МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ
Кафедра компьютерных технологий и систем

КИНЧИКОВ Максим Иванович

ОБРАБОТКА РЕЧЕВЫХ СИГНАЛОВ МЕТОДАМИ ИНТЕГРАЛЬНЫХ ПРЕОБРАЗОВАНИЙ

Дипломная работа

Научный руководитель:
доцент кафедры КТС,
кандидат физико-
математических наук
Е.С. Чеб

Допущена к защите
«____»_____20___г.
Заведующий кафедрой компьютерных технологий и систем
доктор педагогических наук, кандидат физико-
математических наук, профессор В.В. Казачёнок

Минск, 2025

ОГЛАВЛЕНИЕ

РЕФЕРАТ

Дипломная работа: 76 с., 31 рис., 3 прил., 24 источника.

Ключевые слова: ИНТЕГРАЛЬНЫЕ ПРЕОБРАЗОВАНИЯ, ПРЕОБРАЗОВАНИЕ ФУРЬЕ, ВЕЙВЛЕТ-ПРЕОБРАЗОВАНИЕ, ОБРАБОТКА РЕЧЕВЫХ СИГНАЛОВ, ШУМОПОДАВЛЕНИЕ, ПЕРЕДИСКРЕТИЗАЦИЯ.

Объект исследования: цифровые речевые сигналы, методы их обработки.

Цель исследования: разработка вычислительных алгоритмов подавления шума и изменения частоты дискретизации речевых сигналов с применением интегральных преобразований и реализация программы на их основе.

Методы исследования: теоретические: интегральные преобразования, цифровые сигналы, теория цифровых фильтров, теория алгоритмов; практические: разработка алгоритмов, программная реализация и визуализация результатов.

Полученные результаты: реализована программа, состоящая из двух модулей шумоподавления и одного модуля передискретизации, осуществляющих соответствующую обработку речевых сигналов, проведен анализ качества получаемых в результате работы программы выходных сигналов, который подтверждает успешность работы программы на основе разработанных алгоритмов.

Область возможного практического применения: цифровая обработка речевых сигналов.

РЭФЕРАТ

Дыпломная праца: 76 с., 31 мал., 3 дадатка, 24 крыніцы.

Ключавыя словы: ІНТЭГРАЛЬНЫЯ ПЕРАЎТВАРЭННІ, ПЕРАЎТВАРЭННЕ ФУР'Е, ВЕЙВЛЕТ-ПЕРАЎТВАРЭННЕ, АПРАЦОЎКА МАЎЛЕНЧЫХ СІГНАЛАЎ, ШУМАПРЫГЛУШЭННЕ, ПЕРАДЫСКРЭТЫЗАЦЫЯ.

Аб'ект даследавання: лічбавыя маўленчыя сігналы, метады іх апрацоўкі.

Мэта даследавання: распрацоўка вылічальных алгарытмаў падаўлення шуму і змены частоты дыскрэтызацыі маўленчых сігналаў з ужываннем інтэгральных пераўтварэнняў і рэалізацыя праграмы на іх аснове.

Метады даследавання: тэарэтычныя: інтэгральныя пераўтварэнні, лічбавыя сігналы, тэорыя лічбавых фільтраў, тэорыя алгарытмаў; практычныя: распрацоўка алгарытмау, праграмная рэалізацыя і візуалізацыя вынікаў.

Атрыманыя вынікі і іх навізна: рэалізавана праграма, якая складаецца з дзвюх модуляў шумапрыглушэння і аднаго модуля перадыскрэтызацыі, якія ажыццяўляюць адпаведную апрацоўку маўленчых сігналаў, праведзен аналіз якасці атрымленных у выніку працы праграмы выхадных сігналаў, які пацвярджае паспяховасць працы праграмы на аснове распрацаваных алгарытмаў.

Вобласць магчымага практычнага прымянення: лічбавая апрацоўка маўленчых сігналаў.

SUMMARY

Diploma work: 76 p., 31 fig., 3 append., 24 references.

Keywords: INTEGRAL TRANSFORMS, FOURIER TRANSFORM, WAVELET TRANSFORM, SPEECH SIGNAL PROCESSING, NOISE REDUCTION, RESAMPLING.

The object of the research: digital speech signals, methods of their processing.

The aim of the research: development of computational algorithms for noise suppression and changing the sampling frequency of speech signals using integral transformations and implementation of a program based on them.

Research methods: theoretical: integral transformations, digital signals, theory of digital filters, theory of algorithms; practical: development of algorithms, software implementation and visualization of results.

The results of the work and their novelty: a program consisting of two noise suppression modules and one resampling module that perform the corresponding processing of speech signals has been implemented; an analysis of the quality of the output signals obtained as a result of the program's operation has been carried out, which confirms the successful operation of the program based on the developed algorithms.

Recommendations on the usage: digital speech signal processing.

ВВЕДЕНИЕ

Речь является неотъемлемой частью нашего ежедневного общения. Мы используем голос, чтобы выражать свои мысли, передавать информацию и устанавливать контакт с окружающими. В современном мире, где информационные технологии исключительно важны, и голосовые интерфейсы все больше распространяются, понимание и анализ речевых сигналов становится актуальным исследовательским направлением.

Речевые сигналы сегодня используются во множестве сфер. Например, как уже упоминалось, в голосовых интерфейсах: технологиях голосового управления и голосовых помощников. Анализ речевых сигналов позволяет совершенствовать их работу, улучшать распознавание речи и создавать более эффективные и удобные пользовательские интерфейсы. Распознавание и анализ речевых сигналов могут быть использованы в системах биометрической аутентификации, например, в голосовое распознавание, для обеспечения безопасности и защиты данных. Анализ речевых сигналов может быть полезным в области медицины и реабилитации: он может помочь в диагностике и мониторинге различных заболеваний, связанных с речью. Также, анализ речи может быть использован для разработки систем коммуникации для людей с нарушениями речи или физическими ограничениями. Понимание речи и выделение информации из аудиоданных помогает создавать более эффективные системы машинного перевода и анализа текста.

Одной из основных задач обработки речевых сигналов является задача шумоподавления. Ей уделяется большое внимание, что связано с широким кругом возможных применений данного вида обработки сигнала. Речевые сигналы зачастую имеют значительные искажения, обусловленные фоновым шумом или различными посторонними звуками. Всё это значительно усложняет восприятие речевого сигнала человеком, а также вызывает сложности в работе устройств распознавания речи, ее модификации и т. д. Для решения данных проблем активно используется предварительное подавление шума в речевых сигналах.

К распространенным задачам, связанным с обработкой, также относят задачу изменения частоты дискретизации сигнала. Различные устройства, работающие со звуком, имеют различные требования к частоте дискретизации аудиосигнала. Так, например, устройство может работать только с частотой дискретизации в 44,1 кГц, а исходный сигнал мог быть записан с частотой 48 кГц. Изменение частоты дискретизации позволяет привести аудиосигнал к требуемому формату и обеспечить совместимость между устройствами.

Помимо этого, может возникать необходимость в объединении нескольких аудиосигналов в один и для этого нужно, чтобы они имели одну и ту же частоту дискретизации. Иногда уменьшение частоты дискретизации может применяться для уменьшения размера хранимого файла, так как при выполнении этой операции размер файла уменьшается пропорционально изменению частоты дискретизации. Конечно же, все вышеперечисленное, в частности, актуально и для речевых сигналов.

Большинство методов обработки сигналов речевых сигналов тесно связаны с интегральными преобразованиями, в том числе и вышеперечисленные. Самыми популярными преобразованиями в этой области являются преобразование Фурье и вейвлет-преобразование.

Цель данной дипломной работы заключается в рассмотрении нескольких методов обработки речевых сигналов, а именно, шумоподавления (Фурье и вейвлет) и передискретизации. Соответственно, предполагается разработка алгоритмов подавления шума в речевых сигналах, позволяющего повысить их качество путем уменьшения фонового шума без снижения разборчивости речевого сообщения и разработка алгоритма изменения частоты дискретизации речевого сигнала. Также целью является применение этих алгоритмов на практике, путем реализации программ шумоподавления и передискретизации.

Для достижения поставленных целей необходимо решить следующие задачи:

- изучить сущность речевого сигнала и его представление в цифровом виде;
- изучить преобразование Фурье и его свойства;
- изучить вейвлет-преобразование и его свойства;
- разработать вычислительный алгоритм для подавления шума в речевых сигналах, основывающийся на преобразовании Фурье;
- разработать иной метод шумоподавления, базирующийся на вейвлет-преобразовании;
- разработать вычислительный алгоритм для изменения частоты дискретизации речевых сигналов;
- реализовать программу, основанную на вышеупомянутых алгоритмах;
- провести анализ данных, полученных в результате работы программы;
- сделать выводы на основе проведенных исследований.

ГЛАВА 1
МАТЕМАТИЧЕСКИЕ ОСНОВЫ ОБРАБОТКИ РЕЧЕВЫХ СИГНАЛОВ

1.1 Понятие речевого сигнала

Сперва введем пару понятий, необходимых для определения речевого сигнала.

Информация – свойство материи, сведения об объектах и явлениях окружающей среды, их параметрах, свойствах и состоянии, которые уменьшают имеющуюся о них степень неопределенности, неполноты знаний. Являясь свойством материи, информация может рассматриваться как величина [15].

Сигнал – носитель информации. Информация передаётся сигналом за счёт изменения характеристик или параметров сигнала во времени. Сигнал может быть определён как функция, переносящая информацию о состоянии или поведении физической системы. Тем самым сигнал также можно понимать как математическую функцию от одной или нескольких независимых переменных [15].

Речевую информацию будем рассматривать как информацию, источником которой является голосовой аппарат человека, и которая передается в процессе речевой коммуникации, в том числе с ее последующей фиксацией (записью) на материальном носителе, и его дальнейшем воспроизведением [2].

Тогда речевым сигналом можно назвать сигнал, который переносит речевую информацию.

Звук – это колебательный процесс, возникающий в воздухе (или другой упругой среде) под действием каких-либо колеблющихся предметов, заставляющих колебаться окружающие их частицы воздуха. Плотность воздушной среды при этом то увеличивается, то уменьшается в соответствии с колебаниями источника звука. В простейшем случае это «чистый» тон (звук камертона), при котором источник излучает только одну частоту, и изменение мгновенных значений колебания строго подчиняются закону синуса. В общем случае звук состоит из множества таких синусоидальных колебаний и, соответственно, является результатом наложения друг на друга таких синусоидальных волн [15].

Речь является одной из форм звука, который образуется в результате колебания голосовых связок человека. Следовательно речь обладает теми же свойствами, что и звук. Звуковые колебания, распространяющиеся в упругой среде, описываются волновым уравнением. К основным характеристикам звуковой волны относят временную динамику, амплитудные параметры (звуковое давление, интенсивность) и частотой спектр. Амплитуды разных волн могут складываться, например, $A + (-A) = 0$ [15].

1.2 Оцифровка акустических сигналов

Чтобы получить электрический сигнал из исходного акустического, используются микрофоны. Электрический (аналоговый) сигнал в свою очередь проходит оцифровку – преобразуется в цифровой сигнал при помощи аналого-цифрового преобразователя.

Цифровой сигнал – это сигнал, который можно представить в виде дискретных числовых значений (в случае компьютеров – двоичный цифровой сигнал). Преобразование аналогового сигнала в цифровой осуществляется путем дискретизации сигнала по времени и его квантованием по амплитуде. Квантование ограничивает динамический диапазон цифрового сигнала, дискретизация по времени ограничивает частотный диапазон сигнала [15].

Любая обработка звуковых данных, включая речь, начинается с аналого-цифрового преобразования. В результате получается цифровой сигнал, характеризующийся дискретными значениями как во временной области, так и по уровню амплитуды. Именно с такой цифровой формой сигнала осуществляется дальнейшая работа. Для того, чтобы лучше понимать, как устроен цифровой сигнал, с которым мы будем работать, мы разберем дискретизацию и квантование сигналов.

