Project Evaluation From

Project Title: OFDM system construction and the performance of OFDM system on different channels, different modulation methods and different coding methods.

1601019 Class: Name: Zhang Ziteng Criterion Good **Excellent** Poor Fair Relevance and Appropriateness of the Topic Proposal(Task description and planning) **Technical Quality** Workload and Complexity Writing and Representation (including the language) Adequate Illustrations or **Drawings Oral Representation** and Question **Answering Overall Rate** (Excellent ---------- Poor) 100 90 80 70 60 < 60

OFDM system construction and the performance of OFDM system on different channels, different modulation methods and different coding methods.

Zhang Ziteng 16010199044,

Class 1601019

Abstract: Orthogonal frequency division multiplexing (OFDM) is a multi-carrier parallel transmission technology which has the capability of anti-multipath interference. It is the key technology of the fourth-generation mobile communication system. This paper introduces OFDM system construction and the performance comparison of and different channels, different modulation modes and different coding modes in OFDM system.

Keywords: Orthogonal frequency division multiplexing (OFDM); computer simulation; MATLAB;16OAM;OPSK

1 introduction

OFDM is a frequency division multiplexed multicarrier transmission method, and each multiplexed signal (each carrier) is orthogonal. This technology has been recognized by the industry as the core technology of the new generation of wireless mobile communication systems and has high research value. The OFDM technology converts high-speed data streams into multiple parallel low-speed data streams by serial/parallel conversion, and then distributes them to subchannels on mutually orthogonal subcarriers of different frequencies for transmission. The biggest advantage of OFDM is its ability to combat frequency selective fading or narrowband interference while maintaining high spectrum utilization. This paper will discuss OFDM system construction and the performance of OFDM system on different channels, different modulation methods and different coding methods.

2 Principle of OFDM

The main idea of OFDM is to divide a channel into N subchannels, one carrier on each subchannel, called a subcarrier, and each subcarrier is orthogonal to each other. In implementation, a high-speed serial input data signal stream is converted into N parallel low-speed sub-data streams, and modulated onto each sub-carrier for transmission. After serial/parallel conversion, N parallel data are simultaneously output and modulated on N subcarriers. No. N subcarriers can be expressed as $f_n = f_c + n/T_s$. Then an OFDM signal at the m th moment can be expressed as

$$S_m(t) = \text{Re}\{\sum_{n=0}^{N-1} d(n)e^{j2\pi f_n t}\}, 0 < t < T$$

Since OFDM is emitted after the serial-parallel conversion of N symbols, the symbol rate is the original OFDM symbol rate1/N, so $T_s = NT_s$. When the guard interval between OFDM symbols is not considered,

$$S_{m}(t) = \operatorname{Re}\left\{\sum_{n=0}^{N-1} d(n)e^{j2\pi f_{n}t}\right\}$$

$$= \operatorname{Re}\left\{\sum_{n=0}^{N-1} d(n)e^{j2\pi \frac{n}{NT_{s}}t}e^{j2\pi f_{c}t}\right\}$$

$$= \operatorname{Re}\left\{X(t)e^{j2\pi f_{c}t}\right\}, 0 < t < T$$

Where, $\mathbf{X}(\mathbf{t}) = \sum_{n=0}^{N-1} d(n) e^{j2\pi\frac{n}{NT_s}t}$. $\mathbf{X}(\mathbf{t})$ is the complex equivalent baseband of the transmitted signal signal. Sampling $\mathbf{X}(\mathbf{t})$ with a sampling rate of $1/T_s$, then when $\mathbf{k} = \mathbf{k}T_s$, the sampled value $\mathbf{X}(\mathbf{k})$ satisfies $\mathbf{X}(\mathbf{k}) = \mathbf{X}(\mathbf{k}T_s) = \sum_{n=0}^{N-1} d(n) e^{j\frac{2\pi}{NT_s}nk}$. $\mathbf{X}(\mathbf{k})$ is exactly the result of the N-point inverse discrete Fourier transform (IDFT) of $\mathbf{d}(\mathbf{n})$. In practical applications, we can use the IFFT operation to complete the subcarrier modulation process of the OFDM complex equivalent baseband signal, and complete the demodulation process with Fast Fourier Transform (FFT).

3 OFDM system structure

The structure of an OFDM system is shown in Figure 1. The OFDM system consists of input data, channel coding, constellation mapping, training sequence insertion, serial-to-parallel conversion, IFFT, serial-to-parallel conversion, cyclic prefix, channel, de-cyclic prefix, and serial-to-parallel conversion, FFT, parallel-serial

conversion, channel estimation, demodulation, channel decoding, the data output.

3.1 Input data

Input data is the binary data that source generates to be transmitted. In MATLAB, we use the randi([0,1],1,

the received codeword. Since this RS code can correct t m-ary error code words, the RS code is particularly suitable for channels with burst errors. Here we use the RS code of 5 input 7 output. In MATLAB we use the gf() function to convert the input signal into a Galois field function, and then use *rsenc(data,7,5)* to generate a 11-input 15-output source RS code. We use

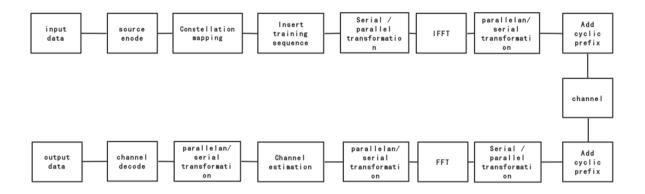


Figure 1 OFDM system structure

length) function to generate random numbers between 0and1 as the input data.

3.2 Channel coding and decoding

Source coding is a kind of transformation of source symbols for the purpose of improving communication effectiveness. Here we use RS coding and concatenated coding with RS (reed Solomon) coding and convolutional coding and compare the performance of these two coding methods.

RS encoding is an encoding method based on a finite field, finite field element by which is transformed configured generator polynomial g (x), and so that the codeword polynomial calculated for each information segment is g (x) of times. The RS code generator polynomial is generally selected as follows: g(x) = $(x-a)(x-a^2)...(x-a^{2t}) = \prod_{i=1}^{2t} (x-a^i)$, Where a^i is an element in $gf(2^m)$. If d(x) is used to represent the segment polynomial, the information polynomial c(x) can be constructed as follows. First calculate the business formula h(x)remainder: r(x): $\frac{x^{n-k}d(x)}{g(x)} = h(x)g(x) + r(x)$, Take the remainder r(x) as the check word; Then let c(x) = x $x^{n-k}d(x) + r(x)$, that is, place the information bits in the first half of the codeword, and supervise the bitwise manner in the second half of the codeword then we can $get \frac{c(x)}{g(x)} = \frac{x^{n-k}d(x)}{g(x)} + r(x) + r(x) = h(x)g(x).$ Therefore, the codeword polynomial c(x) must be divisible by the generator polynomial g(x). If the receiver detects that the remainder is not 0, it can be judged that there is an error in

rsdec(gf(yrsgs41,4), nn, kk) to decode the demodulated data.

The convolutional code is an error control code, and the convolutional code is represented by (n, k, L), where n is the number of output bits, k is the number of bits input, and L is the constraint length. As a memory error correcting code, the encoding rule is to encode k input bits into n output bits, and the encoded symbols are related not only to the input k bits but also to the previous L-1 group bits. Here we use a 1/2 feedback convolutional encoder (2, 1, 7) with a constraint length of 7.

For convolutional codes, we can use Viterbi decoding to decode. This decoding method obtains a decoded codeword closest to the encoded codeword based on the information that has been received. The coding criterion used to obtain such a codeword is maximum likelihood decoding.

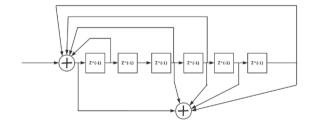


Figure 2 convolutional codes generator

In MATLAB, we use *trellis = poly2trellis(7,[133 171])* which is in Figure 2, This encoder has a constraint length of 7, a generator polynomial matrix of [133 171]. The first generator polynomial matches the feedback connection polynomial because the first output

corresponds to the systematic bits. The feedback polynomial is represented by the binary vector [1 1 1 1 0 1 1], corresponding to the upper row of binary digits in the diagram. These digits indicate connections from the outputs of the registers to the adder. The initial 1 corresponds to the input bit. The octal representation of the binary number 1111011 is 171.

The second generator polynomial is represented by the binary vector [1 0 1 1 0 1 1], corresponding to the lower row of binary digits in the diagram. The octal number corresponding to the binary number 1011011 is 137. We use this function to generate a convolutional code with a constraint length of 7, and encode the source information by the function source coded data=convenc(inforSource, trellis).We use the vitdec (De Bit, trellis, 42, 'trunc', 'hard') function to decode the demodulated data.

