Санкт-Петербургский Политехнический Университет Петра Великого
Институт компьютерных наук и технологий
Кафедра компьютерных систем и программных технологий

Телекоммуникационные технологии

Отчет по лабораторным работам №6,7
Цифровая модуляция и помехоустойяивое кодирование

Работу выполнил:
Маринченко В.А.
Группа: 33501/4
Преподаватель:
Богач Н.В.

Санкт-Петербург
2018

Содержание

1 Название работы

Раздел «Цифровая модуляция», лабораторная работа №6 «Цифровая модуляция», раздел «Моделирование телекоммуникационных сигналов» лабораторная работа №7 «Помехоустойчивое кодирование».

2 Цели работы

- Изучение методов модуляции цифровых сигналов.
- Изучение методов помехоустойчивого кодирования и сравнение их свойств

3 Постановка задачи

1. Получить сигналы BPSK, PSK, OQPSK, genQAM, MSK модуляторов
2. Построить их сигнальные созвездия
3. Провести сравнение изученных методов модуляции цифровых сигналов
4. Провести кодирование/декодирование сигнала, полученного с помощью функции randerr кодом Хэмминга 2-мя способами: с помощью встроенных функций encode/decode, а также через создание проверочной и генераторной матриц и вычисление синдрома. Оценить корректирующую способность кода.
5. Выполнить кодирование/декодирование циклическим кодом, кодом БЧХ, кодом Рида-Соломона. Оценить корректирующую способность кода.

4 Цифровая модуляция

Числа при передаче информации в цифровой форме с периодом T поступают от источника информации и называются символами (symbol), а частота передачи символов – символьной скоростью (symbol rate). В практике передачи данных распространена двоичная (binary) последовательность символов, где числа передаются значениями 0 и 1. Цифровая модуляция и демодуляция включают в себя две стадии. При модуляции цифровое сообщение сначала преобразуется в аналоговый модулирующий сигнал, а затем осуществляется аналоговая модуляция. При демодуляции сначала получается аналоговый демодулированный сигнал, а затем он преобразуется в цифровое сообщение. Аналоговый несущий сигнал модулируется цифровым битовым потоком.

Существуют три фундаментальных типа цифровой модуляции (или шифтинга) и один гибридный:

- ASK – Amplitude shift keying (Амплитудная двоичная модуляция)
- FSK – Frequency shift keying (Частотная двоичная модуляция)
- PSK – Phase shift keying (Фазовая двоичная модуляция)
- ASK/PSK. Одна из частных реализаций схемы ASK/PSK, которая называется QAM - Quadrature Amplitude Modulation (квадратурная амплитудная модуляция (КАМ)

При КАМ изменяется как фаза, так и амплитуда несущего сигнала. Это позволяет увеличить число кодируемых в единицу времени бит и при этом повысить помехоустойчивость их передачи по каналу связи. В настоящее время число кодируемых информационных бит на одном интервале может достигать 8-9, а число состояний в сигнальном пространстве, соответственно, от 256 до 512. Фазовый шифтинг представляет «0» как сигнал без сдвига, а «1» как сигнал со сдвигом.

BPSK : используется единственный сдвиг фазы между «0» и «1» — 180 градусов, половина периода.

QPSK: используется 4 различных сдвига фазы (по четверти периода) и может кодировать 2 бита в символе.

4.1 Matlab

Закодируем различные сообщения для модуляции с помощью BPSK, PSK, MSK, OQPSK, genQAM, получим их сигнальные созвездия:

