第5章 差错控制系统的MATLAB/Simulink仿真

差错控制技术是指在发送端加入差错控制码元，即监督码元，利用监督码元和信息码元之间的某种确定关系来进行自动纠错和检错。一般来说，增加的监督码元越多，传输效率就越低，纠错和检错能力反而就越强。所以，差错控制技术是通过降低传输效率来提高信息在通信系统中传输的可靠性。本章介绍几种常用差错控制编码的基本原理，并通过 MATLAB/Simulink 对其进行仿真分析。

5.1 基于线性分组码的差错控制系统仿真

对信源编码器输出的序列进行分组，并对每一组独立变换，称为分组码，记为(n,k)码，其中 k 表示每分组输入符号数，n 为编码输出符号数。编码后的码组具有抗信道干扰的能力。若这种变换是线性变换，则称变换后的码组为线性分组码；若变换是非线性的，则称变换后的码组为非线性分组码。常用的是线性分组码。

线性分组码具有如下两个性质：

(1) 线性(包含全零码字，封闭性)；

(2) 最小码距等于除零码外的码字的最小码重。

为了具体说明线性分组码的基本原理，以(7,3)线性分组码为例。设(7,3)线性分组码为 $\boldsymbol{C}=(c_6,c_5,c_4,c_3,c_2,c_1,c_0)$，其中 c_6,c_5,c_4 为信息位，c_3,c_2,c_1,c_0 为监督位。将信息流分成每 3 位为一组，构成原码，即 $\boldsymbol{A}=(a_2,a_1,a_0)$，按下列线性方程进行编码：

$$\begin{cases} c_6=a_2 \\ c_5=a_1 \\ c_4=a_0 \\ c_3=a_1\oplus a_0 \\ c_2=a_2\oplus a_1 \\ c_1=a_2\oplus a_1\oplus a_0 \\ c_0=a_2\oplus a_0 \end{cases} \tag{5-1}$$

写成矩阵形式，则可表示为：

$$\boldsymbol{C}=[c_6,c_5,c_4,c_3,c_2,c_1,c_0]=[a_2,a_1,a_0]\begin{bmatrix}1&0&0&0&1&1&1\\0&1&0&1&1&1&0\\0&0&1&1&0&1&1\end{bmatrix}=\boldsymbol{A}\cdot\boldsymbol{G} \tag{5-2}$$

式中，$\boldsymbol{A}$ 为原码；$\boldsymbol{C}$ 为生成的线性分组码；$\boldsymbol{G}$ 为生成线性分组码的生成矩阵。一般情况下，生成(n,k)线性分组码的生成矩阵大小为 $k\times n$。

由式(5-1)进一步变换可以得到监督位与信息位的关系：

$$\boldsymbol{H}\cdot\boldsymbol{G}^{\mathrm{T}}=\boldsymbol{0}^{\mathrm{T}} \tag{5-3}$$

式中，H 为监督矩阵。监督矩阵中的每一个行向量与线性分组码中的任一码元内积为 0，并且其中每一个行向量都线性无关。

1. 线性分组码的编码译码仿真

采用 MATLAB 仿真程序完成对(7,4)线性分组码的编码、解码,其中信息位和检验位的约束关系为 c1＝a1；c2＝a2；c3＝a3；c4＝a4；c5＝a1＋a2＋a3；c6＝a2＋a3＋a4；c7＝a1＋a2＋a4；生成矩阵为 G,校验矩阵为 H,原码为 A,生成码字为 C,纠错后的码字为 Cr。

```
clear all;
G1 = eye(4);                                    % 生成 4×4 单位阵
G2 = [1,0,1;1,1,1;1,1,0;0,1,1];                 % 约束关系
G = [G1,G2];                                    % 生成矩阵 G
fprintf('生成矩阵为:G = ')
disp(G);
A = [0,0,0,1;0,0,1,0;0,0,1,1;0,1,0,0;0,1,0,1;0,1,1,0;0,1,1,1;1,0,0,0;1,0,0,1;1,0,1,0;
1,0,1,1;1,1,0,0;1,1,0,1;1,1,1,0;1,1,1,1;]; % A = [a1,a2,a3,a4]编码的原码
fprintf('原码为:A = ')
disp(A);
C1 = A * G;
C = mod(C1,2);                                  % 模 2 运算
fprintf('输出的编码为:C = ')
disp(C);
H = gen2par(G)                                  % 生成校验矩阵
fprintf('校验矩阵为:H = ')

%% 以下输入接收到的码字,译出原码
Rev = input('请输入 7 位接收码字,用空格隔开:','s');
Rev = str2num(Rev)                              % 接收到的码字
S1 = Rev * (H');                                % S 为校阵子
S = mod(S1,2);
E = [1,1,1,1,1,1,1];
for i = 1:7;                                    % 取出 H 中的每一列,与 S 相加
  Hi = H(:,[i]);
  Sum = S + Hi';
  Sum = mod(Sum,2);
  if (all(Sum(:) == 0));                        % 如果 S 与 H 的第 i 列之和 Sum 为 0 矩阵,则表示
                                                % Rev 中第 i 个码字有误
      fprintf('接收码字中错误码位是第:');
      disp(i)
  else
      E(1,i) = 0;
  end;
end;
Cr = mod((Rev + E),2);
fprintf('正确接收码字:Cr = ');
disp(Cr);
```