For concatenated coding, we can consider the coding, channel, and decoding as a generalized channel. This channel also has errors, so it can be further error-corrected. When two concatenation codes are concatenated to form one concatenated code, the coding in the generalized channel is called an inner code, and the channel coding in which the generalized channel is a channel is called an outer code. Here we think that the convolutional code is the inner code and the RS code is the outer code.

3.3 Constellation mapping and demodulation

The digital baseband signal can be written as a result of weighted accumulation of different basis functions. The coefficient $\{s_{ij}\}$ representing the signal $S_i(t)$ is written as a vector $S_i = \{S_{i1}, S_{i2} \dots S_{iN}\}^T \in \mathbb{R}^N$, which is referred to as a signal constellation point of $S_i(t)$. All signal

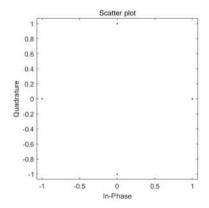


Figure 3 constellation point of QPSK

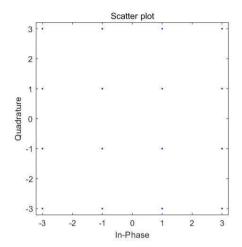
constellation points $\{S_1, S_2 ... S_M\}$ constitute a signal constellation. For base signal given $\{\phi_1(t), \phi_2(t) \dots \phi_N(t)\}\$, the signal $S_i(t)$ has a one-to-one correspondence with its constellation point S_i . QPSK carries information through the phase. Send signal

$$S_i(t) = Ag(t)\cos\left[\frac{2\pi(i-1)}{4}\right]\cos 2\pi f_c t$$

$$-Ag(t)\sin\left[\frac{2\pi(i-1)}{4}\right]\sin 2\pi f_c t$$
The constellation points (S_{i1}, S_{i2}) are given by $S_{i1} = \frac{[2\pi(i-1)]}{4}$

and $S_{i2} = A \sin \left[\frac{2\pi(i-1)}{4} \right] i = 1 \dots 4$.

Constellation diagram is shown in Figure 3. 16QAM carries information in both amplitude and phase.



The transmission signal of 16QAM $S_i(t) =$

Figure 4 constellation point of 16QAM

 $A_i \cos(\theta_i) g(t) \cos(2\pi f_c t) A_i \sin(\theta_i) g(t) \sin(2\pi f_c t) \ 0 \le t \le T$. For a square constellation, S_{i1} and S_{i2} are at (2i-1-4)d, i=1 ... 4. Constellation diagram is shown in Figure 4.

3.4 Serial to parallel transformation

Since the data generated by the source needs to be modulated onto different carriers, and the previous analysis can be used to perform OFDM modulation by the IFFT method, it is necessary to convert the original serially transmitted data into N long parallel data for IFFT

transmit. When the signal is transmitted in the channel, it is transmitted in serial form, so the parallel to serial conversion is performed later.

3.5 Cyclic prefix

Due to the multipath phenomenon in the actual wireless channel, a bundle of signals arrives at the receiving terminal through multiple paths from the transmitting terminal, and the delays of signal propagation are different due to different distances of multiple paths. Signals with different delays are superimposed to cause mutual interference between symbols.

In order to solve the problem of interference between codes caused by multipath, we can increase the symbol time. When the symbol time is much larger than the time delay of the channel, the influence of interference between

codes on the symbol decision will be greatly reduced, but the result is a reduction in the transmission speed of the symbols. In order to effectively reduce the influence of inter-code crosstalk while minimizing the impact on the transmission rate of the symbol itself, we can insert a guard interval between each OFDM symbol, and the guard interval length Ts is generally larger than the maximum of the radio channel. The delay spreads so that the multipath component of one symbol does not interfere with the next symbol.

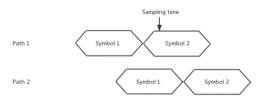


Figure 3 mutual interference between symbols

During this guard interval may be inserted without any signal, that is, an idle transmission period. However, in this case, due to the influence of multipath propagation, adding a guard interval of 0 will cause the waveforms in the integration interval to be discontinuous, destroying the orthogonality between the subcarriers, and inter-channel interference will occur. At this time, the result of the sampling time is not affected by the symbol symbols of other paths, but interference occurs between different subcarriers.

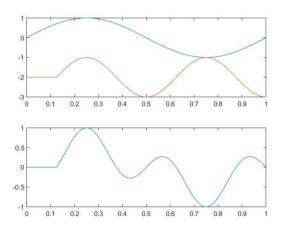


Figure 4 the orthogonality destory between the subcarriers

To resolve inter-subcarrier interference, a cyclic prefix needs to be added within the guard interval. The signal in the cyclic prefix is the same as the portion of the OFDM symbol tail that is the cyclic prefix length, and the length is greater than the maximum delay of the channel. After doing this, for each subcarrier, the waveform at the guard interval becomes continuous, that is, the waveform of each subcarrier is continuous during the whole period

of the guard interval plus the symbol duration. In the interval, the two waveforms are multiplied and then the result of integration is 0, and the orthogonality of the subcarriers is restored. In an actual system, before the OFDM symbol is sent to the channel, the cyclic prefix is first added and then sent to the channel for transmission. At the receiving terminal, the cyclic prefix portion at the beginning of the received symbol is first discarded, and then the remaining portion of the width NT_s is subjected to Fourier transform and then demodulated. Since the length of the cyclic prefix is greater than the maximum delay of the channel, the delay signal with a delay less than the guard interval will not generate ICI during the demodulation process.

3.6 IFFT and FFT

As can be seen from the second section, we can OFDM the baseband symbols by IFFT and demodulate the OFDM signals by FFT.

3.7 Inserting pilot and channel estimation

Due to the randomness of channel noise and the influence of channel multipath, in order to obtain the original data stream, we need to estimate the channel at the receiving terminal to obtain the reference phase and amplitude on each subcarrier of the OFDM symbol, so as to recover raw data bits without error. The accuracy of the channel estimation directly affects the performance of the entire OFDM system. There are two common methods of channel estimation: channel estimation based on pilot information and blind channel estimation based on cyclic prefix. Here we use channel estimation based on pilot information. The insertion pilot method uses the pilot to obtain the channel information of the pilot position, and then obtains the channel information of the entire data transmission by the information of the pilot portion. Several commonly used algorithms for channel estimation of pilot position in OFDM systems are forced zero estimation (ZF), maximum likelihood estimation (ML), least squares estimation (LS), minimum mean square error estimation (MMSE), pilot position's channel information is the ratio of the signal received by the receiving termianl to the signal sent by the transmitting terminal. Here we use least squares estimation (LS).

Assume that the channel model is Y = XH + W, where Y is the received signal and X is the known pilot transmit signal. H is channel information and W is channel noise. The least squares estimation is to estimate the channel information H such that $J = (Y - Y_e)^H (Y - Y_e) = (Y - X_e H_e)^H (Y - X_e H_e)$ is the smallest, where Y_e is the estimated received pilot, H_e is the estimated pilot information, and X_e is the estimated transmit pilot. The derivation gives $H = X^{-1}Y$. It can be seen that the LS estimation only needs to know the transmission signal X, observe the noise W, and other statistical features of the

received signal Y for the parameter H to be determined without other information, so the biggest advantage of the LS channel estimation algorithm is that the structure is simple and the calculation amount Small, the channel characteristics of the pilot position subcarriers can be obtained only by performing a division operation on each

direct path and one indirect paths in the Rayleigh channel, and the delays are $2*10^{\circ}$ (-6) seconds, the path loss is -3db.

4 Result analysis

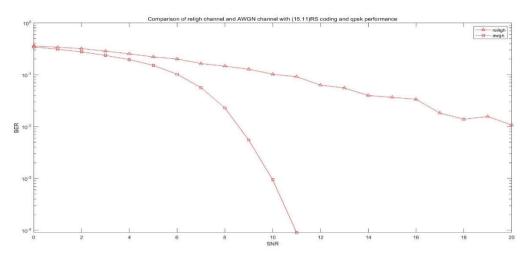


Figure 5 Comparison of AWGN channel and Rayleigh channel performance

carrier. In MATLAB, we use the *commsrc.pn* ('GenPoly', [1 0 0 0 0 1 1], 'NumBitsOut', pilot_num*pilot_len) to generate the pn code. We use pn code as the piloted carrier for the cross-correlation between any two pn code sequences is 0. The Pn sequence is a pseudo-random sequence, and the autocorrelation value of the pn sequence is 1. When the period of the pn sequence is sufficiently large, the cross-correlation of any two pn code sequences is almost 0. Using this property, we join at the transmitting end. The pn code sequence is used as a training sequence.

