网络建模与分析大作业-实验报告

姓名：贾宏宇 学号：2019214512 班级：软硕192班 ### 一、实验环境介绍

1.1 电脑软硬件配置

- 软件平台：

  操作系统：Windows 10专业版，64位操作系统

  开发软件：Microsoft Visual Studio Premium 2012

- 硬件配置：

  CPU：Intel i7-9700

  内存：16GB

1.2 编译环境

- 编程语言：C++、MFC框架
- 编译选项：使用Release选项、本地Windows调试器

1.3 运行环境与方式

程序运行界面如图所示，提供下述功能点:

1. `ASCII`编码与解码
2. Huffman编码与解码
3. Hamming码纠错与检错功能
4. OOK信号调制与解调
5. 自定义OOK调制周期
6. 为OOK调制添加指定方差的高斯白噪声
7. 动态绘制OOK解调与调制图像

二、霍夫曼编码

给定传输内容，理解霍夫曼编码对内容的压缩作用，设计并实现霍夫曼编码和解码算法，展示编码和解码的结果，分析霍夫曼编码的效率和数据压缩率。

2.1 引言

霍夫曼编码使用不定长编码，而非固定位数的二进制编码方法。依据特定字符串中不同字符出现频率的差别，来构造一棵二叉树，使这棵树叶子结点的带权路径之和最小。该编码方法极为有效地提升了编码的效率以及数据压缩率。

2.2 编码和解码算法实现概述

由于霍夫曼编码采用的是不定长编码，因此其编码方法必须保证任一字符的编码都不是另一个字符的编码的前缀，这种编码方法又被称为前缀码。

霍夫曼编码使用二叉树构造前缀码，让权值大的树更靠近树根，也就是让出现频率高的字符编码更短，这就使带权路径之和取得了最小值。在具体的实现过程中，我使用优先队列保存树节点，在优先队列中出现频率最低的节点被放在了队列的头部，只要优先队列中存在超过两个节点，就不断地进行`pop`操作，构造新的父节点；对于解码过程，只需要将接收到的二进制字符串不断地从根节点向下遍历即可（遇零走左分支，遇一走右分支），伪代码以及实现细节详见**代码说明文档**。

2.3 结果分析

计算机中的字符都用`ASCII`码表示，而`ASCII`码是定长编码，所有字符的编码都是8位，但是在平常遇到的字符串中并不一定所有字符都会出现，在这种情况下，可以用霍夫曼编码极大提升数据压缩率和编码效率，事先统计字符串中各个字符出现的频率，然后构造哈夫曼树和哈夫曼编码，并把编码作为字典保存，这样文件就得到了压缩。

使用霍夫曼编码和`ASCII`编码对`《sense 8》`美剧中的一段台词进行编码，并将两者编码结果详细展示。此外还对随机生成的数字序列和英文字符序列进行编码，分析其平均码长和数据压缩率。

《sense8》字幕（共161字符）：

```
The improbable unfolding of recent events have led me to consider that no one thing is one thing only. It is in this unfamiliar realm, we find new possibilities.
```

霍夫曼编码（668字符）：

```
00000111011001111001010000000101100011010111000101111110001110101111000001011
11001111010111010001101011111110011101010001111101000101100010001111111001110
01100110101111111100100101101011010111001101011110111010110001111101000001111101
00110101100001001010111100100100001101110001110100110110101111001110111110101
10101011110111101001101100101111111001110010001011010101111011110100110110010
11111110011101010111111101000011100110011000000010100111001000101100101111110100
11011001000101100000101111001111011110000001011101010101111000111010001011100
11111101100000000110110000000011110001110101111000111101111011000000110000101
10100010001001011100001011101010100101001100100011100
```

