МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования
«Сибирский государственный университет телекоммуникаций и информатики»

Кафедра телекоммуникационных систем и вычислительных средств
(ТС и ВС)

ОТЧЕТ
по дисциплине
«Теория массового обслуживания»

по теме:
ЛАБОРАТОРНАЯ РАБОТА №5 МАРКОВСКИЕ ЦЕПИ. ИССЛЕДОВАНИЕ ЭРГОДИЧЕСКИХ СВОЙСТВ

Студент:
Группа № ИА331 *В.А. Павлова*

Предподаватель:
А. В. Андреев

Новосибирск 2025 г.

СОДЕРЖАНИЕ

1 ЗАНЯТИЕ №1. ВРЕМЕННАЯ И ЧАСТОТНАЯ ФОРМЫ СИГНАЛОВ. ПРЕОБРАЗОВАНИЯ ФУРЬЕ. ДИСКРЕТИЗАЦИЯ СИГНАЛОВ

1.1 Цель работы

Получить представление о формах радиосигналов, их частотном и временном представлении, а также о преобразованиях Фурье и аналоговоцифровых преобразованиях сигналов, частоте дискретизации сигналов.

1.2 Задачи к лабораторной работе

1. Сгенерировать и визуализировать непрерывный периодический сигнал в соответствии с вариантом.
2. Определить максимальную частоту в спектре сигнала.
3. Рассчитать минимальную частоту дискретизации по теореме Котельникова.
4. Выполнить оцифровку сигнала и прямое дискретное преобразование Фурье (ПДПФ).
5. Восстановить аналоговый сигнал по дискретным отсчётам.
6. Исследовать влияние увеличения частоты дискретизации.
7. Проанализировать аудиосигнал (голос), определить его спектр и частоту дискретизации.
8. Исследовать эффект прореживания (downsampling) и квантования по уровню.
9. Оформить отчёт с графиками, анализом и выводами.

2 КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

2.1

2.2 Введение в системы мобильной связи

Системы мобильной связи представляют собой сети связи, которые обеспечивают передачу голосовой информации, текстовых сообщений, данных и других типов информации между мобильными устройствами, такими как смартфоны, планшеты и ноутбуки, и сетью оператора связи. Эти системы позволяют пользователям оставаться подключенными к сети и обмениваться информацией в движении, находясь в различных местах.

Ключевыми устройствами, взаимодействующими между собой на радиоинтерфейсе, являются базовые станции (БС) и мобильные абонентские терминалы (АТ), представляющие собой совокупность приемопередающего оборудования, а также специализированного программного обеспечения (ПО).

2.3 Радиосигналы и их преобразование

Важным аспектом при взаимодействии этих устройств между собой является необходимость формирования, передачи и приема радиосигналов – электромагнитных колебаний. Антенны – это устройства, которые непосредственно преобразуют электрическую энергию в электромагнитные колебания или радиосигналы.

В рамках данного занятия студентам предстоит разобраться в следующих аспектах:

- что такое радиосигналы;
- частотное и временное представление сигналов и как помогают преобразования Фурье переходить из одной области в другую;
- как аналоговые сигналы записываются и обрабатываются цифровыми устройствами связи или что такое АЦП (ADC);
- что такое частота дискретизации сигналов (sample rate);
- что такое отсчет сигнала (sample);

– минимальный набор математического аппарата, необходимый для выполнения преобразований Фурье;
– что такое спектр радиосигналов и пр.

2.3.1 Радиосигналы и их оцифровка

Начнем с радиосигналов. Сигнал – это некоторый физический процесс, несущий в себе информацию. На рисунке 1 показан пример зависимости амплитуды (напряжения) радиосигнала от времени – непрерывный, аналоговый сигнал.

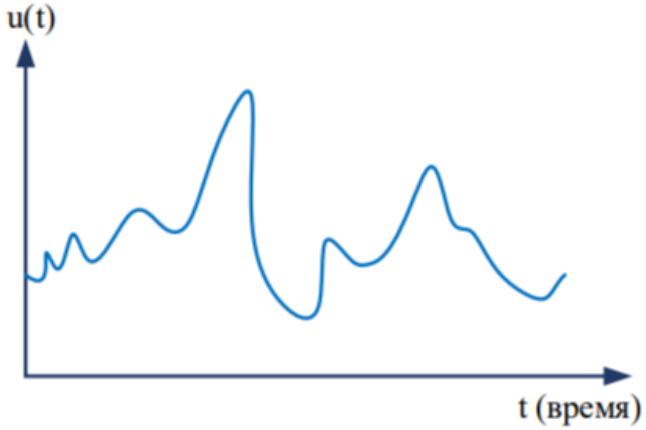

Рисунок 1 — Пример зависимости амплитуды (напряжения) радиосигнала от времени

Вся современная техника радиосвязи, особенно мобильной, начиная со 2-го поколения – цифровая. Это значит, что элементы памяти таких устройств состоят из триггеров, которые могут принимать лишь 2 состояния – 0 или 1. Эти триггеры группируются в регистры, обычно по 8 бит (именно поэтому на языке C/C++ нельзя создать переменную меньше 8 бит, так как это минимальный объем памяти, который можно выделить под хранение переменных или данных).

2.4 Аналого-цифровое преобразование (АЦП)

Закономерен вопрос, а как работать цифровому устройству с сигналом, представленным на рисунке 1? Как разложить его по ячейкам памяти для последующей обработки и извлечения из него информации?

Ответом на данный вопрос является процедура оцифровки аналогового сигнала или **Аналого-цифровое преобразование (АЦП)** – это элемент при-

емника радиосигнала, на вход которого поступает входное напряжение (например, как на рисунке 1), а на выходе – временные отсчеты данного сигнала – цифровые значения дискретизированной амплитуды сигнала, которые уже можно «сложить» в регистры памяти, например, в виде int16 или float значений, как показано на рисунке **??**. То есть задача АЦП – превратить аналоговый сигнал в цифровой для последующей обработки цифровым устройством с целью извлечения из него данных.

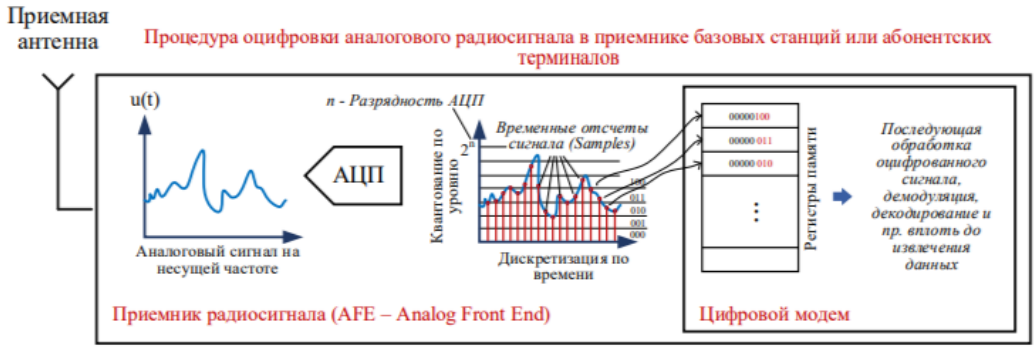

Рисунок 2 — Аналого-цифровое преобразование радиосигнала

Таким образом, АЦП выполняет несколько функций: временная дискретизация («временная нарезка сигнала»), квантование по уровню (представление сигнала конечным числом уровней, округление его точных значений), кодирование (упаковка выхода АЦП с заданной разрядностью в выделенные ячейки памяти, размер которых кратен восьми).

АЦП характеризуется следующими параметрами:

– разрядность АЦП – число бит, которым кодируется напряжение сигнала;
– частота дискретизации, задающаяся опорным генератором (скорость «нарезки» сигнала);
– диапазон входного сигнала – минимальные и максимальные значения напряжения сигнала на входе АЦП, при котором АЦП работает корректно;
– передаточная характеристика – зависимость числового эквивалента выходного двоичного кода от величины аналогового сигнала, имеет вид ступенчатой функцией.

АЦП может вносить искажения в сигнал в случае нелинейности его передаточной характеристики. Кроме того, стабильность опорного генератора,

отвечающего за «нарезку» (дискретизацию) сигнала в строго определенные моменты времени, тоже порой может вызывать нарекания и приводить в тому, что отсчеты будут браться не совсем в предполагаемые моменты времени. Это вызывает джиттер – фазовый шум. Такой шум оказывает существенное влияние на быстро изменяющийся сигнал (с высокой частотой дискретизации, в широкой полосе частот).

Цифро-аналоговое преобразование (ЦАП) решает обратную задачу: на вход устройства подаются цифровые отсчеты, которые затем преобразуются в напряжение. Таким образом формируется ступенчатый и непрерывный сигнал, который затем может быть сглажен фильтром нижних частот.

2.5 Математические основы ЦОС

2.5.1 Дельта-функция

Основополагающим понятием в цифровой обработке сигналов (ЦОС) является дельта-функция, изображенная на рисунке 3, интегрируя которую на бесконечности, можно получить 1 (формула 1). Рассмотрим некоторые полезные для дальнейших вычислений свойства этой функции. Во-первых, сдвиг функции по времени – рисунок 3 и выражение 2, а также стоит обратить внимание на произведение двух дельта-функций в 0 и в каком-то другом моменте времени, интеграл которого будет равен 0 3.

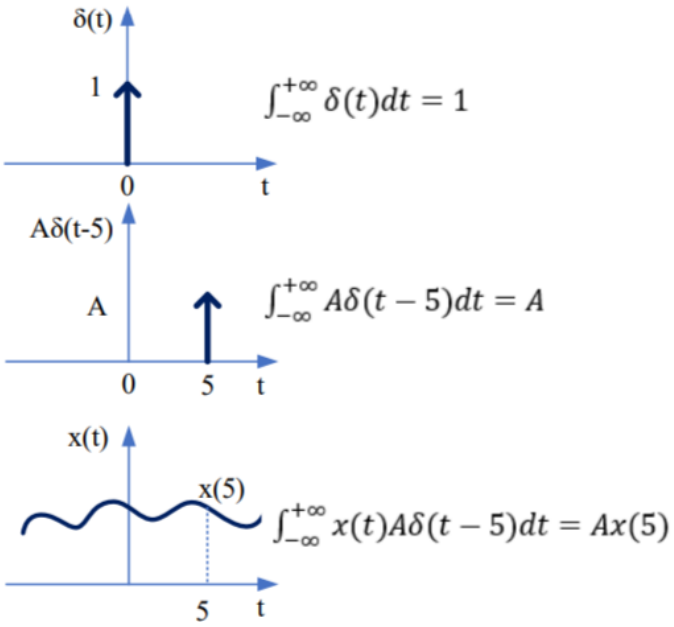

Рисунок 3 — Дельта-функция и ее свойства

$$\int_{-\infty}^{+\infty} \delta(t)dt = 1 \tag{1}$$

$$\int_{-\infty}^{+\infty} A\delta(t-5)dt = A \tag{2}$$

$$\int_{-\infty}^{+\infty} A\delta(t)\delta(t-5)dt = 0 \tag{3}$$

Если взять интеграл произведения непрерывной функции $x(t)$ и дельта-функции, например в точке $t = 5$, то он будет равен произведению значения функции $x(t)$ в $t = 5$ на множитель, стоящий перед дельта-функцией (формула 4).

$$\int_{-\infty}^{+\infty} x(t)A\delta(t-5)dt = A \cdot x(5) \tag{4}$$

Это становится очевидным, если взглянуть на рисунок 3. Функция $A\delta(t-5)$ принимает ненулевое значение лишь в $t = 5$. То есть при перемножении ее на $x(t)$, получится $A \cdot x(5)$.

Сумма дельта-функций (формула 5) показана на рисунке 4:

$$p(t) = \sum_{n=-\infty}^{\infty} \delta(t - nT) \tag{5}$$

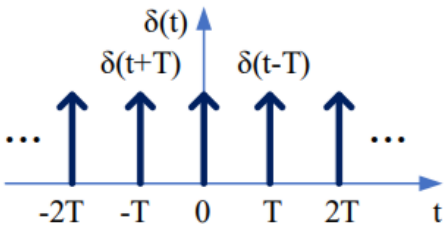

Рисунок 4 — Сумма дельта-функций

И еще один важный нюанс, касающийся дельта-функций – результат преобразования Фурье (позже рассмотрим подробнее) суммы дельта-функций – это также сумма дельта-функций (формула 6). Прямое преобразование Фурье используется для перехода от временной к частотной форме сигнала.

$$p(t) = \sum_{n=-\infty}^{\infty} \delta(t - nT)$$
$$\text{ППФ} \tag{6}$$
$$P(j\omega) = \frac{2\pi}{T} \sum_{k=-\infty}^{\infty} \delta\left(\omega - k\frac{2\pi}{T}\right)$$

где ω_s – это частота дискретизации, равная $\frac{2\pi}{T}$.

