Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ» (ТУСУР)

Е. В. Рогожников
Э. М. Дмитриев
К. В. Диноченко

ФОРМИРОВАНИЕ И ОБРАБОТКА СИГНАЛОВ СИСТЕМ МОБИЛЬНОЙ СВЯЗИ И ИНТЕРНЕТА ВЕЩЕЙ

Методические указания для выполнения практических работ и самостоятельной работы

Томск
2024

УДК 621.376.9
ББК 32.884.1
Р 598

Рецензент:
Крюков Я.В., доцент кафедры телекоммуникаций и основ радиотехники ТУСУРа, кандидат технических наук

Р 598 Формирование и обработка сигналов систем мобильной связи и Интернета вещей: Методические указания для выполнения практических работ и самостоятельной работы / Е. В. Рогожников, Э. М. Дмитриев, К. В. Диноченко – Томск: Томск. гос. ун-т систем управления и радиоэлектроники, 2024. – 94 с.

Настоящее учебно-методическое пособие представляет собой ресурс, охватывающий методические указания для выполнения практических заданий и самостоятельной работы. Основная цель данного пособия состоит в развитии у студентов компетенций в области цифровой обработки сигналов, специфичной для систем мобильной связи и Интернета вещей, а также в приобретении практических навыков в реализации соответствующих алгоритмов. Студентам будет представлена информация о базовых принципах функционирования систем мобильной связи и Интернета вещей, а также о методах формирования и обработки сигналов, используемых в этих системах. Данное пособие предназначено для студентов технических специальностей, а также для преподавателей, ведущих курсы по данной тематике, которое может использоваться как в рамках учебного процесса в вузах, так и для самостоятельного изучения студентами.

Одобрено на заседании кафедры ТОР, протокол № 1 от 31 августа 2023 г.

УДК 621.376.9
ББК 32.884.1

Оглавление

ВВЕДЕНИЕ

Цель данного практикума заключается в приобретении студентами компетенций в области формирования и обработки сигналов в системах мобильной связи и Интернета вещей, а также в освоении навыков моделирования соответствующих алгоритмов. Практикум направлен на развитие практических навыков студентов в применении полученных теоретических знаний для решения задач, связанных с разработкой и оптимизацией сигнальных систем в контексте мобильной связи и Интернета вещей. Он предоставляет студентам возможность ознакомиться с основными принципами функционирования систем связи и обработки данных, а также научиться использовать специализированные инструменты, такие как MATLAB [1] и Octave [2], для моделирования алгоритмов, использующихся в современных системах связи.

Работа № 1

«Начало работы с Octave»

Цель работы: изучить основные функции и блоки Octave и составить тестовую программу.

Задачи лабораторной работы:

- Изучить основные функции и блоки Octave.
- Произвести формирование синусоидального сигнала в Octave.
- Произвести сложение и умножение гармонических сигналов.

1. Теоретический материал

Octave – высокоуровневый интерпретируемый язык программирования, предназначенный для решения задач вычислительной математики. После запуска Octave пользователь видит рабочую область Octave, как на рисунке 1.1.

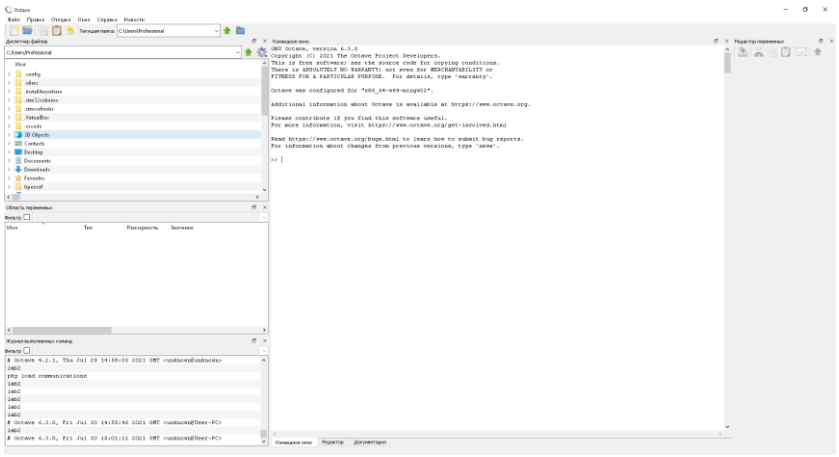


Рисунок 1.1 – Рабочая область Octave

Для создания нового проекта кликните на пустой лист в левом верхнем углу, как это показано на рисунке 1.2.

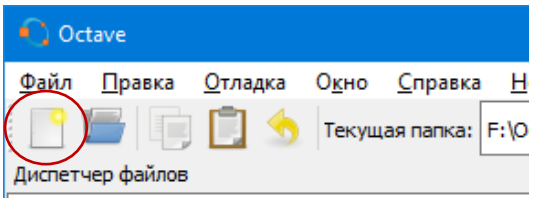


Рисунок 1.2 – Иконка «Создать сценарий»

После создания проекта вы можете начать писать свой код, по завершению скомпилируйте его, нажав на кнопку компиляции, изображенную на рисунке 1.3, или используя горячую клавишу F5.

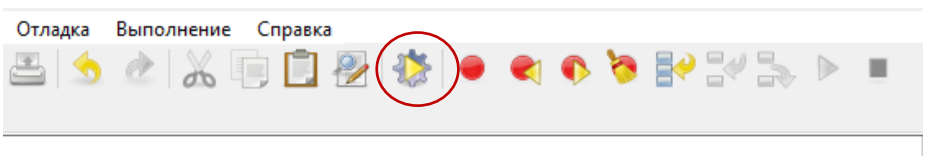


Рисунок 1.3 – Компиляция проекта

Для работы в среде Octave с функциями, относящимися к области телекоммуникаций, необходимо загрузить нужную библиотеку. Для этого перейдите в командное окно и введите «pkg load communications» , как это показано на рисунке 1.4.

```
The 'awgn' function belongs to the communications package from Octave
Forge which you have installed but not loaded.  To load the package, run
'pkg load communications' from the Octave prompt.

Please read <https://www.octave.org/missing.html> to learn how you can
contribute missing functionality.
error: called from
    lab2 at line 26 column 14
>> pkg load communications
```

Рисунок 1.4 – Загрузка библиотек

В работе будут реализованы следующие этапы:

Рисунок 1.5 – Этапы выполнения работы

2. Ход работы

1. Первые три строки программы как правило такие:

```
clc
clear all
close all
```

clc – очищает «Командное окно»;
clear all – удаляет все переменные из «Области переменных», очищает память;
close all – закрывает все открытые фигуры.

2. Постройте гармонический сигнал во временной и частотной области со следующими параметрами:

F0 = 10 кГц – несущая частота,
Fs = 250 кГц – частота дискретизации,
N = 100 – количество отсчетов.
Ниже представлена реализация в Octave.

```
clc
clear all
close all
F0 = ...;
```

```
Fs = ...;
N = ...;
t = (0:N-1)/Fs; % временные отсчеты
sig = sin (2*pi*F0*t);% гармонический сигнал

% Построение временной формы сигнала
figure
plot(t,sig);
grid on
xlabel('Time, s','fontsize',16);
ylabel('Amplitude','fontsize',16);
```

Результат выполнения кода показана на рисунке 2.1.

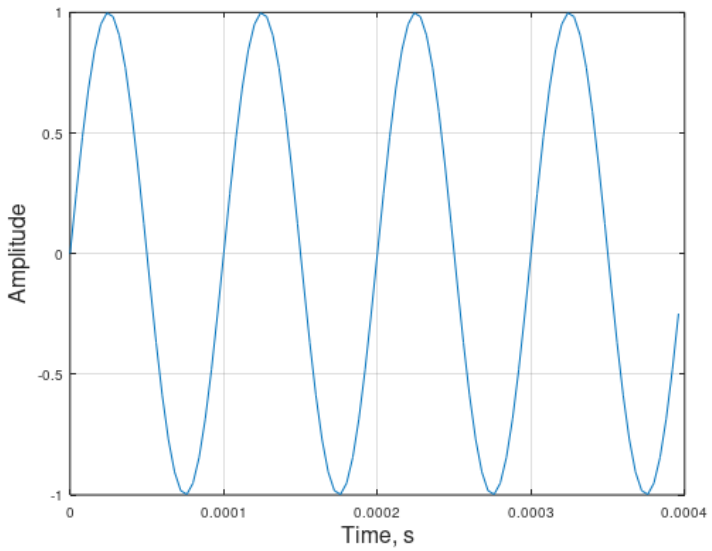


Рисунок 2.1 – Временная форма гармонического сигнала

3. Для того чтобы рассчитать спектр сгенерированного гармонического сигнала sig воспользуйтесь функцией fft(), которая выполняет операцию быстрого или дискретного преобразования Фурье в зависимости от длины массива.

```
spec = fft(...);
```

4. Выполните построение спектра сигнала. Для начала необходимо правильно задать ось частот, которая будет занимать диапазон от 0 Гц до Fs с шагом Fs/N.

```
f = (0:N-1)*(Fs/N);
```

5. Далее по аналогии с пунктом 2 выполните построение переменной spec на частотной оси f. Подпишите ось ординат как 'Amplitude' и ось абсцисс – 'Frequency, Hz'.

```
% Построение временной формы сигнала
figure
plot(..., ...);
grid on
xlabel('...','fontsize',16);
```

```
ylabel('...','fontsize',16);
```

В результате должны получить спектр гармонического сигнала как на рисунке 2.2.

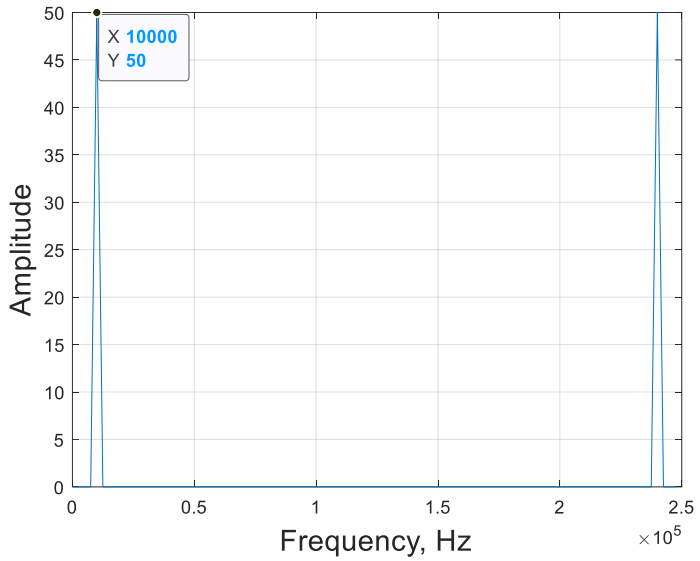


Рисунок 2.2 – Спектр гармонического сигнала

3. Самостоятельные задачи

Задача 1. Постройте спектр синусоидального сигнала, используя функцию fft(). Задайте ось частот аналогично временным отсчетам. Сформируйте еще один синусоидальный сигнал частотой 30 кГц. Используя цикл for, произведите перемножение и суммирование 2 синусоидальных сигналов.

Задача 2. Создайте две битовые последовательности (A и B) состоящие из 100 элементов ($N = 100$), используя функцию randi. Используя цикл for и функцию if, произведите суммирование по модулю 2. Таблица истинности XOR представлена в таблице 3.1.

Таблица 3.1 – Таблица истинности XOR

ВХОД		ВЫХОД
0	0	0
0	1	1
1	0	1
1	1	0

4. Контрольные вопросы к работе

1. Для чего может использоваться среда разработки Octave?
2. Как вывести рисунок на экран?
3. Как рассчитать спектр сигнала?
4. Как задать псевдослучайную битовую последовательность?

5. Требования к оформлению отчета

Содержание отчета:
1. Введение.
2. Описание работы.
3. Рисунки:
   - Гармонический сигнал.

- Спектр гармонического сигнала.
- Временная форма произведения гармонических сигналов.
- Спектр произведения гармонических сигналов.
- Временная форма суммы гармонических сигналов.
- Спектр суммы гармонических сигналов.

4. Выводы.
5. Полный листинг программы.

Работа № 2

«Модуляция/демодуляция сигналов. Расчет вероятности битовой ошибки»

Цель работы: произвести цифровую модуляцию и демодуляцию сигнала. Смоделировать передачу сигнала через канал с аддитивным белым гауссовским шумом. Построить зависимость вероятности битовой ошибки (Bit Error Ratio, BER) от отношения сигнал-шум (Signal-to-Noise Ratio, SNR).

Задачи лабораторной работы:

- Произвести модуляцию и демодуляцию сигнала в среде Octave.
- Имитировать передачу сигнала через канал с аддитивным белым гауссовским шумом.
- Рассчитать значения BER.
- Построить созвездия сигнала и график зависимости BER от SNR.

1. Теоретический материал

Модуляция – процесс изменения одного или нескольких параметров модулируемого несущего сигнала при помощи модулирующего информационного сигнала.

Виды модуляции: амплитудная и фазовая.

В данной лабораторной работе будет рассмотрена квадратурная фазовая модуляция (QPSK).

QPSK модуляция строится на основе кодирования двух бит передаваемой информации одним символом модуляции (рисунок 1.1). Символ модуляции графически отображается точкой созвездия, а математически – комплексным числом I+iQ.

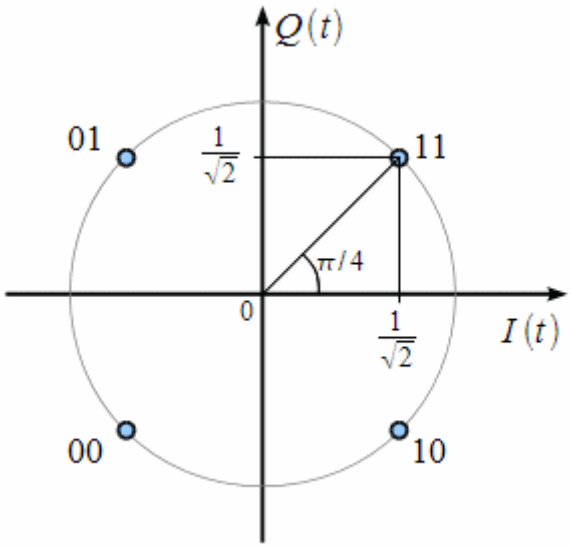


Рисунок 1.1 – Векторная диаграмма QPSK сигнала (Сигнальное созвездие)

То есть при модуляции сигнала каждому возможному значение дибита ставится в соответствие определенный символ модуляции, определяющий амплитуду и фазу несущего колебания. В таблице 1.1 представлены комплексные значения для квадратурной амплитудной модуляции.

Таблица 1.1 – Пример QPSK модуляции

Биты (до модуляции)	I и Q компоненты (после модуляции)
11	1+1i

Окончание таблицы 1.1

01	-1+1i
10	1-1i
00	-1-1i

Для демодуляции сигнальное созвездие делится на четверти. В зависимости от того, к какой области относится принятый символ модуляции, определяется значение битовой последовательности. Например, на рисунке 1.2 все точки созвездия, находящиеся в закрашенной области, вероятнее всего при передаче имели значение 1+1i, что соответствует битовому значению 11 до модуляции.

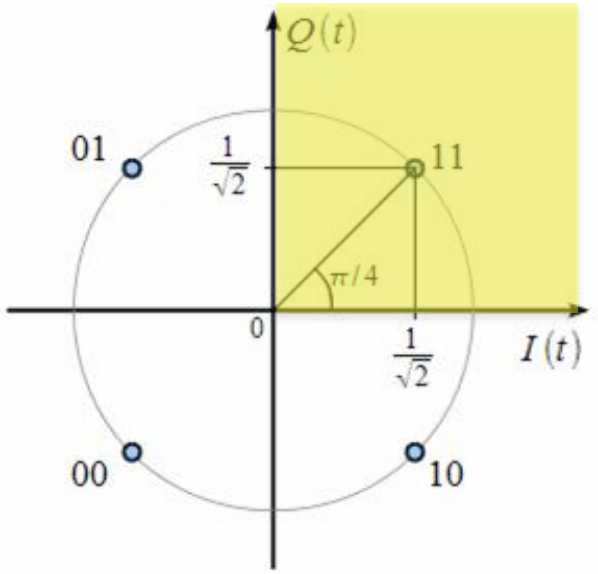


Рисунок 1.2 – Демодуляция

В системах связи возможно возникновение ошибок из-за шума, помех, искажений или битовой рассинхронизации. Вероятность битовой ошибки, BER (Bit Error Rate) – это количество битовых ошибок, деленное на общее число переданных битов в течение периода времени. Коэффициент ошибок является безразмерной величиной и часто выражается в процентах.

В работе будут реализованы этапы, представленные на рисунке 1.3.

Рисунок 1.3 – Этапы выполнения работы

2. Ход выполнения работы

1. Создайте битовую последовательность из 5000 символов, используя функцию randi().

```
clc
clear all
close all
N = ...;
bits = ...;
```

2. Произведите QPSK модуляцию, используя цикл for и условие if. Ниже приведено одно условие из четырёх, остальные допишите, опираясь на таблицу 1.1.

```
k=1;
for i = 1:2:N
  if bits(i) == 1 && bits (i+1) == 1
    mod_data (k) = 1+1i;
  end
  ...
  k=k+1;
end
```

Постройте созвездие модулированного сигнала, используя функцию scatterplot(). Для использования этой функции необходимо загрузить ее, написав «pkg load communications» в командное окно, и выполнить. Если автомасштабирование расположило точки на краях отображаемой области или за ней, задайте оси самостоятельно при помощи функций ylim([min max]), xlim([min max]). Получение созвездие должно соответствовать рисунку 2.1.

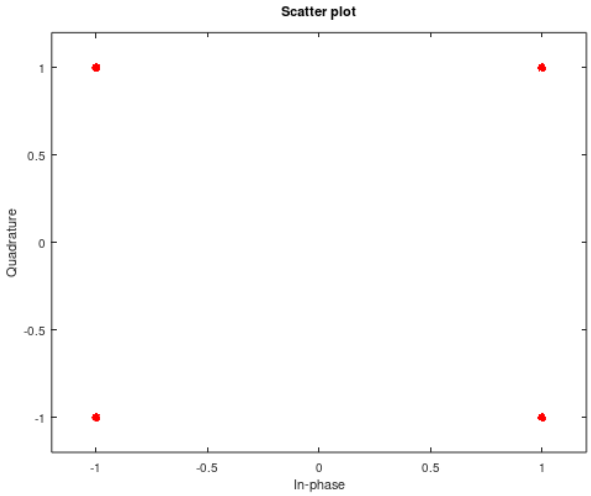


Рисунок 2.1 – Созвездие QPSK сигнала

3. Используя функцию awgn(), добавьте шум к модулированному сигналу. Задайте значение SNR равным 25.

```
SNR=...;
data_noise = awgn(mod_data,SNR,'measured');
```

Постройте созвездие сигнала с шумом используя функцию plot().Сделайте точки толще и измените их цвет на красный добавив «”r.”». Созвездие QPSK сигнала под воздействием шума представлено на рисунке 2.2.

```
plot(data_noise, "r.");
```

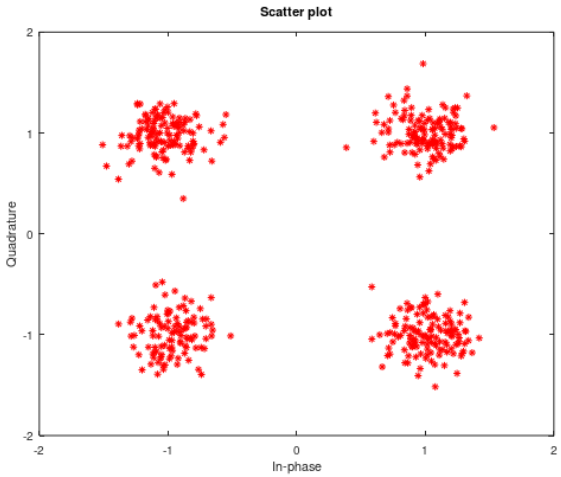


Рисунок 2.2 – Созвездие QPSK сигнала под воздействием шума

Отобразите сигнальные созвездия для этих значений в отчете.

4. Используя условный оператор if и цикл for, осуществите демодуляцию. Ниже приведены только одно условие из четырёх. Добавьте еще три условия, необходимых для демодуляции.

```
k=1;
for i = 1:length(mod_data_noise)
   if real(mod_data_noise (i))  >0 && imag(mod_data_noise (i)) >0
     bits_demod (k) = 1; bits_demod (k+1) = 1;
  end
  ...
  k=k+2;
end
```

5. Для того чтобы посчитать BER необходимо сравнить массивы bits и bits_demod. Битовое сравнение выполняет функция biterr. Функция возвращает количество несовпадений, то есть в данном случае ошибок.

```
BER_1 = biterr(...,...);
```

6. Чтобы построить зависимость BER от SNR, необходимо увеличить количество передаваемых бит N, указанное в начале кода, со значения N=5000 на N=10000.

```
N = ...;
```

Создайте в начале кода два пустых безразмерных массива, они будут использоваться для записи значений BER при разных значениях SNR. Пустой безразмерный массив задаётся посредством присвоения двух квадратных скобок [].

```
BER_B = [];
SNR_S = ...;
```

В предыдущих пунктах было указано фиксированное значение SNR = 25, теперь для того, чтобы построить зависимость BER от SNR, значение SNR необходимо изменять в диапазоне от –10 дБ до 25 дБ с шагом 1 дБ. Для этого измените код, создав цикл, в котором в качестве итерационной переменной будет использоваться переменная SNR.

```
for SNR = -10:1:25
```

В пункте 5 был рассчитан BER для одного значения SNR. Создайте переменную BER в которую будут записываться значения вероятности для текущего SNR. Используйте ранее созданные два накопителя, в которые будут записываться все значения BER и SNR посредством их конкатенации. После пропишите окончание цикла.

```
BER = BER_1/N;
BER_B = [BER_B BER];
SNR_S = [SNR_S SNR];
end;
```

Постройте график зависимости BER от SNR используя команды figure и semiology. «”-ok”» – делает график чёрно-белым и меняет маркер точек. Добавьте подписи к осям и дайте название графику. В результате изображение должно получиться, как на рисунке 2.3.

```
figure
semilogy(SNR_S, BER_B,'-ok')
grid;
title('OFDM Bit Error Rate .VS. Signal To Noise Ratio');
ylabel('BER');
xlabel('SNR [dB]');
```

Чтобы наглядно увидеть, как SNR влияет на созвездие, напишите перед циклом for команду figure. А после построения графика с шумом добавьте следующий код.

grid on – позволяет строить несколько графиков в одном окне, ylim – задает лимит по оси Oy, pause – делает паузу между построениями.

```
grid on
ylim([-2,2]);
pause(0.1)
```

В результате вы получите график зависимости BER от SNR.

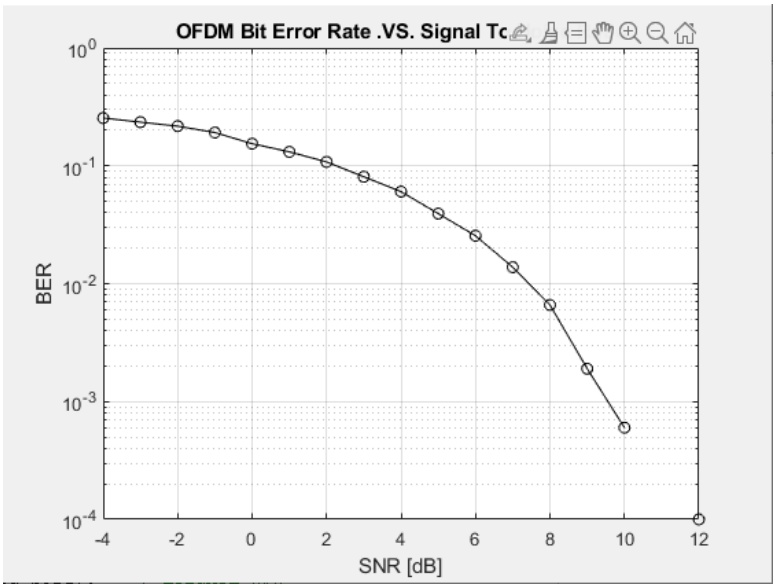


Рисунок 2.3 – Зависимость вероятности битовой ошибки от отношения сигнал/шум

3. Контрольные вопросы к работе:

1. Что такое QPSK модуляция?
2. Что такое BER, SNR?

4. Требования к оформлению отчета:

Содержание отчета:

1. Введение;
2. Блок схема алгоритма, который вы применили в работе;
3. Параметры OFDM символа;
4. Рисунки:
   - Созвездие QPSK сигнала при разном отношении сигнал шум;
   - График зависимости BER от SNR;
5. Выводы;
6. Листинг программы.

Работа № 3

«Формирование и обработка сигнала UNB»

Цель работы: сформировать и обработать сигнал с использованием сверхузкополосной модуляцией (UNB).

Задачи лабораторной работы:

- Составить программу для формирования и обработки сигнала UNB в среде Octave.
- Произвести дифференциальное кодирование/декодирование сигнала.
- Произвести модуляцию и демодуляцию сигнала.
- Имитировать передачу сигнала через канал с аддитивным белым гауссовским шумом.
- Рассчитать контрольную сумму и произвести расчет ошибок.
- Построить спектр одного символа суммарного сигнала, спектр суммарного сигнала, построить вектор данных одного пользователя.

1. Теоретическая часть

В данной работе требуется сформировать сигналы для трех пользователей, полоса канала каждого пользователя равна 100 Гц, каждый канал будет находиться на своей промежуточной частоте. Сначала необходимо вычислить контрольную сумму (CRC). Контрольная сумма – это способ проверки целостности принятой информации на стороне приёмника при передаче по каналам связи, его суть более подробнее описана в практической части работы. Далее в каждый канал будет добавлена битовая преамбула – это информация, которая известная на приемной стороне, необходимая для определения расположения начала пакета данных. После чего нужно модулировать пользовательский сигнал, далее он будет перенесен на промежуточную частоту, это можно сделать умножением сигнала на гармонический сигнал промежуточной частоты. У каждого пользователя своя промежуточная частота. После сигналы всех пользователей суммируются и уже из этого суммарного сигнала, прошедшего через канал, необходимо выделить сигналы каждого пользователя, демодулировать их и убедиться в отсутствие ошибок. Чтобы узнать, присутствуют ли ошибки в сигнале, мы будем использовать алгоритм расчёта контрольной суммы (CRC). Схема формирования сигналов изображена на рисунке 1.1.

