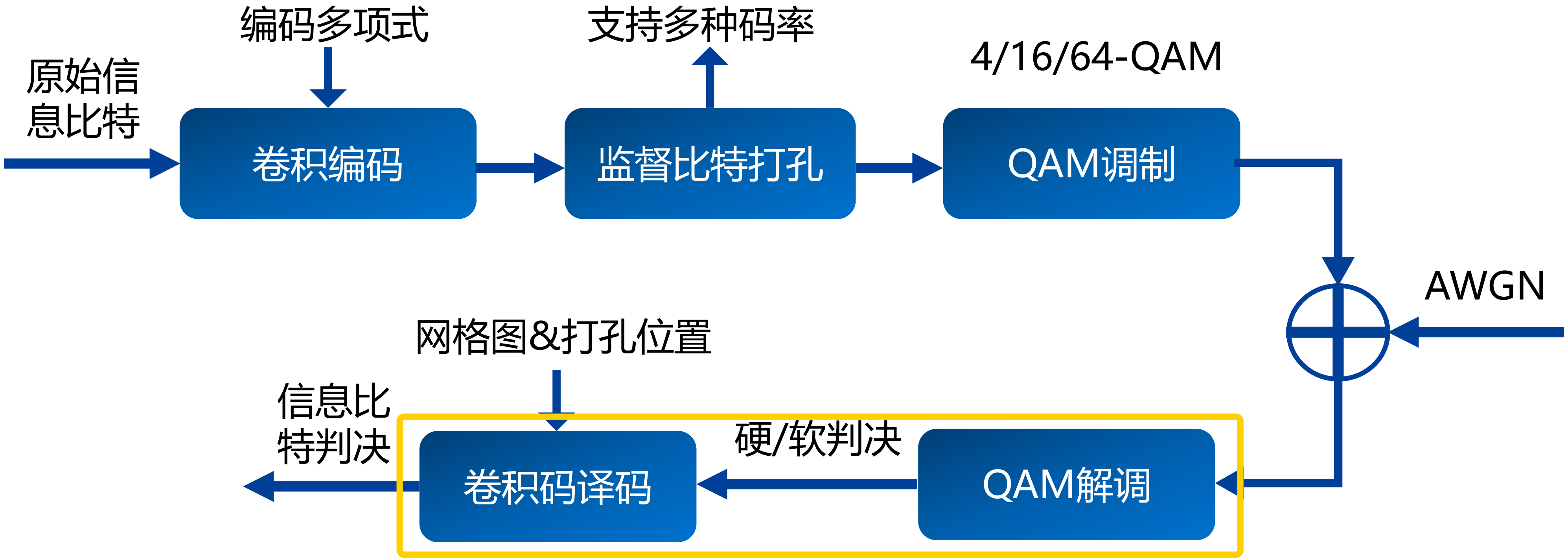
CC_sim.m：主函数，根据随机种子生成伪随机数，支持多种译码算法在不同调制/码率下误比特率/误码字率的蒙特卡洛仿真

initial.m：初始化模块，准备4/16/64-QAM调制的星座点集合与卷积码网格图、打孔位置

CC_encoder.m：编码模块，**采用MATLAB自带函数对信息比特做卷积编码(请自行实现)，并完成了打孔功能，**可实现1/2，2/3，3/4三种码率的卷积编码