ASCII编码（1288字符）：

```
0101010001101000011001010010000001101001011011010111000000111001001101111101100
0100110000101100010011011000110010100100000011101010110111001100110011011110 1
1011000110010001101001011011100110011100100000011011110110011000100000011100 1
0011001010110001101100101011011100111010000100000011001010111011001100101011 0
1110011101000111001100100000011010000110000101110110011001010010000001101100 0
1100101011001000010000001101101011001010010000001110100011011110010000001100 0
1101101111011011100111001101101001011001000110010101110010001000000111010001 1
0100001100001011101000010000001101110011011110010000001101111011011100110010 1
0010000001110100011010000110100101101110011001110010000001101001011100110010 0
0000110111101101110011001010010000001110100011010000110100101101110011001110 0
1000000110111101101110011011000111100100101110001000000100100101110100001000 0
0011010010111001100100000011010010110111000100000011101000110100001101001011 1
0011001000000111010101101110011001100110000101101101011010010110110001101001 0
1100001011100100010000000111001001100101011000010110110001101101001011000010 00
0001110111011001010010000001100110011010010110111001100100000010000001101110011
0010101110111001000000011100000110111101110111001101110011011010010110001001101001
011011000110100101110100001101001011001010111001100101110
```

霍夫曼编码和ASCII编码的平均码长见下表：

平均码长	霍夫曼编码	ASCII编码	数据压缩率
《sense 8》	4.14907	8	48.1%
随机英文字符（100位）	4.07000	8	49.125%
随机数字（100位）	3.31000	8	58.625%

数据压缩率的计算（以《sense 8》台词为例）：相对于ASCII编码，其数据压缩率为：
$1-(668/1288)=48.1\%$

三、基于正弦信号的OOK调制

编码后的数据通过信道进行传输，试设计一种调制解调方法通过信道传输数据。

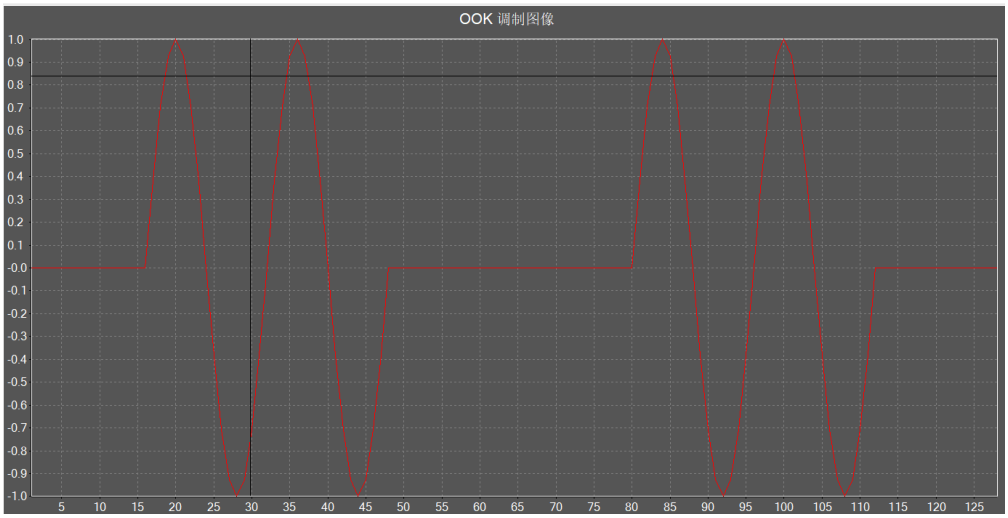
3.1引言

无论是模拟调制，还是数字调制，都是采用调制信号去控制载波信号的三要素：幅度、频率和相位，分别称为调幅、调频和调相。本文采用数字调制ASK中的特例：OOK调制实现信号在信道上的传输，该调制方法只有比特`1`有信号，比特`0`没有信号，所以称为On-Off Keying，简称OOK调制。

