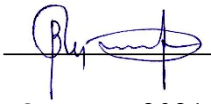
Министерство науки и высшего образования Российской Федерации
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТОМСКИЙ ГОСУДАРСТЕННЫЙ УНИВЕРСИТЕТ (НИ ТГУ)
Радиофизический факультет
Кафедра радиофизики

ДОПУСТИТЬ К ЗАЩИТЕ В ГЭК
Руководитель ООП
д-р физ.-мат. наук, проф.


_____ В.П. Гермогенов

18 июня 2021 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА

РАЗРАБОТКА ПРОГРАММНОЙ РЕАЛИЗАЦИИ МЕТОДОВ ЦИФРОВОЙ МОДУЛЯЦИИ В СИСТЕМАХ СВЯЗИ 5G NR

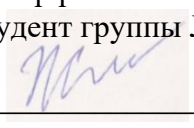
по направлению подготовки 03.03.03 Радиофизика
профиль «Радиофизика, электроника и информационные системы»

Жаринов Вячеслав Федорович

Руководитель ВКР
канд. физ.-мат. наук, доцент
_____ О.Г. Пономарев

17 июня 2021 г.

Автор работы
студент группы № 771


_____ В.Ф. Жаринов

17 июня 2021 г.

Томск – 2021

Министерство науки и высшего образования Российской Федерации.

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ (НИ ТГУ)
Радиофизический факультет
Кафедра радиофизики

ДОПУСТИТЬ К ЗАЩИТЕ В ГЭК
Руководитель ООП
д-р физ.-мат. наук, проф.
_____ В.П. Гермогенов

02 октября 2020 г.

ЗАДАНИЕ

на выполнение выпускной квалификационной работы бакалавра / специалиста /магистра обучающемуся
Жаринову Вячеславу Федоровичу
Фамилия Имя Отчество обучающегося
по направлению подготовки 03.03.03 радиофизика по основной образовательной программе подготовки бакалавров

1 Тема выпускной квалификационной работы
РАЗРАБОТКА ПРОГРАММНОЙ РЕАЛИЗАЦИИ МЕТОДОВ ЦИФРОВОЙ МОДУЛЯЦИИ В СИСТЕМАХ СВЯЗИ 5G NR

2 Срок сдачи обучающимся выполненной выпускной квалификационной работы:
а) в учебный офис / деканат – июнь 2021 б) в ГЭК – Июнь 2021

3 Исходные данные к работе:
Объект исследования – Методы цифровой модуляции в системах мобильной связи пятого поколения
Предмет исследования – Программная реализация алгоритмов цифровой модуляции
Цель исследования – Разработка комплекса алгоритмов и программ, реализующих методы формирования OFDM-сигналов в системах мобильной связи пятого поколения

Задачи:
1. Программная реализация OFDM-модулятора системы мобильной связи пятого поколения
2. Апробация разработанного комплекса программ в численном эксперименте
3. Экспериментальное подтверждение работоспособности разработанного модулятора с использованием программно-определяемого радио (SDR)

Методы исследования:
Численное моделирование
Экспериментальная апробация

Организация или отрасль, по тематике которой выполняется работа, –
по соглашению с Минобрнауки России от «26» ноября 2019 г. № 075-11-2019-031 (проект «Разработка программно-аппаратного комплекса для формирования тестовых сигналов стандарта 5G NR»).

4 Краткое содержание работы
1. Методы цифровой модуляции в системах мобильной связи 5G NR (лит. обзор)

2. Программная реализация алгоритмов первичной цифровой модуляции
3. Программная реализация алгоритмов OFDM-модулятора
4. Апробация разработанного комплекса программ в численном эксперименте
5. Экспериментальное подтверждение работоспособности разработанного модулятора с использованием программно-определяемого радио (SDR)

Руководитель выпускной квалификационной работы
Доцент каф. радиофизики, канд. физ.-мат. наук _____ / О.Г.Пономарев
должность, место работы *подпись* *И.О. Фамилия*

Задание принял к исполнению
Студент 771 группы РФФ ТГУ _____ / В.Ф.Жаринов

ОГЛАВЛЕНИЕ

АННОТАЦИЯ

Выпускная квалификационная работа бакалавра 62 стр., 3 главы, 33 рисунка, 4 приложения.

Ключевые слова: OFDM-МОДУЛЯЦИЯ, ЦИФРОВАЯ МОДУЛЯЦИЯ, 5G NEW RADIO, РЕСУРСНАЯ СЕТКА, РАЗНЕСЕНИЕ ПОДНЕСУЩИХ, ПРОГРАММНО-ОПРЕДЕЛЯЕМОЕ РАДИО.

Объектом исследования являются методы цифровой модуляции в системах мобильной связи пятого поколения.

Предметом исследования является программная реализация алгоритмов цифровой модуляции.

Целью работы является разработка комплекса алгоритмов и программ, реализующих методы формирования OFDM-сигналов в системах мобильной связи пятого поколения.

В результате работы были получены следующие основные результаты.

1) Освоены язык программирования C++, среда разработки Qt Creator, язык среды для научных расчетов Mathworks MatLab.

2) Разработана функция «bfOFDMModulator», программно реализующая алгоритм OFDM-модуляции в 5G NR. Отладка данной функции полностью подтвердила ее работоспособность.

3) Проведен численный эксперимент, подтверждающий работоспособность созданного OFDM-модулятора для различных наборов входных конфигурационных параметров, в том числе при их нестандартном задании.

4) Освоена работа программно-определяемого радио ADALM-PLUTO.

5) Проведен реальный эксперимент, подтверждающий работоспособность OFDM-модулятора при передаче и приеме сигнала с использованием программно-определяемого радио ADALM-PLUTO.

ВВЕДЕНИЕ

На протяжении XX-го столетия весь мир испытывал настоящий бум в области научно-технического прогресса. Об этом красноречиво свидетельствуют исследования в области ядерной физики, изобретение самолета, полет человека в космос и многое другое. Примерно это же время зародилась идея создания системы сотовой связи – сферы исследования настоящей работы.

Начало эры мобильной связи было ознаменовано появлением сетей первого поколения 1G (от англ. G – generation – «поколение») в 1980-х годах. Этот телекоммуникационный стандарт базировался на аналоговой передаче сигнала. Такая особенность накладывала серьезные ограничения как на скорость передачи данных, так и на функционал системы связи в целом. Поэтому в 2G, появившемся в начале 1990-х годов, применялась технология, совмещающая аналоговую и цифровую передачу информации, что позволило существенно увеличить скорость. Третье поколение связи, ставшее актуальным в начале XXI века, использовало уже пакетную передачу данных, сделавшую возможным выход в Интернет. Вошедшее в нашу жизнь с 2010 года 4 поколение мобильной связи характеризовалось еще большей скоростью передачи данных, открывшей перед пользователем такие опции, как, например, видеосвязь. Однако и технологии, применяемые в 4G, имеют свой потолок, который с годами прорисовывается все более явственно, поскольку мир не стоит на месте. Всем известная технология Wi-Fi-сетей (от англ. Wireless Fidelity – «беспроводная точность») тоже имеет свои ограничения. В связи с этим в последние годы все чаще обсуждается тема создания следующего поколения мобильной связи. 5G – это новейший телекоммуникационный стандарт связи, предполагающий принципиально новый уровень сервиса и опций клиентов. Высокие скорости (до 20 Гбит/с), возможность огромного количества подключений на единицу площади, быстрый отклик и низкая задержка, а также хорошее покрытие внутри зданий

– вот лишь немногие преимущества 5G. Все это сделает возможным функционирование как человеко-ориентированных, так и машинно-ориентированных приложений. К примеру, с появлением пятого поколения мобильной связи станет возможным бурное развитие интернета вещей IoT (от англ. Internet of Things – интернет вещей) и такого перспективного направления, как телемедицина, находящихся сейчас в лишь зародыше; на новый уровень смогут выйти технологии виртуальной и дополненной реальности. 5 поколение мобильной связи уже тестировалось в ряде таких высокоразвитых стран, как Япония, Китай и Южная Корея.

Разработкой стандартов системы связи 5G New Radio занимается консорциум 3GPP (от англ. 3rd Generation Partnership Project), в который входят крупнейшие зарубежные компании, такие как Huawei и Nokia [1]. Базируясь на этих стандартах, они создают программное обеспечение системы связи пятого поколения. В России на данный момент программного обеспечения такой системы мобильной связи нет, поэтому его разработка является крайне важной задачей. В подтверждение этого следует упомянуть, что в ноябре 2019 года кафедра радиофизики радиофизического факультета выиграла грант Правительства РФ «Разработка программно-аппаратного комплекса для формирования тестовых сигналов стандарта 5G NR», соисполнителем которого я являюсь. Моей задачей является разработка OFDM-модулятора, являющегося одной из важнейших составляющих всей системы связи. А это значит, что актуальность данной работы обеспечивается в том числе и заинтересованностью в ней Правительства Российской Федерации.

Концептуальным отличием 5G от предыдущего поколения связи является использование волн миллиметрового диапазона, что соответствует частотному ресурсу до 100 ГГц. Однако такое новшество накладывает и более высокие требования к системе связи. Для того, чтобы им удовлетворять, система связи должна быть очень гибкой. Поэтому стандарт

5G New Radio предусматривает применение всевозможной оптимизации энергозатрат (например, в 5G расстояние между поднесущими частотами не является постоянным). К числу отличительных черт 5G также можно отнести и использование больших массивов антенн. Также не обошлось и без внедрения иных ранее не задействованных технологий.

В данной работе функционирование системы связи рассматривается в рамках физического уровня, включающего в себя главным образом процедуры кодирования-декодирования и модуляции-демодуляции. Настоящая работа посвящена OFDM-модуляции сигналов.

Для того, чтобы наглядно продемонстрировать рассматриваемый в данной работе участок из всего многообразия процедур 5G New Radio, обратимся к структуре передатчика, приведенной на рисунке 1:

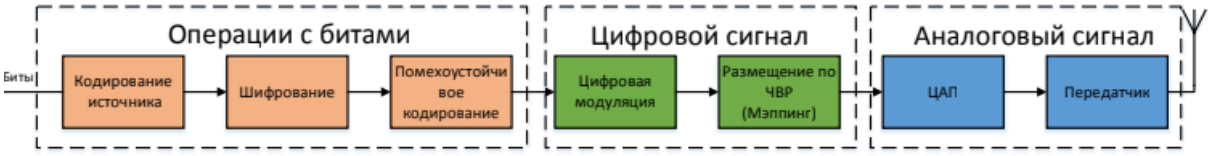


Рисунок 1 – Структура передатчика

На данном рисунке можно наблюдать весь путь, проходимый информацией от битового потока до излучения с антенны. Он заключается в следующих преобразованиях: на начальном этапе информация в виде битового потока подвергается кодированию. Затем она все еще в битовом виде поступает на вход блока, отвечающего за цифровую модуляцию сигнала. Результатом правильной работы этого сегмента является уже 2-мерный массив данных, который далее подается на ЦАП (цифро-аналоговый преобразователь). На заключительном этапе сигнал излучается антенной. Проведенный анализ позволяет сформулировать цель и задачи данной работы.

Цель

Разработка комплекса алгоритмов и программ, реализующих методы формирования OFDM-сигналов в системах мобильной связи пятого поколения.

Задачи

1. Программная реализация OFDM-модулятора системы мобильной связи пятого поколения.

2. Апробация разработанного комплекса программ в численном эксперименте.

3. Экспериментальное подтверждение работоспособности разработанного модулятора с использованием программно-определяемого радио (SDR).

Основная часть данной работы состоит из трех глав. В первой части первой главы рассматривается теория метода, во второй и третьей частях приводятся специфика формирования сигнала в 5G NR и вытекающие особенности реализации соответственно, в четвертой описывается принцип работы программы. Вторая глава работы посвящена описанию численного эксперимента по передаче и приему сигнала, сформированного с применением разработанного OFDM-модулятора. В третьей главе описывается практический эксперимент с использованием программно-определяемого радио (SDR).

1 OFDM-модуляция в 5G New Radio

1.1 Теоретическая часть

Разработанный ещё в 60-х годах прошлого века метод OFDM-модуляции (от англ. Orthogonal Frequency-Division Multiplexing – ортогональное частотное разделение каналов с мультиплексированием) изначально не использовался повсеместно, а применялся лишь в военных разработках ввиду сложностей в технической реализации. Преодолеть их удалось сравнительно недавно с появлением быстрых вычислительных систем. В настоящее время метод OFDM-модуляции активно используется в области цифровых сигналов.

Сам метод OFDM-модуляции представляет собой ортогональное частотное разделение каналов с мультиплексированием [2]. Мультиплексирование означает разделение канала на несколько субканалов или поднесущих частот, передающихся параллельно, а ортогональность обеспечивает равенство 0 взаимной энергии любых двух поднесущих из всего набора.

Это важное свойство позволяет осуществлять передачу на большом числе поднесущих в ограниченном частотном интервале без взаимного искажения. Спектр такого сигнала будет выглядеть следующим образом:

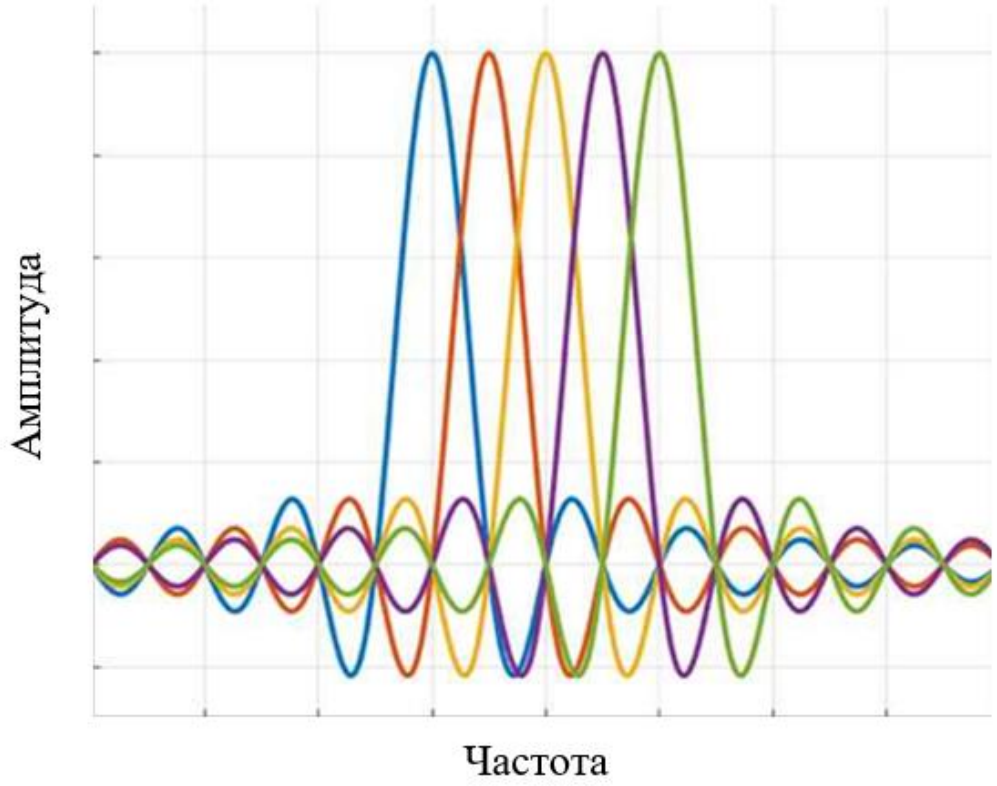


Рисунок 2 – Спектр OFDM-сигнала

При программной реализации метода OFDM-модуляции обращаются к его математической модели – обратному дискретному преобразованию Фурье:

$$x_n = \frac{1}{N}\sum_{k=0}^{N-1} X_k e^{\frac{2\pi i}{N}kn},$$

где N – количество дискретных точек (отсчетов) в одном периоде,

$n=0, \ldots, N-1$,

X_k – спектральные отсчеты сигнала,

x_n – значения сигнала в дискретные моменты времени.

На основании вышесказанного можно рассмотреть принципиальную схему OFDM-модулятора:

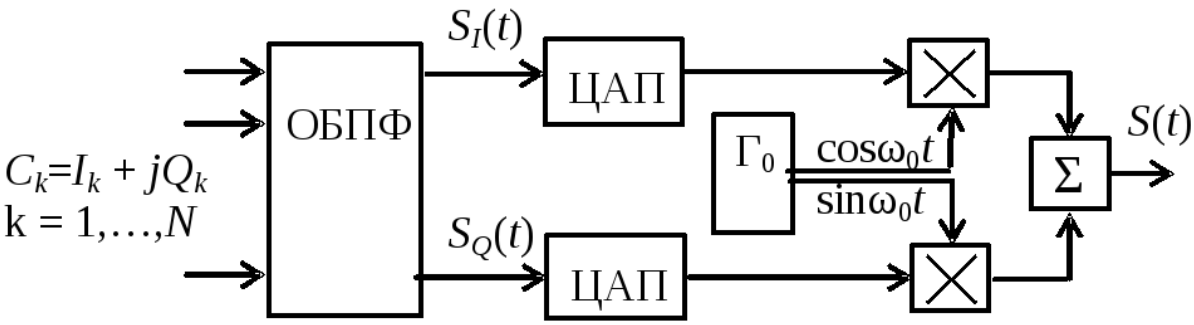


Рисунок 3 – Принципиальная схема OFDM-модулятора

Как это показано на 3 рисунке, распараллеленный поток комплексных символов подвергается обратному преобразованию Фурье, которое реализуется при использовании блока ОБПФ (обратное быстрое преобразование Фурье). Затем реальная часть преобразования Фурье модулируется колебанием генератора, а мнимая – колебанием генератора со сдвигом фазы на 90˚. Сумма двух перемноженных сигналов поступает на антенну [3], [4].

При распространении в реальном мире в результате многократных отражений сигнала от объектов, может сказываться влияние такой помехи, как межсимвольная интерференция. Для того, чтобы препятствовать этому эффекту, технология OFDM-модуляции предусматривает добавление циклического префикса (защитного интервала). Он представляет собой копию конца OFDM-символа (эталонной единицы данных), помещенную в начало:

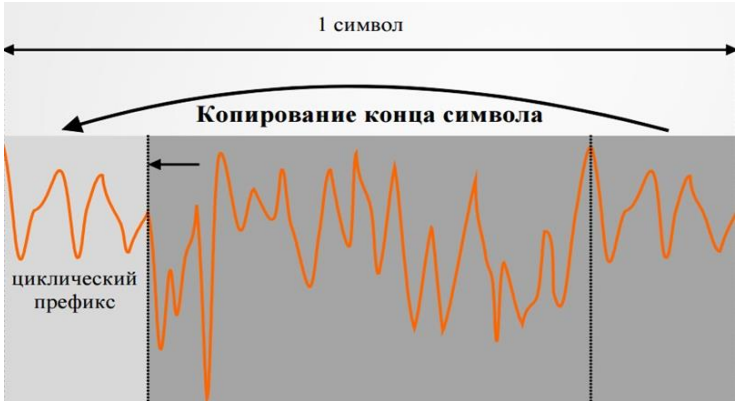


Рисунок 4 – Циклический префикс

Добавление защитного интервала допустимо потому, что поток данных делится между поднесущими, скорость сигнала на каждой из которых меньше первоначальной скорости до распараллеливания; благодаря этому ортогональность частот сохраняется.

На рисунке 5 демонстрируется, как использование ЗИ (защитного интервала) позволяет бороться с межсимвольной интерференцией:

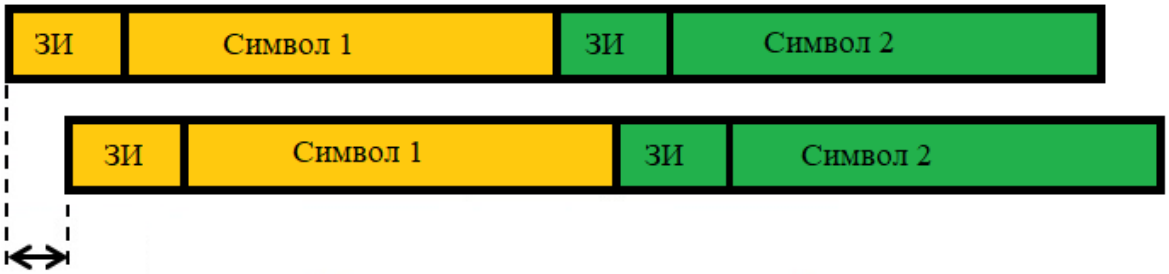


Рисунок 5 – Противостояние циклического префикса межсимвольной интерференции

Здесь можно видеть, что если разница прихода двух лучей не превышает длину циклического префикса, то полезная информация не повреждается [5], [6].

