第5章 通信系统的信源与信道

通信系统一般由信源、信宿(收信者)、发端设备、收端设备和传输媒介等组成。

通信系统都是在有噪声的环境下工作的。设计模拟通信系统时采用最小均方误差准则,即收信端输出的信号噪声比最大。设计数字通信系统时,采用最小错误概率准则,即根据所选用的传输媒介和噪声的统计特性,选用最佳调制体制,设计最佳信号和最佳接收机。

5.1 通信系统的基本模型

通信系统有其特有的模型结构,最基本的模型是点对点通信系统模型,根据信源输出信号类型的不同,又有模拟通信系统模型和数字通信系统模型。下面对这三个模型进行简单的介绍。

1. 点对点通信系统基本模型

最简单的通信系统负责将信号有效地从一个地方传输到另一个地方,称为点对点通信系统。任何点对点通信系统都由发送端(信源和发送设备)、接收端(接收设备和信宿)以及中间的物理信道组成,其模型如图 5-1 所示。

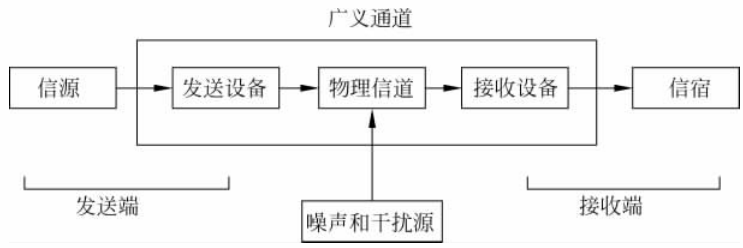

图 5-1 点对点通信系统模型图

信源即信息的发源地,信源可以是人,也可以是机器。在数学上,信源的输出是一个随时间变化的随机函数,根据随机函数的不同形式,信源可以分为连续信源和离散信源两类。

发送设备,负责将信源输出的信号变换为适合信道传输的形式,

使之匹配于信号传输特性并送入信道中。发送设备进行以传输为目的的全部信号处理工作，可能包含不同物理量表示的信号之间的转换，也可能包含信号不同形式之间的转换。

物理信道是信号传输的通路。按照传输媒介的不同，可以分为有线信道和无线信道两类；按照信道参数是否随时间变化，可以分为时不变信道（也称恒参信道）和时变信道（变参信道）两类。

在信道中，信号波形将发生畸变，功率随传输距离增加而衰减，并混入噪声以及干扰。在通信模型中，通常将通信设备内部产生的噪声等价地归并为信道中混入的噪声，这样，信号处理设备就建模为无噪的。

接收设备，其功能与发送设备相反，负责将发送端信息从含有噪声和畸变的接收信号中尽可能正确地提取出来。接收设备的信号处理目的是进行对应于发送设备功能的反变换，如解调、译码、将信号转换为信源发送的原始信号物理形式，同时尽可能好地抑制信道噪声，补偿或校正信道畸变引起的信号失真，最终将还原的信号送给信宿。

2. 模拟通信系统模型

如果信源输出是模拟信号，在发送设备中没有将其转换为数字信号，而是直接对其进行时域或频域处理（如放大、滤波、调制等）之后进行传输，则这样的通信系统称为模拟通信系统。模拟通信系统的模型如图 5-2 所示。

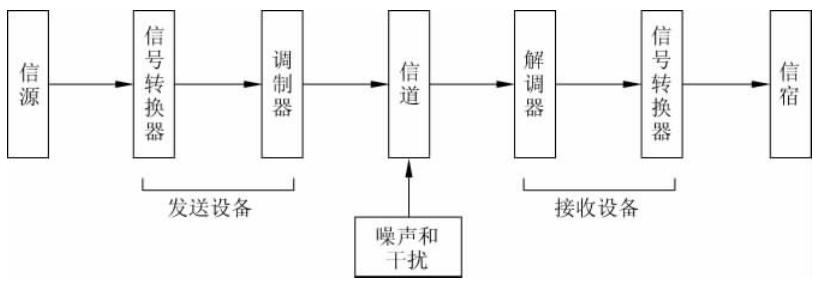

图 5-2　模拟通信系统的模型

在发送端，信号转换器负责将其他物理量表示的仿真信号转换为仿真电信号，如各种传感器、摄像头、话筒等。

基带处理部分对输入的模拟电信号进行放大、滤波后，送入调制器进行调制，转变为频带信号，称为已调信号。频带处理部分负责对已调信号的滤波、上变频以及功率放大，并输出到有线信道中或通过天线发送到无线信道中。

在接收端，信号经过频带处理部分选频接收、下变频和中频放大后，送入解调器进行解调，还原出基带信号，再经过适当的基带信号处理后送入信号转换器，如显示器、扬声器等，最终还原为最初发送类型物理量表示的仿真信号。

对于短距离有线传输，如有线对讲系统，可以不使用调制和解调，这样的系统就是模拟基带传输系统。

但是对于大多数模拟通信系统来说，为了将多路信号复用在同一物理媒介上传输，抑制干扰并匹配天线传输特性，必须使用调制和解调对信号进行频谱搬移，这样的传输

系统就是模拟频带传输系统。

3. 数字通信系统模型

如果信源输出是数字信号,或信源输出的仿真信号经过了模数转换成了数字信号,再进行处理和传输,则这样的通信系统称为数字通信系统。数字通信系统的模型如图 5-3 所示。

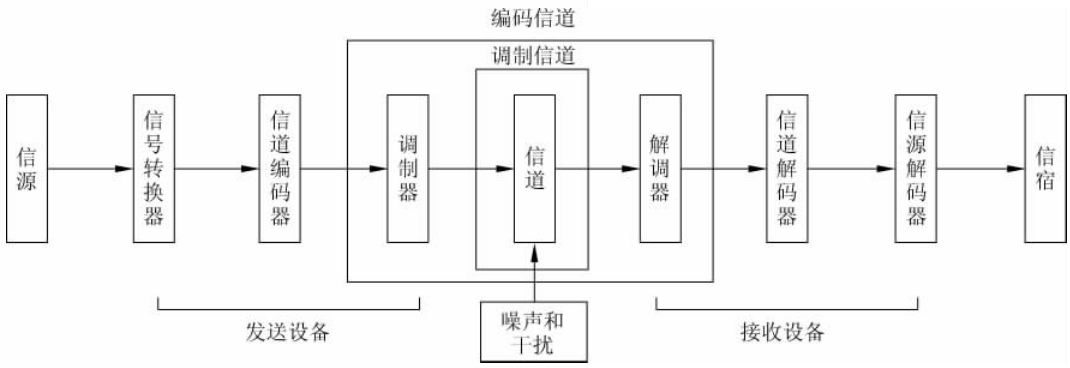

图 5-3 数字通信系统模型

注意:

(1) 图 5-3 中各个模块和模块功能在具体的系统中不一定全部采用。采用哪些模块和模块中哪些具体功能要取决于相应通信系统的具体设计要求。

(2) 发送端和接收端的模块是相互对应的。例如发送端使用了编码器,则接收端必须使用对应的译码器。

在发送端,信源输出的消息经过信源编码得到一个具有若干离散取值的离散时间序列。信源编码的功能为:将仿真信号转换为数字序列;压缩编码,提高通信效率;加密编码,提高信息传输安全性。

信源编码的输出序列将送入信道编码器,信道编码的功能为:

(1) 负责对数字序列进行差错控制编码,如分母编码、卷积编码、交织和扰乱等,以抵抗信道中的噪声和干扰,提高传输可靠性;

(2) 对差错控制编码输出的数字序列进行码型变换(也称为基带调制),如单双极性变换、归零-不归零码变换、差分编码、AMI 编码、HDB3 编码等,其目的是匹配信道传输特性,增加定时信息,改变输出符号的统计特性并使之具有一定的检错能力;

(3) 对输出码型进行波形映射,以适应于带限传输信道,如针对带限信道的无串扰波形成的成形滤波、部分响应成形滤波等。

调制器完成数字基带信号到频带信号的转换,数字调制方式有多种,如幅移键控(ASK)、相移键控(PSK)、频移键控(FSK)、正交幅度调制(QAM)、正交相移键控(QPSK)等,还可能包括扩频调制。调制器输出的频带信号经过功率放大后送入物理信道。

传输信号在物理信道中发生衰落,波形畸变,并混入噪声和干扰。

在接收端,接收信号经过滤波、变频、放大等信号调理后,送入解调器。解调器完成频带数字信号到基带数字信号的变换。

基带数字信号在信道译码器中完成译码，即完成与发送端信道编码器功能相反的变换，其输出的数字序列将送入信源译码器中进行译码，即完成解密、解压缩以及数/模转换等功能，最终向信宿输出接收消息。

在接收端，为了完成解调，通常需要提取发送的调制载波，而为了完成译码，必须使收发双方具有相同的传输节拍，也就是需要定时恢复，从而完成收发双方的同步，同步包括位同步和分组同步（帧同步和群同步）等。

如果数字通信系统中不使用调制器和解调器进行信号的基带-频带转换，则这样的系统称为数字基带传输系统。

5.2 MATLAB 通信仿真函数

MATLAB 通信系统工具箱中提供了许多与通信系统有关的函数命令，其中包括信源产生函数、信源编码/解码函数、信道模型函数、调制/解调函数、滤波器函数等，下面将对这些函数进行介绍。

5.2.1 信源产生函数

在 MATLAB 中，提供了 randerr、randint、randsrc 及 wgn 函数用于产生信源，下面分别对这几个函数进行简要介绍。

1. randerr 函数

该函数用于产生误比特图样。其调用格式为：

out=randerr(m)：产生一个 m×m 维的二进制矩阵，矩阵中的每一行有且只有一个非零元素，且非零元素在每一行中的位置是随机的。

out=randerr(m,n)：产生一个 m×n 维的二进制矩阵，矩阵中的每一行有且只有一个非零元素，且非零元素在每一行中的位置是随机的。

out=randerr(m,n,errors)：产生一个 m×n 维的二进制矩阵，参数 errors 可以是一个标量、行向量或只有两行的矩阵。

- 当 errors 为一标量时，产生的矩阵的每一行中 1 的个数等于 errors。
- 当 errors 为一行向量时，产生的矩阵的每一行中出现 1 的可能个数由 errors 的相应元素指定。
- 当 errors 为两行矩阵时，第一行指定出现 1 的可能个数，第二行说明出现 1 的概率，第二行中所有元素的和应该等于 1。

out=randerr(m,n,prob,state)：参数 prob 为 1 出现的概率；参数 state 为需要重新设置的状态。

out = randerr(m,n,prob,s)：使用随机流 s 创建一个二进制矩阵。

【例 5-1】 利用 randerr 不同调用格式创建一个二进制的误比特图样。

```
>> clear all;
>> out = randerr(8,7,[0 2])
```

```
out =
    0  1  0  0  0  1  0
    0  1  0  0  0  1  0
    0  0  0  0  0  0  0
    0  0  0  0  0  1  1
    0  0  0  0  0  0  0
    0  0  0  0  0  0  0
    0  0  1  0  0  0  1
    0  0  1  0  1  0  0
out2 = randerr(8,7,[0 2; .25 .75])
out2 =
    0  0  0  0  1  0  1
    0  1  0  0  0  0  1
    0  0  1  0  0  1  0
    0  1  0  0  1  0  0
    1  0  0  0  1  0  0
    0  0  0  0  0  0  0
    0  0  0  0  0  0  0
    0  0  0  0  0  0  0
```

2. randint 函数

该函数用于产生均匀分布的随机整数矩阵。其调用格式为：

out＝randint：产生一个不是 0 就是 1 的随机标量，且 0、1 等概率出现。

out＝randint(m)：产生一个 m×m 的整数矩阵，矩阵中的元素为等概率出现的 0 和 1。

out＝randint(m,n)：产生一个 m×n 的整数矩阵，矩阵中的元素为等概率出现的 0 和 1。

out＝randint(m,n,rg)：产生一个 m×n 的整数矩阵，如果 rg 为 0，则产生 0 矩阵；否则矩阵中的元素是 rg 所设定范围内整数的均匀分布。此范围为：

- [0,rg－1]，当 rg 为正整数时。
- [rg＋1,0]，当 rg 为负整数时。
- 从 min 到 max，包括 min 和 max，当 rg＝[min,max]或[max,min]时。

【例 5-2】 利用 randint 函数产生均匀分布的随机整数矩阵。

```
>> clear all;
>> out = randint(6,5,8)
out =
    3  1  2  5  3
    7  5  1  0  3
    4  2  5  4  3
    7  5  6  3  6
    5  5  2  7  2
    7  0  6  0  6
>> out = randint(6,5,[0,7])
out =
    3  2  2  5  5
    0  4  6  4  7
    1  1  1  3  1
    5  5  2  5  5
    3  1  0  5  1
    1  7  4  5  0
```

3. randsrc 函数

该函数是根据给定的数字表产生一个随机符号矩阵。矩阵中包含的元素是数据符号，它们之间相互独立。其调用格式为：

out＝randsrc：产生一个随机标量，这个标量是 1 或－1，且产生 1 和－1 的概率相等。

out＝randsrc(m)：产生一个 m×m 的矩阵，且此矩阵中的元素是等概率出现的 1 和－1。

out＝randsrc(m,n)：产生一个 m×n 的矩阵，且此矩阵中的元素是等概率出现的 1 和－1。

out＝randsrc(m,n,alphabet)：产生一个 m×n 的矩阵，矩阵中的元素为 alphabet 中所指定的数据符号，每个符号出现的概率相等且相互独立。

out＝randsrc(m,n,[alphabet; prob])：产生一个 m×n 的矩阵，矩阵中的元素为 alphabet 集合中所指定的数据符号，每个符号出现的概率由 prob 决定。prob 集合中所有数据相加必须等于 1。

【例 5-3】 利用 randsrc 函数产生一个随机符号矩阵。

```
>> clear all;
>> out = randsrc(7,10,[-3 -1 1 3])
out =
    -1   -1   -1    3    1   -3   3    3    3    1
    -3    3    1    1    3    1   1   -3   -3   -3
     3    3   -1    3    3   -1   3    3   -1    1
    -1   -1    3    1   -1   -3   1    1    3   -3
     3   -3    3   -3    1   -1   1    3    1    1
    -1    3   -3   -3   -3   -3   3    1    3    1
     3    3    3   -1   -3   -3   3   -1    1    1
>> out = randsrc(7,10,[-3 -1 1 3; .25 .25 .25 .25])
out =
     3   -3   -3   -3    1    3    3   -3    3   -3
     3    3    1   -3   -1   -1   -3    3    3    3
     3   -1   -1   -3    1   -3   -3    1    3   -1
     1    3   -3   -3   -1    1   -3    1    1   -1
     3   -3   -1    1   -3    1   -3   -3   -3   -1
     1    1   -3   -1   -1   -1    1    3   -1   -1
    -3    1   -3    1    3   -3    1    3   -3    1
```

4. wgn 函数

该函数用于产生高斯白噪声(White Gaussian Noise)。通过 wgn 函数可以产生实数形式或复数形式的噪声，噪声的功率单位可以是 dBW(分贝瓦)、dBm(分贝毫瓦)或绝对数值。其中，

$$1\text{W}=0\text{dBW}=30\text{dBm}$$

加性高斯白噪声是最简单的一种噪声，它表现为信号围绕平均值的一种随机波动过程。加性高斯白噪声的均值为 0，方差表现为噪声功率的大小。

wgn 函数的调用格式为：

y ＝ wgn(m,n,p)：产生 m 行 n 列的白噪声矩阵，p 表示输出信号 y 的功率(单位：dBW)，并且设定负载的电阻为 1Ω。

y = wgn(m,n,p,imp)：生成 m 行 n 列的白噪声矩阵，功率为 p，指定负载电阻 imp（单位：Ω）。

y = wgn(m,n,p,imp,state)：参数 state 为需要重新设置的状态。

y = wgn(…,powertype)：参数 powertype 指明了输出噪声信号功率 p 的单位，这些单位可以是 dBW、dBm 或 linear。

y = wgn(…,outputtype)：参数 outputtype 用于指定输出信号的类型。当 outputtype 被设置为 real 时，输出实信号；当设置为 complex 时，输出信号的实部和虚部的功率都为 p/2。

【例 5-4】 利用 wgn 函数产生高斯白噪声。

```
>> clear all;
>> y1 = wgn(10,1,0)
y1 =
    -0.1815
    -0.4269
     0.3801
     1.5804
    -0.6620
    -0.1699
     0.3929
    -2.0945
    -0.9653
    -0.0417
>> y1 = wgn(2,6,0)
y1 =
     0.4543    0.6344   -0.5145   -0.3616   0.3742   -0.7158
    -0.7841   -3.7003    0.3443    0.3838   0.9805    0.5870
```

5.2.2 信源编码/解码函数

在 MATLAB 中，提供了一些常用信源编码/解码函数，下面分别对这些函数进行介绍。

1. arithenco/arithdeco 函数

arithenco 函数用于实现算术二进制码编码。函数 arithdeco 用于实现算术二进制码解码。它们的调用格式为：

code=arithenco(seq,counts)：根据指定向量 seq 对应的符号序列产生二进制算术代码，向量 counts 代表信源中指定符号在数据集合中出现的次数统计。

dseq=arithdeco(code,counts,len)：解码二进制算术代码 code，恢复相应的 len 符号列。

【例 5-5】 利用 arithenco/arithdeco 函数实现算术二进制编码/解码。

```
>> clear all;
counts = [99 1];
len = 1000;
seq = randsrc(1,len,[1 2; .99 .01]);          % 随机序列
code = arithenco(seq,counts);                 % 编码
dseq = arithdeco(code,counts,length(seq));    % 解码
isequal(seq,dseq)                             % 检查 dseq 是否与原序列 seq 一致
```

运行程序，输出为：

```
ans =
    1
```

由以上结果可知，检查解码与编码的序列是一致的。当返回结果为 0 时，即表示不一致。

2. dpcmenco/dpcmdeco 函数

dpcmenco 函数用于实现差分码调制编码；dpcmdeco 函数用于实现差分码调制解码。它们的调用格式为：

indx=dpcmenco(sig,codebook,partition,predictor)：参数 sig 为输入信号，codebook 为预测误差量化码本，partition 为量化阈值，predictor 为预测期的预测传递函数系数向量，返回参数 indx 为量化序号。

[indx,quants]=dpcmenco(sig,codebook,partition,predictor)：返回参数 quants 为量化的预测误差。

sig=dpcmdeco(indx,codebook,predictor)：返回参数为输出信号，indx 为量化序号，codebook 为预测误差量化码本，partition 为量化阈值，predictor 为预测期的预测传递函数系数向量。

[sig,quanterror]=dpcmdeco(indx,codebook,predictor)：参数 quanterror 为量化的预测误差。

【例 5-6】 用训练数据优化 DPCM 方法，对一个锯齿波信号数据进行预测量化。

```
>> clear all;
t = [0:pi/60:2 * pi];
x = sawtooth(3 * t);                                    % 原始信号
initcodebook = [ - 1:.1:1];                             % 初始化高斯噪声
% 优化参数,使用初始序列 initcodebook
[predictor,codebook,partition] = dpcmopt(x,1,initcodebook);
% 使用 DPCM 量化 X
encodedx = dpcmenco(x,codebook,partition,predictor);
% 尝试从调制信号中恢复 X
[decodedx,equant] = dpcmdeco(encodedx,codebook,predictor);
distor = sum((x - decodedx).^2)/length(x)               % 均方误差
plot(t,x,t,equant,'*');
```