Дискретизация — взятие измерений (отсчетов) в дискретные моменты времени. Дискретизация сохраняет временную информацию [13].

Как уже было сказано, цифровая обработка данных возможна исключительно при числовом представлении информации. При работе с сигналами это обычно означает использование последовательности значений, полученных путём равномерной дискретизации — процесса измерения амплитуды сигнала через фиксированные временные промежутки.

Пусть $x(t)$ — некоторый непрерывный сигнал. Здесь t — непрерывный аргумент, под которым подразумевается время. Тогда $x(n) = x(nT)$ представляет собой результат равномерной дискретизации непрерывного сигнала $x(t)$. Здесь T — временной шаг дискретизации, а n — порядковый

номер отсчета. Частота дискретизации – величина обратная шагу дискретизации $F_s = \frac{1}{T}$. Она измеряется в герцах (Гц) – количествах отсчетов в секунду. Наиболее распространенные частоты дискретизации – кратные 8 кГц (16, 24, 48, 96 кГц) и кратные 11025 Гц (22,050 и 44,1 кГц). Наиболее часто в речевых приложениях используется частота 16 кГц [15].

Понятно, что, беря отсчеты исходного сигнала, мы теряем информацию о значениях, которые лежат между ними. Чтобы знать, при какой частоте дискретизации возможно обратное восстановление сигнала обратимся к теореме Котельникова.

Формулировка теоремы заключается в следующем: если функция, зависящая от времени, имеет ограниченный частотой F_n спектр, то она полностью определяется дискретными отсчетами ее значений, следующими с частотой $F_s = 2F_n$, и может быть точно реконструирована с помощью выражения [14]:

$$x(t) = \sum_{n=-\infty}^{\infty} x(n) \frac{\sin \frac{\pi(t - nT)}{T}}{\frac{\pi(t - nT)}{T}}. \quad (1.1)$$

Следует учитывать, что возможность точного восстановления исходного сигнала существует лишь в теории, поскольку для этого требуется бесконечное количество отсчётов $x(n)$, что в реальных условиях невозможно.

Частота:

$$F_n = \frac{F_s}{2}, \quad (1.2)$$

упоминаемая в теореме называется граничной частотой Найквиста. Указанная частота применяется в фильтрах низких частот (ФНЧ) для ограничения спектра сигнала перед дискретизацией. При превышении частотой сигнала частоты Найквиста происходит необратимая потеря информации - вместо исходного сигнала $F_0 > F_n$ восстанавливается ложная частота $F_i = F_s - F_0 < F_n$. Это явление, известное как наложение спектров, возникает при недостаточной частоте дискретизации, делая некоторые спектральные компоненты неидентифицируемыми. Именно для предотвращения наложения спектров и используется ФНЧ.

Квантование сигнала — дискретизация амплитуды и представление ее в двоичном коде (битах), то есть происходит присвоение цифровому отсчету значения, соответствующего некоторому фиксированному уровню сигнала. При этом считается, что квантованный сигнал имеет целочисленные

амплитуды, соответствующие двоичным числам. Обыкновенно используют 8-, 16-, 24- и 32-разрядное квантование (1, 2, 3, 4 байта) [15].

Чем больше разрядность квантования, тем шире динамический диапазон, меньше нелинейные искажения и шум, выше разрешающая способность по уровням. В отличие от процесса дискретизации по времени, квантование по уровню вносит в кодируемый сигнал погрешность. Разность между фактическим значением аналогового сигнала и представляющим его двоичным числом, называется погрешностью квантования.

Децибелы – это число, выражающее в логарифмической мере отношение двух величин (измеряемой и эталонной), которое употребляется при большом отношении этих величин. Децибелы для отношения мощностей: $d_P = 10\ \ln\frac{P}{P_0}$, а децибелы для отношения амплитуд: $d_A = 20\ \ln\frac{A}{A_0}$. Разной битовой глубине квантования соответствуют различные динамические диапазоны, измеряемые в децибелах. За эталон обычно принимают минимальное ненулевое кодируемое значение, например, единицу. В цифровой обработке динамический диапазон определяется как логарифмическое отношение максимальной амплитуды сигнала к минимальной.

1.3 Представление речевого сигнала ортогональными функциями

Поскольку, как уже раньше упоминалось, звук является результатом наложения друг на друга синусоидальных волн, речевой сигнал можно представить в виде набора функций синуса и косинуса. При этом мы сразу будем рассматривать оцифрованный сигнал, то есть он будет состоять из дискретных отсчетов. Чтобы это сделать, прибегнем к ряду Фурье, у которого в качестве ортогональной системы функций возьмем систему

$$\left\{1, \cos\frac{\pi t}{T}, \sin\frac{\pi t}{T}, \dots, \cos\frac{k\pi t}{T}, \sin\frac{k\pi t}{T}, \dots\right\} \tag{1.3}$$

тригонометрических функций, где T – период периодической функции $x(t)$. Тем самым речевой сигнал мы можем представить следующим тригонометрическим рядом Фурье:

$$x(t) = \frac{a_0}{2} + \sum_{k=1}^{\infty}\left[a_k \cos(k\omega_1 t) + b_k \sin(k\omega_1 t)\right], \tag{1.4}$$

где

$$a_k = \frac{2}{T}\int_{-\frac{T}{2}}^{\frac{T}{2}} \mathrm{x(t)}\cos(k\omega_1 t)\,dt\,, \tag{1.5}$$

$$b_k = \frac{2}{T}\int_{-\frac{T}{2}}^{\frac{T}{2}} \mathrm{x(t)}\sin(k\omega_1 t)\,dt\,. \tag{1.6}$$

В этих формулах $\omega_1 = \frac{2\pi}{T}$ – «основная» частота ряда частот $k\omega_1$ гармоник, на которые раскладывается сигнал $x(t)$.

Однако в случае цифрового анализа сигналов чаще используют комплексную форм записи ряда Фурье:

$$x(t) = \sum_{k=-\infty}^{\infty} C_k e^{ik\omega_1 t}\,, \tag{1.7}$$

где

$$C_k = \frac{1}{T}\int_{-\frac{T}{2}}^{\frac{T}{2}} x(t)e^{-ik\omega_1 t}dt\,. \tag{1.8}$$

Отметим, что если сигнал непрерывен, что в случае оцифровки невозможно, то мы бы вместо ряда Фурье рассматривали интеграл Фурье.

1.4 Непрерывное преобразование Фурье и его свойства

Сперва рассмотрим непрерывное преобразование Фурье (более общий случай), а потом перейдем к его дискретной версии, который более актуален для цифровых сигналов.

Прямым преобразованием Фурье называется формула

$$\hat{x}(\omega) = \int_{-\infty}^{\infty} x(t)e^{-i\omega t}\,dt. \tag{1.9}$$

В данной формуле $x(t)$ является обрабатываемым сигналом (функция от времени), а $\hat{x}(\omega)$ – так называемым спектром этого сигнала (функция от частоты сигнала) или образом Фурье. Тем самым, при помощи преобразования Фурье мы переходим из временной области в частотную [3; 7; 16].

Модуль спектральной функции называют амплитудным спектром, а ее аргумент – фазовым спектром.

Преобразование

$$x(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} \hat{x}(\omega)e^{i\omega t}\, d\omega \tag{1.10}$$

называют обратным преобразованием Фурье. Очевидно, это преобразование, наоборот, позволяет перейти из частотной области во временную. Тем самым, преобразование Фурье является обратимой операцией. Формулы (1.9) и (1.10) называют парой непрерывных преобразований Фурье.

Чтобы преобразование Фурье было применимо, должны выполняться условия Дирихле, а сигнал быть абсолютно интегрируемым. Это означает, что интеграл его модуля должен быть конечной величиной:

$$\int_{-\infty}^{\infty} |x(t)|\, dt < \infty. \tag{1.11}$$

Теперь рассмотрим свойства непрерывного преобразования Фурье. Пусть $x(t)$ и $\hat{x}(\omega)$ соответственно сигнал и спектр этого сигнала, а преобразование Фурье, применяемое к ним, будем обозначать оператором $\leftrightarrow$. Рассмотрим следующие свойства преобразования Фурье [16]:

1. Линейность: если $x_1(t) \leftrightarrow \widehat{x_1}(\omega)$ и $x_2(t) \leftrightarrow \widehat{x_2}(\omega)$, то

$$\alpha x_1(t) + \beta\, x_2(t) \leftrightarrow \alpha\widehat{x_1}(\omega) + \beta\, \widehat{x_2}(\omega). \tag{1.12}$$

То есть, если мы берем какую-то линейную комбинацию функций, то преобразование Фурье этой комбинации будет такой же линейной комбинацией образов Фурье этих функций. Это свойство позволяет сводить сложные функции и их образы Фурье к более простым.

2. Теорема запаздывания: если $x(t) \leftrightarrow \hat{x}(\omega)$, то

$$x(t - t_0) \leftrightarrow \hat{x}(\omega)e^{-i\omega t_0}. \tag{1.13}$$

Это значит, что амплитудный спектр не зависит от сдвига сигнала по времени. Если мы подвинем сигнал влево или вправо вдоль оси времени, то поменяется лишь его фазовый спектр.

3. Теорема смещения: если $x(t) \leftrightarrow \hat{x}(\omega)$, то

$$x(t)e^{i\omega_0 t} \leftrightarrow \hat{x}(\omega - \omega_0). \tag{1.14}$$

Здесь суть та же: сигнал не зависит от сдвига амплитудного спектра по частоте.

4. Формулы свертки: если $x_1(t) \leftrightarrow \widehat{x_1}(\omega)$ и $x_2(t) \leftrightarrow \widehat{x_2}(\omega)$, то

$$x_1(t) \cdot \beta\, x_2(t) \leftrightarrow \widehat{x_1}(\omega) \otimes \widehat{x_2}(\omega), \tag{1.15}$$

где

$$\widehat{x_1}(\omega) \otimes \widehat{x_2}(\omega) = \int_{-\infty}^{\infty} \widehat{x_1}(\omega - \tau)\, \widehat{x_2}(\tau) d\tau\,. \tag{1.16}$$

И наоборот

$$x_1(t) \otimes \beta\, x_2(t) \leftrightarrow \widehat{x_1}(\omega) \cdot \widehat{x_2}(\omega). \tag{1.17}$$

Эти свойства позволяют свести свертку функций к поточечному перемножению их образов Фурье и наоборот — поточечное перемножение функций к свертке их образов Фурье.

5. Масштабирование: если $x(t) \leftrightarrow \hat{x}(\omega)$, то

$$x(at) \leftrightarrow \frac{1}{|a|}\hat{x}\left(\frac{\omega}{a}\right). \tag{1.18}$$

Растяжение (сжатие) исходной функции по оси времени пропорционально сжимает (растягивает) её образ по оси частот.

6. Комплексное сопряжение: если $x(t) \leftrightarrow \hat{x}(\omega)$, то

$$\overline{x(t)} \leftrightarrow \overline{\hat{x}(-\omega)}. \tag{1.19}$$

Это свойство говорит о симметрии образов Фурье. В частности, из этого свойства следует что у образа действительной функции амплитудный спектр всегда является четной функцией, а фазовый спектр - нечетной. По этой причине на графиках спектров практически никогда не рисуют отрицательную

часть спектра — для действительных сигналов она не дает никакой новой информации.

Помимо этих еще есть ряд свойств, связанных с δ-функцией Дирака, но они здесь рассматриваться не будут.