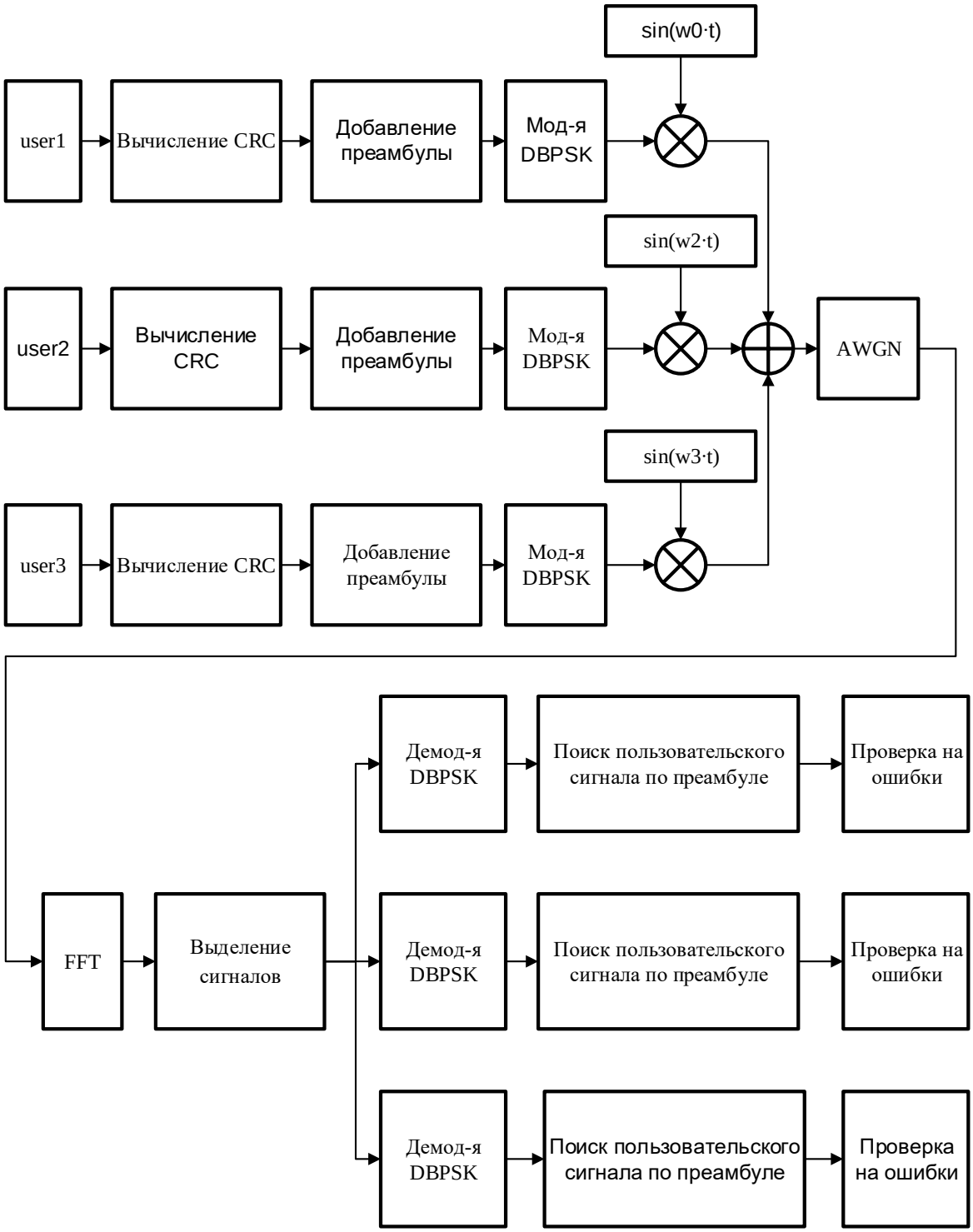


Рисунок 1.1 – Функциональная схема формирования и обработки сигналов
user1, user2, user3 – сигналы первого, второго и третьего пользователя;
AWGN – добавление аддитивного белого Гауссовского шума;
FFT – быстрое преобразование Фурье

В качестве модуляции используется BPSK с дифференциальным кодированием (DBPSK). Данная модуляция содержит кодер, который применяет операцию суммирования по модулю два к текущему и предыдущему биту информации, то есть кодируется изменение бита, после чего биты модулируются с использованием BPSK. Такой подход позволяет исключить необходимость в определении начальной фазы принимаемого сигнала и строгой временной синхронизации. Более подробно принцип работы дифференциального кодера показан на рисунке 1.2.

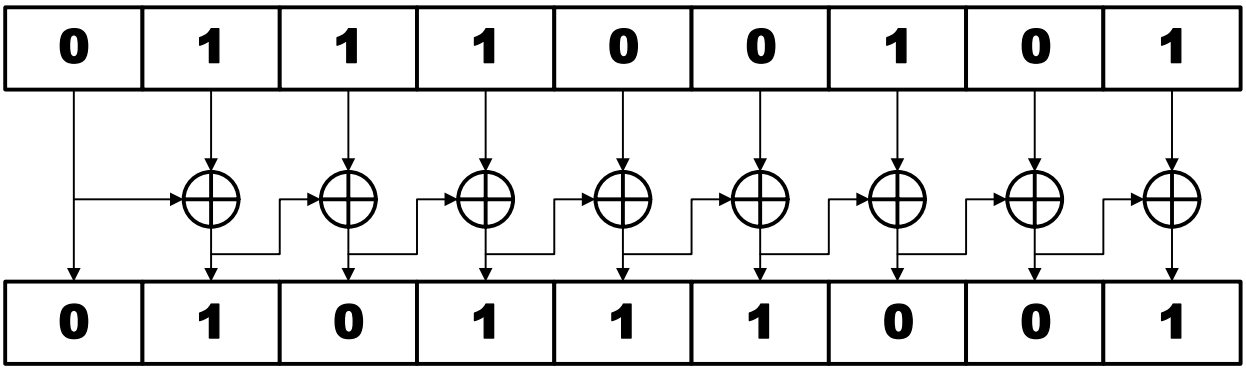


Рисунок 1.2 – Дифференциальное кодирование

Декодирование происходит аналогичным образом, только суммируются предыдущий и следующий биты. Пример декодера дан на рисунке 3.

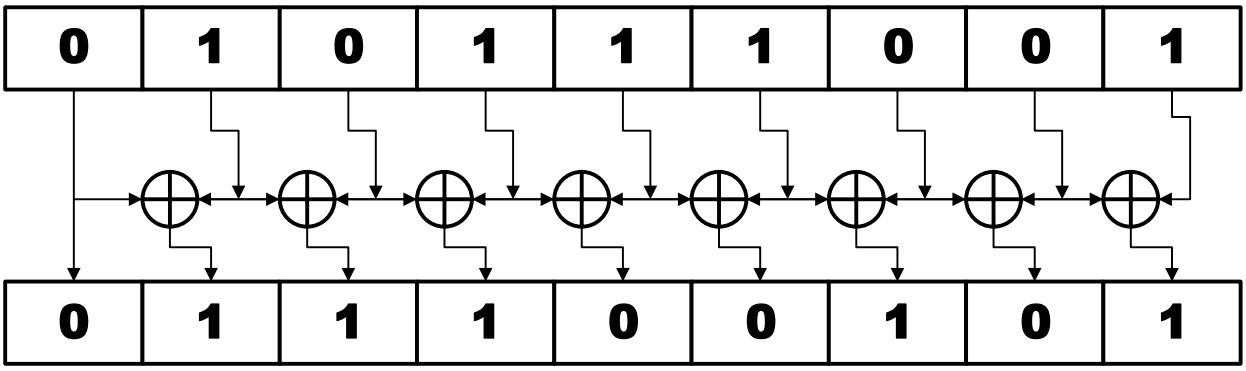


Рисунок 1.3 – Дифференциальное декодирование

Демодуляция осуществляется путем применения преобразования Фурье к каждому модулированному символу, знак фазы указывает на значение символа, если фаза положительна символ равен минус единице, если отрицательная – плюс единице.

2. Ход работы

1. Создайте новый проект в Octave. Запишите следующие команды для очистки значений моделирования при каждом повторном перезапуске кода программы:

```
clc
close all
clear all
```

Задайте основные параметры, используя таблицу 2.1.

Таблица 2.1 – Параметры модели

Параметры	Значение/формула
Отношение сигнал шум, SNR	50
Число бит, N	128
Длина преамбулы, p	16
Частота дискретизации, f_S	8 кГц
Коэффициент передискретизации, L	50
Промежуточная частота для каждого пользователя, f_0, f_2, f_3	800, 1600, 2400 Гц
Длина контрольной суммы, CRC_Type	16
Длительность сигнала, T_S, рассчитывается по формуле	$(L \cdot (N + p + crc)) / f_s$

Опираясь на таблицу 2.1 запишите основные параметры следующим образом:

```
SNR = ...;
N = ...;
p = ...;
fs = ...;
L = ...;
CRC_Type = ...;
...
```

Создайте битовую последовательность для каждого пользователя, содержащую 128 бит используя функцию randi.

```
data = randi([0, 1], 1, N);
data2 = ...;
...
```

2. На следующем этапе следует рассчитать контрольную сумму (CRC) для каждого пользователя, а именно CRC-16. Для выполнения расчета CRC нужно разделить пользовательское сообщение на полином, полученный остаток от деления и будет контрольной суммой, который добавляется в конец информационного сообщения. Для этой цели используйте готовую функцию CRC_CODEC, которая дана в прикрепленных файлах к заданию. Нужно переместить файл CRC_CODEC.m в папку с вашим проектом и далее обратиться к ней в программе.

```
crc = CRC_CODEC (data,16,1);
crc2 = ...;
...
```

3. Далее создайте преамбулу для каждого пользователя и добавьте ее перед основной информацией.

```
preamb = randi([0, 1], 1, p);
preamb2 = ...;
...
pdata = [preamb, crc];
pdata2 = ...;
...
```

4. Следующим этапом необходимо осуществить процедуру дифференциального кодирования. Структурная схема кодера показана на рисунке 2.1.

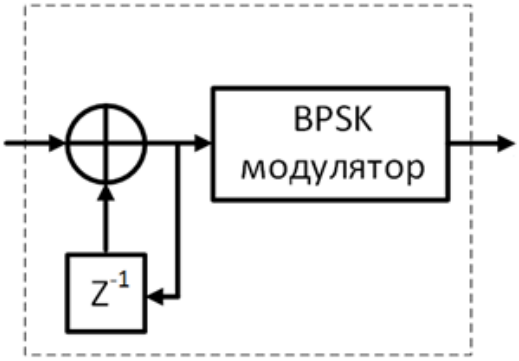


Рисунок 2.1 – Структурная схема дифференциального кодера

Суммирование производится по модулю два, что соответствует логическому XOR (исключающее ИЛИ), Z^{-1} означает задержку на один бит информации. Из рисунков 2 и 4 видно, что первый бит не изменяется и проходит дальше. Второй бит суммируется по модулю два с первым, третий суммируется по модулю с результатом суммы второго и первого и так далее. В Octave данный кодер можно реализовать, используя цикл for и функцию xor. Следовательно, мы должны создать в цикле вектор строку, каждому элементу которого будет присваиваться результат суммирования по модулю текущего элемента исходной битовой последовательности и предыдущего элемента самого вектора строки. Учтите, что первый элемент последовательности остается неизменным, значит суммирование и цикл должны начинаться со второго элемента.

```
mod_data(1) = pdata(1);
mod_data2(1) = ...;
…
for i = 2:length(pdata)
mod_data(i) = xor(pdata(i), mod_data(i-1));
mod_data2(i) = xor(...);
...
end
```

5. Далее примените двоичную фазовую манипуляцию (BPSK) к вашей закодированной последовательности данных. При двоичной фазовой манипуляции квадратурная компонента равна нулю, а синфазная принимает значения -1 и +1. Диаграмма созвездия BPSK модуляции представлена на рисунке 2.2. Для выполнения данной операции в Octave используйте функцию pskmod().

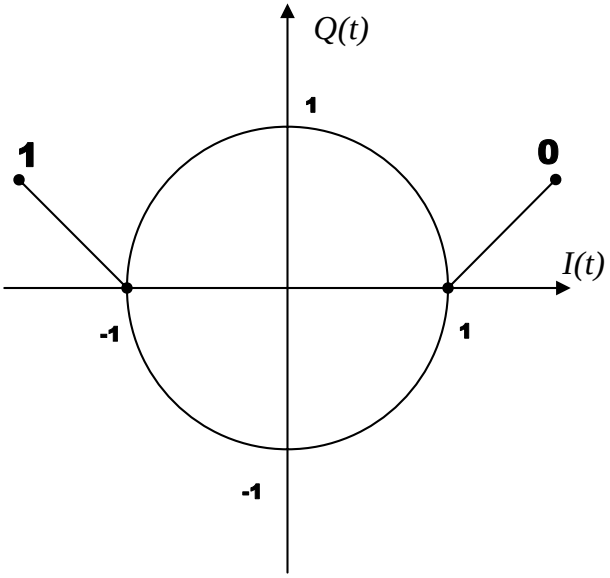


Рисунок 2.2 – Диаграмма созвездия BPSK сигнала

```
mod_data = pskmod(mod_data, order);
mod_data2 = ...
...
```

order – порядок модуляции, для BPSK, он равен 2.

6. Теперь нужно передискретизировать данные. Коэффициент передискретизации L указывает число повторений каждого символа.

```
k=1;
for i = 1:L:L*(N+p+ CRC_Type)
data_L (i:i+L-1) = mod_data(k);
data_L2 (i:i+L-1) = ...
...
k=k+1;
end;
```

В этом фрагменте вы создаете новый массив, в котором L = 50 раз подряд повторяется каждый бит. Таким образом, к примеру, вместо массива 100 бит вы получите массив 5000 бит. Постройте получившийся вектор данных одного пользователя из трех, на ваш выбор, используя функцию plot()**.**

```
figure
plot(...,'linewidth',3);
```

Подпись – 'linewidth',3 установит толщину линий на графике равной 3 пт.

Вы должны увидеть изображение, как на рисунке 2.3.

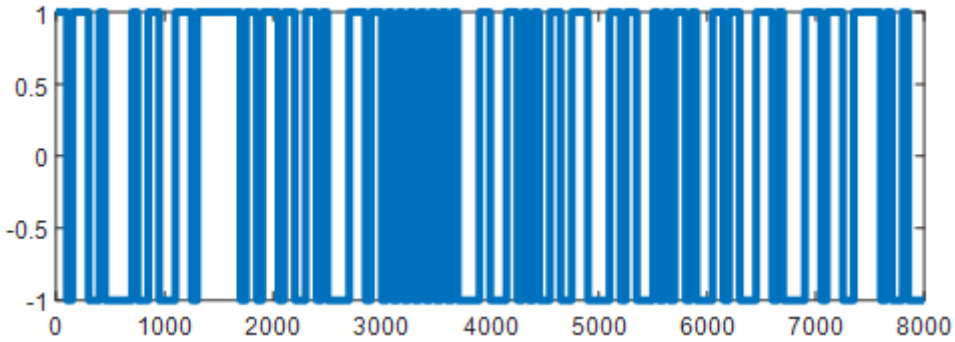


Рисунок 2.3 – BPSK сигнал в Octave

Рассчитайте длительность сигнала

```
Ts = L*N/fs;
```

7. Затем, создайте гармонические сигналы промежуточной частоты для каждого пользователя и перенесите BPSK последовательность на эту частоту, у каждого пользователя своя частота:

```
sig = sin(2*pi*f0*t);
sig2 = ...
...
sig_carrier = sig.*data_L;
sig_carrier2 = ...
...
```

f0 – промежуточная частота, t – рассчитанный вектор временных отсчетов.

Так как у нас после передискретизации 8000 отсчетов сигнала, нужно создать вектор строку в котором будут записаны 8000 временных отсчетов для этого сигнала, величиной от 1 до 8000. Далее делим вектор временных отсчетов на частоту дискретизации. Так как частота дискретизации равна fs = 8000 кГц, то длительность всего сигнала равна 1 секунде, а период дискретизации (шаг по времени) равен 1/fs.

```
t = (0:L*length(pdata)-1)/fs;
```

Далее произведите поэлементное умножение сгенерированного гармонического колебания на передискретизированную BPSK последовательность. Таким образом вы осуществите перенос BPSK сигнала с нулевой частоты на промежуточную. Постройте график сигнала на промежуточной частоте, он будет содержать большое число отсчетов, как на рисунке 2.4, используйте инструмент «zoom in» для того, чтобы более подробно рассмотреть график. Для этого после выбора инструмента выделите область, которую хотите посмотреть в более близком масштабе.

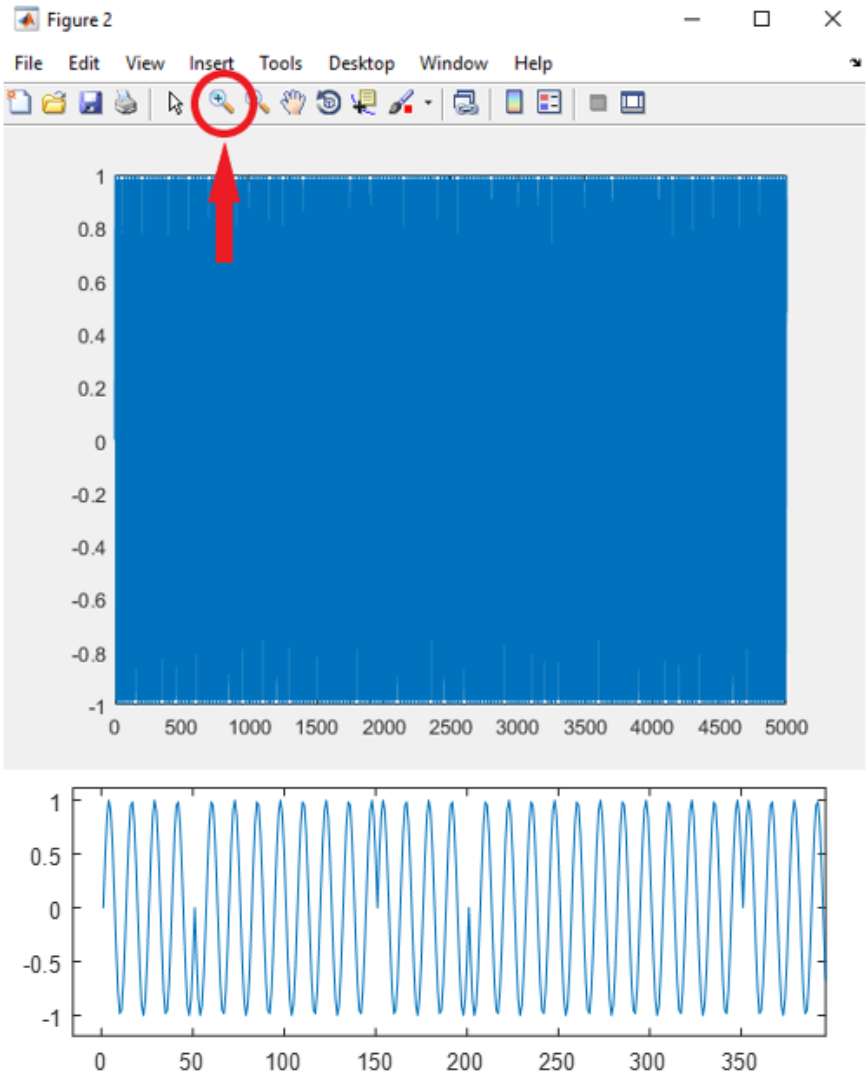


Рисунок 2.4 – BPSK сигнал на промежуточной частоте

Из рисунка 2.4 видно, что у сгенерированного гармонического сигнала происходит изменение фазы в соответствие с BPSK сигналом, где он равен «1» сигнал не изменяется, где равен «-1» у сигнала происходит изменение фазы.

8. Далее сложите все три сигнала, это будет эквивалентно тому, как сигналы складываются в приемной антенне.

```
sumpass= ...;
```

Добавьте в суммарный сигнал белый шум используя функцию awgn.

```
sig_carrier_n = awgn(sumpass, SNR, 'measured');
```

sumpass – суммарный сигнал;

SNR – отношение сигнал-шум;

measured – данная подпись обозначает, что шум будет добавляться относительно текущей мощности сигнала.

Перенесите зашумленный суммарный сигнал в область частот применив к нему функцию fft().

```
sp_cf = fft(sig_carrier_n);
```

Создайте ось частот для этого сигнала как:

```
freq_val = 1:1:fs;
```

Таким способом вы создадите вектор строку частот с шагом по частоте в 1 Гц, содержащий 8000 элементов. Постройте спектр, он должен соответствовать рисунку 1.8.

```
figure
plot(freq_val,abs(sp_cf));
```

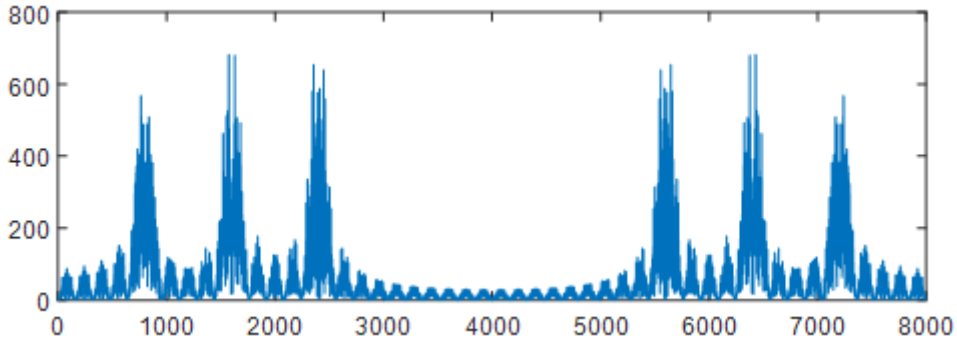


Рисунок 2.5 – Спектр суммарного сигнала

В правой части графика располагается зеркальное отражение сигнала, на него не стоит обращать внимания. В левой части четко видны сигналы пользователей на своих промежуточных частотах.

9. Для демодуляции сигнала следует рассматривать отдельно каждый символ и определить его значение. После передискретизации каждый символ представлен L=50 отсчетами, всего символов N=128 и преамбула также содержит p=32. Для удобства вычислений преобразуем вектор строку нашего сигнала в матрицу размером [N x L] = [160 x 50]. В таком случае каждая строка будет соответствовать одному символу. Выполним это помощью цикла for.

```
k = 1;
for i = 1:L:L*(N+p+ CRC_Type)
sp_chip (k, :) = fft(sig_carrier_n(i:i+L-1));
k=k+1;
end
```

В цикле осуществляется перенос сигнала в частотную область, далее каждые 50 отсчетов сигнала записываются в строки матрицы sp_chip размером [160 x 50]. Вспомните в самом начале процесс передискретизации, где каждый бит повторялся 50 раз.

При наличие фазового сдвига, BPSK созвездие может разворачиваться на некоторый угол (рисунок 2.6) и в зависимости от его разворота, может потребоваться дополнительная обработка сигнала. Например, если сигнал приобретет случайный фазовый сдвиг, при котором созвездие развернется на 45 градусов, то для успешной демодуляции потребуется рассматривать и синфазную, и квадратурную составляющую сигнала. Так как в данной работе случайный фазовый сдвиг не вводился в канале, то рассматривать сигнал мы будем на одной квадратуре.

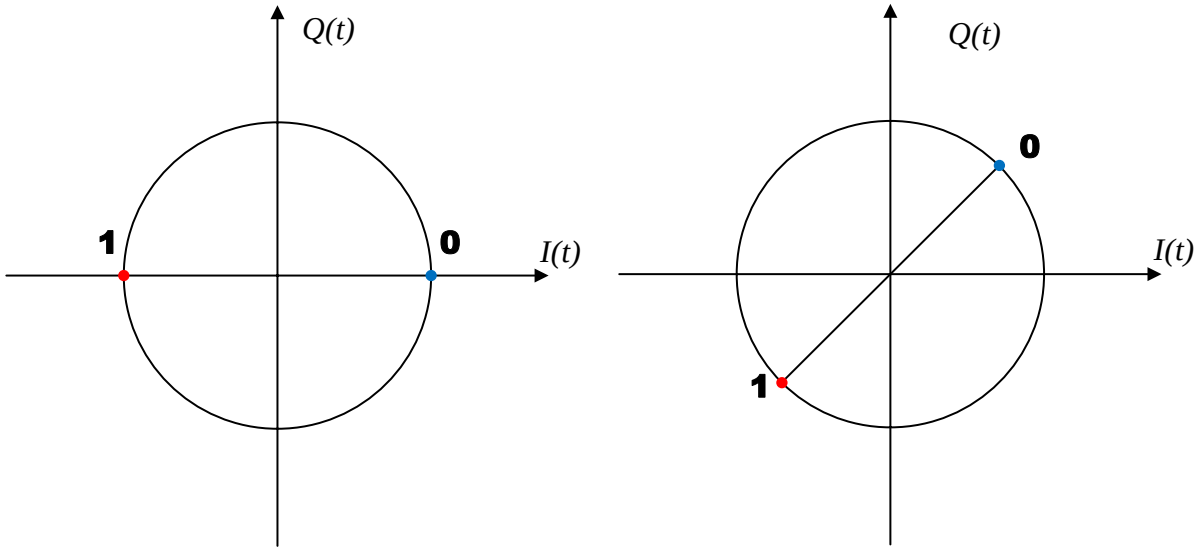


Рисунок 2.6 – Поворот созвездия

После выполнения вышеописанных операций постройте график квадратурной составляющей, первой строки матрицы sp_chip следующим образом:

```
figure
plot (imag(sp_chip(1,1:end/2)));
```

Сравните график с рисунком 2.7.

