# Телекоммуникационные технологии

*Описание лабораторных работ*

19 февраля 2018 г.

# Содержание

Лабораторный практикум по курсу телекоммуникационные технологии охватывает нижние уровни эталонной модели взаимодействия открытых систем ISO/OSI. На физическом уровне основные методы преобразования сигналов при передаче по каналу связи это аналоговая и цифровая модуляция/демодуляция, потенциальное и импульсное кодирование, фильтрация, расширение спектра.

Лабораторные работы посвящены моделированию преобразования сигналов с каналах связи в пакетах MatLab/Octave.

Работы 1-3 преследуют целью освоение приемов работы и необходимого минимума команд MatLab/Octave, кроме этого они требуют повторения основ частотно-временного анализа сигналов и способствуют развитию у студентов частотно-временных представлений, что облегчает в дальнейшем понимание физических принципов работы современных технологий связи в работах 4-7.

При составлении отчетов по выполненным работам целесообразно включить в них следующие пункты:

1. Название работы
2. Цель
3. Постановка задачи
4. Теоретический раздел, содержащий основные соотношения между наблюдаемыми в работе явлениями
5. Ход работы
6. Выводы, поясняющие моделируемые явления.

# Элементы частотно-временного анализа

## Лабораторная работа №1. Сигналы телекоммуникационных систем

- Цель: Познакомиться со средствами генерации и визуализации простых сигналов.
- Постановка задачи: В командном окне MATLAB и в среде Simulink промоделировать синусоидальный и прямоугольный сигналы с различными параметрами. Получить их спектры. Вывести на график.
- В выводах укажите признаки классификации сигналов.

## Лабораторная работа №2. Ряд Фурье. Преобразование Фурье. Корреляция

- Цель: Получить представление о спектрах телекоммуникационных сигналов.
- Постановка задачи:
  - Для сигналов, построенных в лабораторной работе №1, выполните расчет преобразования Фурье. Перечислите свойства преобразования Фурье.
  - С помощью функции корреляции найдите позицию синхропосылки [101] в сигнале [0001010111000010]. Получите пакет данных, если известно, что его длина составляет 8 бит без учета синхропосылки. Вычислите корреляцию прямым методом, воспользуйтесь алгоритмом быстрой корреляции, сравните время работы обоих алгоритмов.
  - Быстрая корреляция
- В выводах расскажите о примерах применения преобразования Фурье в телекоммуникационных технологиях.

## Лабораторная работа №3. Линейная фильтрация

- Цель: изучить воздействие ФНЧ на тестовый сигнал с шумом.
- Постановка задачи: сгенерировать гармонический сигнал с шумом и синтезировать ФНЧ. Получить сигнал во временной и частотной областях до и после фильтрации. Сделать выводы о воздействии ФНЧ на спектр сигнала.
- Теоретические положения к лабораторной работе: Справочные материалы: В.С. Гутников Фильтрация измерительных сигналов, Глава 5, сс. 70-73, рис.20.

  Для создания модели в Simulink используйте блок Discrete FIR Filter раздела Discrete главной библиотеки и блок Digital Filter design из Signal Processing Blockset/Filtering/Filter Designs.
- В выводе опишите процесс прохождения сигнала через линейную цепь на примере фильтра и поясните неполное удаление шума линейным фильтром.

# Технологии модуляции

### Функции модуляции/демодуляции

Справочные материалы:

1. А.Б. Сергиенко Цифровая обработка сигналов. Функции модуляции и демодуляции пакета Signal Processing, пакета Communications сс.481–507;

2. Справочник по MatLab)