3.8 Channel

The AWGN channel is a Gaussian additive white noise channel. The Gaussian additive white noise means that the noise of the channel is evenly distributed in the spectrum, and the amplitude is normally distributed. The noise of the AWGN channel is superimposed on the transmitted signal. The Rayleigh fading channel is a statistical model of the radio signal propagation environment. Due to noise, multipath problems and shadowing in the channel, the signal amplitude is random, i.e. "fading", and its envelope obeys the Rayleigh distribution. Here, we use <code>awgn(tx_signal_serial, SNR(a), 'measured')</code> to simulate the AWGN channel, using <code>chan=comm.RayleighChannel('SampleRate',550000, ...</code>

'PathDelays',[0 2e-6],'AveragePathGains',[0 - 3],'MaximumDopplerShift',100); to simulate the Rayleigh channel. It is assumed here that there are two paths, one

4.1 Comparison of AWGN channel and Rayleigh channel performance

The comparison of AWGN channel and Rayleigh channel performance is shown in the Figure 5. We adopt the QPSK modulation method, and the channel coding adopts (15,11)RS coding. We pass the data signal through the AWGN channel and use the biterr (data, old data) function to compare the output data with the original data to calculate the bit error rate. Similarly, we pass the data signal through the Rayleigh channel and then pass the AWGN channel to imitate the real Rayleigh channel, and then calculate the bit error rate in the same way as before. From the Figure we can see that When the signal-to-noise ratio is small, the error rate of the AWGN channel and the Rayleigh channel are not much different, and when the signal-to-noise ratio is relatively large, the performance of the AWGN channel is better than that of the Rayleigh channel. This is because in the Rayleigh channel, there is not only the influence of noise, but also the effects of multipath and shadow. When the signal-to-noise ratio is large enough, the AWGN channel error rate tends to be zero, and the Rayleigh channel's bit error rate fluctuates within a certain range. Therefore, when the signal-to-noise ratio is sufficiently large, the influence of AWGN channel noise on the signal can be ignored. Except for the interference of the signal in the Rayleigh channel, it is mainly related to multipath and has little relationship with noise.

The comparison of RS coding and RS with Convolutional coding performance is shown in Figure 11.

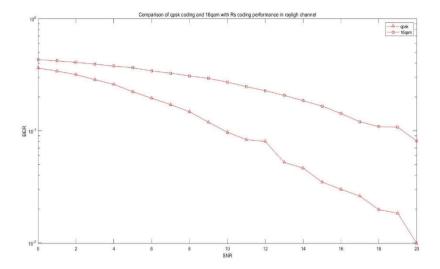


Figure 6 Comparison of QPSK and 16QAM performance

4.2 Comparison of QPSK and 16QAM performance

In the case of Rayleigh channel, the coding mode is rs coding, the performance of QPSK is much larger than that of 16QAM. Because when the signal-to-noise ratio is the same, that is, the power of the signal is the same, the larger the constellation is, the energy allocated to each constellation point is small, and the anti-interference is poor. In practice, the quality of the channel is better than the simulation. The transmission rate of 16QAM is higher than QPSK, so we actually use 16QAM.

4.3 Comparison of RS coding and RS with Convolutional coding performance

When the signal-to-noise ratio is small, the RS coding and convolutional code concatenated coding performance is worse than the RS coding only. This is because the concatenated coding has a threshold effect. When the signal-to-noise ratio is small, the concatenation is due to the principle of information non-increasing. The performance improvement of the code is less than the loss of information caused by system cascading, resulting in the performance of the concatenated code is not as good as a single code. When the signal-to-noise ratio is increased, the concatenated coding of the RS coding and the convolutional code is better than the performance of only the convolutional code coding. The concatenated code is more reliable than the single RS coding, and is more adaptable to the channel with stronger interference.

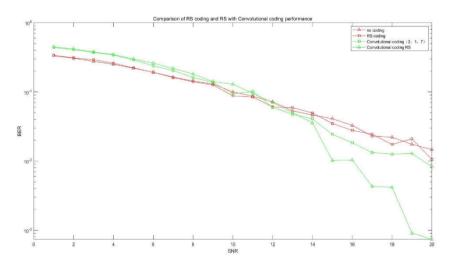


Figure 7 Comparison of different coding performance

The limited number of simulation points and the influence of the multipath of the Rayleigh channel leads to the last part of the waveform fluctuates.

5 Reference

[1] 巩林玉,陈长兴,金贵斌,刘保军,GONG Lin-yu,CHEN Chang-xing,JIN Gui-bin,LIU Bao-jun- 《邮电设计技术》2009 年 4 期

[2]马一森 李 智 赵建华 徐 炜 张 东.OFDM 通信系统的分析及其 Simulink 仿真研究[J].计算机与数字工程,;,2010.11(2):137-139

6 Code

6.1 Code of comparing of AWGN channel and Rayleigh channel performance

function

[error_bit_all,error_symbol_all]=rs_awgn_coding() sonCarrierNum temp=88;

symbols_Per_Carrier=1000;%每子载波含符号数/帧数bits Per Symbol=1;%每符号含比特数

modulate bit=2;%调制阶数(每个符号比特数)

IFFT_bin_length=2^ceil(log2(sonCarrierNum_temp));%FFT 点数

PrefixRatio=1/4;%保护间隔与 OFDM 数据的比例 1/6~1/4

pilot Inter=1;%插入导频间隔

% CP=PrefixRatio*IFFT_bin_length;%每一个 OFDM 符号添加的循环前缀长度为 1/4*IFFT_bin_length CP=25:

SNR=0:1:20; %信噪比 dB

nn=15;

kk=11;

%------信源输入------

inforSource=randi([0,1],1,sonCarrierNum_temp*symbol
s_Per_Carrier*bits_Per_Symbol);

msg4 temp=reshape(inforSource,4,[])';

msg4=bi2de(msg4_temp,'left-msb');%将原来的数据转换为4位16进制

msg4_togf=reshape(msg4,kk,[]).'; %带转换的矩阵,十一输入

msgGF=gf(msg4 togf,4);%转换为伽罗华域

msgrs=rsenc(msgGF,nn,kk); %(15,11) RS 编码 11 个输入 15 个输出

msgrs1=reshape(msgrs.',1,length(msg4)/kk*nn);% 将 rs 编码输出转成一行

msgrs2=de2bi(double(msgrs1.x),'left-msb');%十进制转 二进制

source_coded_data_rs=reshape(msgrs2',1,length(msg4)/k k*nn*4);%待调制信号 输出一行信号 (数据)

```
_____
data temp1=
reshape(source coded data rs,modulate bit,[])';
                                         %以
每组2比特进行分组,输出两列数据
modulate data=pskmod(bi2de(data temp1),2^(modulate
_bit),pi/4);%输出一列数据
modulate data=reshape(modulate data,60,[]);
[modulate wide, modulate length]=size(modulate data);
modulate data temp=[modulate data(1:30,:);zeros(1,mo
dulate length);modulate data(31:60,:)];%在原来输出数
据的中间插 0
h1=commsrc.pn('GenPoly', [1 0 0 0
1], 'NumBitsOut', 61*modulate length, 'InitialConditions', [
0\ 0\ 0\ 0\ 0\ 1]);
pn code temp=generate(h1);
pn code=2*pn code temp-1;
pn code=reshape(pn code,61,[]);
modulate data pn=zeros(61,2*modulate length);
for i=1:modulate length
   modulate data pn(:,(2*i-
1))=modulate data temp(:,i);
    modulate data pn(:,2*i)=pn code(:,i);
modulate data pn(62:64,:)=0;
modulate data pn out=[modulate data pn(31:64,:);mod
ulate data pn(1:30,:);
%-----ifft-----i
time signal_ifft=ifft(modulate_data_pn_out);
%-----cp------
time_signal_cp=[time_signal_ifft(39:64,:);time_signal_if
ft(1:64,:)];%把 ifft 的末尾 CP 个数补充到最前面
[time signal cp wide,time signal cp length]=size(time
signal cp);
for ii=1:modulate length
    time signal out(:,ii)=[time signal cp(:,2*ii-
1); time signal cp(:,2*ii)];
time signal out 1=reshape(time signal out,[],1);
for a=1:21
% chan=comm.RayleighChannel('SampleRate',550000, ...
        'PathDelays',[0 2e-6],'AveragePathGains',[0 -
3], 'MaximumDopplerShift', 100, 'RandomStream', 'mt1993
7ar with seed', 'Seed', 8007);
```

% chan=comm.RayleighChannel('SampleRate',550000, ...