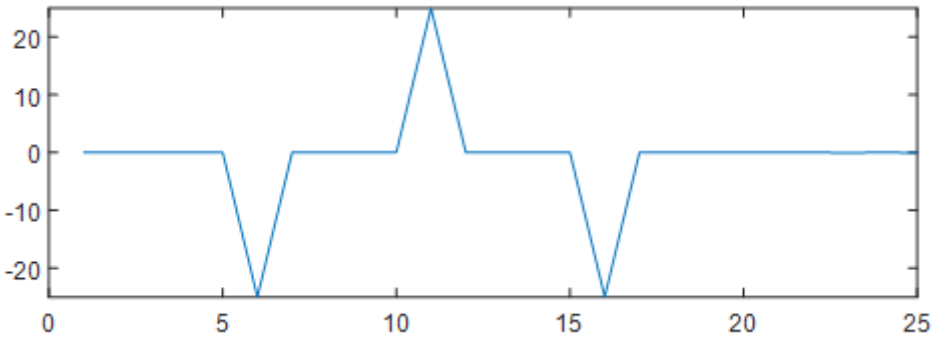


Рисунок 2.7 – Спектр одного символа суммарного сигнала

На рисунке выше изображен спектр одного символа суммарного сигнала, где четко видено три символа каждого пользователя на своей несущей частоте. Фаза отдельно взятого пользовательского сигнала может быть положительной или отрицательной. Если фаза положительна, символ, переносимый сигналом равен минус единице, если отрицательная – плюс единице. Это исходит из того, что после передискретизации и переноса на промежуточную частоту, каждый модулированный символ будет представлять собой сигнал определенной фазы. Следовательно, каждому символу будет соответствовать своя фаза у сигнала.

Для начала необходимо понять, какая спектральная составляющая, какому пользователю соответствует. Рисунок выше построен не на оси частот, а на оси отсчетов, значит, несущую частоту каждого пользователя нужно перевести в номер отсчета соответствующей ей. Во фрагменте программы, представленном ниже, мы определяем какой частоте равен один отсчет, после чего делим несущую частоту каждого пользователя на это значение и получаем номер отсчета (peak, peak2, peak3) для этого пользователя

```
peak = (f0/(fs/L))+1;
peak2 = ...;
peak3 = ...
```

Используйте оператор if, цикл for и функцию max(), чтобы определить значение каждого символа исходя из его фазы. Пример фрагмента программы показан ниже.

```
for i = 1: (N+p+ CRC_Type)
  if (imag(sp_chip(i,peak)))>0
    halfdemod_data(i) = -1;
  else
    halfdemod_data(i) = 1;
end

  if (imag(sp_chip(i,peak2)))>0
    halfdemod_data2(i) = -1;
  else
    halfdemod_data2(i) = 1;
  end

  if (imag(sp_chip(i,peak3)))>0
   ...
  end

end
```

Для начала обратимся к ранее сформированной матрице, где каждая строка обозначает передаваемый символ.

Обращаясь к отсчетам, соответствующим промежуточным частотам пользователей, мы смотрим значение фазы сигнала в конкретном отсчете (на конкретной промежуточной частоте). Так как спектр представлен комплексными числами, выделим необходимую нам квадратурную составляющую функцией imag(). Оператор if здесь означает, что если значение квадратурной составляющей спектра символа положительно, то этот символ равен «-1».

```
if (imag((sp_chip(1,peak/peak2/peak3))))>0
halfdemod_data(1) = -1;
```

Иначе он равен «1».

```
else
halfdemod_data(1) = 1;
end;
```

10. Демодулируйте полученные последовательности символов функцией pskdemod().

```
demod_data = pskdemod(halfdemod_data, order);
demod_data2 = ...
...
```

После этого декодируйте данные. Структурная схема декодера изображена на рисунке 2.8.

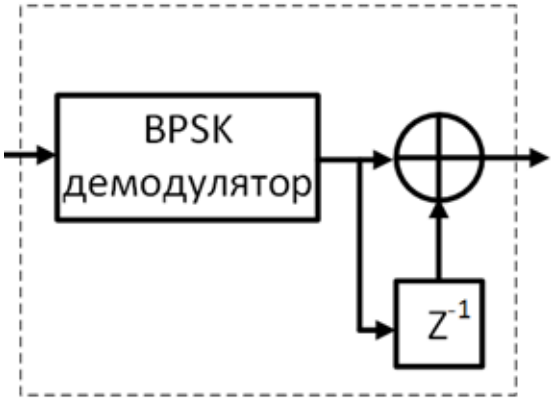


Рисунок 2.8 – Структурная схема дифференциального декодера

Он работает аналогично кодеру (см. пункт 1), отличие в следующем – вы должны суммировать по модулю текущий и предыдущий отсчет демодулированной последовательности.

```
decode_data(1) = demod_data(1);
decode_data2(1) = ...
decode_data3(1) = ...
for i = 2:length(demod_data)
decode_data(i) = xor(demod_data(i),demod_data(i-1));
decode_data2(i) = ...
...
end
```

11. Сейчас нужно определить по преамбуле, где находится пользовательская информация, для этого используем функцию xcorr(). С помощью этой функции мы сравним преамбулу, с пользовательским сообщением и определим ее положение, после чего отсечем ее от основного сообщения и проверим его целостность.

```
T1 = xcorr(decode_data, preamb);
[A, B] = max(T1);
start = B-N- crc;
T2 = xcorr(decode_data2, preamb2);
[A2, B2] = ...;
start2 = ...;
...
```

Переменная start указывает на номер отсчета в массиве decode_data, с которого начинается пользовательская информация.

Последним шагом с помощью ранее рассчитанной контрольной суммы определим наличие ошибок в пользовательском сообщении. Для этого используйте фрагмент кода ниже.

```
[rx_data, errors] = CRC_CODEC(decode_data(start+1:end), CRC_Type,0);
[rx_data2, errors2] = ...;
...
```

Значения всех элементов массивов errors, errors2 … должны быть равны нулю, если это не так, значит в сообщении присутствуют ошибки, перепроверьте свою программу.

3. Требования к оформлению отчета

Для отчетности предоставьте pdf-файл, содержащий следующие материалы по выполненной работе:

- Полный листинг программы;
- Полученные в ходе выполнения работы графики: BPSK сигнал на выходе кодера, BPSK модулированное колебание, Спектр BPSK модулированного колебания, Результат расчета преобразования Фурье на приемной стороне.

Требование к листингу:

Листинг программы представляет собой полученный в результате выполнения хода работы код, предоставленный в текстовом формате .pdf.

Требования к оформлению графиков:

При предоставлении рисунков их нужно оформлять следующим образом:

– указать на рисунке название графика;

– указать на рисунке названия осей;

– указать на рисунке единицы измерения для каждой оси (если присутствуют).

Типовые ошибки при выполнении:

- Использование в данном коде обычных матричных арифметических операторов («+», «-», «/», «*») вместо поэлементных («.+», «.-», «./», «.*») там, где это требуется.
- Построение комплексного спектра сигнала без использования оператора вычисления модуля `abs()`.

Работа № 4

«Формирование сигналов первичной и вторичной синхронизации NB IoT»

Цель работы: сформировать сигналы первичной и вторичной синхронизации, а также опорные сигналы.

Задачи лабораторной работы:

1. Сформировать узкополосный сигнал первичной синхронизации NPSS.
2. Сформировать узкополосный сигнал вторичной синхронизации NSSS.
3. Сформировать узкополосный опорный сигнал NRS.

1. Теоретический материал

1.1 NPSS (Первичный узкополосный сигнал синхронизации)

Первичный узкополосный сигнал синхронизации (NPSS) используется в системах NB-IoT для синхронизации как по времени, так и по частоте. Из-за сложности вычислений в приемнике, необходимых для обнаружения NPSS, все ячейки в сети NB-IoT используют один и тот же NPSS в отличии от LTE, где используется три варианта PSS. В результате приемнику необходимо рассчитывать корреляцию только по одной последовательности, что значительно упрощает процедуру временной синхронизации.

Как и в LTE, в NB-IoT для первичной синхронизации используется последовательность Задова-Чу, но с небольшим отличием, которое заключается в дополнительной переменной S(l):

$$d_l(n) = S(l) \cdot e^{-j\frac{\pi u m(n+1)}{11}}, n = 0,1.....,10 \quad (1.1)$$

где $S(l) = [0001111-1-1111-11]$;

$u = 5;\ u = 5;$

$N = 11;\ N = 11;$

$m = 0 : N - 1.$

Разница между PSS и NPSS заключается в распределении ресурсов. В LTE можно использовать 6 ресурсных блоков (то есть 72 ресурсных элементов) в одном символе OFDM для размещения PSS, но в LTE-NB у вас есть только один ресурсный блок (то есть 12 ресурсных элементов) в символе OFDM. Таким образом, вы можете разместить только небольшое количество данных в одном символе OFDM. При таком небольшом количестве опорных данных будет трудно добиться достаточного уровня корреляции, необходимого для верной синхронизации на дальних дистанциях. Поэтому используется несколько символов OFDM (11 символов OFDM) для размещения синхросигнала. По сути, для передачи NPSS используется целый ресурсный блок. На рисунке 1.1 представлен расположение NPSS на ресурсной сетке.

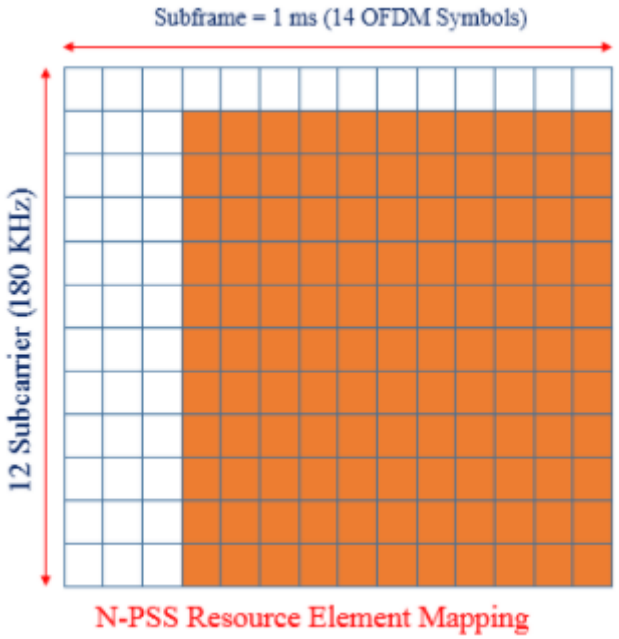


Рисунок 1.1 – Расположение NPSS на ресурсной сетке

1.2 NSSS (вторичный узкополосный сигнал синхронизации)

NSSS необходим для определения идентификатора соты и получения дополнительной информации о структуре кадра. NB-IoT поддерживает 504 уникальных идентификатора физических ячеек (PCID), указанных в NSSS. NSSS имеет интервал повторения 80 мс, в течение которого передаются четыре последовательности NSSS.

В NSSS используется последовательность Задова-Чу и бинарная последовательность, указанная в стандарте 3gpp, которая представлена в таблице 1. NSSS зависит от выбранного идентификатора соты. Рассчитывается NSSS по формуле (1.2).

$$d(n) = b_q(m) e^{-j2\pi\theta_f n} \cdot e^{-j\frac{\pi u n'(n'+1)}{131}}, \qquad (1.2)$$

где $n = 0,1,...,131$;

$n' = n \bmod 131$;

$m = n \bmod 128$;

$u = NCellId \bmod 126 + 3$;

$q = \dfrac{NCellId}{126}$;

$\theta_f = \dfrac{33}{132}(\dfrac{n_f}{2}) \bmod 4$.

Таблица 1.1 – Двоичная последовательность q

q	$b_q(0),.....,b_q(127)$
0	1 1

Окончание таблицы 1.1

1	1 -1 -1 1 -1 1 1 -1 -1 1 1 -1 1 -1 -1 1 -1 1 1 -1 1 1 -1 -1 1 1 -1 -1 1 -1 1 1 -1 -1 1 -1 -1 1 -1 1 1 -1 -1 1 -1 1 1 -1 -1 1 -1 1 1 -1 1 -1 -1 1 -1 1 1 -1 1 -1 -1 1 1 -1 -1 1 -1 1 1 -1 1 -1 -1 1 -1 1 1 -1 -1 1 1 -1 1 -1 -1 1 1 -1 -1 1 -1 1 1 -1 1 -1 -1 1 1 -1 -1 1 -1 1 1 -1
2	1 -1 -1 1 -1 1 1 -1 -1 1 1 -1 1 -1 -1 1 -1 1 1 -1 1 1 -1 -1 1 1 -1 -1 1 -1 1 1 -1 -1 1 -1 -1 1 -1 1 1 -1 -1 1 -1 1 1 -1 -1 1 -1 1 1 -1 1 -1 -1 1 -1 1 1 -1 1 -1 -1 1 1 -1 -1 1 -1 1 1 -1 1 -1 -1 1 -1 1 1 -1 -1 1 1 -1 1 -1 -1 1 1 -1 -1 1 -1 1 1 -1 1 -1 -1 1 1 -1 -1 1 -1 1 1 -1
3	1 -1 -1 1 -1 1 1 -1 -1 1 1 -1 1 -1 -1 1 -1 1 1 -1 1 1 -1 -1 1 1 -1 -1 1 -1 1 1 -1 -1 1 -1 -1 1 -1 1 1 -1 -1 1 -1 1 1 -1 -1 1 -1 1 1 -1 1 -1 -1 1 -1 1 1 -1 1 -1 -1 1 1 -1 -1 1 -1 1 1 -1 1 -1 -1 1 -1 1 1 -1 -1 1 1 -1 1 -1 -1 1 1 -1 -1 1 -1 1 1 -1 1 -1 -1 1 1 -1 -1 1 -1 1 1 -1

Как показано на рисунке 1.2, NSSS занимает 11 OFDM символов с конца подкадра. Передается NSSS в 9 подкадре каждого четного кадра.

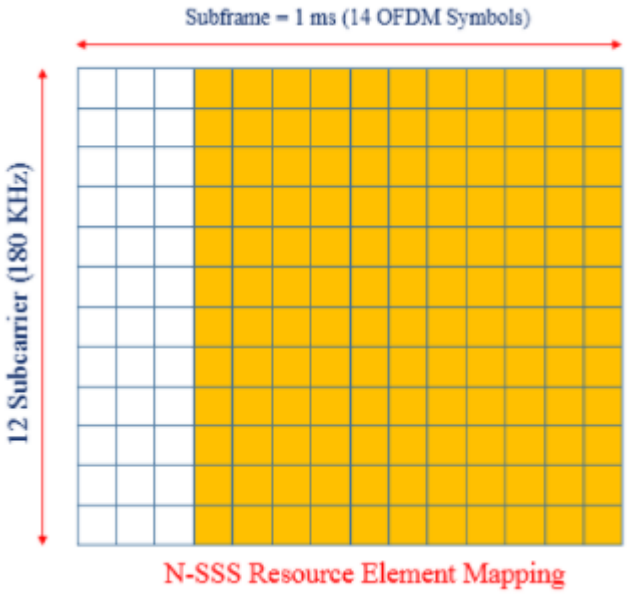


Рисунок 1.2 – Расположение NSSS на ресурсной сетке

1.3 NRS (узкополосный опорный сигнал)

NRS – это опорный сигнал для конкретной соты нисходящего канала, необходимый для эквалайзирования. Последовательность NRS генерируется также, как и в LTE, но расположение ресурсных элементов в ресурсной сетке изменено (рисунок 1.3).

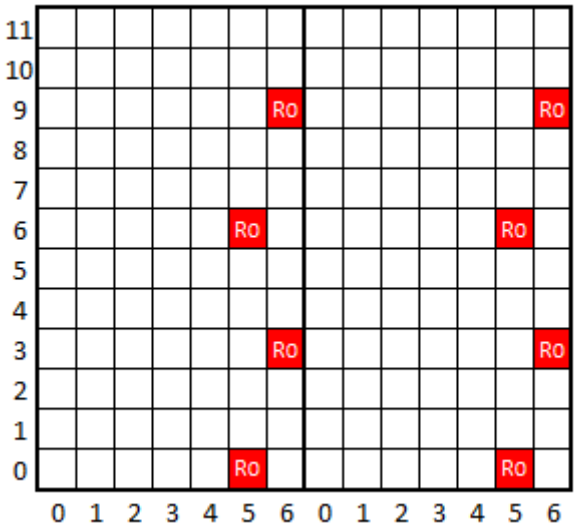


Рисунок 1.3 – Расположение NRS на ресурсной сетке

Для формирования NRS последовательности используется псевдослучайная последовательность c_{init} :

$$
\begin{gathered}
r_{l,n_s}(m)=\frac{1}{\sqrt{2}}(1-2\cdot c(2m))+j\frac{1}{\sqrt{2}}(1-2\cdot c(2m+1)),\\
m=0,1,...,2N_{RB}^{\max,DL}-1,\\
c_{init}=2^{10}\cdot(7\cdot(n_S^{'}+1)+l+1)\cdot(2\cdot N_{ID}^{cell}+1)+2\cdot N_{ID}^{cell}+N_{CP},\\
n_s^{'}=\begin{cases}10\lfloor n_s/10\rfloor+n_s \bmod 2\\ n_s\end{cases},\\
N_{CP}=\begin{cases}1\\0\end{cases}.
\end{gathered}
\qquad (1.3)
$$

Псевдослучайная последовательность, используемая для NB-IOT, как и для LTE, является типом последовательности Голда, формулы которой представлены ниже.

$$
\begin{gathered}
x_1(0)=1, x_1(n)=0, n=1,2,...,30,\\
x_1(n+31)=(x_1(n+3)+x_1(n)) \bmod 2,\\
x_2(n+31)=(x_2(n+3)+x_2(n+2)+x_2(n+1)+x_2(n)) \bmod 2,\\
c(n)=(x_1(n+N_C)+x_2(n+N_C)) \bmod 2,\\
N_C=1600.
\end{gathered}
\qquad (1.4)
$$

Этапы выполнения работы представлены на рисунке 1.4.

Рисунок 1.4 – Этапы выполнения работы

2. Ход работы

1. Для формирования последовательности Задова-Чу задайте основные параметры.

```
u = 5;
N = 11;
m = 0:N-1
```

Задайте начальное значение *S(l)* указанное в теоретическом материале.

```
S = [];
```

Используя формулу (1.1), рассчитайте NPSS.

```
NPSS = …;
```

Используя функцию imagesc(), постройте график NPSS. Результат показан на рисунке 2.1.

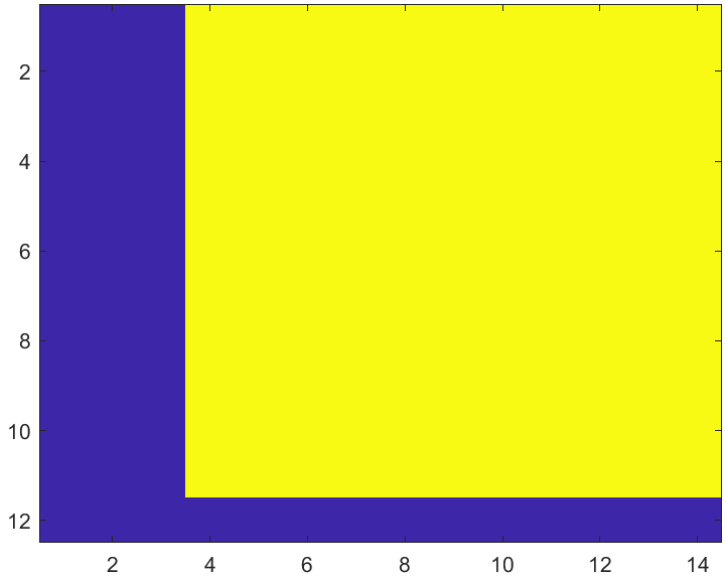


Рисунок 2.1 – Сформированный NPSS

Сохраните созданную последовательность NPSS в папку с файлом, это пригодится в последующих работах. Обратите внимание на то, что в пути к файлу не должно быть русских символов:

```
save('NPSS.mat','NPSS');
```

2. Используя формулу (1.2) и описание к ней, напишите основные параметры NSSS

```
NNCellID = 3;
NFrame = 0
NSubframe = 9;
ns = double(NFrame*10 + NSubframe) ;
u = mod(NNCellID,126) + 3;
```

```
q = fix(NNCellID/126);
N = 131;
nf = fix(ns/10);
theta = 33/132 * mod(nf/2,4);
n = 0:N;
nd = mod(n,N)
seq = exp(-1i*pi*(2*theta*n + u*nd.*(nd+1)/N));
```

Используйте значения из таблицы 1.1, чтобы записать все значения *q*:

```
Seq = [
    […]
    […]
    ];
```

Посчитайте значение NSSS по формуле (1.2):

```
Seq = …;
```

Расположите NSSS в подкадре согласно рисунку 1.2:

```
NSSS = [zeros(12,3),reshape(seq,12,11)];
```

Используя функцию imagesc(), постройте график NSSS. На рисунке 2.2 показан результат построения.

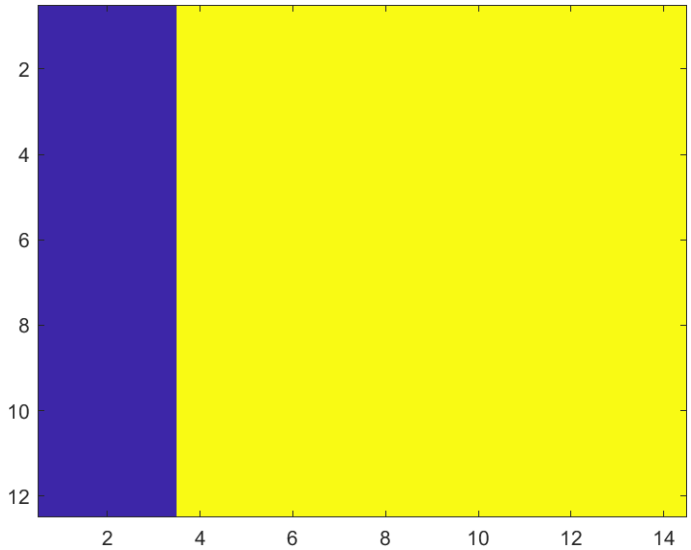


Рисунок 2.2 – Сформированный NSSS

Сохраните созданную последовательность NSSS в папку с файлом.

```
save('……','……');
```

3. Задайте основные параметры необходимые для формирования NRS, согласно формуле (1.3):

```
Nc=1600;
N_rb_dl_max = 4;
len = 2*N_rb_dl_max;
seq_len = 2*(len-1)+1+1; % длина последовательности
ncp = 1; % циклический префикс (1 - расширенный, 0 - обыкновенный)
N_s = 1;  % номер слота
L = 5;
```

Далее по формуле (1.4) необходимо сформировать инициализирующую последовательность:

```
c_init = 1024 * (……)*(……)+ 2*NNCellID + ncp;
```

Создайте два вектора для начальных условий. Запишите условие x1 по аналогии с x1:

```
x1 = zeros(1,1600+seq_len);
x2 = ……;
tmp = c_init;
```

Сгенерируйте два начальных условия для x1 и x1:

```
for(n=0:30)
      x2(30-n+1) = floor(tmp/(2^(30-n)));
      tmp = tmp - (floor(tmp/(2^(30-n)))*2^(30-n)); % Начальное условие многочлена x 2
   end

x1(0+1) = 1; % Начальное условие многочлена x1
```

Выполните генерацию m-последовательности для x1 и x2 согласно формуле (1.4):

```
for(n=0:1600+seq_len)
      x1(n+31+1) = mod(x1(n+3+1) + x1(n+1), 2);
      x2(n+31+1) = mod(x2(n+3+1) + x2(n+2+1) + x2(n+1+1) + x2(n+1), 2);
end
```

Создайте последовательность Голда согласно формуле (1.4):

```
for(n=0:seq_len-1)
      c(n+1) = mod(…) + x2(…);
end
```

Произведите QPSK модуляцию для полученной последовательности:

```
NRC = zeros(1,len);
   for(m=0:len-1)
      NRC(m+1) = (1/sqrt(2))*(1 - 2*c(2*m+1)) + j*(1/sqrt(2))*(1 - 2*c(2*m+1+1));
   end
```

Сохраните созданную последовательность NRC.

```
save('……','……');
```

Используя функцию subplot, постройте на одном графике x1, x1 и c(n). Также постройте созвездие модулированных данных.

```
subplot(3,1,1);
stem(x1);xlim([1 length(x1)]); ylabel('x1(n)');
subplot(3,1,2);
stem(x2);xlim([1 length(x2)]); ylabel('x2(n)');
subplot(3,1,3);
stem(c);xlim([1 length(c)]); ylabel('c(n)');
scatterplot (NRC);
grid = zeros(12,14);
```

Пример иллюстрации сформированной последовательности представлен на рисунке 2.3.

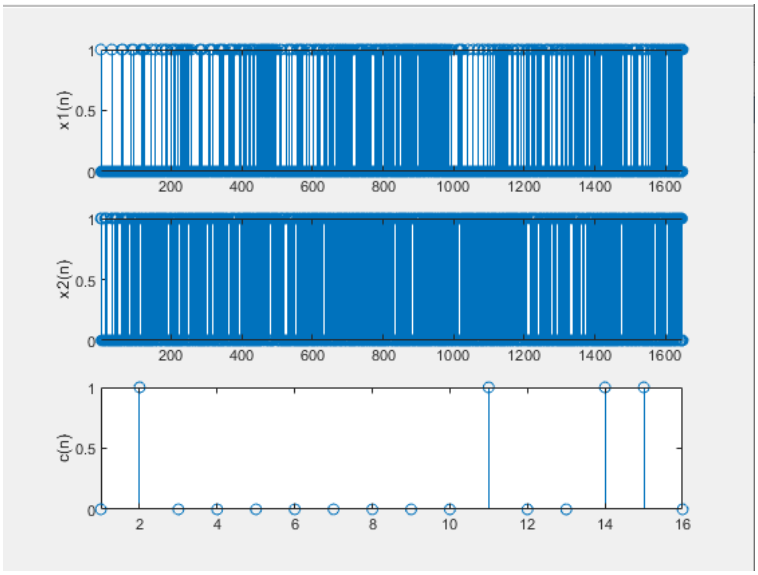


Рисунок 2.3 – Сформированная последовательность

3. Контрольные вопросы

1. Для чего используется первичный сигнал синхронизации?
2. Для чего используется вторичный сигнал синхронизации?
3. Для чего используется опорный сигнал?

Работа № 5

«Формирование кадра нисходящего канала NB-IoT»

Цель работы: расположить NPSS, NSSS, NRS сигналы в радиокадре согласно стандарту 3GPP TS 36.211.

Задачи лабораторной работы:

1. Сформировать узкополосный сигнал первичной синхронизации NPSS.
2. Сформировать узкополосный сигнал вторичной синхронизации NSSS.
3. Расположить сигналы синхронизации в спектральной области OFDM-символа.