程序运行结果：

```
生成矩阵为:
G =
     1     0     0     0     1     0     1
     0     1     0     0     1     1     1
     0     0     1     0     1     1     0
     0     0     0     1     0     1     1
原码为:
A =
     0     0     0     1
     0     0     1     0
     0     0     1     1
     0     1     0     0
     0     1     0     1
     0     1     1     0
     0     1     1     1
     1     0     0     0
     1     0     0     1
     1     0     1     0
     1     0     1     1
     1     1     0     0
     1     1     0     1
     1     1     1     0
     1     1     1     1
输出的编码为:
C =
     0     0     0     1     0     1     1
     0     0     1     0     1     1     0
     0     0     1     1     1     0     1
     0     1     0     0     1     1     1
     0     1     0     1     1     0     0
     0     1     1     0     0     0     1
     0     1     1     1     0     1     0
     1     0     0     0     1     0     1
     1     0     0     1     1     1     0
     1     0     1     0     0     1     1
     1     0     1     1     0     0     0
     1     1     0     0     0     1     0
     1     1     0     1     0     0     1
     1     1     1     0     1     0     0
     1     1     1     1     1     1     1
校验矩阵为:
H =
     1     1     1     0     1     0     0
     0     1     1     1     0     1     0
     1     1     0     1     0     0     1
请输入7位接收码字,用空格隔开:0 1 1 1 1 0 1
接收码字中错误码位是第:     2
正确码字为:Cr =     0     0     1     1     1     0     1
```

2. 线性分组码信号的传输仿真

采用 MATLAB 和 Simulink 交互的方式仿真实现线性分组码编码后的信号在通信

系统中的传输。Simulink 仿真模型如图 5-1 所示。模型中 Bernoulli Binary Generator(伯努利二进制序列产生器)模块产生的信源序列经过 Binary Linear Encoder(二进制线性编码器)进行线性分组码编码。编码后的序列经过 Binary Symmetric Channel(二元对称信道)传输,该信道具有误码概率。在接收端进行译码。译码后的序列和信源序列输入 Error Rate Calculation(误码率统计)模块,统计接收端误码率。

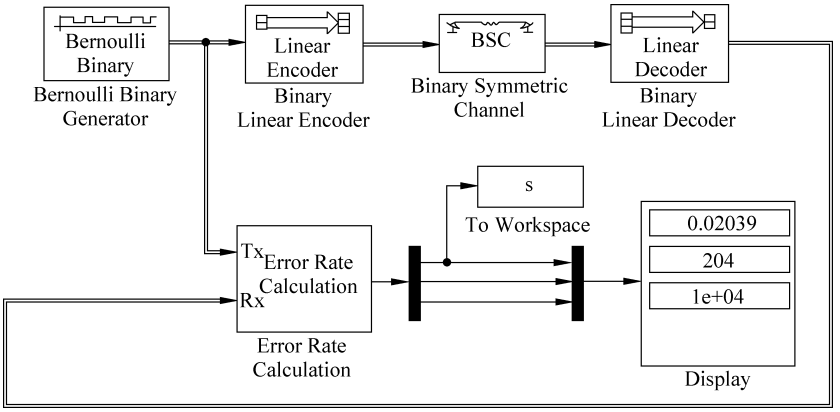

图 5-1　线性分组码的 Simulink 仿真模型

模型中各模块的主要参数设置见表 5-1。

表 5-1　线性分组码 Simulink 仿真参数

模块名称	参数名称	参数值
Bernoulli Binary Generator(伯努利二进制序列产生器)	Probability of zero	0.5
	Initial seed	10000
	Sample time	1
	Frame-based output	Checked
	Samples per frame	4
Binary Linear Encoder(二进制线性编码器)	Generator matrix	[[1 1 0; 0 1 1; 1 1 1; 1 01]eye(4)]
Binary Symmetric Channel(二元对称信道)	Error probability	errB
	Initial seed	2137
Error Rate Calculation(误码率统计)	Receive delay	0
	Computation delay	0
	Computation mode	Entire frame
	Output data	port
To Workspace(工作区间)	Variable name	s
	Limit data point to last	inf
	Decimation	1
	Sample time	−1
	Save format	Array

本例仿真采用 MATLAB 和 Simulink 交互的方式，分析经过线性分组码编码的信号在不同误码率的信道中传输后，接收端收到信号的误码率情况，代码如下：

```
Clear;
x = 0:0.01:0.05;
y = x;
hold off;
fori = 1:length(x)
    errB = x(i);
    sim('m');                                    % 实现 MATLAB 和 Simulink 模块的交互
    y(i) = mean(s);
end
plot(x,y);
hold on
grid on;
xlabel('信道误码率');
ylabel('接收端误码率 Pe ');
```

图 5-2 为线性分组码的 MATLAB 和 Simulink 交互仿真结果。通过对图中信道误码率和接收端信号误码率的分析，可以看出使用线性分组码的差错控制仿真编码以后，差错率明显下降，比如信道误码率在 0.04 时，接收端信号误码率不足 0.015。

