**信道编码实践**

**一、实验目的**

1. 掌握信道编码的基本原理与常见编码方法
2. 实现经典信道编码算法（奇偶校验、汉明码、CRC、卷积码）
3. 分析不同编码方式的纠错能力与性能差异

**二、实验原理**

|  |  |  |
| --- | --- | --- |
| 编码类型 | 核心原理 | 数学表达 |
| 奇偶校验 | 添加1位校验位使数据位中1的个数为奇/偶 | *P*=*d*1⊕*d*2⊕... |
| **汉明码** | 通过校验位覆盖特定数据位，实现单比特纠错 | 2^*r*≥*k*+*r*+1 |
| **CRC** | 利用生成多项式进行模2除法，生成校验码 | *R*(*x*)=(*D*(*x*)⋅*x^n*)mod*G*(*x*) |
| **卷积码** | 通过移位寄存器和异或运算生成连续编码序列 | 状态转移方程+网格图 |

**三、实验流程**

**代码关键逻辑解析**

1. **BER计算流程**
   * **生成数据**：generate\_random\_bits生成随机比特流
   * **编码**：调用各编码方案的编码函数（需适配统一接口）
   * **噪声引入**：simulate\_bsc\_channel模拟二进制对称信道
   * **解码**：处理可能出现的检错失败情况（如CRC校验失败）
   * **误差统计**：对比原始数据和解码数据的差异比特数
2. **各编码方案适配器**
   * **奇偶校验**：当检测到错误时，假设无法纠错（返回None），此处示例强制通过
   * **汉明码**：分块处理每4位数据，补零处理不足4位的尾部
   * **CRC**：需处理字节与比特流的转换，检错失败时返回None
   * **卷积码**：示例简化了解码逻辑，实际需实现维特比算法
3. **结果可视化**
   * 使用对数坐标展示BER变化
   * 不同编码方案用不同颜色/标记区分
   * 网格线辅助观察数量级差异

**四、注意事项**

1. **卷积码解码**
   * 示例中的卷积码解码为简化版，实际应实现维特比译码算法
   * 推荐使用scipy.signal.convolve优化计算
2. **CRC实现细节**
   * 需确保crc\_encode和crc\_check正确处理字节对齐
   * 推荐使用预计算查表法加速CRC计算
3. **测试次数选择**
   * 低误码率（如0.1%）需增加n\_tests次数（如1000次）
   * 高误码率（>5%）可减少测试次数（如50次）
4. **随机种子设置**
   * 在测试前设置np.random.seed(0)保证结果可复现

**五、扩展任务**

1. **高阶挑战**：实现维特比译码算法用于卷积码解码（参考scikit-commpy库）
2. **不同信道**：测试不同信道条件下的性能
3. **创新实验**：使用LDPC码（PyLDPC库），Turbo码（commpy.channelcoding.turbo\_encode/decode），对比传统信道编码性能

**实验报告基本要求:**

1. 简述各编码算法的实现过程
2. 分析不同误码率下的纠错性能差异
3. 比较各编码的冗余度（编码效率）
4. 讨论实际应用场景选择依据
5. 扩展实验（可选）