1.5 Дискретное преобразование Фурье и его свойства

Так как сигнал, с которым мы работаем является цифровым (дискретным), нам не подойдет обычное преобразование Фурье для непрерывных функций. Рассмотрим же тогда дискретную версию этого преобразования. Отметим, что в данном преобразовании неявно подразумевается периодичность сигнала, то есть периодическое повторение сигнала за пределами интервала наблюдения. За N будем считать количество отсчетов сигнала ($\frac{1}{N}$ в данном случае - частота дискретизации). В непрерывном преобразовании от интеграла переходим к сумме и обе суммы рассматриваем по отсчетам нашего сигнала. Тогда получаем следующее [3; 7; 16]:

$$\hat{x}(k) = \sum_{n=0}^{N-1} x(n) e^{-ik\frac{2\pi}{N}n}, \tag{1.20}$$

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} \hat{x}(k) e^{ik\frac{2\pi}{N}n}. \tag{1.21}$$

Формула (1.20) – это формула прямого дискретного преобразования Фурье (ДПФ), а (1.21) – обратного дискретного преобразования Фурье (ОДПФ). Таким образом, вектор N отсчетов сигнала преобразуется в вектор N независимых отсчетов спектра при помощи ДПФ и наоборот в случае ОДПФ.

Помимо стандартных свойств непрерывного преобразования Фурье дискретное имеет еще два новых:

1. Периодичность:

$$x(n + jN) = x(n), \tag{1.22}$$

$$\hat{x}(k + jN) = \hat{x}(k). \tag{1.23}$$

2. Изменение знака:

$$x(-n) = \hat{x}(-k). \tag{1.24}$$

1.6 Быстрое преобразование Фурье

Быстрое преобразование Фурье (БПФ) – это быстрый алгоритм для вычисления дискретного преобразования Фурье (ДПФ). Если ДПФ выполняет порядка $O(n^2)$ операций, то БПФ - $O(n \log n)$, что при большом размере входных данных (дискретного сигнала) значительно ускоряет вычисления. По этой причине на пратике чаще всего используется именно БПФ.

Наиболее распространненым алгоритмом быстрого преобразования Фурье является алгоритм Кули — Тьюки, при котором ДПФ представляется как сумма ДПФ более малых размерностей, которые вычисляются рекурсивно. Тем самым применяется подход «разделяй и властвуй», который и дает сложность в $O(n \log n)$.

ДПФ можно представить как сумму четных и нечетных индексов:

$$\hat{x}(k) = \sum_{n=0}^{\frac{N}{2}-1} x_{2n}(n) e^{-ik\frac{2\pi}{N}2n} + \sum_{n=0}^{\frac{N}{2}-1} x_{2n+1}(n) e^{-ik\frac{2\pi}{N}(2n+1)}. \tag{1.25}$$

На основе формулы (1.25) происходит рекурсивный подсчет ДПФ четных и нечетных индексов сигнала.

Обратное БПФ вводится аналогично:

$$x(k) = \frac{1}{N}\left(\sum_{n=0}^{\frac{N}{2}-1} \hat{x}_{2n}(n) e^{ik\frac{2\pi}{N}2n} + \sum_{n=0}^{\frac{N}{2}-1} \hat{x}_{2n+1}(n) e^{ik\frac{2\pi}{N}(2n+1)} \right). \tag{1.26}$$

Стоит отметить, что такой алгоритм применим только для входных данных, чья размерность является степенью двойки. Для этого входные данные могут дополнять нулевыми значениями до ближайшей подходящей размерности.

1.7 Вейвлет-функция и ее свойства

Рассмотрим теперь вейвлет-преобразование. По своей сути оно схоже с преобразованием Фурье, однако использует принципиально иные базисные функции - компактно локализованные как во временной, так и в частотной областях. Эти базисные функции (вейвлеты) формируются на основе специальных материнских вейвлетов $\psi(t)$, определяющих их структуру и характеристики. Локализация достигается за счет масштабирования и временных сдвигов функции $\psi(t)$. Материнский вейвлет порождает целое семейство вейвлет-функций, определяемое двумя независимыми параметрами [4]:

$$\psi_{ab}(t) = \frac{1}{\sqrt{a}}\psi\left(\frac{t-b}{a}\right). \tag{1.27}$$

Вейвлет получаются из материнского путем изменения аргумента времени материнского вейвлета: a масштабирует время, b – сдвигает. Множитель $\frac{1}{\sqrt{a}}$ нормализует значение (дает независимость от масштабирующего числа a).

Вейвлет функция должна обладать следующими свойствами [4]:

1. Автомодельность: вейвлеты семейства $\psi_{ab}(t)$ имеют то же число колебаний, что и материнский вейвлет $\psi(t)$, из-за того, что первые были получены в результате преобразования второго.
2. Ограниченность: квадрат нормы конечен:

$$\|\psi\|^2 = \int_{-\infty}^{+\infty} |\psi(t)|^2 dt < \infty. \tag{1.28}$$

3. Локализация: функция локализована во времени и по частоте, то есть выполняются условия:

$$|\psi(t)| \le \frac{C}{(1+|t|)^{\alpha}}, \qquad |\Psi(\omega)| \le \frac{C}{(1+|\omega|)^{\alpha}}. \tag{1.29}$$

4. Нулевое среднее: среднее значение функции равняется нулю:

$$\int_{-\infty}^{+\infty} \psi(t)dt = 0. \tag{1.30}$$

Материнские вейвлет-функции разбивают на категории в зависимости от базиса пространства, который они используют. К вещественным непрерывным вейвлетам относят WAVE- или MHAT-вейвлеты («мексиканская шляпа»), DOG и другие. К комплексным – вейвлеты Морле и Пауля. Наиболее популярные вещественные дискретные вейвлеты – вейвлеты Хаара, Добеши и Койфлета (частный случай Добеши).

1.8 Непрерывное вейвлет-преобразование и его свойства

Операция прямого вейвлет-преобразования над функцией $x(t)$ описывается следующим выражением:

$$W_x(a,b) = \int_{-\infty}^{+\infty} x(t)\, \psi_{ab}(t) dt\,. \tag{1.31}$$

Учитывая формулу (1.27), получаем:

$$W_x(a,b) = \frac{1}{\sqrt{a}} \int_{-\infty}^{+\infty} x(t)\, \psi\left(\frac{t-b}{a}\right) dt\,. \tag{1.32}$$

Функция $W_x(a,b)$ называется вейвлет-спектром. В отличие от преобразования Фурье, дающего одномерную спектральную характеристику, вейвлет-анализ порождает двумерный спектр. Это трехмерное представление, известное как вейвлет-спектрограмма. Чтобы отобразить вейвлет-спектр используют либо поверхность $W_x(a,b)$, либо ее проекцию на плоскость (a,b) в виде линий уровня, либо же картину уровней локальных экстремумов этой поверхности [4; 12].

Обратное вейвлет-преобразование для вещественных вейвлетов определяется как [4]

$$x(t) = \frac{1}{C_\psi} \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} W_x(a,b)\, \psi_{ab}(t)\, da\, db, \tag{1.33}$$

где коэффициент нормализацц:

$$C_\psi = \int_{-\infty}^{+\infty} \frac{|\Psi(\omega)|^2}{|\omega|} d\omega < \infty\,. \tag{1.34}$$

Здесь $\Psi(\omega)$ – преобразование Фурье функции $\psi(t)$.

Выражение (1.33) позволяет сделать две вещи: во-первых, обеспечивает возможность восстановления исходной функции $x(t)$ по её вейвлет-преобразованию, а во-вторых, позволяет выразить $x(t)$ в форме суперпозиции базисных вейвлет-функций $\psi_{ab}(t)$ с коэффициентами $W_x(a,b)$. Здесь параметр a характеризует масштабное преобразование (степень сжатия), а параметр b определяет временной сдвиг [8].

Рассмотрим ряд свойств, которые не зависят от выбора вейвлета [12]:

1. Линейность:

$$W(\alpha x_1(t) + \beta x_2(t)) = \alpha W_{x_1}(a,b) + \beta W_{x_2}(a,b). \quad (1.35)$$

2. Сдвиг: смещение сигнала во временной области на b_0 ведет к сдвигу вейвлет-образа также на b_0:

$$W(x(t - b_0)) = W_x(a, b - b_0). \quad (1.36)$$

3. Масштабирование: растяжение (сжатие) сигнала приводит также к растяжению (сжатию) его вейвлет-образа:

$$W\left[x\left(\frac{t}{a_0}\right)\right] = \frac{1}{a_0} W_x\left[\frac{a}{a_0}, \frac{b}{a_0}\right]. \quad (1.37)$$

4. Дифференцирование:

$$W\left(d_t^n x(t)\right) = (-1)^n \int_{-\infty}^{+\infty} x(t)\, d_t^n\, [\psi_{ab}(t)] dt. \quad (1.38)$$

5. Масштабно-временная локализация. Она обусловлена тем, что элементы базиса вейвлет-преобразования хорошо локализованы и обладают подвижным частотно-временным окном.

Благодаря масштабированию вейвлеты эффективно выявляют частотные особенности сигнала, а временные сдвиги позволяют исследовать локальные свойства на различных участках. Это обеспечивает принципиальное преимущество при анализе нестационарных сигналов вейвлет-преобразованием по сравнению с преобразованием Фурье, которое дает лишь усредненную частотную информацию, поскольку использует базисные функции (комплексные экспоненты), определенные на всей временной оси [12].

1.9 Быстрое вейвлет-преобразование

Непрерывное изменение параметров масштаба a и сдвига b приводит к значительной вычислительной нагрузке при расчете вейвлет-спектра, поскольку порождает избыточное множество базисных функций $\psi_{ab}(t)$. Для практической реализации требуется дискретная сетка параметров, сохраняющая возможность точного восстановления исходного сигнала по его вейвлет-коэффициентам. Дискретизация, как правило, осуществляется через степени двойки [12]:

$$a = 2^j, \qquad b = k2^j, \tag{1.39}$$

где j – параметр масштаба, k – параметр сдвига. Оба числа целые.

Теперь запишем вейвлет в следующем виде:

$$\psi_{jk}(t) = \frac{1}{\sqrt{2^j}}\psi\left(\frac{t}{2^j} - k\right). \tag{1.40}$$

В данном случае применяется не непрерывное, а выборочное преобразование, где параметры сдвига и масштаба берутся на дискретной (обычно логарифмической) сетке. Терминологически корректное название - диадное вейвлет-преобразование, поскольку сигнал все еще остается непрерывным. Его аналитическое представление дано в уравнениях (1.41) (прямое) и (1.42) (обратное) [12]:

$$d_{jk} = \int_{-\infty}^{+\infty} x(t)\,\psi_{jk}^*(t)dt\,, \tag{1.41}$$

$$x(t) = \sum_{j=-\infty}^{+\infty}\sum_{k=-\infty}^{+\infty} d_{jk}\,\psi_{jk}(t)\,. \tag{1.42}$$

Если же сигнал – дискретный, то аналогичное преобразование правильно называть дискретным вейвлет-преобразованием.

Вейвлет-коэффициенты ДВП можно вычислить с помощью итерационной процедуры, известной под названием быстрого вейвлет-преобразования (БВП). Его принцип заключается в том, чтобы представить сигнал в виде следующего разложения [6; 12]:

$$x(t) = A_{j_0}(t) + \sum_{j=1}^{j_0} D_j\,(t), \tag{1.43}$$

где последовательность грубой (аппроксимирующей) $A_{j_0}(t)$ и уточненной (детализирующей) $D_j(t)$ составляющих. В последующем компоненты уточняются итерационным методом. Каждый шаг уточнения соответствует определенному масштабу, то есть уровню j_0 анализа (декомпозиции) и синтеза (реконструкции) сигнала. Такое представление каждой составляющей сигнала вейвлетами можно рассматривать как во временной, так и в частотной областях.

Вычисление коэффициентов вышеуказанных составляющих осуществляется посредством следующих формул [6; 10]:

$$s_{jk}(t) = \frac{1}{\sqrt{2^j}} \sum_{t=0}^{L} x(t)\phi\left(\frac{t}{2^j} - k\right), \tag{1.44}$$

$$d_{jk}(t) = \frac{1}{\sqrt{2^j}} \sum_{t=0}^{L} x(t)\psi\left(\frac{t}{2^j} - k\right), \tag{1.45}$$

где s_{jk}, d_{jk} - коэффициенты аппроксимации и детализации соответственно, j – параметр масштаба или уровень разложения, L – длина сигнала, ϕ – масштабирующая функция, ψ – вейвлет-функция. Коэффициенты s_{jk}, представляют собой грубое приближение исходного сигнала, а коэффициенты d d_{jk} выражают его локальные особенности, поэтому их соответственно называют коэффициентами аппроксимации и детализации. Функции ϕ и ψ служат высокочастотными и низкочастотными фильтрами соответственно.