运行程序，输出如下，得到预测量化误差，如图 5-4 所示。

```
distor =
    8.1282e - 04
```

3. compand 函数

该函数按 Mu 律或 A 律对输入信号进行扩展或压缩。其调用格式为：

```
out = compand(in,param,v)
out = compand(in,Mu,v,method)
```

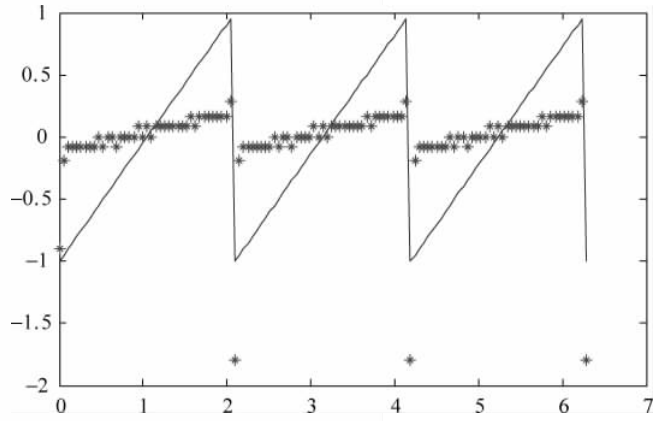

图 5-4　DPCM 预测量化误差图

参数 param 指出 Mu 或 A 的值，v 为输入信号的最大幅值，method 决定具体采用哪种方式进行扩展或压缩。

out＝compand(in，param，v)：参数 param 指出 Mu 或 A 的值，v 为输入信号的最大幅值。

out＝compand(in，Mu，v，'mu/compressor')：利用 Mu 律对信号进行压缩。

out＝compand(in，Mu，v，'mu/expander')：利用 Mu 律对信号进行扩展。

out＝compand(in，A，v，'A/compressor')：利用 A 律对信号进行压缩。

out＝compand(in，A，v，'A/expander')：利用 A 律对信号进行扩展。

【例 5-7】 利用 compand 函数对给定输入信号使用 A 律的 compressors 及 expanders 方法进行压缩与扩展。

```
>> clear all;
>> compressed = compand(1:5,87.6,5,'a/compressor') % 压缩
expanded = compand(compressed,87.6,5,'a/expander') % 扩展
```

运行程序，输出如下：

```
compressed =
    3.5296   4.1629   4.5333   4.7961   5.0000
expanded =
    1.0000   2.0000   3.0000   4.0000   5.0000
```

4. lloyds 函数

该函数能够优化标量量化的阈值和码本。它使用 Lloyds_max 算法优化标量量化参数，用给定的训练序列向量优化初始码本，使量化误差小于给定的容差。其调用格式为：

[partition，codebook]＝lloyds(training_set，initcodebook)：参数 training_set 为给定的训练序列，initcodebook 为码本的初始预测值。

[partition，codebook]＝lloyds(training_set，len)：len 为给定的预测长度。

[partition，codebook]＝lloyds(training_set，…，tol)：tol 为给定容差。

[partition，codebook，distor]＝lloyds(…)：返回最终的均方差 distor。

[partition，codebook，distor，reldistor]＝ lloyds(…)：返回是有关算法的终止值 reldistor。

【例 5-8】 通过一个 2bit 通道优化正弦传输量化参数。

```
>> clear all;
>> % 产生正弦信号的一个完整周期
>> x = sin([0:1000] * pi/500);
>> [partition,codebook,distor,reldistor] = lloyds(x,2^2)
```

运行程序，输出如下：

```
partition =
    -0.5715    0.0037   0.5761
codebook =
    -0.8520   -0.2910   0.2984   0.8539
distor =
     0.0210
reldistor =
    0
```

5. quantiz 函数

该函数用于产生一个量化序号和输出量化值。其调用格式为：

index=quantiz(sig,partition)：根据判断向量 partition，对输入信号 sig 产生量化索引 index，indx 的长度与 sig 向量的长度相同。

[index,quants]=quantiz(sig,partition,codebook)：根据给定的向量 partition 及码本 codebook，对输入信号 sig 产生一个量化序号 index 和输出量化误差 quants。

[index,quants,distor]=quantiz(sig,partition,codebook)：参数 distor 为量化的预测误差。

【例 5-9】 用训练序列和 lloyd 算法，对一个正弦信号数据进行标量量化。

```
>> clear all;
N = 2^4;                                          % 以 4bit 传输信道
t = [0:100] * pi/20;
u = sin(t);
[p,c] = lloyds(u,N);                              % 生成分界点向量和编码手册
[index,quant,distor] = quantiz(u,p,c);            % 量化信号
plot(t,u,t,quant,'+');
```

运行程序，效果如图 5-5 所示。

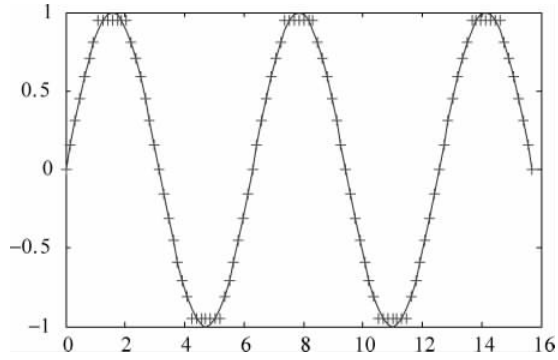

图 5-5 标量量化误差图

5.2.3 信道函数

对于最常用的两种信道,高斯白噪声信道和二进制对称信道,MATLAB 为其提供了对应的函数,下面分别进行介绍。

1. awgn 函数

该函数在输入信号中叠加一定强度的高斯白噪声,噪声的强度由函数参数确定。其调用格式为:

y=awgn(x,SNR):在信号 x 中加入高斯白噪声。信噪比 SNR 以 dB 为单位,x 的强度假定为 0dBW。如果 x 是复数,就加入复噪声。

y=awgn(x,SNR,SIGPOWER):如果 SIGPOWER 是数值,则其代表以 dBW 为单位的信号强度;如果 SIGPOWER 为'measured',则函数将在加入噪声之前测定信号强度。

y=awgn(x,SNR,SIGPOWER,STATE):重置 RANDN 的状态。

y=awgn(…,POWERTYPE):指定 SNR 和 SIGPOWER 的单位。POWERTYPE 可以是'dB'或'linear'。如果 POWERTYPE 是'dB',那么 SNR 以 dB 为单位,而 SIGPOWER 以 dBW 为单位。如果 POWERTYPE 是'linear',那么 SNR 作为比值来度量,而 SIGPOWER 以瓦特(W)为单位。

【例 5-10】 对输入的锯齿波进行叠加高斯白噪声。

```
>> clear all;
t = 0:.1:10;
x = sawtooth(t);                          % 产生锯齿波信号
y = awgn(x,10,'measured');                % 添加高斯白噪声
plot(t,x,t,y)                             % 绘制原信号和输出信号
legend('原始信号','叠加高斯白噪声信号');
```

运行程序,效果如图 5-6 所示。

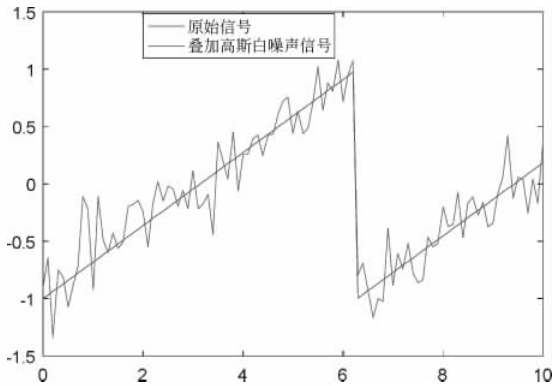

图 5-6 高斯白噪声信号

2. bsc 函数

该函数通过二进制对称信道以误码概率 p 传输二进制输入信号。该函数的调用格式为：

ndata＝bsc(data,p)：给定输入信号 data 及误码概率，返回二进制对称信道误码率。

ndata＝bsc(data,p,s)：参数 s 为一个任意的有效随机流。

ndata＝bsc(data,p,state)：参数 state 指定状态。

[ndata,err]＝bsc(…)：err 指定返回的误差。

【例 5-11】 对输入的二进制信号，进行对称信道后再利用 biterr 函数计算误比特率。

```
>> clear all;
z = randi([0 1],100,100);                    % 随机矩阵
nz = bsc(z,.15);                             % 二进制对称信道
[numerrs,pcterrs] = biterr(z,nz)             % 计算误比特率
```

运行程序，输出如下：

```
numerrs =
        1509
pcterrs =
    0.1509
```

由结果可得，错误码数为 1509，误码率为 0.1509。

5.3 信号与信道

在前面简单介绍了几种产生信号与信道的方法，本节将进一步讲述 MATLAB 及 Simulink 中具备产生信号与信道的函数及模块。

5.3.1 随机数据信号源

本节将介绍几种数字信号产生器，包括伯努利二进制信号产生器、泊松分布整数产生器以及随机整数产生器等。

1. 伯努利二进制产生器

1) MATLAB 函数

将试验 E 重复进行 n 次，若各次试验的结果互不影响，即每次试验结果出现的概率都不依赖于其他各次试验的结果，则称这 n 次试验是相互独立的。

设试验 E 只有两个可能结果 A 及 $\overline{A}$，$P(A)=p$，$P(\overline{A})=1-p=q(0<p<1)$。将 E 独立重复地进行 n 次，则称这一串重复的独立试验为 n 重伯努利试验，简称伯努利试验。伯努利试验是一种很重要的数学模型。它有广泛的应用，是研究最多的模型之一。

以 X 表示 n 重伯努利试验中事件 A 发生的次数，X 是一个随机变量，我们来求它的分布律。X 所有可能取的值为 0,1,2,…,n。由于各次试验是相互独立的，因此，事件 A

在指定的 $k(0\leqslant k\leqslant n)$ 次试验中发生，其他 $n-k$ 次试验中不发生(例如在前 k 次试验中发生，而后 $n-k$ 次试验中不发生)的概率为：

$$\underbrace{p\cdot p\cdots p}_{k\text{个}}\cdot\underbrace{(1-p)\cdot(1-p)\cdots(1-p)}_{n-k\text{个}}=p^k(1-p)^{n-k}$$

由于这种指定的方式共有 C_n^k 种，它们是两两互不相容的，故在 n 次试验中 A 发生 k 次的概率为 $C_n^kP^k(1-p)^{n-k}$，即：

$$P\{X=k\}=C_n^kp^kq^{n-k},\quad k=0,1,2,\cdots,n \tag{5-1}$$

显然：

$$\begin{cases}P\{X=k\}\geqslant 0,\quad k=0,1,2,\cdots,n\\ \sum\limits_{k=0}^{n}C_n^kp^kq^{n-k}=(p+q)^n=1\end{cases} \tag{5-2}$$

即 $P\{X=k\}$ 满足条件式(5-1)、式(5-2)。注意到 $C_n^kp^kq^{n-k}$ 刚好是二项式 $(p+q)^n$ 的展开式中出现 p^k 的一项，故称随机变量 X 服从参数为 n,p 的二项分布，记为：

$$X\sim B(n,p)$$

特别地，当 $n=1$ 时二项分布化为：

$$P\{X=k\}=p^kq^{1-k},\quad k=0,1$$

这就是 0-1 分布。

MATLAB 统计工具箱提供了伯努利二进制的计算函数，包括 binopdf、binocdf、binofit、binoinv、binornd、binostat 等。

【例 5-12】 某人向空中抛硬币 100 次，落下为正面的概率为 0.5，求这 100 次中正面向上次数的概率。

```
>> clear all;
p1 = binopdf(45,100,0.5)                          % 计算 x = 45 的概率
p2 = binocdf(45,100,0.5)                          % 计算 x≤45 的概率即累积概率
x = 1:100;
p = binopdf(x,100,0.5);
px = binopdf(x,100,0.5);
subplot(121);plot(x,p,'rp');                      % 绘制分布函数图像
xlabel('x'); ylabel('p');title('分布函数');
axis square;
subplot(122);plot(x,px,' + ');                    % 绘制概率密度函数图像
xlabel('x'); ylabel('p');title('概率密度函数');
axis square;
```

运行程序，输出如下，效果如图 5-7 所示。

```
p1 =
    0.0485
p2 =
    0.1841
```

2) Simulink 模块

伯努利二进制信号的产生器符合伯努利分布的随机信号。

伯努利二进制信号产生器产生随机二进制序列，并且在这个二进制序列中的 0 和 1 满足伯努利分布，如式(5-3)所示。

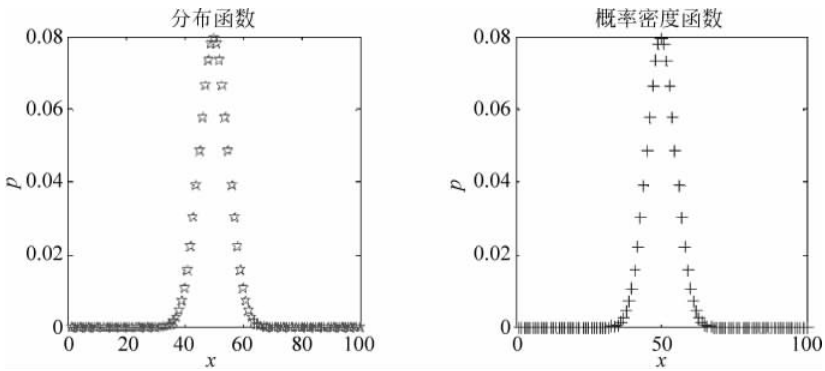

图 5-7　伯努利二进制分布函数及密度函数效果图

$$\Pr(x)=\begin{cases} p, & x=0 \\ 1-p, & x=1 \end{cases} \tag{5-3}$$

即伯努利二进制信号产生器产生的序列中，产生 0 的概率为 p，产生 1 的概率为 $1-p$。根据伯努利序列的性质可知，输出信号的均值为伯努利 $1-p$，方差为 $p(1-p)$。产生 0 的概率 p 由伯努利二进制信号产生器中的 Probability of zero 项控制，它可以是 0 和 1 之间的某个实数。

伯努利二进制信号产生器的输出信号，可以是基于帧的矩阵、基于采样的行或列向量，或者基于采样的一维序列。输出信号的性质可以由二进制伯努利序列产生器中的 Frame-based outputs、Samples per frame 和 Interpret vector parameters as 1-D 三个选项控制。

伯努利二进制信号产生器模块及参数设置对话框如图 5-8 所示。

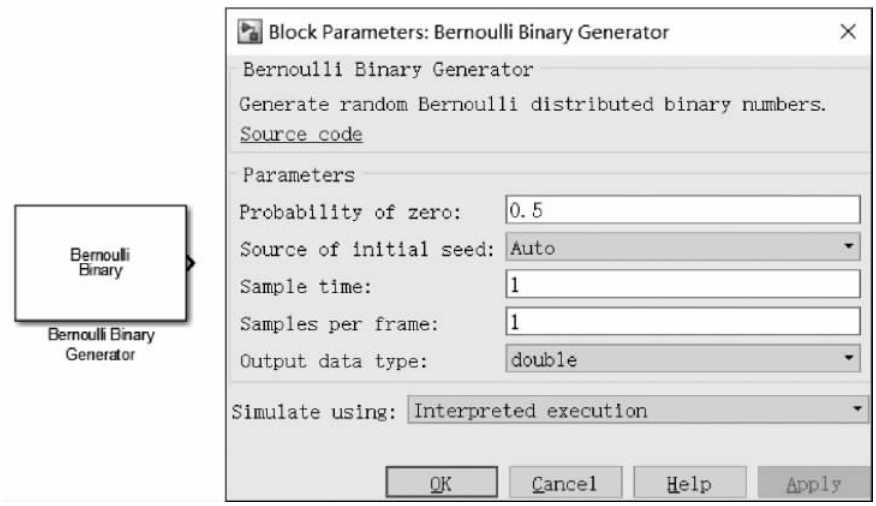

图 5-8　伯努利二进制信号产生器模块及参数设置对话框

伯努利二进制信号产生器中包含多个参数项，下面分别对各项进行简单的介绍。

Probability of zero：伯努利二进制信号产生器输出 0 的概率，对应于式(5-3)中的 p，

p 为 0 和 1 之间的实数。

Source of Initial seed：伯努利二进制信号产生器的随机数种子，它可以是与 Probability of a zero 项长度相同的向量或标量。当使用相同的随机数种子时，伯努利二进制信号产生器每次都会产生相同的二进制序列；不同的随机数种子通常产生不同的序列。当随机数种子的维数大于 1 时，伯努利二进制信号产生器的输出信号的维数也大于 1。

Sample time：输出序列中每个二进制符号的持续时间。

Samples per Frame：指定伯努利二进制信号产生器每帧采样。

Output data type：决定模块输出的数据类型，可以是 boolean、uint8、uint16、uint32、single、double 等众多类型，默认为 double。

Simulate using：指定使用仿真的方式。

2. 泊松分布整数产生器

1) MATLAB 函数

如果离散随机变量 ξ 的取值为非负整数值 $k=0,1,2,\ldots$，且取值等于 k 的概率为：

$$p_k = P(\xi = k) = \frac{\lambda^k}{k!}\exp(-\lambda)$$

则称离散随机变量 ξ 服从泊松分布。泊松分布随机变量的期望和均值为：

$$E(\xi) = \lambda$$

$$\mathrm{Var}(\xi) = \lambda$$

两个分别服从参数为 λ_1 和 λ_2 的独立泊松分布的随机变量之和也是泊松分布的，其参数为 $\lambda_1+\lambda_2$。

在对二项分布的概率计算中，需要计算组合数，这在独立试验次数很多的情况下是不方便的。泊松定理指出，当一次试验的事件概率很小 $p\to 0$，独立试验次数很大 $n\to\infty$，而两者之乘积 $np=\lambda$ 为有限值时，二项分布 $P_k(n,p)$ 趋近于参数为 λ 的泊松分布，即有 $\lim\limits_{n\to\infty}P_k(n,p)=\frac{\lambda^k}{k!}\mathrm{e}^{-\lambda}$。利用泊松分布可以对单次事件概率很小而独立试验次数很大的二项分布概率进行有效的建模及近似计算。

如果产生一系列参数同为 λ 的指数分布的随机数 $t_i(i=1,2,\cdots)$，可认为在时间段 $\sum\limits_{i=1}^{k}t_i$ 上发生了 k 个事件，因此在单位时间段 $t=1$ 上发生的事件数 k 满足方程：

$$\sum_{i=1}^{k}t_i \leqslant 1 < \sum_{i=1}^{k+1}t_i \tag{5-4}$$

利用这一关系即可产生参数为 λ 的泊松分布随机数，即不断产生参数为 λ 的指数分布的随机数 t_i，$i=1,2,\cdots$，并将它们累加起来，如果累加到 $k+1$ 个的结果大于 1，则将计数值 k 作为泊松分布的随机数输出。

设随机数 x_i 是均匀分布在区间[0,1]上的随机数，则根据前述反函数法，$t_i=-\frac{1}{\lambda}\ln x_i$

将是参数为λ的指数分布随机数。将其代入式(5-4)可得：

$$\sum_{i=1}^{k}-\frac{1}{\lambda}\ln x_i \leqslant 1 < \sum_{i=1}^{k+1}-\frac{1}{\lambda}\ln x_i \tag{5-5}$$

利用式(5-5)计算时需要计算对数求和，效率较低。实际上，式(5-5)可简化为：

$$\prod_{i=1}^{k} x_i \geqslant \exp(-\lambda) > \prod_{i=1}^{k+1} x_i \tag{5-6}$$