Для передачи по любому каналу связи цифровое сообщение, представляющее собой последовательность символов (чисел), необходимо преобразовать в аналоговый сигнал — изменяющуюся во времени физическую величину (например, напряжение). Кроме того, канал связи способен пропускать лишь определенную полосу частот. Преобразование сигнала для переноса в заданный частотный диапазон осуществляется путем модуляции. Обратный процесс назвается демодуляцией. Пакет MatLab Communications содержит функции для реализации аналоговой и цифровой модуляции/демодуляции. При аналоговой модуляции входным сигналом является непрерывная функция, при цифровой — последовательность символов. Модулированный сигнал может представляться либо в вещественном виде (passband simulation), либо в виде комплексной огибающей (baseband simulation). Соответственно приведенной классификации имеется 8 функций: (в новых версиях пакета, начиная с Matlab 2008, имеется возможность использовать класс modem для построения модемов, для справки: help modem/types)

```
ammod - аналоговая модуляция, вещественный выходной сигнал;
ammodce - аналоговая модуляция, выходной сигнал в виде комплексной огибающей;
ademod - аналоговая демодуляция, вещественный входной сигнал;
ademodce - аналоговая демодуляция, входной сигнал в виде комплексной огибающей;
dmod - цифровая модуляция, вещественный выходной сигнал;
dmodce - цифровая модуляция, выходной сигнал в виде комплексной огибающей;
ddemod - цифровая демодуляция, вещественный входной сигнал;
ddemodce - цифровая демодуляция, входной сигнал в виде комплексной огибающей;
```

Функциями пакета поддерживаются следующие виды аналоговой модуляции:

- амплитудная модуляция;

- амплитудная модуляция с подавленной несущей;

- однополосная модуляция;
- частотная модуляция;
- фазовая модуляция;
- квадратурная модуляция

При цифровой модуляции возможны следующие ее виды:

- амплитудная манипуляция;
- частотная манипуляция;
- минимальная частотная манипуляция;
- фазовая манипуляция;
- квадратурная манипуляция

Цифровая модуляция и демодуляция включают в себя две стадии. При модуляции цифровое сообщение сначала преобразуется в аналоговый модулирующий сигнал с помощью функции modmap, а затем осуществляется аналоговая модуляция. При демодуляции сначала получается аналоговый демодулированный сигнал, а затем он преобразуется в цифровое сообщение с помощью функции demodmap. Три последних функции этой группы предназначены для работы с конкретными сигнальными созвездиями квадратурной манипуляции. Функции qaskenco и qaskdeco производят кодирование и декодирование сообщения с использованием "квадратного" созвездия, а функция apkconst позволяет вывести на экран изображение "концентрического" созвездия.

### Моделирование каналов связи

Сформированный в результате модуляции сигнал поступает в канал связи, где он подвергается воздействию шумов и помех. Поэтому функции моделирования каналов связи должны обеспечивать внесение в сигнал искажений согласно используемым статистическим моделям.

Данная группа функций пакета Communications в данный момент представлена лишь одной функцией

`awgn`

которая позволяет добавить к сигналу аддитивный белый нормальный шум, обеспечивая при этом заданное отношение сигнал/шум.

## Лабораторная работа №4. Аналоговая модуляция

- Цель: изучение амплитудной модуляции/демодуляции сигнала.
- Постановка задачи:
  1. Сгенерировать однотональный сигнал низкой частоты.
  2. Выполнить амплитудную модуляцию (АМ) сигнала по закону $u(t) = (1 + MU_m \cos(\Omega t)) \cos(\omega_0 t + \phi_0)$ для различных значений глубины модуляции M. Используйте встроенную функцию MatLab ammod [1]
  3. Получить спектр модулированного сигнала.
  4. Выполнить модуляцию с подавлением несущей $u(t) = MU_m \cos(\Omega t) \cos(\omega_0 t + \phi_0)$. Получить спектр.
  5. Выполнить однополосную модуляцию:

     $$u(t) = U_m \cos(\Omega t) \cos(\omega_0 t + \phi_0) + \frac{U_m}{2} \sum_{n=1}^{N} M_n (\cos(\omega_0 + \Omega_n)t + \phi_0 + \Phi_n)$$