2.5.2 Теорема Котельникова

В данном разделе столь пристальное внимание уделяется дельта-функциям, так как они используются в процессе оцифровки аналоговых сигналов в ходе временной дискретизации сигналов (выборка отсчетов сигналов). На рисунке 5 представлена процедура временной дискретизации аналогового сигнала в ходе АЦП. $x(t)$ – это непрерывный аналоговый сигнал, который требуется записать в цифровое устройство для последующей обработки. Для этого $x(t)$ перемножается с дельта-функциями, равномерно распределенными по оси времени и расстояние между которыми зависит от частоты дискретизации ω_s (или $f_s = \omega_s/(2\pi)$).

На рисунке 5 для примера частота дискретизации выбрана равной 20 отсчетам в секунду ($1/T = 1/0.05 = 20$ Гц). Отсчеты отстоят друг от друга на величину $= 0.05$ секунд. Это означает, что чтобы сохранить/записать одну секунду такого сигнала нам потребуется выделить $20 \times$ (количество байт, занимаемых одним отсчетом) байт памяти. И речь не просто про память цифровых устройств, но и про требования к скорости обработки таких массивов данных, ведь одно дело, когда на обработку поступает 20 байт в секунду, а другое – 1 гигабайт. Разумеется, разработчики таких систем и ПО для них заинтересованы сохранять минимальное необходимое количество отсчетов для последующей обработки.

Прежде всего непрерывный сигнал нужно дискретизировать по времени – разбить ось времени на промежутки равной длины и сохранить значения амплитуд в эти моменты времени. Но закономерен вопрос – а как часто нужно брать эти отсчеты, чтобы не потерять данные, содержащиеся в этом сигнале при его дальнейшем восстановлении, какой должна быть частота дискретизации (количество отсчетов в секунду). Ответом на этот вопрос является **теорема Котельникова (или Найквиста)**: непрерывный сигнал с ограниченным спектром можно точно восстановить по его дискретным отсчетам, если они

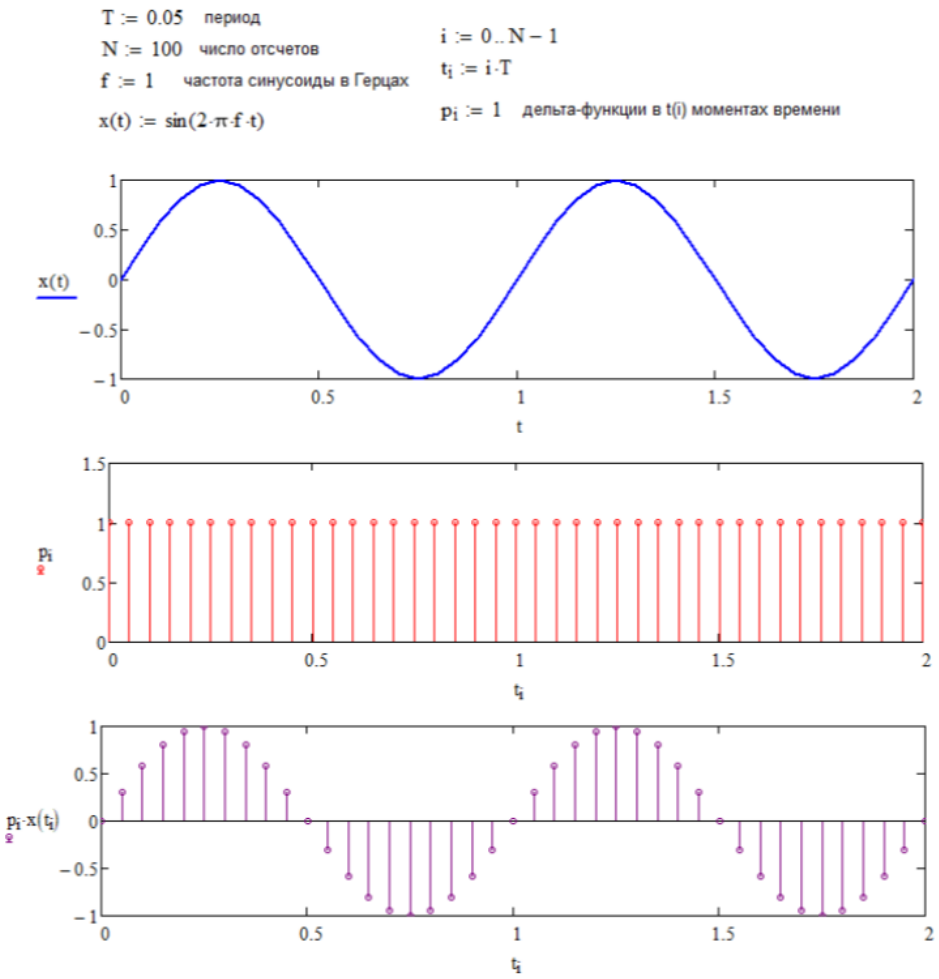

Рисунок 5 — Процедура временной дискретизации аналогового сигнала в ходе АЦП

были взяты с частотой дискретизации, превышающей максимальную частоту сигнала минимум в два раза.

Рассмотрим на примере, что произойдет, если частота дискретизации $\omega_s = 2\pi/T$ будет выбрана меньшей, чем требуется в соответствии с теоремой Котельникова (рисунок 6). Справа показаны спектры временных сигналов, полученные путем выполнения прямого преобразования Фурье (ППФ) над временными отсчетами.

Видно, что, если дискретизация выполняется недостаточно быстро, результирующая форма сигнала существенно искажается в частотной области, и соответственно во временной. Такой сигнал будет невозможно корректно восстановить.

Итак, в результате был получен дискретный сигнал (формула 7):

$$x[n] = x(t)|_{t=nT} = x(nT) \tag{7}$$

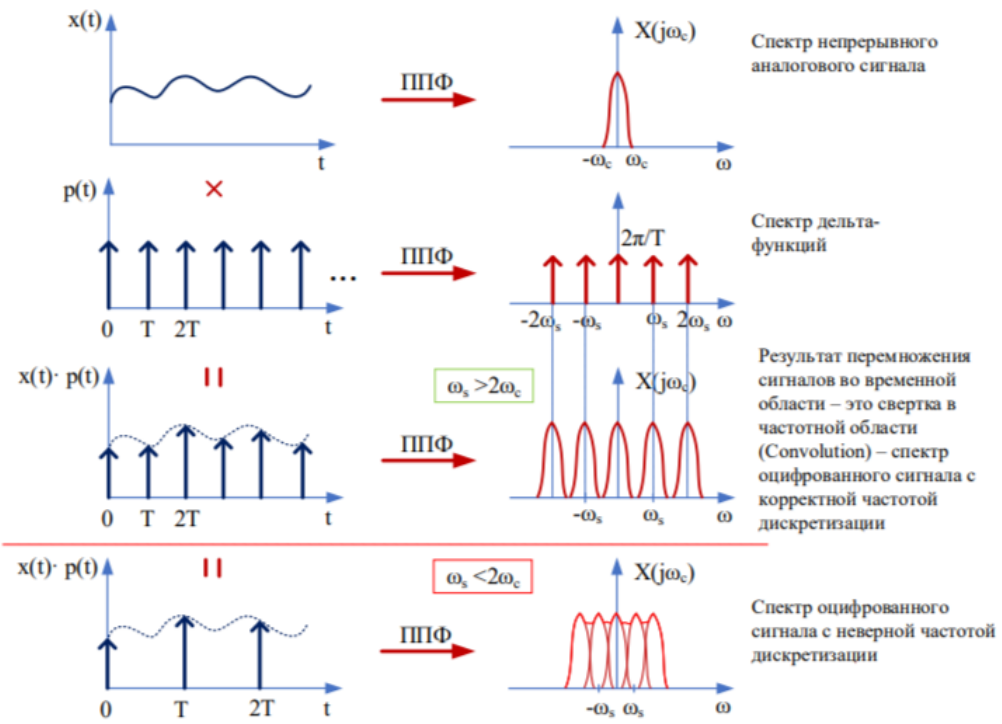

Рисунок 6 — Сравнение корректной и некорректной дискретизации сигнала

где n – номер дискретного отсчета.

2.6 Преобразования Фурье

Важнейшим утверждением для цифровой обработки сигналов стало утверждение математика Ж.Б.Фурье о том, что любой сколь угодно сложный сигнал можно представить в виде суммы простых сигналов, например, синусоид. В самом деле, если нажать на клавишу фортепиано, мы услышим какой-то простой звук, на определенной частоте, но при одновременном нажатии нескольких клавиш, звуки суммируются и получается более сложное звучание. Такая же логика применима к сигналам, используемым в радиосвязи, достаточно сложным, но все же поддающимся декомпозиции на сумму простых колебаний.

Основываясь на этой логике, математические преобразования Фурье позволяют переключаться между различными эквивалентными представлениями сигналов – временным и частотным. Однако, для простых сигналов не составляет труда перейти от временной к частотной форме. Рассмотрим для примера периодические функции синуса и косинуса и их частотное представление. Для начала вспомним, что данные функции можно записать в комплексном виде следующим образом (тождества Эйлера):

$$\cos(x) = \frac{1}{2}(e^{jx} + e^{-jx}); \quad \sin(x) = \frac{1}{2j}(e^{jx} - e^{-jx}) \tag{8}$$

Если мы заменим аргумент x на $2\pi ft$, то получим:

$$\cos(2\pi ft) = \frac{1}{2}(e^{j2\pi ft} + e^{-j2\pi ft}); \quad \sin(2\pi ft) = \frac{1}{2j}(e^{j2\pi ft} - e^{-j2\pi ft}) \tag{9}$$

Также можно заметить, что

$$\frac{1}{2j} = -\frac{1}{2}j = \frac{1}{2}e^{-j\frac{\pi}{2}} \tag{10}$$

Данные преобразования (8)-(10) позволяют увидеть, что функции синуса и косинуса, состоят их суммы/разности двух радиус-векторов, вращающихся со скоростью/частотой f в противоположных направлениях (направление вращения определяется знаком перед частотой сигнала). Попробуем изобразить данные сигналы во временной и частотной областях (рисунок 7).

В частотной области можно отобразить зависимость амплитуды от частоты сигналов. В нашем случае у обоих сигналов есть две компоненты на

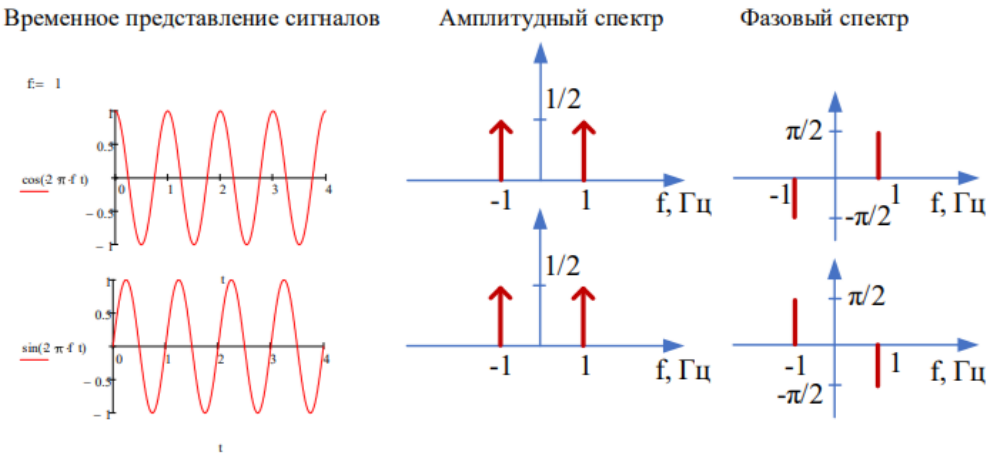

Рисунок 7 — Временное и частотное представление периодических сигналов $\cos(2\pi ft)$ и $\sin(2\pi ft)$

частотах f и $-f$, с амплитудами, равными $\sqrt{(\frac{1}{2})^2 + (\frac{1}{2})^2} = 1/2$, которые показаны в виде дельта-функций. Если изображать лишь зависимость амплитуды от частоты, то может показаться, что сигналы идентичны, но ведь очевидно, что они отличаются по начальной фазе. Поэтому для корректного представления необходимо дополнять частотное представление еще и зависимостью начальной фазы от частоты (фазовый спектр).

Обратите еще раз внимание на наличие отрицательных частот в спектрах сигнала. Отрицательная частота не имеет физического смысла, лишь математический – показывает на направление вращения вектора по часовой стрелке (отрицательное приращение угла).

Если сравнить спектры периодических сигналов на рисунке 7 со спектрами реальных сигналов на рисунке 6, то можно сделать предположение, что просуммировав некоторое число синусоид с различными параметрами (частотами, амплитудами и фазами) можно получить любой сколь угодно сложный сигнал. Именно на этой идее и построены преобразования Фурье, которые будут рассмотрены ниже.

Задача **прямого преобразования Фурье (ППФ)** – получить частотное представление радиосигнала, имея его временные значения. И наоборот – **обратное преобразование Фурье (ОПФ)**, зная частотную характеристику сигнала, восстанавливает его временной вид.