这样，泊松随机数的产生就简化为连乘运算和条件判断，具体算法如下：

(1) 初始化：置计数器 i：=0，以及乘积变量 v：=1；

(2) 计算连乘：产生一个区间[0,1]上均匀分布的随机数 x_i，并赋值 v：$v\times x_i$；

(3) 判断：如果 $v\geqslant\exp(\lambda)$，则令 i：=i+1，返回步骤(2)；否则，将当前计数值作为泊松随机数输出，然后转到步骤(1)。

MATLAB统计工具箱提供的泊松分布计算指令包括 poisspdf、poisscdf、poissfit、poissinv、poissrnd、poisstats 等。

【例 5-13】 生成泊松分布的随机数。

```
>> clear all;
%设置泊松分布的参数
lambda = 4;
%产生 len 个随机数
len = 5;
y1 = poissrnd(lambda,[1 len])
%产生 P 行 Q 列的矩阵
P = 3;
Q = 4;
y2 = poissrnd(lambda,P,Q)
%显示泊松分布的柱状图
M = 1000;
y3 = poissrnd(lambda,[1 M]);
figure(1);
t = 0:1:max(y3);
hist(y3,t);
axis([0 max(y3) 0 250]);
xlabel('取值');
ylabel('计数值');
```

运行程序，输出如下，效果如图 5-9 所示。

```
y1 =
    5   4   4   2   4
y2 =
    3   5   6   7
    6   5   6   3
    4   6   3   3
```

2) Simulink 模块

泊松分布整数产生器产生服从泊松分布的整数序列。

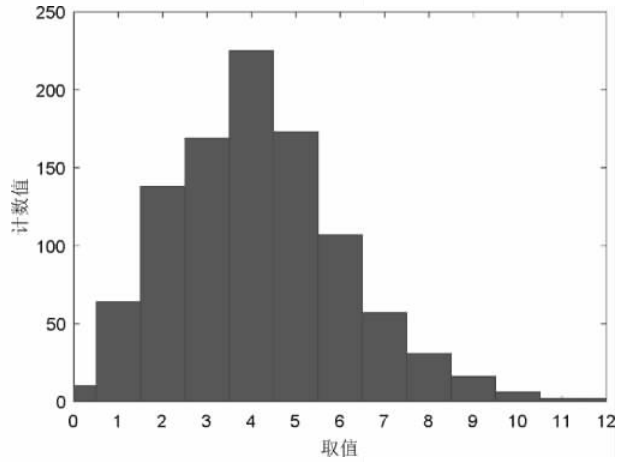

图 5-9　泊松分布频率直方图

泊松分布整数产生器利用泊松分布产生随机整数。假设 x 是一个服从泊松分布的随机变量,那么 x 等于非负整数 k 的概率可以用式(5-7)表示。

$$\Pr(k)=\frac{\lambda^k e^{-k}}{k!},\quad k=0,1,2,\cdots \tag{5-7}$$

其中,λ 为一正数,称为泊松参数,并且泊松随机过程的均值和方差都等于 λ。

利用泊松分布整数产生器可以在双传输通道中产生噪声,在这种情况下,泊松参数 λ 应该比 1 小,通常远小于 1。泊松分布参数产生器的输出信号,可以是基于帧的矩阵、基于采样的行或列向量,也可以是基于采样的一维序列。输出信号的性质可以由泊松分布整数产生器中的 Frame-based outputs、Samples per frame 和 Interpret vector parameters as 1-D 三个选项控制。

泊松分布整数产生器模块及参数设置对话框如图 5-10 所示。

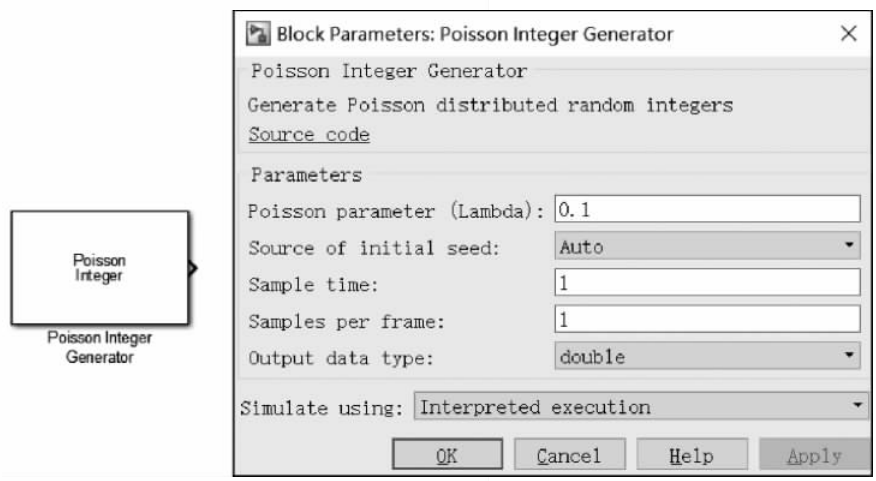

图 5-10　泊松分布整数产生器模块及参数设置对话框

泊松分布整数产生器对话框中包含多个参数项,下面分别对各项进行简单的说明。

Lambda:确定泊松参数 λ,如果输入为一个标量,那么输出向量的每一个元素分享相

同的泊松参数。

Source of Initial seed：泊松分布整数产生器的随机数种子。当使用相同的随机数种子时，泊松分布整数产生器每次都会产生相同的二进制序列；不同的随机数种子通常产生不同的序列。当随机数种子的维数大于 1 时，泊松分布参数产生器的输出信号的维数也大于 1。

Sample time：输出序列中每个整数的持续时间。

Samples per frame：该参数用来确定每帧的采样点的数目。本项只有当 Frame-based outputs 项选中后才有效。

Output data type：决定模块输出的数据类型，可以是 boolean、uint8、int16、uint16、single、double 等众多类型，默认为 double。

Simulate using：指定使用仿真的方式。

3. 随机整数产生器

随机整数产生器用来产生[0，M－1]范围内具有均匀分布的随机整数。

随机整数产生器输出整数的范围[0，M－1]可以由用户自己定义。M 的大小可随机整数产生器中的 M-ary number 项中随机输入。M 可以是标量也可以是向量。如果 M 为标量，那么输出均匀分布且互不相关的随机变量。如果 M 为向量，其长度必须和随机整数产生器中 Source of Initial seed 的长度相同，在这种情况下，每一个输出对应一个独立的输出范围。如果 Source of Initial seed 是一个常数，那么产生的噪声是周期重复的。

随机整数产生器的输出信号，可以是基于帧的矩阵、基于采样的行或列向量，也可以是基于采样的一维序列。输出信号的性质可以由 Sample time、Samples per frame 和 Output data type 三个选项控制。

随机整数产生器模块及参数设置对话框如图 5-11 所示。

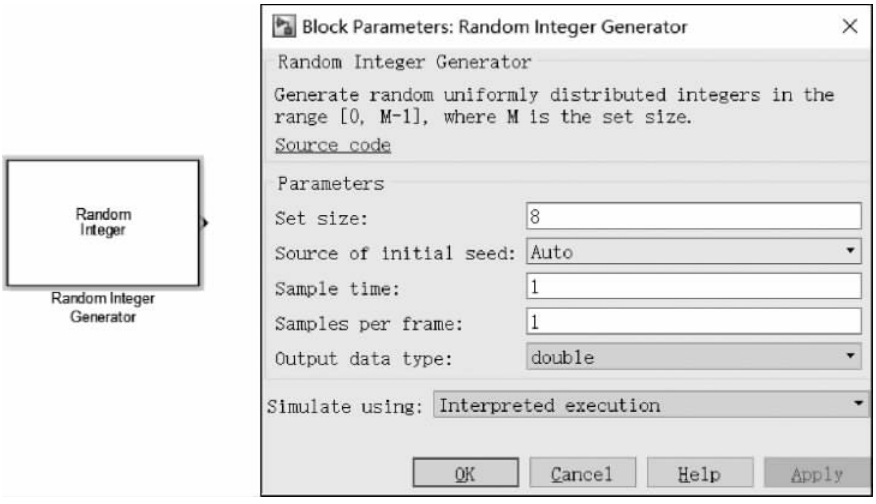

图 5-11　随机整数产生器模块及参数设置对话框

随机整数产生器对话框包含多个参数项,下面分别对各项进行简单的介绍。

Set size:输入正整数或正整数矢量,设定随机整数的大小。

Source of Initial seed:随机整数产生器的随机种子。当使用相同的随机数种子时,随机整数产生器每次都会产生相同的二进制序列;不同的随机数种子通常产生不能的序列。当随机数种子的维数大于1时,随机整数产生器的输出信号的维数也大于1。

Sample time:输出序列中每个整数的持续时间。

Sample of Frame:指定随机整数产生器每帧采样。

Sample per frame:该参数用来确定每帧的采样点的数目。本项只有当Frame-based outputs选项中后有效。

Output data type:决定模块输出的数据类型,可以是boolean、uint8、uint16、uint32、single、double等众多类型,默认为double。如果想要输出为boolean型,M-ary number项必须为2。

5.3.2 序列产生器

序列产生器用来产生一个具有某种特性的二进制序列,这种序列可能有比较独特的外相关属性或互相关属性。

1. PN序列产生器

PN序列产生器用于产生一个伪随机序列。

PN序列产生器利用线性反馈移位寄存器(LFSR)来产生PN序列。线性反馈移位寄存器可以通过简单的移位暂存器产生器结构得到实现。

PN序列产生器中共有 r 个寄存器,每个寄存器都以相同的采样频率更新寄存器的状态,即第 k 个寄存器在 $t+1$ 时刻的状态 m_k^{t+1} 等于第 $k+1$ 个寄存器在 t 时刻的状态 m_{k+1}^t。PN序列产生器可以用一个生成的多项式表示:

$$g_r z^r + g_{r-1} z^{r-1} + g_{r-2} z^{r-2} + \cdots + g_1 z + g_0$$

Simulink提供了PN序列产生器模块,其模块及参数设置对话框如图5-12所示。

PN序列产生器中包含多个参数项,下面分别对各项进行简要介绍。

Sample time:输出序列中每个元素的持续时间。

Frame-based outputs:指定PN序列产生器以帧格式产生输出序列。

Sample per frame:该参数用来确定每帧的采样点的数目。本项只有当Frame-based outputs项选中后有效。

Output variable-size signals:选择该项后即设定输入单变量的范围。

Maximum output size:设定输出数据的大小,在Output variable-size signals项选中时有效。

Reset on nonzero input:选择该项之后,PN序列产生器提供一个输入端口,用于输入复位信号。如果输入不为0,PN序列产生器会将各个寄存器恢复到初始状态。

Enable bit-packed outputs:选定后激活Number of packed bits、Interpret bit-packed values as signed两项。

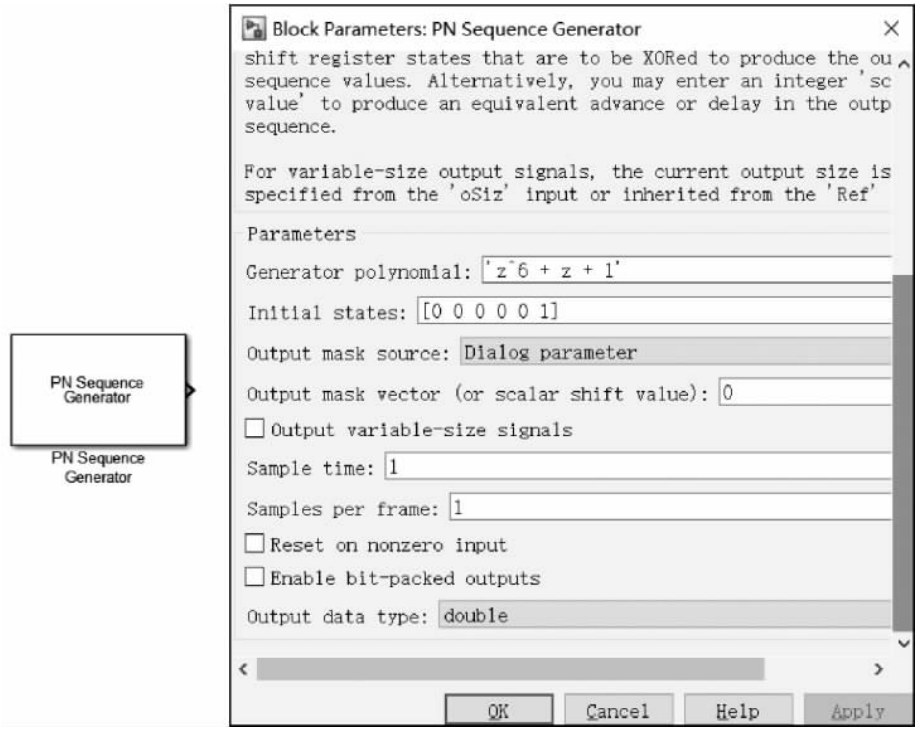

图 5-12　PN 序列产生器模块及参数设置对话框

Number of packed bits：设定输出字符的位数(1～32)。

Interpret bit-packed values as signed：有符号整数与无符号整数判断项。如果该项被选定，最高位为 1 时，表示为负。

Output data type：决定模块输出的数据类型，默认为 double。

Output mask source：选择模块中的输出屏蔽信息的给定方式。此项为复选框。如果选定 Dialog parameter，则可在 Output mask vector(or scalar shift value)项中输入；如果选定 Input port，则需要在弹出的对话框中输入。

Output mask vector(or scalar shift value)：给定输出屏蔽(或移位量)。输入的整数或二进制向量决定了生成的 PN 序列相对于初始时刻的延时。如果移位限定为二进制向量，那么向量的长度必须和生成多项式的次数相同。此项只有在 Output mask source 选定为 Dialog parameter 时有效。

2. Gold 序列产生器

Gold 序列产生器用来产生 Gold 序列。Gold 序列的一个重要的特性是其具有良好的互相关性。Gold 序列产生器根据两个长度为 $N=2^n-1$ 的序列 u 和 v 产生一个 Gold 序列 $G(u,v)$，序列 u 和 v 称为一个“优选对”。但是想要成为“优选对”进而产生 Gold 序列，长度 $N=2^n-1$ 的序列 u 和 v 必须满足以下几个条件：

(1) n 不能被 4 整除；

(2) $v=u[q]$，即序列 v 是通过对序列 u 每隔 q 个元素进行一次采样得到的序列，其

中 q 是奇数，$q=2^k+1$ 或 $q=2^{2k}-2^k+1$；

(3) n 和 k 的最大公约数满足条件：$\gcd(n,k)=\begin{cases}1, & n\equiv 1 \bmod 2\\ 2, & n\equiv 2 \bmod 4\end{cases}$。

由“优选对”序列 u 和 v 产生的 Gold 序列 $G(u,v)$ 可表示为：

$$G(u,v)=\{u,v,u\oplus v,u\oplus Tv,u\oplus T^2v,\cdots,u\oplus T^{N-1}v\}$$

其中，$T^n x$ 表示将序列 x 以循环移位的方式向左移 n 位。⊕代表模二加。值得注意的是，由于长度 N 的两个序列 u 和 v 产生的 Gold 序列 $G(u,v)$ 中包含了 $N+2$ 个长度为 N 的序列，Gold 序列产生器可根据设定的参数输出其中的某一个序列。

如果有两个 Gold 序列 X、Y 属于同一个集合 $G(u,v)$，并且长度 $N=2^n-1$，那么这两个序列的互相关函数只能有三种可能：$-t(n)$、-1、$t(n)-2$。其中，

$$t(n)=\begin{cases}1+2^{(n+1)/2}, & n\text{ 为偶数}\\ 1+2^{(n+2)/2}, & n\text{ 为奇数}\end{cases}$$

Gold 序列实际上是把两个长度相同的 PN 序列产生器产生的“优选对”序列进行异或运算后得到的序列，如图 5-13 所示。

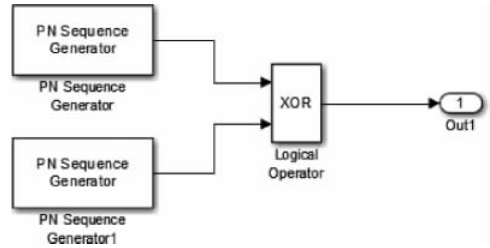

图 5-13　Gold 序列产生器结构图

Gold 序列产生器模块及参数设置对话框如图 5-14 所示。

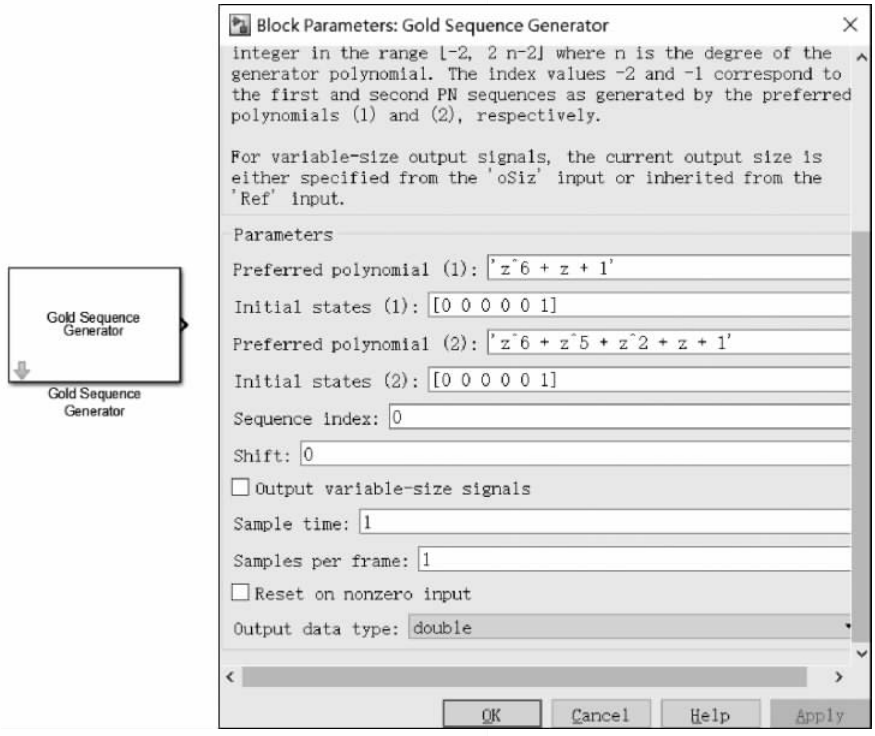

图 5-14　Gold 码发生器模块及其参数设置对话框

Gold 码发生器对话框中包含多个参数项，下面分别对各项进行简单的介绍。

Preferred polynomial(1)："优选对"序列 1 的生成多项式，可以是二进制向量的形式，也可以是由多项式下标构成的整数向量。

Initial states(1)："优选对"序列 1 的初始状态。它是一个二进制向量，用于表明与优选对序列 1 对应的 PN 序列产生器中每个寄存器的初始状态。

Preferred polynomial(2)："优选对"序列 2 的生成多项式，可以是二进制向量的形式，也可以是由多项式下标构成的整数向量。

Initial states(2)："优选对"序列 2 的初始状态。它是一个二进制向量，用于表明与优选对序列 2 对应的 PN 序列产生器中每个寄存器的初始状态。