3], 'MaximumDopplerShift', 100);

'PathDelays',[0 2e-6],'AveragePathGains',[0 -

```
% Rayleigh signal=chan(time signal out 1);
                                               d2=de2bi(d1,'left-msb').';
                                               rx decode=reshape(d2,1,[]);
awgn signal=awgn( time signal out 1.a-1.'measured'):%
                                               添加高斯白噪声
% awgn signal=time signal out 1;
[error num,error ratio]=biterr(inforSource,rx decode);
                                               error bit all(a,1)=error ratio;
                                               receive signal serial=awgn signal;
receive signal perallel=reshape(receive signal serial,ti
                                               error symbol data1=
me signal cp wide,[]);
                                               reshape(rx decode,modulate bit,[])';
                                                                              %以每组2比特
%-------去循环前缀-------
                                               进行分组,输出两列数据
                                               error symbol data receive=bi2de(error symbol data1);
  receive data=receive signal perallel(27:90,:);
                                               error symbol data2=
reshape(inforSource,modulate bit,[])';
                                                                               %以每组 2 比
                                               特进行分组,输出两列数据
frequency data no cp=fft(receive data);
                                               error symbol data transmite=bi2de(error symbol data2
frequency data=[frequency data no cp(35:64,:);frequen
                                               );%输出一列数据
cy data no cp(1:31,:);
                                               [error symbol num,error symbol ratio]=symerr(error s
[frequency data wide, frequency data length]=size(freq
                                               ymbol data receive, error symbol data transmite);
uency data);
                                               error symbol all(a,1)=error symbol ratio;
end
                                               end
channel condition=zeros(frequency data wide,modulate
                                               <sup>0</sup>/<sub>0</sub>------
                                               function
estimate data=zeros(frequency_data_wide,modulate_len
                                               [error bit all,error symbol all]=rs reiligh coding()
gth);
                                               sonCarrierNum temp=88;
for iii=1:modulate length
                                               symbols Per Carrier=1000;%每子载波含符号数/帧数
                                               bits Per Symbol=1;%每符号含比特数
channel condition(:,iii)=frequency data(:,2*iii)./pn code
                                               modulate bit=2;%调制阶数(每个符号比特数)
(:,iii);
                                               IFFT bin length=2\ceil(log2(sonCarrierNum temp));\%
    estimate data(:,iii)=frequency data(:,(2*iii-
                                               FFT 点数
1))./channel condition(:,iii);
                                               PrefixRatio=1/4;%保护间隔与 OFDM 数据的比例
real data temp=[estimate data(1:30,:);estimate data(32:
                                               1/6 \sim 1/4
61.:)];
                                               pilot Inter=1;%插入导频间隔
% CP=PrefixRatio*IFFT bin length ;%每一个 OFDM
                                               符号添加的循环前缀长度为 1/4*IFFT bin length
demodulate data temp=reshape(real data temp,1,[]);
                                               CP=25:
demodulate data=pskdemod(demodulate data temp,2^(
                                               SNR=0:1:20; %信噪比 dB
modulate bit),pi/4);
                                               nn=15;
demodulate data bits temp=reshape(demodulate data,[]
                                               kk=11:
                                               %------信源输入--------信源输入-------
demodulate data bits=de2bi(demodulate data bits tem
                                               inforSource=randi([0,1],1,sonCarrierNum temp*symbol
real data temp1 = reshape(demodulate data bits',1,[]);
                                               s Per Carrier*bits Per Symbol);
%------译码-------
                                               %-----信道编码-------信道编码------
[real data temp1 wide,real data length]=size(real data
                                               msg4 temp=reshape(inforSource,4,[])';
                                               msg4=bi2de(msg4 temp,'left-msb');%将原来的数据转
yrsgs4=reshape(real data temp1 ,4,real data temp1 wi
                                               换为4位16进制
de*real data length/4).';
                                               msg4 togf=reshape(msg4,kk,[]).'; %带转换的矩阵,十
yrsgs41=bi2de(yrsgs4,'left-msb');
yrsgs41=reshape(yrsgs41,nn,length(yrsgs41)/nn).';
ygsrsdecode=rsdec(gf(yrsgs41,4),nn,kk);
                                               msgGF=gf(msg4 togf,4);%转换为伽罗华域
d1=reshape(ygsrsdecode.x',1,[]);
```

```
msgrs=rsenc(msgGF,nn,kk); %(15,11) RS 编码 11 个输
入 15 个输出
                                                % chan=comm.RayleighChannel('SampleRate',550000, ...
msgrs1=reshape(msgrs.',1,length(msg4)/kk*nn);% 将 rs
                                                        'PathDelays',[0 2e-6],'AveragePathGains',[0 -
编码输出转成一行
                                                3], 'MaximumDopplerShift', 100, 'RandomStream', 'mt1993
msgrs2=de2bi(double(msgrs1.x),'left-msb');%十进制转
                                                7ar with seed', 'Seed', 8007);
                                                chan=comm.RayleighChannel('SampleRate',550000, ...
source coded data rs=reshape(msgrs2',1,length(msg4)/k
                                                    'PathDelays',[0
                                                                   2e-6], 'Average Path Gains', [0
k*nn*4);%待调制信号 输出一行信号(数据)
                                                3], 'MaximumDopplerShift', 100);
Rayleigh signal=chan(time signal out 1);
                                                awgn signal=awgn( Rayleigh signal,a-1,'measured');%
data temp1=
                                                添加高斯白噪声
reshape(source coded data rs,modulate bit,[])';
                                        %以
                                                % awgn signal=time signal out 1;
每组2比特进行分组,输出两列数据
                                                modulate data=pskmod(bi2de(data temp1),2^(modulate
bit),pi/4);%输出一列数据
                                                   receive signal serial=awgn signal;
receive signal perallel=reshape(receive signal serial,ti
                                                me signal cp wide,[]);
modulate data=reshape(modulate data,60,[]);
                                                %------去循环前缀-------
[modulate wide, modulate length]=size(modulate data);
modulate data temp=[modulate data(1:30,:);zeros(1,mo
                                                   receive_data=receive_signal_perallel(27:90,:);
dulate length);modulate data(31:60,:)];%在原来输出数
                                                %------fft------
据的中间插0
h1=commsrc.pn('GenPoly', [1
                            0 0 0 0
                                                frequency data no cp=fft(receive data);
1], 'NumBitsOut', 61*modulate length, 'InitialConditions', [
                                                frequency data=[frequency data no cp(35:64,:);frequen
000001);
                                                cy data no cp(1:31,:);
pn code temp=generate(h1);
                                                [frequency data wide, frequency data length]=size(freq
pn code=2*pn code temp-1;
                                                uency data);
pn code=reshape(pn code,61,[]);
                                                modulate data pn=zeros(61,2*modulate length);
for i=1:modulate length
   modulate data pn(:,(2*i-
                                                channel condition=zeros(frequency data wide,modulate
1))=modulate data temp(:,i);
                                                estimate data=zeros(frequency data wide,modulate len
   modulate data pn(:,2*i)=pn code(:,i);
end
                                                for iii=1:modulate length
modulate data pn(62:64,:)=0;
modulate data pn out=[modulate data pn(31:64,:);mod
                                                channel condition(:,iii)=frequency data(:,2*iii)./pn code
ulate data pn(1:30,:);
%-----ifft-----i
                                                (:,iii);
                                                    estimate data(:,iii)=frequency data(:,(2*iii-
                                                1))./channel condition(:,iii);
time signal ifft=ifft(modulate data pn out);
                                                end
%-----cp------
                                                real data temp=[estimate data(1:30,:);estimate data(32:
time signal cp=[time signal ifft(39:64,:);time signal if
                                                ft(1:64,:)];%把 ifft 的末尾 CP 个数补充到最前面
[time signal cp wide,time signal cp length]=size(time
                                                demodulate data temp=reshape(real data temp,1,[]);
signal cp);
                                                demodulate data=pskdemod(demodulate data temp,2^(
modulate bit),pi/4);
                                                demodulate data bits temp=reshape(demodulate data,[]
for ii=1:modulate length
                                                ,1);
   time signal out(:,ii)=[time signal cp(:,2*ii-
                                                demodulate data bits=de2bi(demodulate data bits tem
1); time signal cp(:,2*ii)];
                                                real data temp1 = reshape(demodulate data bits',1,[]);
time signal out 1=reshape(time signal out,[],1);
```

