Федеральное государственное автономное образовательное учреждение высшего образования «Санкт-Петербургский политехнический университет Петра Великого»
Институт компьютерных наук и технологий
Кафедра компьютерных систем и программных технологий

Телекоммуникационные технологии

Лабораторная работа №8:
Модель телекоммуникационного канала

Работу выполнил:
Сергеев А.А.
Группа: 33531/2
Преподаватель:
Богач Н.В.

Санкт-Петербург
2019

Содержание

1 Цель

Создать модель телекоммуникационного канала.

2 Постановка задачи

Пакетный сигнал длительностью 200 мкс состоит из 64 бит полезной информации и 8 нулевых tail-бит. В нулевом 16-битном слове пакета передается ID, в первом – период излучения в мс, во втором – сквозной номер пакета, в третьем – контрольная сумма (CRC-16). На передающей стороне пакет сформированный таким образом проходит следующие этапы обработки:

1. Помехоустойчивое кодирование сверточным кодом с образующими полиномами 753, 561 (octal) и кодовым ограничением 9. На выходе кодера количество бит становится равным 144.
2. Перемежение бит. Количество бит на этом этапе остается неизменным.
3. Модуляция символов. На этом этапе пакет из 144 полученных с выхода перемежителя бит разбивается на 24 символа из 6 бит. Генерируется таблица функций Уолша длиной 64 бита. Каждый 6-битный символ заменяется последовательностью Уолша, номер которой равен значению данных 6-ти бит. Т.о. на выходе модулятора получается $24 * 64 = 1536$ знаковых символов.
4. Прямое расширение спектра. Полученная последовательность из 1536 символов периодически умножается с учетом знака на ПСП длиной 511 символов. Далее к началу сформированного символьного пакета прикрепляется немодулированная ПСП. Т.о. символьная длина становится равной 2047. Далее полученные символы модулируются методом BPSK.

По имеющейся записи сигнала из эфира и коду модели передатчика создать модель приемника, в которой найти позицию начала пакета и, выполнив операции демодуляции, деперемежения и декодирования, получить передаваемые параметры: ID, период и номер пакета. Запись сделана с передискретизацией 2, т.е. одному BPSK символу соответствуют 2 лежащих друг за другом отсчета в файле. Известно, что $ID = 4$, период 100 мс, номер пакета 373. Запись сделана с передискретизацией 2, т.е. одному BPSK символу соответствуют 2 лежащих друг за другом отсчета в файле. Запись сделана на нулевой частоте и представляет из себя последовательность 32-х битных комплексных отсчетов, где младшие 16 бит вещественная часть, старшие 16 бит – мнимая часть.

3 Ход работы

3.1 Помехоустойчивое кодирование

Помехоустойчивое кодирование сверточным кодом с образующими полиномами 755, 561 (octal) и кодовым ограничением 9:

```
1 trellis = poly2trellis(9, [753 561]);
2 convolved_signal = convenc(data, trellis);
```

Функция $poly2trellis$ принимает в качестве аргументов кодовое ограничение и образующие полиномы, возвращая структуру $trellis$, с помощью которой в последующем методе $convenc$ происходит кодирование сообщения. Метод $convenc$ осуществляет само кодирование, принимая в качестве аргументов изначальное сообщение и структуру $trellis$. Каждый символ в исходном сообщении состоит из $\log_2(trellis.numInputSymbols)$, в закодированном – из $\log_2(trellis.numOutputSymbols)$. Размер исходного сообщения – 72 бита, длина закодированных данных – 144 бита.