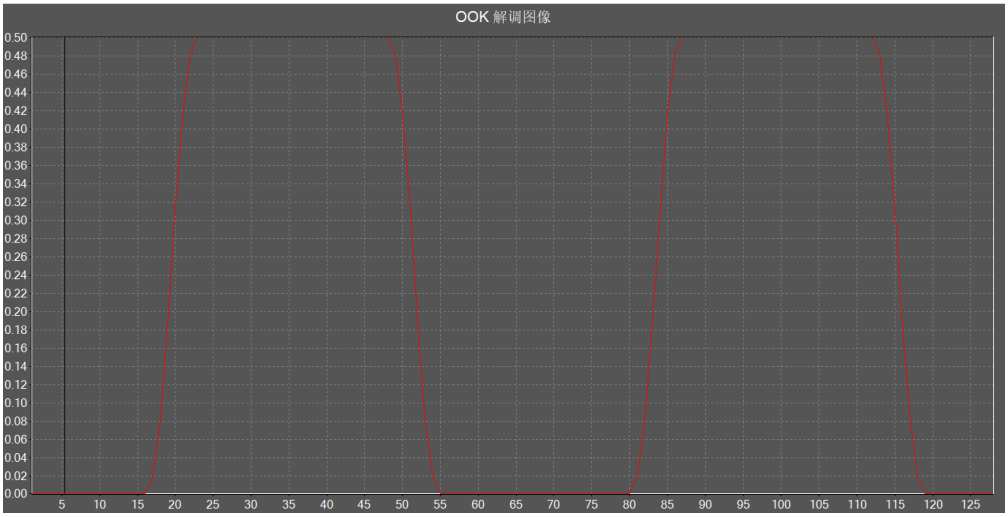
3.2 OOK调制实现概述

基于正弦信号实现OOK调制，给定编码周期`period`，那么对于每一个待编码的比特，都可按照公式$t[n]=x[n]sin(\Omega_c n)$来进行调制，其中$\Omega_c=2\pi/period$，$x[n]$的取值只有两种即`0`或`1`。在进行解码时，使用相似的公式$z[n]=t[n]sin(\Omega_c n)$进行解调。而后对$z[n]$进行滑动平均滤波得到最终结果$r[n]$，伪代码以及实现细节详见**代码说明文档**。

3.3 结果分析



周期设定为16，对二进制序列`01100110`进行OOK调制的结果如上图所示，可以看到`y`轴取值在$[0, 1]之间，`x`轴长度为`128`，对该波形进行解调，得到结果如下图所示。



在理论情况下信道中不存在噪声，因此该解码结果较为理想，在下一节会详细讲述添加噪声之后的编码结果与分析。

四、信道中添加高斯白噪声

在真实传输中，数据传输可能会有噪声产生影响，试在生成的传输信号上添加噪声，测试噪声对解码成功率的影响。

4.1 引言

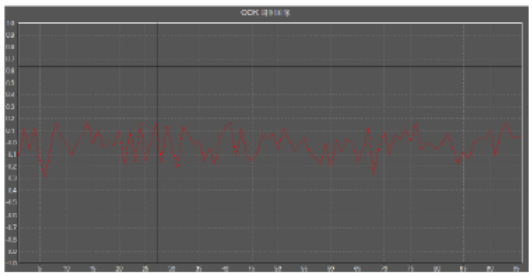
对于一个纯随机过程来说，若其期望为0，方差为常数，则称之为白噪声过程。高斯噪声则是指概率密度函数服从正态分布的噪声。高斯分布，记为$N(\mu, \sigma^2)$，其中μ为高斯分布的均值（数学期望），σ^2为高斯分布的方差。由于高斯白噪声能够反映实际通信信道中的噪声情况，能够比较真实的反映信道噪声的一些特性，并且可以用具体的数学表达式表示，适合分析、计算系统的抗噪声性能，所以广泛应用于通信系统的理论分析。