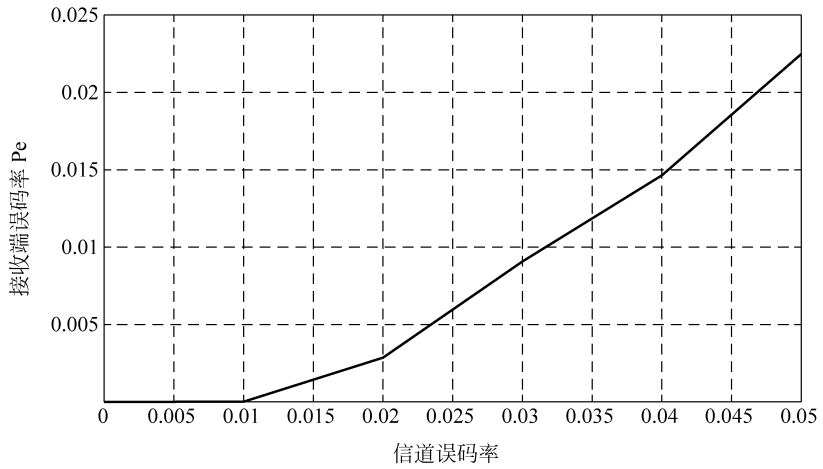

图 5-2　线性分组码的传输仿真结果

5.2　基于循环码的差错控制系统仿真

设 $\boldsymbol{C}$ 是某(n,k)线性分组码的码字集合，如果将任一码字 $c=(c_{n-1},c_{n-2},\ldots,c_0)$ 向左移一位，记为 $c^{(1)}=(c_{n-2},c_{n-3},\cdots,c_0,c_{n-1})$ 也属于码集 $\boldsymbol{C}$，则该线性分组码为循环码。循环码的特点是具有循环性，即任何许用码字的循环移位仍然是一个许用码字。

对于一个任意长为 n 的码字 $c=(c_{n-1},c_{n-2},\cdots,c_1,c_0)$，用多项式的形式表示，即

$c(x)=c_{n-1}x^{n-1}+c_{n-2}x^{n-2}+\cdots+c_1x+c_0$，此多项式即为码多项式，系数不为 0 的 x 的最高次数称为多项式 $c(x)$ 的次数或阶数。在进行码多项式简单运算时，所得系数需进行模 2 运算。这就是循环码的编码原理。

构造循环码主要是要找出一组线性分组码的生成矩阵，将信息位与生成矩阵 $\boldsymbol{G}$ 相乘得到码字。由于循环码的码字多项式是生成多项式 $g(x)$ 的倍式，且根据线性码生成矩阵的特性，(n,k) 码的生成矩阵可以由 (n,k) 码 k 个不相关的码组构成。根据以上两点，可以挑选出 k 个线性不相关的循环码的码多项式。

接收端检测时，用码多项式 $y(x)$ 除以生成多项式 $g(x)$。若不能除尽，则说明接收码不属于循环码，在传输过程中发生错误，即 $\frac{y(x)}{g(x)}=Q(x)+\frac{R(x)}{g(x)}$。因此，可用其余式 $R(x)$ 是否为零来判断接收码组中是否有错。但当码字中错误位数超过循环码的检错能力时，接收码多项式仍有可能被 $g(x)$ 整除，利用以上方法则不能检测出误码。

1. *循环码的编码仿真*

下面程序完成(7,4)循环码的编码。代码中 cyclpoly(n,k,'all')返回(n,k)循环码的所有生成多项式(1 个生成多项式为返回矩阵的 1 行)；cyclgen(n,g)返回循环码的监督矩阵和生成矩阵，其中 g 为生成多项式向量；rem(msg * G,2)返回循环码的所有需用码组，其中 G 为生成矩阵，msg 为信息矩阵。

```
clear all;
close all;
n = 7;
k = 4;
p = cyclpoly(n,k,'all');                    % 产生循环码的生成多项式
[H,G] = cyclgen(n,p(1,:));                  % 产生循环码的生成矩阵和校验矩阵
Msg = [0 0 0 0;0 0 0 1;0 0 1 0;0 0 1 1;0 1 0 0;0 1 0 1;0 1 1 0;0 1 1 1;
    1 0 0 0;1 0 0 1;1 0 1 0;1 0 1 1;1 1 0 0;1 1 0 1;1 1 1 0;1 1 1 1 ];
C = rem(Msg * G,2)
```

运行结果：

```
C =
     0     0     0     0     0     0     0
     0     1     1     0     0     0     1
     1     1     0     0     0     1     0
     1     0     1     0     0     1     1
     1     1     1     0     1     0     0
     1     0     0     0     1     0     1
     0     0     1     0     1     1     0
     0     1     0     0     1     1     1
     1     0     1     1     0     0     0
     1     1     0     1     0     0     1
     1     1     0     1     0     0     1
     0     1     1     1     0     1     0
     0     0     0     1     0     1     1
```

```
0    1    0    1    1    0    0
0    0    1    1    1    0    1
1    0    0    1    1    1    0
1    1    1    1    1    1    1
```

2. 循环码信号的传输仿真

循环码的 Simulink 仿真模型如图 5-3 所示。Bernoulli Binary Generator 模块产生的信源序列经过 Binary Cyclic Encode 编码后在 Binary Symmetric Channel 中传输。Error Rate Calculation 模块将接收端 Binary Cyclic Decode 译码后的序列和信源序列输入进行比较，统计误码率。