Тогда сигнал на уровне разложения j_n может быть разложен в ряд следующего вида [6; 10].

$$x(t) = \sum_{k} s_{j_n k}\,\phi_{j_n k} + \sum_{j \geq j_n k} d_{jk}\,\psi_{jk}, \tag{1.46}$$

Быстрое вейвлет-преобразование – аналог БПФ. Вместо большого количества вычислений, связанных с функциями масштабирования и вейвлета, применяют уже вычисленные коэффициенты соответствующего вейвлета, что значительно сокращает время выполнения вычислений. В

быстром вейвлет-преобразовании используются следующие формулы для вычисления коэффициентов [6]:

$$s_{j+1,k} = \sum_m h_m s_{j,2k+m}\,, \tag{1.47}$$

$$d_{j+1,k} = \sum_m g_m s_{j,2k+m}\,. \tag{1.48}$$

Данные вычисления позволяют рассчитывать коэффициенты разложения на основании коэффициентов предыдущего уровня. В случае быстрого вейвлет-преобразования происходит дальнейшая декомпозиция именно детализирующей составляющей. Пример дерева, которое получается в результате декомпозиции представлен на рисунке 1.1 [10].

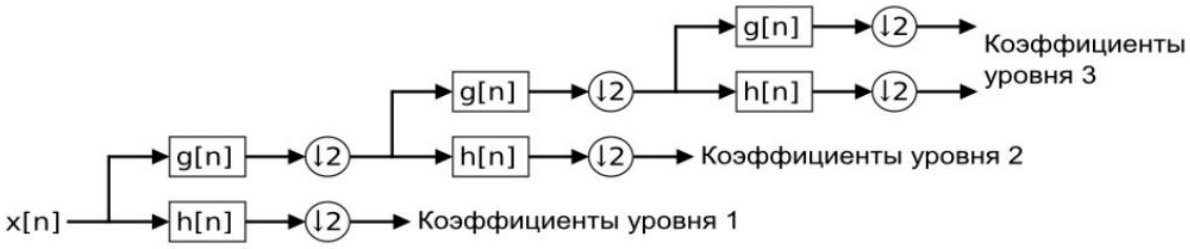

Рисунок 1.1 – Дискретное вейвлет преобразование глубины 3

Таким образом, в практических приложениях с применением быстрого вейвлет-преобразования используются только коэффициенты h_m и g_m, сами же вейвлеты не вычисляются и в расчетах не используются.

1.10 Спектрограмма

Одним из основных инструментов при анализе аудиосигналов и особенно речевых сигналов является спектрограмма. В основном она представляет из себя график, где одним из измерений является время, а другим – частота [2; 22]. При этом, нередко частоты на графике отображаются не линейно, а логарифмически, что связано с тем, что низкие частоты содержат больше информации, чем высокие, что особенно актуально для речевых сигналов. На самом же графике отображается интенсивность сигнала (зачастую в виде децибел) на определенной частоте в определенный момент

времени. Интенсивность в основном отражается цветом (ярче цвет – интенсивнее сигнал). Пример спектрограммы представлен на рисунке 1.2.

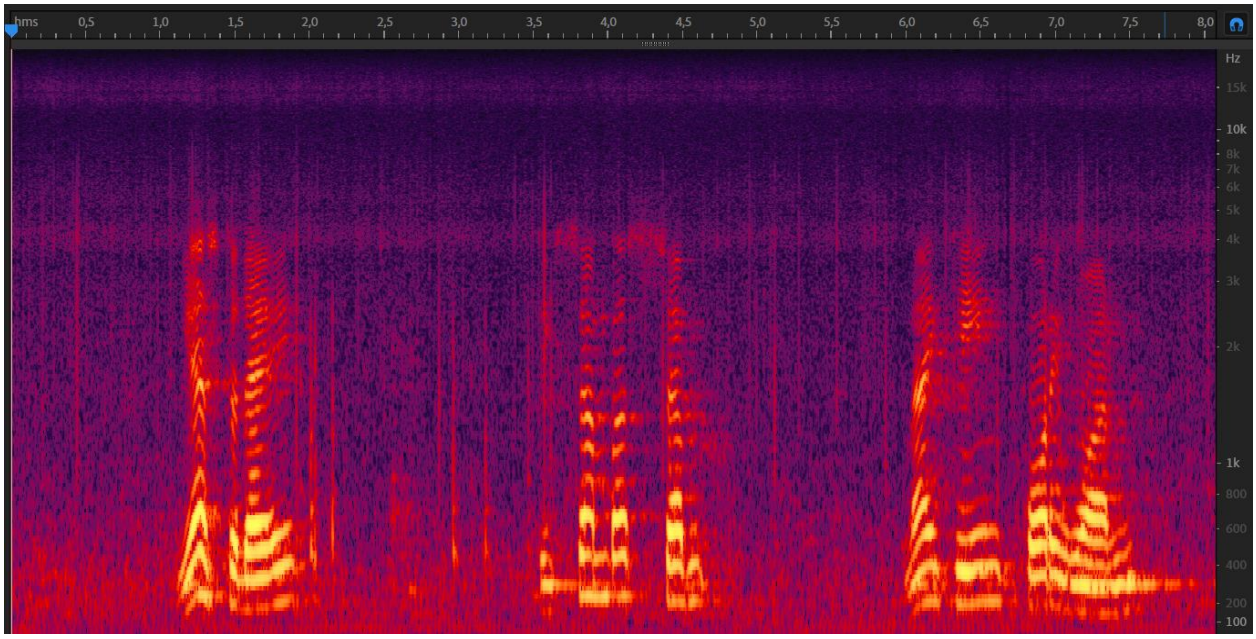

Рисунок 1.2 – Пример спектрограммы из AdobeAudition

Для вычисления спектрограммы дискретного сигнала его разбивают на сегменты, к каждому из которых применяется оконное преобразование Фурье (ДПФ с применением оконной функции). Находя спектр каждого сегмента, мы в итоге получаем спектрограмму [22].

Помимо всего стоит упомянуть, что размер и форма окна анализа могут быть разнообразными, что в последствии влияет на точность получаемых данных. Меньшее (более короткое) окно даст более точные результаты по времени за счет точности представления частоты. Более крупное (более длинное) окно обеспечит более точное представление частоты. Это пример принципа неопределенности Гейзенберга, согласно которому в случае двух сопряженных переменных, чем точнее измеряется одна из них, тем менее точно можно измерить вторую. Так что стоит учитывать это при работе со спектрограммами и выбирать размер сечения в зависимости от величины, чья точность нам важнее [2].

Спектрограммы, в отличие от просто временного или частотного представления сигнала, отображает сразу всю основную интересующую нас информацию: время, частоты и интенсивность сигнала на конкретных частотах в конкретные моменты времени. Это позволяет провести наиболее полный анализ изучаемого сигнала.

ГЛАВА 2
РАЗРАБОТКА ВЫЧИСЛИТЕЛЬНЫХ АЛГОРИТМОВ И РЕАЛИЗАЦИЯ ПРОГРАММЫ ОБРАБОТКИ РЕЧЕВЫХ СИГНАЛОВ

2.1 Алгоритм спектрального вычитания

Одной из основных задач обработки речевого сигнала является повышение его качества путем уменьшения фонового шума при прежней разборчивости речи.

В вопросе шумоподавления в основном прибегают к преобразованию Фурье. Самый простым и распространенным методом подавления шума в речевых сигналах является метод спектрального вычитания. Как и многие другие, этот метод основывается на предположении, что спектр исходного речевого сигнала состоит из спектра чистого сигнала и спектра шума. В этом пункте главы мы сперва рассмотрим базовый алгоритм спектрального вычитания, а затем добавим в него некоторые модификации.

Пусть у нас имеется зашумленный сигнал $x(n)$. Распишем его как сумму чистого сигнала $s(n)$ и сигнала с шумом $d(n)$ [11]. Получим следующее уравнение:

$$x(n) = s(n) + d(n). \tag{2.1}$$

Если воспользуемся дискретным (так как изначальный сигнал дискретный) преобразованием Фурье, то мы перейдем от функций отсчетов времени к функциям частот (спектрам). После применения получаем следующее:

$$\hat{x}(\omega) = \hat{s}(\omega) + \hat{d}(\omega). \tag{2.2}$$

Спектр $\hat{x}(\omega)$ можно представить в экспоненциальной форме:

$$\hat{x}(\omega) = |\hat{x}(\omega)|e^{i\varphi_x(\omega)}, \tag{2.3}$$

где $|\hat{x}(\omega)|$ – амплитудный спектр исходного сигнала, а $\varphi_x(\omega)$ – фазовый.

Аналогично поступим со спектром шума: $\hat{d}(\omega) = \left|\hat{d}(\omega)\right|e^{i\varphi_d(\omega)}$. Поскольку у нас нет точной информации о шумовом сигнале и, соответственно, о его шуме, то придется прибегнуть к некоторым заменам. Что касается фазового спектра, если мы обратимся к теореме запаздывания (1.13) из предыдущей главы, то вспомним, что амплитудный спектр не зависит от сдвига фазового, так что, если мы примем, что $\varphi_d(\omega) = \varphi_x(\omega)$, то мы не потеряем в разборчивости речевого сигнала. Амплитудный же спектр мы заменим средним значением шума исходного сигнала на кадрах без речи, для чего будем использовать детектор речевой активности (о нем позже): $\left|\hat{d}(\omega)\right| = \left|\widehat{d_x}(\omega)\right|$ [11]. Тогда спектр оценки шума имеет следующий вид:

$$\widehat{d_x}(\omega) = \left|\widehat{d_x}(\omega)\right|e^{i\varphi_x(\omega)}. \tag{2.4}$$

Таким образом можем выразить интересующий нас спектр чистого сигнала, который получается при условии оговорок, примененных к спектру шума. То есть мы получаем следующую оценку спектра чистого сигнала:

$$\widehat{s_x}(\omega) = \left(|\hat{x}(\omega)| - \left|\widehat{d_x}(\omega)\right|\right)e^{i\varphi_x(\omega)}. \tag{2.5}$$

Формула (2.5) дает представление о главном принципе, лежащем в основе метода спектрального вычитания. Отметим, что амплитудный спектр чистого сигнала, исходя из этого, выражается следующей формулой:

$$|\widehat{s_x}(\omega)| = |\hat{x}(\omega)| - \left|\widehat{d_x}(\omega)\right|. \tag{2.6}$$

В более общей форме в методе спектрального вычитания амплитудный спектр вычисляется по следующей формуле:

$$|\widehat{s_x}(\omega)|^p = |\hat{x}(\omega)|^p - \left|\widehat{d_x}(\omega)\right|^p, \tag{2.7}$$

где p – показатель степени. В качестве степени обычно берут $p = 2$ – вычитание спектров мощности. Заметим, что при $p = 1$ можно перейти к формуле (2.6).