Sequence index：用于限定 Gold 序列 $G(u,v)$的输出，其范围是$[-2,-1,0,1,2,\cdots,2^n-2]$。

Shift：指定 Gold 码发生器的输出序列的时延。该参数是一个整数，表示序列延时 Shift 个采样周期后输出。

Sample time：输出序列中每个元素的持续时间。

Frame-based outputs：指定 Gold 码发生器是否是以帧格式产生输出序列。

Sample per frame：该参数用来确定每帧的采样点数目。本项只有当 Frame-based outputs 项被选中后有效。

Output variable-size signals：选择该项后即设定输入单变量的范围。

Maximum output size：设定输出数据的大小，在 Output variable-size signals 项选中时有效。

Reset on nonzero input：选择该项之后，Gold 码发生器提供一个输入端口，用于输入复位信号。如果输入不为 0，Gold 码发生器会将各个寄存器恢复到初始状态。

Output data type：决定模块输出的数据类型，可以是 boolean、double、Smallest、unsigned、integer 等类型，默认为 double。

3. Walsh 序列产生器

Walsh 序列产生器产生一个 Walsh 序列。

如果用 W_i 表示第 i 个长度为 N 的 Walsh 序列，其中 $i=0,1,\cdots,N-1$，并且 Walsh 序列的元素是+1 或−1，$W_i[k]$表示 Walsh 序列 W_i 的第 k 个元素，那么对于任意的 i，$W_i[0]=0$。对于任意两个长度为 N 的 Walsh 序列 W_i 和 W_j，有 $W_iW_j^{\mathrm{T}}=\begin{cases}0, & i\neq j\\ N, & i=j\end{cases}$。

在 Simulink 中提供对应的模块用于实现 Walsh 序列产生，该模块及参数设置对话框如图 5-15 所示。

Walsh 码发生器中包含多个参数项，下面分别对各项进行简单的说明。

Code length：设定输出 Walsh 码的长度 N，且需满足 $N=2^n$，$n=0,1,2,\cdots$。

Code index：Walsh 码的序号，为[0，N−1]范围内的整数，表示序列中过零点的数目。

Sample time：输出序列中每个元素的持续时间。

Frame-based outputs：指定 Walsh 码发生器是否是以帧格式产生输出序列。

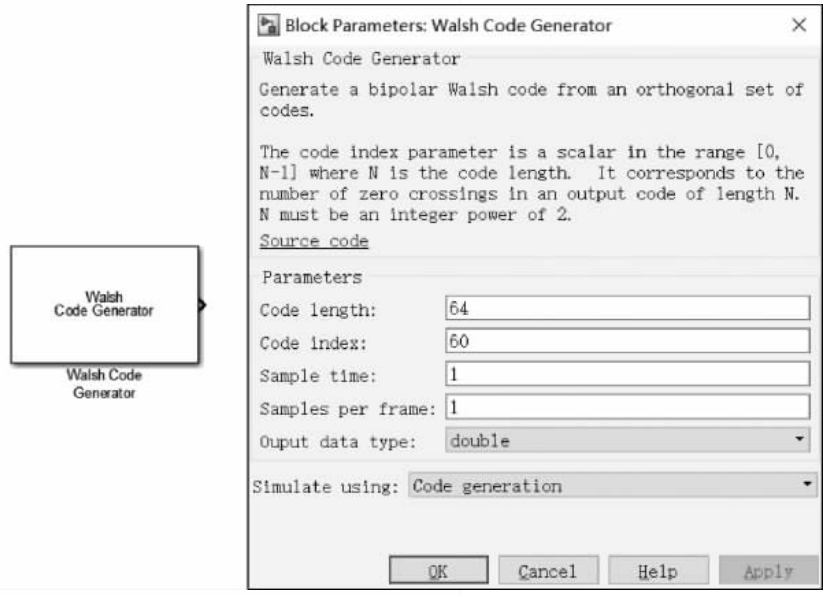

图 5-15　Walsh 序列产生器模块及参数设置对话框

Sample per frame：该参数用来确定每帧的采样点数目。本项只有当 Frame-based outputs 项被选中后有效。

Output data type：决定模块输出的数据类型，可以是 double、int8 类型，默认为 double。

Simulate using：指定使用仿真的方式。

5.3.3　噪声源发生器

噪声的存在，对通信系统具有很大的影响。但噪声本身是一个随机过程，很难通过一种简单的方法预测某个时刻噪声信号的强度。下面对 MATLAB 本身提供的几种具有不同的随机特征噪声产生器进行简要介绍。

1. 均匀分布随机噪声产生器

设连续型随机变量 X 具有概率密度：

$$f(x)=\begin{cases}\dfrac{1}{b-a}, & a<x<b\\ 0, & \text{其他}\end{cases} \tag{5-8}$$

则称 X 在区间(a,b)上服从均匀分布，记为 $X\sim U(a,b)$，其中 a,b 是分布参数。

在区间(a,b)上服从均匀分布的随机变量 X，具有下述意义的等可能性，即它落在区间(a,b)中任意等长度的子区间内的可能性是相同的，或者说它落在子区间的概率只依赖于子区间的长度而与子区间的位置无关。实际上，对于任一长度 l 的子区间$(c,c+l)$，$a\leqslant c<c+l\leqslant b$，有：

$$P\{c<X\leqslant c+l\}=\int_c^{c+l}f(x)\mathrm{d}x=\int_c^{c+l}\frac{1}{b-a}\mathrm{d}x=\frac{1}{b-a} \tag{5-9}$$

由式(5-8)和式(5-9)得 X 的分布函数为：

$$F(x)=\begin{cases}0, & x<a\\ \dfrac{x-a}{b-a}, & a\leqslant x<b\\ 1, & x\geqslant b\end{cases} \tag{5-10}$$

在 MATLAB 中提供了 unifrnd 函数创建均匀分布。

【例 5-14】 (投掷硬币的计算机模拟)投掷硬币 1000 次,试模拟掷硬币的结果。

```
>> clear all;
n = 1000;
t1 = 0; t2 = 0; a = [];
for j = 1:n
    a(j) = unifrnd(0,1);
    if a(j)< 0.5
        t1 = t1 + 1;
    else
        t2 = t2 + 1;
    end
end
p1 = t1/n
p2 = t2/n
```

运行程序,输出如下：

```
p1 =
    0.4910
p2 =
    0.5090
```

说明：当再次运行程序时,结果与上面的不一定相同,因为这相当于又做了一次投掷硬币 1000 次的实验。当程序中 n=1000 改为 n=100 000 时,就相当于投掷硬币 100 000 次的实验。

2. 高斯随机噪声产生器

正态分布也称高斯分布,可采用函数变换法产生标准正态分布随机数。设 r_1 和 r_2 是两个独立的在区间[0,1]上均匀分布的随机数,则：

$$r_1=\sqrt{-2\ln r_1}\cos 2\pi r_2$$

$$r_2=\sqrt{-2\ln r_1}\sin 2\pi r_2$$

是两个独立同分布的标准高斯随机数,即其均值为零,方差为 1,记为 $x_1\sim N(0,1)$ 和 $x_2\sim N(0,1)$。MATLAB 中用函数 randn 产生标准正态分布随机数。

中心极限定理指出,无穷多个任意分布的独立随机变量之和的分布趋近于正态分布。基于此,另外一种产生近似高斯随机数的方法是：用 12 个独立同分布于[0,1]区间的均匀分布随机数之和来构成正态分布,其均值为 6,方差为 1。因此,得到标准正态分布随机数的方法是：

$$y=\sum_{i=1}^{12}x_i-6$$

其中，x_i 是在[0,1]区间的独立均匀分布的随机数。与函数变换法相比，该方法计算简单，避免了函数运算，但是产生一个正态随机数需要 12 个独立均匀分布的随机数，计算效率较低，而且这样产生的正态分布随机数的区间是[−6,6]。

【例 5-15】 调用 randn 函数生成 8×8 的正态随机数矩阵，并将矩阵按列拉长画出频数直方图。

```
>> clear all;
x = randn(8)                          % 创建 8×8 的正态随机数矩阵,其元素服从标准正态分布
y = x(:);                             % 将 x 按列拉长生成一个列向量
hist(y);                              % 绘制频数直方图
xlabel('标准正态分布');ylabel('频数');
```

运行程序，效果如图 5-16 所示。

```
x =
  -1.3749   -0.8214    0.7399    1.9760    2.9549    0.3822   -0.8017   -0.2586
   2.3209   -0.0006   -0.2289    0.0853   -0.2191   -0.4931    0.3263   -0.3523
   0.3636   -1.8679   -1.4063   -1.1567    0.0090   -0.5342    2.1855   -0.3219
   0.0551   -0.7443    0.7503   -0.4562   -0.3830    0.2369    0.3323   -0.6775
   1.0042    1.3606   -0.7747   -0.0228   -1.0098   -1.8448   -1.1998   -1.3507
  -1.9244    0.0991   -0.8570    0.5903    0.5913    1.5002    1.6903    0.4683
   1.8628    0.4532   -1.5976   -1.2596    0.4799    0.6953   -0.7644    0.3144
  -3.0943    0.1051   -0.2425    0.2095   -0.2286   -0.3329   -0.8776   -0.4738
```

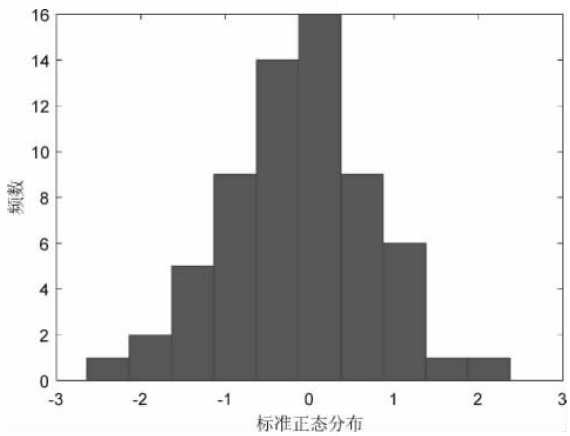

图 5-16　标准正态分布频率直方图

3. 瑞利噪声产生器

自由度为 2 的中心χ^2分布$\left(\text{即参数为}\lambda=\frac{1}{2\sigma^2}\text{的指数分布}\right)$随机变量的平方根所得出新的随机变量服从瑞利分布，即，如果随机变量 Y 的概率密度满足式 $p(y)=\frac{1}{2\sigma^2}\exp\left(-\frac{y}{2\sigma^2}\right)$，则随机变量 $R=\sqrt{Y}$ 服从瑞利分布，其概率密度函数为：

$$p(r)=\frac{r}{\sigma^2}\exp\left(-\frac{r^2}{2\sigma^2}\right),\quad x\geqslant 0 \tag{5-11}$$

瑞利分布的均值和方差分别为：

$$\mathrm{E}(R)=\sqrt{\frac{\pi\sigma^2}{2}}$$

$$\mathrm{Var}(R)=\left(1-\frac{\pi}{2}\right)\sigma^2$$

因此，产生瑞利分布随机数的方法是首先产生参数为 $\lambda=\frac{1}{2\sigma^2}$ 的指数分布随机变量（可由 0～1 范围内的均匀随机数 x 通过变换函数 $y=-2\sigma^2\ln x$ 得到，也可由两个独立的零均值 σ^2 方差的同分布正态随机数求平方和得出），然后对其求平方根即可。

MATLAB 统计工具箱给出了瑞利分布相关计算函数，如 raylpdf、raylcdf、raylinv、raylrnd、raylstat 等。

【例 5-16】 分别绘制瑞利分布的频率直方图及概率密度曲线。

```
>> clear all;
 %设置瑞利分布的参数
B = 10;m = 3;n = 4;
y = raylrnd(B,m,n);                    %创建瑞利分布
subplot(121);hist(y,10);
xlabel('取值');ylabel('计数值');
title('频率直方图');
axis square;
x = 0:0.1:3;
p = raylpdf(x,1);
subplot(122);plot(x,p);
xlabel('取值');ylabel('计数值');
title('概率密度曲线');
axis square;
```

运行程序，效果如图 5-17 所示。

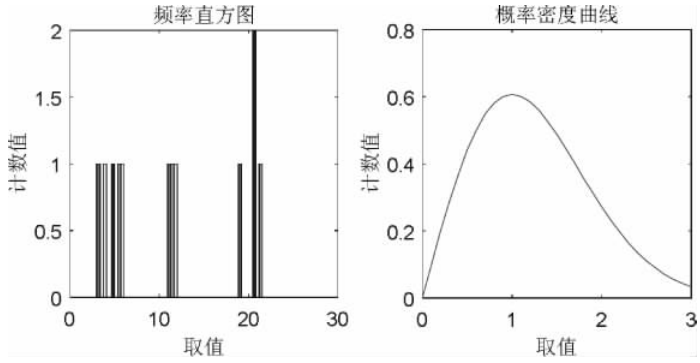

图 5-17 频率直方图及概率密度曲线效果图

5.4 信道

在信号传输的过程中，它会不可避免地受到各种干扰，这些干扰统称为噪声。根据信道中占据主导地位的噪声的特点，信道可以分成加性高斯噪声信道、多径瑞利退化信道和莱斯退化信道等。下面将分别进行介绍。

5.4.1 加性高斯白噪声信道

加性高斯白噪声是最简单的一种噪声,它表现为信号围绕平均值的一种随机波动过程。加性高斯白噪声的均值为0,方差表现为噪声功率的大小。加性高斯白噪声信道模块的作用就是在输入信号中加入高斯白噪声。

加性高斯白噪声信道模块及参数设置对话框如图5-18所示。

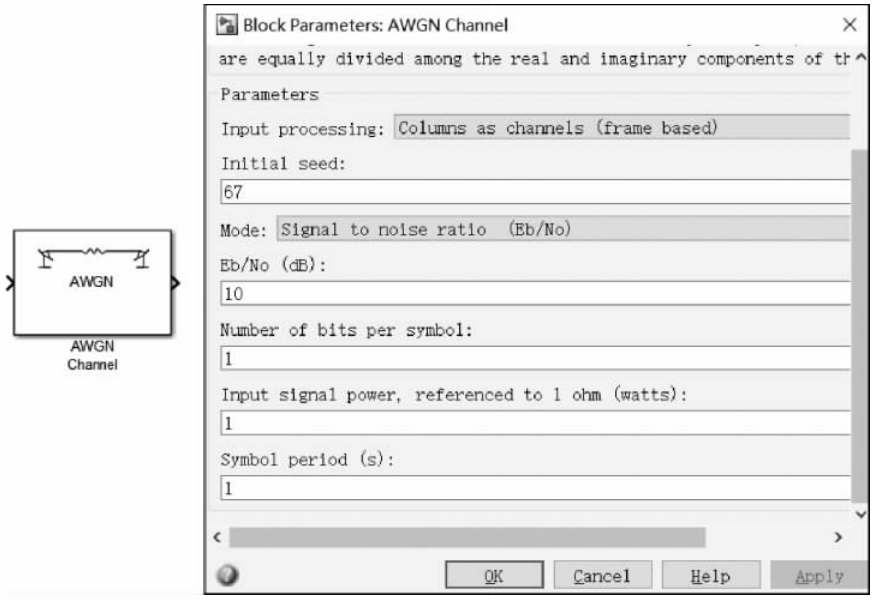

图5-18 加性高斯白噪声信道模块及参数设置对话框

加性高斯白噪声信道模块中包含多个参数项,下面分别对各项进行简单的介绍。

Initial seed:加性高斯白噪声信道模块的初始化种子。不同的初始种子值对应不同的输出,相同的值对应相同的输出。因此具有良好的可重复性,便于多次重复仿真。当输入矩阵为信号时,初始种子值可以是向量,向量中的每个元素对应矩阵的一列。

Mode:加性高斯白噪声信道模块中的模式设定。当设定为Signal to noise ration (Eb/No)时,模块根据信噪比E_b/N_0确定高斯噪声功率;当设定为Signal to noise ration (Es/No)时,模块根据信噪比E_s/N_0确定高斯噪声功率,此时需要设定三个参量:信噪比Es/No、输入信号功率和信号周期。当设定为Signal to noise ration(SNR)时,模块根据信噪比SNR确定高斯噪声功率,此时需要设定两个参量:信噪比SNR及信号周期。当设定为Variance from mask时,模块根据方差确定高斯噪声功率,这个方差由Variance指定,而且必须为正。当设定为Variance from port时,模块有两个输入,一个输入信号,另一个输入确定高斯白噪声的方差。

当输入信号为复数时,加性高斯白噪声信道模块中的E_b/N_0、E_s/N_0和SNR之间有特定的关系,如式(5-12)、式(5-13)所示。

$$E_s/N_0 = (T_{sym}/T_{samp}).\mathrm{SNR} \tag{5-12}$$

$$E_s/N_0 = E_b/N_0 \log_{10}(K) \tag{5-13}$$

在式(5-12)中，T_{sym}表示输入信号的符号周期，T_{samp}表示输入信号的采样周期。式(5-13)中E_b/N_0表示比特能量与噪声谱密度的比，K代表每个字符的比特数。加性高斯白噪声信道模块中复信号的噪声功率谱密度等于N_0，而在实信号当中，信号噪声的功率谱密度等于$N_0/2$，因此对于实信号形式的输入信号，E_s/N_0和SNR之间的关系可以表示成式(5-14)所示。

$$E_s/N_0 = 0.5(T_{sym}/T_{samp}).\mathrm{SNR} \tag{5-14}$$

Eb/No(dB)：加性高斯白噪声信道模块的信噪比E_b/N_0，单位为dB。本项只有当Mode项选定为Signal to noise ration(E_b/N_0)时有效。

Es/No(dB)：加性高斯白噪声信道模块的信噪比E_s/N_0，单位为dB。本项只有当Mode项选定为Signal to noise ration(E_s/N_0)时有效。

SNR(dB)：加性高斯白噪声信道模块的信噪比SNR，单位为dB。本项只有当Mode项选定为Signal to noise ration(SNR)时有效。

Number of bits per symbol：加性高斯白噪声信道模块每个输出字符的比特数，本项只有当Mode项选定为Signal to noise ration(E_b/N_0)时有效。

Input signal power：加性高斯白噪声信道模块输入信号的平均功率，单位为W。本项只有在参数Mode设定在Signal to noise ration(E_b/N_0、E_s/N_0、SNR)三种情况下有效。选定为Signal to noise ration(E_b/N_0、E_s/N_0)时，表示输入符号的均方根功率；选定为Signal to noise ration(SNR)时，表示输入采样信号的均方根功率。

Symbol period：加性高斯白噪声信道模块每个输入符号的周期，单位为s。本项只有在参数Mode设定在Signal to noise ration(E_b/N_0、E_s/N_0)情况下有效。

Variance：加性高斯白噪声信道模块产生的高斯白噪声信号的方差。本项只有在参数Mode设定为Variance from mask时用效。

5.4.2 多径瑞利退化信道

瑞利退化是移动通信系统中的一种相当重要的退化信道类型，它在很大程度上影响着移动通信系统的质量。在移动通信系统中，发送端和接收端都可能处在不停的运动状态之中，发送端和接收端之间的这种相对运动产生多普勒频移。多普频移与运动速度和方向有关，计算公式为：

$$f_d = (vf/c)\cos\theta$$

其中，v是发送和接收端之间的相对运动速度，θ是运动方向和发送端与接收端连线之间的夹角。c为光速，f为频率。

多径瑞利退化信道模块实现基带信号多径瑞利退化信道仿真，其输入为标量或帧格式的复信号。它对无限移动通信系统建模有很重要的意义。

5.4.3 多径莱斯退化信道

在移动通信系统中，如果发送端和接收端之间存在着一条占优势的视距传播路径，

这种信号就可以模拟成多径莱斯退化信道。当发送端和接收端之间既存在着视距传播路径,又有多条反射路径时。它们之间的信道可以同时用多径莱斯退化模块和多径瑞利退化信道来仿真。