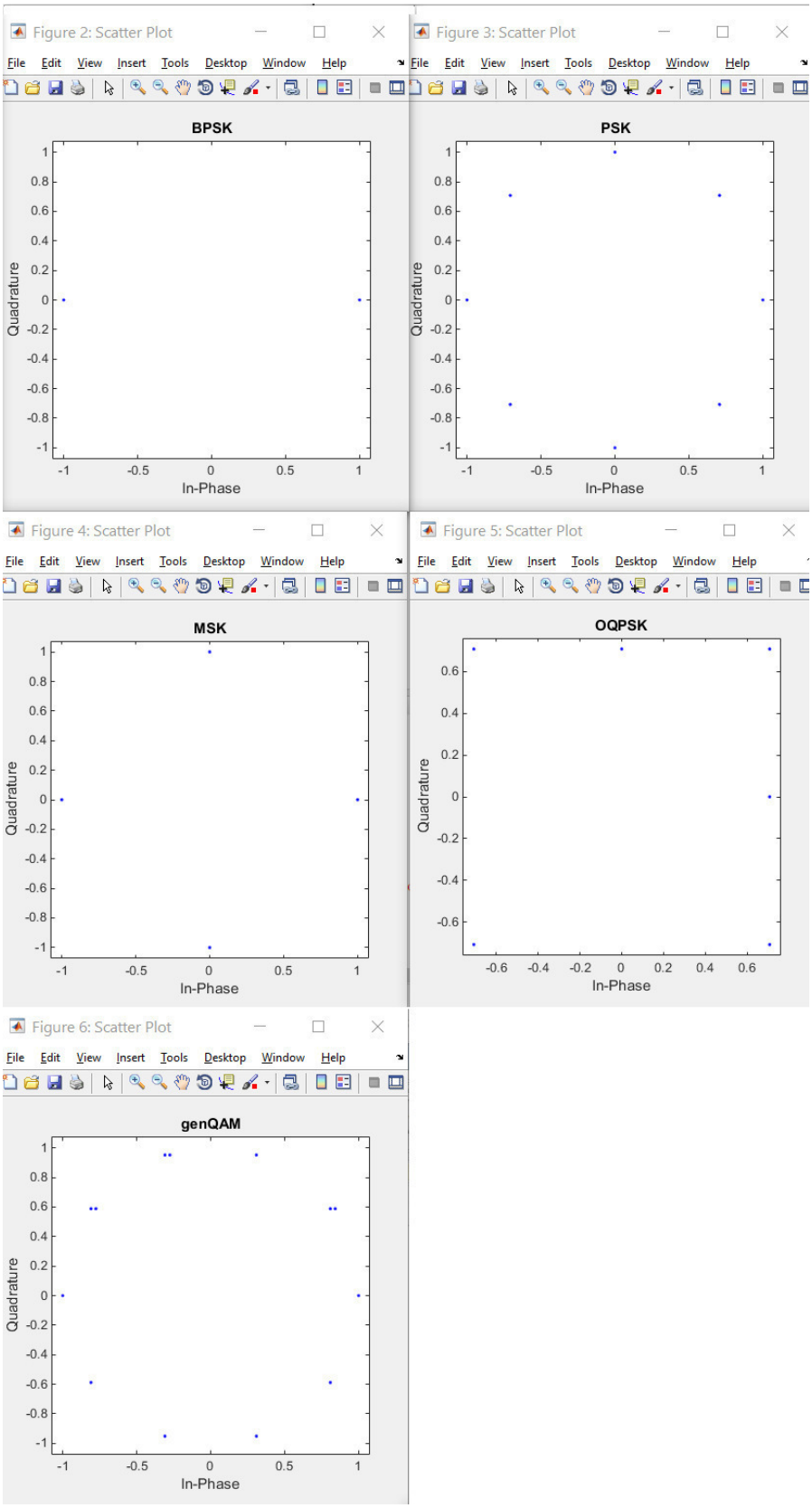

Рис. 1: Сигнальные созвездия

Листинг 1: Код MatLab

```
function n6


%BPSK
bpsk_modulator = modem.pskmod('M', 2);
bpsk_demodulator = modem.pskdemod('M', 2);
bpsk_msg = randi([0 1], 256/2, 1);
bpsk_modSignal = modulate(bpsk_modulator, bpsk_msg);
bpsk_demodSignal = demodulate(bpsk_demodulator, bpsk_modSignal);
iter = 1:32;
for i = iter
    err_sigmod = awgn(bpsk_modSignal, i);
    err_sigdemod = demodulate(bpsk_demodulator, err_sigmod);
    err = biterr(bpsk_msg, err_sigdemod);
    bpsk_fig(i) = err;
end


%PSK
psk_modulator = modem.pskmod('M', 8);
psk_demodulator = modem.pskdemod('M', 8);
psk_msg = randi([0 7], 256/8, 1);
psk_modSignal = modulate(psk_modulator, psk_msg);
psk_demodSignal = demodulate(psk_demodulator, psk_modSignal);
for i = iter
    err_sigmod = awgn(psk_modSignal, i);
    err_sigdemod = demodulate(psk_demodulator, err_sigmod);
    err = biterr(psk_msg, err_sigdemod);
    psk_fig(i) = err;
end


%MSK
msk_msg = randi([0 1], 256/2, 1);
msk_modSignal = mskmod(msk_msg, 1);
msk_demodSignal = mskdemod(msk_modSignal, 1);
for i = iter
    err_sigmod = awgn(msk_modSignal, i);
    err_sigdemod = mskdemod(err_sigmod, 1);
    err=biterr(msk_msg, err_sigdemod);
    msk_fig(i) = err;
end


%OQPSK
oqpsk_msg = randi([0 3], 256/4, 1);