Заметим, что в уравнении (2.7) можно получить значение меньше нуля, если спектр шума окажется больше спектра сигнала в некоторой точке, что может случиться, учитывая отсутствие гарантии точности оценки шума. Чтобы избежать такой ошибки, сперва введем некоторые обозначения:

$$P_{\widehat{s_x}}(\omega) = |\widehat{s_x}(\omega)|^2, \quad P_{\hat{x}}(\omega) = |\hat{x}(\omega)|^2, \quad P_{\widehat{d_x}}(\omega) = \left|\widehat{d_x}(\omega)\right|^2. \tag{2.8}$$

Тогда запишем вычитание спектров в следующем виде [11]:

$$T(\omega) = P_{\hat{x}}(\omega) - P_{\widehat{d_x}}(\omega),. \tag{2.9}$$

$$P_{\widehat{s_x}}(\omega) = \begin{cases} T(\omega), & T(\omega) > 0, \\ 0, & T(\omega) \leq 0. \end{cases} \tag{2.10}$$

Теперь мы знаем значение амплитуды спектра чистого речевого сигнала. Если применим обратное преобразование Фурье к этой формуле, то получим чистый речевой сигнал. Пусть $\hat{f}^{-1}$ – оператор обратного преобразования Фурье. Тогда чистый речевой сигнал вычисляется по следующей формуле:

$$s(n) = \hat{f}^{-1}\left[\sqrt{P_{\widehat{s_x}}(\omega)}e^{i\varphi_x(\omega)}\right]. \tag{2.11}$$

Перейдем к детектору речевой активности. Он нужен для вычисления степени мощности шума в зашумленном сигнале. Один из вариантов реализации заключается в том, чтобы на каждом кадре вычислять общую мощность энергии сигнала и затем сравнивать ее с предположительной мощностью шума. Тем самым фреймы, на которых мощность энергии больше мощности шума, будем расценивать как фреймы с речью, остальные – с шумом. В нашей программе в качестве энергии сигнала будем использовать децибел – параметр сегментного соотношения сигнал/шум (децибел) [11]:

$$SegRelEn = 10\ln\frac{\sum P_{\hat{x}}(\omega)}{\sum P_{\widehat{d_x}}(\omega)}. \tag{2.12}$$

Тогда в процессе прохода по кадрам мы будем сравнивать вычисленное на нем значение $SegRelEn$ с пороговым значением $N_{treshold}$. Тогда при выполнении неравенства

$$SegRelEn > N_{treshold} \tag{2.13}$$

можно делать вывод, что фрейм содержит речевую активность. Отметим, что $N_{treshold}$ определяется опытным путем [11].

Таким образом оценка спектра мощности шума для фреймов, не содержащих речевую активность, производится методом экспоненциального усреднения [11]:

$$P_{\widehat{d_x}}(\omega) = \gamma P_{\widehat{d_x}}(\omega) + (1-\gamma)P_{\hat{x}}(\omega). \tag{2.14}$$

Здесь γ – коэффициент усреднения, который обычно выбирается в интервале $0{,}9 < \gamma < 1$.

Получается, для каждого фрейма мы сперва определяем, есть ли в поступившем фрейме речевая активность. Если ее там нет, мы пересчитываем общий спектр мощности шума.

Основная проблема базового метода спектрального вычитания заключается в возникновении «музыкального шума». Под «музыкальным шумом» понимают звуковые тона, которые изменяются во времени. Это так называемые спектральные «трассы», которые возникают из-за очень широких пиков. Такие пики возникают из максимумов исходного сигнала, которые в результате вычитания спектра шума просто спустились на несколько тонов вниз, но в отличие от занулившихся минимумов все еще остаются локальными максимумами текущего сигнала. Если эти максимумы достаточно широкие, то тогда и возникают спектральные «трассы» [11].

Модифицируем исходный алгоритм, чтобы избавиться от «музыкальных шумов». Для этого дополним нашу формулу так, чтобы она минимизировала спектральные максимумы. Изменим формулы (2.9) и (2.10) базового алгоритма спектрального вычитания [11]:

$$T(\omega) = P_{\hat{x}}(\omega) - \alpha P_{\widehat{d_x}}(\omega); \tag{2.15}$$

$$P_{\widehat{s_x}}(\omega) = \begin{cases} T(\omega), & T(\omega) > \beta P_{\widehat{d_x}}(\omega), \\ \beta P_{\widehat{d_x}}, & T(\omega) \leq \beta P_{\widehat{d_x}}(\omega). \end{cases} \tag{2.16}$$

Здесь β – параметр, определяющий спектральный минимум шума (то есть компоненты спектра оценки чистого сигнала в результате очистки не становятся меньше значения $\beta P_{\widehat{d_x}}(\omega)$). α – параметр, который влияет на количество удаляемых спектральных максимумов.

Стоит отметить, что при $\alpha = 1$ и $\beta = 0$, формулы (2.15) – (2.16) модифицированного метода будут идентичны формулам (2.9) – (2.10) базового.

Рассмотрим роль коэффициентов α и β. Выбор $\alpha > 1$ позволяет дополнительно уменьшить итоговое значение спектральных максимумов, что позволяет убрать широкополосный шум. β – спектральный порог, который позволяет опуститься значениям в сигнале ниже некоторого порога $\beta P_{\widehat{d_x}}(\omega)$. При $\beta > 0$ трассы сокращаются, поскольку разница между соседними пиками и минимумами не столь велика.

2.2 Алгоритм порогового шумоподавления

В случае, если речевой сигнал представляет собой нестационарный случайный процесс, для его обработки целесообразно применять методы вейвлет-анализа, позволяющие разложить сигнал по функциям, локализованным как в частотной, так и во временной области. В силу этого алгоритмы вейвлет-анализа могут эффективно выделять временные и частотные особенности речевого сигнала.

Конкретно в данном случае речь будет идти о шумоподавлении в речевых сигналах при помощи вейвлет-преобразования, которое имеет ряд преимуществ по сравнению с преобразованием Фурье. Последнее в первую очередь не имеет возможности отразить на спектре временную составляющую, тем самым возникают сложности с работой с нестационарными сигналами. Оконное преобразование Фурье, хоть и дает возможность локализовать время, но все еще не решает данный вопрос полностью. В сравнении с ним вейвлет-преобразование позволяет получить высокое разрешения по времени у высоких частот, а у низких – высокое по частотам. Это дает больше свободы и гибкости при анализе сигналов. Так при помощи вейвлет-преобразования можно выявлять как крупные, так и мелкие структуры сигнала. Также оно меньше подвержено эффекту утечки, который возникает в оконном преобразовании Фурье из-за несоответствия окна анализируемому сигналу.

В случае шумоподавления это интегральное преобразование имеет преимущество при обработке зашумленных сигналов с нестационарными типами шумов. Далее мы рассмотрим один из простейших методов шумоподавления речевого сигнала посредством вейвлет-преобразования: пороговое шумоподавление.

Алгоритм порогового шумоподавления является одним из простейших и состоит из следующих этапов:

1. Принимается решение об используемом вейвлете, а затем на его основе к исходному зашумленному сигналу применяем дискретное вейвлет-преобразование с декомпозицией до некоторого уровня N.
2. Детализация: для уровня N выбирается порог и ко всем уровням применяется пороговая обработка детализирующих коэффициентов.
3. Осуществляется реконструкция сигнала с помощью вейвлет-преобразования, использующая исходные коэффициенты аппроксимации уровня N и скорректированные детализирующие коэффициенты со всех N уровней.

Вейвлет подбирается опытным путем или эмпирически. В результате декомпозиции мы получим N наборов детализирующих коэффициентов и один набор аппроксимирующих. Ключевым вопрос будет заключаться в том, как выбирать пороговое значение для каждого из наборов.

Ключевым элементом данного метода являются пороговые функции разных типов. Они позволяют ограничивать величину детализирующих коэффициентов. Установка конкретного порогового значения и обнуление коэффициентов, не достигающих этого уровня, приводит к существенному подавлению шумовой составляющей. Различают жесткий и мягкий пороги. Пример графиков пороговой функции представлен на рисунке 2.1 [12].

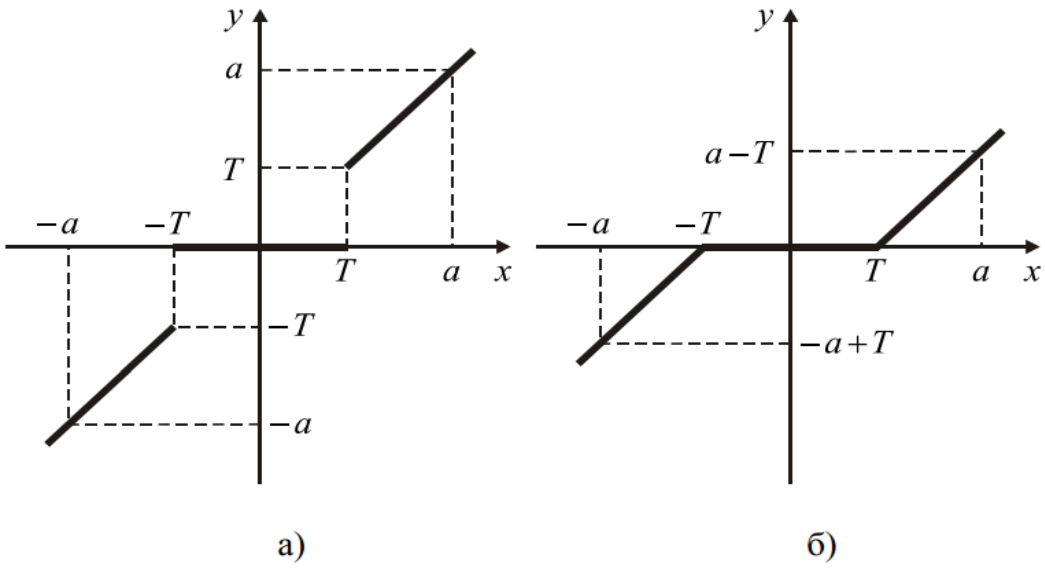

Рисунок 2.1 – Пороговые функции обработки коэффициентов преобразования: а) жесткая пороговая функция; б) мягкая пороговая функция

Сперва рассмотрим жесткую пороговую функцию, пример которой представлен как раз-таки на рисунке 2.1а. Ее можно представить в виде следующего выражения:

$$y(x) = \begin{cases} x, & |x| \geq T, \\ 0, & |x| < T. \end{cases} \tag{2.17}$$

Величина T здесь представляет собой пороговое значение, способ определения которого будет рассмотрен далее, а x и y обозначают соответственно входной и обработанный коэффициенты [12].

Теперь перейдем к мягкой пороговой функции, которую можно было увидеть на рисунке 2.1б. Она в свою очередь представима следующей формулой:

$$y(x) = \begin{cases} sign(x)(|x| - T), & |x| \geq T, \\ 0, & |x| < T. \end{cases} \quad (2.18)$$

Главное различие между мягкой и жесткой пороговыми функциями состоит в отсутствии разрыва в точке порога T у первой. Таким образом, мягкая пороговая функция сохраняет непрерывность, что обеспечивает более качественную обработку зашумленного сигнала вблизи точек разрыва. Однако уменьшение коэффициентов разложения на величину порога при мягкой обработке негативно сказывается на качестве восстановления сигнала. Экспериментальные данные свидетельствуют, что жесткая пороговая обработка демонстрирует лучшие результаты по численным оценкам качества реконструкции [12].

Еще остается открытым вопрос, как выбирать пороговое значение T. Для этого мы используем метод универсального порога (universal threshold), который является самым часто используемым в следствии своей простоты и эффективности [18]. Формула для порога, согласно этому методу, выглядит следующим образом:

$$T = \sigma\sqrt{2\ln(N)}\ , \quad (2.19)$$

где N – длина сигнала, а σ вычисляется методом медианной оценки (median estimate method). Формула выглядит следующим образом:

$$\sigma = \frac{M_x}{0{,}6745}\ , \quad (2.20)$$

где M_x – это медиана по абсолютным значениям множества $\{X\}$ детализирующих коэффициентов самого низкого уровня разложения [12; 18].

После модификации детализирующих коэффициентов остается произвести обратное дискретное вейвлет-преобразование (восстановление сигнала).

2.3 Алгоритм повышения частоты дискретизации

Помимо шумоподавления одной из базовых задач обработки сигналов является изменение частоты дискретизации (передискретизация или ресемплинг). Некоторые устройства и программы работают с сигналами

конкретной частоты, так что ее нужно предварительно изменить во входном сигнале. Как уже разъяснялось в первой главе, сигнал при записи дискретизуется, то есть каждую секунду с периодом T делаются отсчеты сигнала, по которым в последствие восстанавливается сигнал. Величина $F_s = \frac{1}{T}$ и является частотой дискретизации. Суть же передискретизации в том, чтобы изменить количество отсчетов на каждую секунду сигнала. Процесс повышения частоты дискретизации (upsampling) зачастую называют интерполяцией. Обратный же процесс (downsampling) – децимацией [19; 23].