Другой вид помех, с которыми также хорошо справляется OFDM – узкополосные помехи. Их воздействие на сигнал подавляет малую часть поднесущих, однако впоследствии переносимая ими информация восстанавливается благодаря корректирующим кодам. Остальные субканалы при этом доходят до адресата неповрежденными.

В методе OFDM-модуляции есть возможность осуществления фазовой и частотной синхронизации. Данная идея осуществляется путем использования пилот-сигналов, при создании которых поднесущие не модулируются. Эта, а также многие другие возможности говорят о большой гибкости технологии OFDM-модуляции. Однако в случае использования пилотных символов заметно увеличивается энергопотребление реального

устройства (OFDM-модулятора). Во многом благодаря этому пилотные символы не включены в стандарт 5G NR.

1.2 Формирование сигнала в 5G NR

В 5G передаваемая информация представляет собой трехмерный массив, измерениями которого являются частота, время и номер антенного порта. Последнее объясняется тем, что для приема и передачи данных стандарт New Radio предполагает использование системы из нескольких антенн MIMO (от англ. Multiple Input/Multiple Output – множественный вход/множественный выход). На рисунке 6(а) представлена подобная антенна, рисунок 6(б) иллюстрирует принцип ее работы, заключающийся в пространственном разделении потоков.

Рисунок 6(а) – Антенна massive MIMO

Рисунок 6(б) – Принцип работы

Иными словами, такой массив можно рассматривать как набор двумерных массивов количества антенных портов с ячейками по частоте и времени. Тогда каждый массив будет представлять собой частотно-временную ресурсную сетку следующего вида:

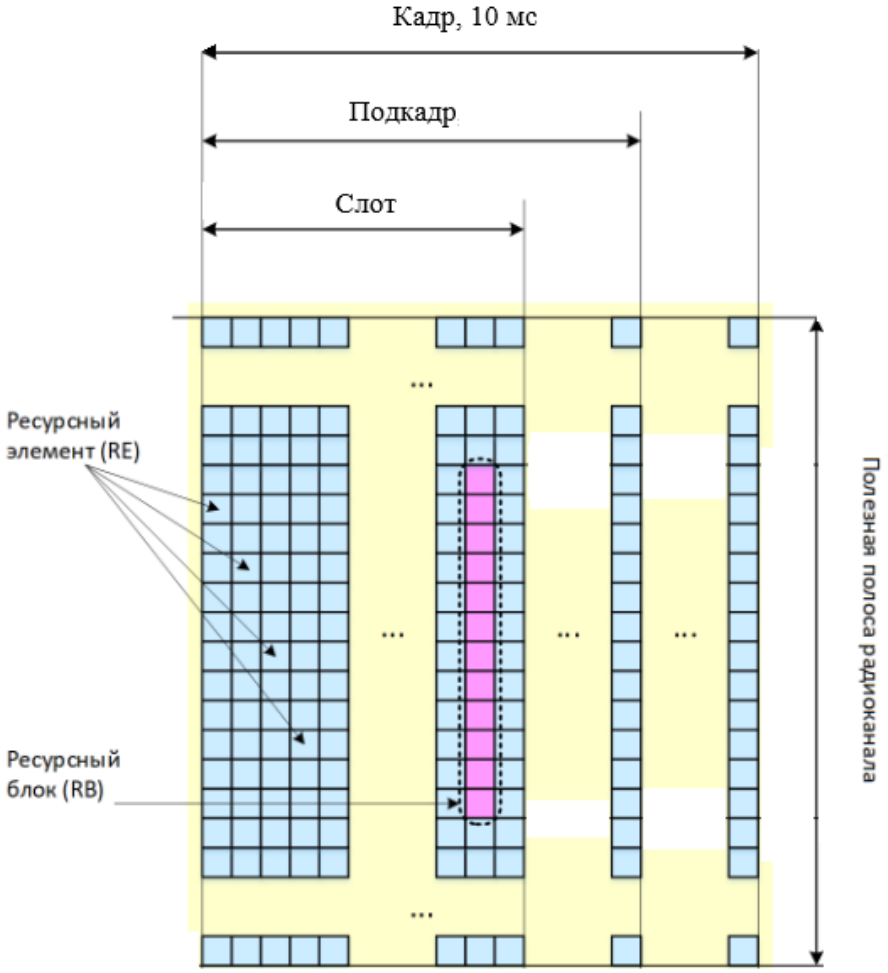


Рисунок 7 – Ресурсная сетка

Во временной области эталонной единицей являются символы. 14 или 12 таких символов (в зависимости от того, нормальным или расширенным является циклический префикс) образуют слот. Один или несколько слотов составляют подкадр длительностью в 1 миллисекунду. Подкадры, в свою очередь, складываются в кадры, длительностью по 10 миллисекунд [7]. В частотной же области каждые 12 последовательных ячеек для фиксированного OFDM-символа образуют ресурсный блок, состоящий из 12 ресурсных элементов – наименьших единиц частотно-временного распределения [8].

1.3 Особенности практической реализации OFDM в 5G NR

Для того, чтобы получить вышеописанное распределение на выходе OFDM-модулятора, на его вход нужно подать следующие параметры:

1) Разнесение поднесущих Δf

Одной из отличительных особенностей стандарта сотовой связи 5 поколения является переменный интервал разнесения поднесущих. Это означает, что расстояние между поднесущими частотами может варьироваться в зависимости от задаваемых условий в соответствии с выражением:

$$\Delta f = 15 \times 2^{\mu} \text{ кГц},$$

где μ принимает одно из значений: 0, 1, 2, 3, 4. Таким образом, расстояние между поднесущими может быть равно 15, 30, 60, 120 или 240 кГц.

2) Длина циклического префикса

Существует 2 варианта циклического префикса: нормальный и расширенный. В каждом случае длительность циклического префикса различна. Следовательно, различными будут и длительности, выделяемые под полезную информацию.

3) Количество ресурсных блоков NRB – размер полосы пропускания.

4) Ресурсная сетка

Таким образом, для того, чтобы выполнить OFDM-модуляцию для заданной ресурсной сетки, должна быть известна в первую очередь она сама, а также разнесение поднесущих, циклический префикс и количество ресурсных блоков.

В этом случае результатом OFDM-модуляции будут модуляционные символы, являющиеся результатом суммирования по всем несущим частотам, как это показано на следующем рисунке:

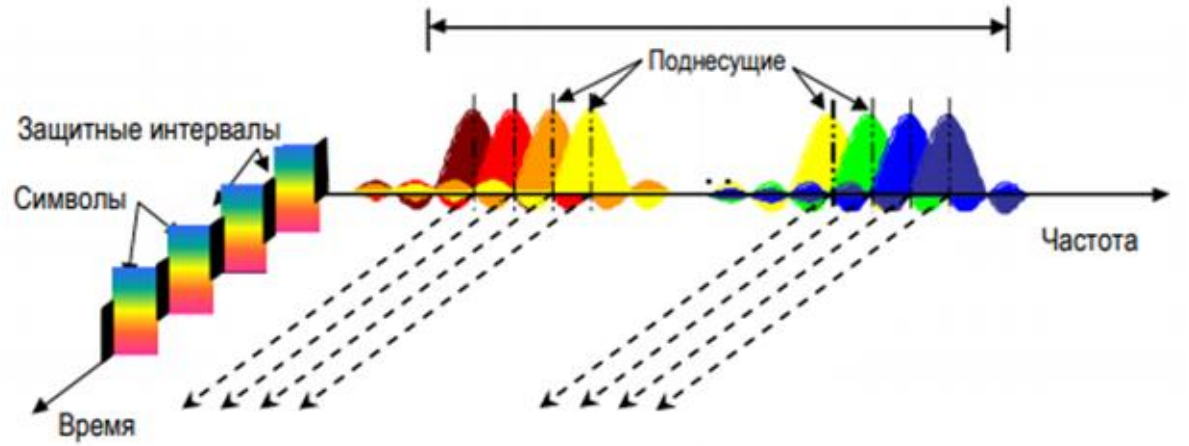


Рисунок 8 – Символы OFDM-модуляции

1.4 Описание работы программы

Для написания программы был создан проект на языке программирования C++ в среде Qt Creator. Такой выбор обусловлен дальнейшей возможностью создания консольного приложения для компьютера.

Проект состоит из самой функции OFDM-модуляции bfOFDMModulator, возвращающей двумерный массив waveform, а также из функции OFDMInfo, возвращающей структуру с полями, необходимыми для выполнения функции bfOFDMModulator:

1) Windowing - количество выборок во временной области, к которым применяется оконное отображение и перекрытие символов OFDM;
2) SamplingRate - частота дискретизации OFDM модулятора;
3) NFFT - количество точек в быстром преобразовании Фурье, используемом в OFDM-модуляции;

4) CyclicPrefixLengths - длина циклического префикса (в выборках) каждого OFDM символа в подкадре;

5) SymbolsPerSlot - количество символов в слоте;

6) NSubcarriers - количество поднесущих;

7) SymbolsPerSubframe - количество символов в подкадре.

Полный текст программы приводится в приложении А.

Проверка правильности работы функции производилась путем запуска проекта в Qt Creator и сравнения выходных значений с выходными значениями ранее написанной программы в среде Mathworks Matlab. Таким образом, была установлена полная работоспособность программы.

Данная функция вошла в состав программы для ЭВМ «Формирование синхросигналов мобильной связи стандарта 5G NR», предназначенной для генерации сигналов в нисходящем канале системы мобильной радиосвязи стандарта 5G NR, свидетельство о регистрации №2020665979, заявка № 2020664689/69, как одна из важнейших процедур стандарта 5G NR. Ниже представлен графический интерфейс этого консольного приложения (параметры PDSCH), а также графики сетки ресурсов и выходного сигнала:

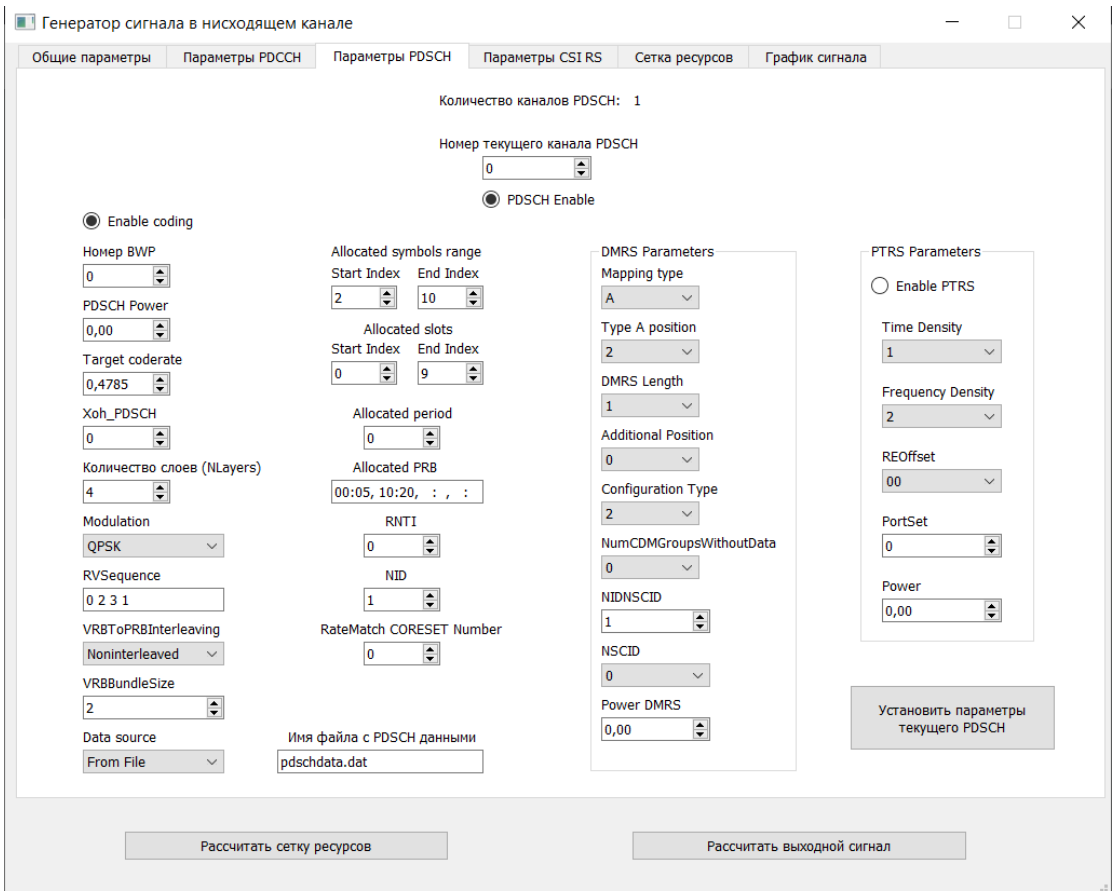


Рисунок 9 – Графический интерфейс приложения, вкладка «Параметры PDSCH»

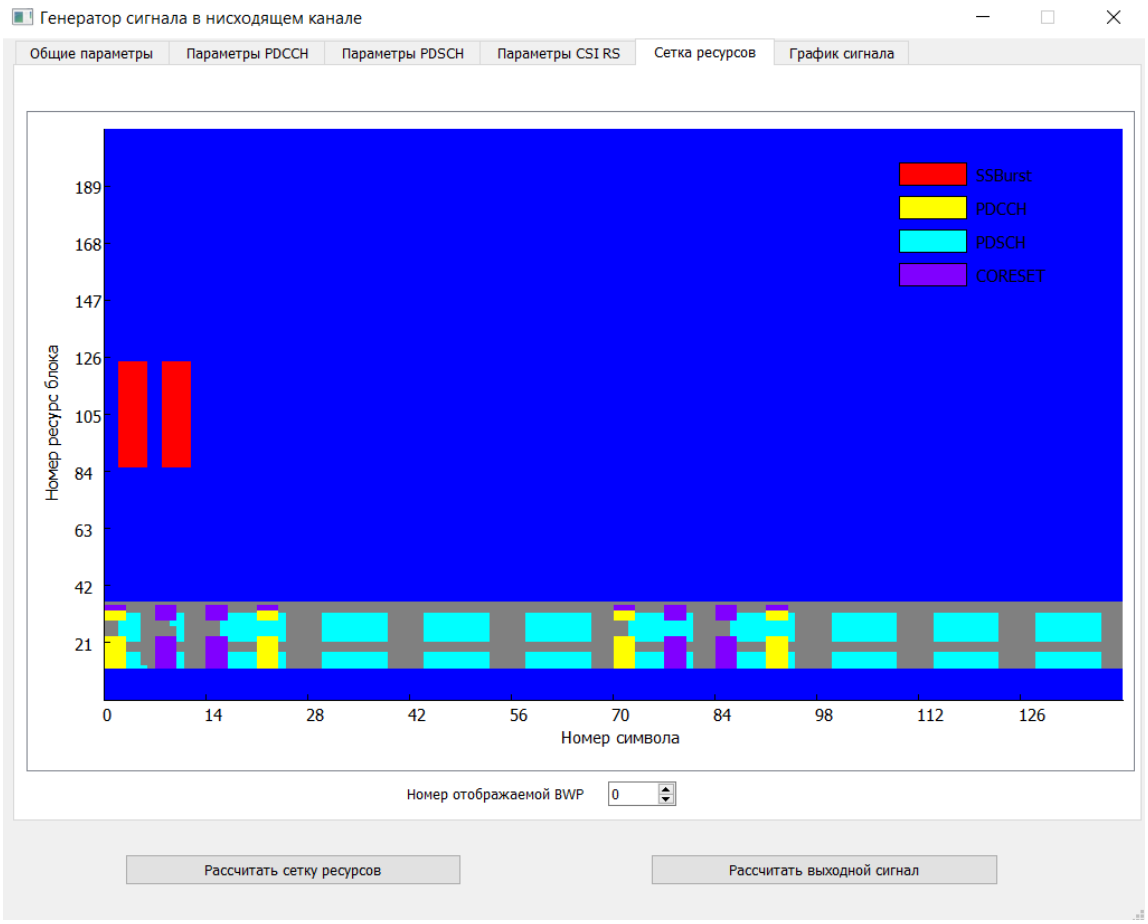


Рисунок 10 – Рассчитанная сетка ресурсов

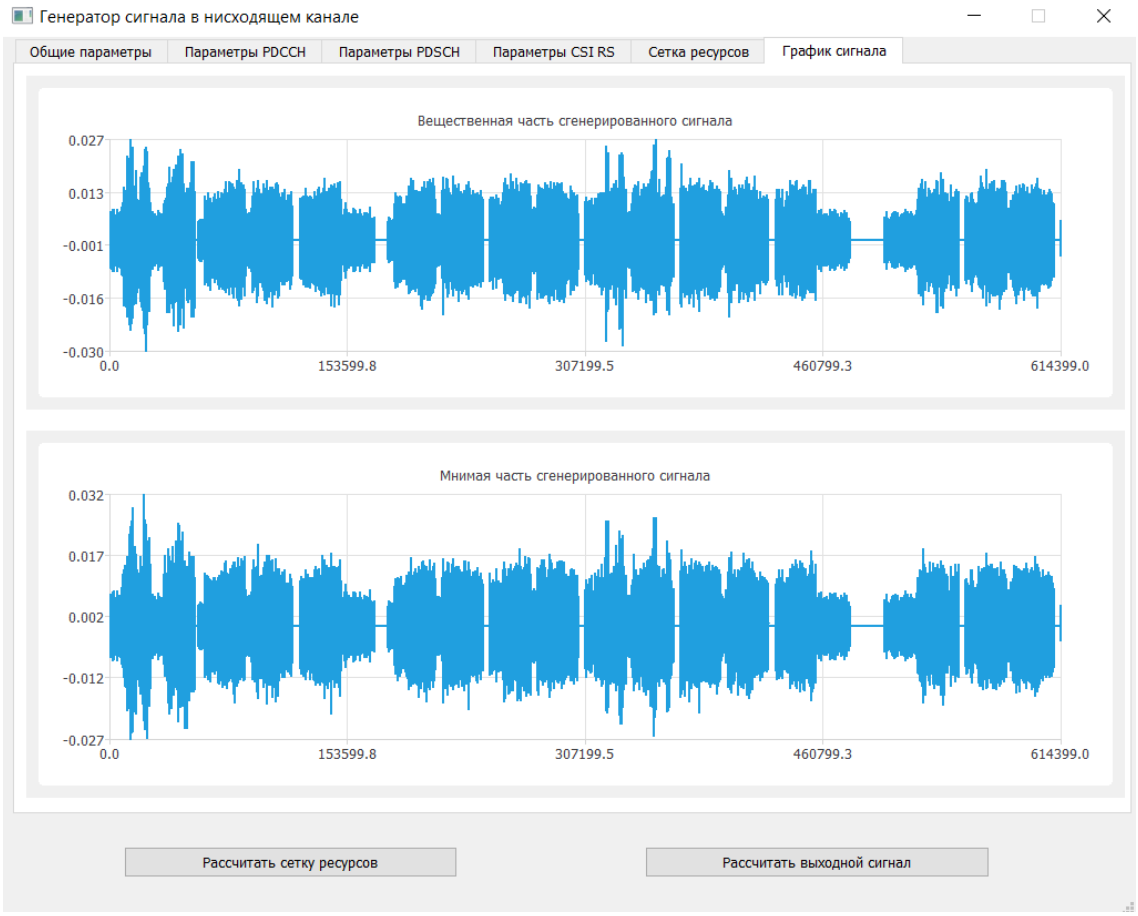


Рисунок 11 – Выходной сигнал

2 Численный эксперимент

Чтобы подтвердить работоспособность созданного OFDM-модулятора, был проведен численный эксперимент, суть которого заключалась в реализации передачи и приема PDSCH-сигнала (от англ. Physical Downlink Shared Channel – общий физический канал нисходящей линии связи, предназначенный для передачи пользовательской информации), сформированного при помощи данного метода. Для этой цели использовался пример NR PDSCH Throughput (от англ. Throughput – пропускная способность) из расширения 5G NR Toolbox среды для научных расчетов Mathworks MatLab. В этом примере реализованы передающий и приемный алгоритмы PDSCH-сигнала.