4.2 高斯白噪声实现概述

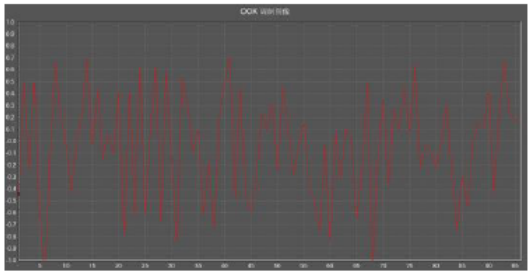
对于高斯白噪声的生成方式，经过调研，在C++中存在封装好的库函数，使用`normal_distribution`进行调用即可实现，伪代码以及实现细节详见**代码说明文档**。

4.3 结果分析

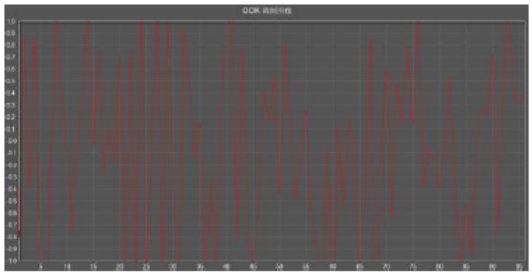
由于对通过信道传输数据的调制方式采用OOK的方法进行传输，因此单独查看高斯白噪声在每个采样节点的分布，只需要对全零的二进制序列，进行调制后添加噪声即可。噪声的均值为0，对于不同方差，生成的图像分别如下图所示。其中对于不同噪声对于解码成功率的影响，详见下一节的比较。



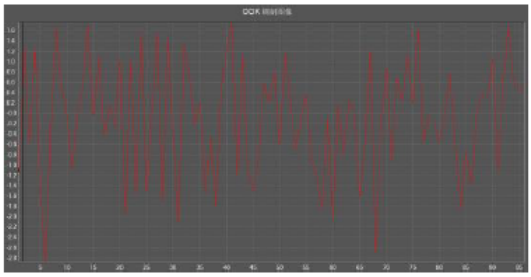
$\mu = 0.1$



$\mu = 0.4$



$\mu = 0.7$

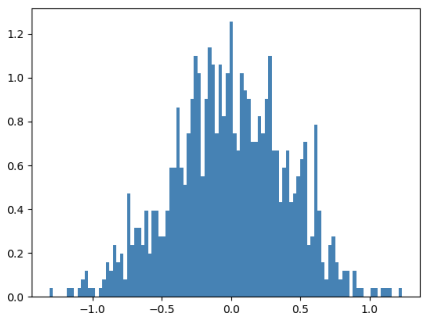


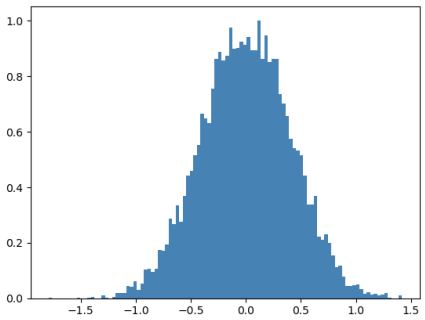
$\mu = 1.0$

此外，为了验证生成数据的正确性，单独将`normal_distribution`所生成的数据（均值为0，方差0.4）保存在data.txt文件中，而后编写python脚本对生成的数据进行直方图统计，左图是数据量为1000的分布，右图是数据量为10000的分布，可以发现其分布基本满足正态分布。

```python
import matplotlib.pyplot as plt

fd = open('data.txt', 'r')
li = []
for line in fd.readlines():
    li.append(float(line))
plt.hist(li, bins=100, color='steelblue', density=True)
plt.show()
```





五、Hamming Code

理解纠错检错的原理，展示方法如何在数据发生错误时能够对数据进行一定程度检错或者纠错。思考相关方法怎么能够在有噪声的情况下提高网络性能，并展示相关方法的效果。

5.1 引言

Hamming Code（汉明码）是一种线性纠错码，与其他的错误校验码类似，汉明码也利用了奇偶校验位的概念，通过在数据位后面增加一些比特，验证数据的有效性。汉明码不仅可以用来检测传输数据时发生的错误，还可以用来修正错误。（汉明码只能修正一位错误，对于两位或者两位以上的错误无法正确纠正）