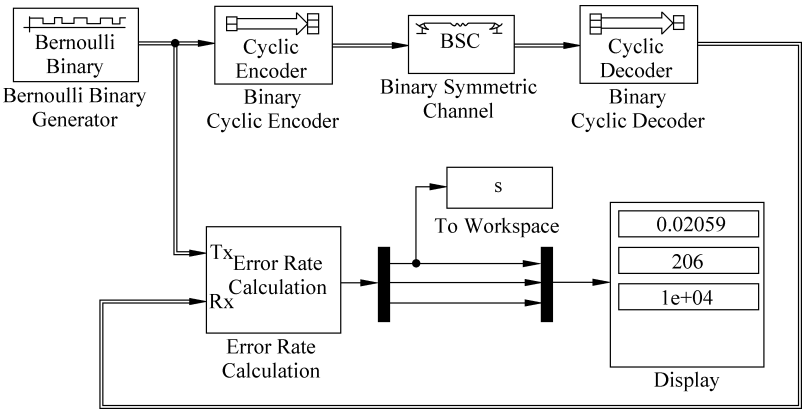

图 5-3 循环码 Simulink 仿真模型

表 5-2 是模型中各模块的主要参数设置。

表 5-2 循环码 Simulink 仿真参数

模块名称	参数名称	参数值
Bernoulli Binary Generator	Probability of zero	0.5
	Initial seed	10000
	Sample time	1
	Samples per frame	4
Binary Cyclic Encoder	Codeword length N	7
	Message length K	4
Binary Symmetric Channel	Error probability	errB
	Initial seed	2137
Binary Cyclic Decoder	Codeword length N	7
	Message length K	4
Error Rate Calculation	Receive delay	0
	Computation delay	0
	Computation mode	Entire frame
	Output data	port

续表

模块名称	参数名称	参数值
To Workspace	Variable name	s
	Limit data point to last	inf
	Decimation	1
	Sample time	−1
	Save format	Array

本例仿真同样采用 MATLAB 和 Simulink 交互的方式，利用 5.1 节线性分组码的 MATLAB 和 Simulink 交互代码调用图 5-3 循环码 Simulink 仿真模型，可以完成不同信道误码率下接收端的误码率统计，仿真结果见图 5-4。从图中可以看出，循环码同样降低了接收端的误码率，提高了通信质量。

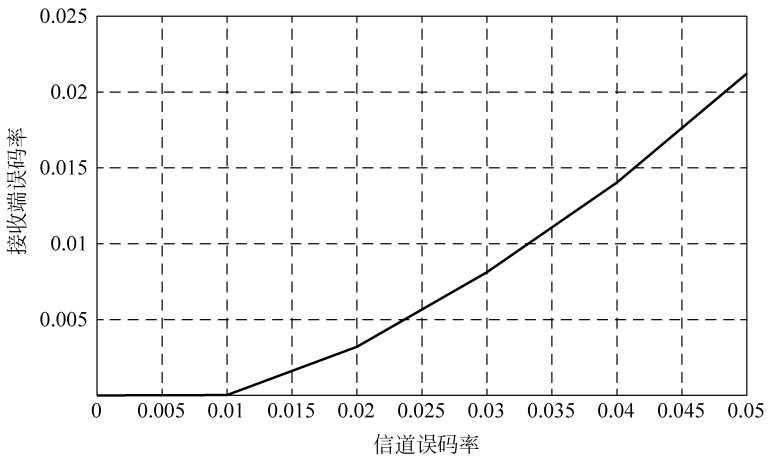

图 5-4　循环码信号传输的仿真结果

5.3　基于卷积码的差错控制系统仿真

卷积码是结合本码组和其他码组之间的关系来进行编码的。卷积码通常记作(n,k,N)的形式，其中 n 为编码后码组的长度，k 为输入的信息位的长度，N 为编码约束长度。约束长度就是编码过程中互相约束的码组个数。卷积码编码的码字不仅与本码组中 k 个码字有关，同时和前$(N-1)$组输入的信息码字有关。相互有关系的码字一共有 $N\times n$ 个。约束长度越长纠错能力就越强，但码率会因此降低。$R=k/n$ 是卷积码的码率，表示信息位在所有要传输的码字中占有的比重。卷积码的一般结构如图 5-5 所示。