Модель передатчика PDSCH-сигнала включает в себя следующую последовательность основных процедур (рисунок 12):

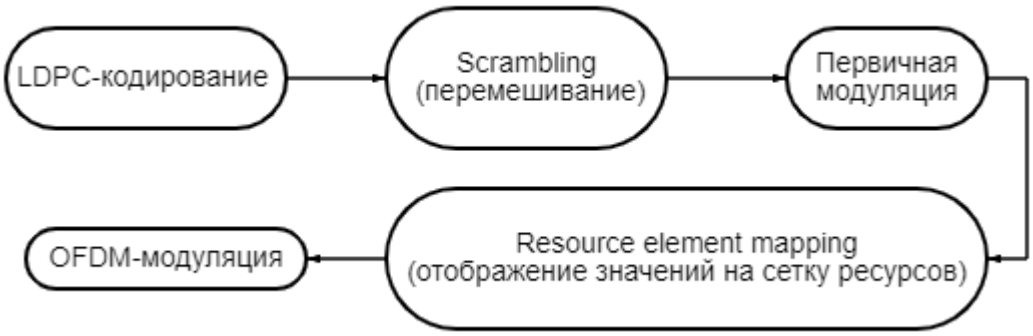


Рисунок 12 – Основные процедуры передатчика PDSCH-сигнала

1) LDPC-кодирование (от англ. low-density parity-check – кодирование с малой плотностью проверок на чётность) – один из основных видов кодирования в стандарте 5G NR.

2) Scrambling (от англ. перемешивание), необходимо для достижения равной плотности ошибок во всех блоках битов. Операция является обязательной для правильной работы помехоустойчивых кодов, так как при декодировании они могут безошибочно восстанавливать информацию, число ошибок которой не выше некоторого порогового значения.

3) Первичная модуляция согласно одной из схем: QPSK (англ. quadrature phase shift keying – квадратурная фазовая манипуляция), 16QAM, 64QAM, 256QAM (англ. quadrature amplitude modulation – квадратурная амплитудная модуляция). В данном случае использовалась схема QPSK-модуляции, как наиболее надежный вариант для предстоящего

практического эксперимента. Она предполагает для каждой пары последовательных бит $b(2n)$, $b(2n+1)$ на входе модулятора отображение в комплексное число $d(n)$, определяемое выражением:

$$d(n) = \frac{1}{\sqrt{2}}[(1-2b(2n)) + j(1-2b(2n+1))]$$

4) Resource element mapping – отображение полученных ранее комплексных значений на сетку ресурсов.

5) OFDM-модуляция, результатом которой является сформированный сигнал txWaveform стандарта 5G NR.

В приемнике выполняется вся последовательность операций, обратных к перечисленным выше, в инверсном порядке: OFDM-демодуляция, Resource element demapping, первичная демодуляция, descrambling и декодирование.

Исходными конфигурационными параметрами были разнесение поднесущих Δf = 30 кГц и количество ресурсных блоков NRB = 51. При заданных таким образом параметрах была сформирована сетка ресурсов Resource Grid, а затем, применив метод OFDM-модуляции, мы перешли от ресурсной сетки к комплексно-значным OFDM-символам. На следующих рисунках приведены временные зависимости действительной и мнимой частей сгенерированного сигнала txWaveform для 1 подкадра, состоящего из 14 OFDM-символов для параметра разнесения поднесущих μ = 1 и NRB = 51:

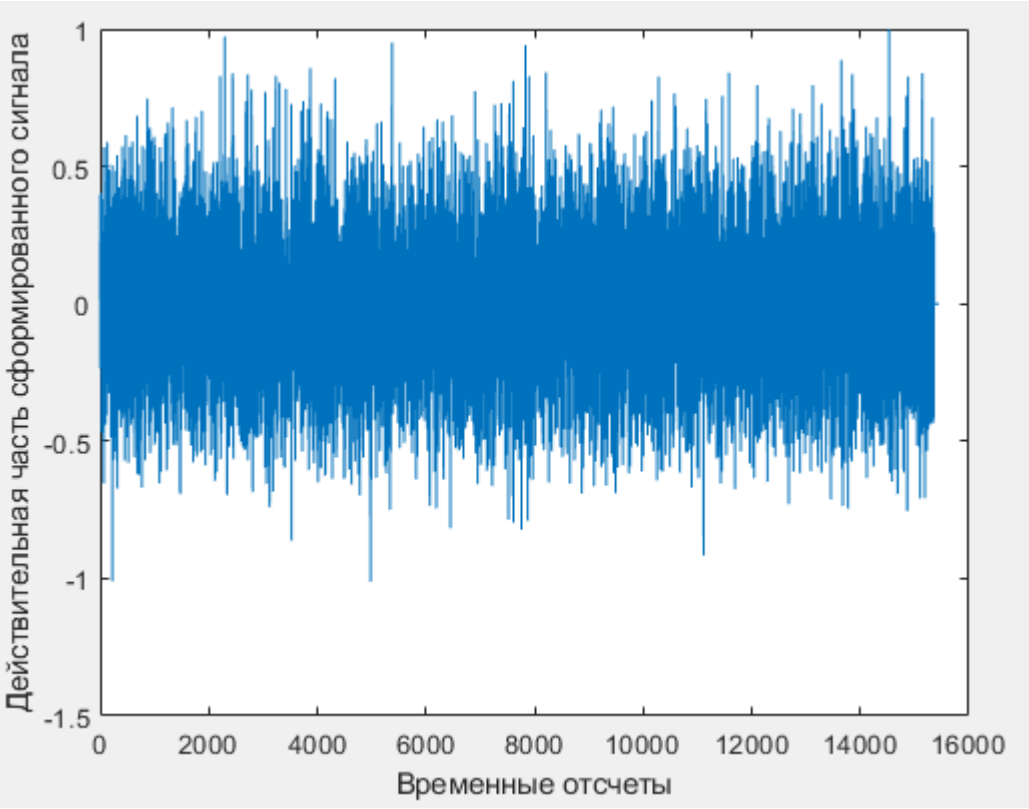


Рисунок 13 – Временная зависимость действительной части передаваемого сигнала

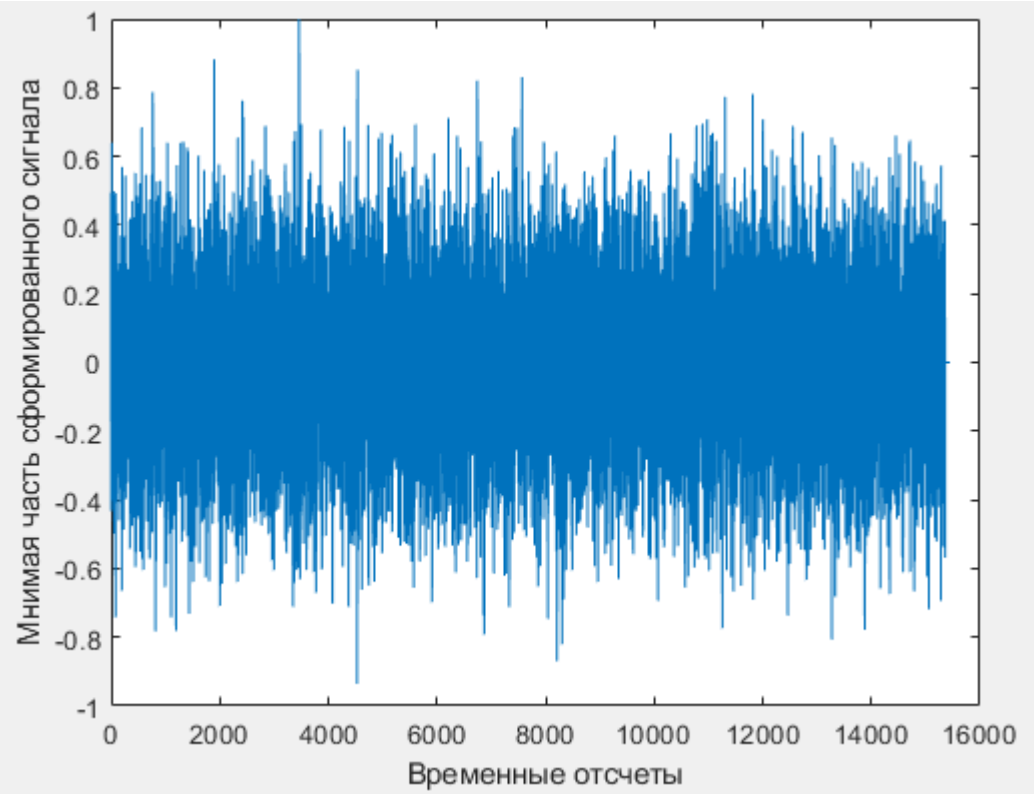


Рисунок 14 – Временная зависимость мнимой части передаваемого сигнала

Этот сигнал является выходным сигналом для модели передатчика. При его пропускании через приемный алгоритм при отсутствии шумов на входе QPSK-демодулятора имеем следующее сигнальное созвездие:

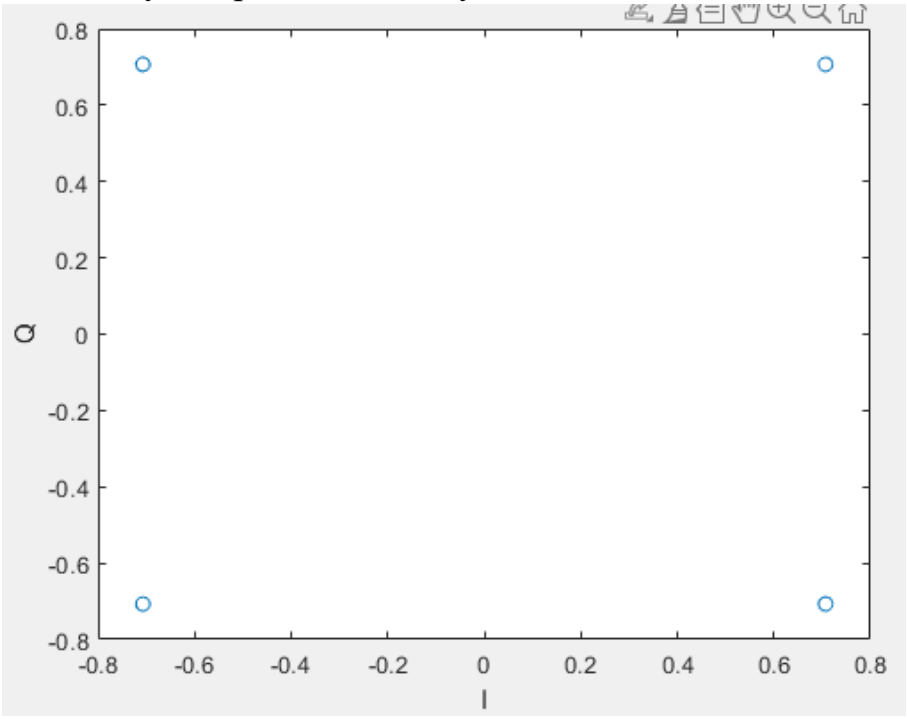


Рисунок 15 – Сигнальное созвездие на входе QPSK-демодулятора

Оно полностью совпадает с теоретическим (рисунок 16, где по осям x и y отложены синфазная и квадратурная составляющие сигнала), поскольку шумов в системе нет (они появятся при проведении эксперимента).

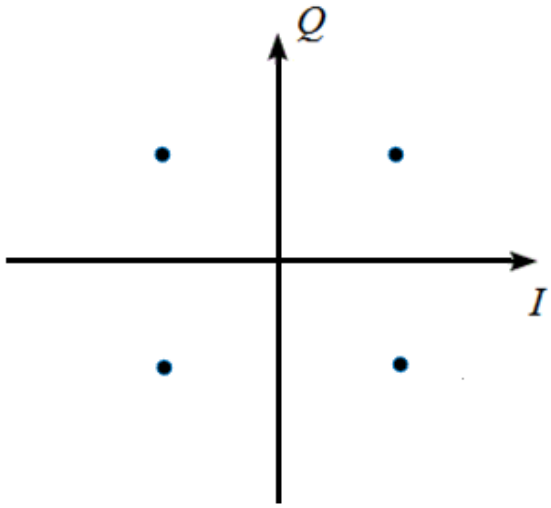


Рисунок 16 – Теоретическое созвездие на входе QPSK-демодулятора

При этом входной битовый поток в передатчике и выходной битовый поток в приемнике полностью совпали, что свидетельствует о правильной работе приемопередающего алгоритма.

Чтобы исследовать возможности данного алгоритма в нестандартных условиях, с максимально узкой частотной полосой, были выбраны следующие параметры: Δf = 15 кГц и NRB = 9. Такое количество ресурсных блоков было получено из соотношения

$$NFFT = \max\left(2^{ceil(Log_2(NRB*12/0.85))}, 128\right),$$

где $NFFT$ – количество точек в быстром преобразовании Фурье. Более того, из алгоритма были исключены временные синхросигналы SSBurst. Это возможно, так как в данном случае мы знаем, где начало и конец передаваемого сигнала. Таким образом, параметры подобраны так, что достигнута полоса частот минимальной ширины.

Как и в предыдущем случае, численный эксперимент дал идеальный результат: построенное созвездие совпало с теоретически ожидаемым, а входной и выходной потоки битов оказались одинаковыми. Из этого следует важный вывод: созданный OFDM-модулятор может работать в нестандартных условиях (узкая полоса частот) не хуже, чем при обычном наборе параметров.

3 Реальный эксперимент

Отличные результаты, полученные в вычислительном эксперименте, позволили пойти еще дальше и провести реальный эксперимент на практике с использованием программно-определяемого радио ADALM-PLUTO (PlutoSDR), поддерживаемого средой Mathworks MatLab [9], [10]. Оно имеет независимые каналы приема и передачи, которые могут работать одновременно (дуплексный режим), и частотный диапазон до 6 ГГц.

Рисунок 17 – Программно-определяемое радио ADALM-PLUTO

Данный модуль излучает сгенерированный сигнал txWaveform, перенесенный на высокую частоту, с антенны «Tx» и принимает его антенной «Rx», отделяя от несущего сигнала, на частоте 1 ГГц (центральная частота) с полосой пропускания в 192 кГц. Это значение было рассчитано из следующего соотношения:

$SamplingRate = NFFT * \Delta f * 1000 = 1.92$ МГц,

где $SamplingRate$ – ширина частотного диапазона для всего кадра.

Следовательно, для подкадра, то есть для 1/10 кадра, $SamplingRate =$ 192 кГц.

Таким образом реализуется пропускание сгенерированного PDSCH-сигнала через модуль ADALM-PLUTO.

На графиках ниже представлены реальные части комплексного сигнала txWaveform (рисунок 18), переносимого впоследствии на несущую частоту, и принятого демодулированного сигнала rxWaveform (рисунки 19-28 – выборка из 10 реализаций приема сигнала).

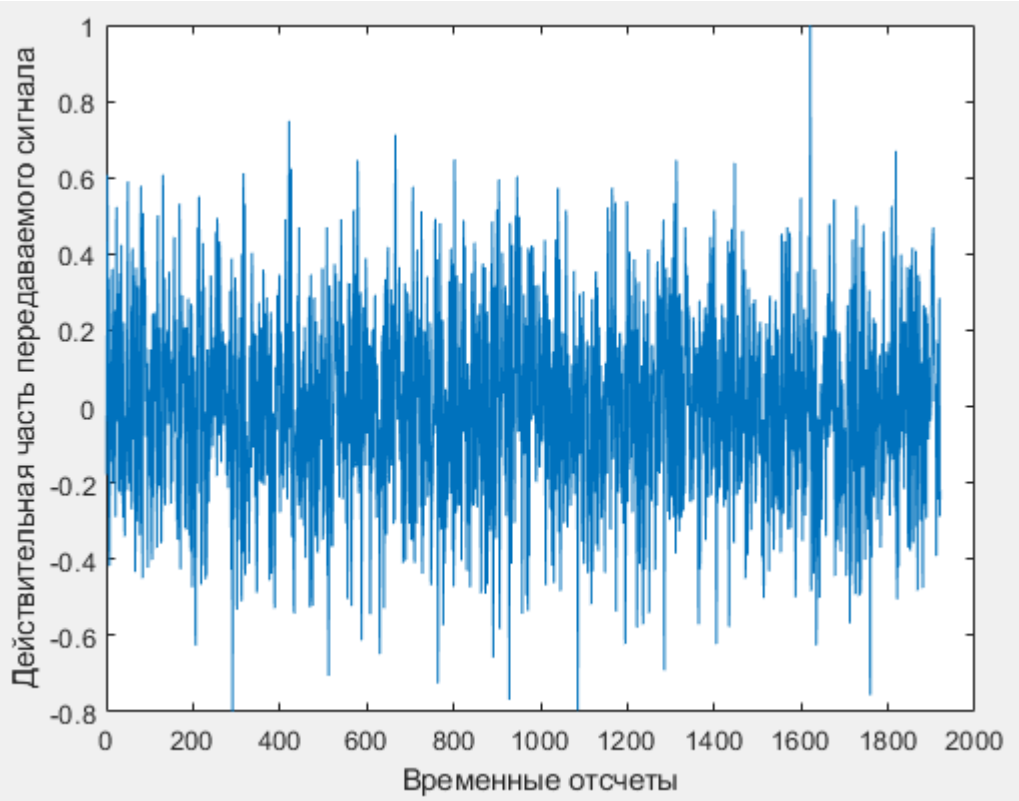


Рисунок 18 – Временная зависимость реальной части передаваемого сигнала txWaveform

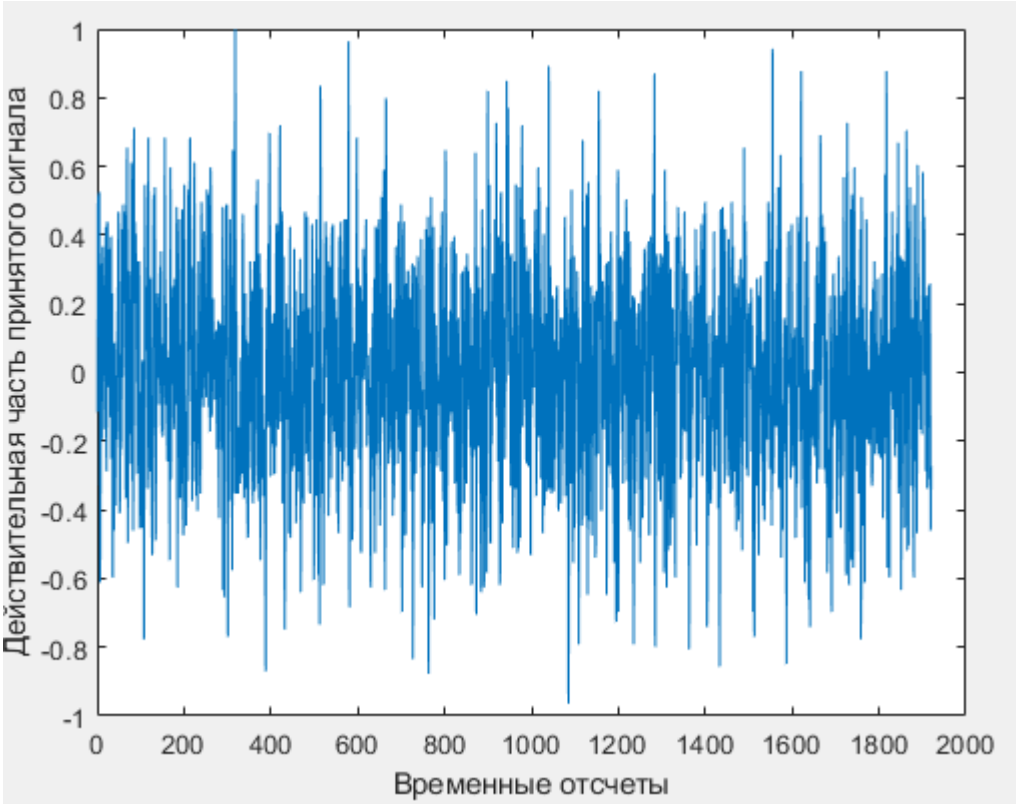


Рисунок 19 – Временная зависимость реальной части принятого сигнала rxWaveform (1 реализация)