3.2 Перемежение бит

На этом этапе количество бит остаётся неизменным. На этом этапе происходит процесс перестановки бит в сообщении, чтобы избежать последовательных ошибок при кодировании. Таблица перемежения – это перестановка чисел от 0 до 143 (по длине сообщения).

```
interleaver = int16([  0; 133; 122; 111; 100;  89;  78;  67;  56;  45;  34;  23;
                      12;   1; 134; 123; 112; 101;  90;  79;  68;  57;  46;  35;
                      24;  13;   2; 135; 124; 113; 102;  91;  80;  69;  58;  47;
                      36;  25;  14;   3; 136; 125; 114; 103;  92;  81;  70;  59;
                      48;  37;  26;  15;   4; 137; 126; 115; 104;  93;  82;  71;
                      60;  49;  38;  27;  16;   5; 138; 127; 116; 105;  94;  83;
                      72;  61;  50;  39;  28;  17;   6; 139; 128; 117; 106;  95;
                      84;  73;  62;  51;  40;  29;  18;   7; 140; 129; 118; 107;
                      96;  85;  74;  63;  52;  41;  30;  19;   8; 141; 130; 119;
                     108;  97;  86;  75;  64;  53;  42;  31;  20;   9; 142; 131;
                     120; 109;  98;  87;  76;  65;  54;  43;  32;  21;  10; 143;
                     132; 121; 110;  99;  88;  77;  66;  55;  44;  33;  22;  11])

interleaved_signal = convolved_signal(interleaver+1);
```

3.3 Модуляция

Сначала полученные после перемежения 144 бита разбиваются на 24 символа из 6 бит. Затем генерируется таблицу функций Уолша длиной 64 бита. Для этого каждый 6-битный символ заменяется последовательностью Уолша, номер которой равен значению данных 6-ти бит. Функции Уолша – это семейство функций, образующих ортогональную систему, принимающих на всей области определения значения 1 и -1.

```
signal_matrix = reshape(interleaved_signal(1:N*2), [N*2/6, 6]);
y = int16(hadamard(64));
row_number = int16(zeros(1, N*2/6));
for k=1:N*2/6
    line = signal_matrix(k,:);
    row_number(k) = bi2de(line) + 1;
end

signal = y(row_number, :);
signal_to_modulate = reshape(signal', 1, []);
```

После этого на выходе получаем $144 \cdot 6 = 1536$ знаковых символов.

3.4 Прямое расширение спектра

Расширение спектра производится методом прямой последовательности, повышая тактовую частоту модуляции. Каждому символу из сообщения ставится в соответствие некоторая достаточно длинная псевдослучайная последовательность. Здесь берется ПСП длиной 511 символов:

```
PRS = int16([1;  1;  1;  1;  1;  1; -1; -1; -1; -1;  1;  1;  1;  1; -1;  1;
             1;  1; -1; -1; -1; -1;  1; -1;  1;  1; -1; -1;  1;  1; -1;  1;
             1; -1;  1;  1;  1;  1; -1;  1; -1; -1; -1; -1;  1;  1;  1; -1;
            -1;  1;  1; -1; -1; -1; -1;  1; -1; -1;  1; -1; -1; -1;  1; -1;
             1; -1;  1;  1;  1; -1;  1; -1;  1;  1;  1;  1; -1; -1;  1; -1;
            -1;  1; -1;  1;  1;  1; -1; -1;  1;  1;  1; -1; -1; -1; -1; -1;
            -1;  1;  1;  1; -1;  1;  1;  1; -1;  1; -1; -1;  1;  1;  1;  1;
            -1;  1; -1;  1; -1; -1;  1; -1;  1; -1; -1; -1; -1; -1;  1;  1;
            -1;  1; -1;  1; -1;  1; -1;  1;  1;  1;  1;  1; -1;  1; -1;  1;
             1; -1;  1; -1; -1; -1; -1; -1;  1;  1; -1;  1;  1;  1; -1;  1;
             1; -1;  1;  1; -1; -1; -1; -1; -1;  1;  1; -1; -1; -1; -1; -1;
             1;  1;  1; -1;  1;  1;  1;  1; -1; -1; -1;  1;  1;  1;  1;  1;
            -1; -1;  1;  1; -1;  1; -1; -1;  1;  1; -1;  1; -1;  1;  1;  1;
            -1; -1; -1;  1;  1; -1;  1; -1; -1; -1;  1; -1;  1;  1;  1;  1;
             1;  1;  1; -1;  1; -1; -1;  1; -1;  1;  1; -1; -1; -1; -1; -1;
             1; -1; -1; -1;  1; -1; -1; -1;  1;  1; -1; -1; -1; -1; -1; -1;
            -1;  1;  1;  1;  1; -1;  1;  1; -1; -1; -1; -1;  1;  1; -1;  1;
            -1;  1; -1; -1;  1;  1;  1;  1;  1;  1; -1;  1;  1; -1;  1; -1;
            -1;  1; -1; -1;  1; -1; -1;  1;  1; -1;  1;  1;  1;  1;  1;  1;
            -1; -1;  1; -1;  1; -1;  1; -1; -1; -1;  1; -1;  1;  1; -1; -1;
             1; -1;  1;  1;  1;  1; -1; -1; -1; -1;  1;  1; -1; -1; -1;  1;
            -1;  1;  1; -1; -1; -1; -1;  1; -1; -1; -1; -1;  1;  1;  1; -1;
             1;  1; -1;  1;  1;  1;  1; -1; -1; -1; -1; -1;  1;  1;  1  ]);

n_repeat = int16(length(signal_to_modulate) / length(PRS'));
n_compl  = length(signal_to_modulate) - length(PRS') * n_repeat;
```

```
36
37 signal_to_modulate = signal_to_modulate .* [repmat(PRS', 1, n_repeat), PRS(1:n_compl)'];
```