2.6.1 Быстрое преобразование Фурье

Существуют разновидности преобразования Фурье, называемые **быстрым прямым преобразованием Фурье (БППФ или FFT)** и **обратным быстрым преобразованием Фурье (ОБПФ или IFFT)**, которые, как правило, чаще используются в цифровой обработке сигналов, благодаря меньшим вычислительным затратам на их осуществление. Алгоритмы быстрого преобразования Фурье основываются на том, что в вычислениях есть много периодически повторяющихся значений (в силу периодичности функций синуса и косинуса). БПФ группирует слагаемые с одинаковыми множителями, существенно уменьшая число умножений за счет исключения повторных вычислений. В результате быстродействие БПФ может в сотни раз превосходить быстродействие стандартного алгоритма.

БПФ и ОБПФ, используемые в модемах мобильных устройств, это прежде всего преобразования дискретного во времени сигнала, базирующиеся на алгоритме Кули-Тьюки, при котором время вычисления для N отсчетов будет порядка $N \log_2 N$. Количество элементов (отсчетов временного сигнала, например), подаваемое на вход такой функции, должно быть равно $N = 2^k$, где $k = 0,1,..,\infty$.

Уравнение для ППФ имеет вид (11):

$$F(x) = \sum_{n=0}^{N-1} f(n) e^{-j2\pi(x\frac{n}{N})} \tag{11}$$

где N – размерность преобразования Фурье, $F(x)$ – сигнал в частотной области и $f(n)$ – дискретизированный сигнал во временной области.

ОПФ имеет вид (12):

$$f(n) = \frac{1}{N} \sum_{n=0}^{N-1} F(x) e^{j2\pi(x\frac{n}{N})} \tag{12}$$

И еще один немаловажный факт, можно обратить внимание на то, что на вход преобразования Фурье могут идти временные отсчеты в комплексной форме и не всегда понятно, откуда появились комплексные числа, если временной сигнал выглядит как что-то вещественное. Если вкратце, то перед

АЦП может стоять квадратурный демодулятор, домножающий вещественный входной сигнал на функции синуса и косинуса и выдавая на вход АЦП мнимую и реальную составляющую сигналов, с которыми и работает АЦП.

3 ВЫПОЛНЕНИЕ РАБОТЫ

3.1 Ход работы

1. Сгенерируем и визуализируем непрерывный периодический сигнал в соответствии с вариантом.

23	$y(t) = \sin(2\pi f t) + \sin(10\pi f t), \quad f = 13$

Рисунок 8 — Вариант 23

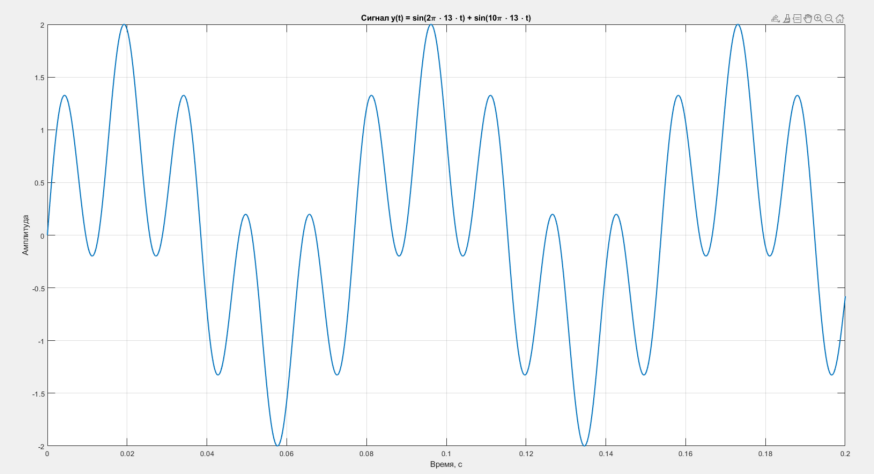

Рисунок 9 — непрерывный сигнал

2. Определим максимальную частоту в спектре сигнала.

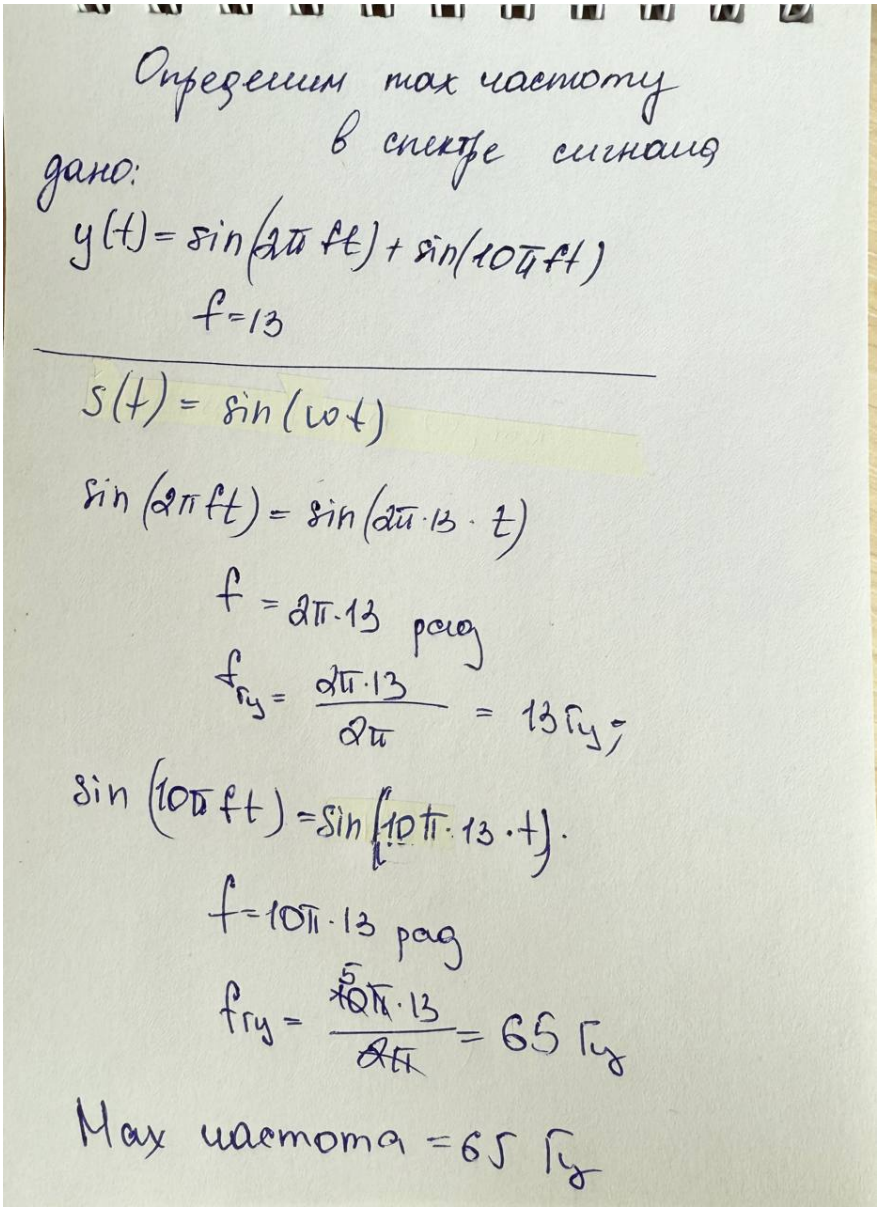

Рисунок 10 — Решение

Максимальная частота равна 65 герцам.

3. Рассчитаем минимальную частоту дискретизации по теореме Котельникова. теорема Котельникова : fs > 2*fmax

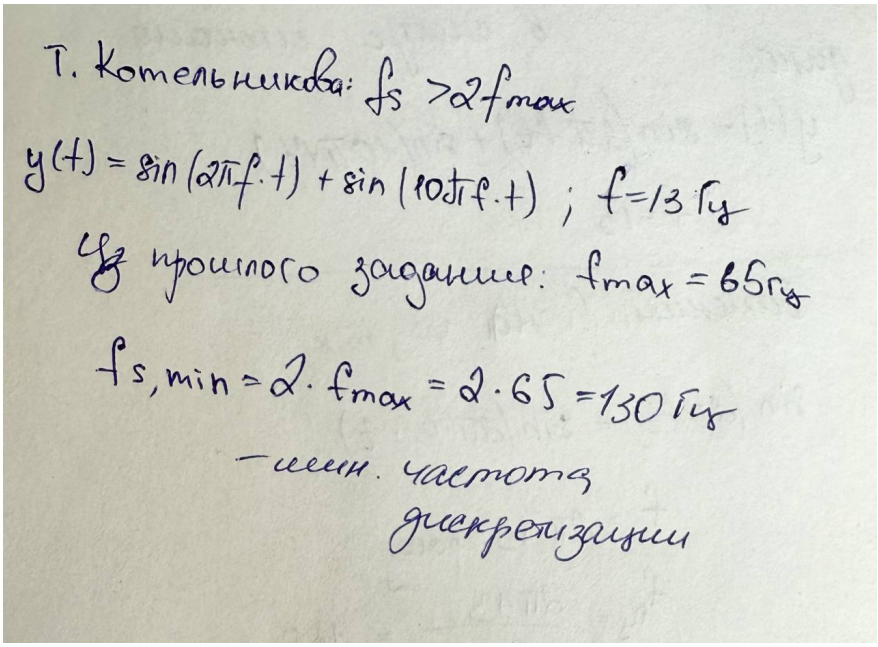

Рисунок 11 — Решение

Минимальная частота дискретизации равна 130 Герц. чтобы получить более подробные результаты бес потерь в амплитудном спетре возьмем минимальную частоту дискретизации +1.

4. Выполним оцифровку сигнала и прямое дискретное преобразование Фурье (ПДПФ). Исходный сигнал задан функцией:

$$y(t) = \sin(2\pi f t) + \sin(10\pi f t), \quad f = 13 \text{ Гц} \tag{13}$$

Из предыдущих этапов определены следующие параметры:

- Максимальная частота в спектре:

$$f_{\max} = 65 \text{ Гц} \tag{14}$$

- Минимальная частота дискретизации по теореме Котельникова:

$$f_s = 2 \cdot f_{\max} = 2 \cdot 65 = 130 \text{ Гц} \tag{15}$$

Для оцифровки сигнала используются следующие параметры:

- Длительность сигнала:

$$T = 1 \text{ с} \tag{16}$$

- Частота дискретизации:

$$f_s = 130 \text{ Гц} \tag{17}$$

– Период дискретизации:

$$T_s = \frac{1}{f_s} = \frac{1}{130} \approx 0.007692 \text{ с} \tag{18}$$

– Число отсчётов:

$$N = f_s \cdot T = 130 \cdot 1 = 130 \text{ отсчётов} \tag{19}$$

Таблица 1 — Параметры оцифровки сигнала

Параметр	**Значение**	**Единицы измерения**
Частота сигнала f	13	Гц
Максимальная частота $f_{\max}$	65	Гц
Частота дискретизации f_s	130	Гц
Период дискретизации T_s	0.007692	с
Длительность сигнала T	1	с
Число отсчётов N	130 +1	отсчётов