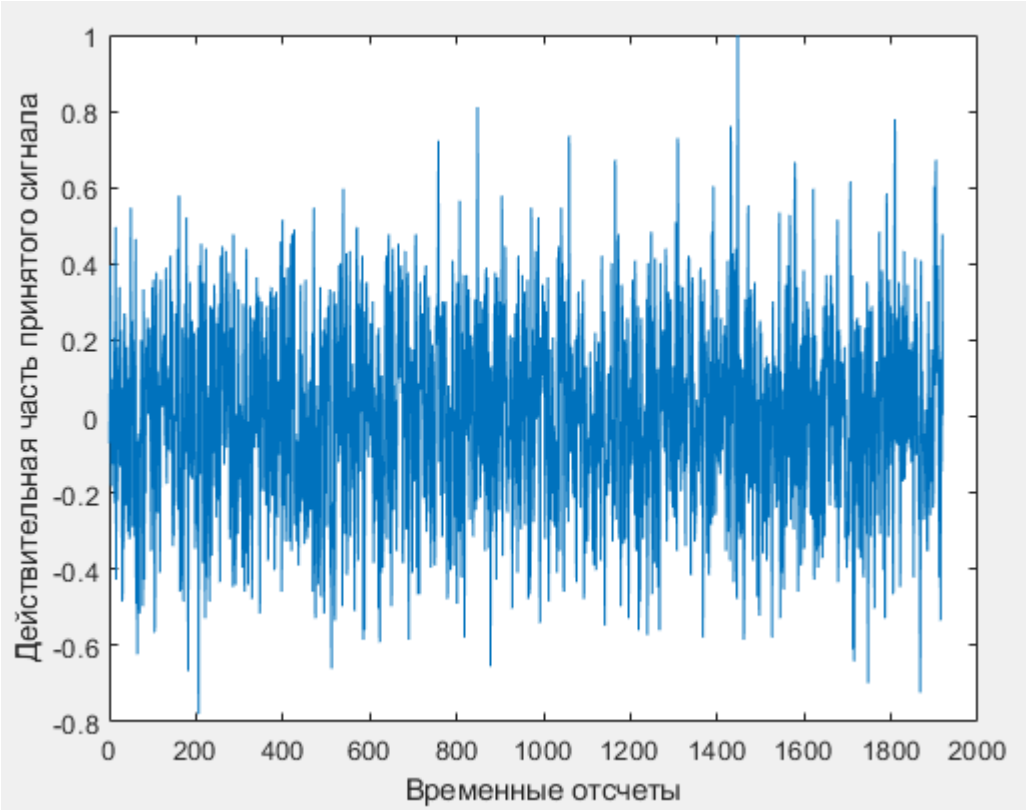


Рисунок 20 – Временная зависимость реальной части принятого сигнала rxWaveform (2 реализация)

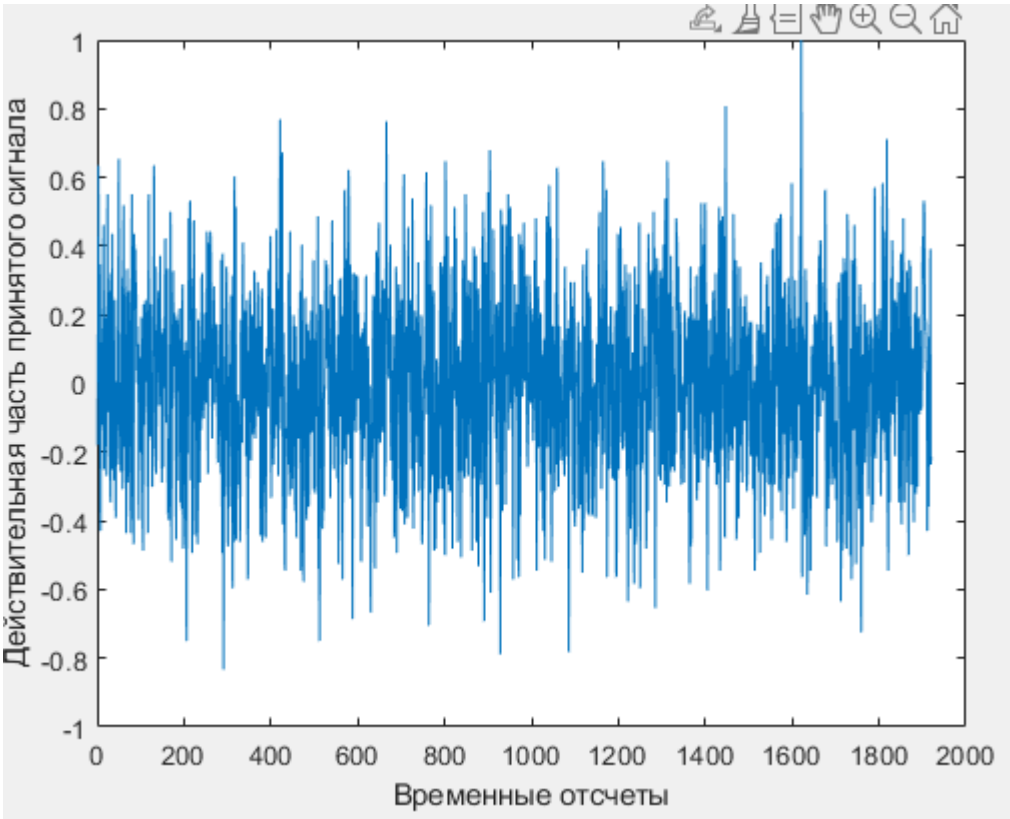


Рисунок 21 – Временная зависимость реальной части принятого сигнала rxWaveform (3 реализация)

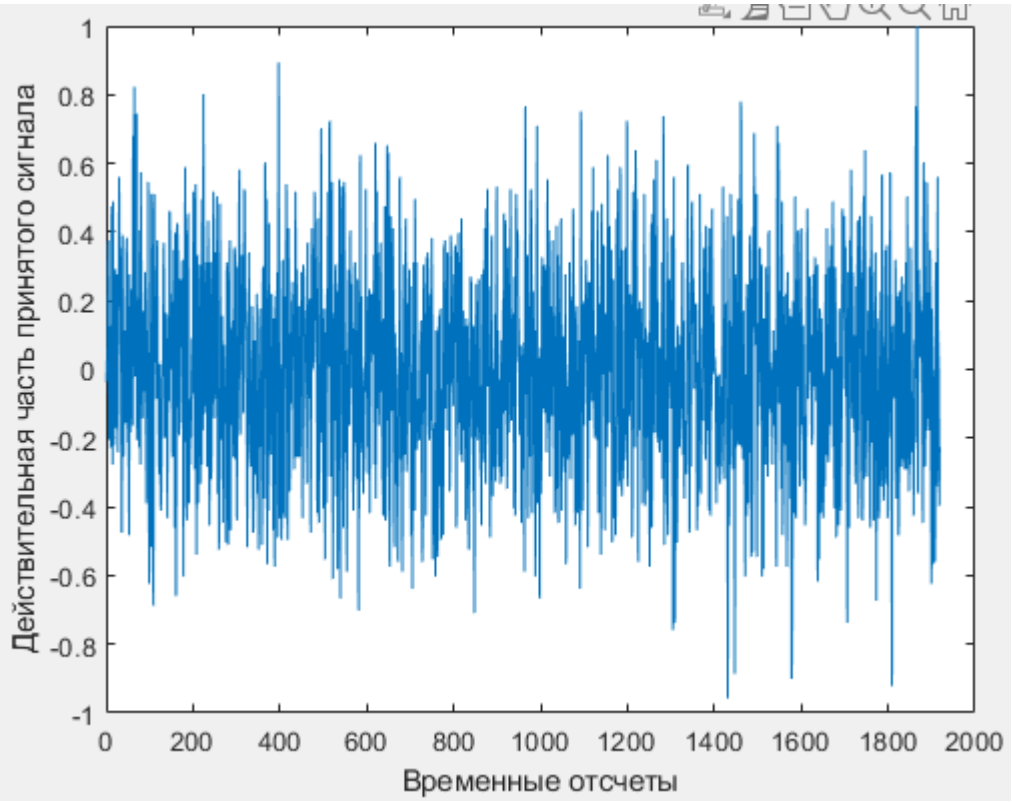


Рисунок 22 – Временная зависимость реальной части принятого сигнала rxWaveform (4 реализация)

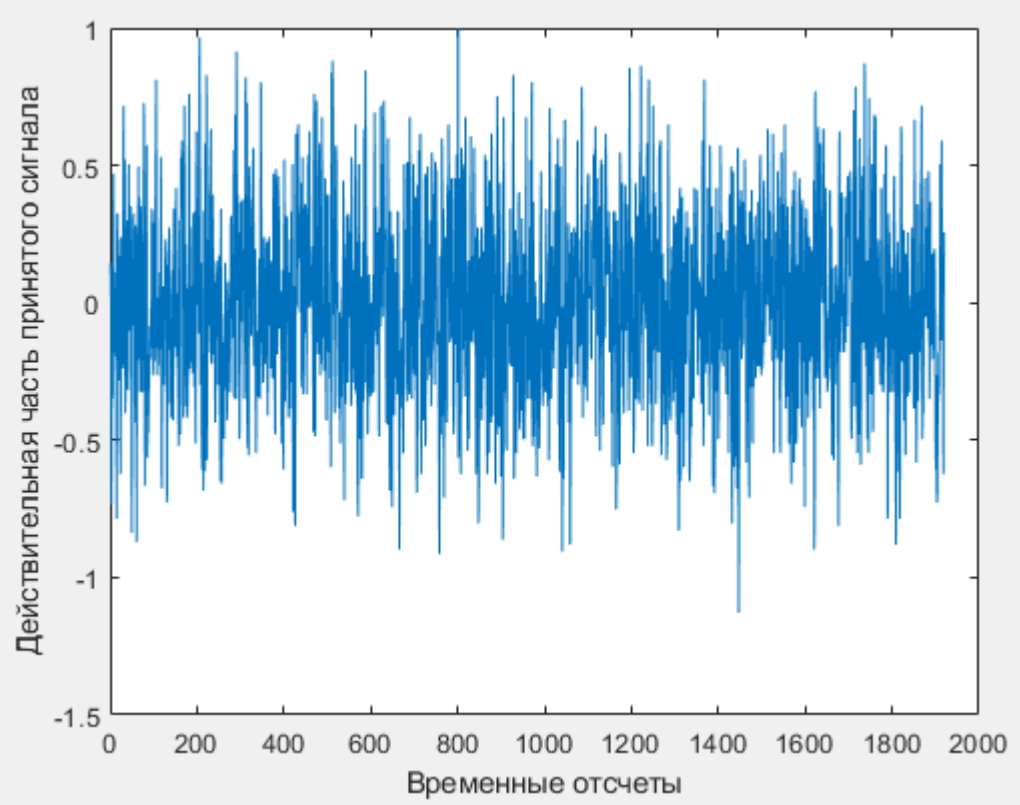


Рисунок 23 – Временная зависимость реальной части принятого сигнала rxWaveform (5 реализация)

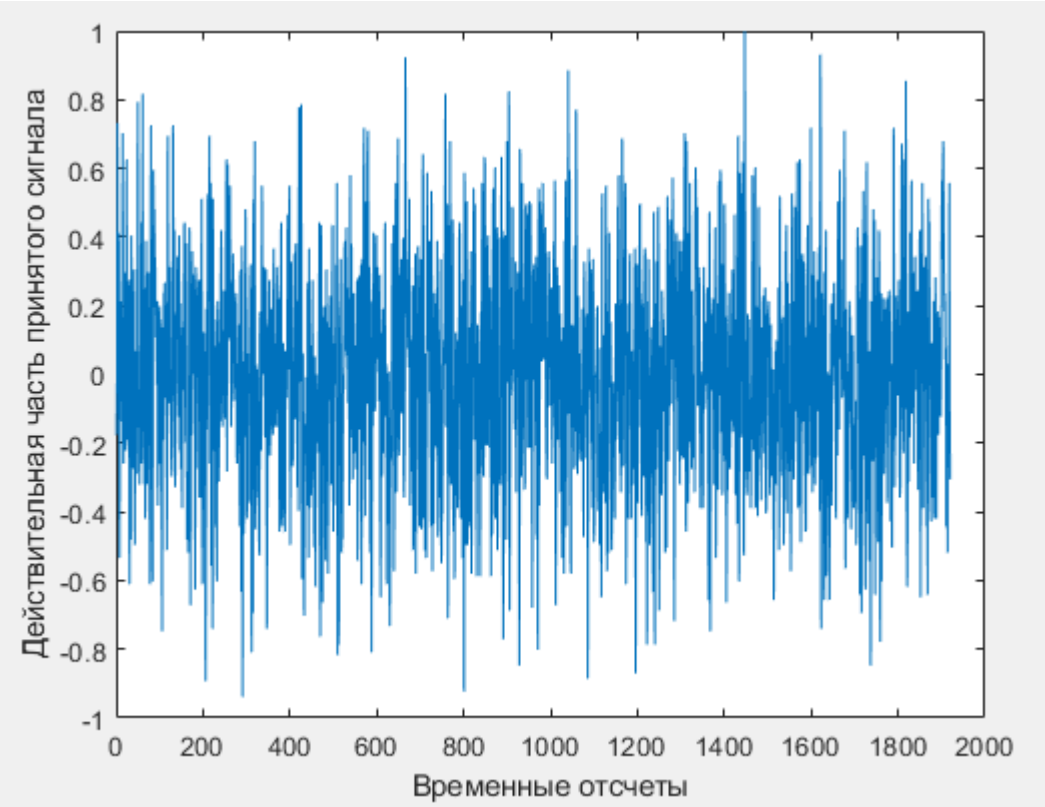


Рисунок 24 – Временная зависимость реальной части принятого сигнала rxWaveform (6 реализация)

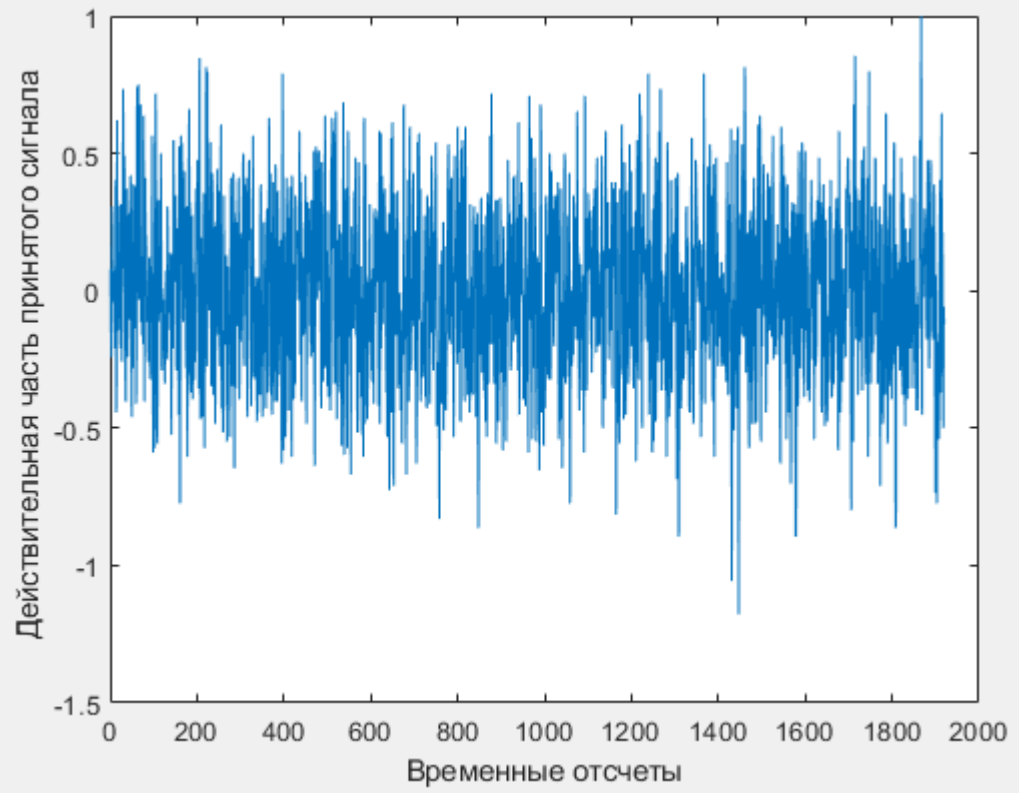


Рисунок 25 – Временная зависимость реальной части принятого сигнала rxWaveform (7 реализация)

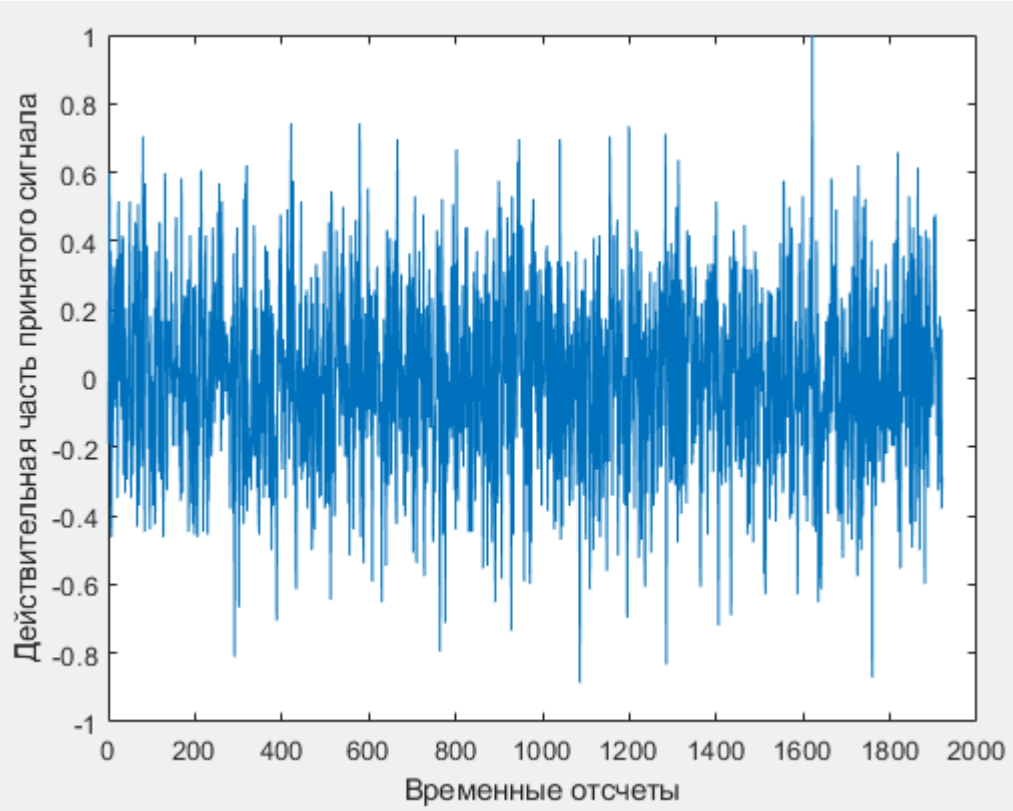


Рисунок 26 – Временная зависимость реальной части принятого сигнала rxWaveform (8 реализация)