for a=1:21

%	inforSource=randi([0,1],1,sonCarrierNum_temp*symbol
	s_Per_Carrier*bits_Per_Symbol);
[real_data_temp1_wide,real_data_length]=size(real_data_	%信道编码
temp1);	
yrsgs4=reshape(real_data_temp1 ,4,real_data_temp1_wi	msg4_temp=reshape(inforSource,4,[])';
de*real data length/4).';	msg4=bi2de(msg4 temp,'left-msb');%将原来的数据转
yrsgs41=bi2de(yrsgs4,'left-msb');	
	换为 4 位 16 进制
yrsgs41=reshape(yrsgs41,nn,length(yrsgs41)/nn).';	msg4_togf=reshape(msg4,kk,[]).'; %带转换的矩阵,十
ygsrsdecode=rsdec(gf(yrsgs41,4),nn,kk);	一输入
d1=reshape(ygsrsdecode.x',1,[]);	•
d2=de2bi(d1,'left-msb').';	msgGF=gf(msg4_togf,4);%转换为伽罗华域
rx_decode=reshape(d2,1,[]);	msgrs=rsenc(msgGF,nn,kk);%(15,11) RS 编码 11 个输
%	入 15 个输出
	msgrs1=reshape(msgrs.',1,length(msg4)/kk*nn);% 将 rs
[error_num,error_ratio]=biterr(inforSource,rx_decode);	编码输出转成一行
error_bit_all(a,1)=error_ratio;	msgrs2=de2bi(double(msgrs1.x),'left-msb');%十进制转
%误符号率	二进制
	source_coded_data_rs=reshape(msgrs2',1,length(msg4)/k
error_symbol_data1=	
	k*nn*4);%待调制信号 输出一行信号(数据)
I \ =	%调制
进行分组,输出两列数据	
error_symbol_data_receive=bi2de(error_symbol_data1);	data temp1=
error symbol data2=	reshape(source coded data rs,modulate bit,[])'; %以
reshape(inforSource,modulate_bit,[])'; %以每组 2 比	
特进行分组,输出两列数据	每组2比特进行分组,输出两列数据
	modulate_data=pskmod(bi2de(data_temp1),2^(modulate
error_symbol_data_transmite=bi2de(error_symbol_data2	_bit),pi/4);%输出一列数据
);%输出一列数据	%插入导频
[error_symbol_num,error_symbol_ratio]=symerr(error_s	
ymbol data receive, error symbol data transmite);	
error symbol all(a,1)=error symbol ratio;	modulate_data=reshape(modulate_data,60,[]);
end	[modulate_wide,modulate_length]=size(modulate_data);
	modulate_data_temp=[modulate_data(1:30,:);zeros(1,mo
end	dulate length);modulate data(31:60,:)];%在原来输出数
	据的中间插 0
6.2 Code of comparing of QPSK and 16QAM	h1=commsrc.pn('GenPoly', [1 0 0 0 0 1
performance	
function	1],'NumBitsOut',61*modulate_length,'InitialConditions',[
[error_bit_all,error_symbol_all]=rs_qpsk_coding()	0 0 0 0 0 1]);
sonCarrierNum_temp=88;	pn_code_temp=generate(h1);
symbols_Per_Carrier=1000;%每子载波含符号数/帧数	pn_code=2*pn_code_temp-1;
	pn_code=reshape(pn_code,61,[]);
bits_Per_Symbol=1;%每符号含比特数	modulate_data_pn=zeros(61,2*modulate_length);
modulate_bit=2;%调制阶数(每个符号比特数)	for i=1:modulate length
IFFT bin length=2^ceil(log2(sonCarrierNum temp));%	modulate data pn(:,(2*i-
FFT 点数	
	1))=modulate_data_temp(:,i);
PrefixRatio=1/4;%保护间隔与 OFDM 数据的比例	modulate_data_pn(:,2*i)=pn_code(:,i);
1/6~1/4	end
pilot_Inter=1;%插入导频间隔	modulate_data_pn(62:64,:)=0;
% CP=PrefixRatio*IFFT bin length ;%每一个 OFDM	modulate_data_pn_out=[modulate_data_pn(31:64,:);mod
符号添加的循环前缀长度为 1/4*IFFT_bin_length	ulate data $pn(1:30,:)$];
	%ifft
CP=25;	
SNR=0:1:20; %信噪比 dB	time_signal_ifft=ifft(modulate_data_pn_out);
nn=15;	%cp
kk=11;	
%信源输入	
/0 ロ が、相り/ C	time_signal_cp=[time_signal_ifft(39:64,:);time_signal_if
	ft(1:64,:)];%把 ifft 的末尾 CP 个数补充到最前面

```
[time_signal_cp_wide,time_signal_cp_length]=size(time
                                               signal cp):
demodulate data temp=reshape(real data temp,1,[]);
                                                demodulate data=pskdemod(demodulate data temp,2^(
for ii=1:modulate length
                                                modulate bit),pi/4);
   time signal out(:,ii)=[time signal cp(:,2*ii-
                                                demodulate data bits temp=reshape(demodulate data,[]
1); time signal cp(:,2*ii)];
                                                demodulate data bits=de2bi(demodulate data bits tem
time signal out 1=reshape(time signal out,[],1);
                                                real data temp1 = reshape(demodulate data bits',1,[]);
for a=1:21
[real data temp1 wide,real data length]=size(real data
% chan=comm.RayleighChannel('SampleRate',550000, ...
        'PathDelays',[0 2e-6],'AveragePathGains',[0 -
                                                temp1):
3], 'MaximumDopplerShift', 100, 'RandomStream', 'mt1993
                                                yrsgs4=reshape(real data temp1 ,4,real data temp1 wi
7ar with seed', 'Seed', 8007);
                                                de*real data length/4).';
chan=comm.RayleighChannel('SampleRate',550000, ...
                                                yrsgs41=bi2de(yrsgs4,'left-msb');
    'PathDelays',[0
                  2e-6],'AveragePathGains',[0 -
                                                yrsgs41=reshape(yrsgs41,nn,length(yrsgs41)/nn).';
3], 'MaximumDopplerShift', 100);
                                                ygsrsdecode=rsdec(gf(yrsgs41,4),nn,kk);
Rayleigh signal=chan(time signal out 1);
                                                d1=reshape(ygsrsdecode.x',1,[]);
awgn signal=awgn( Rayleigh signal,a-1,'measured');%
                                                d2=de2bi(d1,'left-msb').';
                                                rx decode=reshape(d2,1,[]);
添加高斯白噪声
                                                awgn signal=time signal out 1;
[error num,error ratio]=biterr(inforSource,rx decode);
                                                error bit all(a,1)=error ratio;
  receive signal serial=awgn_signal;
                                                receive signal perallel=reshape(receive signal serial,ti
me signal cp wide,[]);
                                                error symbol data1=
%以每组2比特
                                                reshape(rx decode,modulate bit,[])';
                                                进行分组,输出两列数据
  receive data=receive signal perallel(27:90,:);
                                                error symbol data receive=bi2de(error symbol data1);
%------fft------
                                                error symbol data2=
                                                reshape(inforSource,modulate bit,[])';
                                                                               %以每组2比
frequency data no cp=fft(receive data);
                                                特进行分组,输出两列数据
frequency data=[frequency data no cp(35:64,:);frequen
                                                error symbol data transmite=bi2de(error symbol data2
cy data no cp(1:31,:);
                                                );%输出一列数据
[frequency data wide, frequency data length]=size(freq
                                                [error symbol num,error symbol ratio]=symerr(error s
uency_data);
                                                ymbol data receive, error symbol data transmite);
error symbol all(a,1)=error symbol ratio;
                                                end
channel condition=zeros(frequency data wide,modulate
                                                end
                                                %----
                                                    ______
estimate data=zeros(frequency data wide,modulate len
                                                function
                                                [error bit all,error symbol all]=rs 16qam coding()
for iii=1:modulate length
                                                sonCarrierNum temp=176;
                                                symbols Per Carrier=1000;%每子载波含符号数/帧数
channel condition(:,iii)=frequency data(:,2*iii)./pn code
                                                bits Per Symbol=1;%每符号含比特数
                                                modulate bit=4;%调制阶数(每个符号比特数)
    estimate data(:,iii)=frequency data(:,(2*iii-
                                                IFFT bin length=2\ceil(log2(sonCarrierNum temp));\%
1))./channel condition(:,iii);
                                                FFT 点数
end
                                                PrefixRatio=1/4;%保护间隔与 OFDM 数据的比例
real data temp=[estimate data(1:30,:);estimate data(32:
61,:)];
                                                1/6 \sim 1/4
                                                pilot Inter=1;%插入导频间隔
```

```
modulate_data_pn_out=[modulate data pn(31:64,:);mod
% CP=PrefixRatio*IFFT bin length :%每一个 OFDM
                                              ulate data pn(1:30,:);
符号添加的循环前缀长度为 1/4*IFFT bin length
                                              _____ifft-----ifft-----
CP=25:
SNR=0:1:20; %信噪比 dB
                                              time signal ifft=ifft(modulate data pn out);
nn=15;
                                              %-----cp------
kk=11;
%------信源输入--------信源输入-------
                                              time signal cp=[time signal ifft(39:64,:);time signal if
                                              ft(1:64.:)]:%把 ifft 的末尾 CP 个数补充到最前面
inforSource=randi([0,1],1,sonCarrierNum temp*symbol
                                              [time signal cp wide,time signal cp length]=size(time
s Per Carrier*bits Per Symbol);
                                              signal cp);
%-----信道编码------信道编码------
                                              msg4 temp=reshape(inforSource,4,[])';
                                              for ii=1:modulate length
msg4=bi2de(msg4 temp,'left-msb');%将原来的数据转
                                                  time signal out(:,ii)=[time signal cp(:,2*ii-
换为4位16进制
                                              1); time signal cp(:,2*ii)];
msg4 togf=reshape(msg4,kk,[]).'; %带转换的矩阵, 十
                                              end
一输入
                                              time signal out 1=reshape(time signal out,[],1);
msgGF=gf(msg4 togf,4);%转换为伽罗华域
                                              for a=1:21
                                              msgrs=rsenc(msgGF,nn,kk); %(15,11) RS 编码 11 个输
入 15 个输出
                                              % chan=comm.RayleighChannel('SampleRate',550000, ...
msgrs1=reshape(msgrs.',1,length(msg4)/kk*nn);% 将 rs
                                                      'PathDelays',[0 2e-6],'AveragePathGains',[0 -
编码输出转成一行
                                              3],'MaximumDopplerShift',100,'RandomStream','mt1993
msgrs2=de2bi(double(msgrs1.x),'left-msb');%十进制转
                                              7ar with seed', 'Seed', 8007);
二讲制
                                              chan=comm.RayleighChannel('SampleRate', 550000, ...
source coded data rs=reshape(msgrs2',1,length(msg4)/k
                                                  'PathDelays',[0
                                                                2e-6], 'Average Path Gains', [0
k*nn*4);%待调制信号 输出一行信号(数据)
                                              3], 'MaximumDopplerShift', 100);
Rayleigh signal=chan(time signal out 1);
_____
                                              awgn signal=awgn( Rayleigh signal,a-1,'measured');%
data temp1=
                                              添加高斯白噪声
                                       %以
reshape(source_coded_data_rs,modulate_bit,[])';
                                                  awgn signal=time signal out 1;
每组2比特进行分组,输出两列数据
                                              modulate data=qammod(bi2de(data temp1),2^(modulat
e bit));%输出一列数据
                                                 receive signal_serial=awgn_signal;
receive signal perallel=reshape(receive signal serial,ti
_____
                                              me_signal_cp_wide,[]);
modulate data=reshape(modulate data,60,[]);
                                              %------去循环前缀------
[modulate wide,modulate length]=size(modulate data);
modulate data temp=[modulate data(1:30,:);zeros(1,mo
                                                 receive data=receive signal perallel(27:90,:);
dulate length);modulate data(31:60,:)];%在原来输出数
                                              据的中间插0
h1=commsrc.pn('GenPoly', [1
                           0 0
                                              frequency data no cp=fft(receive data);
1],'NumBitsOut',61*modulate length,'InitialConditions',[
                                              frequency data=[frequency data no cp(35:64,:);frequen
0000011);
                                              cy data no cp(1:31,:);
pn code temp=generate(h1);
                                              [frequency data wide, frequency data length]=size(freq
pn code=2*pn code temp-1;
                                              uency data);
pn code=reshape(pn code,61,[]);
                                              modulate data pn=zeros(61,2*modulate length);
                                              -----
for i=1:modulate length
                                              channel condition=zeros(frequency data wide,modulate
   modulate data pn(:,(2*i-
1))=modulate data temp(:,i);
                                              estimate data=zeros(frequency data wide,modulate len
   modulate data pn(:,2*i)=pn code(:,i);
                                              for iii=1:modulate length
modulate data pn(62:64,:)=0;
```