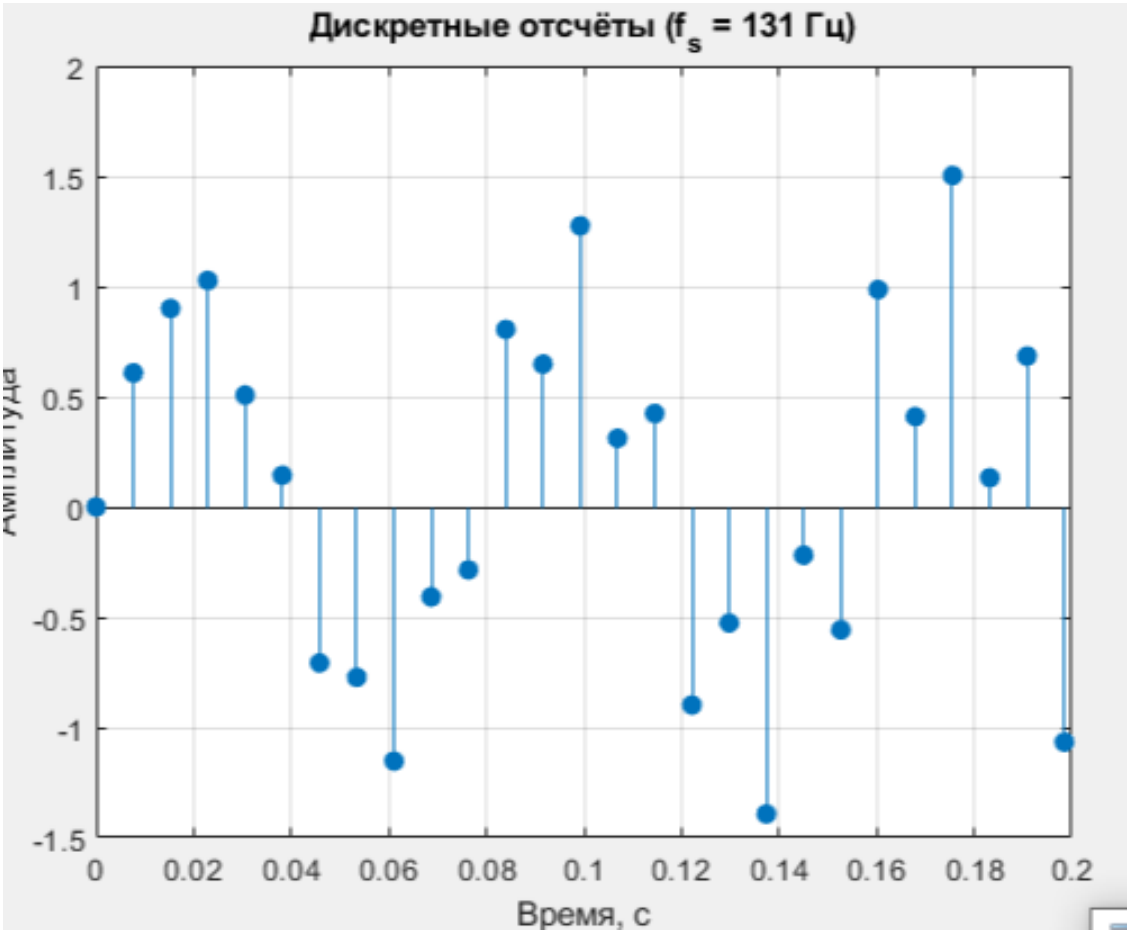

Рисунок 12 — оцифрованный сигнал

5. Выполним прямое дискретное преобразование Фурье для массива временных отсчетов сигнала и оценим ширину данного спектра

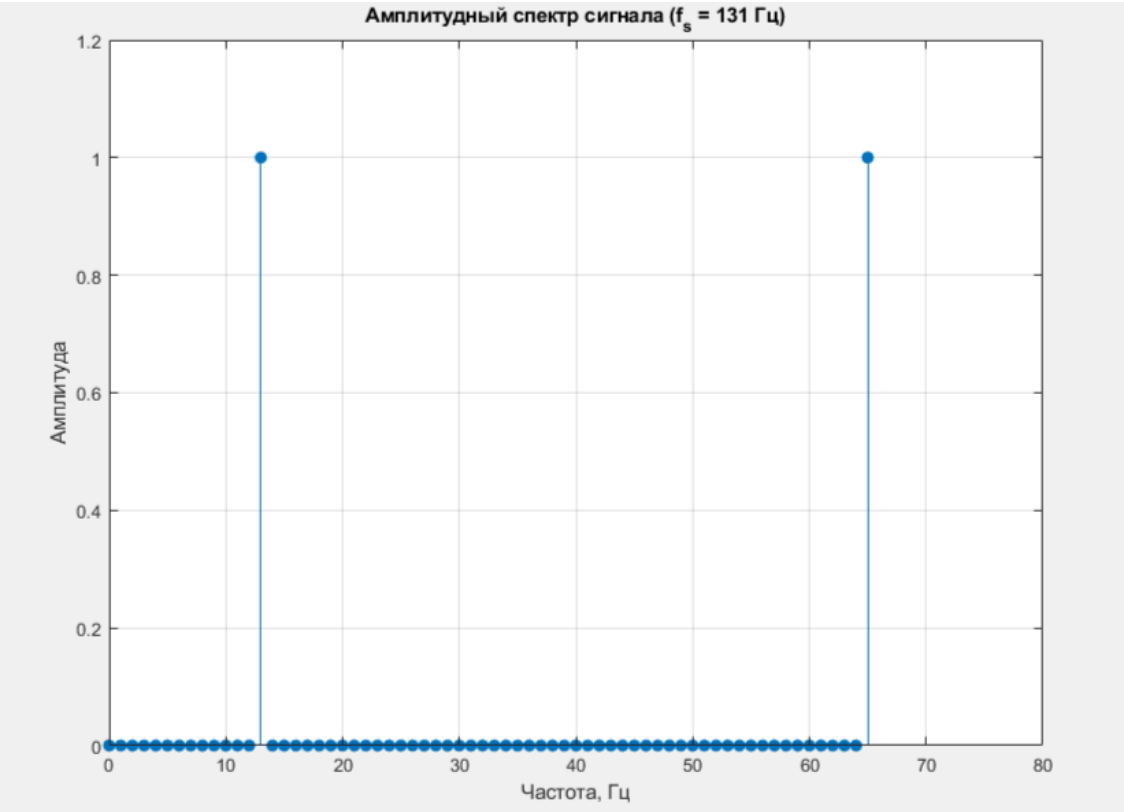

Рисунок 13 — амплитудный спектр сигнала

```
Ширина спектра: 52 Гц (от 13 до 65 Гц)
Объём памяти (float64): 1048 байт
Объём памяти (float32): 524 байт
```

Рисунок 14

6. Восстановим аналоговый сигнал по дискретным отсчётам.

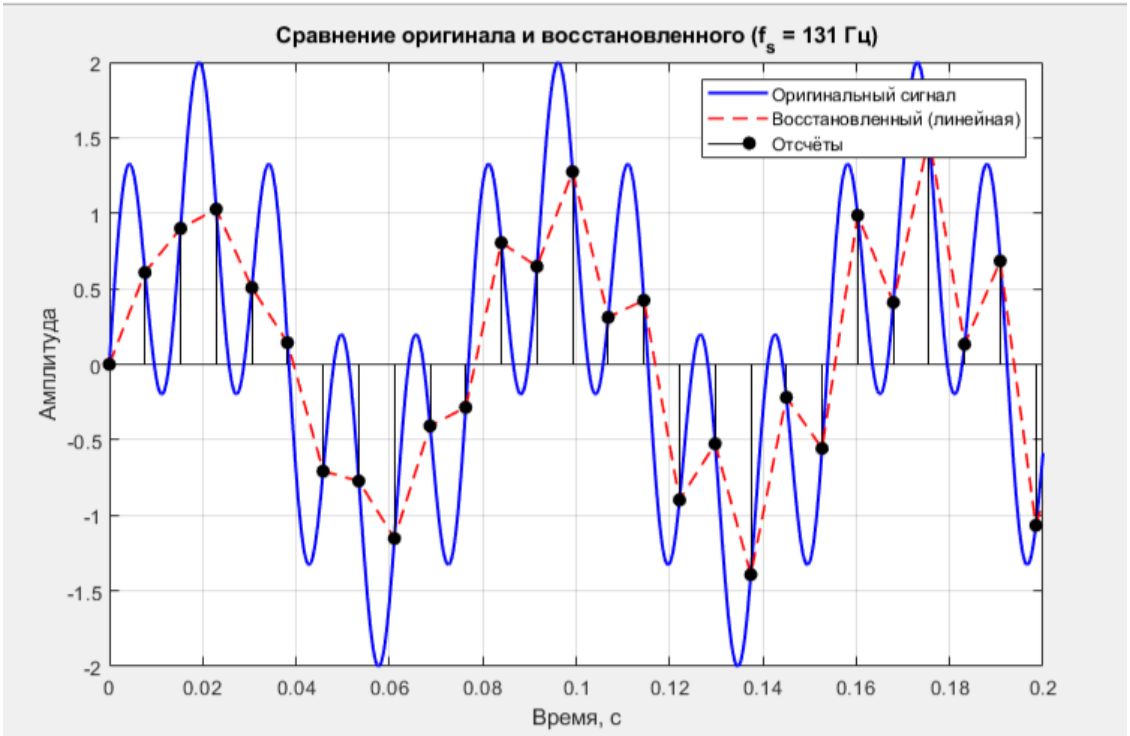

Рисунок 15 — восстановленный сигнал

7. Увеличим частоту дискретизации в 4 раза и проделайте задания из п.4-6.

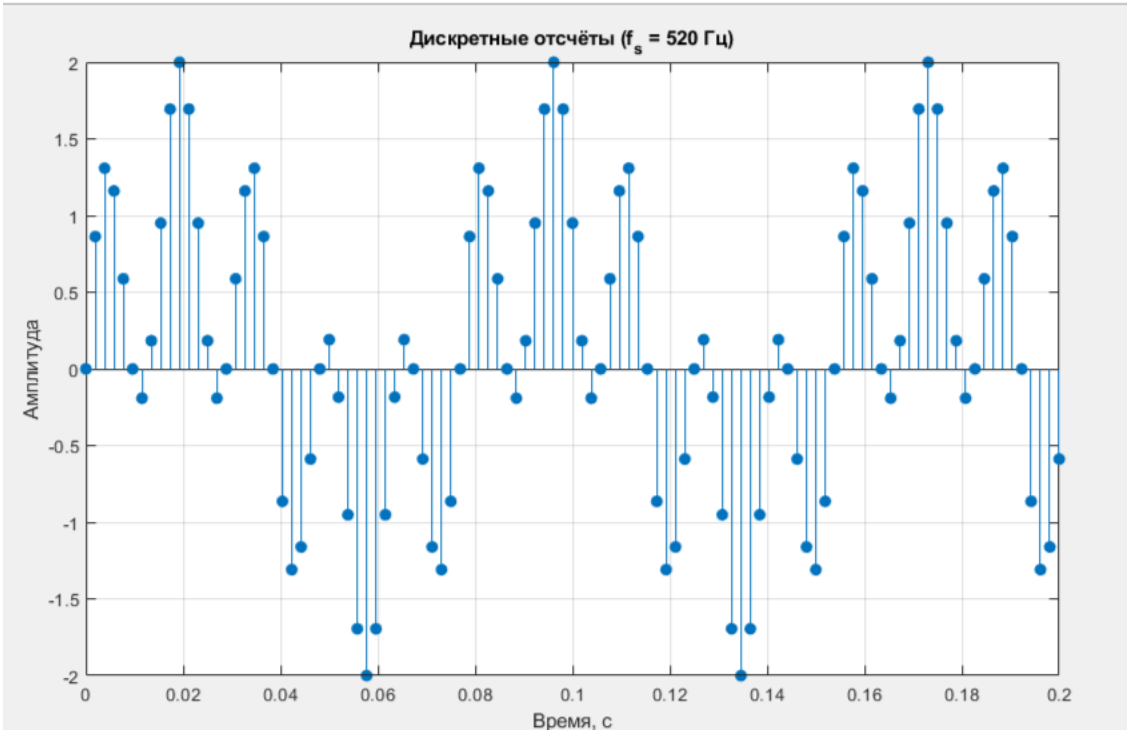

Рисунок 16 — оцифрованный сигнал

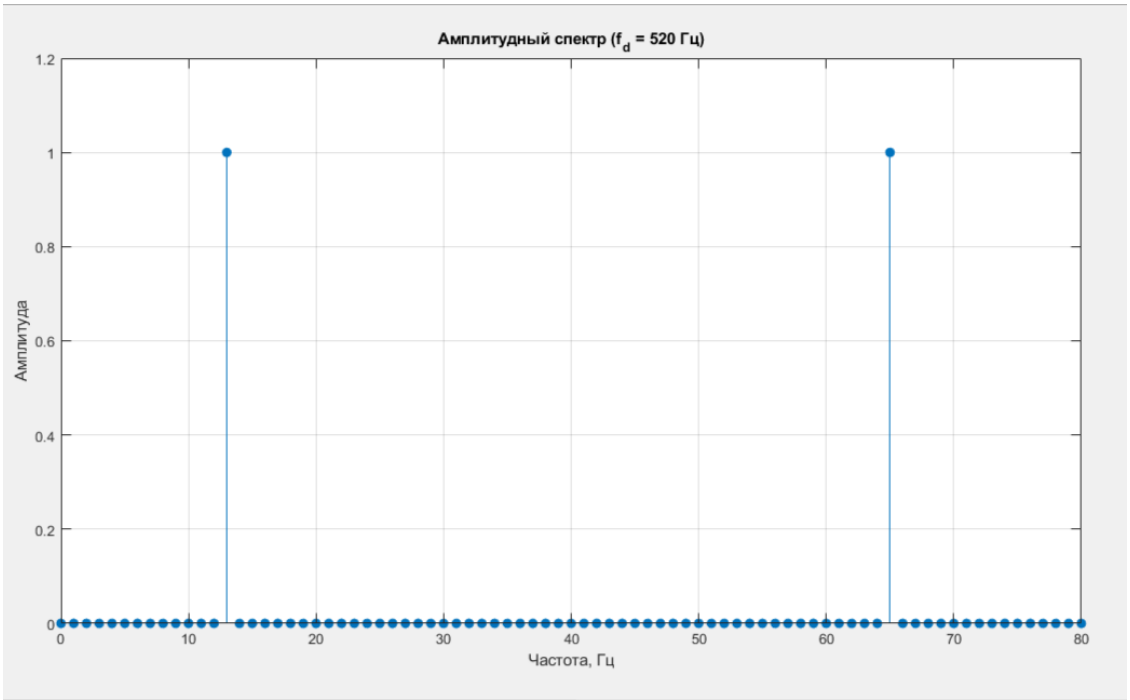

Рисунок 17 — амплитудный спектр сигнала

```
 Результаты при f_d = 520 Гц (в 4 раза выше)
Ширина спектра: 52 Гц (от 13 до 65 Гц)
Объём памяти (float64): 4160 байт
```