QAM_mod.m：调制模块，支持编码比特的4/16/64-QAM调制

仿真平台介绍

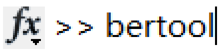


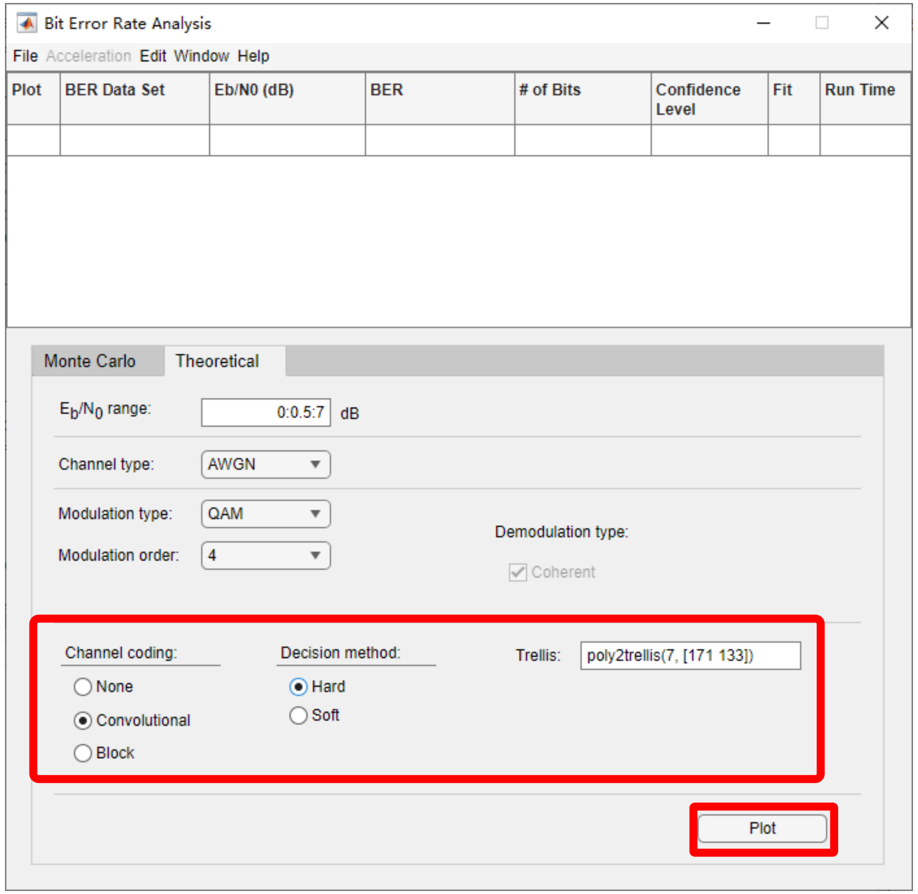
hard_demod/soft_demod.m：解调模块，生成接收序列的硬判决比特/软判决LLR

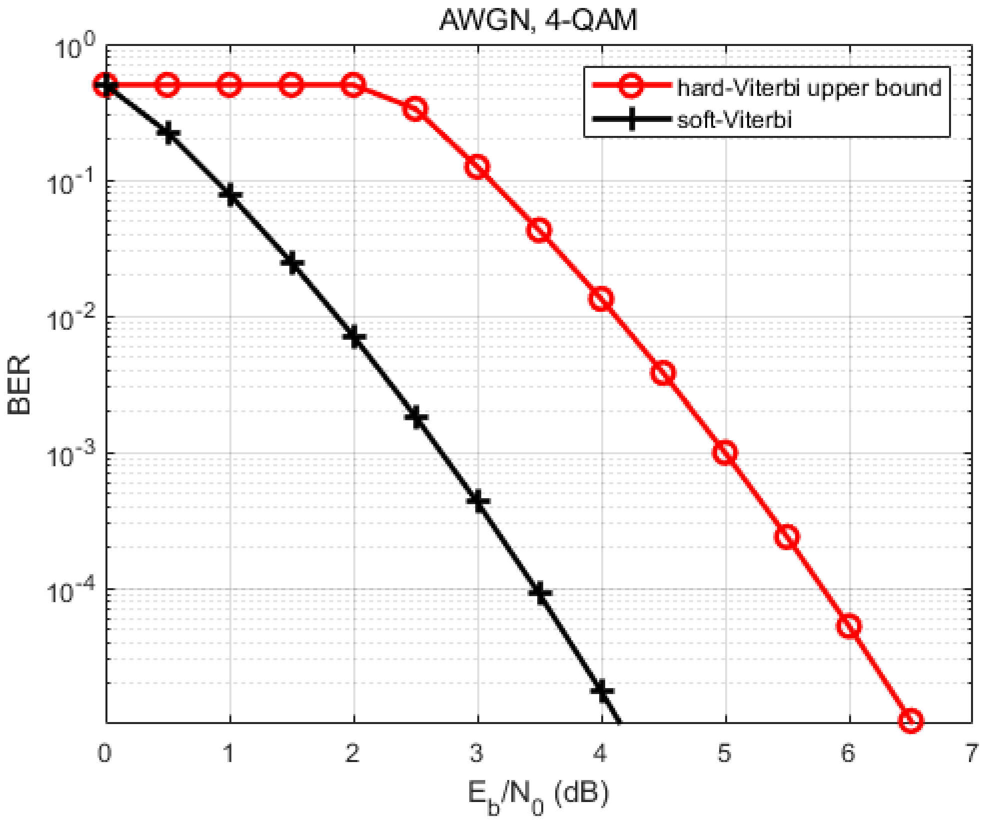
CC_decoder：卷积码译码模块，包含**MATLAB自带的硬判决和软判决Viterbi译码函数(请自行实现其中之一)**以及最大后验概率 (MAP) 译码算法

