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

Лабораторная работа №7

Помехоустойчивое кодирование

Руководитель

_____ Богач Н. В.

Выполнил

_____ Солдатова Е. И.
группа 33501/3

Санкт-Петербург
2018

1. Цель работы

Изучение методов помехоустойчивого кодирования и сравне- ние их свойств

2. Постановка задачи

1) 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.

2) Выполнить кодирование/декодирование циклическим кодом, кодом БЧХ, кодом Рида-Соломона. Оценить корректирующую способность кода.

3. Теоретическая часть

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, корректирующая способность которых, соответственно, ниже.

4. Ход работы

Результат кодирования/декодирования кодом Хэмминга с использованием стандартных функций:

```
msg = [0 1 1 1]
code = encode(msg,7,4)
code(1) = not(code(1))
dec,err = decode(code,7,4)
```

Результат работы программы:

```
msg =

     0     1     1     1


code =

     0     0     1     0     1     1     1


code =

     1     0     1     0     1     1     1


dec =

     0     1     1     1


err =

     1
```

Результат кодирования/декодирования кодом Хэмминга с использованием проверочной и генераторной матриц и вычисление синдрома:

```
msg = [0 1 1 1]
h,g,n,k = hammgen(3)
m = msg*g;
m = rem(m,ones(1,n).*2);
m(4) = not(m(4));
synd = m*h';
synd = rem(synd,ones(1,n-k).*2);
stbl = syndtable(h);
tmp = bi2de(synd,'left-msb')
z = stbl(tmp+1,:)
rez = xor(m,z)
```

Результат работы программы:

```
stbl =
msg =

     0     1     1     1


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


n =

     7


k =

     .


synd =

     1     1     0


tmp =

     6


z =

     0     0     0     1     0     0     0


rez =

  1×7 logical array

   0   0   1   0   1   1   1
```

Результат кодирования/декодирования циклическим кодом:

```
msg = [0 1 1 1]
pol = cyclpoly(7,4)
h,g = cyclgen(7,pol);
code = msg*g;
code = rem(code,ones(1,n).*2);
code(2) = not(code(2));
synd = code*h';
synd = rem(synd,ones(1,n-k).*2);
stbl = syndtable(h);
```

```
tmp = bi2de(synd,'left-msb')
z = stbl(tmp+1,:)
rez = xor(code,z)
```

Результат работы программы:

```
msg =

     0     1     1     1


pol =

     1     0     1     1


tmp =

     2


z =

     0     1     0     0     0     0     0


rez =

  1×7 logical array

   0   1   0   0   1   1   1
```

Результат кодирования/декодирования кодом БЧХ:

```
msg = [0 1 1 1]
codebch = comm.BCHEncoder(7,4)
decbch = comm.BCHDecoder(7,4)
temp = msg';
code = step (codebch , temp(:))'
code(2) = not(code(2))
decode = step (decbch , code')'
```

Результат работы программы:

```
msg =

     0     1     1     1


codebch =

  comm.BCHEncoder with properties:

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

```
decbch =

  comm.BCHDecoder with properties:

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


code =

     0     1     1     1     0     1     0


code =

     0     0     1     1     0     1     0


decode =

     0     1     1     1
```

Результат кодирования/декодирования кодом Рида-Соломона:

```
m = 3;
n = 2m̂ - 1;
k = 3;
msg = gf(0 1 2; 3 4 5; 6 7 6,m)
code = rsenc(msg,n,k)
errs = gf([0 0 0 4 0 0 0; 2 0 0 0 2 0 0; 3 4 5 0 0 0 0 ],m);
code = code + errs
dec,errnum = rsdec(code,n,k)
```

Результат работы программы:

```
decbch =

msg = GF(2^3) array. Primitive polynomial = D^3+D+1 (11 decimal)

Array elements =

   0   1   2
   3   4   5
   6   7   6


code = GF(2^3) array. Primitive polynomial = D^3+D+1 (11 decimal)

Array elements =

   0   1   2   2   3   1   3
   3   4   5   3   2   2   4
   6   7   6   2   7   3   3


code = GF(2^3) array. Primitive polynomial = D^3+D+1 (11 decimal)

Array elements =

   0   1   2   6   3   1   3
   1   4   5   3   0   2   4
   5   3   3   2   7   3   3
```

```
decbch =

dec = GF(2^3) array. Primitive polynomial = D^3+D+1 (11 decimal)

Array elements =

   0   1   2
   3   4   5
   5   3   3


errnum =

     1
     2
    -1
```

5. Выводы

В ходе данной работы были получены навыки кодирования цифровых сигналов. Кодирование таких сигналов происходит по принципу избыточности. Каждый из исследованных кодов имеет свои преимущества и недостатки, поэтому использование конкретного из них должно быть обусловлено постановкой определенной задачи. Код Хэмминга достаточно простой в использовании, не требует больших мощностей. Однако он может исправить только одну допущенную ошибку в переданном сообщении. Код Рида-соломона способен исправлять несколько ошибок, так же он может оперировать десятичными числами, а не только двоичными.