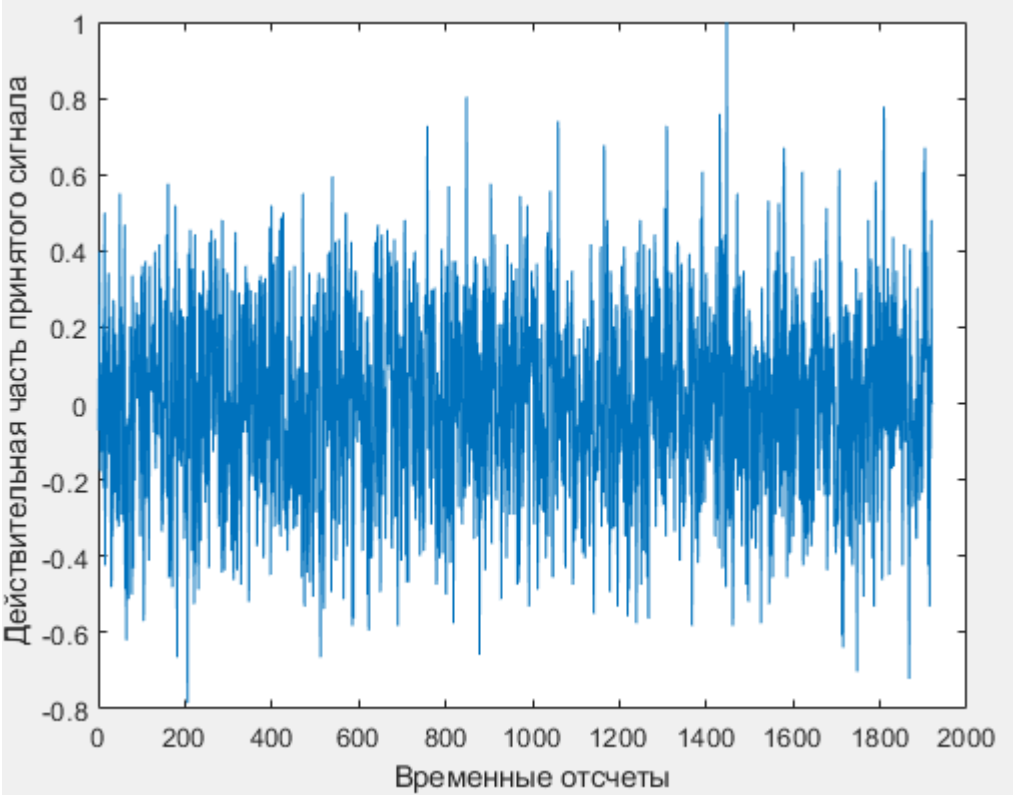


Рисунок 27 – Временная зависимость реальной части принятого сигнала rxWaveform (9 реализация)

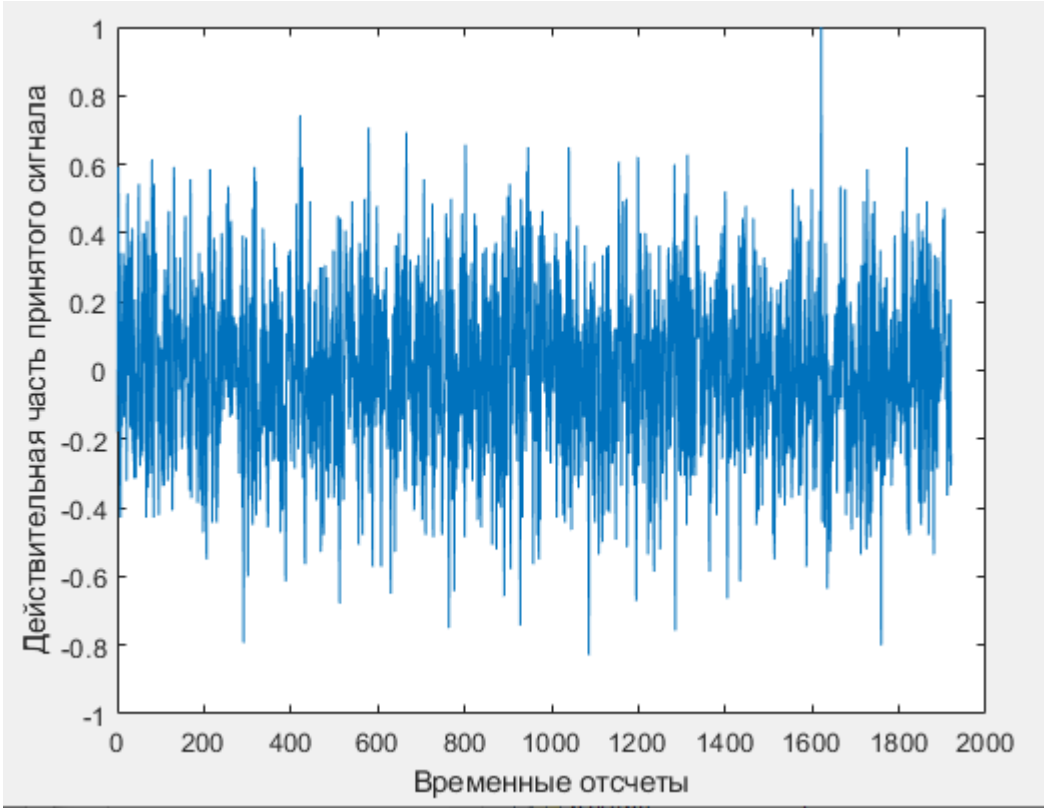


Рисунок 28 – Временная зависимость реальной части принятого сигнала rxWaveform (10 реализация)

В результате обработки принятых сигналов в приемном алгоритме были получены следующие сигнальные созвездия на входе первичной демодуляции для двух реализаций:

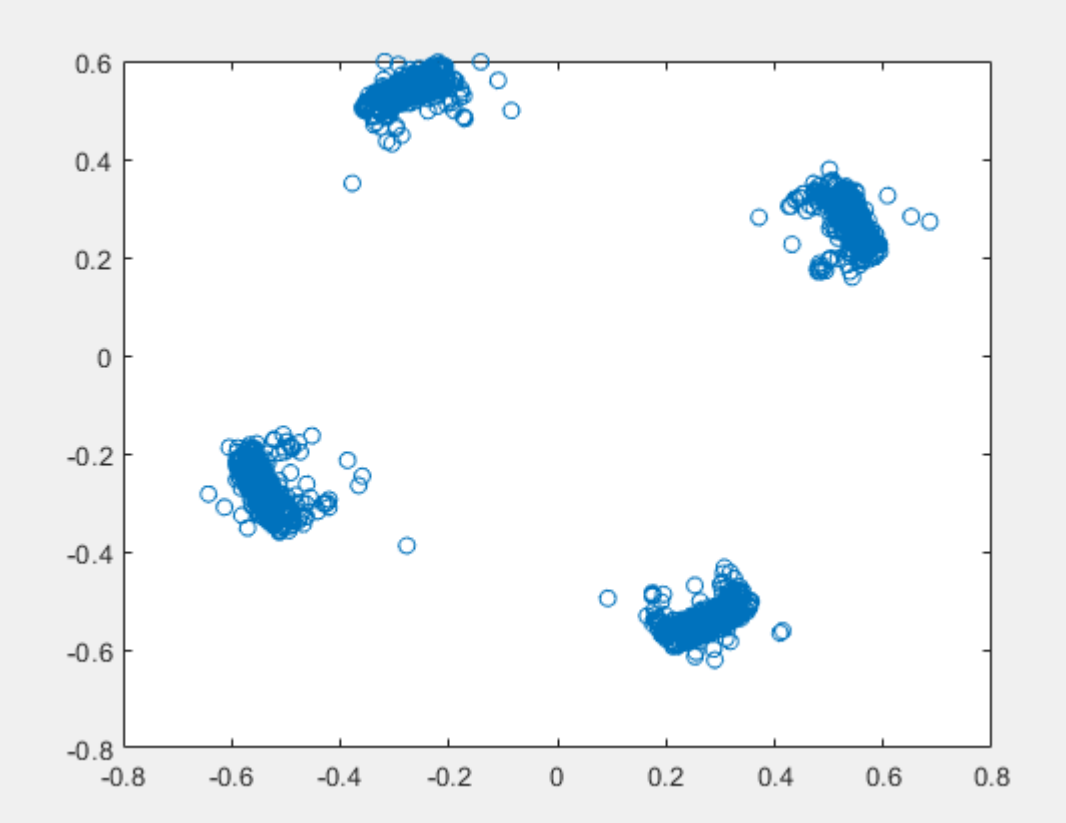


Рисунок 29 – Сигнальное созвездие на входе первичной QPSK-демодуляции

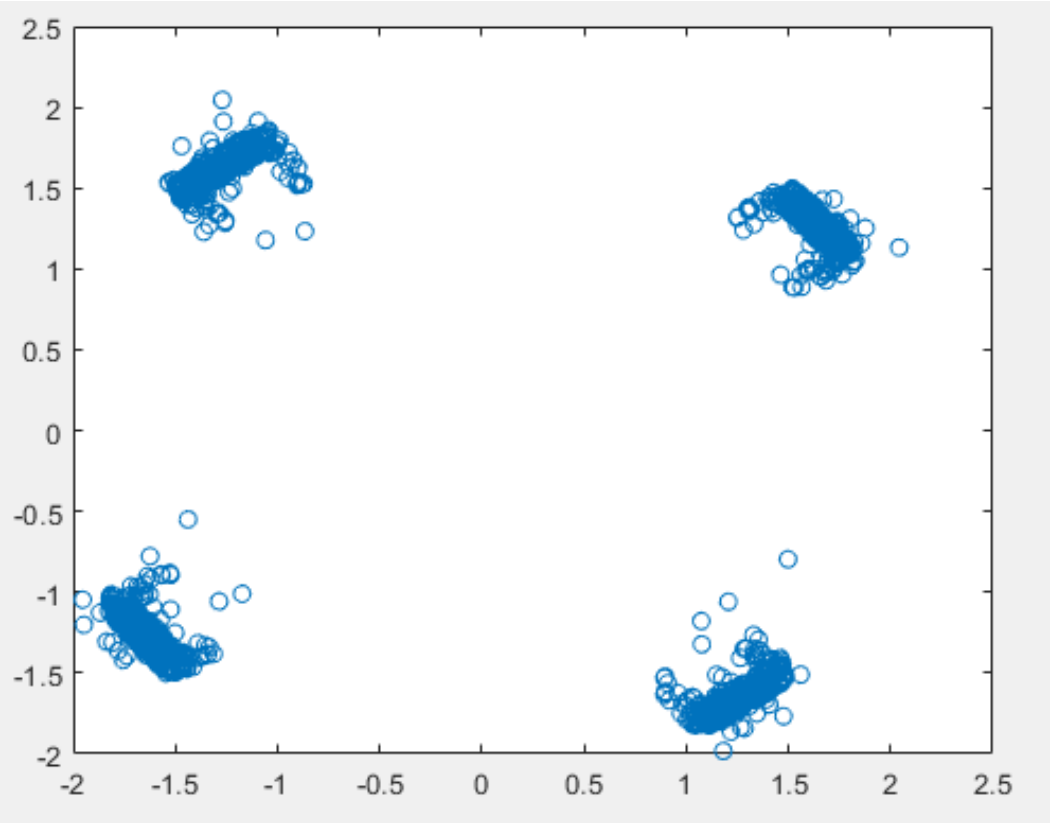


Рисунок 30 – Сигнальное созвездие на входе первичной QPSK-демодуляции

Сопоставляя данные рисунки с теоретически ожидаемым (рисунок 31), можно видеть, что комплексные модуляционные символы на созвездиях

образуют 4 сгустка в каждой полуплоскости. Это свидетельствует о безошибочности обработки принятого сигнала.

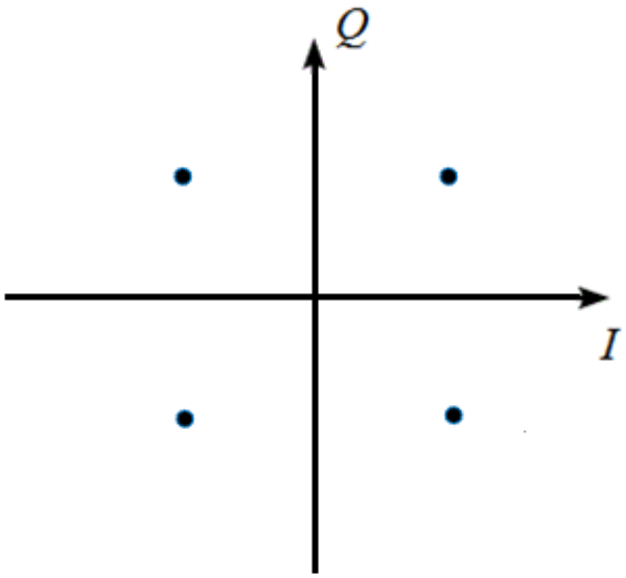


Рисунок 31 – сигнальное созвездие при QPSK-модуляции

При этом входной битовый поток в передатчике и выходной битовый поток в приемнике полностью совпали. Это означает, что сформированный OFDM-модулятором сигнал оказался пригоден не только для численного эксперимента, но и для передачи и приема в реальном мире при наличии множества помех и шумов.

Однако не во всех случаях наблюдались идеальные сигнальные созвездия. Встречались и графики следующего вида:

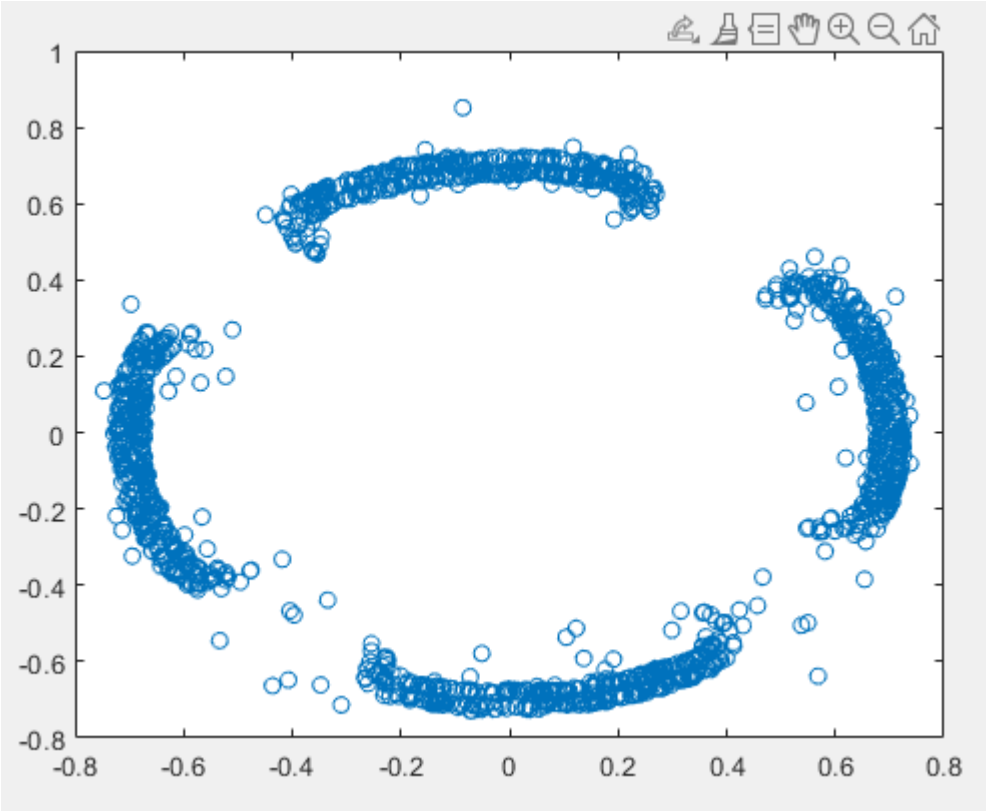


Рисунок 32 – Сигнальное созвездие на входе первичной QPSK-демодуляции (1 реализация)

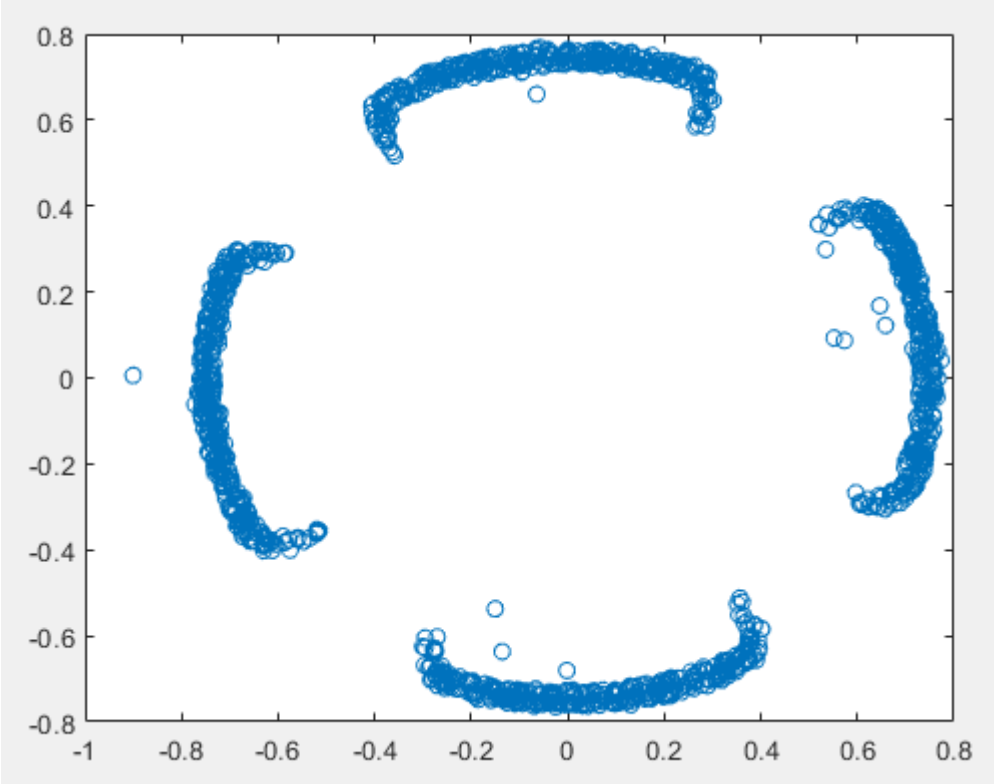


Рисунок 33 – Сигнальное созвездие на входе первичной QPSK-демодуляции (2 реализация)

На этих рисунках видно, что при приеме сигнала имеет место «уход частоты», в результате чего происходят ошибки на этапе QPSK-демодуляции в приемнике, так как точки созвездия попадают в неправильную четверть координатной плоскости. Причинами этого эффекта служат наличие окружающих объектов, вызывающих многократные отражения, шумов, а также близость приемной и передающей антенн у ADALM-PLUTO (в меньшей степени). Решение данной проблемы – применение процедуры equalization (от англ. – выравнивания). Она заключается в применении, помимо сформированного сигнала txWaveform, DM-RS-сигнала (от англ. DeModulation Reference Signal – эталонный опорный сигнал демодуляции). Сравнивая известные значения этого сигнала с его принятыми значениями в приемнике, можно вычислить коэффициенты, на которые надо домножить значения принятого информационного сигнала на каждой частоте.

ЗАКЛЮЧЕНИЕ

Данная научно-исследовательская работа посвящена разработке алгоритмов и программ, реализующих методы формирования OFDM-сигналов в 5G NR. В ходе ее выполнения были получены следующие основные результаты.

1) Освоены язык программирования C++, среда разработки Qt Creator, язык среды для научных расчетов Mathworks MatLab.

2) Разработана программная реализация методов и алгоритмов цифровой модуляции в системах мобильной радиосвязи пятого поколения.

3) Проведен численный эксперимент, подтверждающий работоспособность созданного модулятора для различных наборов входных конфигурационных параметров, в том числе при их нестандартном задании.

4) Освоена работа программно-определяемого радио ADALM-PLUTO.

5) Проведен реальный эксперимент по передаче радиосигналов стандарта 5G NR при помощи программно-определяемого радио.

Созданное программное обеспечение реализует процедуру OFDM-модуляции в 5G и является неотъемлемым звеном всей системы связи. В дальнейшем оно может найти применение в различных радиоустройствах стандарта 5G NR, например, в специализированной измерительной аппаратуре или базовых станциях. Так настоящее программное обеспечение вошло в состав программы для ЭВМ «Формирование синхросигналов мобильной связи стандарта 5G NR», свидетельство о регистрации №2020665979, заявка № 2020664689/69.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1) 3rd Generation Partnership Project (3GPP), Physical channels and modulation. 3GPP 38.211 V.15.7.0, 2018. – [Электронный ресурс]. – URL: : https://www.3gpp.org/ftp/Specs/archive/38_series/38.212 (дата обращения 15.06.2020).

2) Пономарев О. Г. Цифровая модуляция сигналов : учебно-методическое пособие / О. Г. Пономарев ; Том. гос. ун-т. - Томск : [б. и.], 2010. - 46 с.: ил.

3) Почему в WiMax и LTE используют OFDM – [электронный ресурс]. – URL: https://habr.com/ru/post/129101/ (дата обращения 15.02.2021)

4) Учебное пособие по методу мультиплексирования с ортогональным частотным разделением сигналов (OFDM). – [Электронный ресурс]. – URL: : https://ru.scribd.com/doc/97568106/OFDM-%D0%B2-2-%D0%BF%D0%B0%D0%BB%D1%8C%D1%86%D0%B0%D1%85 (дата обращения 15.02.2021).

5) Всё об OFDM модуляции — просто о сложном. – [Электронный ресурс]. – URL: : https://zvondozvon.ru/radiosvyaz/ofdm (дата обращения 15.02.2021).