理论BER性能

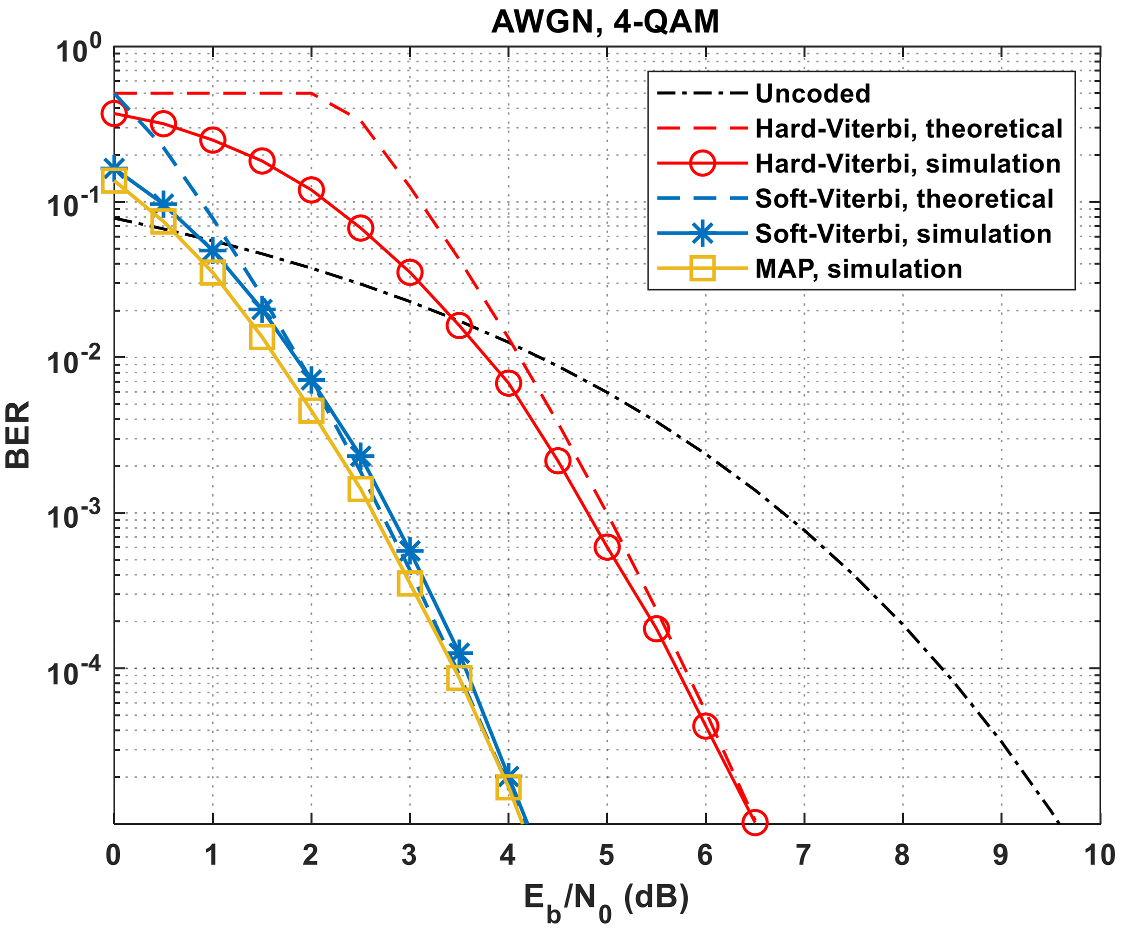
- BERTOOL给出了**各种调制下的硬判决Viterbi译码的性能上界渐进曲线。**对于QPSK/4-QAM，还给出了**软判决Viterbi译码的性能渐进曲线**