Сперва рассмотрим алгоритм повышения частоты дискретизации. Пусть мы имеем сигнал длины N и частотой дискретизации F_s. В результате применения ДПФ мы получим спектр X, представленный на рисунке 2.2. Спектр содержит N частот, вся основная информация в котором хранится в диапазоне от 0 до $\frac{N}{2}$ Гц. Оставшиеся значения частот от $\frac{N}{2} + 1$ до N называют отрицательным спектром, они являются зеркальным отражением основной части спектра.

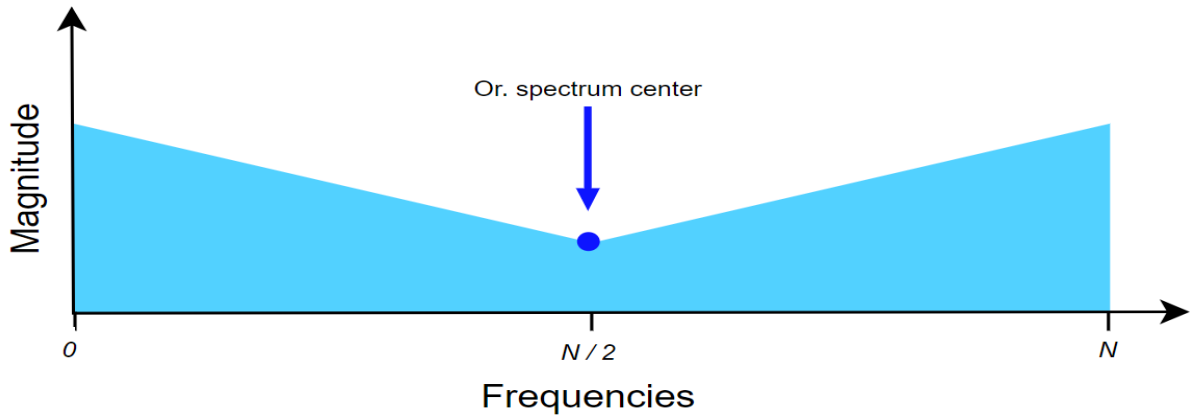

Рисунок 2.2 – Спектр оригинального сигнала

В результате повышения частоты дискретизации количество информации (отсчетов), соответственно, увеличится, так как общее количество отсчетов можно посчитать по формуле:

$$N = F_s * t, \tag{2.21}$$

где N – количество отсчетов, а t – общее время сигнала. Таким образом, для повышения частоты дискретизации нужно в исходный сигнал добавить $N' - N$ отсчетов, где N' – количество отсчетов после повышения частоты. Пусть $F_s' > F_s$– новое значение частоты дискретизации, тогда $k = \frac{F_s'}{F_s} > 1$ – коэффициент увеличения частоты дискретизации и значение N' может быть вычислено по формуле:

$$N' = N * k. \tag{2.22}$$

Таким образом по имеющимся N точкам мы должны спрогнозировать, где будут находится еще $N' - N$ точек сигнала, что, по сути, является задачей интерполяции, а это уже объясняет название метода. Эту интерполяцию мы будем производить при помощи обратного ДПФ: мы поместим на высокие частоты сигнала (в центр нашего спектра) $N' - N$ нулевых точек. Если посмотреть на формулу обратного ДПФ (1.21), то можно обратить внимание, что добавление нулевых частот не будет влиять на общую сумму каждой из компонент восстановленного сигнала. Повлияет же только измененное значение N. В случае множителя $\frac{1}{N}$ перед суммой, оно изменит каждое значение получаемого сигнала в k раз. По этой причине после обратного ДПФ нужно будет произвести масштабирование полученного сигнала, то есть домножение каждого отсчета на k. В случае же знаменателя аргумента экспоненты, изменение числа N позволяет нам получить дополнительные промежуточные значения между уже имеющимися отсчетами. То есть каждый k-й отсчет обработанного сигнала будет иметь значение исходного, а остальные будут получены в результате интерполяции обратным ДПФ при условии, что выходной сигнал будет масштабирован.

Таким образом, после получения спектра X, который изображен на рисунке 2.2, в его середину должны быть добавить нулевые частоты. Так как середина спектра – $\frac{N}{2}$, следовательно, начиная с $\left(\frac{N}{2} + 1\right)$-го элемента нового спектра X', мы помещаем $N' - N$ нулевые частоты, а после них будут располагаться элементы из второй половины X: с $\frac{N}{2} + 1$ по N. Вышеуказанное проиллюстрировано на рисунке 2.3 и может быть формально описано следующей системой:

$$X'(k) = \begin{cases} X(k), & 0 \le k < \dfrac{N}{2} \\ \dfrac{1}{2}X\left(\dfrac{N}{2}\right), & k = \dfrac{N}{2} \\ 0, & \dfrac{N}{2} < k < N' - \dfrac{N}{2} \\ \dfrac{1}{2}X\left(\dfrac{N}{2}\right), & k = N' - \dfrac{N}{2} \\ X(k - N' + N), & N' - \dfrac{N}{2} < k < N' \end{cases} \tag{2.23}$$

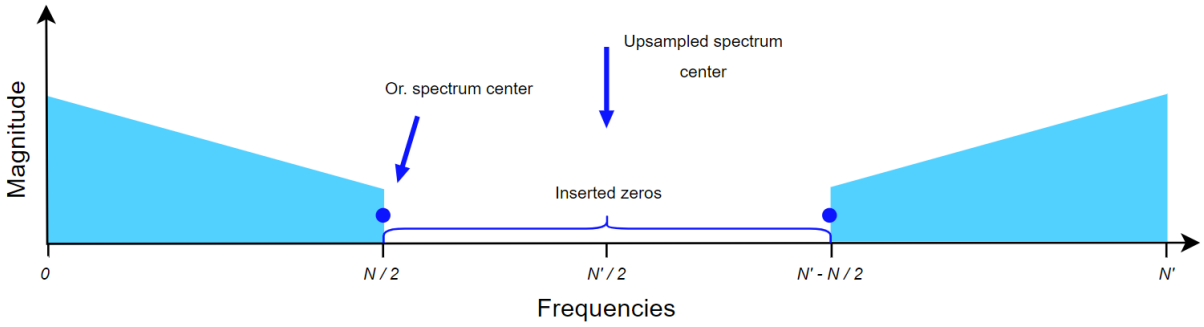

Рисунок 2.3 – Спектр сигнала после вставки в него нулевых элементов для дальнейшей интерполяции

Как видно из формулы (2.21), на месте разрыва спектра X значение нового спектра X' равно половине предыдущей середины.

Таким образом мы дополнили изначальный спектр до нужного размера и можем применить операцию обратного ДПФ. Получившийся результат, как уже отмечалось раньше, нужно масштабировать умножением всех отсчетов на k. Полученный сигнал имеет длину N'и частоту дискретизации F_s', так что мы достигли повышения дискретизации [19; 23].

2.4 Алгоритм понижения частоты дискретизации

Снова рассмотрим сигнал длины N с частотой дискретизации F_s. Сразу обозначим, что мы должны получить частоту дискретизации $F_s' < F_s$, коэффициент понижения частоты дискретизации, соответственно, $k = \frac{F_s'}{F_s} < 1$ и новый сигнал, который мы должны получить, имеет длину, вычисляемую по формуле (2.22). В результате применения ДПФ мы снова получаем спектр X с рисунка 2.2.

В результате понижения частоты дискретизации мы получим сигнал с меньшим количеством отсчетов, что объясняется по аналогии с интерполяцией. Соответственно, для понижения частоты дискретизации нам нужно удалить из сигнала $N - N'$ отсчетов сигнала. Для этого, как и в прошлом случае, воспользуемся ДПФ. Получив при помощи него спектр, мы удалим из его середины те самые $N' - N$ отсчетов. Если опять же обращаться к формуле обратного ДПФ, то все будет происходить по аналогии с интерполяцией: основное влияние производит изменение числа N. Множитель перед суммой изменяет отсчеты в k раз относительно от их значения в сигнале до децимации, что мы компенсируем масштабированием после обратного ДПФ. Знаменатель аргумента экспоненты увеличивает степень числа e в k раз.

Соответственно, мы «проскакиваем» некоторые значения отсчетов, оставляя только каждый $\frac{1}{k}$ -й отсчет исходного сигнала. Тем самым мы получаем прореживание сигнала и, соответственно, частота дискретизации уменьшается. Единственное, нужно отметить, что если в случае интерполяции добавление к сумме нулевых элементов в обратном ДПФ не влияло на значение отсчетов сигнала, то в случае децимации, недосчитавшись некоторых удаленных значений спектра, сумма в итоге изменится, так что в оставшихся после децимации отсчетах не всегда будет содержаться та же информация, что и до прореживания. Подробнее про это и другие проблемы будет сказано в следующем пункте главы.

Вернемся к спектру X. Как уже говорилось, чтобы получить спектр X', нужно удалить из середины его спектра $N - N'$ значений. Соответственно, для этого из спектра X мы возьмем первые $\frac{N'}{2}$ и последние $\frac{N'}{2}$ элементов. Тем самым по обе стороны от старого центра спектра $\frac{N}{2}$ будет удалено по $\frac{N-N'}{2}$ частот. Иллюстрация этого этапа приведена на рисунке 2.4. Формально же это все можно описать следующей системой:

$$X'(k) = \begin{cases} X(k), & 0 \le k < \frac{N'}{2} \\ 0, & k = N' - \frac{N}{2} \\ X(N - N' + k), & \frac{N'}{2} < k < N'. \end{cases} \quad (2.24)$$

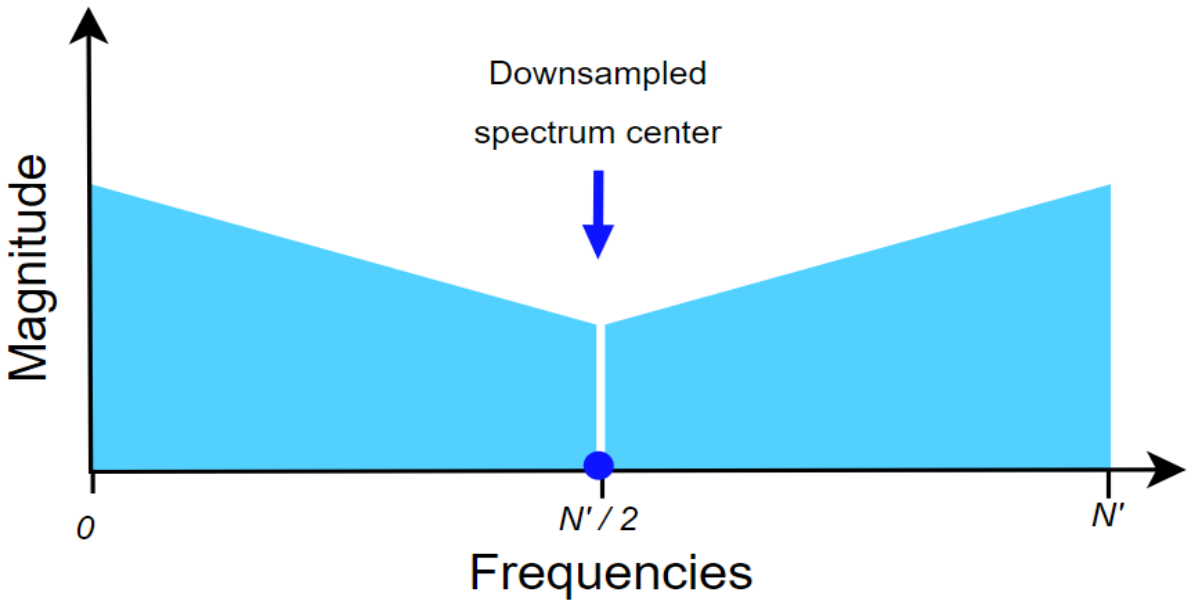

Рисунок 2.4 – Спектр сигнала после удаления из него элементов в процессе децимации

Как видно, в точке разрыва спектра X значение нового спектра X' будет равно нулю.