卷积码中 k 和 n 的值通常比较小，延时也相应小，适合串行传输。

下面利用 MATLAB 和 Simulink 两种方式进行仿真，对输入信号进行卷积编码，经过 AWGN 信道传输，接收端解码后统计传输误码率。

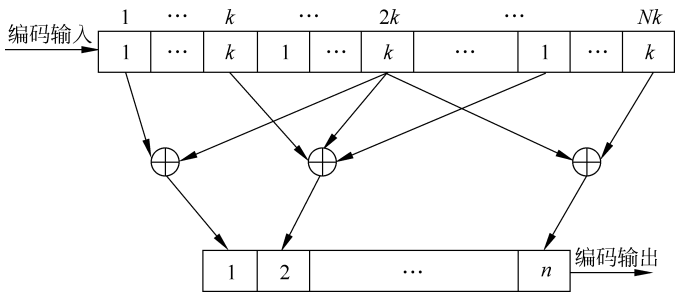

图 5-5 卷积码编码器结构

1. 卷积码的 MATLAB 仿真

下面采用(2,1,9)卷积码。程序利用 randint 函数产生信源序列,经过 convenc 卷积编码后,进行 BPSK 调制并在 AWGN 信道中传输。卷积码生成多项式为 G0=561(八进制),G1=753(八进制)。接收端 BPSK 解调后利用软判决滑动窗维特比译码,译码深度为 40。

```
clear all;
close all
SNRdB = 0:0.5:3;                          % 设置信噪比范围,0～3dB
declen = 40;                              % 译码深度
SNRnum = length(SNRdB);                   % 信噪比数目
iter = 10;                                % 每个信噪比下的迭代次数
for i = 1:SNRnum                          % 循环内计算每个信噪比下的误码率
  for j = 1:iter                          % 每个信噪比下迭代计算误码率 10 次,再求平均误码率
     trel = poly2trellis(9,[561 753]);    % 卷积码(2,1,9)网格图,约束长度为 9
     siglen = 1000000;                    % 设置信号长度
     msg = randi([0,1],siglen,1);         % 生成 0,1 序列,长度同信号长度
     encode = convenc(msg,trel,0);        % 从 0 状态开始作卷积编码
     I = 0.5 * ones(siglen * 2,1);
     y = encode - I;
     bpsk = sign(y);                      % BPSK 调制
     channelout = awgn(bpsk,SNRdB(i));    % 添加高斯白噪声,AWGN 信道
     debpsk = channelout * 0.5 + 0.5;     % 解调
     parti = 0:.15:.9;                    % 设置量化等级划分
     codebk = 0:7;                        % 设输出等级
     [x,qcode] = quantiz(debpsk,parti,codebk); % 量化,准备维特比软判决
     % 维特比译码,量化级数 = 2^3
     decode =  vitdec(qcode',trel,declen,'cont','soft',3);
     % 计算本次 BER
     [errorbit,errorrate(j)] = biterr(decode(declen + 1:end),msg(1:end - declen));
  end
   BER(i) = sum(errorrate)/iter;          % 求平均 BER
end
semilogy(SNRdB,BER);                      % 绘制不同信噪比下的误码率
```

```
xlabel('信噪比 SNR(dB)');
ylabel('误码率');
grid on;
```

图 5-6 为卷积码 MATLAB 仿真结果。从图中可以看出,接收端误码率随着信道信噪比的提高迅速降低。

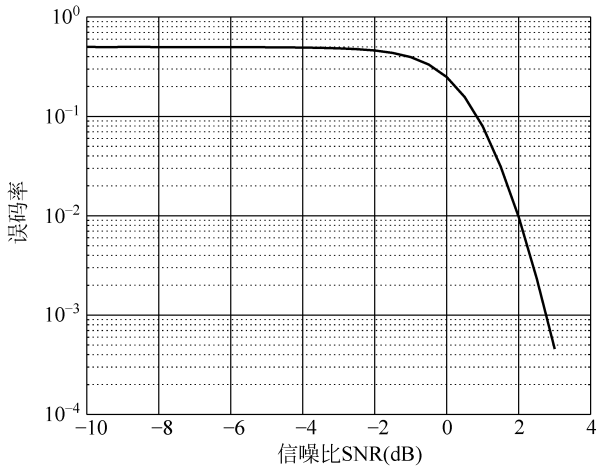

图 5-6　卷积码 MATLAB 仿真结果

2. 卷积码的 Simulink 仿真

图 5-7 为卷积码的 Simulink 仿真模型。信源序列由 Bernoulli Binary Generator 模块产生,经过 Convolution Encoder 编码后进行 BPSK 调制,输入 AWGN 信道。接收端将信号 BPSK 解调后,采用 Viterbi Decoder 译码。译码后的序列和信源序列一起输入 Error Rate Calculation 模块统计误码率。

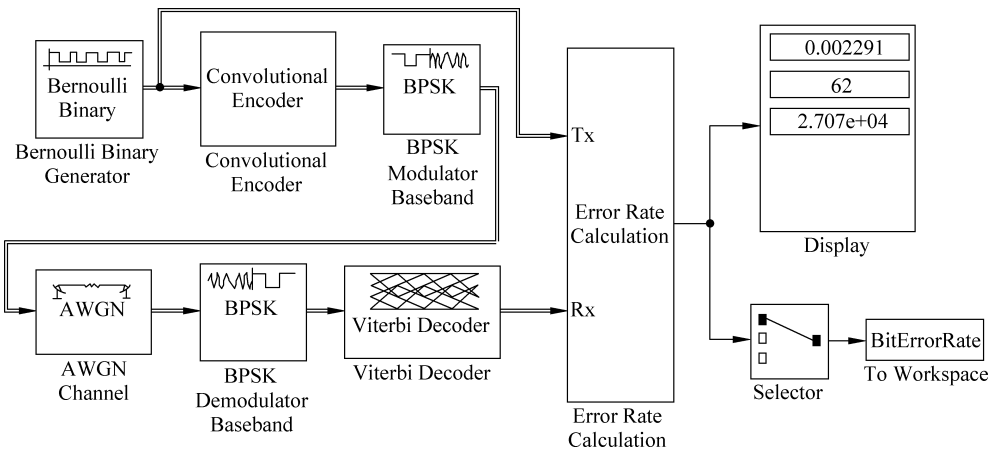

图 5-7　卷积码的 Simulink 仿真模型

表 5-3 为卷积码的 Simulink 仿真参数设置。

表 5-3 卷积码 Simulink 仿真参数

模块名称	参数名称	参数值
Bernoulli Binary Generator	Probability of zero	0.5
	Initial seed	61
	Sample time	0.02/268
	Frame-based output	Checked
	Samples per frame	268
BPSK Modulator Baseband (BPSK 调制)	Phase offset(rad)	0
	Samples per symbol	1
AWGN Channel(加性高斯白噪声信道)	Initial seed	67
	Mode	Signal to noise ratio(SNR)
	SNR(dB)	SNR
	Input signal power(watts)	1
Convolution Encoder	Trellis structure	STRUCTURE
	Operation mode	Truncated(reset every frame)
Selector(数据选通器)	Number of input dimensions	1
	Index mode	One-based
	Index Option	Index Vector(dialog)
	Index	[1]
Viterbi Decoder	Trellis structure	STRUCTURE
	Decision type	Hard Decision
	Operation mode	Truncated
	Traceback depth	34