channel condition(:,iii)=frequency data(:,2*iii)./pn code bits Per Symbol=1;%每符号含比特数 (:,iii); modulate bit=2;%调制阶数(每个符号比特数) estimate data(:,iii)=frequency data(:,(2*iii-IFFT bin length=2\ceil(log2(sonCarrierNum temp));\% 1))./channel condition(:,iii); FFT 点数 end PrefixRatio=1/4;%保护间隔与 OFDM 数据的比例 real data temp=[estimate data(1:30,:);estimate data(32: 1/6~1/4 61,:)]; pilot Inter=1;%插入导频间隔 % CP=PrefixRatio*IFFT bin length;%每一个 OFDM -----符号添加的循环前缀长度为 1/4*IFFT bin length demodulate data temp=reshape(real data temp,1,[]); CP=25; demodulate data=qamdemod(demodulate data temp,2\(^{\)}(SNR=0:1:20: %信噪比 dB modulate bit)); demodulate data bits temp=reshape(demodulate data,[] nn=15: kk=11; %------信源输入------demodulate data bits=de2bi(demodulate data bits tem p); real data temp1 = reshape(demodulate data bits',1,[]); inforSource=randi([0,1],1,sonCarrierNum temp*symbol s Per Carrier*bits Per Symbol); [real data temp1 wide,real data length]=size(real data ----temp1); msg4 temp=reshape(inforSource,4,[])'; yrsgs4=reshape(real data temp1 ,4,real data temp1 wi msg4=bi2de(msg4 temp,'left-msb');%将原来的数据转 de*real data length/4).'; 换为4位16进制 yrsgs41=bi2de(yrsgs4,'left-msb'); msg4 togf=reshape(msg4,kk,[]).'; %带转换的矩阵, 十 yrsgs41=reshape(yrsgs41,nn,length(yrsgs41)/nn).'; 一输入 ygsrsdecode=rsdec(gf(yrsgs41,4),nn,kk); msgGF=gf(msg4 togf,4);%转换为伽罗华域 d1=reshape(ygsrsdecode.x',1,[]); msgrs=rsenc(msgGF,nn,kk); %(15,11) RS 编码 11 个输 d2=de2bi(d1,'left-msb').'; 入 15 个输出 rx decode=reshape(d2,1,[]); msgrs1=reshape(msgrs.',1,length(msg4)/kk*nn);% 将 rs 编码输出转成一行 [error num,error ratio]=biterr(inforSource,rx decode); msgrs2=de2bi(double(msgrs1.x),'left-msb');%十进制转 error bit all(a,1)=error ratio; 二进制 source_coded_data_rs=reshape(msgrs2',1,length(msg4)/k k*nn*4);%待调制信号 输出一行信号(数据) error symbol data1= st2 = 4831: reshape(rx decode,modulate bit,[])'; %以每组2比特 inter rs code = randintrly(source coded data rs,st2); % 进行分组,输出两列数据 Interleave. error symbol data receive=bi2de(error symbol data1); error symbol data2= reshape(inforSource,modulate bit,[])'; %以每组 2 比 data temp1= reshape(inter rs code,modulate bit,[])'; 特进行分组,输出两列数据 以每组2比特进行分组,输出两列数据 error symbol data transmite=bi2de(error symbol data2 modulate data=pskmod(bi2de(data temp1),2^(modulate);%输出一列数据 bit),pi/4);%输出一列数据 [error symbol num,error symbol ratio]=symerr(error s ymbol data receive, error symbol data transmite); error symbol all(a,1)=error symbol ratio; modulate data=reshape(modulate data,60,[]); end [modulate wide, modulate length]=size(modulate data); end modulate data temp=[modulate data(1:30,:);zeros(1,mo dulate length);modulate data(31:60,:)];%在原来输出数 6.3 Code of comparing of RS coding and RS with 据的中间插0 Convolutional coding performance

function [error bit all,error symbol all]=rs coding()

sonCarrierNum temp=88;