多径莱斯退化信道模块对基带信号的多径莱斯退化信道进行仿真,其输入为标量或帧格式的复信号。多径莱斯退化信道模块及参数设置对话框如图 5-19 所示。

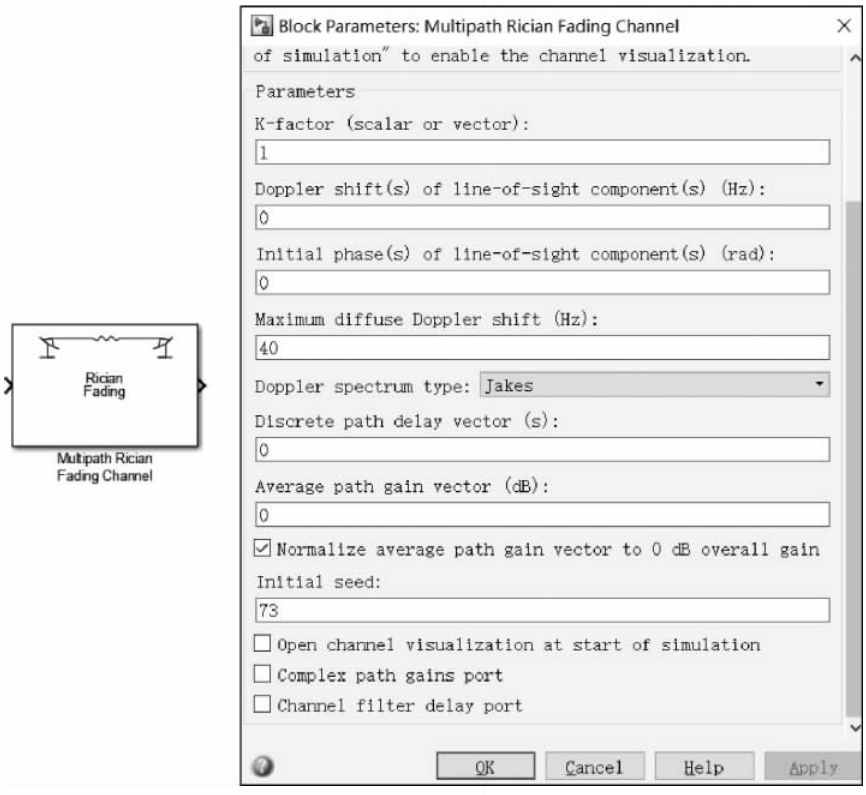

图 5-19　多径莱斯退化信道模块及参数设置对话框

多径莱斯退化信道模块中包含多个参数项,下面分别对这些参数项进行简单介绍。

K-factor(scalar or vector):多径莱斯退化信道模块中的 K 因子。它表示视距传播路径的能量与其他多径信号的能量之间的比值。K 因子越大,表示发送端和接收端之间的视距传播路径的能量越强;当 K 因子等于 0 时,发送端和接收端之间不存在视距传播路径,此时莱斯退化信道就演变成瑞利退化信道。

Doppler shift(s) of line-of-sight component(s)(Hz):多径莱斯退化信道模块中的视距传播路径多普勒频移,单位为 Hz。

Initial pahse(s) of line-of-sight component(s) (rad):设置视距的初始化相位值。

Maximum diffuse Doppler shift(Hz):多径莱斯退化信道模块中最大的扩散多普勒频移设定,必须为正数。

Doppler spectrum type:多普勒频谱类型。

Discrete path delay vector(s):多径莱斯退化信道模块输入信号各路径的时延,单位为 s。

Average path gain vector(dB)：多径莱斯退化信道模块输入信号各路径的增益，单位为 dB。

Normalize average path gain vector to 0 dB overall gain：选定本参数后，多径莱斯退化信道模块把参数 Average path gain vectort 乘上一个系数作为增益向量，使得所有路径的接收信号强度和等于 0dB。

Initial seed：多径莱斯退化信道的初始化种子。

Open channel visualization at start of simulation：多径莱斯退化信道模块中通道可视化选项。选定该项，仿真开始时将会打开通道可视化工具。

Complex path gains port：多径莱斯退化信道模块复路径增益端口项。选定后，输出每个通道的复数路径增益。这是一个 $N\times M$ 多通道结构，其中 N 为每帧样品数，M 为离散的路径数。

Channel filter delay port：多径莱斯退化信道模块信道滤波延时端口项，选定后，输出本模块中由于滤波引起的延时。单路径时，延时为 0；多路径时，延时大于 0。

5.5 信号观测设备

在通信系统的仿真过程中，用户希望能够把接收到的数据通过某种方式保存或显示出来，以直观的形态对仿真的结果进行评估，这就需要用到信号观测设备。MATLAB 提供了若干个模块用于实现这种功能。

5.5.1 星座图

星座图又称离散时间发散图，通常用来观测调制信号的特性和信道对调制信号的干扰特性。星座图模块接收复信号，并且根据输入信号绘制发散图。星座图模块只有一个输入端口，输入信号必须为复信号。双击星座图模块，弹出如图 5-20 所示的示波器窗口，单击示波器中的 按钮，即可打开其参数设置对话框，星座图模块及参数设置对话框如图 5-21 所示。

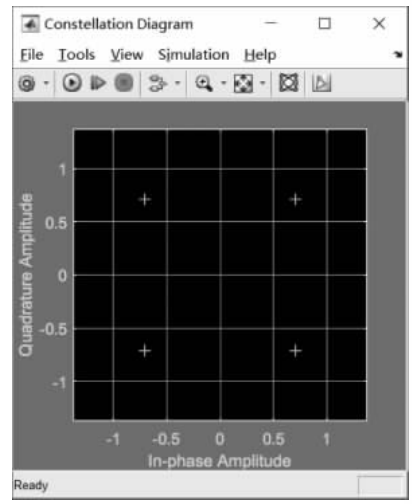

图 5-20 星座图示波器

由图 5-21 可见，星座图参数设置窗口中包含两个选项，下面分别对这两个选项进行介绍。

1. Main 选项

Main 选项为星座图的主选项，用来设定星座图的绘制方式。该项为默认项，如图 5-21 所示，其包含如下参数：

- Samples per symbol：设定星座图中每个符号的采样点数目。
- Offset(samples)：开始绘制星座图之前应该忽略的采样点个数。该项必须是小于

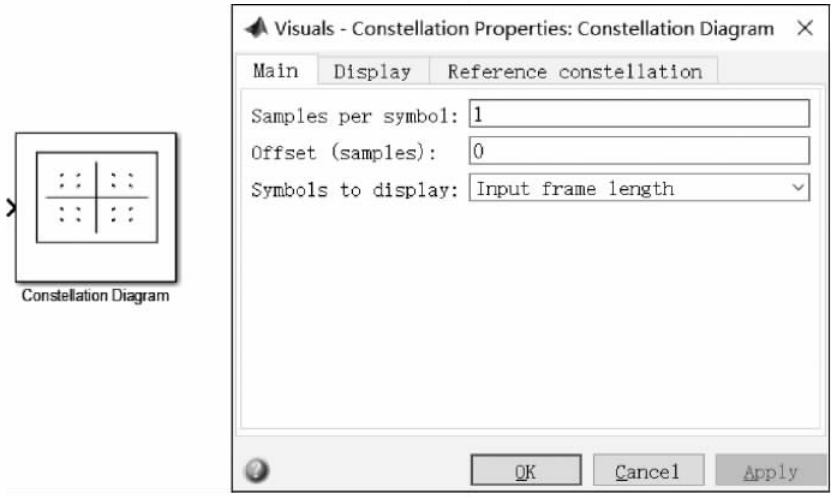

图 5-21　星座图模块及参数设置对话框

Sample per symbol 项的非负整数。

- Symbols to display：符号显示形式。
- Reference constellation：星座图参考，为一个矩阵。

2. Display 选项

该选项主要用于设定星座图的显示形式，选定该项后，显示如图 5-22 所示。

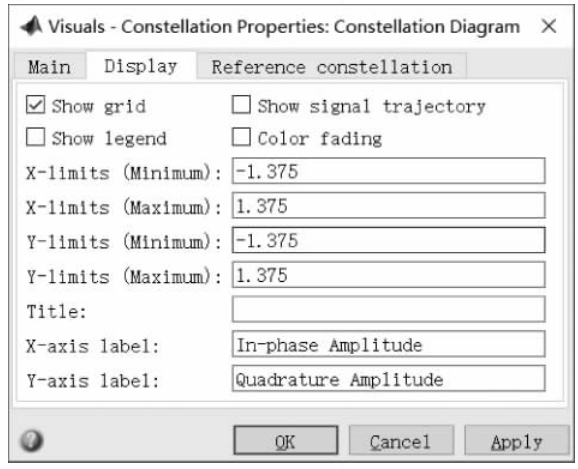

图 5-22　Display 选项

Display 选项各参数的含义为：

- Show grid：显示网格。
- Show legend：显示图例。
- Color fading：颜色渐变复选框。选定后，眼图中每条迹上的点的颜色深度随着仿真时间的推移而逐渐减弱。
- Show reference constellation：显示星座图参考线。

- Reference marker：设定星座图中每个采样点的绘制方式。
- X-limits(Minimum)：设定星座图观测仪横坐标的最小值。
- X-limits(Maximum)：设定星座图观测仪横坐标的最大值。
- Y-limits(Minimum)：设定星座图观测仪纵坐标的最小值。
- Y-limits(Maximum)：设定星座图观测仪纵坐标的最大值。
- Title：设置星座图标题。
- X-axis label：设置星座图横坐标的标签。
- Y-axis label：设置星座图纵坐标的标签。

3. Reference constellation 选项

该选项用于设定星座图的参考线，选定该项后，显示如图 5-23 所示。

Reference constellation 选项各参数的含义为：

- Show reference constellation：显示星座参考线。
- reference constellation：选择参考线的模型。
- Average reference powser：指定星座的平均参考功率。
- Reference phase offset(rad)：指定星座的参考相位偏移。

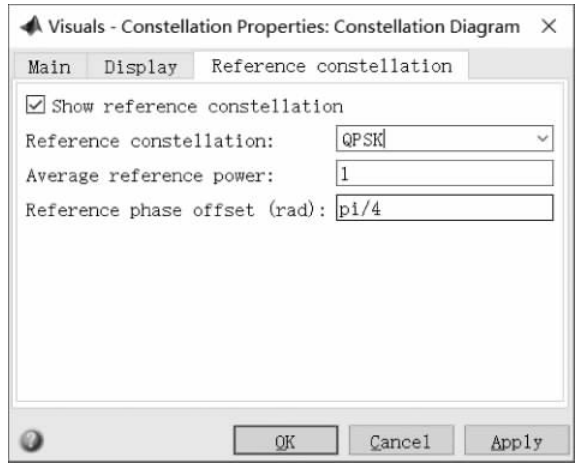

图 5-23　Reference constellation 选项

5.5.2　误码率计算器

误码率计算器模块分别从发射端和以间接手段得到输入数据。再对两个数据进行比较，根据比较的结果计算误码率。

应用这个模块，既可以得到误比特率，也可以得到误符号率。当输入信号是二进制数据时，则统计的结果是误比特率，否则，统计得到的结果是误符号率。误码率计算器模块只比较两个输入信号的正负关系，而不具体地比较它们的大小。误码率计算器模块及参数设置对话框如图 5-24 所示。

误码率计算器模块中有若干参数，下面分别对其进行简单说明。

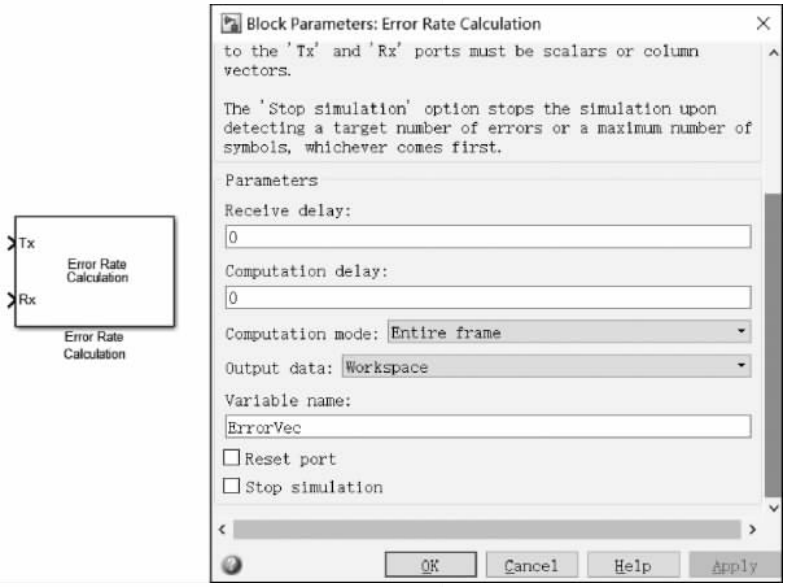

图 5-24　误码率计算器模块及参数设置对话框

(1) Receive delay：接收端时延设定项。

在通信系统中，接收端需要对接收到的信号进行解调、解码或解交织，这些过程可能会产生一定的时延，使得到达误码率计算器接收端的信号滞后于发送端的信号。为了弥补这种时延，误码率计算器模块需要把发送端的输入数据延时若干个输入数据，本参数即表示接收端输入的数据滞后发送端数据输入数据的大小。

(2) Computation delay：计算时延设定项。在仿真过程中，有时需要忽略初始的若干输入数据，这就可以通过本项设定。

(3) Computation mode：计算模式项。误码率计算器模块有三种计算模式。分别为帧计算模式、掩码模式和端口模式。其中帧计算模式对发送端和接收端的所有输入数据进行统计。在掩码模式下，模块根据掩码对特定的输入数据进行统计，掩码的内容可由参数项 Selected samples from frame 设定。在端口模式下，模块会新增一个输入端 Sel，只有此端口的输入信号有效时才统计错误率。

(4) Selected samples from frame：掩码设定项。本参数用于设定哪些输入数据需要统计。本项只有当 Computation mode 项设定为 samples from mask 时有效。

(5) Output data：设定数据输出方式，有 Worksapce 和 Port 两种方式。Worksapce 方式时将统计数据输出到工作区，Port 方式时将统计数据从端口中输出。

(6) Variable name：指定用于保存统计数据的工作区间变量的名称。本项只有在 Output data 设定为 Workspace 时有效。

(7) Reset port：复位端口项。选定此项后，模块增减一个输入端口 Rst，当这个信号有效时，模块被复位，统计值重新设定为 0。

(8) Stop simulation：仿真停止项。选定本项后，如果模块检测到指定对象的错误，或数据的比较次数达到了门限，则停止仿真过程。

(9) Target number of errors：错误门限项。用于设定仿真停止之前允许出现错误的最大个数。本项只有在 Stop simulation 选定后有效。

(10) Maximum number of symbols：比较门限项。用于设定仿真停止之前允许比较的输入数据的最大个数。本项只有在 Stop simulation 选定后有效。

5.6 信源编译码

信源编码是用量化的方式将一个源信号转化为一个数字信号，所得信号的符号为某一有限范围内的非负整数。信源译码就是将信源编码的信号恢复到原来的信号。

5.6.1 信源编码

信源编码也称为量化或信号格式化，它一般是为了减少冗余或为后续的处理做准备而进行的数据处理。在 Simulink 中，提供了 A 律编码、Mu 律编码、差分编码和量化编码等模块，下面分别进行介绍。

1. A 律编码模块

模拟信号的量化有两种方式：均匀量化和非均匀量化。均匀量化把输入信号的取值范围等距离地分割成若干个量化区间，无论采样值大小怎样，量化噪声的均值和均方根固定不变，因此实际过程中大多采用非均匀量化。比较常用的两种非均匀量化的方法是 A 律压缩和 Mu 律压缩。

如果输入信号为 x，输出信号为 y，则 A 律压缩满足：

$$y=\begin{cases}\dfrac{A\mid x\mid}{1+\log A}\operatorname{sgn}(x), & 0\leqslant x\leqslant\dfrac{V}{A}\\[2ex]\dfrac{V(1+\log(A\mid x\mid/V))}{1+\log A}\operatorname{sgn}(x), & \dfrac{V}{A}\leqslant x\leqslant V\end{cases}\tag{5-15}$$

式中，A 为 A 律压缩参数，最常用采用的 A 值为 87.6；V 为输入信号的峰值；log 为自然对数；sgn 函数当输入为正时，输出 1，当输入为负时，输出 0。

模块的输入并无限制。如果输入为向量，则向量中的每一个分量将会被单独处理。A 律压缩编码模块及参数设置对话框如图 5-25 所示。

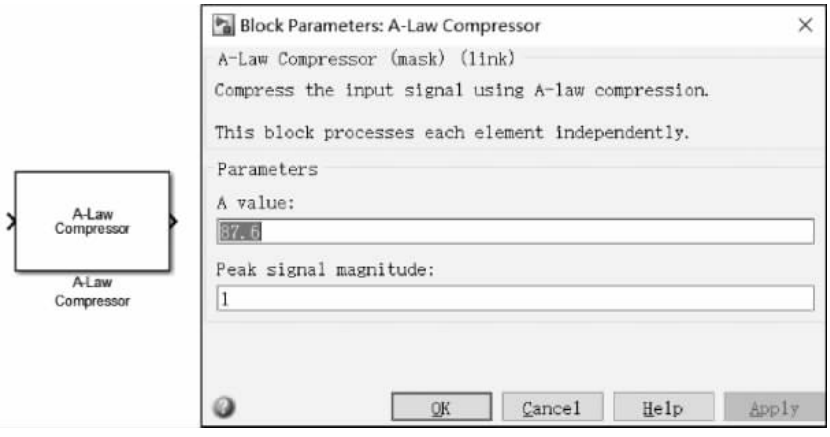

图 5-25 A 律压缩编码模块及参数设置对话框

A 律压缩编码模块参数设置对话框中包含两个参数，下面分别对其进行简单说明。

- A value：用于指定压缩参数 A 的值。
- Peak signal magnitude：用于指定能输入信号的峰值 V。

2. Mu 律编码模块

与 A 律压缩编码类似，Mu 律压缩编码中如果输入信号为，输出信号为，则 Mu 律压缩满足：

$$y=\frac{V\log(1+\mathrm{Mu}\mid x\mid/V)}{\log(1+\mathrm{Mu})}\mathrm{sgn}(x)$$

式中，Mu 为 Mu 律压缩参数；V 为输入信号的峰值；log 为自然对数；sgn 函数当输入为正时，输出 1，当输入为负时，输出 0。

模块的输入并无限制，如果输入为向量，则向量中的每一个分量将会被单独处理。Mu 律压缩编码模块及参数设置对话框，如图 5-26 所示。

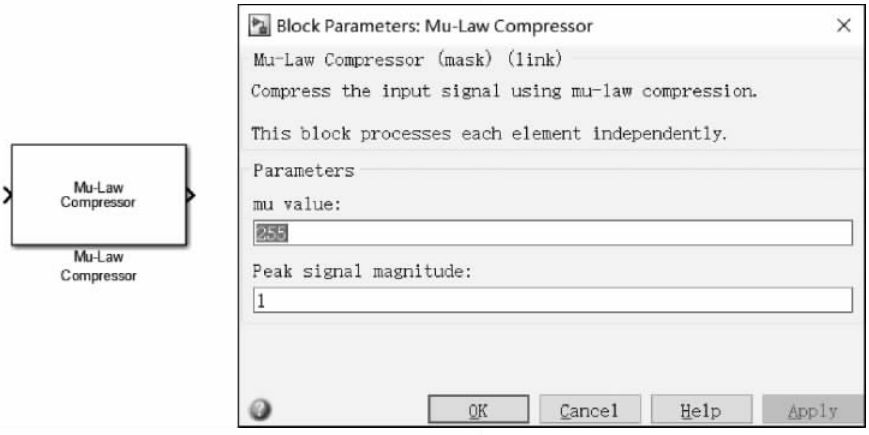

图 5-26　Mu 律压缩编码模块及参数设置对话框

Mu 律压缩编码模块参数对话框的参数含义为：

- mu value：用于指定 Mu 律压缩参数 Mu 的值。
- Peak signal magnitude：用于指定能输入信号的峰值 V，也是输出信号的峰值。