     положив n=1
  6. Выполнить синхронное детектирование и получить исходный однополосный сигнал.
  7. Рассчитать КПД модуляции.

     $$\eta_A M = \frac{U_m^2 M^2/4}{P_U} = \frac{M^2}{M^2 + 2}$$

- Справочные материалы:
  1. Н.В. Богач и др. Обработка сигналов в информационных системах, с. 110-118, 125-127

[1] Используйте при небходимости помощь MatLab: help <*имя функции*>

## Лабораторная работа №5. Частотная и фазовая модуляция

- Цель: изучение частотной и фазовой модуляции/демодуляции сигнала.
- Постановка задачи:
  1. Сгенерировать однотональный сигнал низкой частоты.
  2. Выполнить фазовую модуляцию/демодуляцию сигнала по закону $u(t) = (U_m \cos(\Omega t + ks(t))$, используя встроенную функцию MatLab pmmod, pmdemod
  3. Получить спектр модулированного сигнала.
  4. Выполнить частотную модуляцию/демодуляцию по закону

     $$u(t) = U_m \cos(\omega_0 t + k \int_0^t s(t)dt + \phi_0)$$

     используя встроенные функции MatLab fmmod, fmdemod
- Справочные материалы:
  1. Н.В. Богач и др. Обработка сигналов в информационных системах, с. 118-125, 127-133

## Лабораторная работа №6. Цифровая модуляция

- Цель: изучение методов модуляции цифровых сигналов.

  Функция randerr предназначена для моделирования ошибок в канале. Она возвращает матрицу, в каждой строке которой имеется заданное число случайно расположенных ненулевых элементов.

  Для оценки помехоустойчивости системы связи необходимо произвести сравнение исходного (передаваемого) сообщения с сообщением, полученным в результате приема, и определить число ошибок, возникших в процессе передачи, а также вероятность ошибки. Это можно выполнить функциями symerr и biterr, первая из которых подсчитывает число несовпадающих символов в двух сообщениях, а вторая — число несовпадающих битов в двоичных представлениях этих символов. Кроме числа ошибок, обе функции могут возвращать долю ошибок в общем числе символов (битов) и индикаторы мест возникновения ошибок.

  Последние две функции данной группы предназначены для графического отображения сигналов с квадратурной манипуляцией. Функция eyediagram выводит глазковую диаграмму, а функция scatterplot — диаграмму рассеяния.

- Постановка задачи:

  1. Получить сигналы BPSK, PSK, OQPSK, genQAM, MSK, M-FSK модуляторов [2]
  2. Построить их сигнальные созвездия
  3. Провести сравнение изученных методов модуляции цифровых сигналов

```
% EXAMPLE: Construct modulation objects
% to perform QPSK modulation and demodulation

h = modem.pskmod('M', 4);              % Modulator object
g = modem.pskdemod('M', 4);            % Demodulator object
msg = randint(10,1,4);                 % Modulating message
modSignal = modulate(h,msg);           % Modulate signal
demodSignal = demodulate(g,modSignal); % Demodulate signal
```

- Справочные материалы:

[2] Используйте "help modem/types"для справки по конструктору объектов modem

1. Н.В. Богач и др. Обработка сигналов в информационных системах, с. 137-141

# Моделирование телекоммуникационных каналов

Функции encode и decode осуществляют, соответственно, кодирование и декодирование сообщения с использованием блочного кода. Тип используемого кода задается в числе параметров функций. Линейный блочный код в общем случае описывается порождающей матрицей (generator matrix). Кодирование блока (вектора) производится путем его умножения на порождающую матрицу. При контроле ошибок на приемной стороне используется проверочная матрица кода (parity-check matrix). Преобразование порождающей матрицы в проверочную и обратно осуществляется функцией gen2par. Если умножение кодированного блока на проверочную матрицу не дает нулевого вектора, то полученный результат (его называют синдром — syndrome) позволяет определить, какие именно символы были искажены в процессе передачи. Для двоичного кода это позволяет исправить ошибки. Декодирование линейного блочного кода, таким образом, можно осуществить с помощью таблицы, в которой для каждого значения синдрома указан соответствующий вектор ошибок. Создать такую таблицу на основании проверочной матрицы кода позволяет функция syndtable. Функция gfweight позволяет определить кодовое расстояние для линейного блочного кода по его порождающей или проверочной матрице.