oqpsk_modSignal = oqpskmod(oqpsk_msg);
for i = iter
    err_sigmod = awgn(oqpsk_modSignal, i);
    err_sigdemod = oqpskdemod(err_sigmod);
    err = biterr(oqpsk_msg, err_sigdemod);
    oqpsk_fig(i) = err;
end


%genQAM
M = 10;
qam_f = exp(1i*2*pi*[0:M-1]/M);
qam_h = modem.genqammod('Constellation', qam_f);
qam_g = modem.genqamdemod('Constellation', qam_f);
qam_msg = randi([0 9], 100, 1);
qam_modSignal = modulate(qam_h, qam_msg);
qam_errSignal = (randerr(1, 100, 3) ./ 30)';
qam_modSignal = qam_modSignal + qam_errSignal;
qam_demodSignal = demodulate(qam_g, qam_modSignal);
for i = iter
    err_sigmod = awgn(qam_modSignal, i);
    err_sigdemod = genqamdemod(err_sigmod, qam_f);
    err = biterr(qam_msg, err_sigdemod);
    qam_fig(i) = err;
end;

% Plots
figure;
hold on;
```

```
grid minor;
plot(iter, bpsk_fig);
plot(iter, psk_fig);
plot(iter, msk_fig);
plot(iter, oqpsk_fig);
plot(iter, qam_fig);
xlabel('noise');
ylabel('error');
title('Error-noise dependency');
legend('BPSK', 'PSK', 'MSK', 'OQPSK', 'genQAM');


scatterplot(bpsk_modSignal);
title('BPSK');

scatterplot(psk_modSignal);
title('PSK');

scatterplot(msk_modSignal);
title('MSK');

scatterplot(oqpsk_modSignal);
title('OQPSK');

scatterplot(qam_modSignal);
title('genQAM');

end
```

Рассмотрим водопадные кривые зависимости числа ошибок от шума в сигнале:

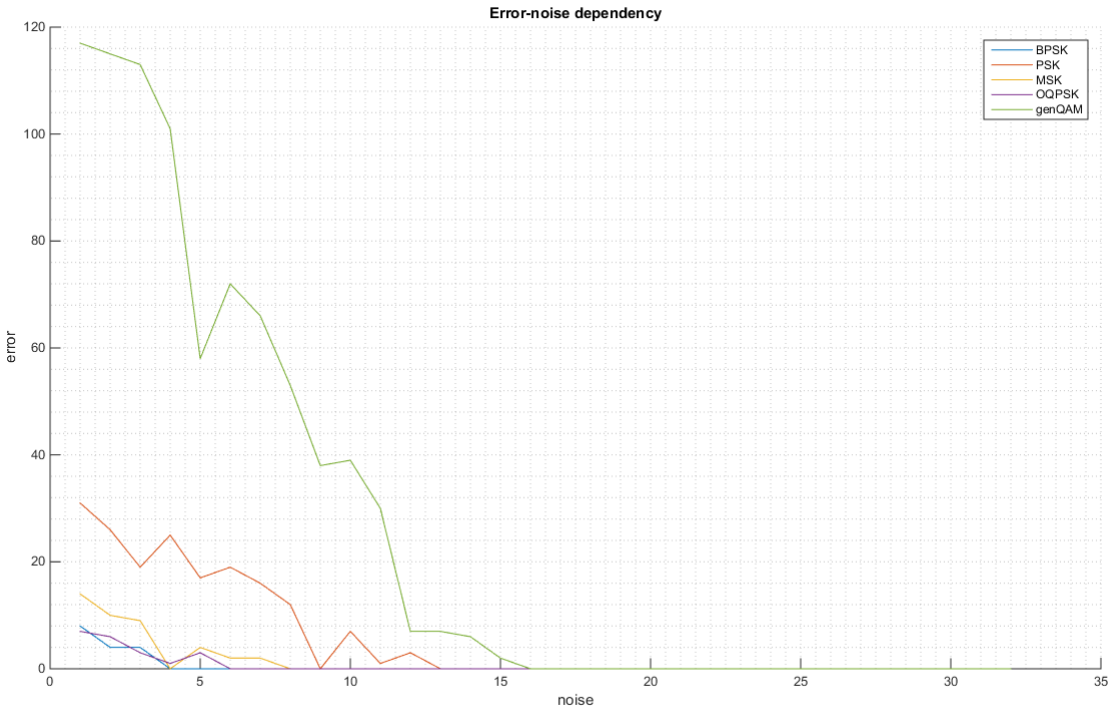

Рис. 2: Зависимость числа ошибок от SNR

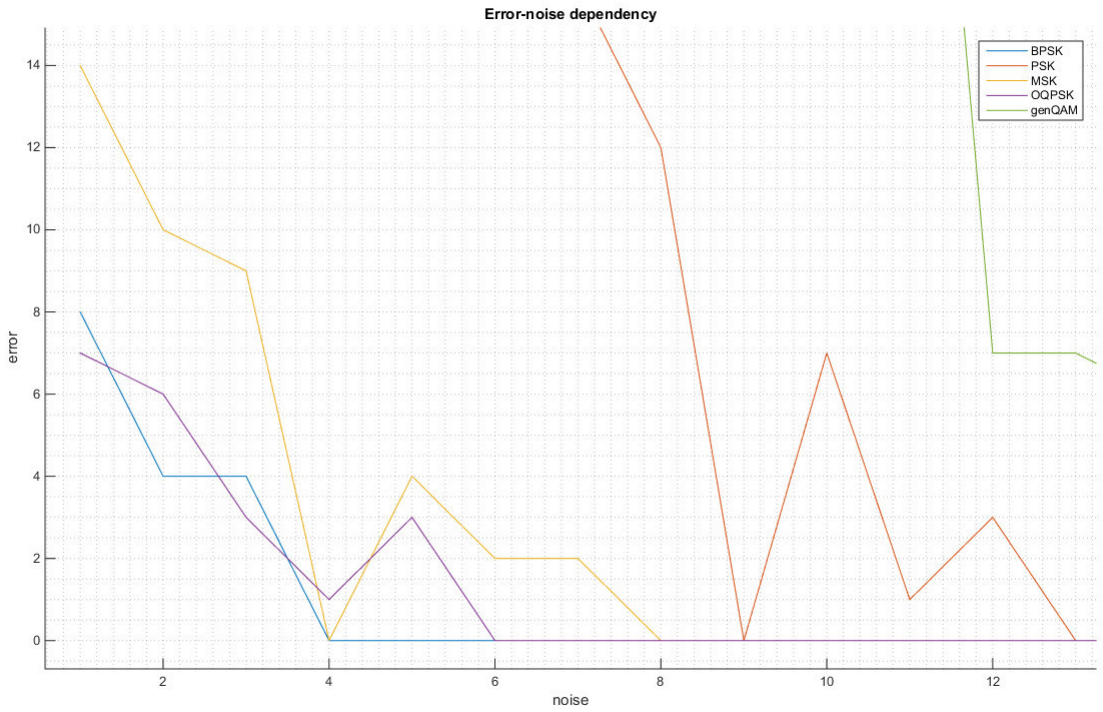

Рис. 3: Зависимость числа ошибок от SNR

Как видно по графику, наименее подверженные ошибкам метод модуляции - BPSK.

4.2 Выводы

Рассмотрим примененные методы цифровой модуляции и демодуляции:

- PSK: фаза несущего колебания имеет фиксированные позиции с одинаковым шагом, соответствующие цифровой посылке. С этим видом модуляции возникают проблемы синхронизации из-за трудности однозначной интерпретации поворота созвездия.
- QAM: изменение фазы и амплитуды, что дает повышенную информационную плотность.
- MSK: частотная манипуляции, в которой нет фазовых ступеней, а частота изменяется в момент пересечения несущей нулевого уровня.
- BPSK: имеет низкую плотность информации ввиду всего двух ступеней фаз, но обладает большой помехоустойчивостью благодаря большой дистанции между этими двумя состояниями.

Описанные методы имеют две основные характеристики: информационная плотность и помехоустойчивость, причем при увеличении величины одной характеристики уменьшается величина другой. Это объясняется различимостью состояний сигнала при демодуляции.

5 Кодирование и декодирование

Кодированием и декодированием (в широком смысле) называют любое преобразование сообщения в сигнал и обратно, сигнала в сообщение, путем установления взаимного соответствия. Преобразование следует считать оптимальным, если в конечном итоге производительность источника и пропускная способность канала окажутся равными, т.е. возможности канала будут полностью использованы. Данное преобразование разбивается на два этапа:

- модуляция-демодуляция, позволяющая осуществить переход от непрерывного сигнала радиоканала к дискретному;
- кодирование-декодирование (в узком смысле), во время которого все операции выполняются над последовательностью символов.