Рисунок 18

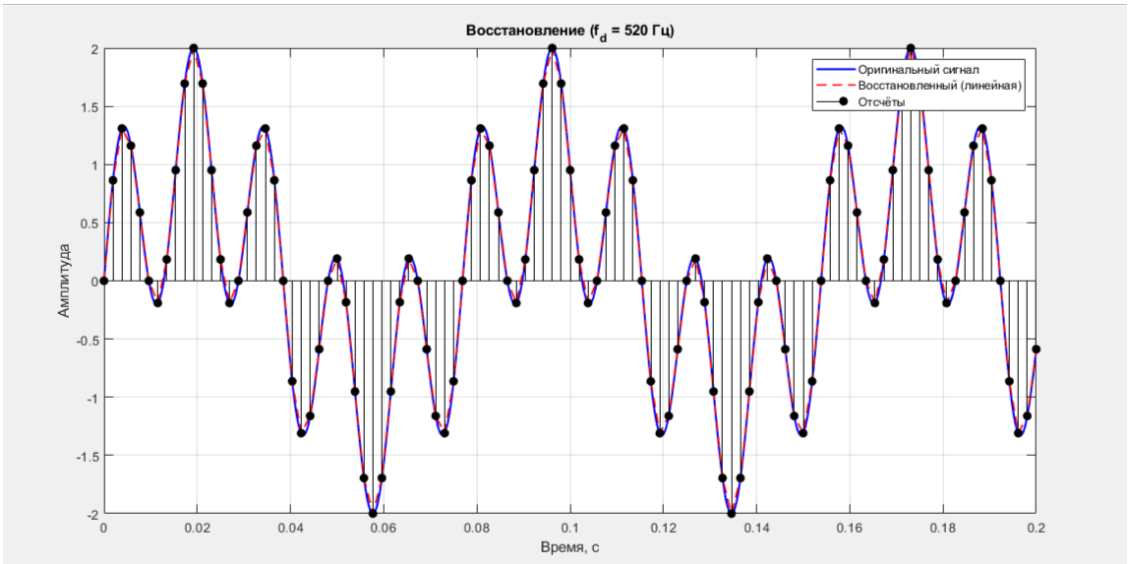

Рисунок 19 — восстановленный сигнал

8. Запишем аудиофайл со своим голосом.Проанализируем визуально спектр голоса.

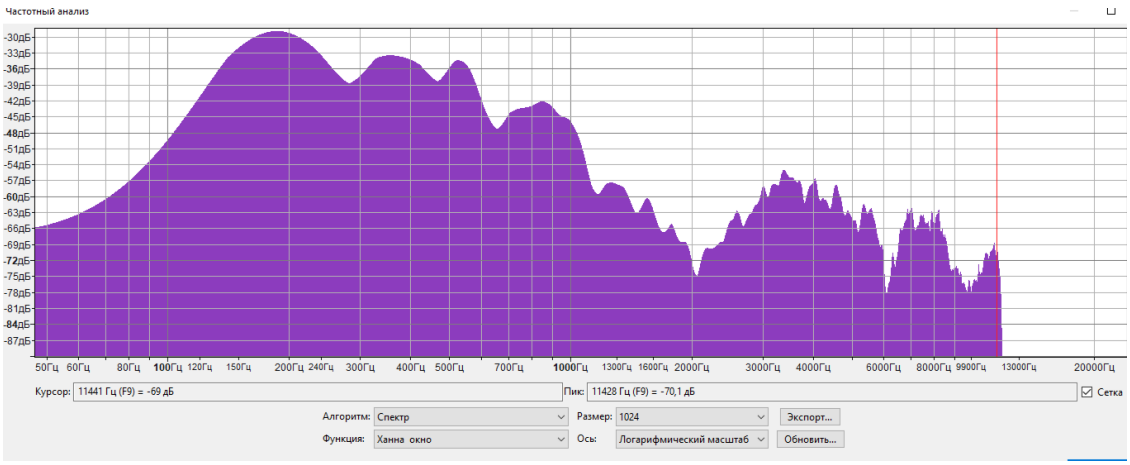

Рисунок 20 — частотный спектр голоса

9. Определим максимальную частоту в спектре данного сигнала. По рисунку выше видно, что максимальная частота равно чуть больше чем 11,8 килогерц. Возьмем максимальную частоту 12КГц.
10. Выберем требуемую для оцифровки частоту дискретизации. По теореме Котельникова, которую мы использовали выше, минимальная частота дискретизации должна быть в два раза больше чем максимальная частота в спектре. Соответсвенно, 2*12КГц = 24 КГц - минимальная частота дискретизации.
11. Проанализируем аудиосигнал (голос), определим частоту дискретизации, которая была использована при записи голоса на цифровой носитель.

```
Файл найден!
Размер массива y: 301056 x 1
Частота дискретизации Fs: 44100 Гц
Число отсчётов (N): 301056
Длительность записи: 6.827 сек
Частота дискретизации (расчётная): 44100.0 Гц
Значения совпадают.
```

Рисунок 21 — результаты

Для корректных вычислений пропишу в коде проверку, что файл найден. Полученные результаты: минимальная частота дискретизации 44.1 килогерца, совпадает с расчетной. Проверим вычисления - Для этого возьмем количество элементов в файле с записью(301056)

и разделим на длительность записи в секундах(6,83с).Частота дискретизации Fs будет равна 301056/6,82 = 44143 Гц. Можем сделать вывод, что вычисления корректны.

12. Проредим полученный массив y(уменьшим частоту дискретизации и воспроизведите полученный сигнал). Возьмем коэффициент прореживания 10. качество звука стало хуже.
13. Выполним прямое дискретное преобразование Фурье для оригинального звучания и для прореженного сигнала, выведем на график амплитудный спектр сигнала, определим его ширину.

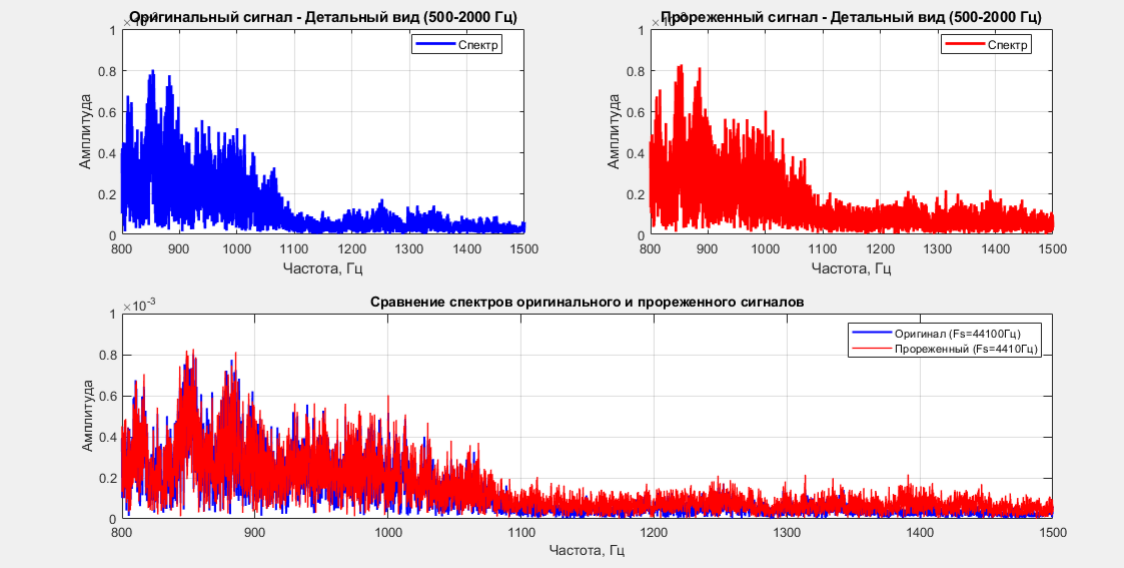

Рисунок 22 — сравнение амплитудных спектров

Ширина спектра оригинального сигнала: 8277.2 Гц (от 98.4 до 8375.7 Гц) Максимальная частота в спектре: 8375.7 Гц Ширина спектра прореженного сигнала: 2199.9 Гц (от 4.4 до 2204.3 Гц) Максимальная частота в спектре: 2204.3 Гц

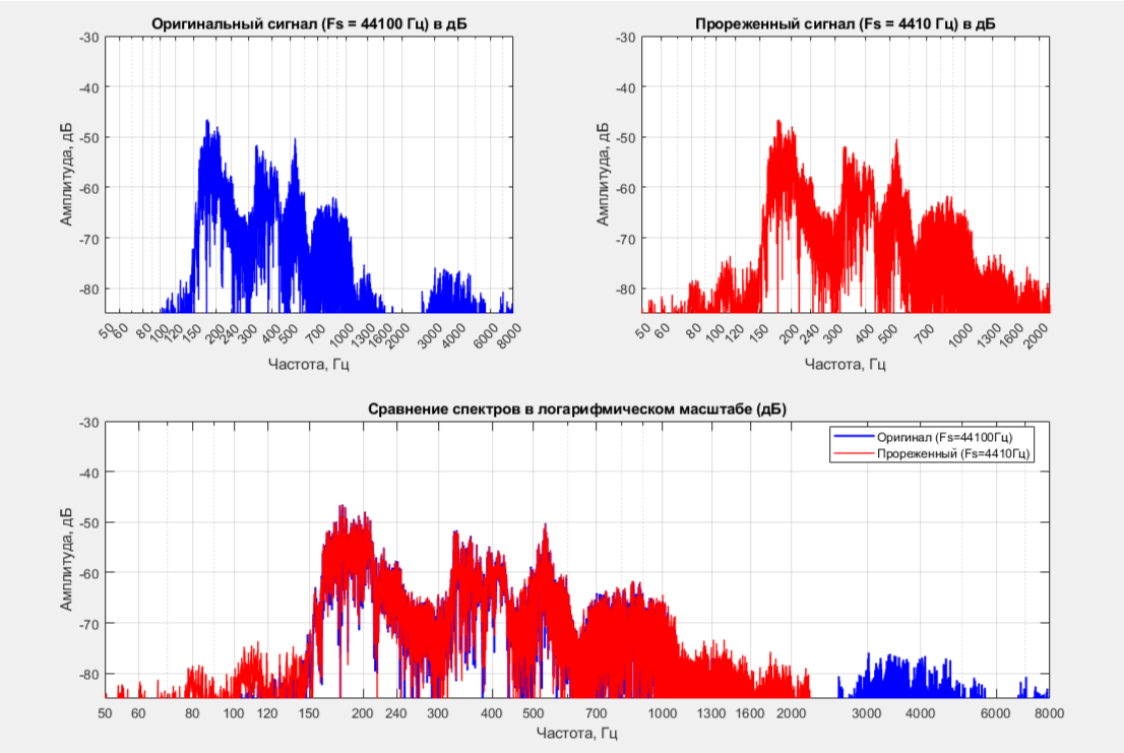

Рисунок 23 — сравнение амплитудных спектров в децибелах

14. Оценим влияние разрядности АЦП на спектр сигнала. Для этого напишем функцию, которая округляет значения отсчетов сигнала до какого-то числа, определяемого разрядностью АЦП. Для результирующего дискретного сигнала выполним прямое преобразование Фурье. Сравним полученный спектр со спектром исходной синусоиды, отсчеты которой не подвергались квантованию по уровню.