1. Теоретический материал

NB-IoT имеет большие отличия от LTE на физическом уровне. Эти отличия заключается как в алгоритмах формирования опорных сигналов, так и в структуре кадра физического уровня.

Общими параметрами для NB-IoT и LTE являются:

- Длина подкадра – 1мс.
- Один кадр состоит из 10 подкадров.
- Количество поднесущих в ресурсном блоке – 12.

Но также есть отличия:

- Пропускная способность: NB-IoT – 180 кГц, LTE – от 1.4 МГц до 20 МГц.
- Количество ресурсных блоков в полосе пропускания: NB-IoT – 1, LTE – от 6 до 110 в зависимости от ширины полосы.
- В NB-IoT NPSS и NSSS расположены в разных подкадрах, тогда как PSS, SSS в LTE расположены в одном подкадре.
- В случае NB-IoT NPSS передается в каждом кадре, а NSSS только в четных, тогда как PSS и SSS в LTE передаются в каждом радиокадре.

Рисунок 1.1 показывает три режима работы NB-IoT: IN-Band, Guardband, Standalone.

Рисунок 1.1 – Режимы работы NB-IOT

В данной лабораторной работе мы будем реализовывать режим работы: Standalone

Рассмотрим детально карту кадра, изображенного на рисунке 1.2.

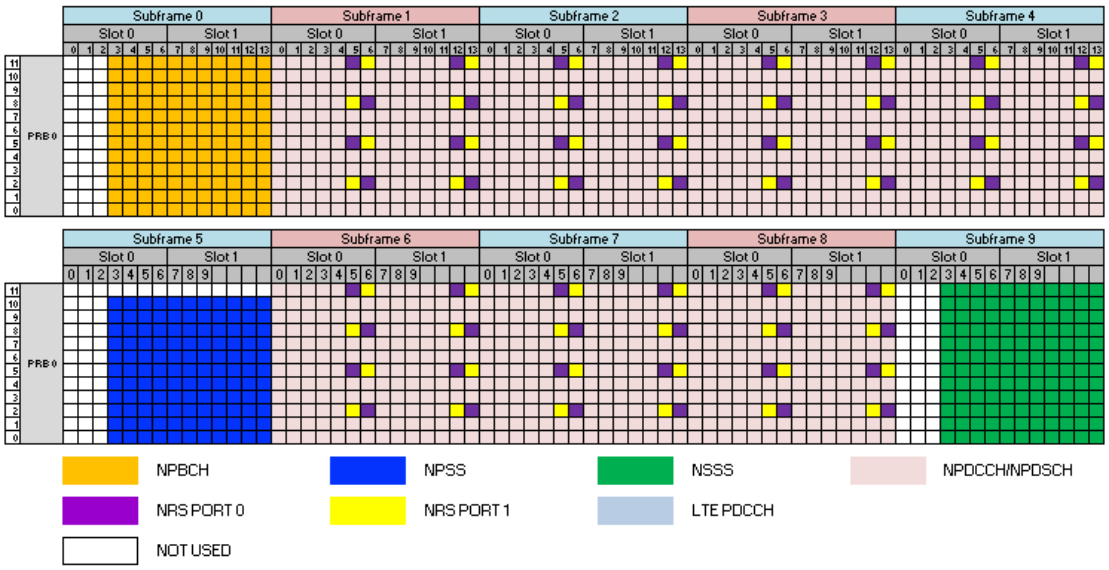


Рисунок 1.2 – Структура кадра Standalone

Основные сигналы кадра:

• NPBCH – занимает весь нулевой подкадр, за исключением первых трех символов в подкадре.

• NPSS – занимает весь пятый подкадр, за исключением первых трех символов в подкадре.

• NSSS – занимает весь девятый подкадр каждого четного кадра, за исключением первых трех символов в подкадре

• NRS – расположены в символах 5, 6, 12, 13 каждого подкадра кроме NPSS и NSSS

В работе будут реализованы этапы, представленные на рисунке 1.3:

Рисунок 1.3 – Этапы выполнения работы

2. Ход работы

1. Загрузите сформированные в предыдущей работе сигналы синхронизации.

```
clc
```

```
clear all
close all
load('NSSS.mat')
load('NRS.mat')
load('NPSS.mat')
```

Создайте пустой массив размером 12x14, на основе которого будет сформирован кадр.

```
sf = zeros(…);
```

Далее необходимо создать последовательность бит для заполнения кадра случайными данными.

```
N=10000;
bits = randi(……);
```

Произведите QPSK модуляцию, как делали в предыдущих работах. Предоставлено только 2 условия из 4х.

```
k=1;
for i=1:2:N
  if (bits(i) == 1 && bits(i+1) == 1) data(k)=1+1i;
  end
    if (bits(i) == 1 && bits(i+1) == 0) data(k)=1-1i;
    end
…….
k=k+1;
end
```

Зная из теоретического материала места расположения NRS, расположите значения сформированного NRS в кадре. Предоставлено только 4 условия из 8.

```
for a = 1:1:size(sf,1)
    for b =  1:1:size(sf,2)

        if a == 6 && b == 6
           sf(a,b) = NRC(:,1);
        end
           if a == 12 && b == 6
              sf(a,b) = NRC(:,2);
              end
              if a == 3 && b == 7
           sf(a,b) = NRC(:,3);
           end
           if a == 9 && b == 7
              sf(a,b) = NRC(:,4);
              end
end
end
```

Используя функцию imagesc, постройте график NRS, расположенных в подкадре, он должен быть как на рисунке 2.1.

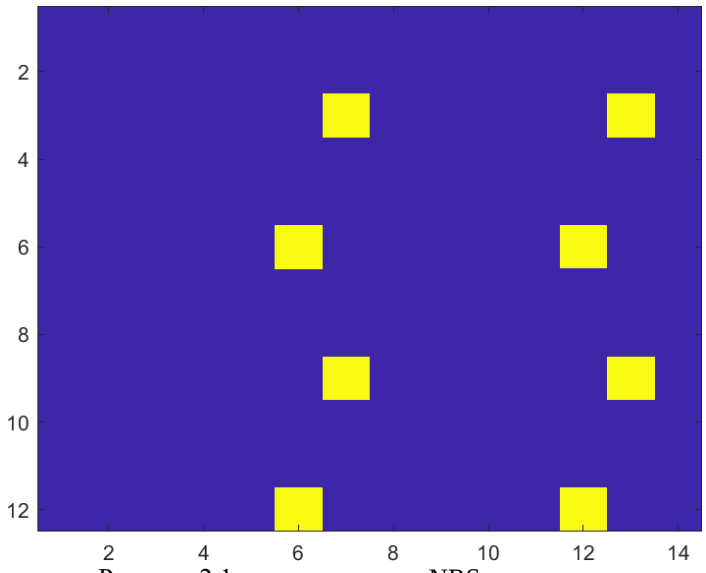


Рисунок 2.1 – расположение NRS в подкадре

Далее разместите модулированные ранее данные в свободных от NRS местах.

```
k=1
for a = 1:1:size(sf,1)
    for b =  1:1:size(sf,2)
       if sf(a,b) == ……
          sf(a,b) = ……
          k=k+1
       end
    end
end
```

Используя функцию imagesc() постройте график NRS подкадра с данными. Пример представлен на рисунке 2.2.

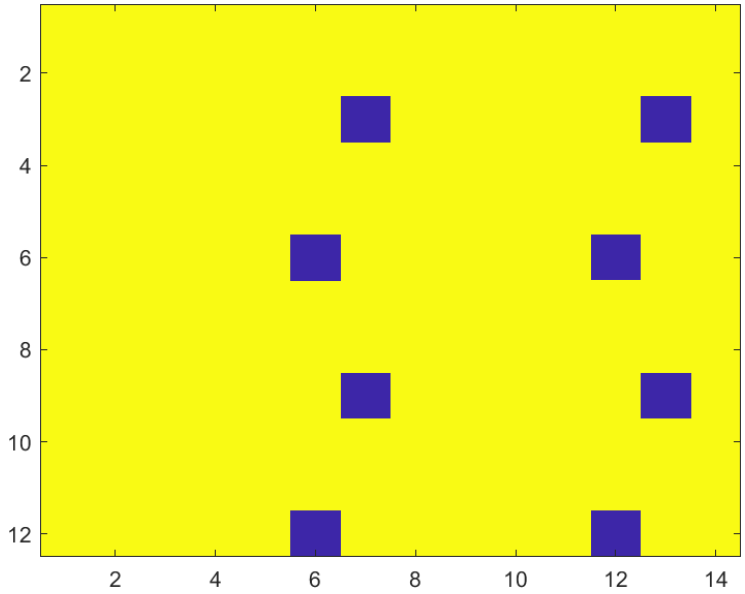


Рисунок 2.2 – Расположение NRS и данные в подкадре

2. Сформируйте один кадр, состоящий из 10 подкадров, разместив в нем NPSS на 5 месте, а NSSS на 9. Все остальные подкадры заполните сформированным ранее sf.

```
frame = [sf,……,NPSS,……,NSSSa];
```

Используя функцию imagesc(), постройте график кадра с данными. На рисунке 2.3 представлен сформированный кадр.

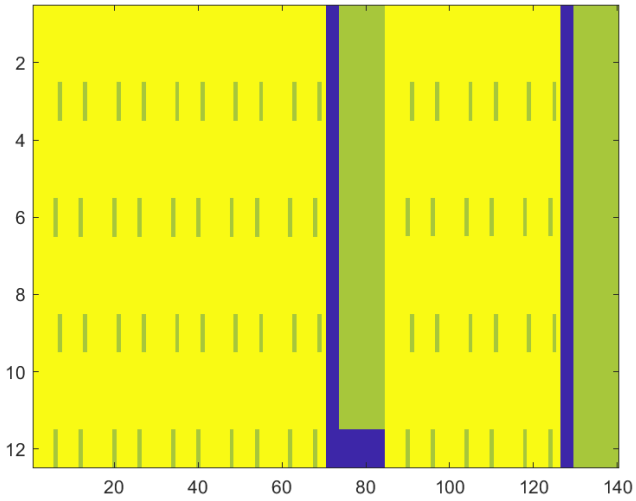


Рисунок 2.3 – Сформированный кадр

3. Так как размер Фурье для NB-IOT равен 128, необходимо расширить наш кадр до размера 128x140.

```
b = zeros(58,140);
frame = [……];
```

Далее произведем обратное преобразование Фурье для каждого столбца.

```
for k=1:……
```

```
        framefft(……) = (ifft(fftshift (frame(……))));
    end
```

4. Так как циклический префикс в NB-IOT может иметь два значения:
10 – расширенный.
9 – обычный.
Необходимо добавить их в кадр согласно заданной последовательности.

```
    CPlen = [10 9 9 9 9 9 9 10 9 9 9 9 9 9];
    cpLengths = double(CPlen);
```

Теперь напишите цикл добавления ЦП.

```
    frame=[];
     for i=1:1:140
          CP=[];
          cpLength = cpLengths(mod(i-1,length(cpLengths))+1);
          CP = [framefft(128-(cpLength+0)+1:end,i); framefft(:,i)];
          frame = [frame;CP];
     end
```

Используя функцию plot() постройте кадр во временной области. Пример того, что должно получиться в итоге представлен на рисунке 2.4.

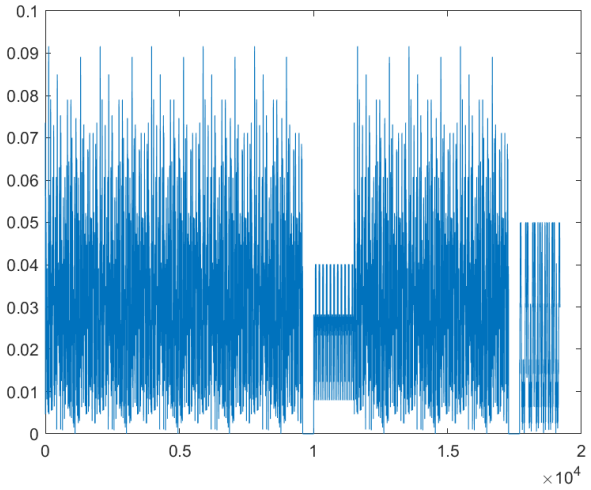


Рисунок 2.4 – Кадр во временной области

Сохраните сформированный кадр используя команды из предыдущей работы с именем «frame».

```
    save('……','……');
```

3. Контрольные вопросы

1. Структура кадра NB-IoT;
2. Отличие NB-IoT кадра от LTE;
3. Для чего делается ОБПФ;
4. Для чего нужен циклический префикс.

Работа № 6

«Кадровая синхронизация NB-IoT»

Цель работы: произвести первичную и вторичную синхронизацию, вычислить идентификатор физической ячейки.

Задачи лабораторной работы:

- Формирование узкополосного первичного сигнала синхронизации NPSS в приемнике
- Поиск узкополосного первичного сигнала синхронизации NPSS.
- Поиск узкополосного вторичного сигнала синхронизации NSSS.
- Вычисление значения сотового идентификатора.

1. Теоретический материал

Для поиска сигналов базовой станции в стандарте NB-IoT предусмотрена передача двух типов синхросигналов – первичного и вторичного. Эти сигналы предоставляют информацию, требуемую пользовательскому оборудованию для синхронизации нисходящей линии связи.

NPSS используется устройствами для синхронизации как по времени, так и по частоте. NPSS сигнал можно обнаружить даже при очень большом сдвиге частоты. В сети NB-IoT используют один и тот же NPSS, в результате устройству нужно искать только один NPSS.

После того, как была выполнена синхронизация по времени, мы обращаемся к NSSS для обнаружения идентификатора соты и получения дополнительной информации о структуре кадра. NB-IoT поддерживает 504 уникальных идентификатора физических ячеек, указанных в NSSS. Физическая идентификация соты заключается в предоставлении псевдоуникального значения для идентификации соты.

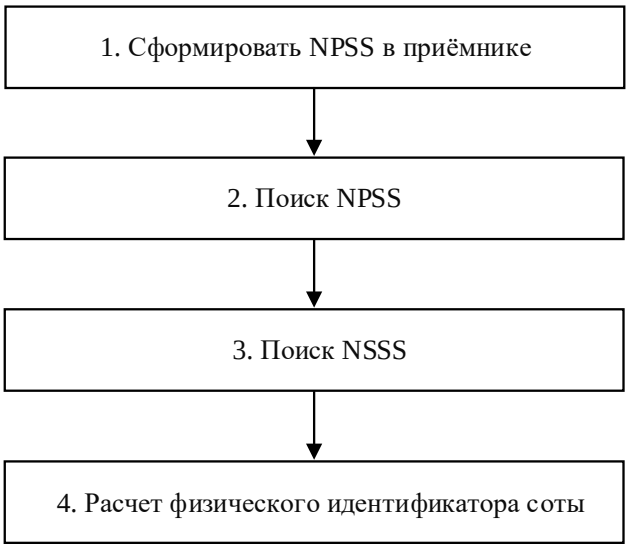


Рисунок 1.1 – Этапы выполнения работы

2. Ход выполнения работы

Данная работа является продолжением работы «Формирование сигналов первичной и вторичной синхронизации». Из предыдущей работы вам необходимо взять код по формированию сигналов синхронизации PSS и SSS, а также загрузить сформированный кадр.

1. Сформируйте в приемнике NPSS последовательность по аналогии с предыдущими работами, добавьте защитные интервалы, циклический префикс.

```
clc
clear all
close all
load('frame.mat')
nfft = 128;
CPlen = [……];
cpLengths = double(……);
u = …;
N = …;
m = ……;
S = [……];
NPSS = [……].'.*S;
b = zeros(……);
frame1 = [ ……];
for k=1:…
  frameifftNPSS(…) = (ifft(fftshift (…(…))));
 end
PSSseq=[];
for i=1:1:…
    CP=[];
    cpLength = cpLengths(…(…,length(……))+1);
    CP = [……(nfft -(……)+1:end,i); frameifftNPSS(……)];
    PSSseq = [……];
  end
```

2. Далее для нахождения начала NPSS рассчитываем взаимокорреляционную функцию между пилотным сигналом и принятым сигналом. Найдите максимум ВКФ, используя функцию max.

```
cor = xcorr(……)
[value ind_psss] = max(……);
```

Пример ВКФ приведен на рисунке 2.1.
Используем положение максимума ВКФ для определения начала символа с данными.

```
startPSSS = ((ind_psss-floor(length(cor)/2))+1);
```

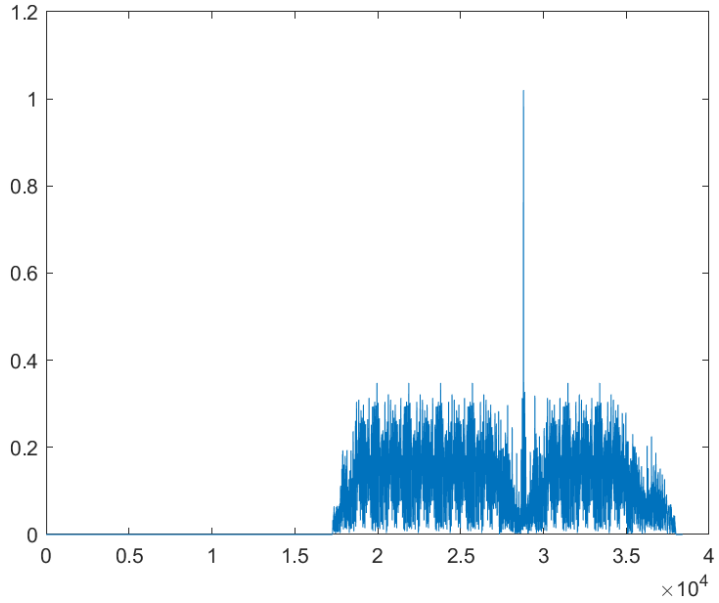


Рисунок 2.1 – ВКФ NPSS

3. Зная, что NPSS находится в 5 подкадре, а NSSS в 9, найдите начало NSSS сигнала, и вырежьте его.

```
startSSS = ……;

sss = frame(……);
```

Далее необходимо из полученного NSSS, удалить циклический префикс:

```
frame_rx = [];
 start = 1;
 for i = 1:14
 cpLength = cpLengths(mod(i-1,length(cpLengths))+1);
 start = start + cpLength;
 frame_rx(:,i)  = sss(start:start+ nfft -1);
 start = start+nfft;
 end
```

Переводите сигнал в частотную область и выполните разворот спектра:

```
for k=1:14
     framefft(…) = fftshift(fft(…(……)));
end
```

Удалите защитные интервалы:

```
framerx = framefft(……);
framerx = reshape(framerx,[],168);
```

4. Сформируйте в приемнике NSSS последовательность. Так как NSSS зависит от CellID, которое может принимать значения от 1 до 503. Реализуйте формирование NSSS, как

в предыдущих работах. Необходимо найти максимум корреляционной функции всех возможных значений опорного сигнала с принятым.

```
data=[];
data2=[];
for NNCellID = 1:503
   NFrame = ……;
   NSubframe = ……; %Номер подкадра
ns = double(……) ;
nf = fix(……);
u = mod(……) + 3;
q = fix(……);
N = ……;
theta = …/… * mod(……);
n = ……;
nd = [……];
seq = exp(……));
bseq                                                                                                    =
[……];
seq = […… ].*seq;
grid = [……];
Nssframe = reshape(…,[],……);
data = max(abs(xcorr (… , ……)));
data2(NNCellID+1)=data;
end
```

Пример ВКФ приведен на рисунке 2.2.

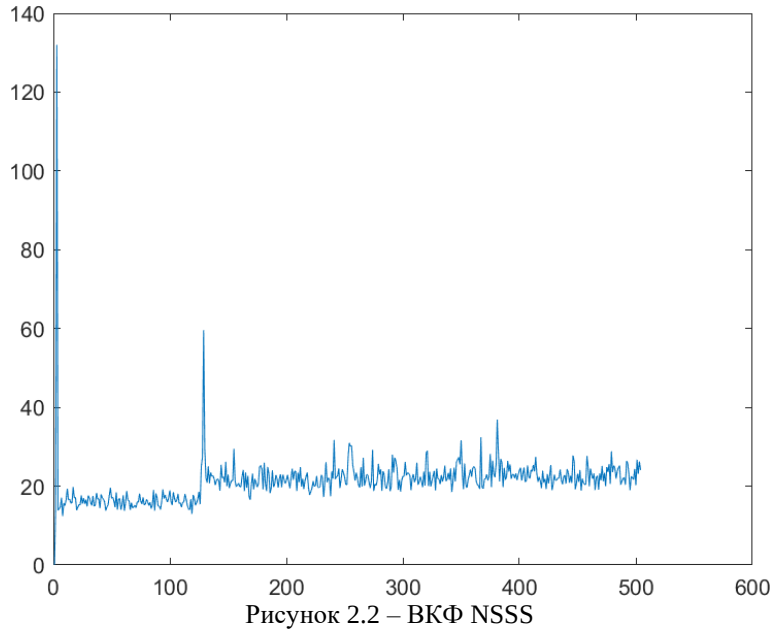


Рисунок 2.2 – ВКФ NSSS

Используя значения максимума, найдите значение NNCellID

```
[d,Cellid] = max(data2);
Cellid = Cellid - 1;
```

3. Контрольные вопросы

1. Что такое кадровая синхронизация?
2. К чему приводит ошибка кадровой (временной синхронизации)?
3. Что такое ВКФ?
4. Каково назначение NNCellID?

Работа № 7

«Частотная синхронизация NB-IoT»

Цель работы: произвести оценку и устранение частотного сдвига в кадре NB-IoT.
Задачи лабораторной работы:

- Загрузить сформированный ранее фрейм NB-IoT.
- Добавить частотный сдвиг в кадр.
- Произвести оценку частотного сдвига.
- Произвести коррекцию частотного сдвига.

1. Теоретический материал

В любой системе связи есть две независимые стороны передающая и принимающая. Генераторы у обоих сторон не синхронизированы друг с другом и могут иметь расхождение по частотам. Это приводит к тому, что в приемнике принятый сигнал будет сдвинут по частоте на разницу между частотами опорных генераторов. Так же частотный сдвиг может возникнуть из-за эффекта Доплера, в результате чего возникают искажения в сигнале при его обработке, что можно увидеть на рисунке 1.1.

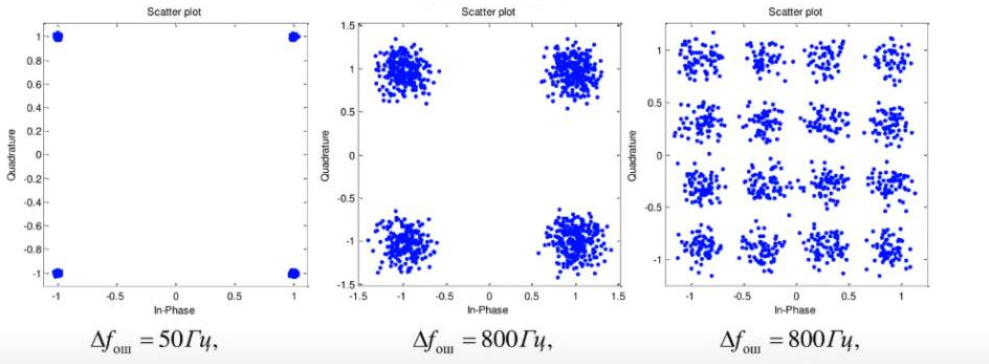


Рисунок 1.1 – Сигналы с ошибкой по частоте

В случае ошибки по частоте для OFDM сигнала, у нас нарушается ортогональность между поднесущими. Уровень шума, возникающий в созвездии, зависит от следующих факторов: величина частотной ошибки и расстояния между поднесущими.

Для оценки ухода частоты нам необходимо найти разность фаз между опорным сигналом, хранящимся в памяти приемного устройства, и принятым. Разность фаз можно найти с использованием ВКФ и поиска его максимума. Для оценки случайной фазы, необходимо использовать две одинаковые половины опорного сигнала, и две одинаковые половины принятого, рассчитать ВКФ между частями принятого и опорного. Далее рассчитываем уход по частоте по формуле, указанной ниже:

$$\Delta f = \frac{\Delta \Phi}{2\pi T}. \qquad (1.1)$$

В работе будут реализованы этапы, представленные на рисунке 1.1.

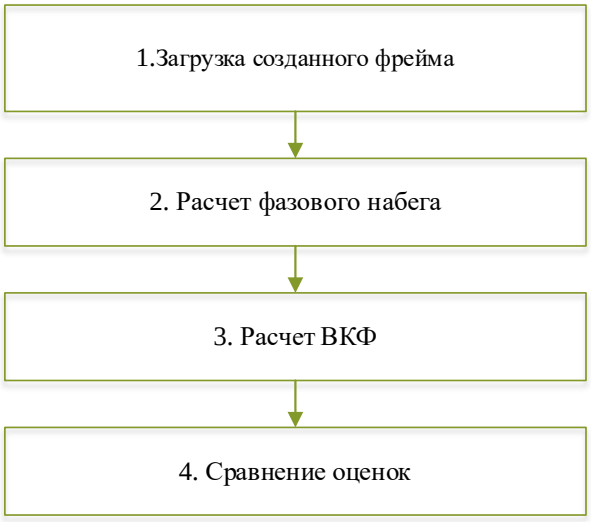


Рисунок 1.1 – Этапы выполнения работы

2. Ход работы

В данной работе необходимо ввести частотный сдвиг в OFDM сигнал. Затем в приемнике произвести оценку частотного сдвига и произвести коррекцию частотного сдвига.

1. Загрузка созданного фрейма

Загрузите сформированный в шестой лабораторной работе кадр и сформированный NPSS сигнал из пятой работы. Далее задайте величину частотного сдвига

```
clc
clear all
close all
load('frame.mat')
load('NPSS.mat')
nfft = 128;
nn =19200;
fs = 1.920e6;
T = nfft/fs;
D_freq = 500;
```

Используя функцию scatterplot() постройте созвездие первого OFDM символа. Полученное созвездие должно соответствовать рисунку 2.1.