*Циклические коды*

Циклические коды — это подкласс линейных кодов, обладающие тем свойством, что циклическая перестановка символов в кодированном блоке дает другое возможное кодовое слово того же кода. Для работы с циклическими кодами в пакете Communications есть две функции. Задав число символов в кодируемом и закодированном блоках, с помощью функции cyclpoly можно получить порождающий полином циклического кода. Далее, использовав этот полином в качестве одного из параметров функции cyclgen, можно получить порождающую и проверочную матрицы для данного кода. *Коды БЧХ* Коды БЧХ являются одним из подклассов циклических блочных кодов. Для работы с ними функции высокого уровня вызывают специализированные функции bchenco (кодирование) и bchdeco (декодирование). Кроме того, функция bchpoly позволяет рассчитывать параметры или порождающий полином для двоичных кодов БЧХ.

*Коды Хэмминга*

Коды Хэмминга являются одним из подклассов циклических блочных кодов. Порождающий полином для кодов Хэмминга неприводим и

примитивен, а длина кодированного блока равна 2m – 1. Порождающая и проверочная матрицы для кодов Хэмминга генерируются функцией hammgen.

*Коды Рида—Соломона*

Коды Рида—Соломона являются одним из подклассов циклических блочных кодов. Это единственные поддерживаемые пакетом Communications недвоичные коды. Для работы с кодами Рида—Соломона функции высокого уровня вызывают специализированные функции rsenco (кодирование) и rsdeco (декодирование). Кроме того, функции rsencode и rsdecode позволяют использовать при кодировании и декодировании экспоненциальный формат данных, а функции rsencof и rsdecof осуществляют кодирование и декодирование текстового файла. Функция rspoly генерирует порождающие полиномы для кодов Рида—Соломона.

## Лабораторная работа №7. Помехоустойчивое кодирование

- Цель: Изучение методов помехоустойчивого кодирования и сравнение их свойств.
- 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.

## Лабораторная работа №8. Модель телекоммуникационного канала

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 1747. Далее полученные символы модулируются методом BPSK .