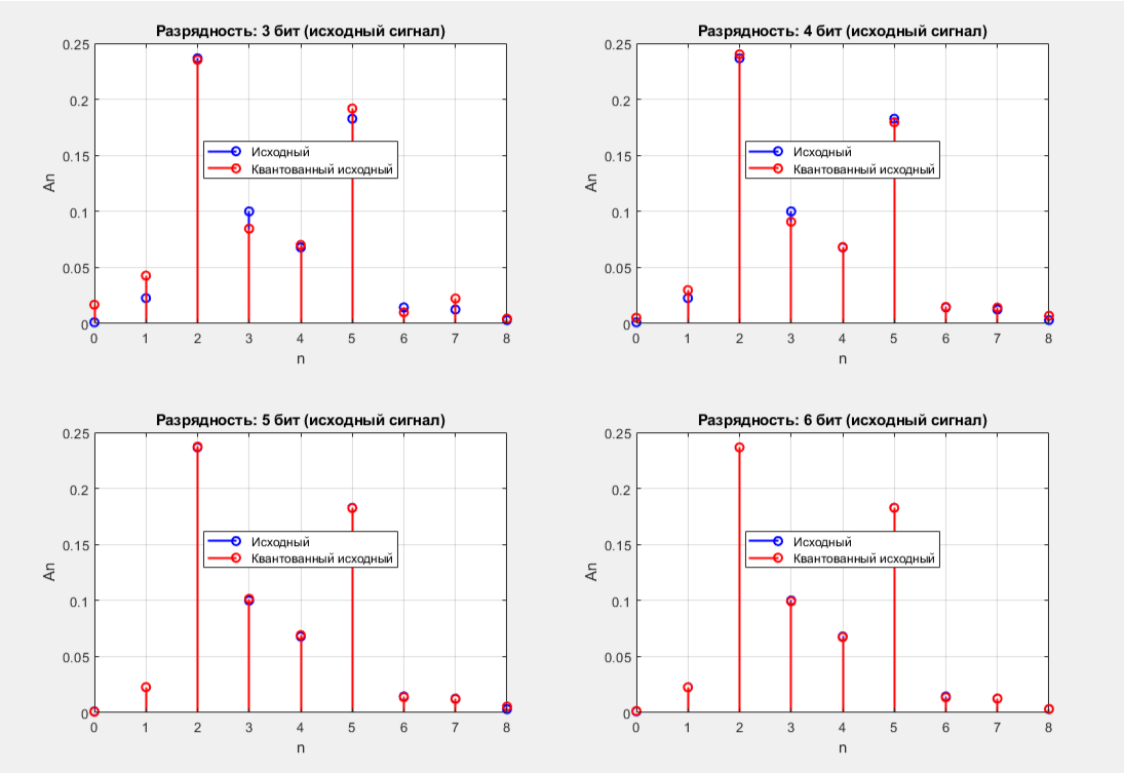

Рисунок 24 — влияние разрядности АЦП на спектр сигнала

15. Выведем среднюю ошибку квантования для случаев, когда разрядность АЦП равна 3/4/5/6.

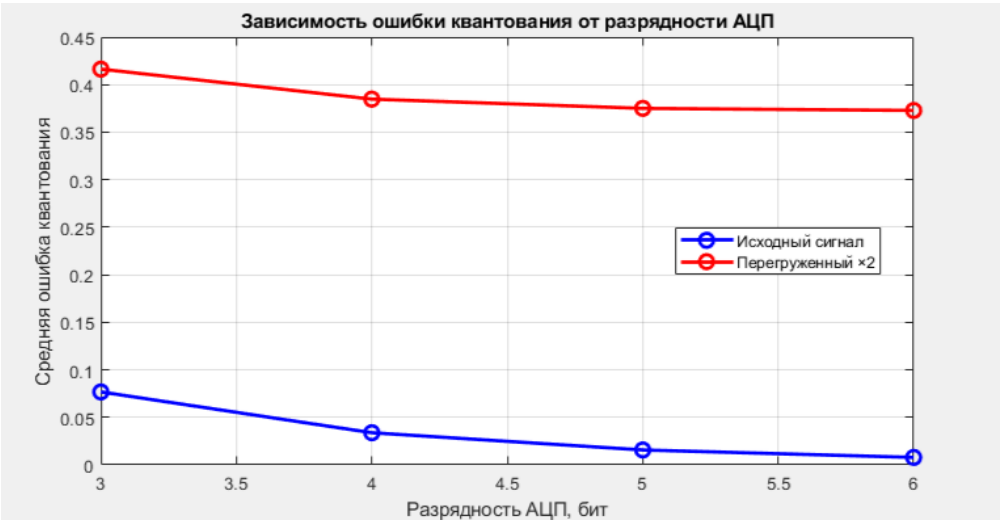

Рисунок 25 — средняя ошибка квантования для исходного и перегруженного сигнала

```
Средняя ошибка квантования для ИСХОДНОГО сигнала:
Разрядность 3 бит: 0.076722
Разрядность 4 бит: 0.033789
Разрядность 5 бит: 0.015683
Разрядность 6 бит: 0.007842

Средняя ошибка квантования для ПЕРЕГРУЖЕННОГО сигнала (×2.0):
Разрядность 3 бит: 0.416306
Разрядность 4 бит: 0.384733
Разрядность 5 бит: 0.375009
Разрядность 6 бит: 0.372826
```

Рисунок 26 — средняя ошибка квантования для исходного и перегруженного сигнала

16. Выведем исходный сигнал и оценим вляние разрядности АЦП на исходный сигнал.

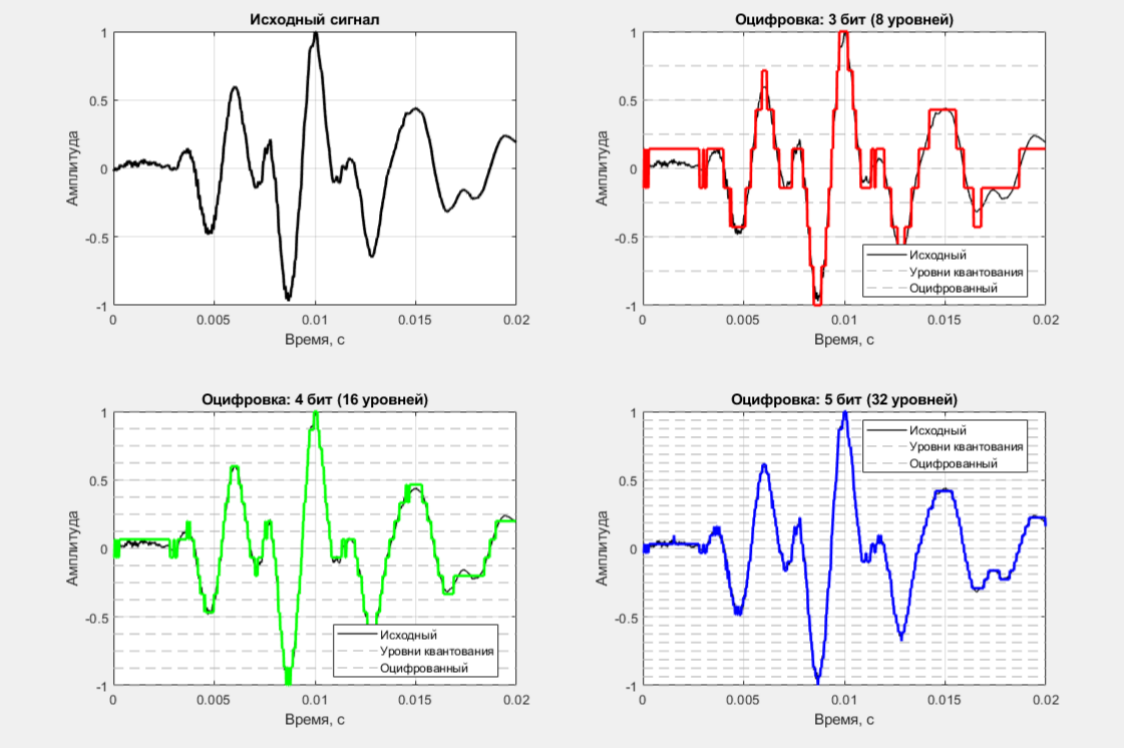

Рисунок 27 — АЦП

17. Выведем перегруженный сигнал и оценим вляние разрядности АЦП на перегруженный сигнал.

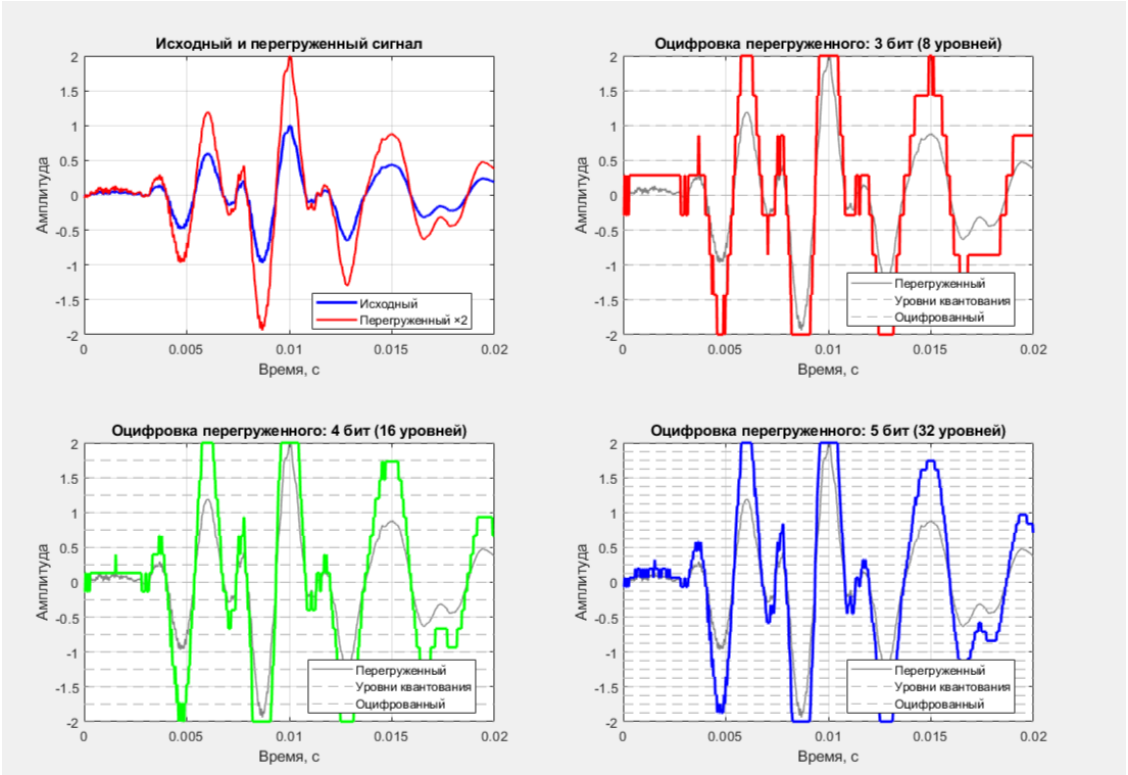

Рисунок 28 — АЦП

4 ВЫВОДЫ И ОТВЕТЫ НА КОНТРОЛЬНЫЕ ВОПРОСЫ

4.1 Выводы по практической работе

В ходе выполнения практической работы были получены следующие выводы:

1. **Теорема Котельникова**
   При проведении работы мы подтвердили, что для восстановления сигнала частота дискретизации должна быть не меньше чем две максимальных частоты в спектре сигнала. Чем выше частота дискретизации, тем выше точность восстановленного сигнала относительно исходного.
2. **Преобразование Фурье**
   Прямое и обратное преобразование Фурье позволяет преобразовать данные из временной области в частотную, анализировать частотный состав сигналов или функций, которые изменяются во времени или пространстве.
3. **Влияние частоты дискретизации**
   При недостаточной частоте дискретизации сигнал очень сильно искажается. Увеличение частоты дискретизации в 4 раза практически полностью устраняет визуальные различия между исходным и восстановленным сигналами.
4. **Восстановление сигналов**
   Точность восстановления напрямую зависит от соблюдения теоремы Котельникова. Визуальное сравнение является эффективным методом оценки качества дискретизации.
5. **Прореживание сигналов**
   Уменьшение частоты дискретизации в 10 раз приводит к существенному ухудшению качества звука, к изменению спектральных характеристик.
6. **Заключение** Работа показала, что правильный выбор частоты дискретизации и разрядности АЦП критически важен для сохранения качества сигнала. Теорема Котельникова и преобразование Фурье

— это не просто теория, а практические инструменты, без которых невозможно проектировать современные системы связи, аудиообработки и цифровой обработки сигналов.

4.2 Ответы на контрольные вопросы

1. **Для чего используются прямое и обратное преобразование Фурье?**
   Прямое преобразование Фурье используется для перехода из временной области в частотную, что позволяет анализировать спектральный состав сигнала. Обратное преобразование Фурье выполняет обратную операцию — восстанавливает сигнал во временной области по его спектральным характеристикам.
2. **Что такое ошибка квантования и дискретизации?**
   Ошибка дискретизации возникает при недостаточной частоте взятия отсчетов (нарушение теоремы Котельникова). Ошибка квантования обусловлена конечной разрядностью АЦП и представляет собой разницу между истинным значением сигнала и его квантованным представлением.
3. **Какое количество разрядов АЦП требуется, чтобы оцифровать голос?**
   Для качественной оцифровки голоса рекомендуется использовать АЦП с разрядностью не менее 8 бит. Для профессиональных применений оптимальной является разрядность 16 бит.
4. **Как математически получить дискретные отсчеты непрерывного сигнала?**
   Дискретные отсчеты получаются путем умножения непрерывного сигнала $x(t)$ на последовательность дельта-функций с периодом T_s (интервал дискретизации):

$$x_d(t) = x(t) \cdot \sum_{n=-\infty}^{\infty} \delta(t - nT_s)$$

5. **Какой спектр у периодического сигнала** $\sin(10\pi t + \pi/2)$**?**
   Данный сигнал имеет одну спектральную компоненту на частоте 5 Гц с амплитудой 1 и фазой $\pi/2$.