本例通过 MATLAB 和 Simulink 交互的方式，完成不同码率、不同信噪比下接收端误码率的计算，代码如下：

```
x = -10:2;                              % x 表示信噪比
% 卷积方式分别取 1/3 卷积和 1/2 卷积
A = [poly2trellis(9, [557 663 711]),poly2trellis(7, [171 133])];
% 不同卷积方式、信噪比下重复调用图 5-7 的 Simulink 模型
for j = 1:2
    STRUCTURE = A(j);
        for i = 1:length(x)
        SNR = x(i);                     % 信道的信噪比依次取 x 中的元素
        sim('book_c');                  % 实现 MATLAB 和 Simulink 模块的交互
        y(j,i) = mean(BitErrorRate);    % 计算 BitErrorRate 的均值
    end
end
semilogy(x,y(1,:),'r',x,y(2,:),'b')     % 绘图，采用对数坐标
xlabel('信道信噪比'),ylabel('接收端误码率');
legend('1/3 卷积码','1/2 卷积码');
grid on
```

图 5-8 为卷积码的 MATLAB 和 Simulink 交互仿真结果。图中两条曲线分别表示不同码率的卷积码在不同信噪比下误码率性能,其中上面的曲线码率为 1/2,下面的曲线码率为 1/3。

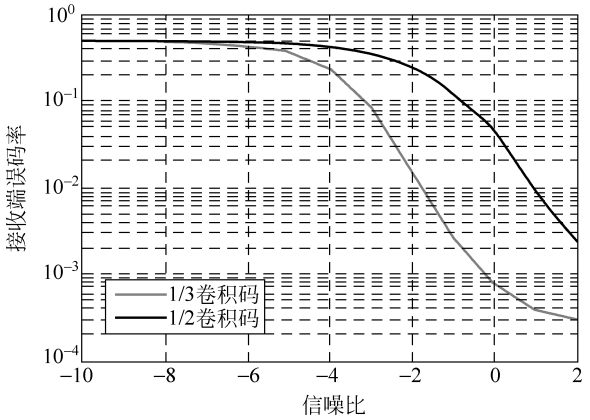

图 5-8 卷积码的仿真结果

5.4 基于循环冗余码的差错控制系统仿真

为了满足实际中对 n、k 取值的多样性要求,通常在传送码字的后部预留一定的空间,用于差错校验。循环冗余码是最常见的校验码,由信息位和校验位两部分组成。其编码方法如下:

(1) 移位:将 k 比特原码左移 r 位,形成 $k+r=n$ 位。

(2) 相除:用生成多项式 $g(x)$,以模 2 除的方式去除移位后的式子,得到的余数就是校验码。

接收端收到数据后对其进行 CRC 校验,方法是将整个数据串当作一个整体去除生成矩阵,判断循环冗余校验器产生的余数。若余数为零,则说明接收正确;否则接收错误,并不纠错。

循环冗余校验编解码的 MATLAB 仿真:

```
clear all;
close all;
 % 编码
crcmsg = input('请输入信息原码,空格隔开:','s');
crcmsg = str2num(crcmsg);
msglen = length(crcmsg);                    % 原码长度
crcgen = input('请输入生成多项式 g(x),空格隔开:','s');
                                            % 如 g(x) = x^5 + x^4 + x + 1,输入 1 1 0 0 1 1
crcgen = str2num(crcgen);
critlen = length(crcgen);
critbit = zeros(1,critlen);                 % 添加冗余比特
```

```
crcencode = [crcmsg zeros(1,critlen)];        %原码左移,左移位数为校验位长度
crcmsg = [crcmsg critbit];
divddvec = crcmsg;                            %被除数向量
for k = 1:msglen                              %开始循环计算长除得到最终余数
   added = zeros(1,msglen-k+1);               % 生成多项式向量左移位数,准备模2除
   divvec = [crcgen added];                   %除数向量
   if divddvec(1) == 0                        %被除数第1位为0,不必除
      divvec = zeros(1,length(divvec));
   end
   divddvec = bitxor(divvec,divddvec);        %模2除,等同异或
   divvec = crcgen;                           %恢复除数
   divddvec(1) = [];                          %去除被除数第1位
end
crclen = length(crcencode)
divdlen = length(divddvec);
divddvec = [zeros(1,crclen-divdlen),divddvec];      %余数序列
crcencode = crcencode + divddvec              %编码后序列
%msglen = length(crcencode);                  % 得到冗余编码的长度

%解码
divddvec = crcencode;                         % 余数初始化
crcdecode = crcencode;                        % 解码初始化

divnum = crclen-length(crcgen)+1;             % 长除的循环次数
for k = 1:divnum                              % 计算余数
   added = zeros(1,divnum-k);
   divvec = [crcgen added];                   %除数向量
   if divddvec(1) == 0                        %被除数第1位为0,不必除
      divvec = zeros(1,length(divvec));
   end
   divddvec = bitxor(divvec,divddvec);        %模2除,等同异或
   divvec = crcgen;                           % 恢复除数
   divddvec(1) = [];                          %去除被除数第1位
end
if sum(divddvec) == 0                         % 校验正确
   crcdecode = crcencode(1:divnum-1)
else
fprintf('校验错误');
end
```