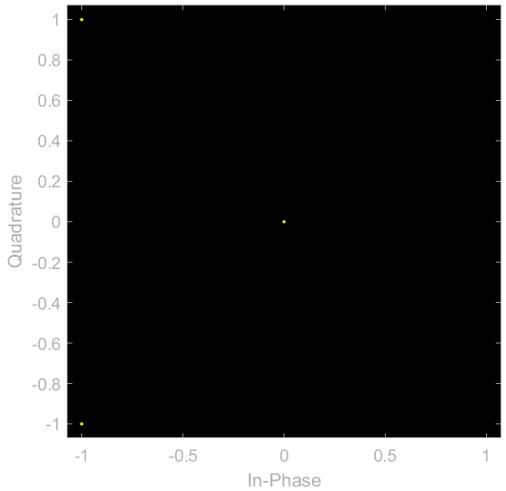


Рисунок 2.1 – Созвездие первого OFDM символа

2. Расчёт Фазового набега

Рассчитайте фазовый набег за время всего сигнала.

```
D_phi1 = D_freq*2*pi*T;
```

Рассчитайте фазовый набег за интервал дискретизации.

```
dphi = D_phi1/nfft;
```

Введите частотный сдвиг.

```
for k = 1:1:nn
   sig_freq_shift(k) = frame(k)*exp(1i* dphi* k);
   end
```

Используя функцию scatterplot, постройте созвездие первого OFDM символа, после ввода частотного сдвига. Полученное созвездие должно соответствовать рисунку 2.2.

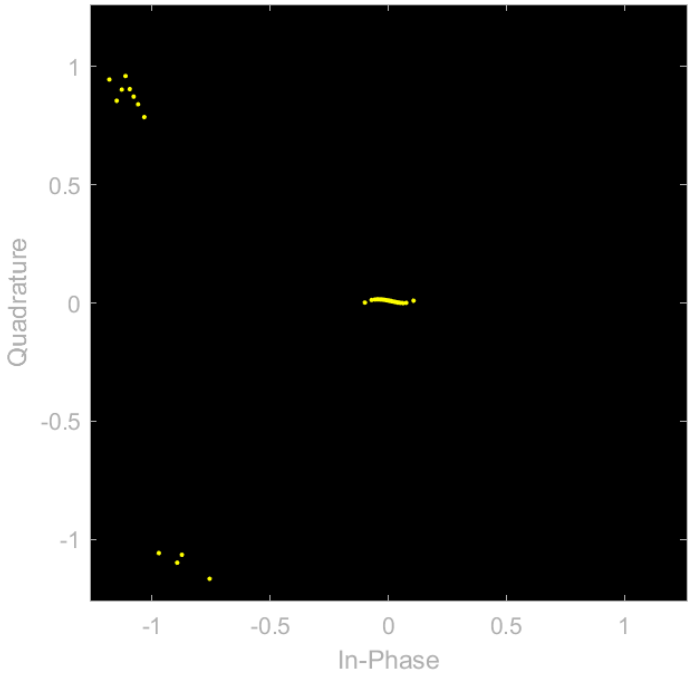


Рисунок 2.2 – Созвездие первого OFDM символа после ввода частотного сдвига

3. Расчёт ВКФ

В лабораторной работе по кадровой синхронизации было найдено начало NPSS сигнала, используя это значение, необходимо вырезать NPSS сигнал из сигнала с добавленным частотным сдвигом.

```
startPSSS = ……;
PSSS = ……(……:……);
PSSS = reshape(PSSS,[],1);
```

Разделите принятый и опорный сигнал на две половины. Произведите расчет ВКФ, первой части принятого сигнала с первой частью опорного (NPSS сформированный в 5 работе), так же и со вторыми частями. Используя функцию max, найдите максимумы ВКФ.

```
frame_ref = NPSSframe(……);
frame_rx = PSSS(……);
corr1 = xcorr(……);
[a,b] = max(……);

frame_ref1 = NPSSframe(……);
frame_rx1 = PSSS(……);
corr2 = xcorr(……);
[c,d] = max(……);
```

Проведите оценку фазового набега

```
est2 = (angle(a)-angle(c));
dt = 1/fs;
tau = (nfft*6)*dt;
ocen_freq = est2/(2*pi*tau);
```

4. Расчет оценки

Рассчитайте частотный сдвиг и сравните его с введённым в начале частотным сдвигом. Они должны быть приблизительно равны.

```
d_phi_ocen = (ocen_freq*2*pi)/fs;
```

Далее рассчитайте фазовый набег за интервал дискретизации по полученной оценке и компенсируйте частотный сдвиг, добавляя фазовый набег, но с отрицательным знаком.

```
for k = 1:1:nn
   sig_freq_shift2(k) = sig_freq_shift(k)*exp(1i* k*(……));
   end
```

Постройте созвездие первого символа после компенсации, пример представлен на рисунке 2.3.

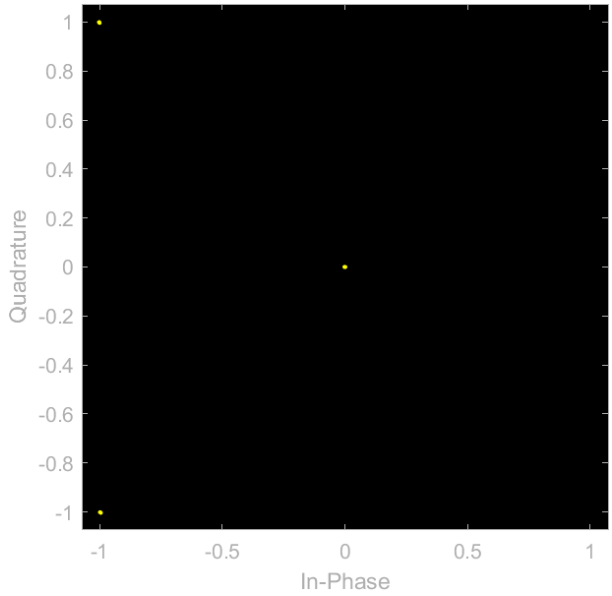


Рисунок 2.3 – Созвездие первого символа после компенсации

3. Контрольные вопросы к работе

1. Что такое частотная синхронизация?
2. К чему приводит ошибка частотной синхронизации?
3. Какие сигнальные конструкции используются для частотной синхронизации?
4. Как компенсировать частотный сдвиг в цифровом виде?

5. Требования к оформлению отчета

1. Введение;
2. Блок схема алгоритма, который вы применили в работе;
3. Рисунки:
- Корреляционная функция,
- Созвездие после введения частотного сдвига,
- Созвездие после устранения частотного сдвига.
4. Выводы;
5. Листинг программы.

Работа № 8

«Оценка канала связи. Эквалайзирование»

Цель работы: произвести оценку канала связи и процедуру эквалайзирования.

Задачи лабораторной работы:

1. Сформировать многолучевой канал.
2. Передать через канал с многолучевостью сформированный кадр с пилотными сигналами.
3. Произвести оценку канала связи и эквалайзирование.

1. Теоретический материал

Из-за многолучевого распространения приходящий к абоненту сигнал представляет собой сумму нескольких отраженных сигналов, как это показано на рисунке 1.1.

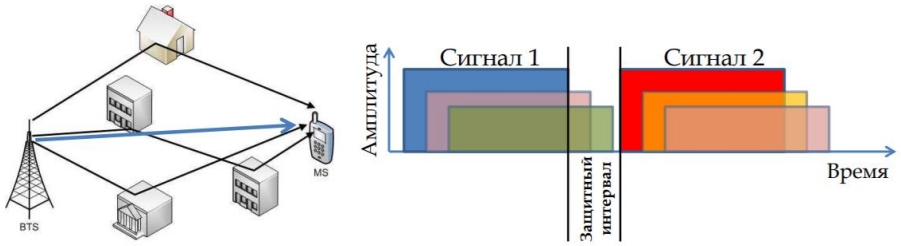


Рисунок 1.1 – Многолучевое распространение

После приёма искаженного каналом сигнала для восстановления его изначальной формы требуется узнать передаточную характеристику канала. Мы имеем изначальный OFDM символ *S(n)* где n – количество отсчетов в символе. В этот сигнал мы встраиваем заранее известные и в передатчике, и в приемнике дискретные пилотные поднесущие $P_k(n)$, где k – номера пилотных поднесущих, n – номера отсчетов, которые занимают эти поднесущие.

В технологии LTE для оценки канала передачи предусмотрены специальные опорные сигналы (Reference Signal, RS). Существует несколько видов опорных сигналов. В данной работе рассматривается только основной опорный сигнал, определяющий ячейку (Cell-Specific), так как этот сигнал передается в любом случае, не зависимо от того, есть ли подключенные к сети абоненты или нет. Остальные виды RS являются вспомогательными. Задачей эквалайзера является наиболее точное сглаживание спектра OFDM-символа по принятым сигналам RS.

Приемнику заранее известен вид RS и его расположение в ресурсной сетке, поэтому приемнику не составляет труда выделить RS в частотной плоскости. Далее, для определения влияния канала нужно определить разницу между принятым и изначальным сигналами RS.

В работе будут реализованы следующие этапы, представленные на рисунке 1.2.

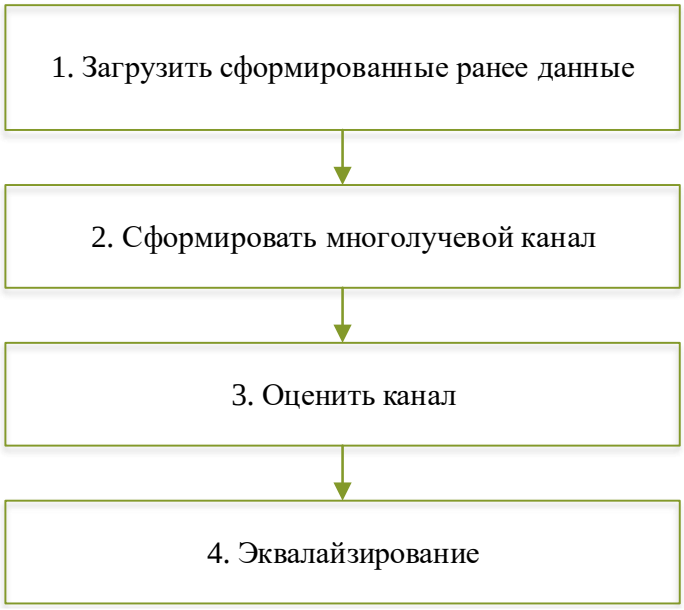


Рисунок 1.2 – Этапы выполнения работы

2. Ход выполнения работы

1. Загрузите сформированную в пятой работе NRS последовательность c, и сформированный кадр из 6 работы.

```
clc
clear all
close all
nfft = 128;
load('NRS.mat')
load('frame.mat')
```

Вырежете из загруженного кадра 1 сабфрейм и постройте его созвездие используя функцию scatterplot().

```
n_frame = 1;
subframe = frame(:, (…….-1)*14+1:……*14)
sf = reshape(subframe, size(…….,1)*size(…….,2),1);
scatterplot(……);
```

Созвездие для информационного символа показана на рисунке 2.1.

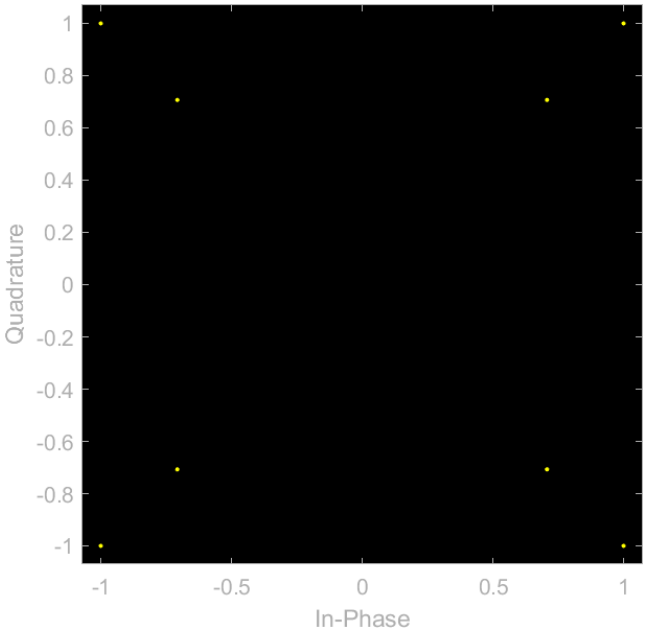


Рисунок 2.1 – Созвездие информационного символа

2. Формирование канала передачи.

Далее выполните моделирование многолучевого канала распространения радиоволн. Установите 5 коэффициентов задержки и 5 коэффициентов ослаблений:

```
fs = 1.920e6;
T = nfft/fs;
pifs2 = 2*pi*fs;
D_time = [0 20 40 60 80] * 1e-8;
atten = [1 0.708 0.724 0.437 0.871];
```

Рассчитайте фазовые набеги, возникающие в результате сдвига на величину, заданную выше.

```
D_phi = D_time .* pifs2;
```

Рассчитайте фазовые набеги за период дискретизации:

```
dphi = D_phi ./ nfft;
```

Добавьте фазовый набег в кадр и произведите обратное преобразование Фурье. Ниже представлен код только для первого луча. По аналогии сделайте для остальных.

```
for j= 1:14
   for k = 1:1:128
   sig_time_shift1(k,j) = frame(k,j)*exp(1i* dphi1* k);

   end
 framefft1(:,j) = (ifft(fftshift (sig_time_shift1(:,j))));

end
```

```
……
```

Просуммируйте сигналы в приемной антенне, умножив каждый на коэффициент ослабления.

```
sig_time_shift = (……+(……*0.708)+(……*0.724)+(……*0.437)+(……*0.871));
```

Используя функцию plot, постройте первый OFDM символ, пример представлен ан рисунке 2.2.

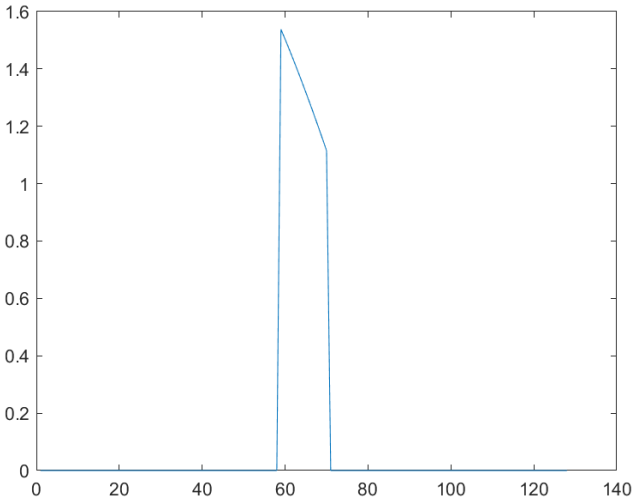


Рисунок 2.2 – Первый OFDM символ

Используя функцию fft переведите сигнал в частотную область, удалите защитные интервалы.

```
for k=1:14
    frame12(:,k) = fftshift(fft(sig_time_shift(:,k)));
end
framerx = frame12(59:70,:);
```

Постройте созвездия сигналов с фазовым набегом, используя функцию scatterplot(). Пример представлен в 2.3.

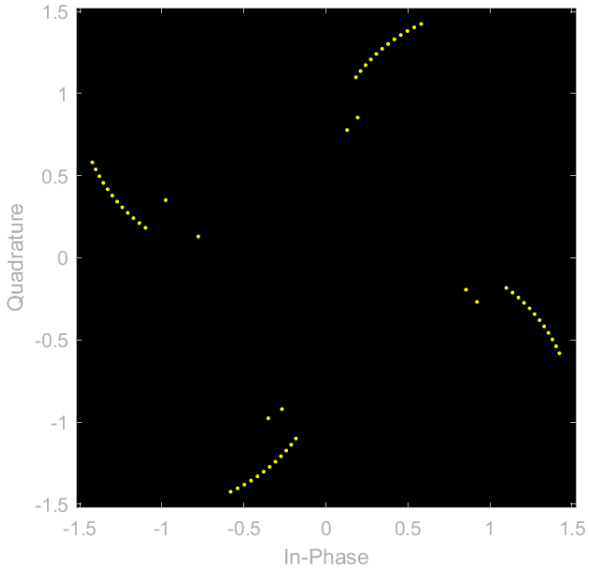


Рисунок 2.3 – Созвездие информационного символа с фазовым набегом

Вырежьте пилотные поднесущие NRS и запишите в отдельный массив.

```
nrcres=[]
  s1 = framerx(…);
  s2 = framerx(…);
  s3 = framerx(…);
  s4 = framerx(…);
  s5 = framerx(…);
  s6 = framerx(…);
  s7 = framerx(…);
  s8 = framerx(…);

nrcres=[……];
```

Произведите оценку передаточной функции канала связи, используя опорную NRS последовательность, загруженную в самом начале и принятую.

```
TF_estimate = nrcres./NRC;
```

3. Используя значения, полученные при оценке передаточной функции, восстановите предположительные значения в столбцах, в которых были расположены пилоты с помощью функции interp1. Ниже предоставлен код только для первого столбца, для остальных трех напишите по аналогии.

```
interpEqCoeffTemp1 =
interp1([6,12],[TF_estimate(1)TF_estimate(2)],1:12,'linear','extrap');
interpEqCoeffTemp2 = ……
interpEqCoeffTemp3 = ……
interpEqCoeffTemp4 = ……
```

Далее по аналогии получите предположительные значения для строк. Полученные значения объедините в один массив. Ниже предоставлен код только для первой строки, для остальных одиннадцати напишите по аналогии.

```
nterpEqCoeffTemp = [];
interpEqCoeffTempstr1 = interp1([5,6,13,14],[interpEqCoeffTemp1(1) interpEqCoeffTemp2(1) interpEqCoeffTemp3(1) interpEqCoeffTemp4(1)],1:14, 'linear','extrap');
interpEqCoeffTempstr2 = ……;
interpEqCoeffTempstr3 = ……;
interpEqCoeffTempstr4 = ……;
interpEqCoeffTempstr5 = ……;
interpEqCoeffTempstr6 = ……;
interpEqCoeffTempstr7 = ……;
interpEqCoeffTempstr8 = ……;
interpEqCoeffTempstr9 = ……;
interpEqCoeffTempstr10 = ……;
interpEqCoeffTempstr11 = ……;
interpEqCoeffTempstr12 = ……;
interpEqCoeffTemp = [……];
```

4. Произведите операцию эквалайзирования, путем деления спектра принятого сигнала на передаточную функцию.

```
vost = framerx./interpEqCoeffTemp;
vost = reshape(vost,168,[]);
```

Выведите на график созвездие сигнала после эквалайзирования. Пример представлен на рисунке 2.4.

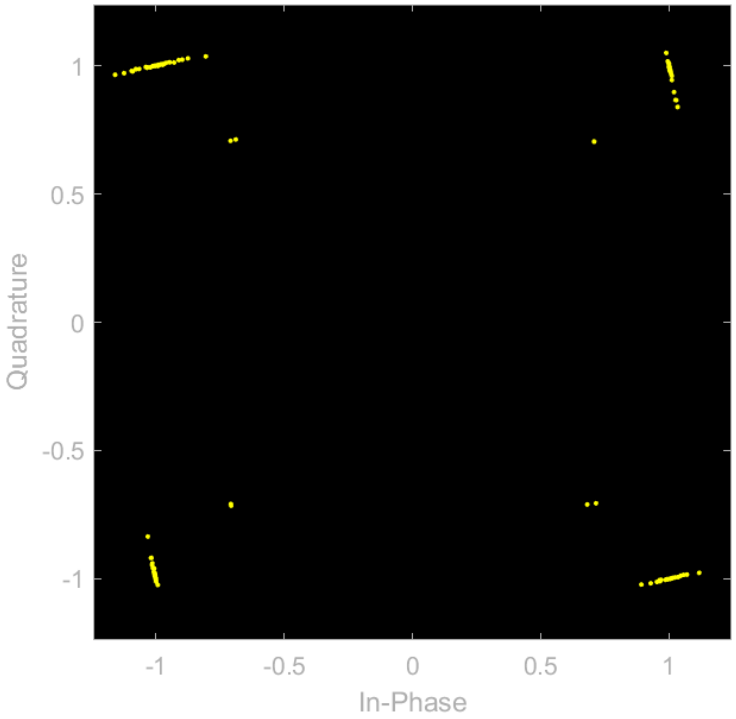


Рисунок 2.4 – Созвездие информационного символа после эквалазирования

3. Контрольные вопросы к работе

1. Что такое пилотный сигнал?

2. Что такое передаточная функция канала РРВ?
3. Как влияет многолучевость канала на передаваемый сигнал?
4. Что такое эквалайзирование?

Работа № 9

«Формирование сигнала, модуляция/демодуляция LoRa»

Цель работы: произвести модуляцию и демодуляцию LoRa сигнала. Передать сигнал через канал, посчитать количество ошибок.

Задачи лабораторной работы:

- Составить программу для передачи ЛЧМ сигнала в среде Octave;
- Произвести модуляцию и демодуляцию сигнала;
- Имитировать передачу сигнала через канал;
- Посчитать количество ошибок;
- Построить up-chirp ЛЧМ сигнал, down-chirp ЛЧМ сигнал, модулированный ЛЧМ сигнал, спектр произведения модулированного и down-chirp сигнала.

1. Краткий теоретический материал

Формирование ЛЧМ-сигнала.

Из материалов лекции Вы узнали, что существуют ЛЧМ сигналы с нарастающей частотой (up-chirp) и убывающей частотой (down-chirp). Внешний вид up-chirp ЛЧМ сигнала и его спектрограмма приведена на рисунке 1.1.

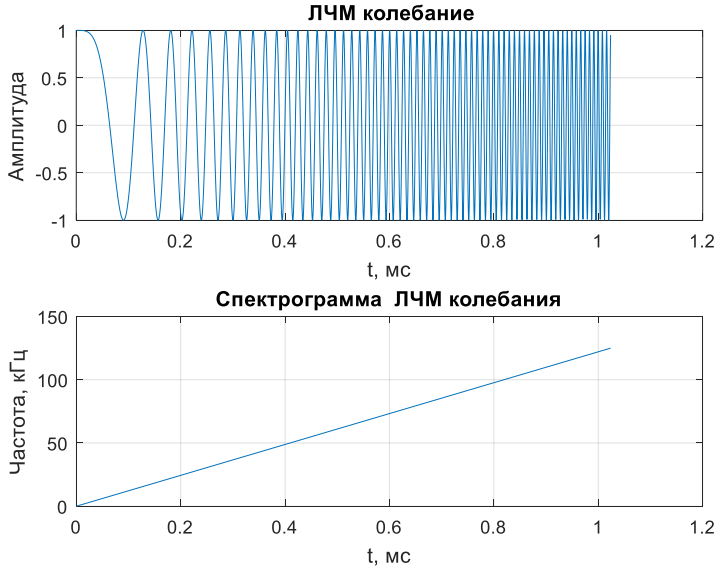


Рисунок 1.1 – ЛЧМ колебание

В зависимости от используемых параметров модуляции, в данном сигнале происходит сдвиг по частоте, в результате которого у сигнала изменяется верхняя и нижняя частота. В данной лабораторной работе будет реализовываться алгоритм с использованием параметров, представленных в таблице 1.1.

Таблица 1.1 – Параметры модели

Параметры	Значение/формула
Полоса сигнала, BW	125 кГц
Коэффициент расширения спектра, SF	7
Длительность сигнала, Ts, рассчитывается по формуле	$\frac{2^{SF}}{BW}$

Окончание таблицы 1.1

Амплитуда сигнала, A	1
Нижняя частота сигнала, f_0	0 кГц
Скорость изменения частоты сигнала, m, рассчитывается по формуле	$m = \frac{BW}{Ts}$
Частота дискретизации, fs	$10 \cdot BW$

Изначально, при формировании сигнала на передающем устройстве исходная битовая последовательность разделяется на пакеты, содержащие SF бит. Далее каждый пакет конвертируется в десятичную систему счисления. Численное значение, которое может принимать пакет, располагается в пределах $0 \ldots 2^{SF}$ и называется информационным символом. Значение символа определяет место сдвига в ЛЧМ сигнале. Процесс модуляции показан на рисунке 1.2, модулированный сигнал (при $SF = 7$, $BW = 125$ кГц и информационном символе равном 27) показан на рисунке 1.3.

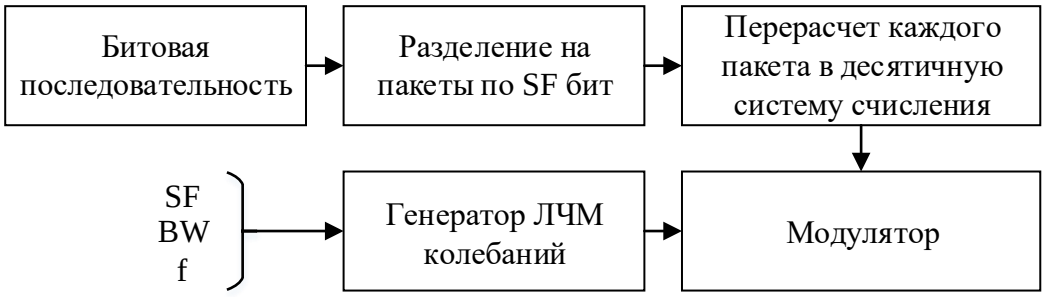


Рисунок 1.2 – Функциональная схема передатчика

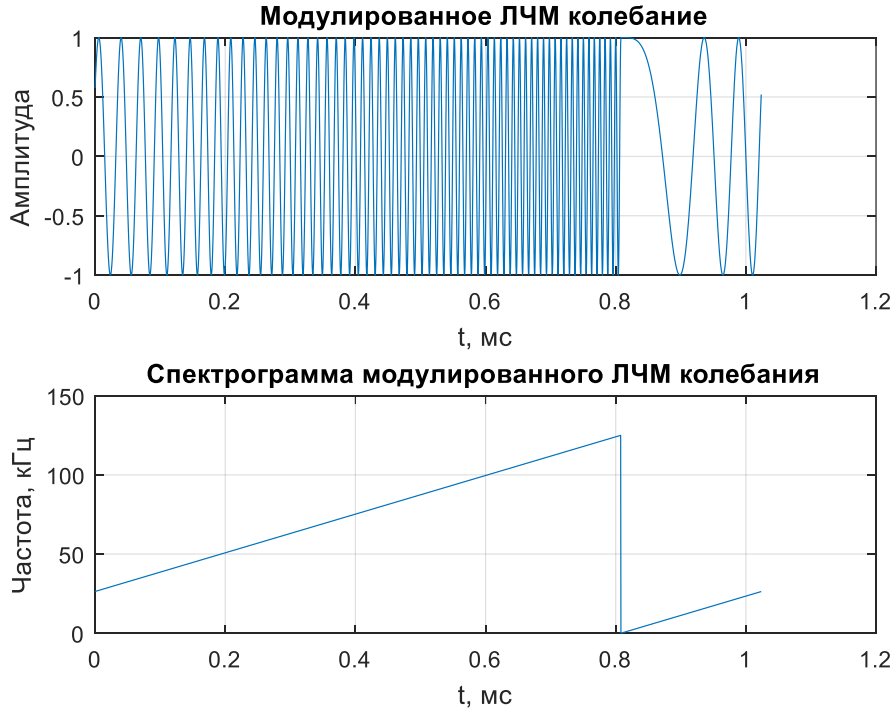


Рисунок 1.3 – Модулированный ЛЧМ сигнал

Демодуляция ЛЧМ-сигнала.