Мы получили спектр X' длины N'. Применив обратное ДПФ, мы получим сигнал, который нужно будет масштабировать. Для этого домножим каждый элемент сигнала на число k. Теперь мы получили сигнал длины N' и с частотой дискретизации F_s'. Это прореженный сигнал, с необходимой нам пониженной частотой дискретизации [23; 24].

2.5 Передискретизация с нецелым коэффициентом изменения

Описанные выше методы децимации и интерполяции хорошо работают при изменении частоты дискретизации в целое число раз. Теперь же предположим, что это число – нецелое. Чтобы произвести интерполяцию или децимацию нужно добавить или удалить нецелое количество отсчетов в спектре, что, конечно же, сделать невозможно. Для преодоления этой проблемы можно предложить два подхода [17; 23].

Для начала предположим, что входной сигнал X имеет N_{in} отсчетов и частоту дискретизации F_s, а после изменения мы должны получить частоту дискретизации F_s' и количество отсчетов N_{out}. Запишем отношение частот в следующем виде:

$$\frac{F_s'}{F_s} = \frac{P}{Q}, \tag{2.25}$$

где P и Q – целые числа без общих делителей. Так, например, если $F_s' = 48$ кГц и $F_s = 44{,}1$ кГц, то

$$\frac{F_s'}{F_s} = \frac{160}{147}. \tag{2.26}$$

Первый подход будет заключатся в последовательном применении интерполяции и децимации к сигналу, соответственно, с коэффициентами P и Q. То есть в случае рассматриваемого примера мы сделаем интерполяцию с коэффициентом 160, а затем к полученному сигналу применим децимацию с коэффициентом 138.

Такой метод может быть вполне применим, если P и Q – небольшие числа. Однако с числами как в рассматриваемом примере могут возникнуть

ряд проблем, связанных с огромным объемом данных. Из-за этого могут возникнуть переполнение данных или неприемлемо длительные вычисления.

По этой причине более универсально и быстро использование метода изменения размера входных данных. Он требует только одного применения ДПФ и обратного ДПФ. Суть метода состоит в следующем: мы к исходному сигналу длины N_{in} добавляем нули до размера $N > N_{in}$. Затем к этому набору данных мы применяем ДПФ и уже спектральные данные длины N мы дополняем или усекаем до размера N' в зависимости от применения интерполяции или децимации. После применения обратного ДПФ нужно убрать $N' - N_{out}$ нулей из конца полученного сигнала.

Остается только разобраться в выборе числа N. Вернемся к уравнению (2.25) и пусть

$$N = QM, \tag{2.27}$$

$$N' = PM, \tag{2.28}$$

где M – целое число, выбор которого мы сейчас разберем. Суть в том, что если изначально нам надо получить $N' = N\frac{P}{Q}$, где NP не кратно Q, так что число получится нецелым, то в случае выбора $N = QM$, мы получим, что $N' = QM\frac{P}{Q} = MP$ и это число будет целым. Число M мы будем выбирать таким образом, чтобы N было как можно ближе к N'. Также стоит отметить, что так как хотя бы одно из чисел P и Q нечетно (иначе они имеют общий делитель), а N и N' – четны, то число M также должно быть четным. Таким образом имеем следующую формулу:

$$M = 2\left\lceil\frac{N_{in}}{2Q}\right\rceil. \tag{2.29}$$

Теперь вернемся к примеру, где $P = 160$, а $Q = 147$ и пусть изначальная длина сигнала будет $N_{in} = 600.000$ отсчетов (то есть при $F_s = 44{,}1$ кГц это около $13{,}6$ секунд). По формуле (2.29) имеем, что $M = 2\left\lceil\frac{600.000}{294}\right\rceil = 4082$. Тогда $N = QM = 600.054$, то есть нужно дополнить исходный сигнал 54 нулями. После применения ДПФ мы находим $N' = PM = 653.120$, так что нам нужно дополнить N еще 53.066-ю нулями (это уже подготовка к интерполяции). Затем к N' элементам спектра мы применяем обратное ДПФ, и остается только убрать лишние нули. Если

$$N_{out} = \left\lceil \frac{F_s'}{F_s} N_{in} \right\rceil, \tag{2.30}$$

то нам остается убрать из сигнала $N' - N_{out}$ избыточных нулей, и мы получим требуемый сигнал длины N_{out}. В случае рассматриваемого примера $N_{out} = \left\lceil \frac{160}{147} 600.000 \right\rceil = 653.062$ и тогда из сигнала нужно убрать 58 нулей.

Стоит отметить, что при выборе числа M может выбираться число чуть большее, чем вычисляемое по формуле (2.29), а именно, которое можно будет разложить на более маленькие простые числа. Это полезно, так как быстрое ДПФ изначально подразумевает работу с данными длины, равной степени двойки, иначе оно работает медленнее. Но если длина данных раскладывается на простые числа небольшой величины, то этот процесс становится сравнительно быстрее.

Напоследок вернемся к проблеме, с которой мы столкнулись при децимации. Обрезав часть частот исходного сигнала, как уже было сказано, некоторое количество информации будет потеряно. При этом, если в удаляемых частотах содержится незначительная часть сигнала, то есть высоких частот в сигнале мало, то такие потери несущественны и на слух разница заметна не будет. Но понижать частоту до совсем низких частот не стоит, так как удалив более значимую их часть, можно потерять качество сигнала вплоть до полной потери разборчивости. Стоит помнить, что человек способен распознавать звуки в диапазоне частот от 5 то 20 кГц. Так что потерю этой части частот допускать точно не стоит.

Также отметим, что при изменении частоты дискретизации мы изменяем и частоту Найквиста (половина частоты дискретизации). Вследствие теоремы Котельникова, выполняя децимацию, мы можем столкнуться с проблемой: при снижении значения частоты Найквиста, под этой границей могут оказаться частоты ранее удовлетворявшие условию теоремы. Тогда может возникнуть наложение спектра, которого раньше не было. Если для децимации с небольшим коэффициентом и спектром без большого количества высокочастотного сигнала, этот эффект будет незначительным и не будет неразличим на слух, то в иных случаях могут возникнуть сильные дефекты сигнала, которые будут давать неприемлемый результат. Во избежание наложения спектра можно убирать все частоты, находящиеся выше частоты Найквиста или же использовать сглаживающие фильтры. В обоих случаях качество сигнала по итогу упадет, так что, применяя децимацию, всегда стоит учитывать сопутствующие потери качества.

2.6 Общая структура программы

Для написания программы использовался язык программирования Python, а также дополнительный функционал, представленный в некоторых сторонних библиотеках:

- NumPy – библиотека, поддерживающая работу с многомерными массивами и высокоуровневые математические функции;
- SciPy – библиотека, предназначенная для выполнения научных и инженерных расчетов. Среди прочего обладает возможностью обрабатывать сигналы;
- PyWavelets – библиотека, предоставляющая функционал для применения вейвлет преобразований к различным видам сигналов;
- Matplotlib – библиотека, позволяющая визуализировать данные графикой.

Программа состоит из трех модулей: модуль для подавления шума в речевых сигналах при помощи преобразования Фурье (спектральное вычитание) – приложение А; модуль для шумоподавления при помощи вейвлет-преобразования (пороговое шумоподавление) – приложение Б; модуль передискретизации речевых сигналов на основе преобразования Фурье – приложение В. Далее каждому модулю будет уделен свой пункт, где будет подробно рассмотрен алгоритм работы данного модуля. Результирующие графики и метрики будут подробно рассмотрены в следующей главе.

2.7 Модуль шумоподавления при помощи преобразования Фурье

Функционал модуля заключается в следующем. Сперва модуль программы осуществляет чтение аудиофайла с записью зашумленного речевого сигнала в формате WAV. Затем осуществляется подавление шума при помощи модифицированного метода спектральных вычитаний. Полученный очищенный сигнал сохраняется в виде WAV файла, а затем выводятся амплитудные графики исходного и очищенного сигналов, а также разностный график двух сигналов.

Подробнее алгоритм модуля выглядит следующим образом [11]:

1. Производится чтение сигнала при помощи функции библиотеки SciPy и последующая его нормализация (во избежание переполнения при дальнейшей работе с сигналом).

2. По заданной длительности фрейма и полученной из сигнала частоты дискретизации, определяем количество отсчетов в одном фрейме. На основе этого вычисляем размер перекрытия фреймов (перекрытие предотвращает резкие амплитудные перепады на стыках фреймов), а также задается оконная функция необходимого размера (необходима для предотвращения растекания спектра). Оконная функция также нормализуется.

3. Вычисляется спектр мощности шума по первым 5-ти фреймам сигнала (предполагается, что сигнал начинается с шума, а не речи).

4. Осуществляется пофреймовое спектральное вычитание. Вычисляется спектр сигнала при помощи быстрого дискретного преобразования Фурье, реализованного в библиотеке NumPy. Производится модифицированное спектральное вычитание по формулам, описанным в прошлой главе. Затем выполняется детектинг речевой активности и пересчет спектра мощности шума, если фрейм содержит шум. В конце выполняется обратное дискретное преобразование Фурье и полученный сигнал суммируется с результирующим с учетом перекрытия.

5. Полученный обработанный сигнал возвращается к исходным масштабам (был нормирован).

6. Производится запись сигнала в формате WAV.

7. Производится подсчет метрик на основе исходного и обработанного сигналов.

8. Выводятся графики исходного зашумленного и полученного очищенного сигналов при помощи функционала библиотеки Matplotlib, у каждого из которых прописываются метрики, а также разностный график двух сигналов.

В результате выполнения модуля программы мы получаем очищенный от шума сигнал. Также мы получаем амплитудные графики исходного и обработанного сигналов, а также для наглядного сравнения результатов работы программы – разностный график двух сигналов. Помимо этого, на графиках указываются показатели нескольких метрик, о которых мы поговорим подробнее в следующей главе, где также будет рассмотрено несколько примеров работы модуля и проанализируем полученные результаты.

2.8 Модуль шумоподавления при помощи вейвлет-преобразования

Функционал модуля программы аналогичен оному методу, основанному на преобразовании Фурье. Данный модуль также осуществляет чтение аудиофайла с записью зашумленного речевого сигнала в формате WAV, а затем осуществляет подавление шума при помощи ране описанного метода порогового шумоподавления. Результат, представленный очищенным сигналом, сохраняется в виде WAV файла, а затем выводятся амплитудные графики исходного и очищенного сигналов и разностный график двух сигналов.

Подробный алгоритм модуля программы:

1. Производим чтение зашумленного сигнала.
2. В качестве материнского вейвлета выбираем D4 из семейства вейвлетов Добеши и при помощи встроенной функции библиотеки PyWavelets производим декомпозицию сигнала на N уровней и в результате этого получаем список из детализирующих коэффициентов всех уровней и аппроксимирующих последнего уровня.
3. Производится вычисление медианы детализирующих коэффициентов на каждом из уровней декомпозиции и на ее основе пороговое значение по формуле (2.19).
4. Проходим по коэффициентам разложения и, согласно формуле (2.17), оставляем текущее значение или зануляем его.
5. Восстанавливаем очищенный от шума сигнал при помощи все тех же встроенных функций библиотеки PyWavelets.
6. Производим запись сигнала в формате WAV.
7. Производится подсчет метрик на основе исходного и обработанного сигналов.
9. Выводятся графики исходного зашумленного и полученного очищенного сигналов при помощи функционала библиотеки Matplotlib, у каждого из которых прописываются метрики, а также разностный график двух сигналов.