В свою очередь, кодирование-декодирование делится на два противоположных по своим действиям действиям этапа:

- устранение избыточности в принимаемом от источника сигнале (экономное кодирование);
- внесение избыточности в передаваемый по каналу цифровой сигнал (помехоустойчивое или избыточное кодирование) для повышения достоверности передаваемой информации.

Циклический код — линейный, блочный код, обладающий свойством цикличности, то есть каждая циклическая перестановка кодового слова также является кодовым словом. Используется для преобразования информации для защиты её от ошибок.

К циклическим кодам относятся коды Хэмминга, которые являются одним из немногочисленных примеров совершенных кодов. Они имеют кодовое расстояние $d = 3$ и исправляют все одиночные ошибки. Длина кода выбирается из условия $2^{n-k}-1 = n$, которое имеет простой смысл: число различных ненулевых синдромов равно числу символов в кодовой последовательности.

Среди циклических кодов широкое применение нашли коды Боуза-Чоудхури-Хоквингема (БЧХ). Можно показать, что для любых целых положительных чисел m и $l < n/2$ существует двоичный код БЧХ длины $n = 2^m - 1$, с кодовым расстоянием $d > 2l + 1$. Для кодов БЧХ умеренной длины и ФМ при передаче символов можно добиться значительного выигрыша (4 дБ и более). Он достигается при скоростях ($1/3 < k/n < 3/4$). При очень высоких и очень низких скоростях выигрыш от кодирования существенно уменьшается.

Частным случаем БЧХ кодов являются коды Рида-Соломона.

6 Matlab

Проведем кодирование и декодирование различными методами в MatLab. Далее представлены результаты кодирования в рабочем поле MatLab.

```
>> n7
 Hamming:
 Msg              = 1000
 Code             = 1101000
 Code with error = 1100000
 Decoded          = 1000
 Errors           = 1

 Hamming syndrome:
 Msg              = 1000
h =

     1     0     0     1     0     1     1
     0     1     0     1     1     1     0
     0     0     1     0     1     1     1


g =

     1     1     0     1     0     0     0
     0     1     1     0     1     0     0
     1     1     1     0     0     1     0
     1     0     1     0     0     0     1


 Syndrome         = 110
 Decoded          = 1101000

 Cyclic code:
 Msg              = 1000
 Pol              = 1011
 Code             = 1011000
 Code with error = 1001000
 Decoded          = 1011000

 BCH:
 Msg              = 1000
codebch =

  System: comm.BCHEncoder

  Properties:
```

```
              CodewordLength: 7
               MessageLength: 4
    PrimitivePolynomialSource: 'Auto'
    GeneratorPolynomialSource: 'Auto'
        PuncturePatternSource: 'None'


decbch =

  System: comm.BCHDecoder

  Properties:
                 CodewordLength: 7
                  MessageLength: 4
       PrimitivePolynomialSource: 'Auto'
       GeneratorPolynomialSource: 'Auto'
           PuncturePatternSource: 'None'
               ErasuresInputPort: false
    NumCorrectedErrorsOutputPort: true


 Code            = 1000101
 Code with error = 0000101
 Decoded         = 1000

 Reed-Solomon:
 Msg 1           = 21206715045001624442337036427O
 Code            =↙
21251660677161150441045071260165724244626042335253700474364311627O↙
6513
 Add 2 errors
 Decoded         = 21206715045001624442337036427O
 Errors          = 0

 Msg 2           = 55317736226070203674646032271O
 Code            =↙
55343241775153362276726024467022505036653574635104600262322434571O↙
7221
 Add 1 errors
 Decoded         = 55317736226070203674646032271O
 Errors          = 0
```

```
 Msg 3            = 47170151050742573546436167255 3
 Code             =↙
4717214701766051074635071264425235473571134647655361740267253045534↙
324
 Add 0 errors
 Decoded          = 47170151050742573546436167255 3
 Errors           = 0

>>
```