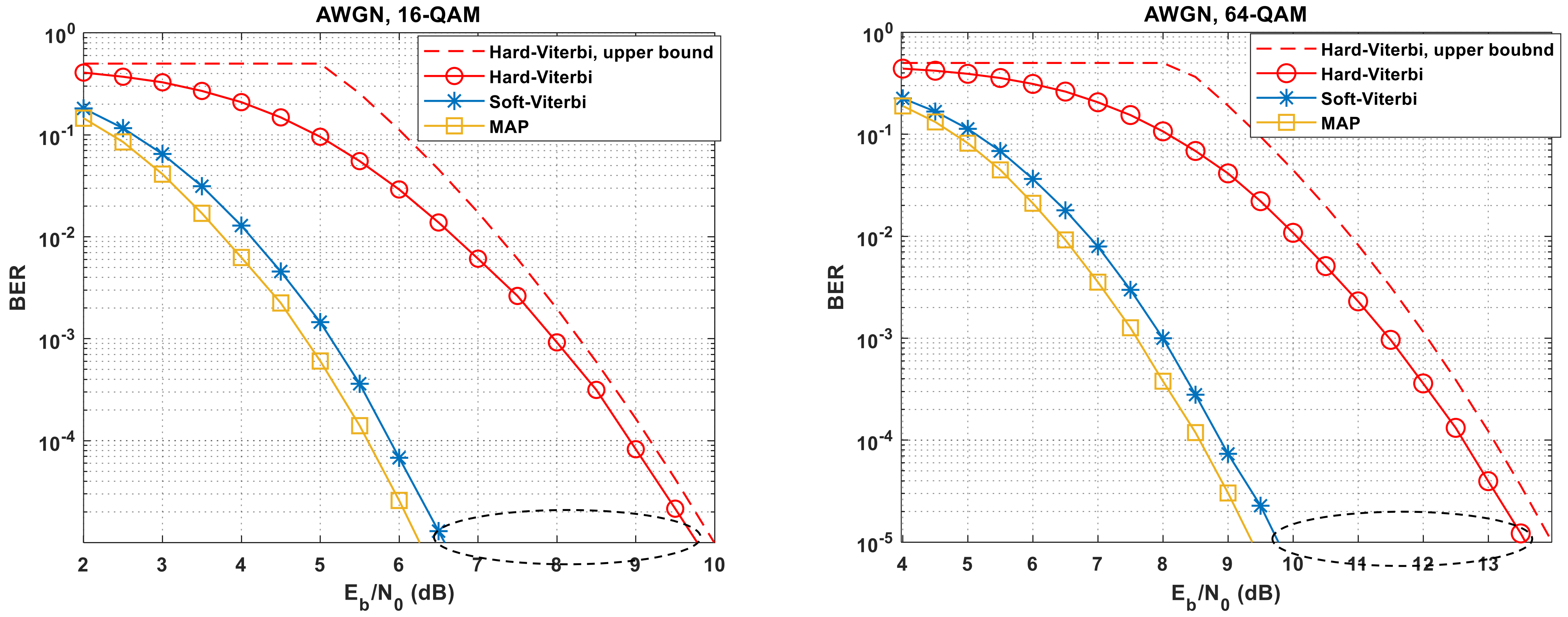
(2,1,7)卷积码性能：仿真 vs 理论



通过仿真观察到，4-QAM调制下软判决译码约胜过硬判决2dB多；

思考：在低信噪比区域，为何未编码系统的BER性能更好?

(2,1,7)卷积码性能：硬判决 vs 软判决



16/64-QAM调制下，软判决的性能约胜过硬判决3~4dB

目录

实验工作

- 编写(2,1,7)卷积码编码模块并验证(与MATLAB自带函数进行比较)
- 对于1/2，2/3，3/4码率的卷积码，**任选下面两种译码模式之一**
  - 硬判决Viterbi译码
  - 软判决Viterbi译码

  编写译码模块并验证（与MATLAB自带函数进行比较，注意凿孔的处理)
- 完成1/2，2/3，3/4码率，4/16/64-QAM调制下的BER性能仿真。对于1/2码率的仿真结果，结合BERTOOL给出的理论曲线进行对比

报告要求

- 关于实验报告的完成要求：
  - 所要求递交的实验报告应该至少包含以下内容：
    - **卷积码编码实现细节与验证结果**
    - **Viterbi译码实现细节(包含凿孔的处理)与验证结果**
    - **Viterbi译码仿真结果与分析(码率1/2的仿真结果与BERTOOL理论曲线进行比较；若实现软判决译码，请在仿真结果图上标注其相对于硬判决理论曲线的性能增益)**
    - **思考题的回答**
    - **实验工作分工等**
  - 提交时间：2022/5/30 23：59 之前