3. 差分编码模块

差分编码又称为增量编码，它用一个二进制数来表示前后两个采样信号之间的大小关系。在 MATLAB 中，差分编码器根据当前时刻之前的所有输入信息计算输出信号，这样，在接收端即可只按照接收到的前后两个二进制信号恢复出原来的信息序列。

差分编码模块对输入的二进制信号进行差分编码，输出二进制的数据流。输入的信号可以是标量、向量或帧格式的行向量。如果输入信号为 $m(t)$，输出信号为 $d(t)$，那么 t_k 时刻的输出 $d(t_k)$ 不仅与当前时刻的输入信号 $m(t_k)$ 有关，而且与前一时刻的输出 $d(t_{k-1})$ 有关，如下式所示。

$$\begin{cases} d(t_0) = (m(t_0)+1)\bmod 2 \\ d(t_k) = (m(t_{k-1})+m(t_k)+1)\bmod 2 \end{cases}$$

即输出信号 y 取决于当前时刻以及当前时刻之前所有的输入信号的数值。

差分编码模块及参数设置对话框如图 5-27 所示。

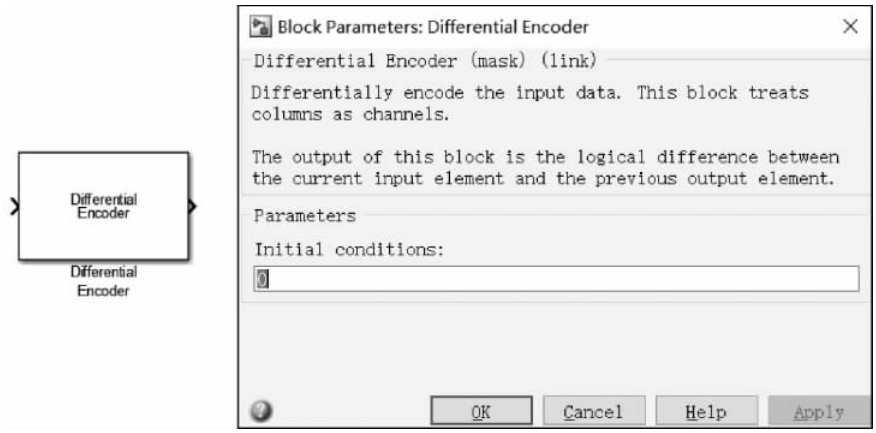

图 5-27 差分编码模块及参数设置对话框

差分编码模块中包含一个参数，含义为：

Initial conditions：用于指定信号符号之间的间隔。

4. 量化编码

量化编码模块用标量量化法来量化输入信号。它根据量化间隔和量化码本把输入信号转换成数字信号，并且输出量化指标、量化电平、编码信号和量化均方误差。

模块的输入信号可以是标量、向量或矩阵。模块的输入输出信号长度相同。

量化编码模块及参数设置对话框如图 5-28 所示。

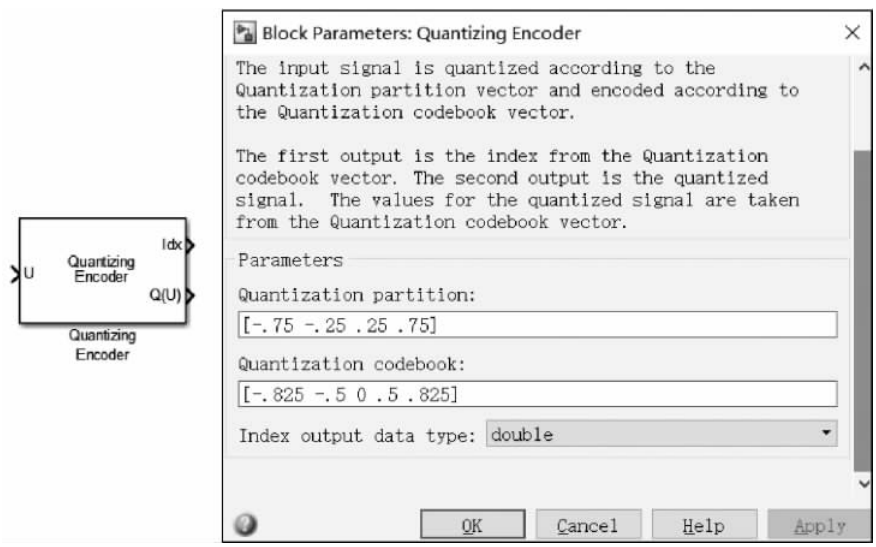

图 5-28 量化编码模块及参数设置对话框

量化编码模块中包含三个参数,主要含义为:

- Quantization partition:用于指定量化区间,为一个长度为 n 的向量(n 为码元素)。该向量分量要严格按照升序排列。如果设该参量为 p,那么模块的输出 y 与输入 x 之间的关系满足:

$$y=\begin{cases}0, & x\leqslant p(1)\\ m, & p(m)<x\leqslant p(m+1)\\ n, & p(n)\leqslant x\end{cases}$$

- Quantization codebook:表示量化区间的量化值,是一个长度为 $n+1$ 的向量。
- Index output data type:索引输出数据类型。

5.6.2 信源译码

在 Simulink 中也提供了对应的模块实现译码。

1. A 律译码模块

A 律译码模块用来恢复被 A 律压缩模块压缩的信号。它的过程与 A 律压缩编码模块正好相反。A 律译码模块的特征函数是 A 律压缩编码模块特征函数的反函数,如下式所示:

$$x=\begin{cases}\dfrac{y(1+\log A)}{A}, & 0\leqslant|y|\leqslant\dfrac{V}{1+\log A}\\ \exp(|y|(1+\log A)/V-1)\dfrac{V}{A}\operatorname{sgn}(y), & \dfrac{V}{1+\log A}\leqslant|y|\leqslant V\end{cases}$$

A 律译码模块及参数设置对话框如图 5-29 所示。

A 律译码模块参数设置对话框中包含两个参数,含义为:

- A value:用于指定压缩参数 A 的值。
- Peak signal magnitude:用于指定能输入信号的峰值 V,同时也是输出信号的峰值。

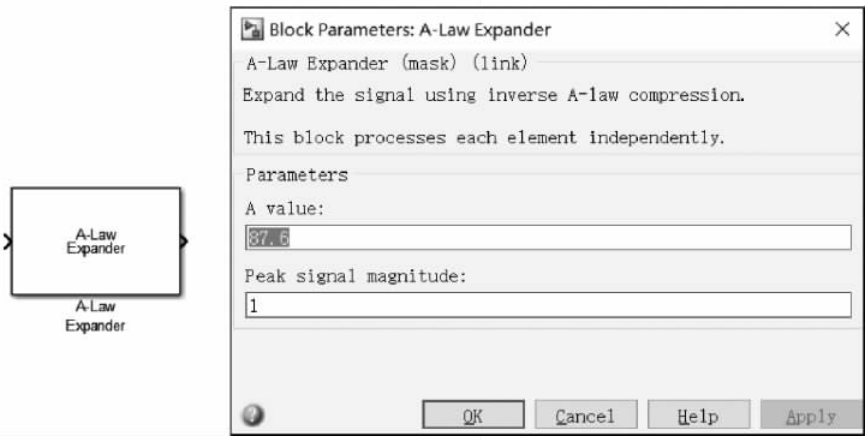

图 5-29 A 律译码模块及参数设置对话框

2. Mu 律译码模块

Mu 律译码模块用来恢复被 Mu 律压缩模块压缩的信号。它的过程与 Mu 律压缩编码模块正好相反。Mu 律译码模块的特征函数是 Mu 律压缩编码模块特征函数的反函数，如下式所示：

$$x = \frac{V}{\mathrm{Mu}}(\mathrm{e}^{|y|\log(1+Mu)/V} - 1)\mathrm{sgn}(y)$$

Mu 律译码模块及参数设置对话框如图 5-30 所示。

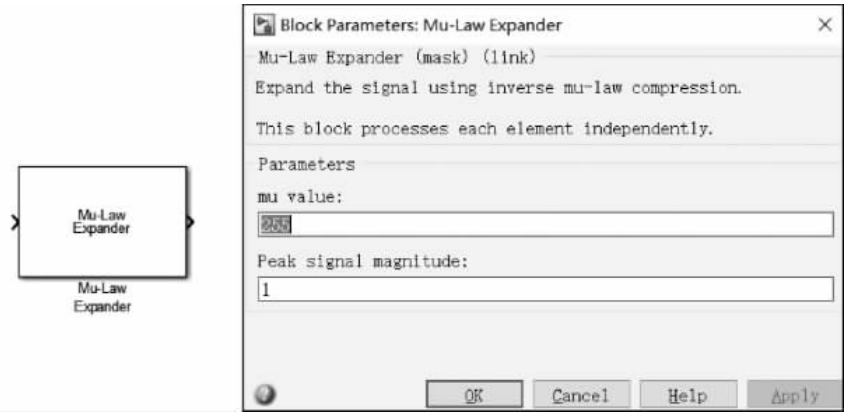

图 5-30 Mu 律译码模块及参数设置对话框

Mu 律译码模块参数设置对话框中包含两个参数，含义为：

- mu value：用于指定 Mu 律压缩参数 Mu 的值。
- Peak signal magnitude：用于指定能输入信号的峰值 V，也是输出信号的峰值。

3. 差分译码模块

差分译码模块对输入信号进行差分译码。模块的输入输出均为二进制信号，且输入输出之间的关系和差分编码模块中的两者关系相同。

差分译码模块及参数设置对话框如图 5-31 所示。

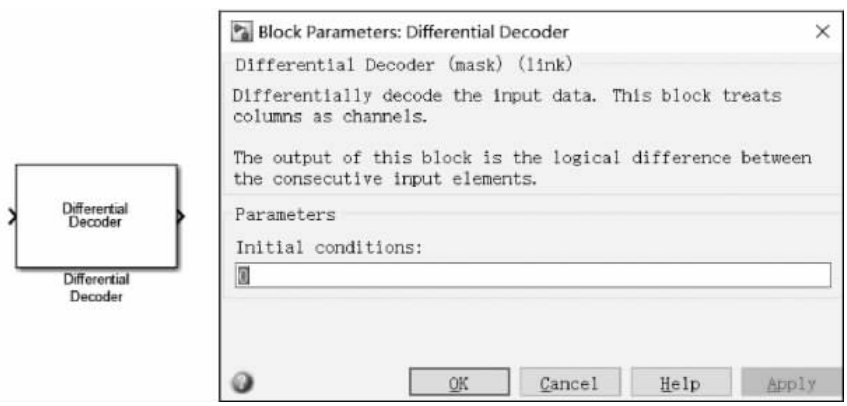

图 5-31 差分译码模块及参数设置对话框

差分译码模块参数设置对话框包含一个参数,含义为:

Initial conditions:用于指定信号符号之间的间隔。

4. 量化译码模块

量化译码模块用于从量化信号中恢复出消息,它执行的是量化编码模块的逆过程。模块的输入信号是量化的区间号,可以是标量、向量或矩阵。如果输入为向量,那么向量的每一个分量将被分别单独处理。量化译码模块中的输入输出信号的长度相同。

量化译码模块及参数设置对话框如图 5-32 所示。

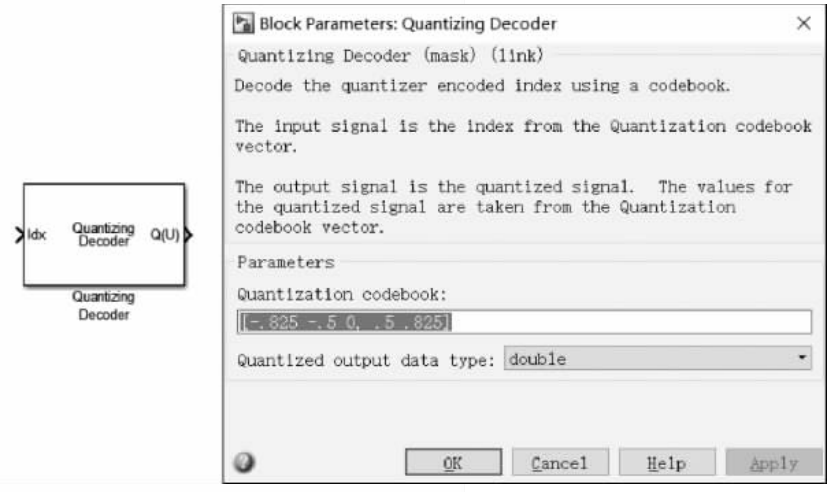

图 5-32 量化译码模块及参数设置对话框

量化译码模块中包含两个参数,含义为:

- Quantization codebook:表示每一个非负整数输入所对应的输出实向量。
- Quantization output data type:索引输出数据类型。

5.7 MATLAB/Simulink 通信系统仿真实例

在前面简单介绍了利用 MATLAB 及 Simulink 实现信源产生和信道产生,本节将通过具体实例进行演示。

5.7.1 MATLAB 编码实例

信源编码可分为两类:无失真编码和限失真编码。目前已有各种无失真编码算法,例如 Huffman 编码和 Lempel-Ziv 编码。这里介绍无失真编码中的最佳变长编码——Huffman 码。Huffman 编码的基本原理就是为概率较小的信源输出分配较长的码字,而对那些出现可能性较大的信源输出分配较短的码字。

Huffman 编码算法及步骤如下:

(1) 将信源消息按照概率大小顺序排列。

(2) 按照一定的规则,从最小概率的两个消息开始编码。例如,将较长的码字分配给较小概率的消息,把较短的码字分配给概率较大的消息。

(3) 将经过编码的两个消息的概率合并,并重新按照概率大小排序,重复步骤(2)。

(4) 重复上面的步骤(3),一直到合并的概率达到 1 时停止。这样便可以得到编码树状图。

(5) 按照从上到下编码的方式编程,即从树的根部开始,将 0 和 1 分别放到合并成同一节点的任意两个支路上,这样就产生了这组 Huffman 码。

Huffman 码的效率为:

$$\eta = \frac{\text{信息熵}}{\text{平均码长}} = \frac{H(X)}{\bar{L}}$$

【例 5-17】 利用 Huffman 编码算法实现对某一信源的无失真编码。该信源的字符集为 $X=\{x_1,x_2,\cdots,x_6\}$,相应的概率向量为:$P=\{0.30,0.10,0.21,0.09,0.05,0.25\}$。

首先将概率向量 P 中的元素进行排序,$P=\{0.30,0.25,0.21,0.10,0.09,0.05\}$。然后根据 Huffman 编码算法得到 Huffman 树状图,如图 5-33 所示,编码之后的树状如图 5-34 所示。

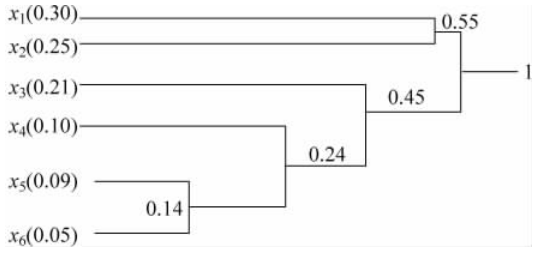

图 5-33 Huffman 树状图

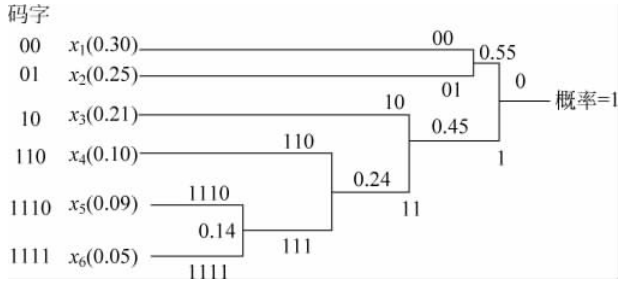

图 5-34 Huffman 编码树

由图 5-34 可知 x_1,x_2,x_3,x_4,x_5,x_6 的码字依次分别为 00、01、10、110、1110、1111。

平均码长为:

$$\bar{L}=2\times(0.30+0.25+0.21)+3\times0.10+4\times(0.09+0.05)=2.38\text{b}$$

信源的熵为:

$$H(X)=-\sum_{i=1}^{6}p_i\log_2 p_i=2.3549\text{b}$$

所以,Huffman 码的效率为:

$$\eta = H(X)/\bar{L} = 0.9895$$

因此,可以利用 MATLAB 将 Huffman 编码算法编写成函数文件 huffman_code,实现对具有概率向量 P 的离散无失真信源的 Huffman 编码,并得到其码字和平均码长。

在 M 文件编辑器中输入以下 huffman_code.m 函数代码。

```
function [h,e] = huffman_code(p)
% Huffman 代码如下
if length(find(p<0))~ = 0,
    error('Not a prob.vector');                    % 判断是否符合概率分布的条件
end
if abs(sum(p) - 1)> 10e - 10,
    error('Not a prob.vector');
end
n = length(p);
for i = 1:n - 1,                                   % 对输入的概率进行从大到小排序
    for j = i:n
        if p(i)< = p(j)
            P = p(i);
            p(i) = p(j);
            p(j) = P;
        end
    end
end
disp('概率分布');
p                                                  % 显示排序结构
q = p;
m = zeros(n - 1,n);
for i = 1:n - 1,
    [q,e] = sort(q);
    m(i,:) = [e(1:n - i + 1),zeros(1,i - 1)];
    q = [q(1) + q(2) + q(3:n),e];
end
for i = 1:n - 1,
    c(i,:) = blanks(n * n);
end
% 以下计算各个元素码字
c(n - 1,n) = '0';
c(n - 2,2 * n) = '1';
for i = 2:n - 1
    c(n - i,1:n - 1) = c(n - i + 1,n * (find(m(n - i + 1,:) == 1)) - (n - 2):n * (find(m(n - i +
1,:) == 1)));
    c(n - i,n) = '0';
    c(n - i,n + 1:2 * n - 1) = c(n - i,1:n - 1);
    c(n - i,2 * n) = '1';
    for j = 1:i - 1
        c(n - i,(j + 1) * n + 1:(j + 2) * n) = c(n - i + 1,n * (find(m(n - i + 1,:) == j + 1) -
1) + ...
1:n * find(m(n - i + 1,:) == j + 1));
    end
end
for i = 1:n
    h(i,1:m) = c(1,n * (find(m(1,:) == i) - 1) + 1:find(m(1,:) == i) * n);
    e(i) = length(find(abs(h(i,:))~ = 32));
end
e = sum(p. * e); % 计算平均码长
```

在命令行窗口中，只需调用函数文件 huffman_code，计算如下：

```
>> p = [0.30 0.10 0.21 0.09 0.05 0.25];
>> [h,e] = huffman_code(p)
```

输出结果为：

```
概率分布
p =
    0.3000   0.2500   0.2100   0.1000   0.0900   0.0500
h =                                              %输出各个元素码字
    11
    10
    00
    010
    0111
    0110
e = 2.3800                                       %输出平均码长
```