Демодуляция осуществляется перемножением модулированного ЛЧМ сигнала с немодулированным ЛЧМ сигналом убывающей частоты (down-chirp). После чего по их произведению вычисляется преобразование Фурье и ищется отсчет с максимальной амплитудой, номер отсчета численно равен информационному символу. Если номер отсчета больше базы сигнала, то из него следует отнять ее значение. Схема демодулятора дана на рисунке 1.4.

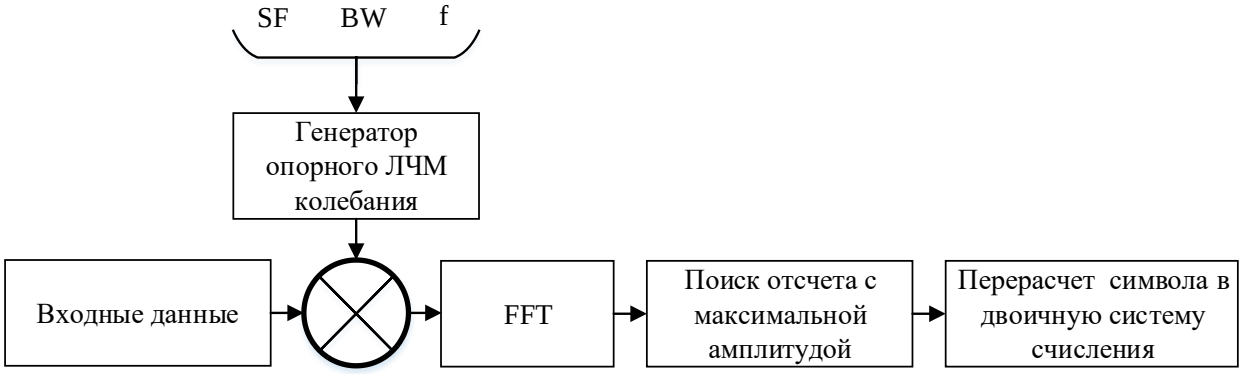


Рисунок 1.4 – Функциональная схема приемника

В данной работе Вам предлагается написать код в программном пакете Octave, реализующий модуляцию битовой последовательности посредством линейной частотной модуляции по алгоритму, представленному на функциональной схеме на рисунке 1.2, а также демодуляцию сигнала с ЛЧМ по схеме, представленной на рисунке 1.4.

2. Ход работы

1. Параметры ЛЧМ

Создайте новый проект в *Octave*. Запишите следующие команды для очистки значений моделирования при каждом повторном перезапуске кода программы:

```
clc
close all
clear all
```

Для формирования ЛЧМ – сигнала в *Octave*, сначала нужно записать его основные параметры, используя таблицу 2.1.

Таблица 2.1 – Параметры модели

Параметры	Значение/формула
Полоса сигнала, BW	125 кГц
Коэффициент расширения спектра, SF	7
База сигнала, B, рассчитывается по формуле	2^{SF}
Длительность сигнала, Ts , рассчитывается по формуле	$\frac{B}{BW}$
Амплитуда сигнала, A	1
Нижняя частота сигнала, f_0	0 кГц

Окончание таблицы 2.1

Скорость изменения частоты сигнала, m, рассчитывается по формуле	$m = \frac{BW}{Ts}$
Частота дискретизации, fs	$10 \cdot BW$

Пример записи переменных из таблицы 1.2 в Octave приведен ниже.

```
BW = 125e3;
SF = 7;
B = ...;
Ts =...;
A = ...;
f0 = ...;
...
```

2. Формирование up-chirp ЛЧМ сигнала

Формирование up-chirp ЛЧМ сигнала осуществляется по формуле (2.1).

$$s = A \cdot \cos\left(2 \cdot \pi \cdot \left(f_0 \cdot t + \frac{m \cdot t^2}{2}\right)\right), \quad (2.1)$$

где f_0 – нижняя частота (частота с которой начинается генерация ЛЧМ колебания);

m – скорость изменения частоты сигнала:

$$m = \frac{BW}{Ts}. \quad (2.2)$$

Теперь нужно определиться с параметром t. Вначале рассчитайте период дискретизации как $ts = 1/fs$. Зная период дискретизации и длительность сигнала, можно задать для него вектор временных отсчетов t. Чтобы определить количество отсчетов необходимо длительность сигнала разделить на период дискретизации $num\_samples = Ts/ts$. Следовательно, вы должны задать параметр t вектором, содержащим $num\_samples = Ts/ts$ отсчетов, изменяться t будет от 0 до $Ts - ts$ с интервалом ts.

```
num_sumples = ...;
t = ...;
```

Далее сформируйте ЛЧМ сигнал, используя формулу (2.1), в Octave она будет выглядеть как:

```
upchirp = A*cos(2*pi*(f0*t + (m*t.^2)/2));
```

Теперь постройте получившийся ЛЧМ сигнал, для этого используйте функцию plot(). Синтаксис для нее записан ниже.

```
figure
plot(t, upchirp);
...
```

Вы должны увидеть такое же изображение, как на рисунке 2.1

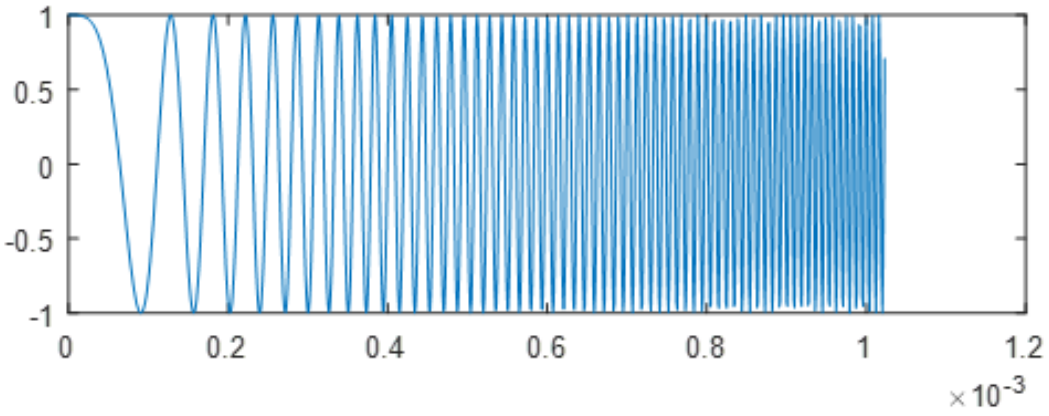


Рисунок 2.1 – ЛЧМ сигнал в OCTAVE

3. Генерация данных

Сгенерируйте случайную последовательность из нулей и единиц (это будет вашим сообщением). В работе будет моделироваться передача одного информационного символа, следовательно, длинна сообщения должна быть равна используемому SF, то есть 7. Для этого используйте функцию randi().

```
data = randi(...);
```

Затем конвертируйте эту последовательность в десятичную систему счисления, для этого можно использовать функцию bi2de().

```
code_word = bi2de(...);
```

4. ЛЧМ Модуляция

Используя полученное число, внесите в ЛЧМ сигнал сдвиг, как на рисунке 1.3. Обратите также внимание на рисунок 1.1, видно, что некоторая часть ЛЧМ сигнала из начала переносится в его конец, фактически она сдвигается. Из спектрограмм рисунков 1.1 и 1.3 это видно более отчетливо. То есть для модуляции необходимо просто сдвинуть часть сигнала из начала в конец. Длительность сдвигаемой части определяется информационным символом, выражение для нее приведено ниже:

$$T_0 = k \cdot T_{\text{int}}, \tag{2.3}$$

где k – текущее значение информационного символа (переменная code_word из фрагмента программы выше); T_{int} – длительность интервала, из которых состоит ЛЧМ сигнал

$$T_{\text{int}} = N \Big/ \left(2^{SF} \cdot f_s\right), \tag{2.4}$$

где N – количество отсчетов при заданном коэффициенте расширения спектра.

$$N = \left(2^{SF} \cdot f_s\right) \Big/ BW \tag{2.5}$$

Как говорилось выше, общее число отсчетов в сигнале равно $num\_samples = Ts/ts$. Значит, чтобы определить номер отсчета, с которого следует сдвигать сигнал, нужно разделить длительность сдвигаемой части на период дискретизации (2.6).

$$num\_shift = T_0/ts = \left(k/BW\right)/ts. \tag{2.6}$$

Полученное число num_shift обязательно должно быть меньше $num\_samples$. Если это не так, перепроверьте предыдущие вычисления.

```
num_shift = ...;
```

Теперь нужно осуществить сдвиг в сигнале. Сам сигнал в Octave выглядит как вектор строка размером $[1 \times num\_samples]$, значит, мы должны взять отсчеты $[1 \times num\_shift]$ и перенести их в конец, а отсчеты от $[(num\_shift+1) \times num\_samples]$ сдвинуть в начало.

```
mod_signal = ...;
```

Постройте получившийся сигнал используя функцию plot(). В результате вы должны увидеть сигнал подобный изображенному на рисунке 2.2. Учтите, что у вас сдвиг в сигнале может отличаться.

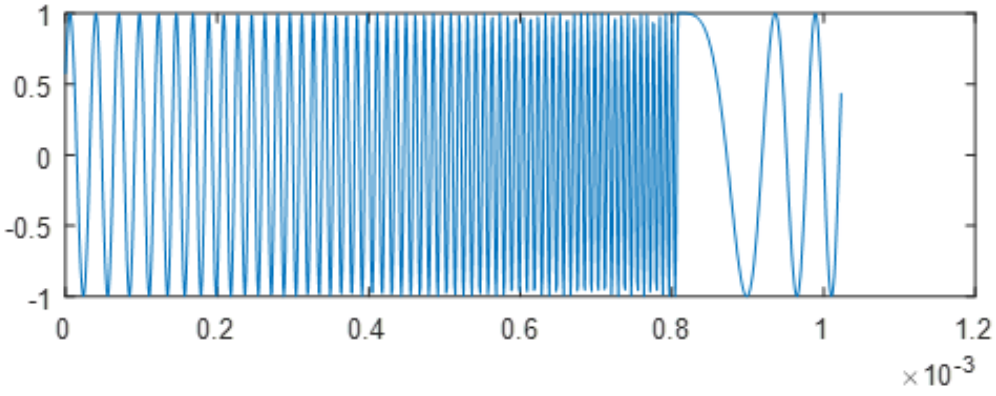


Рисунок 2.2 – Модулированный ЛЧМ сигнал в Octave

5. ЛЧМ демодуляция

Далее можно приступать к демодуляции сигнала. Создайте down-chirp сигнал, это можно сделать, применив функцию flip() к исходному (немодулированному) up-chirp сигналу, образованному по формуле (1.1).

```
downchirp_signal = flip(...)
```

Постройте down-chirp, он должен быть идентичен сигналу на рисунке 2.3.

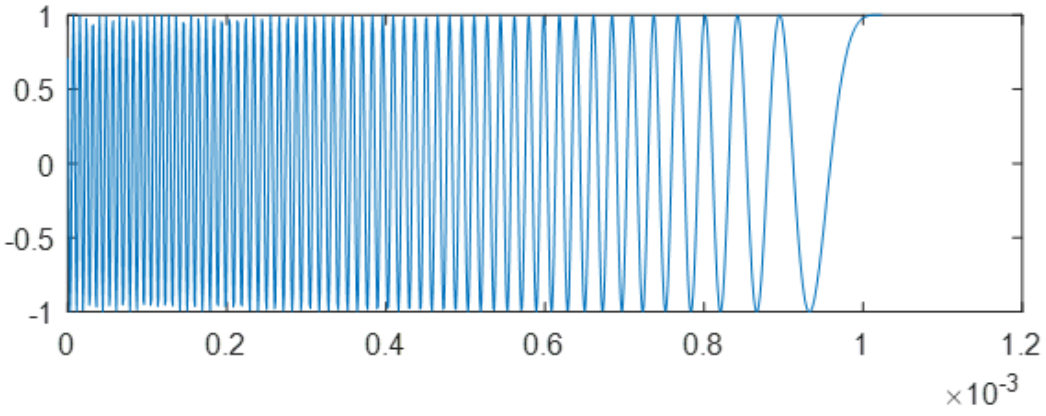


Рисунок 2.3 – Down-chirp ЛЧМ сигнал в OCTAVE

Перемножьте его с модулированным сигналом и примените к их произведению быстрое преобразование Фурье – fft().

```
fourier=abs(fft(downchirp_signal.*mod_signal))
```

Обратите внимание на точку перед знаком умножения, это обозначает элементное перемножение элементов массивов. abs() здесь обозначает взятие по модулю. После этого шага найдите в спектре отсчет с максимальной амплитудой при помощи функции max(), если его значение не превышает базы сигнала, значит он должен быть численно равен информационному символу. В противном случае вычтите из номера отсчета значение базы сигнала. База сигнала равна 2^{SF} .

```
[maxValue,indexMax] = max( fourier )
if indexMax>B
info_symbol = indexMax - B - 1;
else
info_symbol = indexMax - 1;
end
```

Также вы можете построить спектр сигнала и графически найти нужный отсчет:

```
figure
plot(...);
...
```

Результат построения спектра показан на рисунке 2.4.

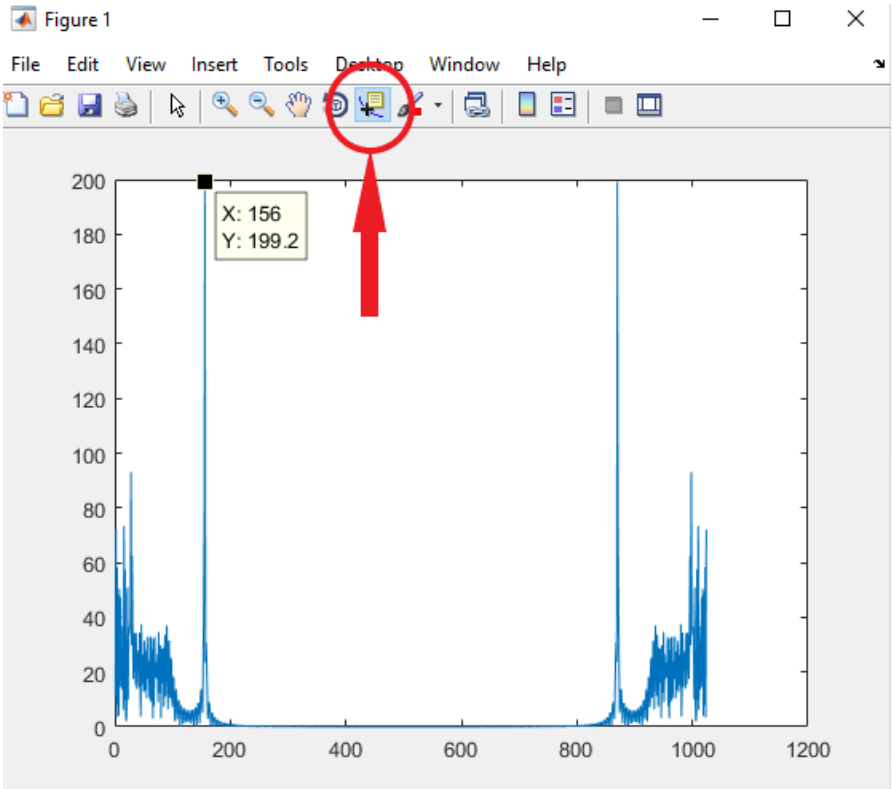


Рисунок 2.4 – Спектр произведения модулированного up-chirp ЛЧМ сигнала с down-chirp сигналом

Узнав значение информационного символа, переведите его в двоичную систему счисления применив к нему функцию de2bi(), обратите внимание, что длина полученного массива, может быть меньше, чем у исходного. Это связано с тем, что данная функция опускает не значащие разряды, то есть нули. Поэтому обозначьте в функции, сколько всего разрядов она должна в итоге отобразить (в нашем случае это SF).

```
demod_data = de2bi(...,SF)
```

6. Расчёт ошибок

Последним шагом с помощью функции biterr() сравните исходную битовую последовательность, с той, которая получилась у вас после демодуляции.

```
num_errors = biterr(...)
```

Число ошибок (значение переменной num_errors**)** должно быть равно нулю, ели это не так, то перепроверьте свою программу.

С помощью функций subplot() и plot() можете отобразить все полученные у вас сигналы в одном графическом окне. Результат выполнения фрагмента ниже показан на рисунке 2.5.

```
figure
subplot(2,2,1)
plot(t,upchirp)
title('upchirp ЛЧМ колебание');
ylabel('Амплитуда'); xlabel('t');
subplot(2,2,2)
plot(t,downchirp_signal)
title('downchirp ЛЧМ колебание');
ylabel('Амплитуда'); xlabel('t');
...
```

subplot(A,B,C) – функция позволяет разбить область вывода графической информации на несколько подобластей, всего подобластей A×B, C – указывает используемую подобласть для вывода рисунка.

title – функция добавляет заголовок над рисунком.

ylabel – функция добавляет подпись оси Y.

xlabel – функция добавляет подпись оси X.

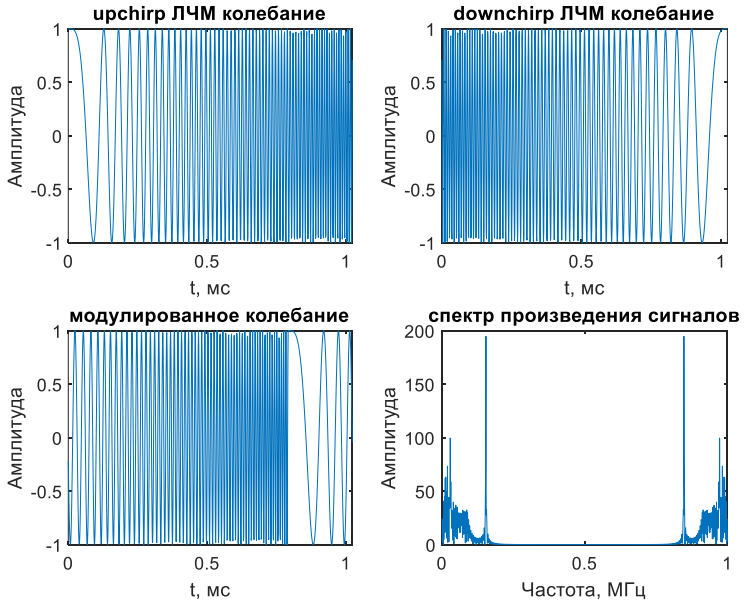


Рисунок 2.5 – Отображение рисунков в одном окне

3. Требования к отчетности и критерии оценивания

Для отчетности предоставьте pdf-файл, содержащий следующие материалы по выполненной работе:

- Полный листинг программы;
- Полученные в ходе выполнения работы графики: up-chirp ЛЧМ сигнал; down-chirp ЛЧМ сигнал; модулированный ЛЧМ сигнал; спектр произведения модулированного и down-chirp сигнала.

Требование к листингу:

Листинг программы представляет собой полученный в результате выполнения хода работы код, предоставленный в текстовом формате .pdf.

Требования к оформлению графиков:

При предоставлении рисунков их нужно оформлять следующим образом:

– указать на рисунке название графика;

– указать на рисунке названия осей;

– указать на рисунке единицы измерения для каждой оси (если присутствуют).

Оформление листинга программы и графиков смотрите в предоставленном примере выполнения задания.

Типовые ошибки при выполнении:

- Использование в данном коде обычных матричных арифметических операторов («+», «-», «/», «*») вместо поэлементных («.+», «.-», «./», «.*») там, где это требуется.
- Построение комплексного спектра сигнала без использования оператора вычисления модуля abs().
- Формирование массивов индексации циклов с нуля, в то время как в Octave вся индексация происходит с первого элемента.

Работа № 10

«Временная синхронизация LoRa»

Цель работы: передать сигнал через канал с аддитивным белым Гауссовским шумом, произвести временную синхронизацию LoRa сигнала, посчитать количество ошибок.

Задачи лабораторной работы:

- Составить программу для передачи и синхронизации ЛЧМ сигнала в среде Octave;
- Произвести модуляцию и демодуляцию сигнала;
- Добавить преамбулу в модулированный сигнал;
- Имитировать передачу сигнала через канал с аддитивным белым Гауссовским шумом;
- Посчитать количество ошибок;
- Построить, up-chirp ЛЧМ сигнал, down-chirp ЛЧМ сигнал, модулированный ЛЧМ сигнал, спектр произведения модулированного и down-chirp сигнала, модулированный ЛЧМ сигнал с преамбулой.

1. Ход работы

В работе будут реализованы этапы, представленные на рисунке 1.1.

Рисунок 1.1 – Этапы выполнения работы

1. Выполнение пункта 1-4 из предыдущей работы.

Обратитесь к лабораторной работе №10 и выполните из нее пункты с первого по пятый для формирования upchirp, mod_signal и downch_flip сигналов или скопируйте и модифицируйте код предыдущей работы.

2. Добавление преамбулы.

Далее необходимо добавить преамбулу к получившемуся модулированному ЛЧМ сигналу. В качестве преамбулы в технологии LoRaWan используется upchirp и downchirp_flip сигналы. Добавьте к модулированному сигналу 2 upchirp и 2 downchirp_flip сигнала в начало.

```
signal = [……,mod_signal];
```

Постройте получившийся сигнал используя функцию plot(). В результате Вы должны увидеть сигнал подобный изображенному на рисунке 1.2.

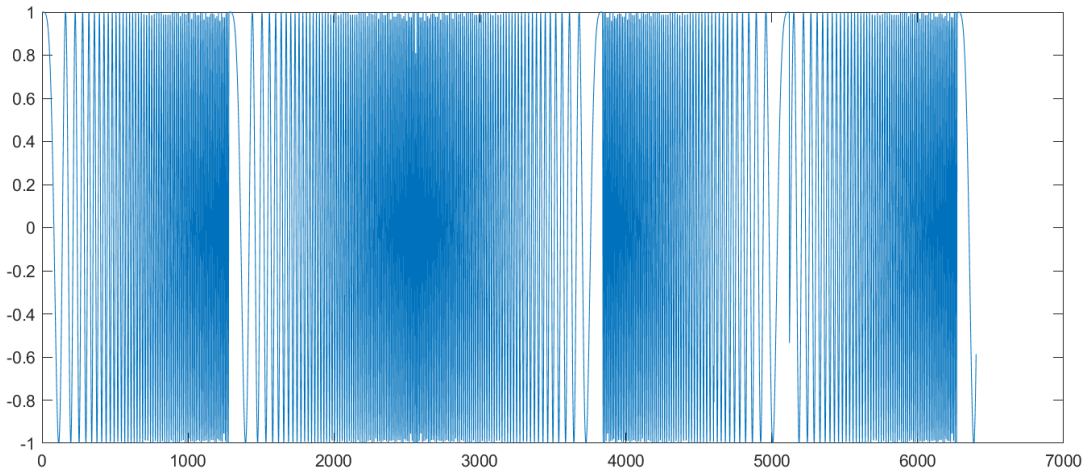


Рисунок 1.2 – Модулированный ЛЧМ сигнал с преамбулой в *Octave*

3. Добавление аддитивного Гауссовского шума и задержки в канал.

Добавим аддитивный Гауссовский шум и задержку в канал, задайте значение SNR = 5.

```
Delay = 10;
chanell_chirp_delay = [zeros(1,Delay),channel_chirp];
channel_chirp_sto = awgn(...);
```

4. Корреляция.

Так как передатчик может начать генерацию радиосигнала в любой момент времени необходим механизм, обеспечивающий синхронизацию приемника по сигналу от передатчика. В качестве такого механизма используется преамбула, которую мы добавили в сигнал выше.

Для временной синхронизации необходимо провести корреляцию развернутого принятого сигнала с преамбулой и найти максимум корреляционной функции.

```
corr = flip(xcorr(...));
[A, B]=max(corr)
```

Постройте график корреляционной функции используя функцию plot(). В результате вы должны увидеть сигнал подобный изображенному на рисунке 1.3.

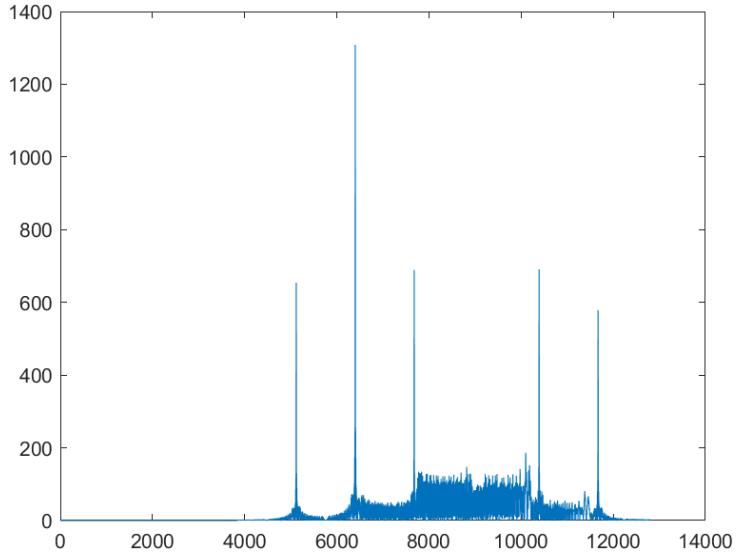


Рисунок 1.3 – Корреляционная функция принятого сигнала в Octave

5. Поиск начала сообщения.
Теперь необходимо найти начало выделенного сообщения после коррелятора:

```
start = ((B-floor(length(corr)/2)));
Rx = channel_chirp_sto(start:start+length(mod_signal)*5-1);
```

Запишите сообщение без преамбулы.

```
Rx2 = Rx(1280*4+1:1280*5);
```

Постройте график принятого сигнала без преамбулы, он должен совпадать с переданным сигналом. На рисунке 1.4 изображено принятый сигнал во временной области.