6. **Что такое быстрое преобразование Фурье?**
   Быстрое преобразование Фурье (БПФ) — это алгоритм вычисления дискретного преобразования Фурье с вычислительной сложностью $O(N \log N)$ вместо $O(N^2)$ для прямого вычисления, что делает его значительно более эффективным для больших массивов данных.
7. **Как определяется минимальная требуемая для оцифровки частота дискретизации сигнала?**
   Согласно теореме Котельникова, минимальная частота дискретизации f_s должна быть не менее чем в два раза выше максимальной частоты f_{max} в спектре сигнала:

$$f_s \geq 2 \cdot f_{max}$$

5 СКРИПТЫ MATLAB

5.1 Скрипт 1 - реализация сигналов

```
    f = 13;  % Гц

%% непрерывный сигнал
t = 0:0.0001:1;
y = sin(2*pi*f*t) + sin(10*pi*f*t);

figure('Name', '1. Непрерывный сигнал');
plot(t, y, 'LineWidth', 1.5);
title(['Сигнал y(t) = sin(2\pi \cdot ', num2str(f), ' \cdot t) + sin
    (10\pi \cdot ', num2str(f), ' \cdot t)']);
xlabel('Время, с');
ylabel('Амплитуда');
grid on;
xlim([0, 0.2]);

%% Макс частота и частота дискретизации по Котельникову
f1 = f;           % 13 Гц
f2 = 5 * f;       % 65 Гц тк(.. sinп(10ft) = sinп(2·5f·t))
f_max = max(f1, f2);  % 65 Гц
fd_min = 2 * f_max;   % 130 Гц

fprintf('Максимальная частота в спектре: %d Гц\n', f_max);
fprintf('Минимальная частота дискретизации Котельников(): %d Гц\n',
    fd_min);

%% Оцифровка сигнала дискретные( отсчёты)
fd = fd_min +1;
T = 1;            % 1 секунда
N = fd * T;  % 131 отсчёт

td = (0:N-1) / fd;
yd = sin(2*pi*f*td) + sin(10*pi*f*td);

figure('Name', '4. Дискретные отсчёты (fs = 131 Гц)');
stem(td, yd, 'filled');
title(['Дискретные отсчёты (f_s = ', num2str(fs), ' Гц)']);
xlabel('Время, с');
ylabel('Амплитуда');
```

```
grid on;
xlim([0, 0.2]);

%%  Прямое ДПФ (FFT), ширина спектра и объём памяти
Y = fft(yd);

f_axis = (0:N/2) * fd / N;
Y_mag = abs(Y(1:N/2+1)) / N * 2;
Y_mag(1) = Y_mag(1) / 2;  % постоянная составляющая

spectrum_width = f2 - f1;  % 52 Гц

memory_64 = N * 8;    % байт

fprintf('\n Результаты ПДПФ при f_s = %d Гц \n', fs);
fprintf('Ширина спектра: %d Гц от( %d до %d Гц)\n', spectrum_width,
    f_min, f_max);
fprintf('Объём памяти (float64): %d байт\n', memory_64);

figure('Name', '5. Амплитудный спектр (fs = 131 Гц)');
stem(f_axis, Y_mag, 'filled');
title(['Амплитудный спектр сигнала (f_s = ', num2str(fs), ' Гц)']);
xlabel('Частота, Гц');
ylabel('Амплитуда');
grid on;
xlim([0, 80]);
ylim([0, 1.2]);

%% Восстановление сигнала
t_interp = 0:0.0001:0.2;
y_interp = interp1(td, yd, t_interp, 'linear');

y_orig = sin(2*pi*f*t_interp) + sin(10*pi*f*t_interp);

figure('Name', '6. Восстановление (fs = 131 Гц)');
plot(t_interp, y_orig, 'b-', 'LineWidth', 1.5); hold on;
plot(t_interp, y_interp, 'r--', 'LineWidth', 1.2);
stem(td(td <= 0.2), yd(td <= 0.2), 'k', 'filled');
legend('Оригинальный сигнал', 'Восстановленный линейная()', 'Отсчёты');
xlabel('Время, с');
ylabel('Амплитуда');
title(['Сравнение оригинала и восстановленного (f_s = ', num2str(fs), '
    Гц)']);
grid on;
```

```
xlim([0, 0.2]);
%% пункт 7
fd2 = fd_min*4;
N2 = round(fd2 *T);
t2 = (0:N2-1) / fd2;
y2 = sin(2*pi*f*t2) + sin(10*pi*f*t2);

figure('Name', '4. Дискретные отсчёты (fs = 520 Гц)');
stem(t2, y2, 'filled');
title(['Дискретные отсчёты (f_s = ', num2str(fd2), ' Гц)']);
xlabel('Время, с');
ylabel('Амплитуда');
grid on;
xlim([0, 0.2]);

% ДПФ для fd2
Y2 = fft(y2);
f_axis2 = (0:N2/2) * fd2 / N2;
Y_mag2 = abs(Y2(1:N2/2+1)) / N2 * 2;
Y_mag2(1) = Y_mag2(1) / 2;

% Восстановление
y2_interp = interp1(t2, y2, t_interp, 'linear');

% Вывод
memory_64_2 = N2 * 8;
fprintf('\n Результаты при f_d = %d Гц в( 4 раза выше) \n', fd2);
fprintf('Ширина спектра: %d Гц от( %d до %d Гц)\n', spectrum_width,
    f_min, f_max);
fprintf('Объём памяти (float64): %d байт\n', memory_64_2);

% График спектра для fd2
figure('Name', '7a. Спектр при fd = 520 Гц');
stem(f_axis2, Y_mag2, 'filled');
title(['Амплитудный спектр (f_d = ', num2str(fd2), ' Гц)']);
xlabel('Частота, Гц');
ylabel('Амплитуда');
grid on;
xlim([0, 80]);
ylim([0, 1.2]);

% График восстановления для fd2
figure('Name', '7b. Восстановление при fd = 520 Гц');
plot(t_interp, y_orig, 'b-', 'LineWidth', 1.5); hold on;
```

```
plot(t_interp, y2_interp, 'r--', 'LineWidth', 1.2);
stem(t2(t2 <= 0.2), y2(t2 <= 0.2), 'k', 'filled');
legend('Оригинальный сигнал', 'Восстановленный линейная()', 'Отсчёты');
xlabel('Время, с');
ylabel('Амплитуда');
title(['Восстановление (f_d = ', num2str(fd2), ' Гц)']);
grid on;
xlim([0, 0.2]);
```

Листинг 5.1 — Код лабораторной работы на MATLAB

5.2 Скрипт 2 - работа с голосом

```
[y, Fs] = audioread('C:\Users\KL\DesktopСибГУТИ\\3 курсОСМС\\lab1\voice
    .wav');

if exist('voice.wav', 'file') == 2
    fprintf('Файл найден!\n');
    [y, Fs] = audioread('voice.wav');
else
    fprintf('Файл не найден в текущей папке\n');
    fprintf('Текущая папка: %s\n', pwd);
end
fprintf('Размер массива y: %d x %d\n', size(y, 1), size(y, 2));
fprintf('Частота дискретизации Fs: %d Гц\n', Fs);

N = length(y);
duration = N / Fs;
Fs_calc = N / duration;

fprintf('Число отсчётов (N): %d\n', N);
fprintf('Длительность записи: %.3f сек\n', duration);
fprintf('Частота дискретизации расчётная(): %.1f Гц\n', Fs_calc);

y1 = downsample(y, 10);
Fs_new = Fs / 10;

fprintf('\Исходнаяn частота дискретизации: %d Гц\n', Fs);
fprintf('Коэффициент прореживания: 10\n');
fprintf('Новая частота дискретизации: %d Гц\n', Fs_new);
fprintf('Исходное количество отсчетов: %d\n', length(y));
fprintf('После прореживания: %d отсчетов\n', length(y1));

fprintf('\Воспроизведениеn прореженного сигнала...\n');
```

```
audio = audioplayer(y1, Fs_new);
play(audio);

Y_orig = fft(y);
L_orig = length(Y_orig);
P2_orig = abs(Y_orig/L_orig);
P1_orig = P2_orig(1:L_orig/2+1);
P1_orig(2:end-1) = 2*P1_orig(2:end-1);
f_orig = Fs*(0:(L_orig/2))/L_orig;

Y_down = fft(y1);
L_down = length(Y_down);
P2_down = abs(Y_down/L_down);
P1_down = P2_down(1:L_down/2+1);
P1_down(2:end-1) = 2*P1_down(2:end-1);
f_down = Fs_new*(0:(L_down/2))/L_down;

P1_orig_db = 20*log10(P1_orig + eps);
P1_down_db = 20*log10(P1_down + eps);

threshold_orig_db = max(P1_orig_db) - 40;
mask_orig = P1_orig_db > threshold_orig_db;
f_min_orig = f_orig(find(mask_orig, 1, 'first'));
f_max_orig = f_orig(find(mask_orig, 1, 'last'));
bandwidth_orig = f_max_orig - f_min_orig;

threshold_down_db = max(P1_down_db) - 40;
mask_down = P1_down_db > threshold_down_db;
f_min_down = f_down(find(mask_down, 1, 'first'));
f_max_down = f_down(find(mask_down, 1, 'last'));
bandwidth_down = f_max_down - f_min_down;

figure('Name', '12. Сравнение амплитудных спектров', 'Position', [100,
    100, 1200, 800]);

subplot(2,2,1);
plot(f_orig, P1_orig, 'b', 'LineWidth', 1);
title(['Оригинальный сигнал (Fs = ' num2str(Fs) ' Гц)']);
xlabel('Частота, Гц'); ylabel('Амплитуда');
xlim([0, 8000]);
grid on;
hold on;
plot([f_min_orig, f_max_orig], [max(P1_orig)*0.5, max(P1_orig)*0.5], 'r
    --', 'LineWidth', 2);
```

```
73 legend('Спектр', 'Ширина спектра');
74
75 subplot(2,2,2);
76 plot(f_down, P1_down, 'r', 'LineWidth', 1);
77 title(['Прореженный сигнал (Fs = ' num2str(Fs_new) ' Гц)']);
78 xlabel('Частота, Гц'); ylabel('Амплитуда');
79 xlim([0, Fs_new/2]);
80 grid on;
81 hold on;
82 plot([f_min_down, f_max_down], [max(P1_down)*0.5, max(P1_down)*0.5], 'b
    --', 'LineWidth', 2);
83 legend('Спектр', 'Ширина спектра');
84
85 subplot(2,2,[3,4]);
86 plot(f_orig, P1_orig, 'b', 'LineWidth', 1.5); hold on;
87 plot(f_down, P1_down, 'r', 'LineWidth', 1);
88 title('Сравнение спектров оригинального и прореженного сигналов');
89 xlabel('Частота, Гц'); ylabel('Амплитуда');
90 xlim([0, 8000]);
91 legend(['Оригинал (Fs=' num2str(Fs) 'Гц)'], ['Прореженный (Fs=' num2str
    (Fs_new) 'Гц)']);
92 grid on;
93
94 freq_ticks = [50, 60, 80, 100, 120, 150, 200, 240, 300, 400, 500, 700,
    1000, 1300, 1600, 2000, 3000, 4000, 6000, 8000, 9000];
95 db_ticks = -90:5:-25;
96
97 figure('Name', 'график частотной характеристики', 'Position', [100,
    100, 1400, 700]);
98
99 semilogx(f_orig, P1_orig_db, 'b-', 'LineWidth', 2);
100 hold on;
101 xlim([50, 9000]);
102 ylim([-90, -25]);
103 xticks(freq_ticks);
104 yticks(db_ticks);
105
106 ax = gca;
107 ax.XTickLabelRotation = 45;
108 grid on;
109 grid minor;
110 xlabel('Частота, Гц', 'FontSize', 12, 'FontWeight', 'bold');
111 ylabel('Уровень, дБ', 'FontSize', 12, 'FontWeight', 'bold');
```

```
112 title('Плотный график частотной характеристики', 'FontSize', 14, '
    FontWeight', 'bold');
113 legend(['Оригинал (Fs=' num2str(Fs) 'Гц)'], 'Location', 'southwest', '
    FontSize', 10);
114 hold off;
115
116 figure('Name', 'Сравнение спектров в логарифмическом масштабе', '
    Position', [100, 100, 1200, 800]);
117
118 subplot(2,2,1);
119 semilogx(f_orig, P1_orig_db, 'b', 'LineWidth', 1);
120 title(['Оригинальный сигнал (Fs = ' num2str(Fs) ' Гц) в дБ']);
121 xlabel('Частота, Гц'); ylabel('Амплитуда, дБ');
122 xlim([50, 8000]);
123 ylim([-85, -30]);
124 xticks(freq_ticks(freq_ticks <= 8000));
125 grid on;
126
127 subplot(2,2,2);
128 semilogx(f_down, P1_down_db, 'r', 'LineWidth', 1);
129 title(['Прореженный сигнал (Fs = ' num2str(Fs_new) ' Гц) в дБ']);
130 xlabel('Частота, Гц'); ylabel('Амплитуда, дБ');
131 xlim([50, Fs_new/2]);
132 ylim([-85, -30]);
133 xticks(freq_ticks(freq_ticks <= Fs_new/2));
134 grid on;
135
136 subplot(2,2,[3,4]);
137 semilogx(f_orig, P1_orig_db, 'b', 'LineWidth', 1.5); hold on;
138 semilogx(f_down, P1_down_db, 'r', 'LineWidth', 1);
139 title('Сравнение спектров в логарифмическом масштабе дБ()');
140 xlabel('Частота, Гц'); ylabel('Амплитуда, дБ');
141 xlim([50, 8000]);
142 ylim([-85, -30]);
143 xticks(freq_ticks(freq_ticks <= 8000));
144 legend(['Оригинал (Fs=' num2str(Fs) 'Гц)'], ['Прореженный (Fs=' num2str
    (Fs_new) 'Гц)'], 'Location', 'best');
145 grid on;
```