symbols Per Carrier=1000;%每子载波含符号数/帧数

```
0 0 0 0 1
                                                 %------fft------
h1=commsrc.pn('GenPoly', [1
1], 'NumBitsOut', 61*modulate length, 'InitialConditions', [
000001);
                                                 frequency data no cp=fft(receive data);
pn code temp=generate(h1);
                                                 frequency data=[frequency data no cp(35:64,:);frequen
pn code=2*pn code temp-1;
                                                 cy data no cp(1:31,:)];
pn code=reshape(pn code,61,[]);
                                                 [frequency data wide, frequency data length]=size(freq
modulate data pn=zeros(61,2*modulate length);
                                                 uency_data);
for i=1:modulate length
                                                 %-----信道估计------
   modulate data pn(:,(2*i-
1))=modulate data temp(:,i);
                                                 channel condition=zeros(frequency data wide,modulate
   modulate data pn(:,2*i)=pn code(:,i);
end
                                                 estimate data=zeros(frequency data wide,modulate len
modulate data pn(62:64,:)=0;
modulate data pn out=[modulate data pn(31:64,:);mod
                                                 for iii=1:modulate length
ulate data pn(1:30,:);
%-----ifft-----
                                                 channel condition(:,iii)=frequency data(:,2*iii)./pn code
                                                 (:,iii);
time_signal_ifft=ifft(modulate_data_pn_out);
                                                     estimate data(:,iii)=frequency data(:,(2*iii-
%-----cp------
                                                 1))./channel condition(:,iii);
                                                 end
time signal cp=[time signal ifft(39:64,:);time signal if
                                                 real data temp=[estimate data(1:30,:);estimate data(32:
ft(1:64,:)];%把 ifft 的末尾 CP 个数补充到最前面
                                                 [time signal cp wide,time signal cp length]=size(time
signal cp);
demodulate data temp=reshape(real data temp,1,[]);
-----
                                                 demodulate data=pskdemod(demodulate data temp,2^(
                                                 modulate bit),pi/4);
for ii=1:modulate length
    time signal out(:,ii)=[time signal cp(:,2*ii-
                                                 demodulate data bits temp=reshape(demodulate data,[]
1);time signal cp(:,2*ii)];
                                                 ,1);
                                                 demodulate data bits=de2bi(demodulate data bits tem
end
time signal out 1=reshape(time signal out,[],1);
                                                 real data temp1 = reshape(demodulate data bits',1,[]);
for a=1:20
% chan=comm.RayleighChannel('SampleRate',550000, ...
                                                 deinter con = randdeintrly(real data temp1,st2); %
        'PathDelays',[0 2e-6],'AveragePathGains',[0 -
                                                 Deinterleave.
3], 'MaximumDopplerShift', 100, 'RandomStream', 'mt1993
                                                 [real data temp1 wide,real data length]=size(deinter c
7ar with seed', 'Seed', 8007);
chan=comm.RayleighChannel('SampleRate',550000, ...
                                                 yrsgs4=reshape(deinter con ,4,real data temp1 wide*re
    'PathDelays',[0
                   2e-6], 'Average Path Gains', [0
                                                 al data length/4).';
3], 'MaximumDopplerShift', 100);
                                                 yrsgs41=bi2de(yrsgs4,'left-msb');
                                                 yrsgs41=reshape(yrsgs41,nn,length(yrsgs41)/nn).';
Rayleigh signal=chan(time signal out 1);
                                                 ygsrsdecode=rsdec(gf(yrsgs41,4),nn,kk);
awgn signal=awgn( Rayleigh signal,a,'measured');% 添
                                                 d1=reshape(ygsrsdecode.x',1,[]);
加高斯白噪声
                                                 d2=de2bi(d1,'left-msb').';
   awgn_signal=time_signal_out_1;
                                                 rx decode=reshape(d2,1,[]);
receive signal serial=awgn signal;
                                                 [error num,error ratio]=biterr(inforSource,rx decode);
                                                 error bit all(a,1)=error ratio;
receive signal perallel=reshape(receive signal serial,ti
                                                 me signal cp wide,[]);
%------去循环前缀-------
                                                 error symbol data1=
                                                 reshape(rx decode,modulate bit,[])'; %以每组2比特
  receive data=receive signal perallel(27:90,:);
                                                 进行分组,输出两列数据
```

```
error symbol data receive=bi2de(error symbol data1);
                                               trellis = poly2trellis(7,[133 171]);
                                                                            %(2,1,7)卷积编码
error symbol data2=
                                               输入一位,输出两位,约束长度为7,1011011-
reshape(inforSource,modulate bit,[])';
                               %以每组 2 比
                                               ->133//1111001-->171
特进行分组,输出两列数据
                                               source coded data con=convenc(inter rs code,trellis);
error symbol data transmite=bi2de(error symbol data2
                                               ):%输出一列数据
[error symbol num,error symbol ratio]=symerr(error s
                                               data temp1=
ymbol data receive, error symbol data transmite);
                                               reshape(source coded data con, modulate bit,[])';
                                                                                         %
error symbol all(a,1)=error symbol ratio;
                                               以每组2比特进行分组,输出两列数据
end
                                               modulate data=pskmod(bi2de(data temp1),2^(modulate
end
                                                bit),pi/4);%输出一列数据
%--
                                               function
[error bit all,error symbol all]=con rs coding()
                                               modulate data=reshape(modulate_data,60,[]);
sonCarrierNum temp=44;
                                               [modulate wide,modulate length]=size(modulate data);
symbols Per Carrier=1000;%每子载波含符号数/帧数
                                               modulate data temp=[modulate data(1:30,:);zeros(1,mo
bits Per Symbol=1;%每符号含比特数
                                               dulate length);modulate data(31:60,:)];%在原来输出数
modulate bit=2:%调制阶数(每个符号比特数)
                                               据的中间插0
IFFT bin length=2\ceil(log2(sonCarrierNum temp));\%
                                               h1=commsrc.pn('GenPoly', [1 0 0 0 0 1
FFT 点数
                                               1], 'NumBitsOut', 61*modulate length, 'InitialConditions', [
PrefixRatio=1/4;%保护间隔与 OFDM 数据的比例
                                               000001);
1/6~1/4
                                               pn code temp=generate(h1);
pilot Inter=1;%插入导频间隔
                                               pn code=2*pn code temp-1;
                                               pn code=reshape(pn code,61,[]);
% CP=PrefixRatio*IFFT bin length ;%每一个 OFDM
                                               modulate data pn=zeros(61,2*modulate length);
符号添加的循环前缀长度为 1/4*IFFT bin length
                                               for i=1:modulate length
CP = 25:
                                                   modulate data pn(:.(2*i-
SNR=0:1:20; %信噪比 dB
                                               1))=modulate data temp(:,i);
nn=15;
                                                   modulate_data_pn(:,2*i)=pn_code(:,i);
kk=11;
%------信源输入-------信源输入-------
                                               modulate data pn(62:64,:)=0;
_____
                                               modulate data pn out=[modulate data pn(31:64,:);mod
inforSource=randi([0,1],1,sonCarrierNum temp*symbol
                                               ulate data pn(1:30,:);
s Per Carrier*bits Per Symbol);
                                               %-----ifft------
%-----信道编码------信道编码-----
                                               time signal ifft=ifft(modulate data pn out);
msg4_temp=reshape(inforSource,4,[])';
                                               %-----cp------
msg4=bi2de(msg4 temp,'left-msb');%将原来的数据转
换为 4 位 16 进制
                                               time signal cp=[time signal ifft(39:64,:);time signal if
msg4 togf=reshape(msg4,kk,[]).'; %带转换的矩阵, 十
                                               ft(1:64,:)];%把 ifft 的末尾 CP 个数补充到最前面
                                               [time signal cp wide,time signal cp length]=size(time
                                               signal_cp);
msgGF=gf(msg4 togf,4);%转换为伽罗华域
                                               msgrs=rsenc(msgGF,nn,kk); %(15,11) RS 编码 11 个输
                                               -----
入 15 个输出
                                               for ii=1:modulate length
msgrs1=reshape(msgrs.',1,length(msg4)/kk*nn);% 将 rs
                                                   time signal out(:,ii)=[time signal cp(:,2*ii-
编码输出转成一行
                                               1);time signal cp(:,2*ii)];
msgrs2=de2bi(double(msgrs1.x),'left-msb');%十进制转
                                               end
                                               time signal out 1=reshape(time signal out,[],1);
source coded data rs=reshape(msgrs2',1,length(msg4)/k
                                               for a=1:20
k*nn*4);%待调制信号 输出一行信号(数据)
                                               st2 = 4831:
inter rs code = randintrlv(source coded data rs,st2); %
                                               % chan=comm.RayleighChannel('SampleRate',550000, ...
```