运行结果：

```
请输入信息原码,空格隔开:1 1 1 1 0 0 0 1 0
请输入生成多项式 g(x),空格隔开:1 1 0 1 1
crclen =14
crcencode =1 1 1 1 0 0 0 1 0 0 0 1 0 0
crcdecode = 1 1 1 1 0 0 0 1 0
```

5.5 基于 LDPC 码的差错控制系统仿真

低密度校验码也称 LDPC 码,属于信道编码的一种,分为规则码和非规则码。因为低密度校验码编码方式不止一种,所以低密度校验码码的结构也会有所分类,码结构的不同或者选用低密度校验码译码算法不同,这些因素都会影响 LDPC 码的性能表现。在传统的信道中,如果信道中的噪声水平低于一个特定值,LDPC 码的传输错误概率会随着码长的增加而减小。理论上,如果码长变得无穷大,传输的错误概率会无限趋近于零,只是在物理上这个理念并不能被实现。受益于 LDPC 码本身的编码方式,其自带抗突发性错误这个特性,所以当其运用在通信系统中时是不需要交织器的,结果就是减少了通信系统中的时延,从而提高通信系统的性能。

低密度校验(LDPC)码具有以下特点:

(1) LDPC 码译码具有很低的复杂度,所以不会因为码长的增加而急剧增加运算量,译码算法不仅在理论上,在物理上也是可以实现的,并且实际译码的仿真性接近于理论分析。

(2) LDPC 码译码方法采用迭代译码算法,在物理上可以实现并行操作,并且译码速度高于其他编码。

(3) LDPC 码拥有大的吞吐量,可以提高传输系统的传输效率,尤其在硬件模块更能体现这个优点。

(4) LDPC 码的奇偶校验矩阵具有稀疏性,译码复杂度随着码长的改变而线性改变,所以不会出现码长过大而导致计算复杂度指数型增长这种情况。因为编码的码结构特性本身就自带抗突发差错这个能力,不像 Turbo 码那样需要交织器引入就拥有随机性,因此没有因交织器的存在而带来延时。

LDPC 码也是一种分组码,其校验矩阵只含有很少量非零元素。正是校验矩阵 $\boldsymbol{H}$ 的这种稀疏性,保证了译码复杂度和最小码距都只随码长呈现线性增加。校验矩阵 $\boldsymbol{H}$ 的每一行对应一个校验方程,每一列对应码字中的一比特。因此,对于一个二进制码,如果它有 m 个奇偶校验约束关系,码字的长度为 n,则校验矩阵是 $m \times n$ 的二进制矩阵。对于 $m \times n$ 维校验矩阵为 $\boldsymbol{H}$,当且仅当向量 $c=[c(1)\ c(2)\ \cdots\ c(m)]$ 满足

$$\boldsymbol{H} \cdot \boldsymbol{c}^{\mathrm{T}} = 0 \tag{5-4}$$

时,它才是该码的一个有效码字。

LDPC 码可通过伪随机的方法构造,需要在给定一些设计规范后得到所有 LDPC 码的集合,而不仅限于如何选择一些特殊的校验矩阵使之满足设计规范。LDPC 码常常通过 Tanner 图来表示,而 Tanner 图所表示的其实是 LDPC 码的校验矩阵。Tanner 图包含两类顶点:n 个码字比特顶点(称为比特节点),分别与校验矩阵的各列相对应;m 个校验方程顶点(称为校验节点),分别与校验矩阵的各行相对应。校验矩阵的每行代表一个校验方程,每列代表一个码字比特。所以,如果一个码字比特包含在相应的校验方程中,那么就用一条连线将所涉及的比特节点和校验节点连起来,所以 Tanner 图中的连线

数与校验矩阵中1的个数相同。图5-9是式(5-5)校验矩阵的Tanner图，其中上面的圆形节点表示校验节点，下面的圆形节点表示比特节点，黑线表示的是一个6循环。

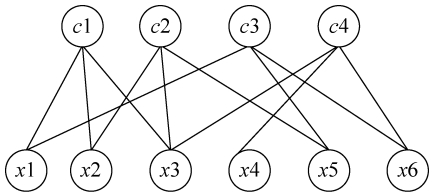

图5-9 LDPC码的Tanner图

$$\boldsymbol{H}=\begin{bmatrix}1&1&0&1&0&0\\0&1&1&0&1&0\\1&0&0&0&1&1\\0&0&1&1&0&1\end{bmatrix} \tag{5-5}$$

除了校验矩阵 $\boldsymbol{H}$ 是稀疏矩阵外，LDPC码本身与任何其他的分组码并无二致。如果现有的分组码可以被稀疏矩阵所表达，那么用于LDPC码的迭代译码算法也可以成功地移植到它身上。不同的是，LDPC码的设计是以构造一个校验矩阵开始的，然后才通过它确定一个生成矩阵进行后续编码。