6) OFDM Модуляция. – [Электронный ресурс]. – URL: : https://sites.google.com/site/nickmikhcomunications/home/articles/ofdm-modulacia (дата обращения 15.02.2021).

7) Сеть радиодоступа 5G, часть 1. – [Электронный ресурс]. – URL: :

https://itechinfo.ru/content/сеть-радиодоступа-5g-часть-1 (дата обращения 15.02.2021).

8) Сеть радиодоступа 5G, часть 2. – [Электронный ресурс]. – URL: : https://itechinfo.ru/node/191 (дата обращения 15.02.2021).

9) ADALM-PLUTO SDR Active Learning Module. – [Электронный ресурс]. – URL: : https://static.chipdip.ru/lib/319/DOC005319632.pdf (дата обращения 20.04.2021).

10) Портативный SDR модуль ADALM PLUTO. – [Электронный ресурс]. – URL: : https://nsk.terraelectronica.ru/news/6245 (дата обращения 20.04.2021).

ПРИЛОЖЕНИЕ А

Программа на языке программирования C++, реализующая OFDM-модуляцию сигнала в 5G NR

```
#include <ofdminfo.h>

/*
Функция bfOFDMModulator является базовой функцией 5G NR, она выполняет ODFM-
модуляцию сигнала.
Входные параметры:
1) ресурсная сетка ResourceElementsGrids - 3-мерный массив комплексных
чисел.
2) структура WAVECONFIG, содержащая подструктуру BANDWIDTHPART со следующими
полями:
SubcarrierSpacing - разнесение поднесущих частот, равное 15, 30, 60, 120,
240.
CyclicPrefix - циклический префикс, принимающий значение Normal или
Extented.
NRB - число ресурсных блоков.
3) j - номер текущей BWP.
Выходной параметр:
waveform - двумерный массив комплексных чисел.

Для реализации обратного быстрого преобразования используется библиотека
fftw3. Для подключения ее к проекту
необходимо:
1) разархивировать файлы из fftw.rar в
C:/Qt/Qt5.13.0/Tools/QtCreator/lib/fftw
2) в переменную окружения PATH (меню Пуск->Панель управления->Система и
безопасность->Система->Дополнительные параметры системы, кнопка Переменные
среды)
добавить путь C:/Qt/Qt5.13.0/Tools/QtCreator/lib/fftw
3) при компиляции выбрать MinGW 32-bit

Функция OFDMInfo возвращает структуру с необходимыми полями для выполнения
функции bfOFDMModulator.
Входные параметры:
1) структура WAVECONFIG, содержащая подструктуру BANDWIDTHPART со следующими
полями:
SubcarrierSpacing - разнесение поднесущих частот, равное 15, 30, 60, 120,
240.
CyclicPrefix - циклический префикс, принимающий значение Normal или
Extented.
NRB - число ресурсных блоков.
2) j - номер текущей BWP.
Выходной параметр:
1) структура OFDMinfo с полями:
  а) Windowing - количество выборок во временной области, к которым
применяется оконное отображение и перекрытие символов OFDM;
  б) SamplingRate - частота дискретизации OFDM модулятора;
  в) NFFT - количество точек в быстром преобразовании Фурье, используемом в
OFDM-модуляции;
  г) CyclicPrefixLengths - длина циклического префикса (в выборках) каждого
OFDM символ в подкадре;
  д) SymbolsPerSlot - количество символов в слоте;
  е) NSubcarriers - количество поднесущих;
  ж) SymbolsPerSubframe - количество символов в подкадре;
*/
```

```
QVector <double> raised_cosine_window(uint n, uint w)
{
    uint k = 0;
    uint l = 0;
    QVector <double> p(w);
    for (uint i = 0; i < w; i++)
    {
        p[i] = 0.5 * (1 - sin(M_PI * (w + 1 - 2 * (p[i] + i + 1)) / (2 *
w)));
    }
    QVector <double> ones(n - w);
    for (uint i = 0; i < n - w; i++)
    {
        ones[i] = 1;
    }
    QVector <double> pp(w);
    for (uint i = 0; i < w; i++)
    {
        pp[i] = p[p.size() - 1 - i];
    }
    QVector <double> P(n + w);
    for (uint i = 0; i < n + w - 1; i++)
    {
        if (i < w)
        {
            P[i] = p[i];
        }
        if ((i >= w) && (i < n))
        {
            P[i] = ones[l];
            l += 1;
        };
        if (i >= n)
        {
            P[i] = pp[k];
            k += 1;
        };
    }
    return P;
}

QVector <double> getCPLengths(WAVECONFIG* wc, uint j)
{
    BANDWIDTHPART* bwp = wc->BWP;
    QVector<double>N_CP_mu(14);
    uint NFFT;
    NFFT = pow(2, ceil(log2(bwp[j].NRB * 12 / 0.85)));
    uint m = log2(ceil(bwp[j].SubcarrierSpacing / 15));
    if (bwp[j].CyclicPrefix == Extented)
    {
        uint mu = 2;
        N_CP_mu.resize(12 * pow(2, mu));
        for (uint i = 0; i < 12 * pow(2, mu); i++)
        {
            N_CP_mu[i] = { 0 };
        }
        for (uint i = 0; i < N_CP_mu.size(); i++)
        {
            N_CP_mu[i] = double(NFFT * 512 / 2048);
        };
    }
    else
```

```
    {
        uint mu = m;
        N_CP_mu.resize(14 * pow(2, mu));
        for (uint i = 0; i < 14 * pow(2, mu); i++)
        {
            N_CP_mu[i] = { 0 };
        }
        for (uint i = 0; i < N_CP_mu.size(); i++)
            if ((i == 0) || (i == 7 * pow(2, mu)))
            {
                N_CP_mu[i] = NFFT * (144 + 16 * pow(2, mu)) / 2048;
            }
            else
            {
                N_CP_mu[i] = NFFT * 144 / 2048;
            }
    }
    return N_CP_mu;
}

OFDMinfo* getOFDMInfo(WAVECONFIG* wc, uint j)
{
    BANDWIDTHPART* bwp = wc->BWP;
    OFDMinfo* OFDMINFO;
    OFDMINFO = new(OFDMinfo);
    double ss = (double)1 / 15;
    if ((log2(bwp[j].SubcarrierSpacing * ss) < 0) ||
((ceil(log2(bwp[j].SubcarrierSpacing * ss))) !=
log2(bwp[j].SubcarrierSpacing * ss)))
    {
        std::cerr << "getOFDMInfo: Invalid SubcarrierSpacing = (%1g) it must
equal 15*(2^n) (15, 30, 60, 120, 240), and n non-negative integer in this
function." << std::endl;
        exit(1);
    }

    OFDMINFO->NFFT = pow(2, ceil(log2(bwp[j].NRB * 12 / 0.85)));
    if (OFDMINFO->NFFT == 128)
    {
        OFDMINFO->Windowing = 4;
    }
    if (OFDMINFO->NFFT == 256)
    {
        OFDMINFO->Windowing = 6;
    }
    if (OFDMINFO->NFFT == 512)
    {
        OFDMINFO->Windowing = 4;
    }
    if (OFDMINFO->NFFT == 1024)
    {
        OFDMINFO->Windowing = 6;
    }
    if (OFDMINFO->NFFT == 2048)
    {
        OFDMINFO->Windowing = 8;
    }
    else
    {
        if (8 - 2 * (11 - (log2(OFDMINFO->NFFT))) > 0)
        {
            OFDMINFO->Windowing = (8 - 2 * (11 - (log2(OFDMINFO->NFFT))));
        }
```

```
        else
        {
            OFDMINFO->Windowing = 0;
        }
    }
    OFDMINFO->CyclicPrefixLengths = getCPLengths(wc, j);

    OFDMINFO->SamplingRate = OFDMINFO->NFFT * bwp[j].SubcarrierSpacing *
1000;
    OFDMINFO->NSubcarriers = bwp[j].NRB * 12;
    OFDMINFO->SymbolsPerSlot = OFDMINFO->CyclicPrefixLengths.size();
    OFDMINFO->SymbolsPerSubframe = OFDMINFO->SymbolsPerSlot * pow(2,
log2(bwp[j].SubcarrierSpacing / 15));
    return  OFDMINFO;
}

bool find_element(int ind, QVector<int>& in)
{
    bool x = false;
    for (int k = 0; k < in.size(); k++) {
        if (ind == in[k]) {
            x = true;
            break;
        }
    }
    return x;
}


QVector <QVector <std::complex <double> > > bfOFDMModulator(QVector <QVector
<QVector <std::complex <double> > > > ResourceElementGrids,
WAVECONFIG* wc, uint j)
{
    BANDWIDTHPART* bwp = wc->BWP;
    OFDMinfo* OFDMINFO;
    OFDMINFO = getOFDMInfo(wc, j);

    uint nrb = wc->BWP[j].NRB;
    uint nSC = nrb * 12;
    uint NFFT = OFDMINFO->NFFT;
    uint firstSC = (NFFT / 2) - nSC / 2 + 1;

    QVector <int> S(3);
    for (uint i = 0; i < 3; i++)
    {
        if (i == 0)
        {
            S[i] = 0;
        }
        if (i == 1)
        {
            S[i] = nSC;
        }
        else {
            S[i] = NFFT;
        }
    };

    bool x = (find_element(ResourceElementGrids.size(), S));
    if (x == false)
    {
```

```
        std::cerr << "The input resource grid must contain a whole number of
resource blocks i.e. number of rows must be an integer multiple of 12." <<
std::endl;
        exit(1);
    }

    uint Nws = OFDMINFO->Windowing;
    uint CyclicPrefixLengths1 = OFDMINFO->CyclicPrefixLengths[0];
    if (Nws > (NFFT - CyclicPrefixLengths1))
    {
        std::cerr << "'The Windowing parameter is invalid" << std::endl;
        exit(1);
    };
    if (fmod(Nws, 2) != 0)
    {
        std::cerr << "For the Windowing parameter the value (%d) must be
even." << std::endl;
        exit(1);
    }

    OFDMINFO->Windowing = Nws;
    uint  CyclicPrefixLengths2 = OFDMINFO->CyclicPrefixLengths[1];
    QVector <double> window0 = raised_cosine_window(NFFT +
CyclicPrefixLengths1, Nws);
    QVector <double> window1 = raised_cosine_window(NFFT +
CyclicPrefixLengths2, Nws);
    QVector <double> CyclicPrefixLengths = OFDMINFO->CyclicPrefixLengths;
    QVector <QVector <double> > exLengths(2, QVector <double>
(CyclicPrefixLengths.size()));

    for (int i = 0; i < CyclicPrefixLengths.size(); i++)
    {
        for (uint j = 0; j < 2; j++)
        {
            if (j < 1)
            {
                exLengths[j][i] = CyclicPrefixLengths[i] + Nws;
            }
            else
            {
                exLengths[j][i] = 0;
            };
        }
    }
    QVector <uint> strideLengths(CyclicPrefixLengths.size());
    for (uint i = 0; i < CyclicPrefixLengths.size(); i++)
    {
        strideLengths[i] = CyclicPrefixLengths[i] + OFDMINFO->NFFT;
    }
    uint symbolsPerSubframe = CyclicPrefixLengths.size();
    int pos = -Nws;

    uint nAnts;
    uint nSymbols;
    for (uint i = 0; i < ResourceElementGrids.size(); i++)
    {
        for (uint j = 0; j < ResourceElementGrids[i].size(); j++)
        {
            nSymbols = ResourceElementGrids[i].size();
            for (uint k = 0; k < ResourceElementGrids[i][j].size(); k++)
            {
                nAnts = ResourceElementGrids[i][j].size();
                break;
```

```
            }
            break;
        }
        break;
    }

    uint Nsamples;
    uint nsamples;
    Nsamples = 0;
    for (uint j = 0; j < nSymbols; j++)
    {
        Nsamples = CyclicPrefixLengths[fmod(j, symbolsPerSubframe)] +
Nsamples;
    }
    nsamples = Nsamples + NFFT * nSymbols;

    QVector <QVector<std::complex <double>>> waveform(nsamples,
QVector<std::complex <double>>(nAnts));
    for (uint i = 0; i < nsamples; i++)
    {
        for (uint j = 0; j < nAnts; j++)
        {
            waveform[i][j] = 0;
        }
    }

    QVector <QVector <std::complex <double> > > ifftin(NFFT, QVector
<std::complex <double> > (nAnts));
    QVector <QVector <std::complex <double> > > iffout(NFFT, QVector
<std::complex <double> > (nAnts));
    QVector <QVector <std::complex <double> > > head(10, QVector
<std::complex<double> > (nAnts));

    for (uint i = 0; i < nSymbols; i++)
    {
        if (ResourceElementGrids.size() == NFFT)
        {
            for (uint j = 0; j < ResourceElementGrids.size(); j++)
            {
                for (uint k = 0; k < nAnts; k++)
                {
                    ifftin[j][k] = ResourceElementGrids[j][i][k];// squeeze
(ResourceElementGrids(:,i,:))
                }
            }
        }

        else
        {
            for (uint j = 0; j < NFFT; j++)
            {
                for (uint k = 0; k < nAnts; k++)
                {
                    ifftin[j][k] = 0;
                }
            }
            for (uint j = 0; j < nSC / 2; j++)
            {
                for (uint k = 0; k < nAnts; k++)
                {
                    ifftin[firstSC + j][k] = ResourceElementGrids[j +
1][i][k]; //
```

```
                    ifftin[firstSC + nSC / 2 + j - 1][k] =
ResourceElementGrids[nSC / 2 + j][i][k];
                }
            }
        }

        double N;
        N = (double)1 / NFFT;

        double in[NFFT][2];
        double out[NFFT][2];

        for (uint k = 0; k < nAnts; k++)
        {
            for (int j = 0; j < NFFT; j++)
            {
                in[j][0] = real(ifftin[j][k]);
                in[j][1] = imag(ifftin[j][k]);
                out[j][0] = -1;
                out[j][1] = -2;
            }
            fftw_plan plan = fftw_plan_dft_1d(NFFT, in, out, FFTW_BACKWARD,
FFTW_ESTIMATE);
            fftw_execute(plan);
            for (int j = 0; j < NFFT; j++)
            {
                iffout[j][k] = { std::complex<double>(out[j][0] *
N,out[j][1] * N) };
            }
            fftw_destroy_plan(plan);
        }

        QVector<double> exLength(exLengths.size());
        for (uint j = 0; j < exLengths.size(); j++)
        {
            exLength[j] = exLengths[j][fmod(i, exLengths[j].size())];
        }

        uint stride = strideLengths[fmod(i, strideLengths.size())];

        QVector <QVector <std::complex <double> > > extended(NFFT +
exLength[0] + exLength[1], QVector <std::complex <double> >(nAnts));
        for (uint j = 0; j < NFFT + exLength[0] + exLength[1]; j++)
        {
            for (uint k = 0; k < nAnts; k++)
            {
                if (j < exLength[0])
                {
                    extended[j][k] = iffout[NFFT - exLength[0] + j][k];
                }
                if ((j >= exLength[0]) & (j < exLength[0] + NFFT))
                {
                    extended[j][k] = iffout[j - exLength[0]][k];
                }
                if (j >= (exLength[0] + NFFT))
                {
                    extended[j][k] = iffout[j - exLength[1] - NFFT][k];
                }
            };
        }

        QVector <QVector <std::complex<double> > >  windowed(NFFT +
exLength[0] + exLength[1], QVector <std::complex<double> >(nAnts));
```

```
if (fmod(i, (symbolsPerSubframe / 2)) == 0)
{
    windowed.resize(window0.size());
    for (uint k = 0; k < nAnts; k++)
    {
        for (uint j = 0; j < window0.size(); j++)
        {
            windowed[j][k] = extended[j][k] * window0[j];
        }
    }

}
else
{
    windowed.resize(window1.size());
    for (uint k = 0; k < nAnts; k++)
    {
        for (uint j = 0; j < window1.size(); j++)
        {
            windowed[j][k] = extended[j][k] * window1[j];
        }
    }
}

if (i == 0)
{
    uint p;
    if (-pos > 0)
    {
        p = -pos;
    }
    else
    {
        p = 0;
    }

    head.resize(p);
    for (uint j = 0; j < p - 1; j++)
    {
        for (uint k = 0; k < windowed[j].size(); k++)
        {
            head[j][k] = windowed[j][k];
        }
    }
    uint L = windowed.size() - head.size();

    if (pos > 0)
    {
        p = pos;
    }
    else
    {
        p = 0;
    }

    for (uint j = 0; j < L; j++)
    {
        for (uint k = 0; k < nAnts; k++)
        {
            if (j + head.size() < windowed.size())
            {
                waveform[j + p][k] = windowed[j + head.size()][k];
            }
```

```
                    else
                    {
                        break;
                    }
                }
            }
        }
        else
        {
            uint L = windowed.size();
            for (uint j = 0; j < windowed.size(); j++)
            {
                for (uint k = 0; k < nAnts; k++)
                {
                    waveform[pos + j][k] = waveform[pos + j][k] +
windowed[j][k];
                }
            }
        }
        pos = pos + stride;
    };


    for (uint j = 0; j < head.size(); j++)
    {
        for (uint k = 0; k < nAnts; k++)
        {
            waveform[waveform.size() - head.size() + j][k] =
waveform[waveform.size() - head.size() + j][k] + head[j][k];
        }
    }

    for (uint j = 0; j < waveform.size(); j++)
    {
        for (uint k = 0; k < nAnts; k++)
        {
            if (fmod(j, 2) == 1)
            {
                waveform[j][k] = -waveform[j][k];
            }
        }
    }
    return waveform;
}
```