Interleave.

```
%
        'PathDelays',[0 2e-6],'AveragePathGains',[0 -
                                                trellis = poly2trellis(7,[133 171]);
                                                                             %(2,1,7)卷积编码
31, 'Maximum Doppler Shift', 100, 'Random Stream', 'mt 1993
                                                输入一位,输出两位,约束长度为7,1011011-
7ar with seed', 'Seed', 8007);
                                                ->133//1111001-->171
chan=comm.RayleighChannel('SampleRate',550000, ...
                                                decode con=vitdec(real data temp1,trellis,34,'trunc','har
                  2e-6],'AveragePathGains',[0 -
    'PathDelays',[0
                                                     %硬判决
3], 'MaximumDopplerShift', 100);
                                                deinter con
                                                                 randdeintrlv(decode con,st2);
                                                                                           %
Rayleigh signal=chan(time signal out 1);
                                                Deinterleave.
awgn signal=awgn( Rayleigh signal,a,'measured');% 添
                                                [real data temp1 wide,real data length]=size(deinter c
加高斯白噪声
     awgn signal=time signal out 1;
                                                yrsgs4=reshape(deinter con ,4,real data temp1 wide*re
al data length/4).':
                                                vrsgs41=bi2de(vrsgs4,'left-msb'):
  receive signal serial=awgn signal;
                                                yrsgs41=reshape(yrsgs41,nn,length(yrsgs41)/nn).';
                                                ygsrsdecode=rsdec(gf(yrsgs41,4),nn,kk);
                                                d1=reshape(ygsrsdecode.x',1,[]);
receive signal perallel=reshape(receive signal serial,ti
                                                d2=de2bi(d1,'left-msb').';
me signal cp wide,[]);
%------去循环前缀-------
                                                rx decode=reshape(d2,1,[]);
                                                _____
  receive data=receive signal perallel(27:90,:);
%------fft------
                                                [error num,error ratio]=biterr(inforSource,rx decode);
                                                error bit all(a,1)=error ratio;
                                                frequency data no cp=fft(receive data);
frequency data=[frequency data no cp(35:64,:);frequen
cy data no cp(1:31,:);
                                                error symbol data1=
[frequency data wide, frequency data length]=size(freq
                                                reshape(rx decode, modulate bit, [])'; %以每组2比特
uency data);
                                                进行分组,输出两列数据
error symbol data receive=bi2de(error symbol data1);
_____
                                                error symbol data2=
channel condition=zeros(frequency data wide,modulate
                                                reshape(inforSource,modulate_bit,[])';
                                                                                %以每组 2 比
length);
                                                特进行分组,输出两列数据
estimate data=zeros(frequency data wide,modulate len
                                                error symbol data transmite=bi2de(error symbol data2
                                                );%输出一列数据
for iii=1:modulate length
                                                [error symbol num,error symbol ratio]=symerr(error s
                                                ymbol data receive, error symbol data transmite);
channel condition(:,iii)=frequency data(:,2*iii)./pn code
                                                error symbol all(a,1)=error symbol ratio;
                                                end
    estimate data(:,iii)=frequency data(:,(2*iii-
                                                <sup>0</sup>/<sub>0</sub>-----
1))./channel condition(:,iii);
                                                function
end
                                                [error bit all,error symbol all]=con rs coding()
real data temp=[estimate data(1:30,:);estimate data(32:
                                                %31 个子频
61,:)];
                                                sonCarrierNum temp=60;
symbols Per Carrier=100;%每子载波含符号数/帧数
                                                bits Per Symbol=1;%每符号含比特数
demodulate data temp=reshape(real data temp,1,[]);
                                                modulate bit=2;%调制阶数(每个符号比特数)
demodulate data=pskdemod(demodulate data temp,2^(
                                                IFFT bin_length=2^ceil(log2(sonCarrierNum_temp));%
modulate bit),pi/4);
demodulate data bits temp=reshape(demodulate data,[]
                                                FFT 点数
                                                PrefixRatio=1/4;%保护间隔与 OFDM 数据的比例
demodulate data bits=de2bi(demodulate data bits tem
                                                1/6~1/4
                                                pilot Inter=1;%插入导频间隔
real data temp1 = reshape(demodulate data bits',1,[]);
                                                % CP=PrefixRatio*IFFT bin length;%每一个 OFDM
符号添加的循环前缀长度为 1/4*IFFT bin length
_____
                                                CP=25:
                                                SNR=0:1:20; %信噪比 dB
                                                nn=15;
```

```
kk=11:
                                               time signal out 1=reshape(time signal out,[],1);
inforSource=randi([0,1],1,sonCarrierNum temp*symbol
                                               for a=1:20
s_Per_Carrier*bits_Per Symbol);
                                               % chan=comm.RayleighChannel('SampleRate',550000, ...
'PathDelays',[0 2e-6],'AveragePathGains',[0 -
                                               3], 'MaximumDopplerShift', 100, 'RandomStream', 'mt1993
                                               7ar with seed', 'Seed', 8007):
trellis = poly2trellis(7,[133 171]); %(2,1,7)卷积编码
                                               chan=comm.RayleighChannel('SampleRate',550000, ...
输入一位,输出两位,约束长度为7,1011011-
                                                   'PathDelays',[0
                                                                 2e-6], 'Average Path Gains', [0
->133//1111001-->171
                                               3],'MaximumDopplerShift',100);
source coded data con=convenc(inforSource,trellis);
                                               Rayleigh signal=chan(time signal out 1);
awgn signal=awgn( Rayleigh signal,a,'measured');% 添
-----
                                               加高斯白噪声
data temp1=
                                                    awgn signal=time signal out 1;
reshape(source coded data con, modulate bit,[])';
                                          %
                                               以每组2比特进行分组,输出两列数据
modulate data=pskmod(bi2de(data temp1),2^(modulate
                                                  receive signal serial=awgn signal;
bit),pi/4);%输出一列数据
receive signal perallel=reshape(receive signal serial,ti
                                               me signal cp wide,[]);
modulate data=reshape(modulate data,60,[]);
                                               %------去循环前缀-------
[modulate wide,modulate length]=size(modulate data);
modulate data temp=[modulate data(1:30,:);zeros(1,mo
                                                  receive data=receive signal perallel(27:90,:);
dulate length);modulate data(31:60,:)];%在原来输出数
                                               %------fft------
据的中间插0
h1=commsrc.pn('GenPoly', [1 0 0 0 0
                                               frequency data no cp=fft(receive data);
1], 'NumBitsOut', 61*modulate length, 'InitialConditions', [
                                               frequency data=[frequency data no cp(35:64,:);frequen
000001);
                                               cv data no cp(1:31.:)]:
pn code temp=generate(h1);
                                               [frequency data wide, frequency data length]=size(freq
pn code=2*pn code temp-1;
                                               uency data);
pn code=reshape(pn code,61,[]);
                                               modulate data pn=zeros(61,2*modulate length);
for i=1:modulate length
                                               channel condition=zeros(frequency data wide,modulate
   modulate data pn(:,(2*i-
1))=modulate data temp(:,i);
                                               estimate data=zeros(frequency data wide,modulate len
   modulate data pn(:,2*i)=pn code(:,i);
                                               for iii=1:modulate length
modulate data pn(62:64,:)=0;
modulate data pn out=[modulate data pn(31:64,:);mod
                                               channel condition(:,iii)=frequency data(:,2*iii)./pn code
ulate data_pn(1:30,:)];
%-----ifft-----ifft
                                                   estimate data(:,iii)=frequency data(:,(2*iii-
                                               1))./channel condition(:,iii);
time_signal_ifft=ifft(modulate_data_pn_out);
%-----cp-----
                                               real data temp=[estimate data(1:30,:);estimate data(32:
                                               61,:)];
time signal cp=[time signal ifft(39:64,:);time signal if
                                               ft(1:64,:)];%把 ifft 的末尾 CP 个数补充到最前面
[time signal cp wide,time signal cp length]=size(time
                                               demodulate data temp=reshape(real data temp,1,[]);
signal cp);
_signal_cp);
%------并串变换-------
                                               demodulate data=pskdemod(demodulate data temp,2^(
                                               modulate bit),pi/4);
                                               demodulate data bits temp=reshape(demodulate data,[]
for ii=1:modulate length
   time signal out(:,ii)=[time signal cp(:,2*ii-
                                               demodulate data bits=de2bi(demodulate data bits tem
1); time signal cp(:,2*ii)];
                                               p);
end
```

```
real data temp1 = reshape(demodulate data bits',1,[]);
trellis = poly2trellis(7,[133 171]); %(2,1,7)卷积编码
输入一位,输出两位,约束长度为7,1011011-
->133//1111001-->171
rx decode=vitdec(real data temp1,trellis,35,'trunc','hard'
); %硬判决
-----
[error num,error ratio]=biterr(inforSource,rx decode);
error bit all(a,1)=error ratio;
_____
error symbol data1=
reshape(rx_decode,modulate_bit,[])'; %以每组2比特
进行分组,输出两列数据
error symbol data receive=bi2de(error symbol data1);
error symbol data2=
reshape(inforSource,modulate bit,[])'; %以每组 2 比
特进行分组,输出两列数据
error symbol data transmite=bi2de(error symbol data2
);%输出一列数据
[error symbol num,error symbol ratio]=symerr(error s
ymbol data receive, error symbol data transmite);
error symbol all(a,1)=error symbol ratio;
end
end
```