LDPC编码方法采用高斯消元算法，算法流程图如图5-10所示。

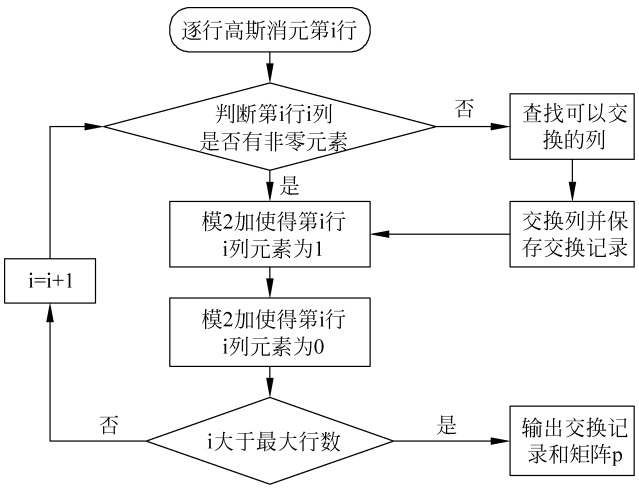

图5-10 高斯消元算法流程图

LDPC码的译码方法采用概率BP译码算法，流程图如图5-11所示。

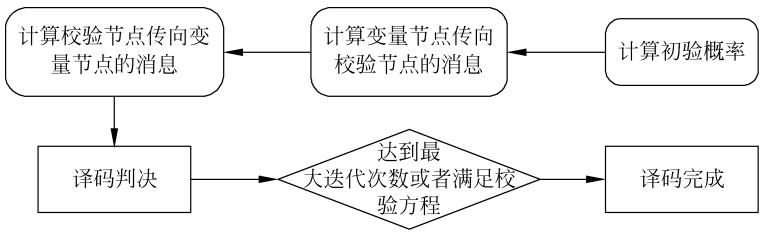

图5-11 概率BP译码算法过程

LDPC码的Simulink仿真模型如图5-12所示。Bernoulli Binary Generator模块采样频率是32400。信道模块和程序设计选用的信道模型一样，都是选用的加性高斯白噪

声信道(AWGN),调制/解调模块选用的是二进制相移键控(BPSK),误码率的计算就用Error Rate Calculation模块,这个模块的作用原理就是把信源的原始数据接入Tx端口,经过信道传输后的输出信号接入Rx端口,将输出信号与原始信号进行对比来计算误码率。对比后的数据就用输出端口接入Display模块来显示误码率的大小。

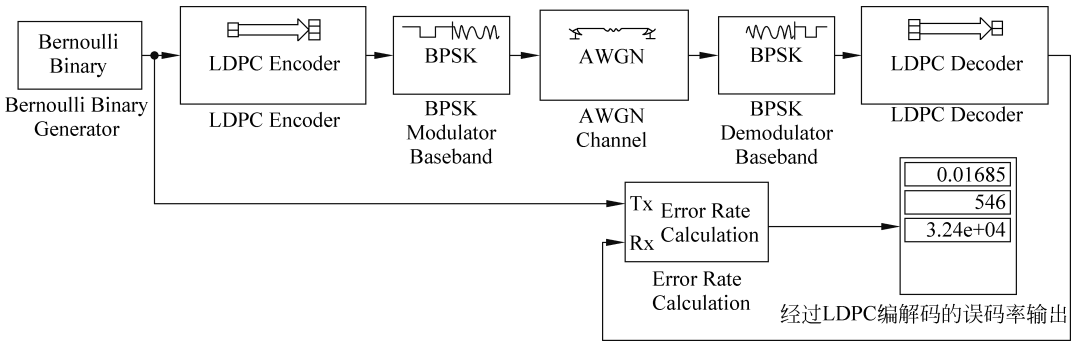

图 5-12　LDPC 码 Simulink 仿真模型

表 5-4 是模型中各模块的主要参数设置。

表 5-4　LDPC 码 Simulink 仿真参数

模 块 名 称	参 数 名 称	参　数　值
Bernoulli Binary Generator	Probability of zero	0.7
	Source of initial seed	61
	Sample time	1
	Samples per frame	32400
LDPC Encoder	Parity-check matrix	dvbs2ldpc(1/2)
BPSK Modulator	Phase offset (rad)	0
	Output type	double
AWGN Channel	Initial seed	67
	Mode	Signal to noise ratio(Eb/No)
	Eb/No(dB)	10
	Number of bits per symbol	1
	Input signal power(watts)	1
	Symbol period (s)	1
LDPC Decoder	Output format	Information part
	Decision type	Soft decision
	Number of interation(迭代次数)	50
BPSK Demodulator	Decision type	Soft decision
	Phase offset (rad)	0
Error Rate Calculation	Receive delay	0
	Computation delay	0
	Computation mode	Entire frame
	Output data	port

图 5-13 是信噪比为 10dB 时 LDPC 码 Simulink 仿真系统的输出结果图，从图中可以看出 LDPC 码经过 BPSK 调制/解调后输出消息的误码率较低。

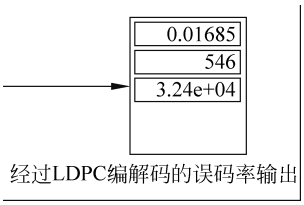

图 5-13　LDPC 码信号传输的仿真结果