【例 5-18】 若输入 A 律 PCM 编码器的正弦信号为 $x(t)=\sin(1600\pi t)$，采样序列为 $x(n)=\sin(0.2\pi n)$，$n=0,1,2,\cdots,10$，将其进行 PCM 编码，给出编码器的输出码组序列 $y(n)$。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
x = [0:0.001:1];                                 %定义幅度序列
y1 = apcm(x,1);                                  %参数为 1 的 A 律曲线
y2 = apcm(x,10);                                 %参数为 10 的 A 律曲线
y3 = apcm(x,87.65);                              %参数为 87.65 的 A 律曲线
plot(x,y1,':',x,y2,'-',x,y3,'-.');
legend('A = 1','A = 10','A = 87.65')
```

运行程序，得到的效果如图 5-35 所示。

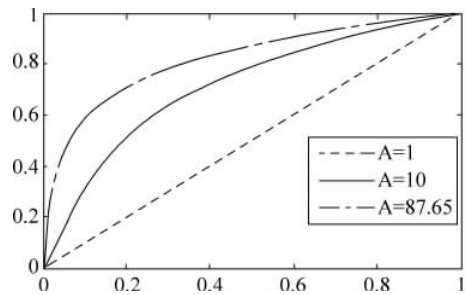

图 5-35 对数量化特性曲线

在运行程序过程中，调用自定义编写的 apcm.m 函数，其源代码如下：

```
function y = apcm(x,a)
% 本函数实现将输入的序列 x 进行参数为 A 的
%对数运算
% A 律量化将得到的结果存在序列 y 中
% x 为一个序列，值在 0 到 1 之间
% a 为一个正实数，大于 1
t = 1/a;
for i = 1:length(x)
    if (x(i)>=0),                                %判断该输入序列值是否大于 0
        if (x(i)<=t),
            y(i) = (a*x(i))/(1+log(a));          %若值小于 1/a，则采用此计算法
        else
            y(i) = (1+log(a*x(i)))/(1+log(a));   %若值大于 1/A，则采用另一计算法
        end
    else
        if (x(i)>=-t),                           %若值小于 0，则算法有所不同
            y(i) = -(a*-x(i))/(1+log(a));
        else
```

```
            y(i) = -(1+log(a*-x(i)))/(1+log(a));
        end                                          %内层条件判断结束
    end                                              %外层条件判断结束
end
%运用上面的压缩特性来解本例
>> x = 0:1:10;
y = sin(0.2*pi*x);
z = apcm(y,87.5)                                     %求 sin(0)到 sin(10)的量化值
z =
    0   0.9029   0.9908   0.9908   0.9029   0.0000   -0.9029   -0.9908   -0.9908   -0.9029
-0.0000
```

【例 5-19】 使用 MATLAB 编程方法实现对 HDB3 码的编码/解码。

HDB3 码规定,每当出现四个连 0 时,用以下两种取代节代替这四个连 0,规则是:

(1) 令 V 表示违反极性交替规则的传号脉冲,B 表示符合极性交替规则的传号脉冲,当相邻两个 V 脉冲之间的传号脉冲数为奇数时,以 000V 作为取代节。

(2) 当相邻两个 V 脉冲之间的信号脉冲数为偶数时,以 B00V 作为取代节。

这样,就能始终保持相邻 V 脉冲之间的 B 脉冲数为奇数,使得 V 脉冲序列自身也满足极性交替规则。

对 HDB3 码解码很容易,根据 V 脉冲极性破坏规则,只要发现当前脉冲极性与上一个脉冲极性相同,就可判断当前脉冲为 V 脉冲,从而将 V 脉冲连同之前的 3 个传输时隙均置为 0,即可清除取代节,然后取绝对值即可恢复归零二进制序列。

实现的 MATLAB 代码为:

```
>> clear all;
xn = [1 0 1 1 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 0];    %输入单极性码
yn = xn;                                             %输出 yn 初始化
num = 0;                                             %计算器初始化
for k = 1:length(xn)
    if xn(k) == 1
        num = num + 1;                               %"1"计数器
        if num/2 == fix(num/2)                       %奇数个 1 时输出 -1,进行极性交替
            yn(k) = 1;
        else
            yn(k) = -1;
        end
    end
end
%HDB3 编码
num = 0;                                             %连零计数器初始化
yh = yn;                                             %输出初始化
sign = 0;                                            %极性标志初始化为 0
V = zeros(1,length(yn));                             %V 脉冲位置记录变量
B = zeros(1,length(yn));                             %B 脉冲位置记录变量
for k = 1:length(yn)
    if yn(k) == 0
        num = num + 1;                               %连 0 个数计数
        if num == 4
            num = 0;                                 %如果连 0 个数为 4,计数器清 0
            yh(k) = 1*yh(k-4);                       %让 0000 的最后一个 0 改变为与前一个非 0
                                                     %符号相同极性的符号
            V(k) = yh(k);                            %V 脉冲位置记录
            if yh(k) == sign                         %如果当前 V 符号与前一个 V 符号极性相同
```

```
                yh(k) = -1 * yh(k);                 % 则让当前 V 符号极性反转,以满足 V 符号
                                                    % 间相互极性反转要求
                yh(k-3) = yh(k);                    % 添加 B 符号,与 V 符号同极性
                B(k-3) = yh(k);                     % B 脉冲位置记录
                V(k) = yh(k);                       % V 脉冲位置记录
                yh(k+1:length(yn)) = -1 * yh(k+1:length(yn));
                                                    % 并让后面的非 0 符号从 V 开始再交替变化
            end
            sign = yh(k);                           % 记录前一个 V 符号的极性
        end
    else
        num = 0;                                    % 当前输入为[1],则连[0]计数器清 0
    end
end                                                 % 完成编码
re = [xn',yn',yh',V',B']                            % 结果输出
% HDB3 解码
input = yh;
decode = input;                                     % 输出初始化
sign = 0;                                           % 极性标志初始化
for k = 1:length(yh)
    if input(k)~ = 0
        if sign == yh(k)                            % 如果当前码与前一个非 0 码的极性相同
            decode(k-3:k) = [0 0 0 0];              % 则该码判为 V 码并将 * 00V 清 0
        end
        sign = input(k);                            % 极性标志
    end
end
decode = abs(decode);                               % 整流
error = sum([xn' - decode']);                       % 解码的正确性检验
% 作图
subplot(311);stairs([0:length(xn)-1],xn);axis([0 length(xn) -2 2]);
subplot(312);stairs([0:length(xn)-1],yh);axis([0 length(xn) -2 2]);
subplot(313);stairs([0:length(xn)-1],decode);axis([0 length(xn) -2 2]);
```

运行程序,输出如下,效果如图 5-36 所示。

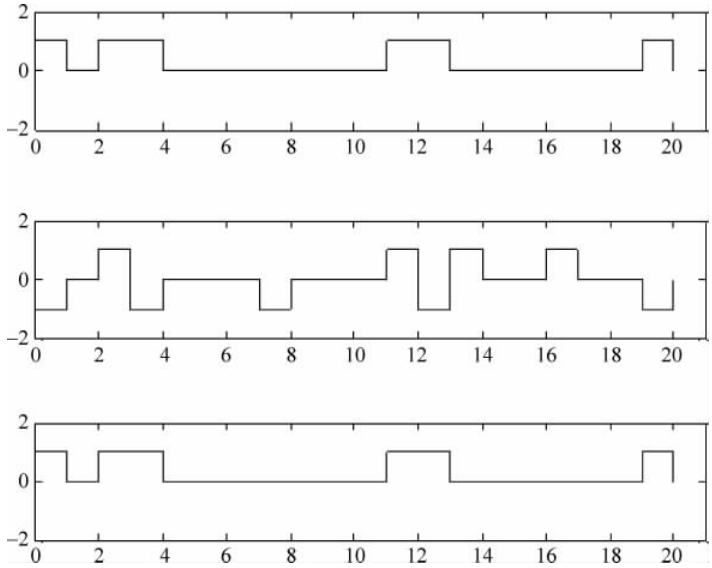

图 5-36 HDB3 码编码/解码仿真图