Задача: 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. 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. Ниже приведена таблица перемежения и последовательность ПСП.

```
static const uint8_t INTERLEAVER[] = {
    0;  17;  34;  51;  68;  85; 104;
    121; 138; 155; 172; 189;
    16;  33;  50;  67;  84; 101; 120;
    137; 154; 171; 188;  13;  32;  49;
    66;  83; 100; 117; 136; 153;
    170; 187;  12;  29;  48;  65;  82;
    99; 116; 133; 152; 169; 186;  11;
    28;  45; 64;  81;  98; 115; 132;
    149; 168; 185;  10;  27;  44;  61;
    80;  97; 114; 131;
    148; 165; 184;   9;  26;  43;  60;
    77;  96; 113; 130; 147; 164; 181;   8;  25;
    42;  59;  76;  93; 112; 129; 146; 163; 180;
    5;  24;  41;  58;  75;  92; 109;
    128; 145; 162; 179;   4;  21;  40;  57;  74;
    91; 108; 125; 144; 161; 178;   3;
    20;  37;  56;  73;  90; 107; 124; 141; 160; 177;
    2;  19;  36;  53;  72;  89;
    106; 123; 140; 157; 176;   1;  18;  35;  52;
    69;  88; 105; 122; 139; 156; 173,
};

const int8 Beacons::syncBits[ BEACONS_SYNC_LEN ] = {
1,  1,  1,  1,  1,  1, -1, -1, -1, -1,  1,  1,  1,
1, -1,  1,  1,  1, -1, -1, -1, -1,  1, -1,  1,  1,
-1, -1,  1,  1, -1,  1, 1, -1,  1,  1,  1,  1, -1,
1, -1, -1, -1, -1,  1,  1,  1, -1, -1,  1,  1, -1,
-1, -1, -1,  1, -1, -1,  1, -1, -1, -1,  1, -1, 1,
-1,  1,  1,  1, -1,  1, -1,  1,  1,  1,  1, -1, -1,
1, -1, -1,  1, -1,  1,  1,  1, -1, -1,  1,  1,  1,
-1, -1, -1, -1, -1, -1,  1,  1,  1, -1,  1,  1,  1,
-1,  1, -1, -1,  1,  1,  1,  1, -1,  1, -1,  1, -1,
 -1,  1, -1,  1, -1, -1, -1, -1, -1, -1,  1,
-1,  1, -1,  1, -1,  1, -1,  1,  1,  1,
 1,  1, -1,  1, -1,  1,  1, -1,  1, -1, -1,
 -1, -1, -1,  1,  1, -1,  1,  1,  1, -1,  1,
1, -1,  1,  1, -1,  1, -1,  1,  1, -1, -1,
-1, -1, -1,  1, -1,  1,  1,  1, -1,  1,  1,  1,  1,
1, -1, -1, -1,  1,  1,  1,  1, -1, -1,  1,  1, -1,
1, -1, -1,  1,  1, -1,  1, -1,  1,  1,  1, -1, -1,
```

```
-1,  1,  1, -1,  1, -1, -1, -1,  1, -1,  1,  1,  1,  1,
1,  1,  1, -1,  1, -1, -1,  1, -1,  1,  1, -1,
-1, -1,  1, -1,  1, -1, -1,  1,  1, -1, -1, -1,  1,  1,
-1, -1, -1, -1, -1, -1, -1,  1,  1, -1, -1,  1,  1,
-1, -1,  1, -1, 1, -1,  1,  1, -1, -1,  1, -1, -1,  1,
1,  1,  1,  1,  1, -1,  1,  1, -1,  1, -1, -1,  1, -1,
-1,  1, -1, -1,  1,  1, -1,  1, 1,  1,  1,  1,  1, -1,
-1,  1, -1,  1,  1, -1,  1, -1,  1, -1, -1, -1, -1,  1, -1,
1, -1, -1, -1,  1, -1, -1,  1,  1,  1, -1, 1,  1, -1, -1,
1, -1,  1,  1,  1,  1, -1,  1,  1, -1, -1, -1, -1,  1,  1,
-1,  1, -1,  1, -1,  1, -1, -1,  1,  1,  1, -1, -1,  1, -1,
-1, -1, -1,  1,  1, -1, -1, -1,  1, -1, -1, -1, -1,  1,
-1, -1, -1, -1, -1, -1, -1, -1,  1, -1, -1, -1,  1, -1,
-1, -1,  1,  1, -1, -1,  1, -1, -1, -1,  1,  1,  1, -1,
1, -1,  1, -1,  1,  1, -1,  1,  1,  -1, -1, -1,  1,  1,
1, -1, -1, -1,  1, -1, -1,  1, -1,  1, -1,  1, -1, -1, -1,
1,  1, -1,  1,  1, -1, -1,  1,  1,  1,  1,  1,
-1, -1,  1,  1,  1,  1, -1, -1, -1,  1, -1,  1,  1, -1,  1,
1,  1, -1, -1,  1, -1,  1, -1, -1,  1, -1, -1, -1, -1, -1,  1, -1,
-1,  1,  1, -1, -1,  1,  1,  1, -1,  1, -1, -1,
-1,  1,  1,  1,  1,  1, -1,  1,  1,  1,  1, -1, -1, -1, -1, -1,  1,  1,  1 };
```