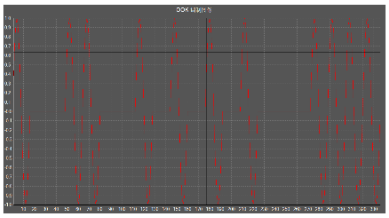
5.2 汉明码实现概述

假设数据由`m`个数据位（信息）和`r`个冗余位（校验）组成，其中`r`个校验位是由与之相关的`m`个数据位的函数计算获得的（也可理解为矩阵运算），令数据块的总长度为$n = m + r$，则利用汉明码进行校验所需的最小校验位个数需满足公式$2^r \geq (m + r + 1)$，在汉明码中二的幂次方的位(1, 2, 4, ...)是校验位，其余位用来填充数据，根据码率的计算公式$R = m/n = 1 - \frac{r+1}{2^r-1}$可知，使用(7, 4)汉明码进行编码得到的码率最高，因此在具体实现时也采用该方法，伪代码以及实现细节详见**代码说明文档。**

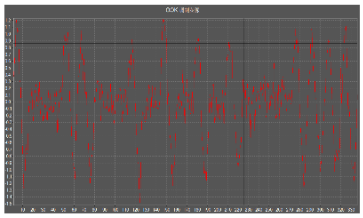
5.3结果分析

本节以`model`为例，进行编码和调制解调，并且增加不同数值的噪声，对比不同方法的鲁棒性和解码正确性。

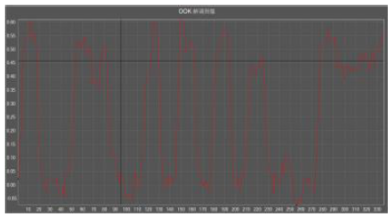
不失一般性，假设高斯白噪声的方差为0.2，固定编码周期为16，对汉明码编码后的二进制字符串进行OOK调制和解调的图片分别如下：



OOK 调制



添加噪声



OOK 解调

除此之外，在周期不同、高斯白噪声方差不同的情况下，对不同的编解码方式进行了详细的比较，截图不进行一一列举，将结果进行整理，表格如下（随机数种子固定为1222，方差取值为$[0.1, 1]$，步长为0.1）：

Noise	period=8	period=16	period=24
ASCII Code	0.5	0.6	0.6
Huffman Code	0.5	0.5	0.9
Hamming Code	**1.0**	**1.0**	**0.9**

该表中统计了不同编码方式，在OOK编码不同周期的情况下，噪声方差取何值时出现错误，例如使用`Hamming Code`编码方式，OOK编码周期为`8`时，噪声方差为`1.0`时解码报错。由此表可知，Hamming Code由于提供纠错检错机制，因此其抵抗噪声的效果最佳。

注：虽然随机数种子固定，但是由于不同编码方式的编码长度不同，因此其进行调制时的高斯白噪声也有所差别，在不同情况下结果可能会有微小差别。

六、实验总结

通过本次《网络建模与分析》课程，系统全面地学习了如何对网络进行建模，以及报文在网络信道中传输可能会遇到的各种情况和报文传输的整体流程，从对字符内容的二进制编码开始，再到经由信道传输调制后的信号。由于信道当中噪声的存在，还需要为信息增加纠错检错机制，使得数据能够在一定程度上纠错。

由各项实验结果可知，汉明码有效的提高了信道的整体容错机制。如果不使用汉明码进行编码，当高斯白噪声的方差取`0.6`左右时，对于其他没有纠错功能的编码方式，很容易报错或者解码错误；而使用汉明码这一编码方式，能够在有噪声的情况下，传递更多完整的信息到达对端，从而有效地提高网络性能。