```
re =
     1    -1    -1     0     0
     0     0     0     0     0
     1     1     1     0     0
     1    -1    -1     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0    -1    -1     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     1     1     1     0     0
     1    -1    -1     0     0
     0     0     1     0     1
     0     0     0     0     0
     0     0     0     0     0
     0     0     1     1     0
     0     0     0     0     0
     0     0     0     0     0
     1     1    -1     0     0
     0     0     0     0     0
```

5.7.2 Simulink 信道实例

下面利用 Simulink 提供的模块，实现信道。

【例 5-20】 设某二进制数字通信系统的码元传输速率为 100bps，仿真模型的系统采样频率为 1000Hz。用示波器观察并比较信号经过高斯白噪声信道前后的不同。

(1) 根据题意，建立如图 5-37 所示的通信系统模型。

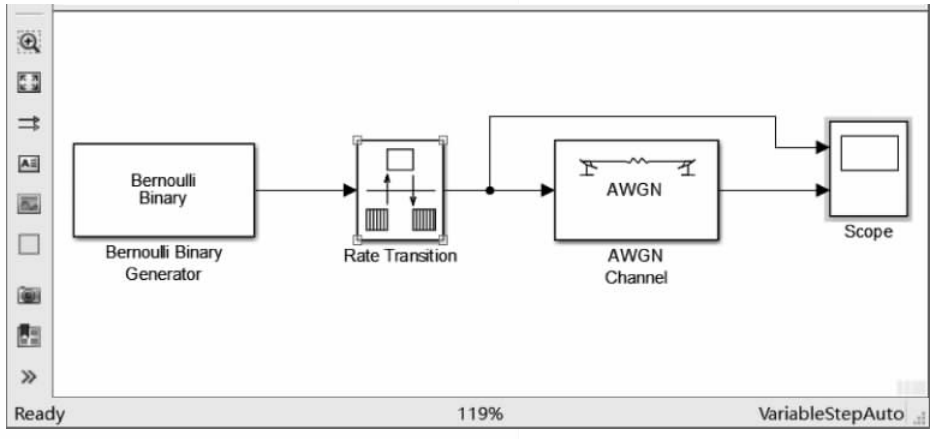

图 5-37 建立的通信系统模型

(2) 设置模块参数。

双击图 5-37 中的 Bernoulli Binary Generator 模块，设置产生零的概率为 0.5，初始种子随意设置，采样时间为 0.01 以产生 100bps 的二进制随机信号，如图 5-38 所示。

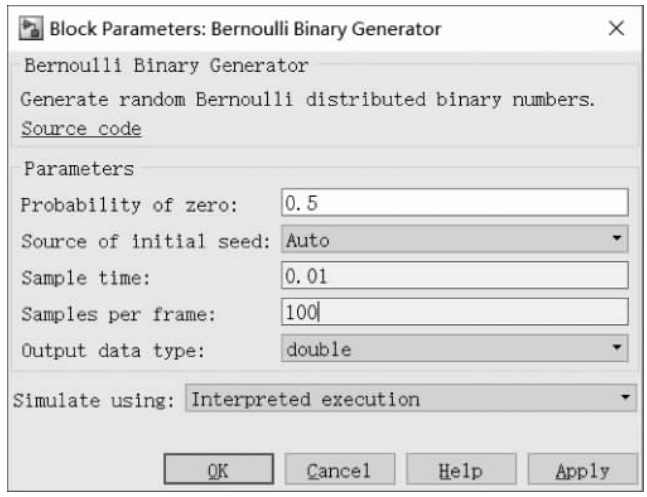

图 5-38　Bernoulli Binary Generator 模块参数设置

双击图 5-37 中的 Rate Transition 模块，设置输出端口的采样时间为 0.001，这样系统采样频率即为 1000Hz，如图 5-39 所示。

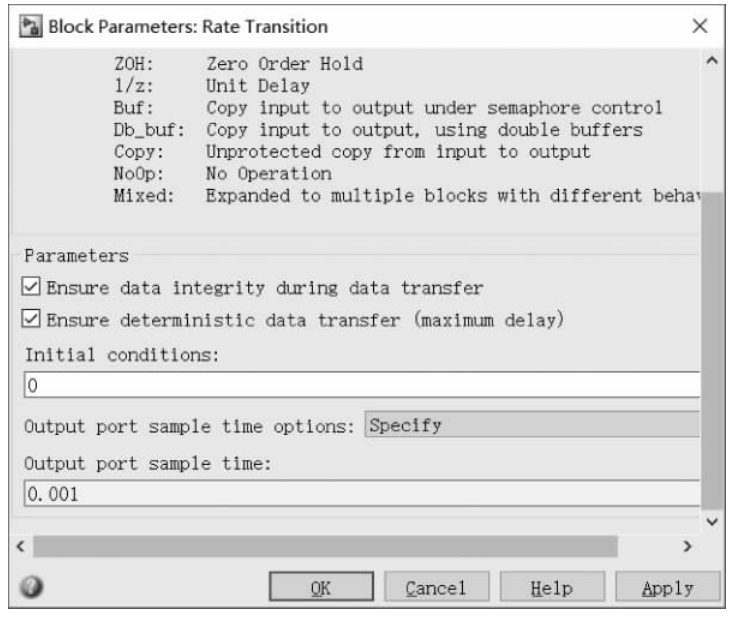

图 5-39　Rate Transition 模块参数设置

双击图 5-37 中的 AWGN Channel 模块，初始种子随意设置，信道模式设为 Signal to noise ratio(Eb/No)，Es/No 设为 25dB，输入信号功率为 1W，输入符号周期为 0.01，如图 5-40 所示。

双击图 5-37 中的 Scope 模块，在弹出的示波器窗口中，单击界面中的 按钮，在弹出的参数设置窗口中，在 General 选项中，将 Number of axes 设置为 2，即可有两个输入，效果如图 5-41 所示。

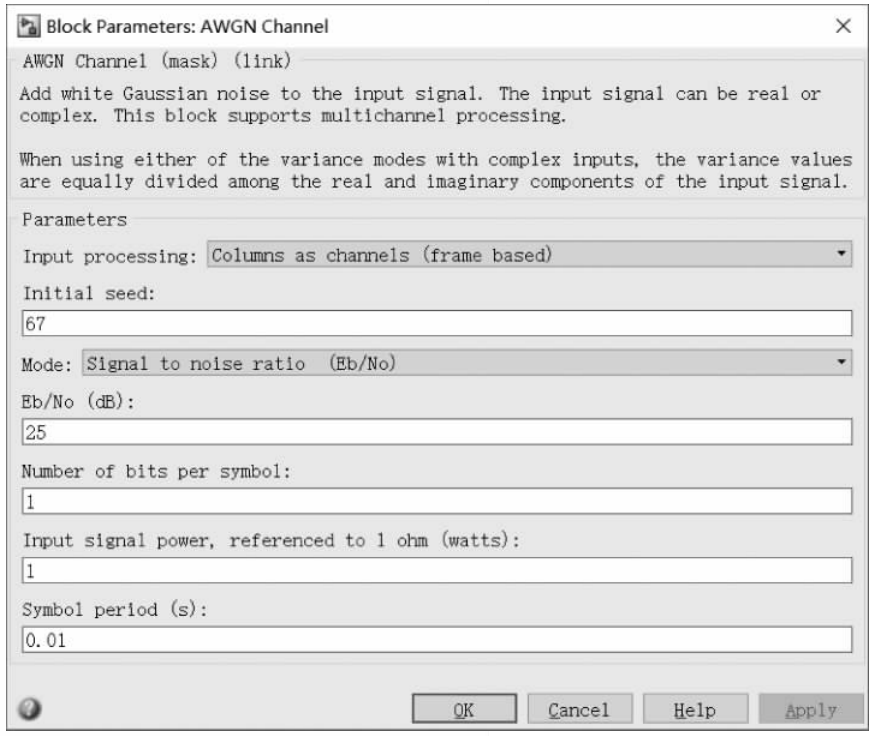

图 5-40　AWGN Channel 模块参数设置

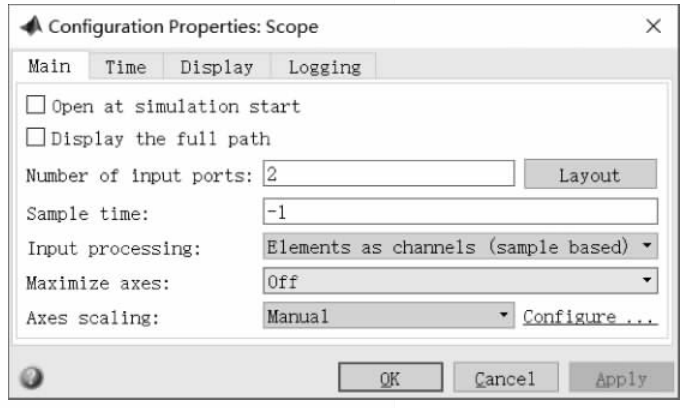

图 5-41　示波器模块参数设置

(3) 设置仿真参数。

将仿真时间设置为 0～10s,固定步长求解器,步长为 0.001,效果如图 5-42 所示。

(4) 运行仿真,仿真效果如图 5-43 所示。上面一个为输入信号进入信道前的波形,下面一个为输入信号进入信号后的波形。

设信道输入符号集合为$\chi=\{x_1,x_2,\cdots,x_j,\cdots,x_N\}$,并设信道输出的符号集合为 $\gamma=\{y_1,y_2,\cdots,y_i,\cdots,y_M\}$,在发送符号 x_j 的条件下,相应接收符号为 y_i 的概率记为 $P(y_i|x_j)$,称为信道转移概率。由信道转移概率构成信道转移概率矩阵,记为:

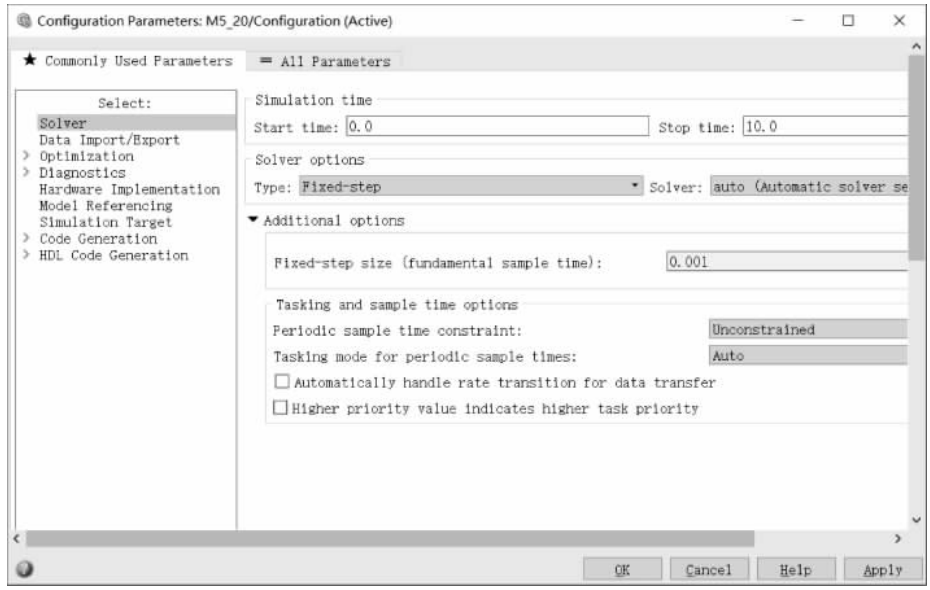

图 5-42　仿真参数设置

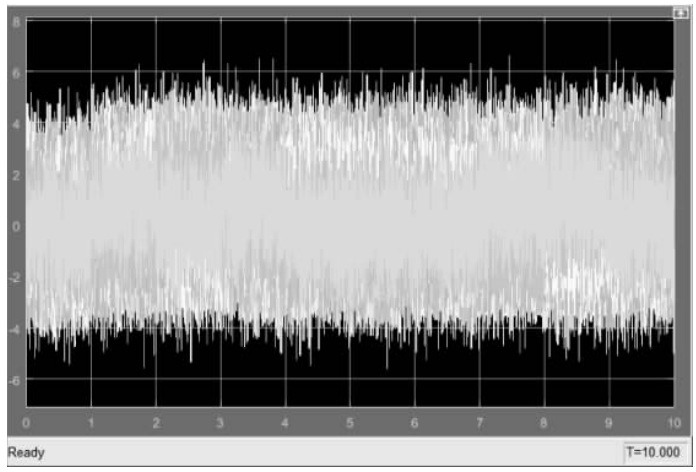

图 5-43　仿真结果

$$P=[P(y_i \mid x_j)]=\begin{bmatrix} P(y_1 \mid x_1) & \cdots & P(y_1 \mid x_N) \\ \vdots & \ddots & \vdots \\ P(y_M \mid x_1) & \cdots & P(y_M \mid x_N) \end{bmatrix}$$

二进制对称信道(BSC)是离散无记忆信道的一个特例，其输入输出符号集合分别为 $\chi=\{0,1\}$，$\gamma=\{0,1\}$。传输中由 0 错为 1 的概率与由 1 错为 0 的概率相等，设为 p。那么，二进制对称信道(BSC)的信道转换概率矩阵为：

$$P=\begin{bmatrix} 1-p & p \\ p & 1-p \end{bmatrix}$$

人们也经常用信道概率转换图来等价地表示离散无记忆信道,例如二进制对称信道,如图 5-44 所示。

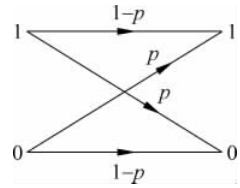

图 5-44　二进制对称信道模型

【例 5-21】 设传输错误概率为 0.013,构建通信系统,统计误码率。要求传输信号为二进制单极性信号,传输速率为 1000bps。

(1) 根据题意,建立如图 5-45 所示的通信系统模型。

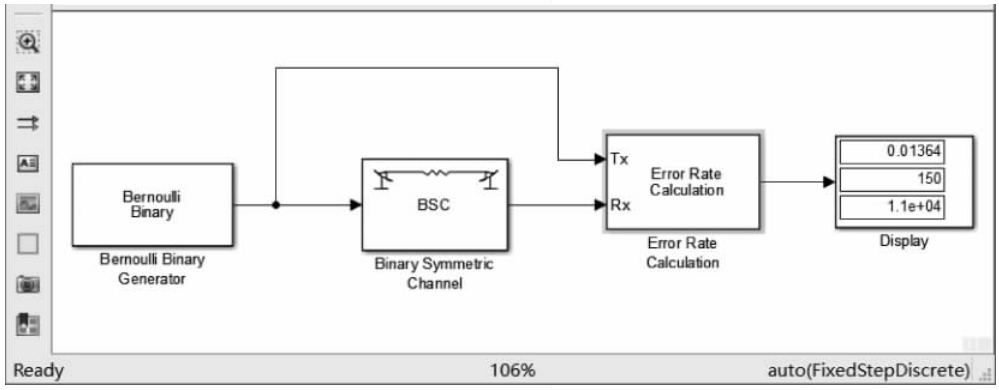

图 5-45　建立的通信系统模型

(2) 模块参数设置。双击图 5-45 中的 Bernoulli Binary Generator 模块,该模块产生速率为 1000bps 的二进制单极性信号,因此,设置产生零的概率为 0.5,初始种子随意设置,采样时间为 0.001。双击图 5-45 中的 Binary Symmetric Channel 模块,设置误码率为 0.013,初始种子随意设置,如图 5-46 所示。

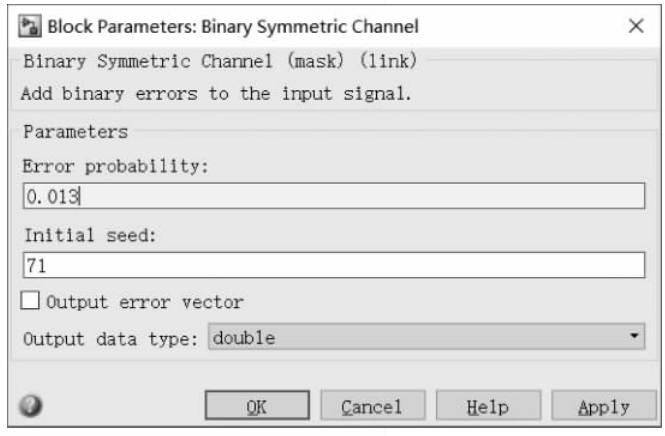

图 5-46　Binary Symmetric Channel 模块参数设置

双击图 5-45 中的 Error Rate Calculation 模块,用来计算误码率。接收延时和计算延时均设为 0,计算模式设为 Entire frame 全帧计算模式,数据输出设为 Port 端口输出(也可以设为 Workspace,输出到 MATLAB 工作空间),如图 5-47 所示。

(3) 设置仿真参数。将仿真时间设置为 0～10s,固定步长求解器,步长为 0.001。

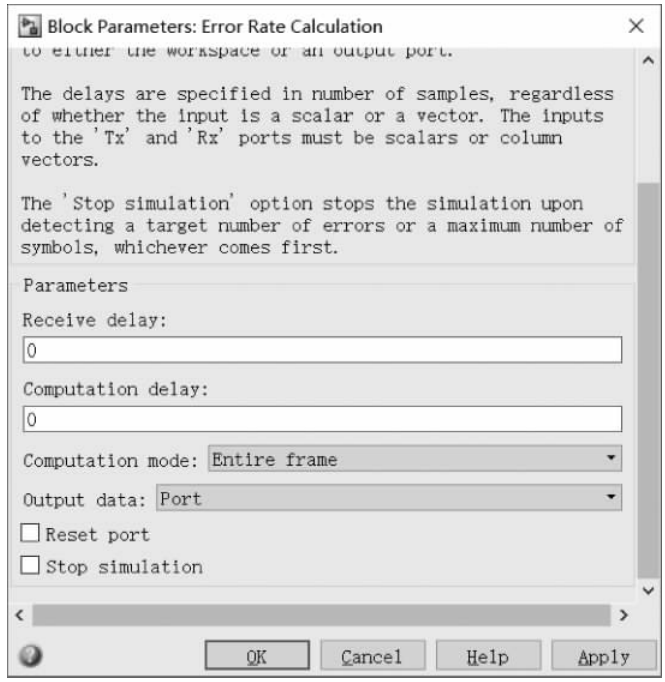

图 5-47　Error Rate Calculation 模块参数设置

（4）运行仿真。仿真结果显示在 Display 模块上，如图 5-45 所示。Display 模块上显示结果有三个，分别代表误码率、总误码数目以及总统计码字数目。从图 5-45 中可看出，Bernoulli Binary Generator 模块输出误码率为 0.013 64，总误码数为 150，总统计码字数为 1.1e+04。

注意：一般，当误码数达到 100 以下，就可以认为统计误码率是足够精确的。

【例 5-22】 对 A 律压缩扩张模块和均匀量化器实现非均匀量化过程的仿真，观察量化前后的波形。

（1）建立模型。根据题意，仿真模型如图 5-48 所示。

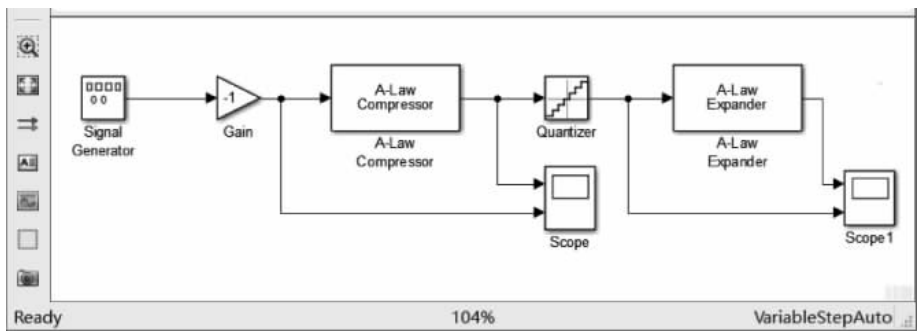

图 5-48　A 律压缩和均匀量化器实现非均匀量化的仿真模型

（2）模块参数设置。双击图 5-48 中的 A-Law Compressor 模块及 A-Law Expander 模块，设量化器的量化级为 8，A 律压缩系数为 87.6。双击图 5-48 中的 Singal Generator 模块，设置信号为 0.5Hz 的锯齿波，幅度为 1。

(3) 仿真参数。其仿真时间为 0～10s,步长采用默认值。

(4) 运行仿真。设置完成后,对模型进行运行仿真,得到仿真效果如图 5-49 所示。

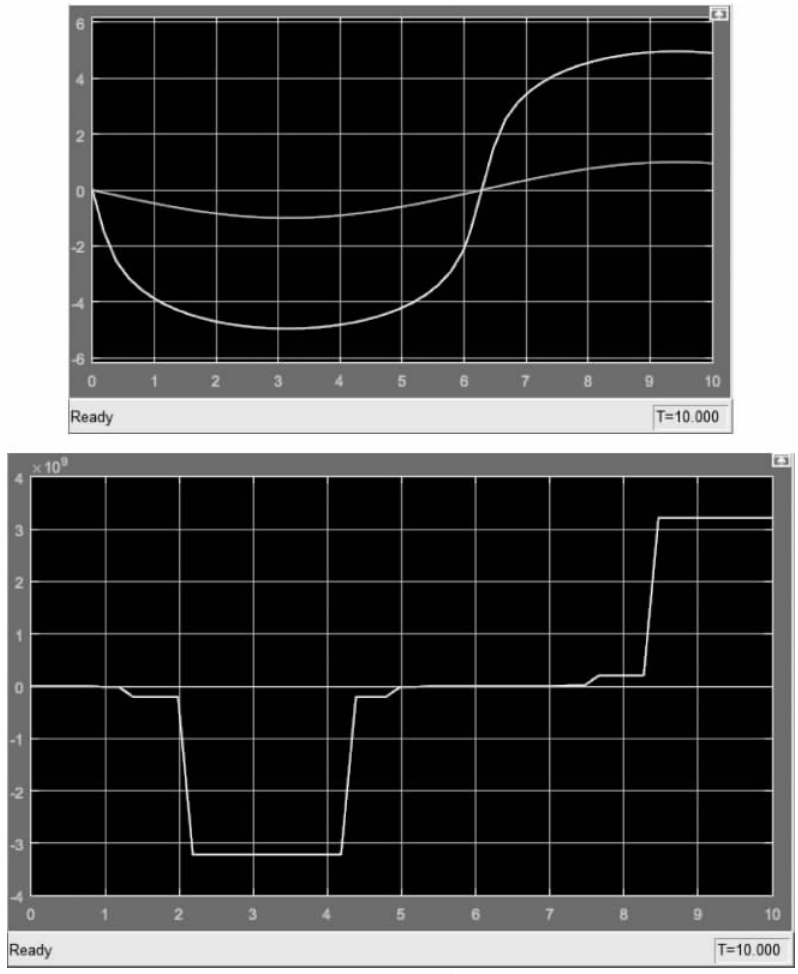

图 5-49 A 律压缩和均匀量化器实现非均匀量化的仿真结果

5.7.3 MATLAB/Simulink 信道实例

【例 5-23】 设计一个 13 折线近似的 PCM 编码模型,使它能够对取值在[－1,1]内的归一化信号样值进行编码。

(1) 建立仿真模型。

测试模型和仿真结果如图 5-50 所示,其中 PCM 编码子系统就是图 5-50 中虚线所围部分。PCM 解码器中首先分离并行数据中的最高位(极性码)和 7 位数据,然后将 7 位数据转换为整数值,再进行归一化、扩张后与双极性的极性码相乘得出解码值。可以将该模型中虚线所围部分封装为一个 PCM 解码子系统备用。

(2) 量化值的编码。

量化以后得到的可以进行线性量化的值,A 律 PCM 的编码表(正值)如表 5-1 所示。

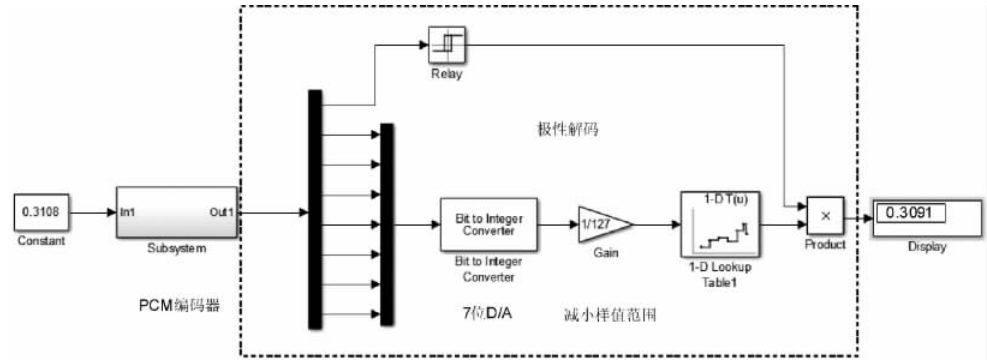

图 5-50　13 折线近似的 PCM 解码器测试模型和仿真结果

表 5-1　国际标准 PCM 对数 A 律量化表

线段编号	间隔数×量化间隔	线段终点值	分层电平编号	分层电平值	编码器输出	量化电平值	量化电平号
7	16×128	4096 … … 2048	(128) 127 … 112	(4069) 3968 … 2048	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 … 1 1 1 1 0 0 0 0	4032 … … 2112	128 … … 113
6	16×64	… … 1024	… 97 96	… 1088 1024	… 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 0	… … 1056	 … 97
5	16×32	… … 512	… 81 80	… 544 512	… 1 1 0 1 0 0 0 1 1 1 0 1 0 0 0 0	… … 528	… … 81
4	16×16	… … 256	… 65 64	… 272 256	… 1 1 0 0 0 0 0 1 1 1 0 0 0 0 0 0	… … 264	… … 65
3	6×8	… … 128	… 49 48	… 136 128	… 1 0 1 1 0 0 0 1 1 0 1 1 0 0 0 0	… … 132	… … 49
2	16×4	… … 64	… 33 32	… 68 64	… 1 0 1 0 0 0 0 1 1 0 1 0 0 0 0 0	… … 66	… … 33
1	32×2	… … 0	… 2 0	… 2 0	… 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0	… … 1	… … 1

从表 5-1 可以得出对应量化值的编码。在本例中，用 $x(i)$来表示采样值，$y(i)$来表示将采样值 $x(i)$进行对数压缩后的值，这样 $x(i)$对应表 5-1 中的分层电平值和量化电平值，而 $y(i)$对应表 5-1 中的分层电平值编号和量化电平编号。

利用13线性法得到的量化编码,MATLAB程序为:

```
>> z = zhe13(y);
>> pcmcode(z);
```

输出结果为:

```
f =
    1   0   0   0   0   0   0   0   -128   -115   0
    0   1   1   1   0   0   1   1      0      0   0
    0   1   1   1   1   1   1   1      0      0   0
    0   1   1   1   1   1   1   1      0      0   0
    0   1   1   1   0   0   1   1      0      0   0
    0   0   0   0   0   0   0   0      0      0   0
    1   1   1   1   0   0   1   1      0      0   0
    1   1   1   1   1   1   1   1      0      0   0
    1   1   1   1   1   1   1   1      0      0   0
    1   1   1   1   0   0   1   1      0      0   0
    1   0   0   0   0   0   0   0      0      0   0
```

对数压缩特性得的编码:

```
>> z = apcm(y,90.88);
>> f = pcmcode(z);
```

输出结果为:

```
f =
    1   0   0   0   0   0   0   0   -126   -115   0
    0   1   1   1   0   0   1   1      0      0   0
    0   1   1   1   1   1   1   0      0      0   0
    0   1   1   1   1   1   1   0      0      0   0
    0   1   1   1   0   0   1   1      0      0   0
    0   0   0   0   0   0   0   0      0      0   0
    1   1   1   1   0   0   1   1      0      0   0
    1   1   1   1   1   1   1   0      0      0   0
    1   1   1   1   1   1   1   0      0      0   0
    1   1   1   1   0   0   1   1      0      0   0
    1   0   0   0   0   0   0   0      0      0   0
```

可以看出对两种量化得到的编码是一样的,13折线近似效果是相当好的。

在运行程序过程中,需要调用用户自定义编写的zhe13.m函数和pcmcode.m函数,它们的源代码分别如下:

```
function y = zhe13(x)
% 本函数实现国际通用的PCM量化A律13折线特性近似
% x为输入的序列,变换后的值赋给序列y
x = x/max(x);                                  % 求出序列的最大值,并同时归一化
z = sign(x);                                   % 求的每一序列的值的符号
x = abs(x);                                    % 取序列的绝对值
for i = 1:length(x),                           % 直接将序列的绝对值量化
    if ((x(i)>=0)&(x(i)<1/64)),                % 序列值位于第1和第2折线
        y(i) = 16 * x(i);
    else
```

```
        if(x(i)>=1/64 & x(i)<1/32),                    %序列值位于第3折线
            y(i)=8*x(i)+1/8;
        else
            if(x(i)>=1/32 & x(i)<1/16),                %若序列值位于第4折线
                y(i)=4*x(i)+2/8;
            else
                if(x(i)>=1/16 & x(i)<1/8),             %若序列位于第5折线
                    y(i)=2*x(i)+3/8;
                else
                    if(x(i)>=1/8 & x(i)<1/4),          %若序列位于第6折线
                        y(i)=x(i)+4/8;
                    else if (x(i)>=1/4 & x(i)<=1/2),%若序列值位于第7折线
                            y(i)=1/2*x(i)+5/8;
                        else if(x(i)>=1/2 & x(i)<=1),%若序列值位于第8折线
                                y(i)=1/4*x(i)+6/8;
                            end
                        end
                    end
                end
            end
        end
    end
end
y=z.*y;    %重新将符号代回序列中   %循环结束
function f=pcmcode(y)
% 本函数实现将输入的值(已量化好)编码输入y为量化后的序列,
%其值应该在0到1之间
%定义出一个二维数组,第一行的8位代表了对应的输入值的编码(8位)
f=zeros(length(y),8);
z=sign(y);                          %得到输入序列的符号,确定编码的首位
y=y*128;                            %将序列值扩展到0到128之间,便于编码
f=fix(y);                           %将计算取整
y=abs(y);                           %只计算绝对值的编码
for i=1:length(y),
    if (y(i)==128),                 %如果输入为1,得到128,为避免出现编码位为2的错误
        y(i)=127.999;               %将其值近似为127.999
    end
end
for i=1:length(y),                  %下面的一段循环时将十进制转化为二进制数
    for j=6:-1:0                    %分别计算序列除以从64到1的数的商
        f(i,8-j)=fix(y(i)/(2^j));
        y(i)=mod(y(i),(2^j));
    end
end
for i=1:length(y),
    if (z(i)==1),                   %输入值是负数
        f(i,1)=0;                   %首位取0
    else
        f(i,1)=1;                   %输入是正数,首位取1
    end
end
f                                   %显示编码结果
```

注意：负值的量化与正值几乎完全相同，区别在于将编码的首位由1改为0。