附录

最大似然译码推导*

- 对于硬判决情况，编码比特$c_i \in \{0,1\}$，接收判决$r_i \in \{0,1\}$。由于噪声干扰，设判决错误概率为$\varepsilon(0 < \varepsilon < 0.5)$，则对于每个条件概率$p(r_i|c_i), i = 1, \ldots, n$，有

$$p(r_i = 1|c_i = 0) = p(r_i = 0|c_i = 1) = \varepsilon$$

$$p(r_i = 1|c_i = 1) = p(r_i = 0|c_i = 0) = 1 - \varepsilon$$

由于各组$\{c_i, r_i\}$独立同分布，似然函数$p(\boldsymbol{r}|\boldsymbol{c}) = \prod_{i=1}^{n} p(r_i|c_i)$，最大似然译码等价于

$$\begin{aligned}
\hat{\boldsymbol{c}} &= \underset{\boldsymbol{c}}{\operatorname{argmax}} \ln p(\boldsymbol{r}|\boldsymbol{c}) \\
&= \underset{\boldsymbol{c}}{\operatorname{argmax}} \sum_{i=1}^{n} \ln p(r_i|c_i) \\
&= \underset{\boldsymbol{c}}{\operatorname{argmax}} \left[d_{\mathrm{H}}(\boldsymbol{r}, \boldsymbol{c}) \ln(\varepsilon) + (n - d_{\mathrm{H}}(\boldsymbol{r}, \boldsymbol{c})) \ln(1 - \varepsilon)\right] \\
&= \underset{\boldsymbol{c}}{\operatorname{argmax}} \left[d_{\mathrm{H}}(\boldsymbol{r}, \boldsymbol{c}) \ln\left(\frac{\varepsilon}{1 - \varepsilon}\right) + n \ln(1 - \varepsilon)\right] \\
&= \underset{\boldsymbol{c}}{\operatorname{argmin}}\, d_{\mathrm{H}}(\boldsymbol{r}, \boldsymbol{c})
\end{aligned}$$

对应于第16页的(2)式，**即硬判决最大似然译码等价于最小化汉明距离**

最大似然译码推导*

- 对于软判决，编码比特$c_i \in \{0,1\}$，接收判决r_i为c_i经过AWGN信道后的输出，有

$$r_i = c_i + n_i$$

其中n_i为均值为0，方差为N_0的高斯白噪声，则条件概率分布(高斯分布)为

$$p(r_i|c_i) = \frac{1}{\sqrt{2\pi N_0}} \exp[-(r_i - c_i)^2/(2N_0)]$$

由于独立同分布，似然函数$p(\boldsymbol{r}|\boldsymbol{c}) = \prod_{i=1}^{n} p(r_i|c_i)$，最大似然译码等价于

$$\begin{aligned}\hat{\boldsymbol{c}} &= \underset{\boldsymbol{c}}{\operatorname{argmax}} \sum_{i=1}^{n} \ln\left(\frac{1}{\sqrt{2\pi N_0}} \exp[-(r_i - c_i)^2/(2N_0)]\right) \\ &= \underset{\boldsymbol{c}}{\operatorname{argmin}} \sum_{i=1}^{n} (r_i - c_i)^2 \\ &= \underset{\boldsymbol{c}}{\operatorname{argmin}}\, d_{\mathrm{E}}(\boldsymbol{r}, \boldsymbol{c})\end{aligned}$$

对应于第16页的(3)式，**即软判决最大似然译码等价于最小化欧式距离**

软判决Viterbi译码的细节补充*

- 软判决Viterbi的输入r_i是通过解调器给出的对数似然比$LLR(c_i)$计算的，由公式

$$LLR(c_i) = ln\left(\frac{\Pr[c_i = +1|r_i]}{\Pr[c_i = -1|r_i]}\right) = \frac{2r_i}{N_0}$$

r_i是发射符号$c_i \in \{-1, +1\}$经过AWGN信道后的直接观测，可计算为 (平台中已给出)

$$r_i = 0.5N_0LLR(c_i)$$

- 第17页给出了软判决Viterbi的分支度量计算：

$$\lambda_l = \left(r_l^{(1)} - c_l^{(1)}\right)^2 + \left(r_l^{(2)} - c_l^{(2)}\right)^2$$

其中$c_l^{(1)}, c_l^{(2)} \in \{-1, +1\}$。上式中**与$c_l^{(1)}, c_l^{(2)}$无关的量可以去掉，不影响最终对$c$的判决结果**，因此将式中平方展开后丢掉无关量，$\lambda_l$可简化计算为

$$\lambda_l = -r_l^{(1)} * c_l^{(1)} - r_l^{(2)} * c_l^{(2)}$$

从而极大地降低了计算复杂度