И к началу сформированной последовательности добавляется немодулированная ПСП:

```
1 signal_to_modulate = [PRS', signal_to_modulate];
```

В конце этого этапа мы имеем сигнал длиной 1747 символов.

3.5 BPSK

На этом этапе прозводится модуляция методом BPSK. Запись сделана с передискретизацией 2. Запись сделана на нулевой частоте и представляет из себя последовательность 32-битных комплексных отсчетов, где младшие 16 бит вещественная часть, а старшие – мнимая.

```
1 IQ = pskmod(double(signal_to_modulate), 2);
2
3 IQ_oversampled = [IQ, IQ];
4 IQ_im_part=imag(IQ_oversampled);
5 IQ_re_part=real(IQ_oversampled);
6 IQ_record = [IQ_re_part, IQ_im_part];
```

3.6 CRC

CRC высчитывается следующим образом:

```
1 crc_gen = crc.generator('Polynomial', '0x1021');
2 crc = generate(crc_gen, data');
3 crc_send = crc(73:88);
```

3.7 Декодирование

Для сверточного декодирования используется алгоритм Витерби – рекуррентная процедура, направленная на поиск пути по кодовой решетке, ближайшего к принимаемой последовательности. Пример использования сверточного кодирования и декодирования алгоритмом Витерби представлен ниже:

```
1 trellis = poly2trellis(4,[7 10]);
2 data = [1 1 1 1 0 1 1 1 0 1 1 0 1 1 0];
3
4 code = convenc(data, trellis);
5 decoded = vitdec(code, trellis, 3, 'trunc', 'hard');
```

Вероятность ошибки в этом случае:

```
1 [n1,r1] = biterr(decoded(3+1:end),data(1:end-3));
```

Получаем вероятность, равную 0.25.

4 Вывод

В работе была представлена модель телекоммуникационного канала, а также этапы обработки сигнала у передатчика. Было рассмотрено помехоустойчивое кодирование сверточным кодом, а также декодирование сигнала алгоритмом Витерби. Особенность сверточных кодов в том, что они работают с сигналом как со сплошным потоком данных. Декодирование таких кодов обычно происходит с использованием алгоритма Витерби, который пытается восстановить переданную последовательность согласно критерию максимального правдоподобия.
Сверточные коды эффективно работают в канале с белым шумом, но плохо справляются с пакетами ошибок.