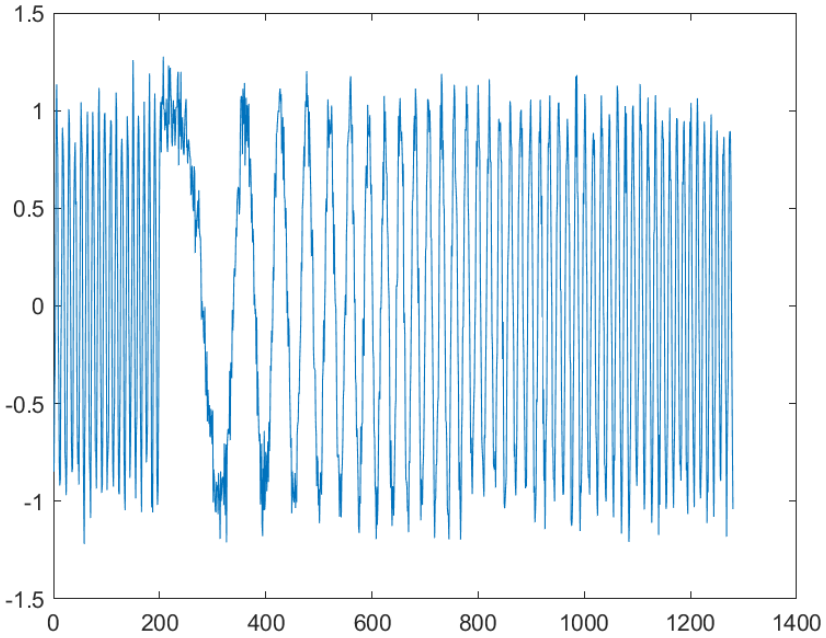


Рисунок 1.4 – Принятый сигнал

6. Демодуляция.

Далее используя лабораторную работу №10 произведите демодуляцию, расчет ошибок и постройте графики: модулированного сигнала, сигнала с преамбулой, сигнал с шумом, БПФ в одном окне.

На рисунке 1.5 представлен пример построения графиков.

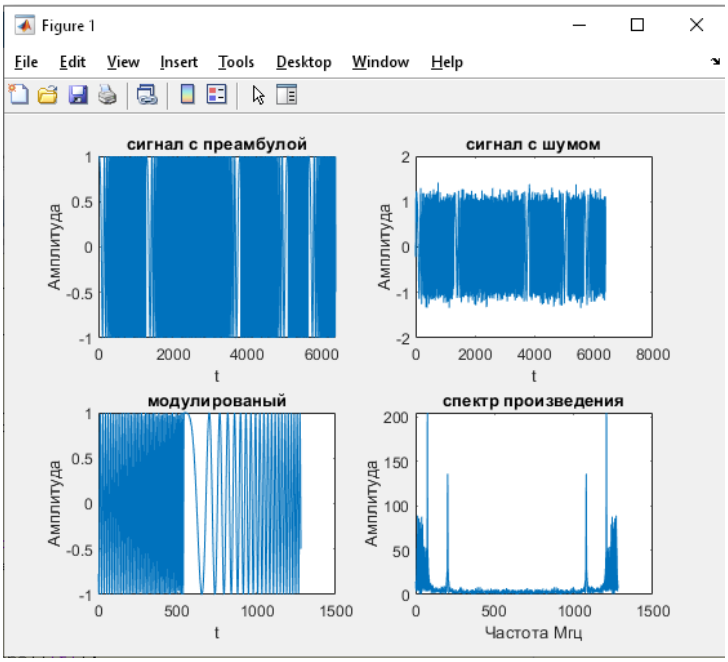


Рисунок 1.5 – Финальные рисунки

2. Требования к отчетности

Для отчетности предоставьте pdf-фаил, содержащий следующие материалы по выполненной работе:

- Полный листинг программы;
- Полученные в ходе выполнения работы графики: up-chirp ЛЧМ сигнал; ЛЧМ сигнал с шумом; Модулированный ЛЧМ сигнал; Спектр произведения модулированного и down-chirp сигнала.

Требование к листингу:

Листинг программы представляет собой полученный в результате выполнения хода работы код, предоставленный в текстовом формате .pdf.

Требования к оформлению графиков:

При предоставлении рисунков их нужно оформлять следующим образом:

– указать на рисунке название графика;

– указать на рисунке названия осей;

– указать на рисунке единицы измерения для каждой оси (если присутствуют).

Оформление листинга программы и графиков смотрите в предоставленном примере выполнения задания.

Работа № 11

«Частотная синхронизация LoRa»

Цель работы: произвести частотную синхронизацию LoRa сигнала.
Задачи лабораторной работы:

- Составить программу для передачи и синхронизации ЛЧМ сигнала в среде Octave;
- Произвести модуляцию и демодуляцию сигнала;
- Добавить преамбулу в модулированный сигнал
- Имитировать передачу сигнала через канал с аддитивным белым Гауссовским шумом
- Произвести частотную синхронизацию методом оценки частоты.
- Посчитать количество ошибок
- Построить, up-chirp ЛЧМ сигнал, down-chirp ЛЧМ сигнал, модулированный ЛЧМ сигнал, спектр произведения модулированного и down-chirp сигнала, модулированный ЛЧМ сигнал с преамбулой, сигналы трех оценок.

1. Ход работы

В работе будут реализованы этапы, представленные на рисунке 1.1.

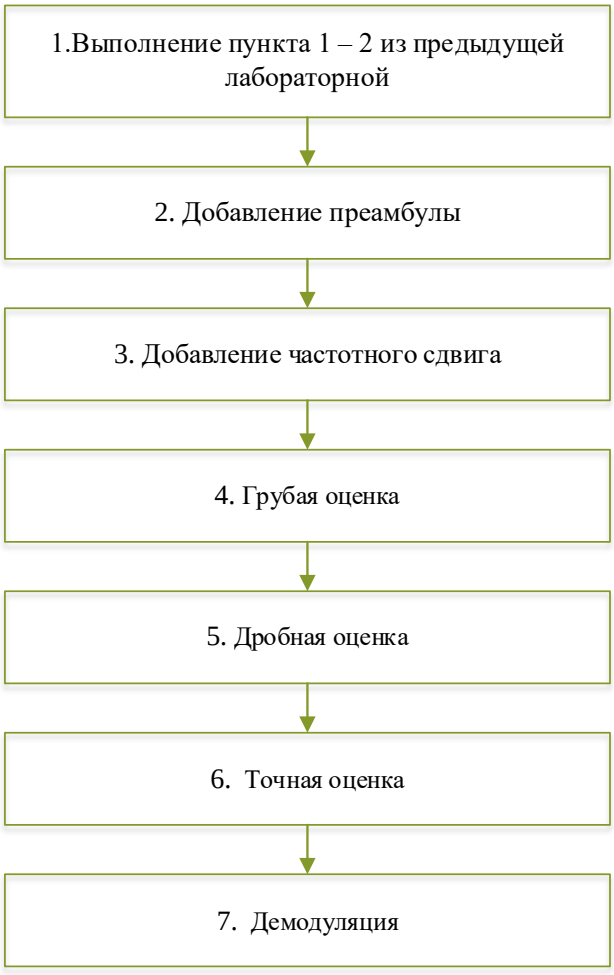


Рисунок 1.1 – Этапы выполнения работы

1. Выполнение пункта 1-2 из предыдущей работы.

Обратитесь к лабораторной работе №10 и выполните из нее пункты с первого по пятый для формирования upchirp, mod_signal, downch_flip сигналов, а также добавьте к сформированной последовательности преамбулы, как это выполнялось в работе №11 или скопируйте и модифицируйте код предыдущей работы.

2. Частотный сдвиг

Введите частотный сдвиг, как показано ниже:

```
freq_shift = 0;
dphi=freq_shift*2*pi*ts;
for j=1:length(signal)
channel_chirp(j)=signal(j)*exp(1i*dphi*j);
end
```

Добавьте аддитивный Гауссовский шум в канал, задайте значение SNR = 5;

```
channel_chirp_sto = awgn(...)
```

Для частотной синхронизации необходимо провести три вида оценки частоты: грубую, уточняющую (дробную), точную:

3. Грубая оценка

Для грубой оценки необходимо демодулировать преамбулу. Так как чирп разделен на 128 частей (называются чипами), необходимо определить какой уход частоты вызывает смещение одной части. Для этого посчитаем fps (frequency per symbol), далее демодулируем преамбулу. Если преамбула окажется модулированной, то это свидетельствует о наличии частотного сдвига.

```
fps = BW/Base;
N = num_samples;
mreamb = channel_chirp_sto(1:2560);
for i = 1:2
[max_a1, ind1] = max(fft(mreamb_rx(i*N-N+1:N*i).*downch_flip));
pre_align(i) = ind1; % pre_align массив с демодулированной информацией
if (pre_align(i)>1280/2)
   pre_align(i) = pre_align(i)-1280;
end
end
```

Далее необходимо провести оценку и сдвиг:

```
est1 = (mean(pre_align-1))*fps;% оценка
dphi1 = est1*2*pi*ts; % сдвиг
for j=1:length(channel_chirp_sto)
channel_chirp_realign(j)=channel_chirp_sto(j).*exp(1i*dphi1*j*(-1));
end
```

Постройте получившийся сигнал, используя функцию plot(). На рисунке 1.2 изображен пример сигнала грубой оценки.

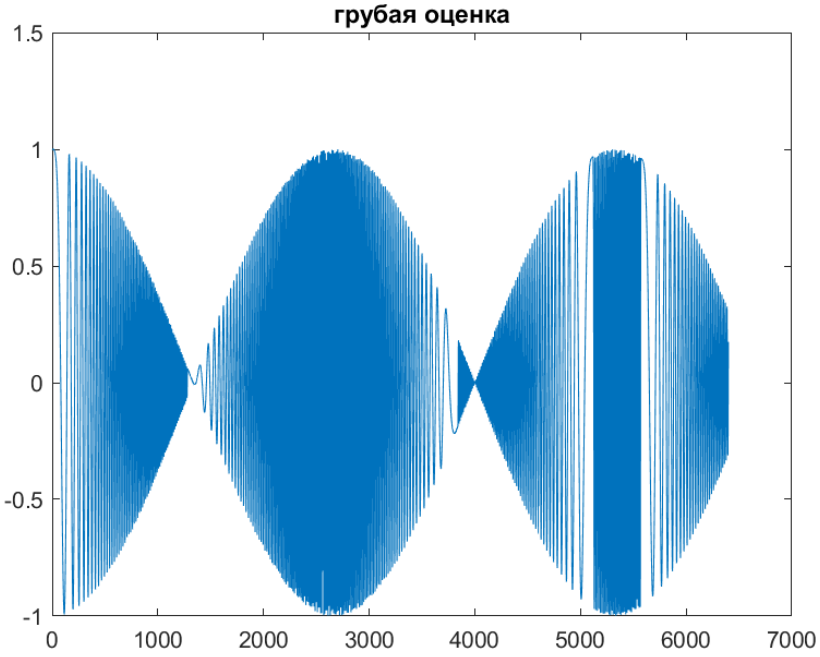


Рисунок 1.2 – Грубая оценка частоты в *Octave*

4. Дробная оценка

Так как одному чипу соответствуют частоты 976 Гц, то максимальная ошибка грубой оценки частоты составляет 976/2 ~ 488.

Принятый символ преамбулы разделите на две части и произведите их сравнение с опорным чирпом. Опорный чирп так же необходимо разделить на две части и сравнить их. Сравниваются не целые символы, а их половины, так как мы не можем оценить сдвиг выше 488 Гц, а под воздействием шума грубая оценка может дать большую ошибку.

```
left_half = channel_chirp_realign(1:N/2); % первая половина принятого
left_ref = downch_flip(1:N/2); % первая половина опорного
right_half = ...    % второя половина опорного
right_ref = ... % вторая половина принятого
```

Далее произведите БПФ и найдите максимум.

```
bpf3 = fft(left_half.*left_ref);
bpf4 = fft(right_half.*right_ref);
[max_a3] = max(bpf3);
[max_a4] = max(bpf4);
a11=max(max_a3);
a12=max(max_a4);
```

Оценка и сдвиг:

```
 est2 = (angle(a12)-angle(a11))/(pi*TS);
dphi2 = est2*2*pi*TS/N;
```

Произведите уточняющее устранение фазового набега:

```
for j=1:length(channel_chirp_realign)
```

```
channel_chirp_frac_est(j) = ...
```

Постройте получившийся сигнал. В результате Вы должны увидеть сигнал подобный изображенному на рисунке 1.3.

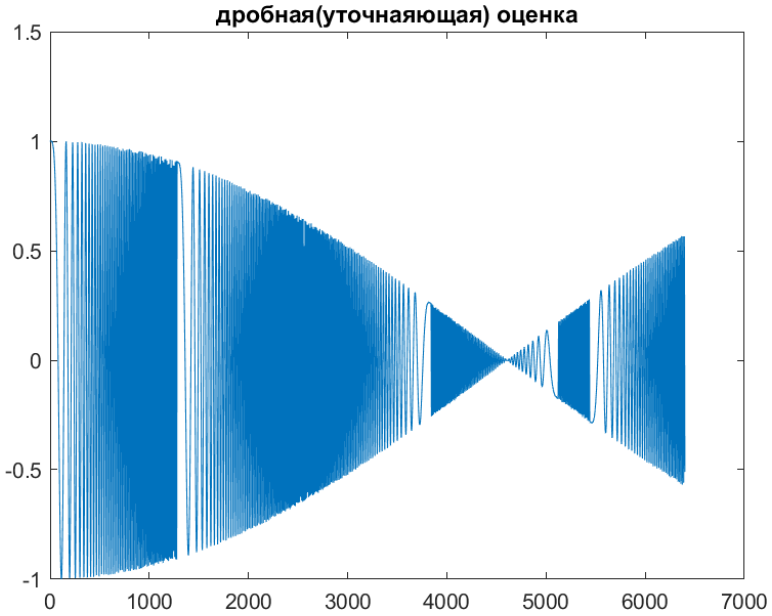


Рисунок 1.3 – уточняющая оценка частоты в OCTAVE

5. Точная оценка

Благодаря уточняющей оценки мы предотвращаем возможность сдвига выше 488Гц, поэтому в точную оценку частоты мы можем сравнить между собой два принятых символа преамбулы. Проще говоря, мы считаем корреляцию между двумя принятыми символами. И находим максимум аргумента корреляционной функции.

```
argumon=channel_chirp_frac_est(1:N).*conj(channel_chirp_frac_est(N+1:2*N));
[arg] = max(sum(argumon));
```

Оцените уход частоты:

```
est3 = -angle(arg)/(2*pi*TS);
dphi3 = est3*2*pi*TS/N;
```

Произведите уточняющее устранение фазового набега:

```
for j=1:length(channel_chirp_realign)
 channel_chirp_frac_est3(j) = ...
```

Постройте получившийся сигнал и сигнал channel_chirp_sto в одном графическом окне. В результате Вы должны увидеть сигнал подобный изображенному на рисунке 1.4.

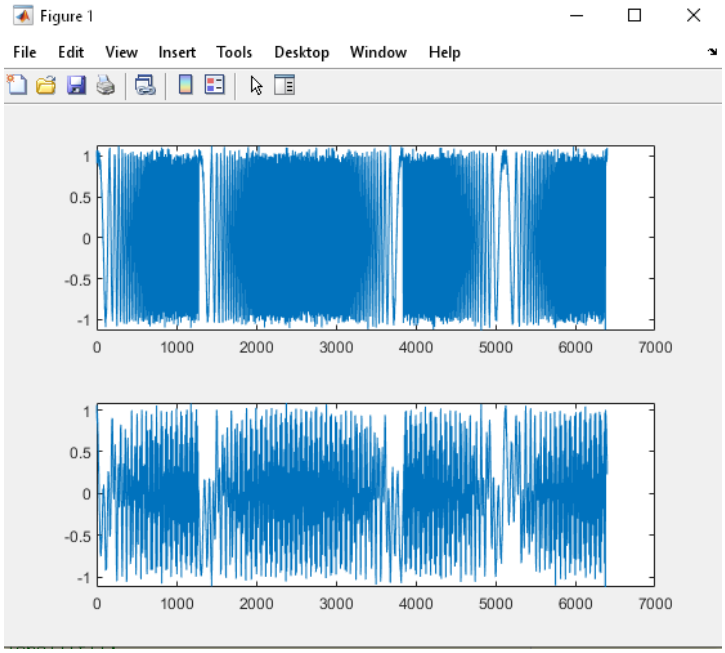


Рисунок 1.4 – Точная оценка частоты в Octave

Посчитаете разницу между введенным частотным сдвигом и тремя переменными оценки частоты, значение должно стремиться к 0. Выделите информационную часть из канала

```
est_full = ...;
Rx2 = channel_chirp_frac_est3(...);
```

6. Демодуляция

Далее используя лабораторную работу №11 произведите демодуляцию, расчет ошибок и постройте в одном графическом окне графики модулированного сигнала, сигнала с шумом, сигналы трех оценок, и сигнал после БПФ. На рисунке 1.5 изображен пример графиков, полученных в результате выполнения задания.

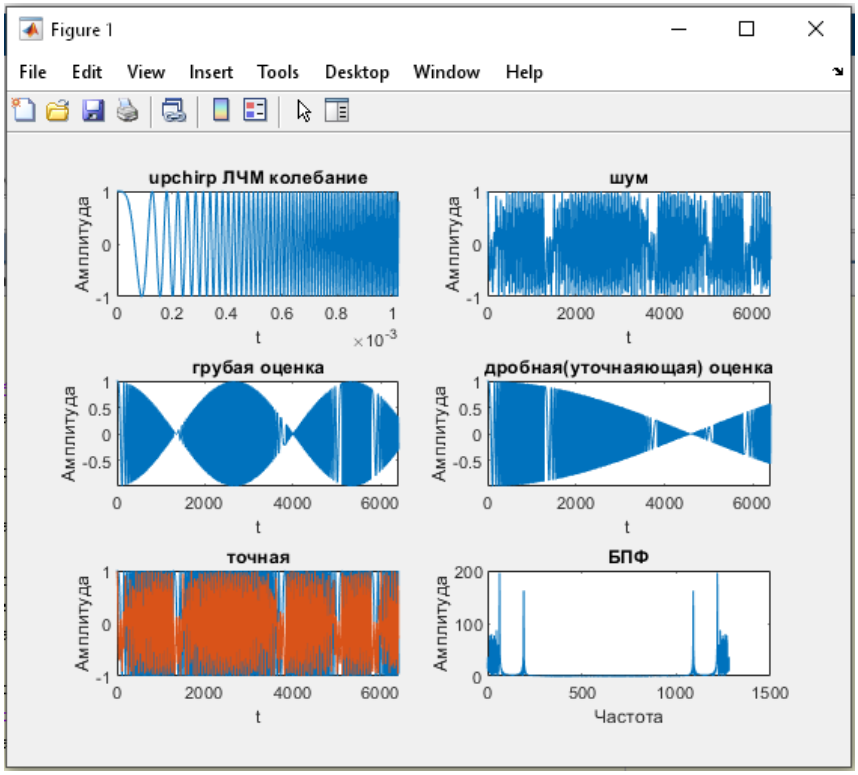


Рисунок 1.5 – Графики в *Octave*

2. Требования к отчетности и критерии оценивания

Для отчетности предоставьте pdf-фаил, содержащий следующие материалы по выполненной работе:

- Полный листинг программы;
- Полученные в ходе выполнения работы графики: up-chirp ЛЧМ сигнал; ЛЧМ сигнал с шумом; модулированный ЛЧМ сигнал; сигналы трех оценок.

Требование к листингу:

Листинг программы представляет собой полученный в результате выполнения хода работы код, предоставленный в текстовом формате .pdf.

Требования к оформлению графиков:

При предоставлении рисунков их нужно оформлять следующим образом:

– указать на рисунке название графика;

– указать на рисунке названия осей;

– указать на рисунке единицы измерения для каждой оси (если присутствуют).

Оформление листинга программы и графиков смотрите в предоставленном примере выполнения задания.

Работа № 12

«Формирование сигналов первичной и вторичной синхронизации стандарта 5G NR»

Цель работы: сформировать первичный (PSS) и вторичный (SSS) сигналы синхронизации.

Задачи лабораторной работы:

- Сформировать первичный сигнал синхронизации (PSS).
- Сформировать вторичный сигнал синхронизации (SSS).
- Провести формирование блока синхронизации (SS/PBCH).

В настоящей лабораторной работе принят следующий перечень сокращений:

5G – Fifth Generation, пятое поколение;

DMRS – Demodulation Reference Signal, опорный сигнал демодуляции;

LTE – Long-Term Evolution, стандарт беспроводной высокоскоростной передачи данных;

NR – New Radio, новое радио;

OFDM – Orthogonal Frequency-division Multiplexing, мультиплексирование с ортогональным частотным разделением каналов;

PSS – Primary Synchronization Signal, первичный сигнал синхронизации;

SSS – Secondary Synchronization Signal, вторичный сигнал синхронизации;

PDSCH – Physical Downlink Shared Channel, физический канал для передачи информации "вниз" с разделением пользователей;

PDCCH – Physical Downlink Control Channel, физический канал управления "вниз";

PBCH – Physical Broadcast Channel, широковещательный физический канал;

RB – Resource Block, ресурсный блок;

SCs – Subcarriers, поднесущие;

SSB – Synchronization Signal Block, блок сигнала синхронизации.

1. Теоретический материал

Для того чтобы осуществить передачу и прием данных мобильная станция должна быть синхронизирована с базовой станцией. Для этого базовая станция передает два нисходящих сигнала синхронизации: первичный сигнал синхронизации (PSS) и вторичный сигнал синхронизации (SSS). Деление на два сигнала направлено на уменьшение сложности процесса поиска идентификатора соты (N_{ID}^{Cell}).

PSS позволяет пользовательскому терминалу осуществить частотную и временную синхронизацию с сигналом активной соты, а также вычислить компонент $N_{ID}^{2} \in \{0,1,2\}$ физического идентификатора соты.

SSS позволяет пользовательскому терминалу вычислить компонент $N_{ID}^{1} \in \{0,1,\ldots,335\}$ физического идентификатора соты.

В 5G NR блок синхронизации (SSB) содержит PSS, SSS сигналы, PBCH (данные) и DMRS (опорный сигнал) в последовательных символах OFDM. Каждый SSB занимает четыре OFDM символа и располагаются на 240 поднесущих (20 RB). PSS и SSS охватывают 127 центральных поднесущих (SCs). На рисунке 1.1 приведена частотно-временная структура SSB.

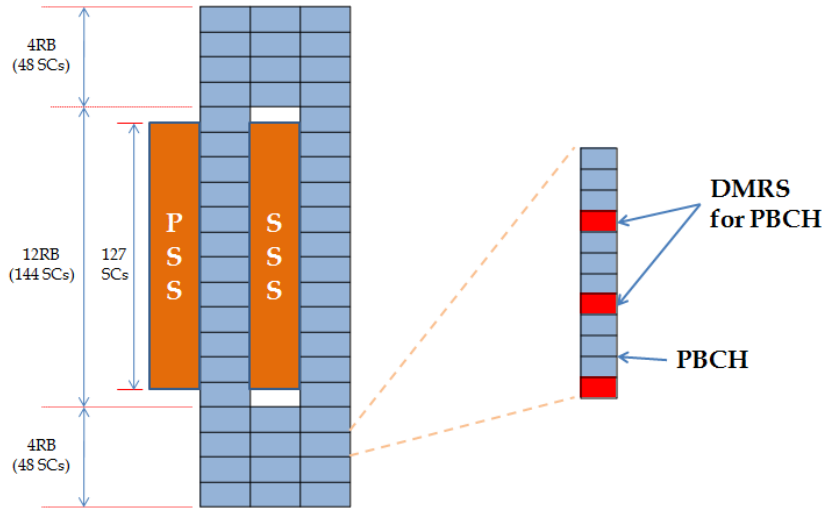


Рисунок 1.1 – Частотно-временная структура SSB

Частотно-временное распределение SSB приведено в таблице 1.1.

Таблица 1.1 – Частотно-временное распределение SSB

Канал или сигнал	Номер символа OFDM "l" относительно начала SSB	Номер поднесущей "k" относительно начала SSB
PSS	0	56, 57, … ,182
SSS	2	56, 57, … ,182
Установка на 0	0	0, 1, … , 55, 183, 184, …, 236
	2	48, 49, …, 55, 183, 184, …,191
PBCH	1 и 3	0, 1, …, 239
	2	0, 1, …,47, 192, 193, …, 239
DMRS для PBCH	1 и 3	$0+v$, $4+v$, $8+v$, …, $236+v$
	2	$0+v$, $4+v$, $8+v$, …, $44+v$ $192+v$, $196+v$, …, $236+v$
$v = N_{ID}^{Cell} \ mod 4$		

Первичный сигнал синхронизации (PSS).

Как говорилось ранее PSS сигнал – является особым сигналом физического уровня, который используется для синхронизации радиоканала, а также для передачи внутригруппового идентификатора.

Формирование PSS осуществляется путем генерации М-последовательности.

М-последовательность – это псевдослучайная двоичная последовательность, порождённая регистром сдвига с линейной обратной связью и имеющая максимальный период. Схема генерации М-последовательности приведена на рисунке 1.2.

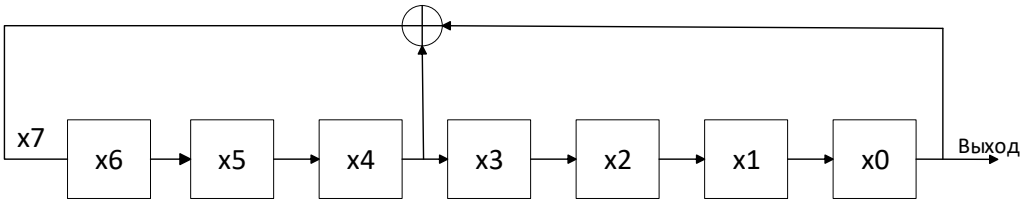


Рисунок 1.2 – Схема генерации М-последовательности

Ниже представлено описание PSS сигнала:

– составлен из 127 значений М-последовательности;

– используется для синхронизации кадров нисходящей линии связи;

– определяет физический идентификатор ячейки.