ПРИЛОЖЕНИЕ Б

Программа на языке Matxworks MatLab, реализующая численный эксперимент

```
%% NR PDSCH Throughput
% This example measures the physical downlink shared channel (PDSCH)
% throughput of a 5G New Radio (NR) link, as defined by the 3GPP NR
% standard. It implements the transport and physical channels (DL-SCH and
% PDSCH). The transmitter model includes PDSCH demodulation reference
% symbols (DM-RS) and synchronization signal (SS) bursts. This example
% supports clustered delay line (CDL) and tapped delay line (TDL)
% propagation channels and assumes perfect synchronization and channel
% estimation.

% Copyright 2017-2019 The MathWorks, Inc.

%% Introduction
% This example measures the PDSCH throughput of a 5G link, as defined by
% the 3GPP NR standard [ <#14 1> ], [ <#14 2> ], [ <#14 3> ], [ <#14 4> ].
%
% The following 5G NR features are modeled:
%
% * DL-SCH transport channel coding
% * PDSCH and PDSCH DM-RS generation
% * SS burst generation (PSS/SSS/PBCH/PBCH DM-RS)
% * Variable subcarrier spacing and frame numerologies (2^n * 15kHz) for
% normal and extended cyclic prefix
% * TDL and CDL propagation channel models
%
% Other features of the simulation are:
%
% * PDSCH precoding using SVD
% * CP-OFDM modulation
% * Slot wise and non slot wise PDSCH and DM-RS mapping
% * SS burst generation (cases A-E, SS/PBCH block bitmap control)
% * Perfect synchronization and channel estimation
% * HARQ operation with 16 processes
% * The example uses a single bandwidth part across the whole carrier
%
% The figure below shows the processing chain implemented. For clarity, the
% DM-RS and SS burst generation have been omitted.
%
% <<../PDSCHLinkExampleProcessingChain.png>>
%
% This example supports operation with one or two codewords, dependent on
% the number of layers selected for the transmission. Note that perfect
% synchronization and perfect channel knowledge are assumed, i.e. SS/PBCH
% block and PDSCH DM-RS signals are not used at the receiver. A single
% precoding matrix for the whole PDSCH allocation is determined using SVD
% by averaging the channel estimate across all allocated PDSCH PRBs.
% Therefore, for large PDSCH allocations, i.e. occupying a wide bandwidth,
% the single precoding matrix may not be well matched to the channel across
% all frequencies, resulting in performance degradation. There is no
% beamforming on the SS/PBCH blocks in the SS burst.

%% Simulation Length and SNR Points
% Set the length of the simulation in terms of the number of 10ms frames.
% A large number of NFrames should be used to produce meaningful throughput
```

```
% results. Set the SNR points to simulate. The SNR for each layer is
% defined per RE, and it includes the effect of signal and noise across all
% antennas.
clear all
simParameters = [];             % Clear simParameters variable
simParameters.NFrames = 1;      % Number of 10ms frames
simParameters.SNRIn = [10]; % SNR range

%% gNodeB and PDSCH Configuration
% Set the key parameters of the simulation. These include:
%
% * The bandwidth in resource blocks (12 subcarriers per resource block).
% * Subcarrier spacing: 15, 30, 60, 120, 240 (kHz)
% * Cyclic prefix length: normal or extended
% * Cell ID
% * Number of transmit and receive antennas
%
% A substructure containing the DL-SCH and PDSCH parameters is also
% specified. This includes:
%
% * Target code rate
% * Allocated resource blocks (PRBSet)
% * Modulation scheme: 'QPSK', '16QAM', '64QAM', '256QAM'
% * Number of layers
% * PDSCH mapping type
% * DM-RS configuration parameters
%
% Other simulation wide parameters are:
%
% * Propagation channel model: 'TDL' or 'CDL'
% * SS burst configuration parameters. Note that the SS burst generation
% can be disabled by setting the |SSBTransmitted| field to [0 0 0 0].

% Set waveform type and PDSCH numerology (SCS and CP type)
simParameters.NRB = 9;                  % Bandwidth in number of resource
blocks (51RBs at 30kHz SCS for 20MHz BW)
simParameters.SubcarrierSpacing = 15;    % 15, 30, 60, 120, 240 (kHz)
simParameters.CyclicPrefix = 'Normal';   % 'Normal' or 'Extended'
simParameters.NCellID = 1;               % Cell identity

% DL-SCH/PDSCH parameters
simParameters.PDSCH.PRBSet = 0:simParameters.NRB-1; % PDSCH PRB allocation
simParameters.PDSCH.SymbolSet = 0:13;           % PDSCH symbol allocation in
each slot
simParameters.PDSCH.EnableHARQ = false;          % Enable/disable HARQ, if
disabled, single transmission with RV=0, i.e. no retransmissions

simParameters.PDSCH.NLayers = 1;                % Number of PDSCH layers
simParameters.NTxAnts = 1;                      % Number of PDSCH
transmission antennas
if simParameters.PDSCH.NLayers > 4              % Multicodeword transmission
    simParameters.NumCW = 2;                        % Number of codewords
    simParameters.PDSCH.TargetCodeRate = [490 490]./1024; % Code rate used
to calculate transport block sizes
    simParameters.PDSCH.Modulation = {'16QAM','16QAM'};   % 'QPSK', '16QAM',
'64QAM', '256QAM'
    simParameters.NRxAnts = 8;                              % Number of UE
receive antennas
else
    simParameters.NumCW = 1;                        % Number of codewords
    simParameters.PDSCH.TargetCodeRate = 490/1024;  % Code rate used to
calculate transport block sizes
```

```
    simParameters.PDSCH.Modulation = 'QPSK';    %16QAM    % 'QPSK', '16QAM',
'64QAM', '256QAM'
    simParameters.NRxAnts = 1;                      % Number of UE receive
antennas
end

% DM-RS and antenna port configuration (TS 38.211 section 7.4.1.1)
simParameters.PDSCH.PortSet = 0:simParameters.PDSCH.NLayers-1; % DM-RS ports
to use for the layers
simParameters.PDSCH.PDSCHMappingType = 'A';     % PDSCH mapping type
('A'(slot-wise),'B'(non slot-wise))
simParameters.PDSCH.DMRSTypeAPosition = 2;      % Mapping type A only. First
DM-RS symbol position (2,3)
simParameters.PDSCH.DMRSLength = 1;             % Number of front-loaded DM-
RS symbols (1(single symbol),2(double symbol))
simParameters.PDSCH.DMRSAdditionalPosition = 0; % Additional DM-RS symbol
positions (max range 0...3)
simParameters.PDSCH.DMRSConfigurationType = 2;  % DM-RS configuration type
(1,2)
simParameters.PDSCH.NumCDMGroupsWithoutData = 0;% CDM groups without data
simParameters.PDSCH.NIDNSCID = 1;               % Scrambling identity
(0...65535)
simParameters.PDSCH.NSCID = 0;                  % Scrambling initialization
(0,1)
% Reserved PRB patterns (for CORESETs, forward compatibility etc)
simParameters.PDSCH.Reserved.Symbols = [];      % Reserved PDSCH symbols
simParameters.PDSCH.Reserved.PRB = [];          % Reserved PDSCH PRBs
simParameters.PDSCH.Reserved.Period = [];       % Periodicity of reserved
resources

% Define the propagation channel type
simParameters.ChannelType = 'TDL'; % 'CDL' or 'TDL'

% SS burst configuration
simParameters.SSBurst.BlockPattern = 'Case A';    % 30kHz subcarrier spacing
simParameters.SSBurst.SSBTransmitted = [0 1 0 1]; % Bitmap indicating blocks
transmitted in the burst
simParameters.SSBurst.SSBPeriodicity = 160;        % SS burst set
periodicity in ms (5, 10, 20, 40, 80, 160)

validateNLayers(simParameters);

%%
% Create gNodeB configuration structure 'gnb' and PDSCH configuration
% structure 'pdsch'
gnb = simParameters;
pdsch = simParameters.PDSCH;

% Specify additional required fields for PDSCH
pdsch.RNTI = 1;

Xoh_PDSCH = 0;     % The Xoh-PDSCH overhead value is taken to be 0 here

%% Propagation Channel Model Configuration
% Create the channel model object. Both CDL and TDL channel models are
% supported [ <#14 6> ].

nTxAnts = simParameters.NTxAnts;
nRxAnts = simParameters.NRxAnts;

if strcmpi(simParameters.ChannelType,'CDL')
```

```
    channel = nrCDLChannel; % CDL channel object

    % Use CDL-C model (Urban macrocell model)
    channel.DelayProfile = 'CDL-C';
    channel.DelaySpread = 300e-9;

    % Turn the overall number of antennas into a specific antenna panel
    % array geometry
    [channel.TransmitAntennaArray.Size, channel.ReceiveAntennaArray.Size] =
...
        hArrayGeometry(nTxAnts,nRxAnts);

elseif strcmpi(simParameters.ChannelType,'TDL')
    channel = nrTDLChannel; % TDL channel object
    % Set the channel geometry
    channel.DelayProfile = 'TDL-C';
    channel.DelaySpread = 300e-9;
    channel.NumTransmitAntennas = nTxAnts;
    channel.NumReceiveAntennas = nRxAnts;
else
    error('ChannelType parameter field must be either CDL or TDL.');
end

%%
% The sampling rate for the channel model is set using the value returned
% from <matlab:help('hOFDMInfo') hOFDMInfo>.

waveformInfo = hOFDMInfo(gnb);
channel.SampleRate = waveformInfo.SamplingRate;

%%
% Get the maximum number of delayed samples by a channel multipath
% component. This is calculated from the channel path with the largest
% delay and the implementation delay of the channel filter. This is
% required later to flush the channel filter to obtain the received signal.

chInfo = info(channel);
maxChDelay = ceil(max(chInfo.PathDelays*channel.SampleRate)) +
chInfo.ChannelFilterDelay;

%% Reserve PDSCH Resources Corresponding to SS burst
% This section shows how to reserve resources for the transmission of the
% SS burst.

% Create SS burst information structure
ssburst = simParameters.SSBurst;
ssburst.NCellID = gnb.NCellID;
ssburst.SampleRate = waveformInfo.SamplingRate;

% ssbInfo = hSSBurstInfo(ssburst);

% Map the occupied subcarriers and transmitted symbols of the SS burst
% (defined in the SS burst numerology) to PDSCH PRBs and symbols in the
% data numerology
% [mappedPRB,mappedSymbols] =
mapNumerology(ssbInfo.OccupiedSubcarriers,ssbInfo.OccupiedSymbols,ssbInfo.NR
B,gnb.NRB,ssbInfo.SubcarrierSpacing,gnb.SubcarrierSpacing);

% Configure the PDSCH to reserve these resources so that the PDSCH
% transmission does not overlap the SS burst
% reservation.Symbols = mappedSymbols;
```

```
% reservation.PRB = mappedPRB;
% reservation.Period = ssburst.SSBPeriodicity * (gnb.SubcarrierSpacing/15);
% Period in slots
% pdsch.Reserved(end+1) = reservation;

%% Processing Loop
% To determine the throughput at each SNR point, the PDSCH data is analyzed
% per transmission instance using the following steps:
%
% * _Update current HARQ process._ Check the CRC of the previous
% transmission for the given HARQ process. Determine whether a
% retransmission is required. If that is not the case generate new data.
% * _Resource grid generation._ Channel coding is performed by
% <matlab:doc('nrDLSCH') nrDLSCH>. It operates on the input transport
% block provided. Internally, it keeps a copy of the transport block in
% case a retransmission is required. The coded bits are modulated on the
% PDSCH by <matlab:doc('nrPDSCH') nrPDSCH>. The precoding operation is
% applied to the resulting signal.
% * _Waveform generation._ The generated grid is then OFDM modulated.
% * _Noisy channel modeling._ The waveform is passed through a CDL or TDL
% fading channel. AWGN is added. The SNR is defined per RE at each UE
% antenna. For an SNR of 0 dB the signal and noise contribute equally to
% the energy per PDSCH RE per receive antenna [ <#14 5> ].
% * _Perform synchronization and OFDM demodulation._ Information returned
% by the channel is used for perfect synchronization. The synchronized
% signal is then OFDM demodulated.
% * _Perform perfect channel estimation._ Perfect channel estimation is
% used.
% * _Precoding matrix calculation._ The precoding matrix W for the next
% transmission is calculated using singular value decomposition (SVD). A
% single matrix is obtained for the full allocation by averaging the
% channel conditions. Therefore, for a channel with frequency selectivity,
% W could be less accurate for larger allocated bandwidths.
% * _Decode the PDSCH._ The recovered PDSCH symbols for all transmit and
% receive antenna pairs, along with a noise estimate, are demodulated and
% descrambled by <matlab:doc('nrPDSCHDecode') nrPDSCHDecode> to obtain an
% estimate of the received codewords.
% * _Decode the Downlink Shared Channel (DL-SCH) and store the block CRC
% error for a HARQ process._ The vector of decoded soft bits is passed to
% <matlab:doc('nrDLSCHDecoder') nrDLSCHDecoder> which decodes the codeword
% and returns the block CRC error used to determine the throughput of the
% system.

% Array to store the maximum throughput for all SNR points
maxThroughput = zeros(length(simParameters.SNRIn),1);
% Array to store the simulation throughput for all SNR points
simThroughput = zeros(length(simParameters.SNRIn),1);

% Set up Redundancy Version (RV) sequence, number of HARQ processes and
% the sequence in which the HARQ processes are used
if pdsch.EnableHARQ
    % In the final report of RAN WG1 meeting #91 (R1-1719301), it was
    % observed in R1-1717405 that if performance is the priority, [0 2 3 1]
    % should be used. If self-decodability is the priority, it should be
    % taken into account that the upper limit of the code rate at which
    % each RV is self-decodable is in the following order: 0>3>>2>1
    rvSeq = [0 2 3 1];
else
    % HARQ disabled - single transmission with RV=0, no retransmissions
    rvSeq = 0;
end
% Specify the order in which we cycle through the HARQ processes
NHARQProcesses = 16;
```

```
harqSequence = 1:NHARQProcesses;

% Create DLSCH encoder system object
encodeDLSCH = nrDLSCH;
encodeDLSCH.MultipleHARQProcesses = true;
encodeDLSCH.TargetCodeRate = pdsch.TargetCodeRate;

% Create DLSCH decoder system object
decodeDLSCH = nrDLSCHDecoder;
decodeDLSCH.MultipleHARQProcesses = true;
decodeDLSCH.TargetCodeRate = pdsch.TargetCodeRate;

txW=[zeros(1932,10)];
for snrIdx = 1:numel(simParameters.SNRIn)

    % Set the random number generator settings to default values
    rng('default');
    reset(decodeDLSCH);

    SNRdB = simParameters.SNRIn(snrIdx);
    fprintf('\nSimulating transmission scheme 1 (%dx%d) and SCS=%dkHz with
%s channel at %gdB SNR for %d 10ms frame(s)\n',...
        nTxAnts,nRxAnts,gnb.SubcarrierSpacing, ...
        simParameters.ChannelType,SNRdB,gnb.NFrames);

    % Initialize variables used in the simulation and analysis
    bitTput = [];           % Number of successfully received bits per
transmission
    txedTrBlkSizes = [];    % Number of transmitted info bits per
transmission

    % Initialize the state of all HARQ processes
    harqProcesses = hNewHARQProcesses(NHARQProcesses,rvSeq,gnb.NumCW);
    harqProcCntr = 0; % HARQ process counter

    % Reset the channel so that each SNR point will experience the same
    % channel realization
    reset(channel);

    % Total number of OFDM symbols in the simulation period
    NSymbols = gnb.NFrames * 10 * waveformInfo.SymbolsPerSubframe;

    % OFDM symbol number associated with start of each PDSCH transmission
    gnb.NSymbol = 0;

    % Running counter of the number of PDSCH transmission instances
    % The simulation will use this counter as the slot number for each
    % PDSCH
    pdsch.NSlot = 0;

    % Index to the start of the current set of SS burst samples to be
    % transmitted
    ssbSampleIndex = 1;

    % Obtain a precoding matrix (wtx) to be used in the transmission of the
    % first transport block
    estChannelGrid = getInitialChannelEstimate(gnb,nTxAnts,channel);
    newWtx = getPrecodingMatrix(pdsch.PRBSet,pdsch.NLayers,estChannelGrid);

    while gnb.NSymbol < NSymbols/10  % Move to next slot, gnb.NSymbol
increased in steps of one slot
```

```
        % Generate a new SS burst when necessary
        if (ssbSampleIndex==1)
            nSubframe = gnb.NSymbol / waveformInfo.SymbolsPerSubframe;
            ssburst.NFrame = floor(nSubframe / 10);
            ssburst.NHalfFrame = mod(nSubframe / 5,2);
%             [ssbWaveform,~,ssbInfo] = hSSBurst(ssburst);
        end

        % Get HARQ process index for the current PDSCH from HARQ index table
        harqProcIdx =
harqSequence(mod(harqProcCntr,length(harqSequence))+1);

        % Update current HARQ process information (this updates the RV
        % depending on CRC pass or fail in the previous transmission for
        % this HARQ process)
        harqProcesses(harqProcIdx) =
hUpdateHARQProcess(harqProcesses(harqProcIdx),gnb.NumCW);

        % Calculate the transport block sizes for the codewords in the slot
        [pdschIndices,dmrsIndices,dmrsSymbols,pdschIndicesInfo] =
hPDSCHResources(gnb,pdsch);
        trBlkSizes = hPDSCHTBS(pdsch,pdschIndicesInfo.NREPerPRB-Xoh_PDSCH);

        % HARQ: check CRC from previous transmission per codeword, i.e. is
        % a retransmission required?
        for cwIdx = 1:gnb.NumCW
            NDI = false;
            if harqProcesses(harqProcIdx).blkerr(cwIdx) % Errored
                if (harqProcesses(harqProcIdx).RVIdx(cwIdx)==1) % end of
rvSeq
                    resetSoftBuffer(decodeDLSCH,cwIdx-1,harqProcIdx-1);
                    NDI = true;
                end
            else    % No error
                NDI = true;
            end
            if NDI
               % trBlk = ones(trBlkSizes(cwIdx),1);
                trBlk = randi([0 1],trBlkSizes(cwIdx),1);
                setTransportBlock(encodeDLSCH,trBlk,cwIdx-1,harqProcIdx-1);
            end
        end

        % Encode the DL-SCH transport blocks
        codedTrBlock = encodeDLSCH(pdsch.Modulation,pdsch.NLayers,...
            pdschIndicesInfo.G,harqProcesses(harqProcIdx).RV,harqProcIdx-1);

        % Get wtx (precoding matrix) calculated in previous slot
        wtx = newWtx;

        % PDSCH modulation and precoding
        pdschSymbols =
nrPDSCH(codedTrBlock,pdsch.Modulation,pdsch.NLayers,gnb.NCellID,pdsch.RNTI);
        pdschSymbols = pdschSymbols*wtx;

        % PDSCH mapping in grid associated with PDSCH transmission period
        pdschGrid =
zeros(waveformInfo.NSubcarriers,waveformInfo.SymbolsPerSlot,nTxAnts);
        [~,pdschAntIndices] = nrExtractResources(pdschIndices,pdschGrid);
        pdschGrid(pdschAntIndices) = pdschSymbols;
```

```
        % PDSCH DM-RS precoding and mapping
        for p = 1:size(dmrsSymbols,2)
            [~,dmrsAntIndices] =
nrExtractResources(dmrsIndices(:,p),pdschGrid);
            pdschGrid(dmrsAntIndices) = pdschGrid(dmrsAntIndices) +
dmrsSymbols(:,p)*wtx(p,:);
        end

        % OFDM modulation of associated resource elements
        txWaveform = hOFDMModulate(gnb, pdschGrid);

        % Add the appropriate portion of SS burst waveform to the
        % transmitted waveform
        Nt = size(txWaveform,1);
        %txWaveform = txWaveform ;
        %ssbSampleIndex = mod(1 + Nt,size(ssbWaveform,1));

        % Pass data through channel model. Append zeros at the end of the
        % transmitted waveform to flush channel content. These zeros take
        % into account any delay introduced in the channel. This is a mix
        % of multipath delay and implementation delay. This value may
        % change depending on the sampling rate, delay profile and delay
        % spread
        %txWaveform = [txWaveform; zeros(maxChDelay, size(txWaveform,2))];
        %save('txWaveformReal.txt','txWaveformReal','-ascii','-append');
        %save('txWaveformImag.txt','txWaveformImag','-ascii','-append');
        %txW(:,gnb.NSymbol/14+1)= txWaveform(:);
        txWReal=real(txW);
        txWImag=imag(txW);
        %save('txWaveformReal1.mat','txWReal');
        %save('txWaveformImag1.mat','txWImag');

        [rxWaveform,pathGains,sampleTimes] = channel(txWaveform);
        rxWaveform = txWaveform;
        % Add AWGN to the received time domain waveform
        % Normalize noise power to take account of sampling rate, which is
        % a function of the IFFT size used in OFDM modulation. The SNR
        % is defined per RE for each receive antenna (TS 38.101-4).
        SNR = 10^(SNRdB/20);    % Calculate linear noise gain
        N0 = 1/(sqrt(2.0*nRxAnts*double(waveformInfo.Nfft))*SNR);
        %noise =
N0*complex(randn(size(rxWaveform)),randn(size(rxWaveform)));
        noise = zeros (1920,1);
        rxWaveform = rxWaveform + noise;

        % Perfect synchronization. Use information provided by the channel
        % to find the strongest multipath component
        pathFilters = getPathFilters(channel); % get path filters for
perfect channel estimation
        %[offset,mag] = nrPerfectTimingEstimate(pathGains,pathFilters);
        offset = 0;
        rxWaveform = rxWaveform(1+offset:end, :);

        % Perform OFDM demodulation on the received data to recreate the
        % resource grid
        rxGrid = hOFDMDemodulate(gnb, rxWaveform);

        %RXWAVEFORM(:,gnb.NSymbol/14+1) = rxWaveform;

        % Perfect channel estimation, use the value of the path gains
        % provided by the channel
```

```
        estChannelGrid =
nrPerfectChannelEstimate(pathGains,pathFilters,gnb.NRB,gnb.SubcarrierSpacing
,pdsch.NSlot,offset,sampleTimes,gnb.CyclicPrefix);

        % Get perfect noise estimate (from the noise realization)
        noiseGrid = hOFDMDemodulate(gnb,noise(1+offset:end ,:));
        noiseEst = var(noiseGrid(:));

        % Get precoding matrix for next slot
        newWtx =
getPrecodingMatrix(pdsch.PRBSet,pdsch.NLayers,estChannelGrid);

        % Apply precoding to Hest
        % Linearize 4D matrix and reshape after multiplication
        K = size(estChannelGrid,1);
        estChannelGrid =
reshape(estChannelGrid,K*waveformInfo.SymbolsPerSlot*nRxAnts,nTxAnts);
        estChannelGrid = estChannelGrid*wtx.';
        estChannelGrid =
reshape(estChannelGrid,K,waveformInfo.SymbolsPerSlot,nRxAnts,pdsch.NLayers);

        % Get PDSCH resource elements from the received grid
        [pdschRx,pdschHest] =
nrExtractResources(pdschIndices,rxGrid,estChannelGrid);
        pdschEq = pdschRx;
        % Equalization
%         [pdschEq,csi] = nrEqualizeMMSE(pdschRx,pdschHest,noiseEst);

        % Decode PDSCH physical channel
        [dlschLLRs,rxSymbols] =
nrPDSCHDecode(pdschEq,pdsch.Modulation,gnb.NCellID,pdsch.RNTI,noiseEst);

        % Scale LLRs by CSI
%         csi = nrLayerDemap(csi); % CSI layer demapping
%         for cwIdx = 1:gnb.NumCW
%             Qm = length(dlschLLRs{cwIdx})/length(rxSymbols{cwIdx}); % bits
per symbol
%             csi{cwIdx} = repmat(csi{cwIdx}.',Qm,1);   % expand by each bit
per symbol
%             dlschLLRs{cwIdx} = dlschLLRs{cwIdx} .* csi{cwIdx}(:);   %
scale
%         end

        % Decode the DL-SCH transport channel
        decodeDLSCH.TransportBlockLength = trBlkSizes;
        [decbits,harqProcesses(harqProcIdx).blkerr] =
decodeDLSCH(dlschLLRs,pdsch.Modulation,pdsch.NLayers,harqProcesses(harqProcI
dx).RV,harqProcIdx-1);

        DECBITS(:,gnb.NSymbol/14+1) = decbits;

        % Store values to calculate throughput (only for active PDSCH
instances)
        if(any(trBlkSizes) ~= 0)
            bitTput = [bitTput trBlkSizes.*(1-
harqProcesses(harqProcIdx).blkerr)];
            txedTrBlkSizes = [txedTrBlkSizes trBlkSizes];
        end

        % Update starting symbol number of next PDSCH transmission
        gnb.NSymbol = gnb.NSymbol + size(pdschGrid,2);
```

```
        % Update count of overall number of PDSCH transmissions
        pdsch.NSlot = pdsch.NSlot + 1;
        % Update HARQ process counter
        harqProcCntr = harqProcCntr + 1;

        % Display transport block error information per codeword managed by
current HARQ process
        fprintf('\n(%3.2f%%) HARQ Proc %d:
',100*gnb.NSymbol/NSymbols,harqProcIdx);
        estrings = {'passed','failed'};
        rvi = harqProcesses(harqProcIdx).RVIdx;
        for cw=1:length(rvi)
            cwrvi = rvi(cw);
            % Create a report on the RV state given position in RV sequence
and decoding error
            if cwrvi == 1
                ts = sprintf('Initial transmission (RV=%d)',rvSeq(cwrvi));
            else
                ts = sprintf('Retransmission #%d (RV=%d)',cwrvi-
1,rvSeq(cwrvi));
            end
            fprintf('CW%d:%s %s. ',cw-
1,ts,estrings{1+harqProcesses(harqProcIdx).blkerr(cw)});
        end
         plot(pdschRx,'o');
         plot(real(txWaveform));

       txWReal1 =real (txWaveform);
       txWImag1 = imag(txWaveform);
       save('txWaveformReal2.mat','txWReal1');
       save('txWaveformImag2.mat','txWImag1');

    end

    % Calculate maximum and simulated throughput
    maxThroughput(snrIdx) = sum(txedTrBlkSizes); % Max possible throughput
    simThroughput(snrIdx) = sum(bitTput,2);      % Simulated throughput

    % Display the results dynamically in the command window
    fprintf([['\n\nThroughput(Mbps) for ', num2str(gnb.NFrames) ' frame(s)
'],...
        '= %.4f\n'], 1e-6*simThroughput(snrIdx)/(gnb.NFrames*10e-3));
    fprintf(['Throughput(%%) for ', num2str(gnb.NFrames) ' frame(s) =
%.4f\n'],...
        simThroughput(snrIdx)*100/maxThroughput(snrIdx));
end

%% Results
% Display the measured throughput. This is calculated as the percentage of
% the maximum possible throughput of the link given the available resources
% for data transmission.

figure;
plot(simParameters.SNRIn,simThroughput*100./maxThroughput,'o-.')
xlabel('SNR (dB)'); ylabel('Throughput (%)'); grid on;
title(sprintf('(%dx%d) / NRB=%d / SCS=%dkHz',...
             nTxAnts,nRxAnts,gnb.NRB,gnb.SubcarrierSpacing));

% Bundle key parameters and results into a combined structure for recording
simResults.simParameters = simParameters;
simResults.simThroughput = simThroughput;
```

```
%%
% The figure below shows throughput results obtained simulating 10000
% subframes (|NFrames = 1000|, |SNRIn = -18:2:16|).
%
% <<../longRunThroughput.png>>
%

%% Appendix
% This example uses the following helper functions:
%
% * <matlab:edit('hArrayGeometry.m') hArrayGeometry.m>
% * <matlab:edit('hNewHARQProcesses.m') hNewHARQProcesses.m>
% * <matlab:edit('hOFDMDemodulate.m') hOFDMDemodulate.m>
% * <matlab:edit('hOFDMInfo.m') hOFDMInfo.m>
% * <matlab:edit('hOFDMModulate.m') hOFDMModulate.m>
% * <matlab:edit('hPDSCHResources.m') hPDSCHResources.m>
% * <matlab:edit('hPDSCHTBS.m') hPDSCHTBS.m>
% * <matlab:edit('hSSBurst.m') hSSBurst.m>
% * <matlab:edit('hUpdateHARQProcess.m') hUpdateHARQProcess.m>

%% Selected Bibliography
% # 3GPP TS 38.211. "NR; Physical channels and modulation (Release 15)."
% 3rd Generation Partnership Project; Technical Specification Group Radio
% Access Network.
% # 3GPP TS 38.212. "NR; Multiplexing and channel coding (Release 15)." 3rd
% Generation Partnership Project; Technical Specification Group Radio
% Access Network.
% # 3GPP TS 38.213. "NR; Physical layer procedures for control (Release
% 15)." 3rd Generation Partnership Project; Technical Specification Group
% Radio Access Network.
% # 3GPP TS 38.214. "NR; Physical layer procedures for data (Release 15)."
% 3rd Generation Partnership Project; Technical Specification Group Radio
% Access Network.
% # R1-166999. "Detailed configuration of F-OFDM and W-OFDM for LLS
% evaluation", 3GPP RAN WG1 #86, Spreadtrum Communications, August 2016.
% # 3GPP TR 38.901. "Study on channel model for frequencies from 0.5 to 100
% GHz (Release 15)." 3rd Generation Partnership Project; Technical
% Specification Group Radio Access Network.
% # 3GPP TS 38.101-4. "NR; User Equipment (UE) radio transmission and
% reception. Part 4: Performance requirements (Release 15)." 3rd Generation
% Partnership Project; Technical Specification Group Radio Access Network.

%% Local Functions

function validateNLayers(simParameters)
% Validate the number of layers
    if length(simParameters.PDSCH.PortSet)~= simParameters.PDSCH.NLayers
        error('The number of elements of PortSet (%d) must be the same as
the number of layers (%d)',...
            length(simParameters.PDSCH.PortSet),
simParameters.PDSCH.NLayers);
    end

    if simParameters.PDSCH.NLayers >
min(simParameters.NTxAnts,simParameters.NRxAnts)
        error('The number of layers (%d) must satisfy NLayers <=
min(NTxAnts,NRxAnts) = min(%d,%d) = (%d)',...

simParameters.PDSCH.NLayers,simParameters.NTxAnts,simParameters.NRxAnts,min(
simParameters.NTxAnts,simParameters.NRxAnts));
    end
end
```

```
function estChannelGrid = getInitialChannelEstimate(gnb,nTxAnts,channel)
% Obtain channel estimate before first transmission. This can be used to
% obtain a precoding matrix for the first slot.

    ofdmInfo = hOFDMInfo(gnb);

    chInfo = info(channel);
    maxChDelay = ceil(max(chInfo.PathDelays*channel.SampleRate)) +
chInfo.ChannelFilterDelay;

    % Temporary waveform (only needed for the sizes)
    tmpWaveform =
zeros((ofdmInfo.SamplesPerSubframe/ofdmInfo.SlotsPerSubframe)+maxChDelay,nTx
Ants);

    % Filter through channel
    [~,pathGains,sampleTimes] = channel(tmpWaveform);

    % Perfect timing synch
    pathFilters = getPathFilters(channel);
    offset = nrPerfectTimingEstimate(pathGains,pathFilters);

    nslot = gnb.NSymbol/ofdmInfo.SymbolsPerSlot;

    % Perfect channel estimate
    estChannelGrid =
nrPerfectChannelEstimate(pathGains,pathFilters,gnb.NRB,gnb.SubcarrierSpacing
,nslot,offset,sampleTimes);

end

function wtx = getPrecodingMatrix(PRBSet,NLayers,hestGrid)
% Calculate precoding matrix given an allocation and a channel estimate

    % Allocated subcarrier indices
    allocSc = (1:12)' + 12*PRBSet(:).';
    allocSc = allocSc(:);

    % Average channel estimate
    [~,~,R,P] = size(hestGrid);
    estAllocGrid = hestGrid(allocSc,:,:,:);
    Hest = permute(mean(reshape(estAllocGrid,[],R,P)),[2 3 1]);

    % SVD decomposition
    [~,~,V] = svd(Hest);
    wtx = V(:,1:NLayers).';

end

function [mappedPRB,mappedSymbols] =
mapNumerology(subcarriers,symbols,nrbs,nrbt,fs,ft)
% Map the SSBurst numerology to PDSCH numerology. The outputs are:
%   - mappedPRB: 0-based PRB indices for carrier resource grid (arranged in
a column)
%   - mappedSymbols: 0-based OFDM symbol indices in a slot for carrier
resource grid (arranged in a row)
%     carrier resource grid is sized using gnb.NRB, gnb.CyclicPrefix,
spanning 1 slot
% The input parameters are:
```

```
%   - subcarriers: 1-based row subscripts for SSB resource grid (arranged in
a column)
%   - symbols: 1-based column subscripts for SSB resource grid (arranged in
an N-by-4 matrix, 4 symbols for each transmitted burst in a row, N
transmitted bursts)
%     SSB resource grid is sized using ssbInfo.NRB, normal CP, spanning 5
subframes
%   - nrbs: source (SSB) NRB
%   - nrbt: target (carrier) NRB
%   - fs: source (SSB) SCS
%   - ft: target (carrier) SCS

    mappedPRB = unique(fix((subcarriers-(nrbs*6) - 1)*fs/(ft*12) +
nrbt/2),'stable');

    symbols = symbols.';
    symbols = symbols(:).' - 1;

    if (ft < fs)
        % If ft/fs < 1, reduction
        mappedSymbols = unique(fix(symbols*ft/fs),'stable');
    else
        % Else, repetition by ft/fs
        mappedSymbols = reshape((0:(ft/fs-1))' + symbols(:)'*ft/fs,1,[]);
    end

end
```