Листинг 2: Код MatLab

```
function n7

n = 7;
k = 4;

% Hamming
msg = randerr(1,k);
code = encode(msg,n,k);
i = randi([1 n]);
errcode = code;
errcode(i) = not(errcode(i));
[dec,err] = decode(errcode,n,k);

display('␣Hamming:␣');
fprintf('␣Msg␣␣␣␣␣␣␣␣␣␣␣␣=␣');
fprintf('%d', msg);
fprintf('\n␣Code␣␣␣␣␣␣␣␣␣␣␣=␣');
fprintf('%d', code);
fprintf('\n␣Code␣with␣error␣=␣');
fprintf('%d', errcode);
fprintf('\n␣Decoded␣␣␣␣␣␣␣␣␣=␣');
fprintf('%d', dec);
fprintf('\n␣Errors␣␣␣␣␣␣␣␣␣␣=␣');
fprintf('%d␣\n\n', err);


% Hamming syndrome
display('␣Hamming␣syndrome:␣');
fprintf('␣Msg␣␣␣␣␣␣␣␣␣␣␣␣=␣');
fprintf('%d', msg);

[h,g,n,k] = hammgen(3);
h
g

m = msg * g;
m = rem(m, ones(1,n) .* 2);
m(4) = not(m(4));
synd = m*h';
fprintf('\n Syndrome         = ');
fprintf('%d',␣synd);

synd␣=␣rem(synd,␣ones(1,n-k)␣.*␣2);
stbl␣=␣syndtable(h);
tmp␣=␣bi2de(synd,'left-msb');
z␣=␣stbl(tmp+1,:);
res␣=␣xor(m,z);
fprintf('\n Decoded          = ');
fprintf('%d',␣res);
fprintf('\n\n');


%␣Cyclic␣code
display(' Cyclic code: ');
fprintf(' Msg              = ');
fprintf('%d',␣msg);

pol␣=␣cyclpoly(n,k);
fprintf('\n Pol              = ');
fprintf('%d',␣pol);
[h,g]␣=␣cyclgen(n,pol);
code␣=␣msg*g;
code␣=␣rem(code,ones(1,n).*2);
i␣=␣randi([1␣n]);
errcode␣=␣code;
errcode(i)␣=␣not(errcode(i));
synd␣=␣errcode*h';
synd = rem(synd,ones(1,n-k).*2);
stbl = syndtable(h);
tmp = bi2de(synd,'left-msb');
z = stbl(tmp+1,:);
res = xor(errcode,z);
fprintf('\n␣Code␣␣␣␣␣␣␣␣␣␣␣=␣');
```

```
fprintf('%d', code);
fprintf('\n Code with error = ');
fprintf('%d', errcode);
fprintf('\n Decoded          = ');
fprintf('%d', res);
fprintf('\n\n');


% BCH
display(' BCH: ');
fprintf(' Msg              = ');
fprintf('%d', msg);

codebch = comm.BCHEncoder(n,k)
decbch = comm.BCHDecoder(n,k)
temp = msg';
code = step (codebch , temp(:))';
i = randi([1 n]);
errcode = code;
errcode(i) = not(errcode(i));
fprintf('\n Code             = ');
fprintf('%d', code);
fprintf('\n Code with error = ');
fprintf('%d', errcode);

res = step (decbch , code')';
fprintf('\n Decoded          = ');
fprintf('%d', res);
fprintf('\n\n');


% Reed-Solomon
display(' Reed-Solomon: ');
hEnc = comm.RSEncoder;
hDec = comm.RSDecoder;
hError = comm.ErrorRate('ComputationDelay',3);

for i = 1 : 3
    msg = randi([0 7], 30, 1);

    code = step(hEnc, msg);
    errcode = code;
    for j = i+1 : 3
        it = randi([1 30]);
        if (errcode(it) > 0)
            errcode(it) = errcode(it) - 1;
        else
            errcode(it) = errcode(it) + 1;
        end
    end

    dec = step(hDec, code);
    errorStats = step(hError, msg, dec);

    fprintf(' Msg %g             = ', i);
    fprintf('%d', msg);
    fprintf('\n Code             = ');
    fprintf('%d', code);
    fprintf('\n Add %g errors', 3-i);
    fprintf('\n Decoded          = ');
    fprintf('%d', dec);
    fprintf('\n Errors           = %d\n\n', errorStats(2));
end

end
```

7 Выводы

В данной лабораторной работе были рассмотрены коды Хэмминга, циклические коды, коды БЧХ и Рида-Соломона. Корректирующая способность кодов БЧХ лучше по сравнению с кодами Хэмминга, которые имеют простую реализацию. Код Рида-Соломона может обрабатывать две ошибки, тогда как остальные самокорректирующие коды обрабатывают одну.