Листинг 5.2 — Код лабораторной работы на MATLAB

5.3 Скрипт 3 - оценка влияние разрядности АЦП на спектр сигнала

```
[y, Fs] = audioread('C:\Users\KL\DesktopСибГУТИ\\3 курсОСМС\\lab1\voice.wav');

% Функция для квантования сигнала с визуализацией линий квантования
function [yq, l] = quantize_signal(y, bits, overload_factor)

    yo = y * overload_factor;% Усиливаем сигнал чтобы вызвать перегрузку
    y_clipped = max(-1, min(1, yo)); % Применяем ограничение сигнал( не может выйти за [-1, 1])

    % Определяем диапазон для квантования
    q_min = -1;
    q_max = 1;
    q_range = q_max - q_min;

    num_levels = 2^bits; % Количество уровней квантования
    delta = q_range / num_levels;% Шаг квантования
    l = q_min:delta:q_max;% Уровни квантования

    % Квантование сигнала
    y_normal = (y_clipped - q_min) / q_range;
    yq = round(y_normal * (num_levels - 1)) / (num_levels - 1);
    yq = yq * q_range + q_min;
    yq = yq * overload_factor;% Усиливаем обратно для отображения перегрузки
end

% Функция для вычисления коэффициентов Фурье
function [an, bn] = fourier_coefficients(s, t, f, n_n)
    T = t(end) - t(1) + (t(2) - t(1)); % Полный период с учетом шага
    an = zeros(1, length(n_n));
    bn = zeros(1, length(n_n));
    for i = 1:length(n_n)
        n = n_n(i);
        sc = cos(2 * pi * n * f * t);  % опорное косинусное колебание
        ss = sin(2 * pi * n * f * t);

        s = s(:); % Убедимся, что все векторы - столбцы
        sc = sc(:);
        ss = ss(:);
        t = t(:);

        if n == 0
```

```
            an(i) = (1/T) * trapz(t, s .* sc);
            bn(i) = (1/T) * trapz(t, s .* ss);
        else
            an(i) = (2/T) * trapz(t, s .* sc);
            bn(i) = (2/T) * trapz(t, s .* ss);
        end
    end
    an = round(an, 4);
    bn = round(bn, 4);
    threshold = 1e-10;
    an(abs(an) < threshold) = 0.0;
    bn(abs(bn) < threshold) = 0.0;
end

% Функция для вычисления амплитудного и фазового спектра
function [An, fi] = compute_amplitude_phase(an, bn)
    An = sqrt(an.^2 + bn.^2);
    fi = atan2(-bn, an) * (180/pi);
    An = round(An, 4);
    fi = round(fi, 4);
end

% Функция для восстановления сигнала из коэффициентов Фурье
function s_reconstructed = reconstruct_signal(an, bn, t, f, n_n)
    s_reconstructed = zeros(size(t));

    for i = 1:length(n_n)
        n = n_n(i);
        if n == 0
            s_reconstructed = s_reconstructed + an(i);
        else
            s_reconstructed = s_reconstructed + an(i) * cos(2*pi*n*f*t)
    + bn(i) * sin(2*pi*n*f*t);
        end
    end
end

fprintf('Размер массива y: %d x %d\n', size(y, 1), size(y, 2));
y = y / max(abs(y)); % Нормализация к [-1, 1]

% Разрядности для анализа
bits_array = [3, 4, 5, 6]; % Разрядности АЦП
overload_factor = 2.0; % Коэффициент перегрузки

```

```
% Найдем сегмент для анализа
segment_length = round(0.02 * Fs); % 20 мс
[max_val, max_idx] = max(abs(y));
start_idx = max(1, max_idx - round(segment_length/2));
end_idx = min(length(y), start_idx + segment_length - 1);

y_segment = y(start_idx:end_idx);
t_segment = (0:length(y_segment)-1) / Fs;

f0 = 100;% Основная частота и гармоники
n_n = 0:8;

% Анализ исходного сигнала
[an_orig, bn_orig] = fourier_coefficients(y_segment, t_segment, f0, n_n
    );
[An_orig, fi_orig] = compute_amplitude_phase(an_orig, bn_orig);

errors_original = zeros(size(bits_array));     % Ошибки для исходного
    сигнала
errors_overloaded = zeros(size(bits_array));   % Ошибки для
    перегруженного сигнала

for i = 1:length(bits_array)
    bits = bits_array(i);

    % Ошибка для исходного сигнала без( перегрузки)
    [y_quant_orig, ~] = quantize_signal(y_segment, bits, 1.0); %
    overload_factor = 1.0
    quant_er_orig = y_segment - y_quant_orig;
    er_orig(i) = mean(abs(quant_er_orig));

    % Ошибка для перегруженного сигнала
    y_q_over = quantize_signal(y_segment, bits, overload_factor);
    y_over = y_segment * overload_factor;
    y_q_over = y_over - y_q_over;
    er_over(i) = mean(abs(y_q_over));
end

figure('Name', 'Оцифровка исходного сигнала', 'Position', [100, 100,
    1200, 800]);
% Исходный сигнал
subplot(2, 2, 1);
plot(t_segment, y_segment, 'k-', 'LineWidth', 2);
title('Исходный сигнал');
```

```
xlabel('Время, с'); ylabel('Амплитуда');
grid on;

% Оцифрованный сигнал для разных разрядностей с линиями квантования
colors = ['r', 'g', 'b', 'm'];
for i = 1:3 % Только 3,4,5 бит
    bits = bits_array(i);

    % Квантование БЕЗ перегрузки только( для визуализации)
    [y_quantized_no_overload, levels] = quantize_signal(y_segment, bits
   , 1.0); % overload_factor = 1.0

    subplot(2, 2, i+1);

    % Отображаем исходный сигнал
    plot(t_segment, y_segment, 'k-', 'LineWidth', 1); hold on;

    % Отображаем линии квантования
    for j = 1:length(levels)
        plot([t_segment(1), t_segment(end)], [levels(j), levels(j)], '
   --', ...
             'Color', [0.7, 0.7, 0.7], 'LineWidth', 0.8);
    end

    % Отображаем квантованный сигнал
    plot(t_segment, y_quantized_no_overload, colors(i), 'LineWidth', 2)
   ;

    title(['Оцифровка: ' num2str(bits) ' бит (' num2str(2^bits) '
   уровней)']);
    xlabel('Время, с'); ylabel('Амплитуда');
    legend('Исходный', 'Уровни квантования', 'Оцифрованный', 'Location'
   , 'best');
    grid on;
end

figure('Name', 'Оцифровка перегруженного сигнала', 'Position', [100,
   100, 1200, 800]);

% Исходный и перегруженный сигнал
subplot(2, 2, 1);
y_overloaded = y_segment * overload_factor;
plot(t_segment, y_segment, 'b-', 'LineWidth', 2); hold on;
plot(t_segment, y_overloaded, 'r-', 'LineWidth', 1.5);
```

```
title('Исходный и перегруженный сигнал');
xlabel('Время, с'); ylabel('Амплитуда');
legend('Исходный', ['Перегруженный ×' num2str(overload_factor)], '
    Location', 'best');
grid on;

% Оцифрованный перегруженный сигнал для разных разрядностей с линиями
    квантования
for i = 1:3 % Только 3,4,5 бит
    bits = bits_array(i);

    % Квантование С перегрузкой
    [y_quantized, levels] = quantize_signal(y_segment, bits,
    overload_factor);
    subplot(2, 2, i+1);
    % Отображаем перегруженный сигнал
    plot(t_segment, y_overloaded, 'k-', 'LineWidth', 1, 'Color', [0.5
    0.5 0.5]); hold on;

    % Отображаем линии квантования
    levels_scaled = levels * overload_factor;
    for j = 1:length(levels_scaled)
        plot([t_segment(1), t_segment(end)], [levels_scaled(j),
    levels_scaled(j)], '--', ...
             'Color', [0.7, 0.7, 0.7], 'LineWidth', 0.8);
    end

    % Отображаем квантованный сигнал
    plot(t_segment, y_quantized, colors(i), 'LineWidth', 2);

    title(['Оцифровка перегруженного: ' num2str(bits) ' бит (' num2str
    (2^bits) ' уровней)']);
    xlabel('Время, с'); ylabel('Амплитуда');
    legend('Перегруженный', 'Уровни квантования', 'Оцифрованный', '
    Location', 'best');
    grid on;
end

figure('Name', 'Влияние разрядности АЦП на спектр', 'Position', [100,
    100, 1200, 800]);
for i = 1:length(bits_array)
    bits = bits_array(i);

    % Квантование исходного сигнала без( перегрузки)
```

```
    y_quantized_original = quantize_signal(y_segment, bits, 1.0); % overload_factor = 1.0

    % Расчет коэффициентов Фурье для квантованного исходного сигнала
    [an_quant_orig, bn_quant_orig] = fourier_coefficients(y_quantized_original, t_segment, f0, n_n);
    An_quant_orig = sqrt(an_quant_orig.^2 + bn_quant_orig.^2);

    subplot(2, 2, i);
    stem(n_n, An_orig, 'b-', 'LineWidth', 1.5, 'MarkerSize', 6); hold on;
    stem(n_n, An_quant_orig, 'r-', 'LineWidth', 1.5, 'MarkerSize', 6);
    title(['Разрядность: ' num2str(bits) ' бит исходный( сигнал)']);
    xlabel('n'); ylabel('An');
    legend('Исходный', 'Квантованный исходный', 'Location', 'best');
    grid on;
end
```

Листинг 5.3 — Код лабораторной работы на MATLAB

5.4 Скрипт 4 - генерация песенки по нотам

```
Fs = 44100;
amplitude = 1;
tempo = 120;
beat_duration = 60/tempo;

% Частоты нот
notes_freq.G3 = 196.00;     % Соль
notes_freq.A3 = 220.00;     % Ля
notes_freq.B3 = 246.94;
notes_freq.C4 = 261.63;     % До
notes_freq.D4 = 293.66;     % Ре
notes_freq.E4 = 329.63;     % Ми
notes_freq.F4 = 349.23;     % Фа
notes_freq.G4 = 392.00;     % Соль
notes_freq.A4 = 440.00;     % Ля
notes_freq.B4 = 493.88;
notes_freq.C5 = 523.25;
notes_freq.D5 = 587.33;
melody_sequence = {
    'G3', 1;    % Соль
    'E4', 1;    % ми
    'E4', 1;    % ми
```

```
    'D4', 1;    % ре
    'E4', 1;    % ми
    'C4', 1;    % до
    'G3', 1;    % соль
    'G3', 1;    % соль

    'G3', 1;    % соль
    'E4', 1;    % ми
    'E4', 1;    % ми
    'F4', 1;    % фа
    'D4', 1;    % ре
    'G4', 1;    % соль

    'G4', 1;    % Соль
    'A3', 1;    % ля
    'A3', 1;    % ля
    'F4', 1;    % фа
    'F4', 1;    % фа
    'E4', 1;    % ми
    'D4', 1;    % ре
    'C4', 1;    % до
    'A3', 1;    % ля
    'E4', 1;    % ми
    'E4', 1;    % ми
    'D4', 1;    % ре
    'E4', 1;    % ми
    'C4', 1;    % до
};

% Генерация мелодии
full_melody = [];
for i = 1:length(melody_sequence)
    note_name = melody_sequence{i, 1};
    duration_beats = melody_sequence{i, 2};
    note_duration = duration_beats * beat_duration;
    freq = notes_freq.(note_name);

    t = 0:1/Fs:note_duration;
    tone = amplitude * sin(2*pi*freq*t);

    full_melody = [full_melody, tone];
end

% Нормализация
```

```
full_melody = full_melody / max(abs(full_melody));

player = audioplayer(full_melody, Fs);
play(player);
fprintf('Воспроизведение началось. Длительность: %.1f сек\n', length(
    full_melody)/Fs);
fprintf('Чтобы остановить: stop(player)\n');
```

Листинг 5.4 — Код лабораторной работы на MATLAB