ПРИЛОЖЕНИЕ В

Программа на языке Mathworks Matlab, задающая параметры tx и rx – передатчика и приемника у ADALM-PLUTO

```
a=zeros(1000,1);
b=a;


% txWaveform1(1:1932)=txW(1:1932,10);
%
% txWaveform1=reshape(txW,[],1);

 load('txWaveformReal2.mat');
 load('txWaveformImag2.mat');

txWaveform2 = txWReal1+1i*txWImag1;
%txWaveform2 = complex(ones(1932,1));

% load('WaveformReal.mat');
% load('WaveformImag.mat');
%txWaveform2 = WaveformReal+1i*WaveformImag;

txWaveform1 =[a;txWaveform2 ;b];
%txWaveform1=txWaveform2;
%txWaveform1 = WaveformReal+1i*WaveformImag;
fs = 192000;%1920000
tx = sdrtx('Pluto');
tx.RadioID = 'usb:0';
tx.CenterFrequency = 1e9;
tx.BasebandSampleRate = fs;
tx.Gain = -5;
tx.SamplesPerFrame = size(txWaveform1,1);
transmitRepeat(tx,txWaveform1);

rx = sdrrx('Pluto');
rx.RadioID = 'usb:0';
rx.ChannelMapping = 1;
rx.ShowAdvancedProperties = true;
rx.CenterFrequency = 1e9;
rx.BasebandSampleRate = fs;
rx.SamplesPerFrame = 2*size(txWaveform1,1);
rx.GainSource = 'manual';
rx.OutputDataType = 'double';
%         rx.EnableQuadratureCorrection = 1;
%         rx.EnableRFDCCorrection = 1;
%         rx.EnableBasebandDCCorrection = 1;
rx.FrequencyCorrection = 127;%4.3;
rx.Gain = 13;

rxWaveformPluto = rx();
%   release(rx);
%   release(tx);
%   clear( 'tx','rx', 'sw');
% plot(real(rxWaveformPluto))

%         for i=1:size(txWaveform1,1)
%           % rxWaveformPluto=circshift(rxWaveformPluto,1);
%
```

```
%
CORREL=(abs(corrcoef(txWaveform1,rxWaveformPluto(i:(i+1919)))));
%            if (CORREL(2,1)>0.8)
%            rxWaveformPlutoCircshift=rxWaveformPluto(i:(i+1919));
%            end
%          end

         for i=1:size(txWaveform1,1)
           % rxWaveformPluto=circshift(rxWaveformPluto,1);

            CORREL=(abs(corrcoef(txWaveform2,rxWaveformPluto(i:(i+1919)))));
            if (CORREL(2,1)>0.8)
            rxWaveformPlutoCircshift=rxWaveformPluto(i:(i+1919));
            end
         end

         clear('rxWaveformPluto');
         CORREL2=(abs(corrcoef(txWaveform2, rxWaveformPlutoCircshift)));
         rxWaveformCircshiftReal=real(rxWaveformPlutoCircshift);
         txWaveformReal=real(txWaveform1);
        % plot(real(rxWaveformPluto));% hold all ;plot(real(txWaveform))

%          sum(rxWaveformPluto - txWaveform)

       %   plot(txWaveformReal/max(txWaveformReal(1001:2920)));hold all
   plot(real(rxWaveformPlutoCircshift)/max(real(rxWaveformPlutoCircshift)));
```

ПРИЛОЖЕНИЕ Г

Программа на языке Mathworks Matlab, реализующая приемный алгоритм сигнала, пропущенного через ADALM-PLUTO

```
[pdschIndices,dmrsIndices,dmrsSymbols,pdschIndicesInfo] =
hPDSCHResources(gnb,pdsch);
        trBlkSizes = hPDSCHTBS(pdsch,pdschIndicesInfo.NREPerPRB-Xoh_PDSCH);

        nRxAnts = 1;
        % Create DLSCH decoder system object
decodeDLSCH = nrDLSCHDecoder;
decodeDLSCH.MultipleHARQProcesses = true;
decodeDLSCH.TargetCodeRate = pdsch.TargetCodeRate;

rxWaveform2 = rxWaveformPlutoCircshift;
% rxWaveform1 = txWaveform;
% rxGrid2 = hOFDMDemodulate(gnb, resMy);

rxGrid2 = hOFDMDemodulate(gnb, rxWaveform2);
[pdschRx2] = nrExtractResources(pdschIndices,rxGrid2);

[dmrsSubs1, dmrsSubs2]  = ind2sub(size(rxGrid2),dmrsIndices);

% dmrsSubs = double(nrPBCHDMRSIndices(ncellid,'IndexStyle','subscript'));

hest1 = zeros([108 14 nRxAnts 1]);
[l_hest,k_hest] = meshgrid(1:size(hest1,2),1:size(hest1,1));
dmrsRx = nrExtractResources(dmrsIndices,rxGrid2);
dmrsEsts = dmrsRx .* conj(dmrsSymbols);
rof = randi([36 50],36,1)/1000;
  fEs = scatteredInterpolant(dmrsSubs2+rof,dmrsSubs1,dmrsEsts);
%  f = scatteredInterpolant(dmrsSubs(:,2),dmrsSubs(:,1),dmrsEsts(:));
hest1(:,:) = fEs(l_hest,k_hest);

pdschHest] = nrExtractResources(pdschIndices,hest1);
pdschEq2,csi] = nrEqualizeMMSE(pdschRx2,pdschHest,0);
% pdschEq1 = pdschRx1;
% Decode PDSCH physical channel
[dlschLLRs2] =
nrPDSCHDecode(pdschRx2,pdsch.Modulation,gnb.NCellID,pdsch.RNTI);
% [dlschLLRs1,rxSymbols1] =
nrPDSCHDecode(pdschEq1,pdsch.Modulation,gnb.NCellID,pdsch.RNTI,noiseEst);
decodeDLSCH.TransportBlockLength = trBlkSizes;
[decbits2,blkerr] =
decodeDLSCH(dlschLLRs2,pdsch.Modulation,pdsch.NLayers,harqProcesses(harqProc
Idx).RV,harqProcIdx-1);
x = double(decbits2)-trBlk;
n = length( find( x(1:end) ==0 ) )

plot(pdschRx2,'o'); hold all; plot(pdschEq2,'o');
```

АНТИПЛАГИАТ
ОБНАРУЖЕНИЕ ЗАИМСТВОВАНИЙ

Отчет о проверке на заимствования №1

Автор: zverezhuck@mail.ru / ID: 9247485
Проверяющий (zverezhuck@mail.ru / ID: 9247485)

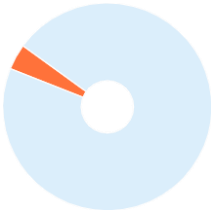
Отчет предоставлен сервисом «Антиплагиат» - users.antiplagiat.ru

ИНФОРМАЦИЯО ДОКУМЕНТЕ

№ документа: 2
Начало загрузки: 07.06.2021 17:43:44
Длительность загрузки: 00:00:02
Имя исходного файла: bakalavrskaya.pdf
Название документа: bakalavrskaya
Размер текста: 71 кБ
Символов в тексте: 72440
Слов в тексте: 8696
Число предложений: 572

ИНФОРМАЦИЯОБ ОТЧЕТЕ

Начало проверки: 07.06.2021 17:43:47
Длительность проверки: 00:00:19
Комментарии: не указано
Модули поиска: Интернет



ЗАИМСТВОВАНИЯ	САМОЦИТИРОВАНИЯ	ЦИТИРОВАНИЯ	ОРИГИНАЛЬНОСТЬ
4,28%	0%	0%	95,72%

Заимствования — доля всех найденных текстовых пересечений, за исключением тех, которые система отнесла к цитированиям, по отношению к общему объему документа. Самоцитирования — доля фрагментов текста проверяемого документа, совпадающий или почти совпадающий с фрагментом текста источника, автором или соавтором которого является автор проверяемого документа, по отношению к общему объему документа.
Цитирования — доля текстовых пересечений, которые не являются авторскими, но система посчитала их использование корректным, по отношению к общему объему документа. Сюда относятся оформленные по ГОСТу цитаты; общеупотребительные выражения; фрагменты текста, найденные в источниках из коллекций нормативноправовой документации.
Текстовое пересечение — фрагмент текста проверяемого документа, совпадающий или почти совпадающий с фрагментом текста источника.
Источник — документ, проиндексированный в системе и содержащийся в модуле поиска, по которому проводится проверка.
Оригинальность — доля фрагментов текста проверяемого документа, не обнаруженных ни в одном источнике, по которым шла проверка, по отношению к общему объему документа.
Заимствования, самоцитирования, цитирования и оригинальность являются отдельными показателями и в сумме дают 100%, что соответствует всему тексту проверяемого документа.
Обращаем Ваше внимание, что система находит текстовые пересечения проверяемого документа с проиндексированными в системе текстовыми источниками. При этом система является вспомогательным инструментом, определение корректности и правомерности заимствований или цитирований, а также авторства текстовых фрагментов проверяемого документа остается в компетенции проверяющего.

№	Доля в отчете	Источник	Актуален на	Модуль поиска
[01]	2,74%	Deep Learning Data Synthesis for 5G Channel Estimation - MATLAB & Simulink https://mathworks.com	11 Июл 2020	Интернет
[02]	1,21%	NR PDSCH Resource Allocation and DM-RS and PT-RS Reference Signals - MATLAB & Simulink https://mathworks.com	11 Июл 2020	Интернет
[03]	0,32%	R1-1719301.zip (2/2) https://3gpp.org	16 Дек 2019	Интернет

Руководитель выпускной квалификационной работы
Канд. физ.-мат. наук, доцент, ТГУ _____ / О.Г. Пономарев