NR PSS последовательность генерируется по формулам:

$$d_{PSS}(n) = 1 - 2x(m), \tag{1.1}$$

$$m = (n + 43N_{ID}^{2}) mod 127, \tag{1.2}$$

$$N_{ID}^{2} \in \{0,1,2\}, \tag{1.3}$$

$$0 \le n < 127, \tag{1.4}$$

где

$$x(i+7) = (x(i+4) + x(i)) mod 2, \tag{1.5}$$

и

$$[x(6)\ x(5)\ x(4)\ x(3)\ x(2)\ x(1)\ x(0)] = [1\ 1\ 1\ 0\ 1\ 1\ 0]. \tag{1.6}$$

Поскольку n принадлежит диапазону от 0 до 126, можно сделать вывод, что длина PSS последовательности равна 127. Так как N_{ID}^{2} принадлежит трем значениям, тогда может быть три варианта PSS последовательности.

SSS (вторичный сигнал синхронизации).

Как говорилось ранее, SSS сигнал – это сигнал физического уровня, который используется для обнаружения группы идентификаторов сот.

Ниже представлено описание сигнала SSS:

– состоит из 127 значений М-последовательности

– используется для синхронизации кадров нисходящей линии связи.

NR SSS последовательность генерируется по формулам:

$$d_{sss}(n) = [1 - 2x_0((n + m_0) \bmod 127)][1 - 2x_1((n + m_1) \bmod 127)], \tag{1.8}$$

$$m_{\ 0} = 15\left\lfloor \frac{N_{ID}^{1}}{112} \right\rfloor + 5N_{ID}^{2}, \tag{1.9}$$

$$m_1 = N_{ID}^{1} \bmod 112, \tag{1.10}$$

$$0 \le n < 127, \tag{1.11}$$

где

$$\begin{aligned} x_0(i+7) &= (x_0(i+4) + x_0(i)) \bmod 2, \\ x_1(i+1) &= (x_1(i+4) + x_1(i)) \bmod 2 \end{aligned}, \tag{1.12}$$

и

$$\begin{aligned} [x_0(6)\ x_0(5)\ x_0(4)\ x_0(3)\ x_0(2)\ x_0(1)\ x_0(0)] &= [0\ 0\ 0\ 0\ 0\ 0\ 1], \\ [x_1(6)\ x_1(5)\ x_1(4)\ x_1(3)\ x_1(2)\ x_1(1)\ x_1(0)] &= [0\ 0\ 0\ 0\ 0\ 0\ 1]. \end{aligned} \tag{1.13}$$

Базовая станция определяет набор идентификаторов соты N_{ID}^{1}, которые принимают значения от 0 до 335, принимая сигнал SSS. После этого мобильная станция вычисляет номер идентификатора соты, который нужен для определения положения пилотных сигналов.

2. Ход выполнения работы

1. Для того чтобы сформировать сигнал PSS воспользуйтесь формулам (1.1 – 1.6). Изначально задайтесь параметрами N_{ID}^{2} и N_{ID}^{1}, от которых будет зависеть идентификатор соты и формирующие последовательности синхросигналов. Помните, что N_{ID}^{2} может принимать значения $\{0,1,2\}$, а N_{ID}^{1} – $\{0,1,\ldots,335\}$.

```
Nid1 = ...;
Nid2 = ...;
```

2. Задайте начальное значение полинома *x*, которое одинаково для всех трех значений N_{ID}^{2}. Обратите внимание на индексы в формуле (1.6). Учтите, что в формуле индексация идет справа налево, в то время как в Octave наоборот – слева направо.

```
x(1:7) = ...;
```

3. Используя цикл for, рассчитайте по формуле (5) 127 значений М-последовательности.

```
for i = 1 : ...
 x(i + 7) = ...;
end
```

4. Рассчитайте значение *m* и d_{pss} по формулам (1.1) и (1.2).

```
for n = 0: ...
m = ...;
     dpss(n+1) = ...;
end
```

5. Выполните аналогичный расчет для N_{ID}^{2}, равным 1 и 2, с использованием цикла for. Для этого имеющийся цикл расчета *m* и d_{pss} внесите в ещё один внешний цикл, а в качестве итерационной переменной используйте значение N_{ID}^{2}, изменяющийся в пределах от 0 до 2.

```
for Nid2 = 0: ...
  for n = 0: ...
    m = mod((…*Nid2), ...);
    dpss(Nid2+1,n+1) = ...;
  end
end
```

6. Аналогично сигналу PSS, самостоятельно сформируйте 336 сигналов вторичной синхронизации по формулам (1.7) – (1.12). При расчёте m_0 воспользуйтесь функцией fix() при делении для того чтобы округлить с недостатком полученные дробные значения, тем самым исключить дробную адресацию в массиве x_0.

```
x0(1:7) = [ ... ];
```

```
x1(1:7) = [ ... ];

for i = 1 : ...
  x0(i+7) = mod((x0(i+4) + x0(i)),2);
  x1(i+7) = mod((x1(i+1) + x1(i)),2);
end

for Nid1 = 0: ...
 m0 = ...;
 m1 = ...;

 for n = 0: ...
    d1(Nid1 + 1,n+1) = ...;
    d2(Nid1 + 1,n+1) = ...;
 end

  dsss =  d1.*d2;
   end
```

7. Создайте пустой двумерный массив размерностью 240 поднесущих на 4 OFDM символа.

```
ssblock = zeros([... ...]);
```

8. Расположите в массиве сигналы PSS и SSS согласно таблице 1.1.

```
ssblock(...,1) = 1*dpss(...,:);
ssblock(...,3) = 2*dsss(...,:);
```

9. Выведите график с сигналами, используя функцию imagesc(). График должен соответствовать рисунку 2.1.

```
figure
imagesc(abs(...))

caxis([0 4]);
axis xy;
```

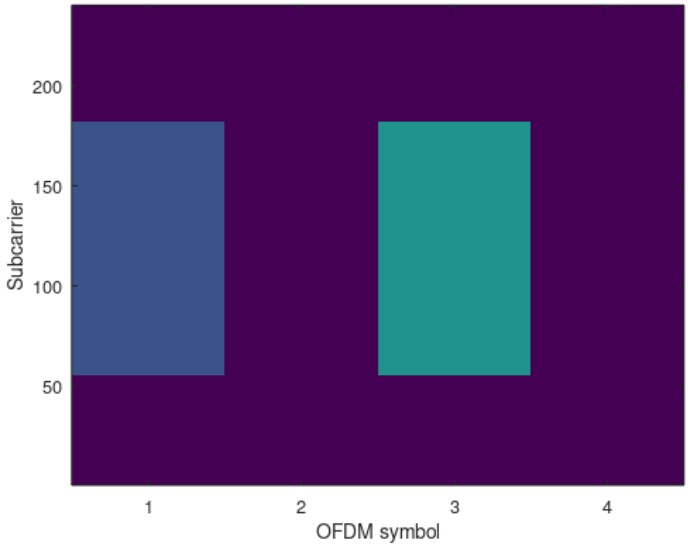


Рисунок 2.1 – Сигналы PSS и SSS

В ходе работы должны быть выполнены следующие этапы:

• Сформированы сигналы PSS и SSS.

• Расположить сигналы синхронизации в частотной области каждого OFDM символа, согласно частотно-временному распределению.

3. Контрольные вопросы к работе

1. Для чего используется первичный сигнал синхронизации?
2. Для чего используется вторичный сигнал синхронизации?
3. Сколько поднесущих занимает один SS/PBCH блок?

Работа № 13

«Формирование OFDM символов в соответствии со стандартом 5G NR, формирование физического широковещательного канала PBCH»

Цель работы: сформировать первичный (PSS) и вторичный (SSS) сигналы синхронизации, широковещательный физический канал (PBCH) и опорный сигнал демодуляции (DMRS), с помощью которых сгенерировать блок синхронизации (SS/PBCH).

Задачи лабораторной работы:

- Сформировать последовательность PBCH.
- Сформировать синхроблок.

В настоящей лабораторной работе принят следующий перечень сокращений:

DMRS – Demodulation Reference Signal, опорный сигнал демодуляции;

MIB – Master Information Block, основной информационный блок;

PBCH – Physical Broadcast Channel, физический канал передачи широковещательной информации;

PPS – Primary Synchronization Signal, первичный сигнал синхронизации;

SSS – Secondary Synchronization Signal, вторичный сигнал синхронизации.

1. Теоретический материал

PBCH используется для трансляции блока главной информации (MIB). Для передачи PBCH используются поднесущие с 0 по 239 первого и третьего символов, а также поднесущие с 0 по 47 и с 192 по 239 второго символа, как показано на рисунке 1.1.

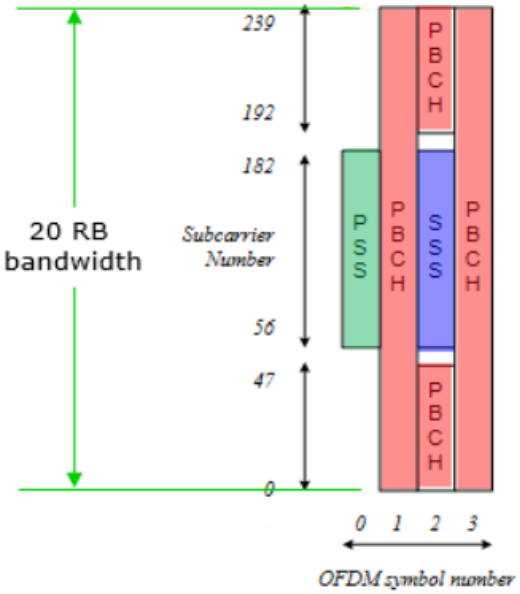


Рисунок 1.1 – Вид синхроблока

В таблице 1.1 представлено частотно-временное расположение всех сигналов по поднесущим в одном синхроблоке.

Таблица 1.1 – Частотно-временная позиция

Канал или сигнал	Номер символа OFDM "*l*" относительно начала блока SS/PBCH	Номер поднесущей "*k*" относительно начала блока SS/PBCH
PSS	0	56, 57, … ,182

Окончание таблицы 1.1

SSS	2	56, 57, … ,182
PBCH	1 и 3	0, 1, …, 239
	2	0, 1, …,47, 192, 193, …, 239
DMRS для PBCH	1 и 3	$0+v$, $4+v$, $8+v$, …, $236+v$
	2	$0+v$, $4+v$, $8+v$, …, $44+v$ $192+v$, $196+v$, …, $236+v$
$v = N_{ID}^{Cell} \ mod 4$		

Расположение PBCH DMRS зависит от v, где N_{ID}^{Cell} соответствующий физический идентификатор соты.

2. Ход работы

В данной работе требуются последовательности PSS и SSS полученные в ходе выполнения лабораторной работы «Формирование сигналов первичной и вторичной синхронизации стандарта 5G NR».

1) Создайте новый проект в Octave. Скопируйте все файлы из папки "LAB FILES" с представленными функциями на рисунке 2.1 в папку проекта.

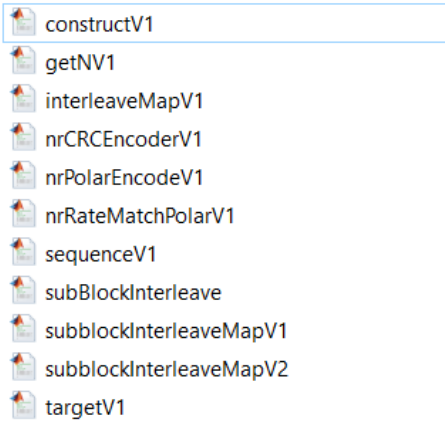


Рисунок 2.1 – Список необходимых файлов

Запишите следующие команды для очистки командного окна, области переменных и закрытии графических окон:

```
close all
clear all
clc
```

Задайте переменные (Nid1, Nid2, ibar_SSB) идентификаторы равными 0.
Nid2 – идентификатор последовательности PSS.
Nid1 – идентификатор последовательности SSS.
ibar_SSB – идентификатор SSB блока.

В ходе работы они могут изменятся в следующий пределах:

– Nid1 от 0 до 355;

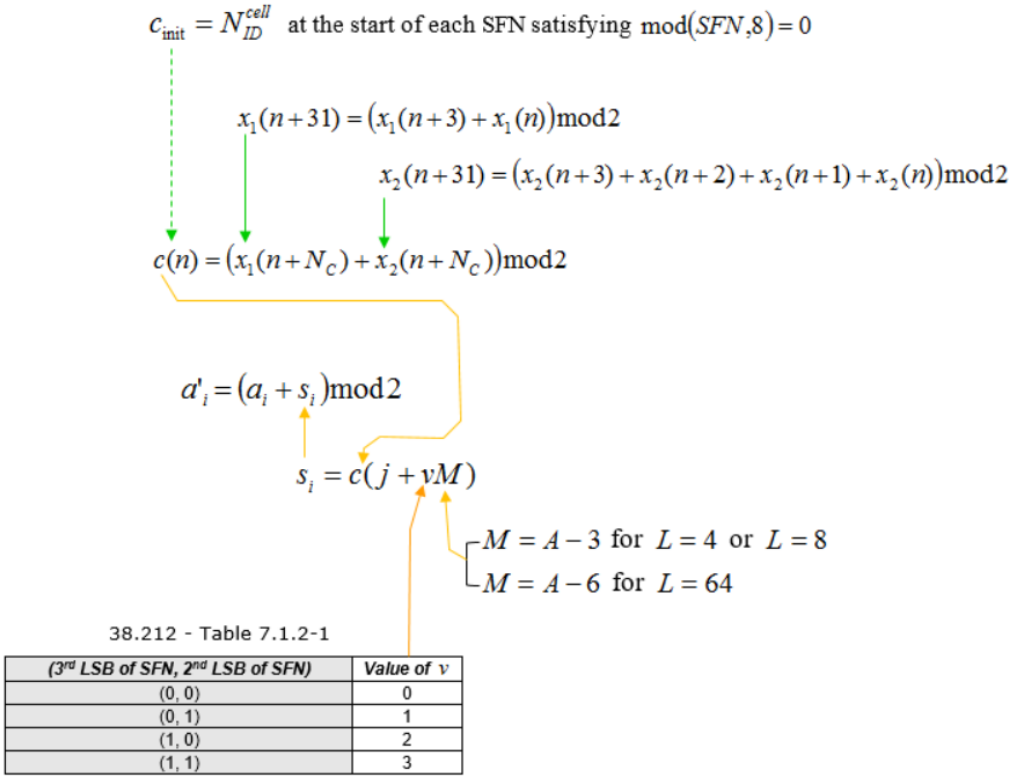
– Nid2 от 0 до 2;

– ibar_SSB от 0 до 7.

Вычислите cinit как сумму трех Nid1 и одного Nid2.

2) Следующим этапом является реализация алгоритма скремблирования, приведенного на рисунке 2.2.

Скремблирование – шифрование потока данных, в результате которой он выглядит, как поток случайных бит.



38.212 - Table 7.1.2-1

(3rd LSB of SFN, 2nd LSB of SFN)	Value of v
(0, 0)	0
(0, 1)	1
(1, 0)	2
(1, 1)	3

Рисунок 2.2 – Алгоритм скремблирования

N_{ID}^{Cell} – идентификационный номер ячейки физического уровня;

N_C – константа равная 1600 согласно стандарту TS 38.211 Section 5.2.1;

L – количество синхроблоков в полукадре (для нашей работы возьмем $L = 64$);

v – фаза последовательности скремблирования, задается как остаток от деления индекса синхроблока ibar_SSB на 4, если L равно 4 или 8, или остаток от деления на 8, если L равно 64

Задайте A равным 140. Затем сгенерируйте случайную последовательность из нулей и единиц pbch_bits длинной 864 бита.

Затем создайте нулевую последовательность x1 длинной 31 бит и задайте первый бит 1. Это инициализирующая последовательность для получения полной последовательности x1. Для создания аналогичной последовательности для x2 создайте нулевую последовательность длинной 31 бит, затем переведите число cinit в двоичную форму с помощью функции de2bi() и присвойте его младшим битам последовательности x2(1: length(de2bi(cinit)).

```
x2 = zeros(1,31);
x2(1:length(de2bi(cinit))) = de2bi(cinit);
```

Согласно рисунку (2.2) последовательность c(n) формируется из бит последовательностей x1 и x2**,** начиная с номера *Nc*, следовательно, длинна последовательностей *Nc* должна равняться (количество “защитных” бит) + *Mnp* (требуемая длинна “c(n)”). Для этого в соответствии со стандартом задайте *Nc*, равным 1600, а *Mnp*, равным 1000.

$$x_1(n+31) = (x_1(n+3) + x_1(n)) \bmod 2 . \quad (2.1)$$

$$x_2(n+31) = (x_2(n+3) + x_2(n+2) + x_2(n+1) + x_2(n)) \bmod 2 . \quad (2.2)$$

Создайте цикл, где *n* меняется от 1 до ($Nc + Mnp$). В этом цикле произведите вычисление требуемых последовательностей x1 и x2 согласно формулам (2.1) и (2.2).

Затем создайте следующий цикл, где *n* меняется от 1 до *Mnp* и произведите вычисление последовательности c(n) согласно формуле (2.3). Полученная последовательность, это последовательность, с которой мы будем скремблировать нашу основную последовательность pbch_bits.

$$c(n) = (x_1(n + Nc) + x_2(n + Nc)) \bmod 2 . \quad (2.3)$$

Задайте $M = A - 6$, а *v* как остаток от деления ibar_SSB на 8. Затем в цикле, где i меняется от 1 до *A*, вычислите *a(i*) равное остатку от деления на два от суммы pbch_bits(i) и *s(i)* согласно формуле (2.4).

$$\begin{gathered} a(i) = (pbch\_bit(i)) + s(i)) \bmod 2 , \\ s(i) = c(j + v \cdot M) , \end{gathered} \quad (2.4)$$

где *j* меняется также, как и *i*, следовательно, формулу (2.4) можно преобразовать в формулу:

$$a(i) = (pbch\_bit(i) + c(i + v \cdot M)) \bmod 2 . \quad (2.5)$$

3) Следующим этапом является проведение CRC кодирования, с помощью встроенной функции. Подайте на ее вход транспонированную последовательность a и номер требуемого полинома "24C", равный 4.

CRC кодирование – алгоритм кодирования, предназначенный для проверки целостности данных.

```
b = nrCRCEncoderV1(a',4);
```

Затем полученную последовательность подайте на полярный кодер. И следующим шагом в функцию выравнивания скоростей (RateMatch).

Полярное кодирование – линейное корректирующее кодирование, основанное на явлении поляризации канала.

Rate matching (RateMatch) – выравнивание скоростей для передачи определенного количества бит.

```
enc = nrPolarEncodeV1(b, 864);
encRM = nrRateMatchPolarV1(enc,56, 864);
```

4) Для модуляции с помощью функции reshape разбейте последовательность на массив размером 432 на 2. Затем переведите полученный массив в десятичные числа с помощью функции bi2de() и промодулируйте с помощью функции модуляции qammod().

```
bits = reshape(encRM,[ 864 /2,2]);
bitsD = bi2de(bits);
pbchSymbols = qammod(bitsD,4);
```

Мы получили отсчеты pbch канала, необходимые для формирования синхроблока.

5) Создайте нулевую матрицу ssblock размерностью 240 на 4, где 240 это номера поднесущих, а 4 это номера символов OFDM как показано на рисунке 1. Так как OCTAVE не может работать с нулевыми отсчетами то номера символов из 0-1-2-3 преобразуются в 1-2-3-4 и номера поднесущих изменятся с 0-239 на 1-240 что следует учитывать в дальнейшей работе. Сформируйте последовательности PSS и SSS аналогично первой лабораторной работе и расставьте их на места в ssblock определенные в таблице 1 и на рисунке 1.

При формировании PSS и SSS значения Nid1 и Nid2 должны были поменяться. Верните переменным Nid1 и Nid2 значения заданные в начале работы.

Так как опорный сигнал PBCH DRMS на данный момент отсутствует, заполните поднесущие, принадлежащие ему, значениями, равными единице. Структура синхроблока представлена на рисунке 2.3.

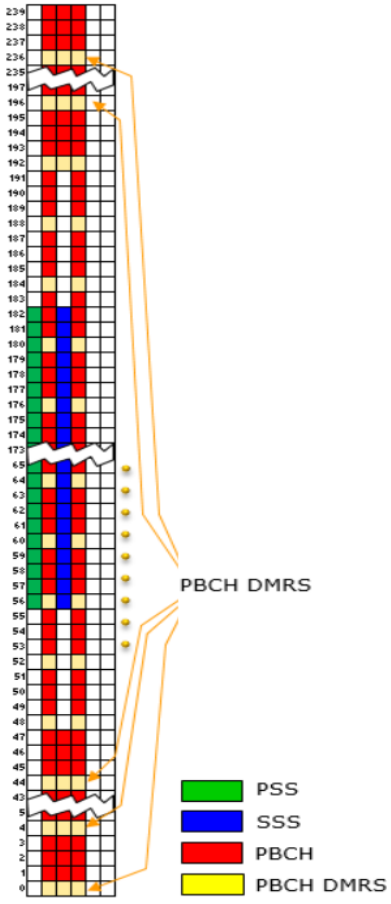


Рисунок 2.3 – Структура синхроблока

Согласно таблице 1.1, расположение DMRS поднесущих зависит от вводимого сдвига, который вычисляется как остаток от деления идентификационного номера ячейки физического уровня на 4. Вычислите NcellID как сумму трех Nid1 и одного Nid2. И затем найдите сдвиг v2**.**

Ниже приведен пример заполнение поднесущих PBCH DMRS.

```
NcellID = 3*Nid1+Nid2;
v2 = mod(NcellID,4);
for i = 1:4:240
        ssblock(v2+i,2) = -1;
        ssblock(v2+i,4) = -1;
end
```

```
i = 1;
while  i < 240
ssblock(v2+i,3) = -1;
if (i == 45)
  i = 193;
end
i = i + 4;
end
```

Постройте график синхроблока.

```
figure
imagesc(abs(ssblock))
```

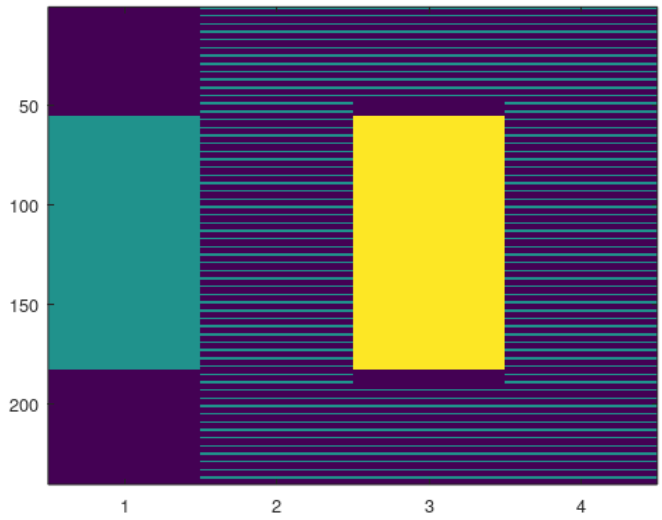


Рисунок 2.4 – График синхроблока

6) Распределите имеющиеся отсчеты pbchSymbols по поднесущим PBCH согласно таблице 1.1 и рисунку 1.1. Требуется помнить, что в поднесущих PBCH присутствуют поднесущие PBCH DMRS, которые в предыдущем пункте были заданы как -1. Учтите это при расстановке отсчетов pbchSymbols. После выполнения данной операции постройте график синхроблока.

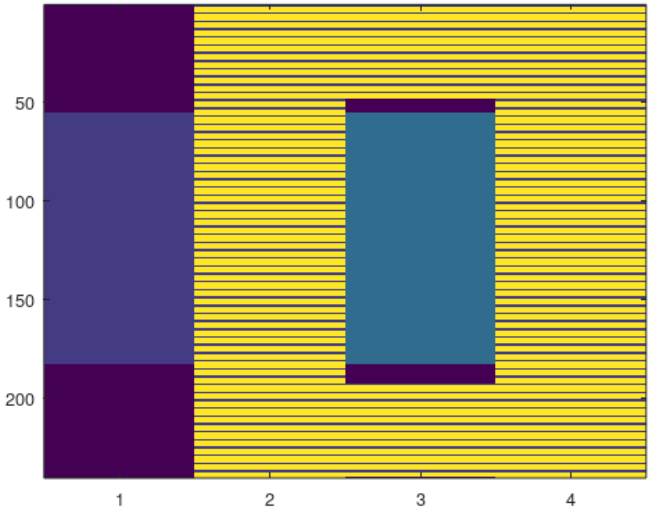


Рисунок 2.5 – График синхроблока

В результате работы будет получен график синхроблока содержащий PSS в первом символе, SSS в третьем, и канал BCH расположенный во втором, третьем и четвертом символах. Канал PBCH также содержит поднесущие со значением -1, в которые в будущем будет размещен опорный сигнал DMRS.

В ходе работы должны быть выполнены следующие операции:

- Вычисление инициирующей последовательности c_{init}.
- Вычисление последовательности скремблирования $c(n)$.
- Скремблирование данных.
- CRC кодирование, полярное кодирование и выравнивание скоростей с помощью функций.
- Расстановка сигналов PSS и SSS в синхроблоке.
- Расстановка сигнала BCH в синхроблоке.

3. Контрольные вопросы к работе

1. Что такое ibar_SSB и из каких идентификаторов он формируется?
2. Что такое скремблирование?
3. В каком символе SSB находится SSS сигнал?
4. Какой сигнал присутствует во втором символе SSB вместе с PBCH?

ЗАКЛЮЧЕНИЕ

Учебно-методическое пособие представляет собой ресурс, способствующий углубленному изучению основных аспектов цифровой обработки сигналов в контексте современных технологий мобильной связи и Интернета вещей.

Цель данного пособия заключается в обеспечении студентов не только теоретическими знаниями, но и практическими навыками, необходимыми для успешного применения этих знаний в реальных проектах.

В пособии освещены основные принципы формирования и обработки сигналов в системах мобильной связи и Интернета вещей. Кроме того, были представлены практические задания, направленные на закрепление полученных знаний и развитие навыков работы с современными инструментами моделирования, таких как MATLAB и Octave.

В процессе изучения данного пособия студенты получат не только теоретические знания, но и практические навыки, которые пригодятся им в будущей профессиональной деятельности.

СПИСОК ЛИТЕРАТУРЫ

1. Makers of MATLAB and Simulink - MATLAB & Simulink : сайт / mathworks. – URL: https://www.mathworks.com/. – Режим доступа: свободный (дата обращения 10.08.2023).

2. GNU Octave : сайт / octave. – URL: https://octave.org/ . – Режим доступа: свободный (дата обращения 10.08.2023).