После исполнения модуля программы мы получаем очищенный от шума сигнал и амплитудные графики исходного и обработанного сигналов, а также разностный график двух сигналов. Результат процедуры шумоподавления все так же будет оцениваться на основе метрик и графиков.

2.9 Модуль передискретизации речевых сигналов

Данный модуль осуществляет чтение аудиофайла в формате WAV и получает информацию о новом значении частоты дискретизации сигнала, затем меняет размер входных данных согласно коэффициенту масштабирования частоты дискретизации, увеличивает или урезает сигнал в зависимости от коэффициента, согласно алгоритмам интерполяции или децимации соответственно. После применения алгоритма удаляются оставшиеся избыточные нули и происходить запись сигнала в новый WAV файл с новой частотой дискретизации. В конце происходит вывод на экран амплитудных графиков исходного и передискретизованного сигналов, а также соответствующих метрик.

Подробнее алгоритм модуля данной программы выглядит следующим образом [24]:

1. Производится чтение сигнала при помощи функции библиотеки SciPy.
2. После получения информации о длине и частоте дискретизации сигнала, находятся числа P и Q из формулы (2.25), при помощи вычисления НОД чисел F_s и F_s'. Находится M согласно формуле (2.29) и вычисляется новая длина сигнала N по формуле (2.27) и длина спектра N' (2.28), которую надо будет получить после его дополнения или усечения.
3. Изначальный сигнал дополняется нулями до размера N'.
4. Применяется БПФ (FFT), реализованное в библиотеке NumPy, для получения спектра сигнала.
5. В зависимости от того, нужно нам повысить или понизить частоту, выполняется дополнение или усечение спектра до размера N' по формулам (2.23) и (2.24) соответственно.
6. Применяется обратное быстрое ДПФ.
7. Масштабируется сигнал, посредством домножения каждого элемента на коэффициент масштабирования
8. Производится удаление лишних нулей в конце сигнала.
9. Производится запись сигнала с новым значением частоты дискретизации в формате WAV.
10. Выводятся графики исходного зашумленного и полученного очищенного сигналов при помощи функционала библиотеки Matplotlib, а также подсчитанные метрики к каждому из них.

В результате выполнения модуля программы, реализованной на основе этого алгоритма, мы получим аудиофайл с измененной частотой дискретизации.

ГЛАВА 3
АНАЛИЗ ПОЛУЧЕННЫХ ДАННЫХ

3.1 Выбор объективной оценки

При анализе обработанного речевого сигнала нужно учитывать, что основным показателем качества обработки сигнала является сохранение или улучшение разборчивости речи и отсутствие возможных в результате обработки сигнала различных артефактов. При этом, поскольку конечной целью является сохранение или улучшение качества сигнала при прослушивании его человеком, основная оценка качества проводится субъективно путем опроса экспертов об их восприятии результата. В частности, к таким методам относится Mean Opinion Score (MOS) - эксперты выставляют оценки от 1 (плохо) до 5 (прекрасно), и тогда сама оценка — это среднее арифметическое оценок экспертов [1].

Однако был разработан и ряд методов объективной оценки качества обработанных речевых сигналов. Их можно разделить на два типа: методы с эталонным сигналом (интрузивные) и методы без эталонного сигнала (неинтрузивные).

Особенность первых заключается в том, что такие методы применяются в случаях, когда у нас есть идеальный сигнал, который мы стремимся получить, что, очевидно, возможно только в лабораторных условиях. К таким методам относят семейство стандартов PESQ, в частности, стандарт ITU-T P.862, который применяется в телекоммуникациях. Более современный стандарт – POLQA (ITU-T P.863). Для более узконаправленных задач также используются метрики STOI, SI-SDR, VISQOL и другие [1].

К другому типу объективных оценок относят методы, которые могут быть применены без наличия эталонной версии сигнала, что более актуально на практике. Среди таких методов можно выделить ITU-T P.563 – стандарт для телекоммуникаций и VoIP (Skype, Zoom); ANIQUE+ – основан на психоакустической модели человеческого слуха и анализирует сигнал с этой точки зрения [1]. В последние годы стали активно развиваться нейронные сети, что хоть и является более тяжеловесным решением, но в то же время зачастую и более точным. Одним из таких примеров оказалась нейронная сеть NISQA.

При выборе инструмента оценивания качества, хотелось отдать предпочтение неинтрузивным методам, поскольку они не ограничивают нас

наличием эталонного сигнала, что позволяют работать с неискусственными записями, которые были сделаны в реальных условиях с натуральными шумами. Среди же решений в этой категории выбор был сделан в пользу нейронных сетей, которые при качественном выборе позволяют получить высокую точность, при этом получив множество оценок по разным метрикам для более детального анализа. Таким образом, мы остановились на нейронной сети NISQA, которая, например, в доработанном виде используется в алгоритмах проверки качества потокового аудио платформ VK Video и VK Звонки [21]. Нейронная сеть в результате анализа предоставляет ряд метрик, каждая из которых выдает оценку от 1 (плохо) до 5 (отлично) [20; 21]:

- noisiness (зашумленность) – степень зашумленности;
- coloration (окраска) – разборчивость речи;
- discontinuity (прерываемость) – прерывания в речи;
- loudness (громкость) – комфортность громкости звука (1 может сигнализировать как и о слишком громком, так и о слишком тихом звуке);
- MOS – общее качество сигнала (уже рассматривалось в начале главы).

Такое обилие метрик позволяет сделать более точные выводы о качестве анализируемого речевого сигнала и в дальнейшем мы будем опираться именно на них.

3.2 Результаты шумоподавления при помощи преобразования Фурье

В результате выполнения работы модуля шумоподавления методом спектрального вычитания мы получаем очищенный от шума сигнал. При анализе будем рассматривать представленные графики, метрики на этих графиках и спектрограммы исходного и обработанного сигналов, полученные в программе AdobeAudition. Также важным показателем является восприятие сигнала на слух, хоть это и субъективный показатель. Далее будут приведены несколько примеров с разными речевыми сигналами при разных параметрах спектрального вычитания (константы α, β). γ будет равен 0,9. Дополнительно, сравним эффективность обычного метода спектрального вычитания и модифицированного (обычный метод можно получить, если взять параметры $\alpha = 1$ и $\beta = 0$).

Рассмотрим первый сигнал (roosevelt_speech.wav). Начнем с обычного метода спектрального вычитания: $\alpha = 1$ и $\beta = 0$. На рисунке 3.1 представлен результат выполнения программы.

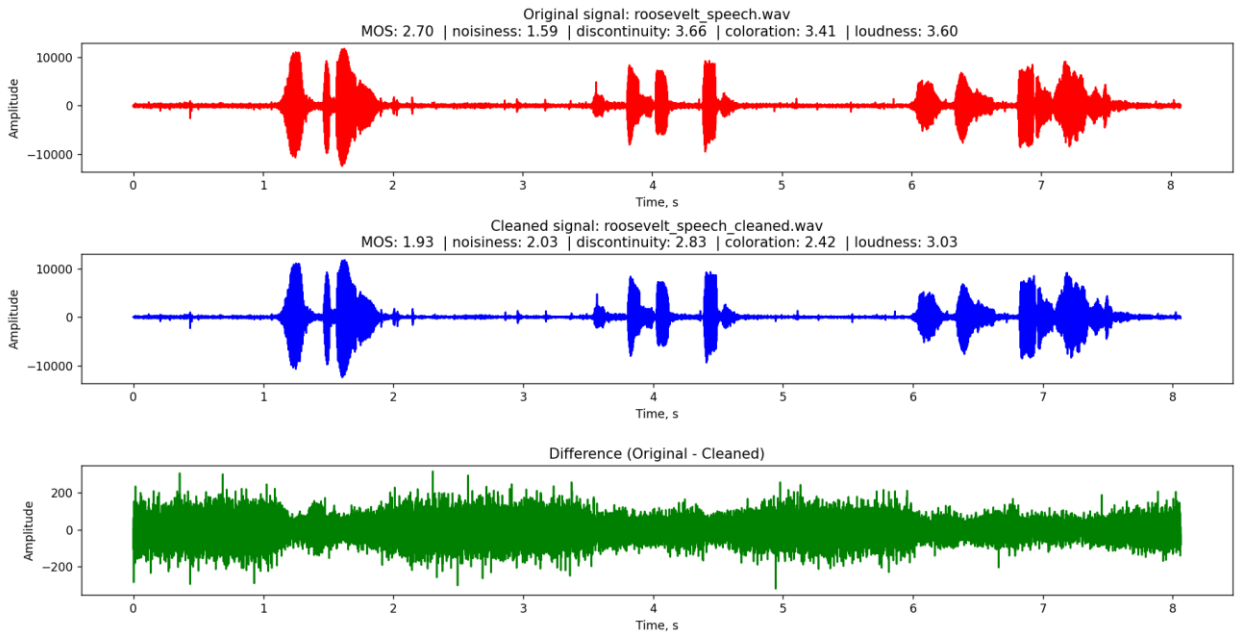

Рисунок 3.1 – Результат выполнения первого модуля программы для roosevelt_speech.wav при $\alpha = 1$ и $\beta = 0$

По метрикам видно, что количество шума было уменьшено, но в то же время упали остальные показатели, которые свидетельствуют об общем падении качества речи на записи. Спектрограммы исходного и полученного сигналов изображены на рисунках 3.2 и 3.3 соответственно.

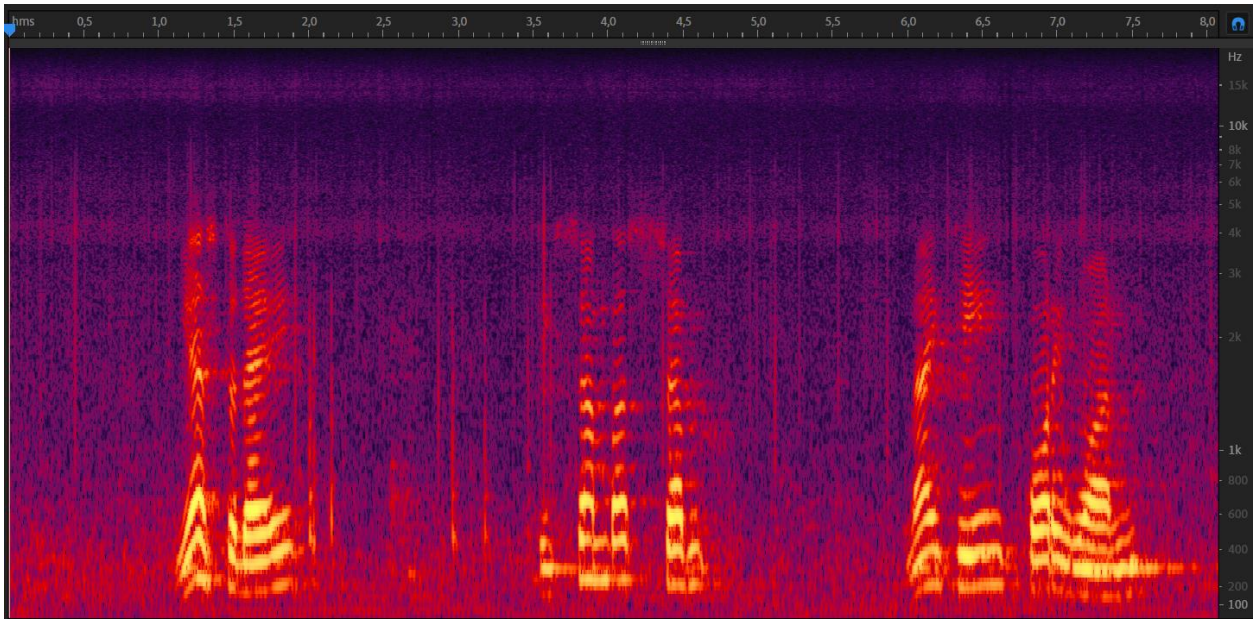

Рисунок 3.2 – Спектрограмма roosevelt_speech.wav до обработки

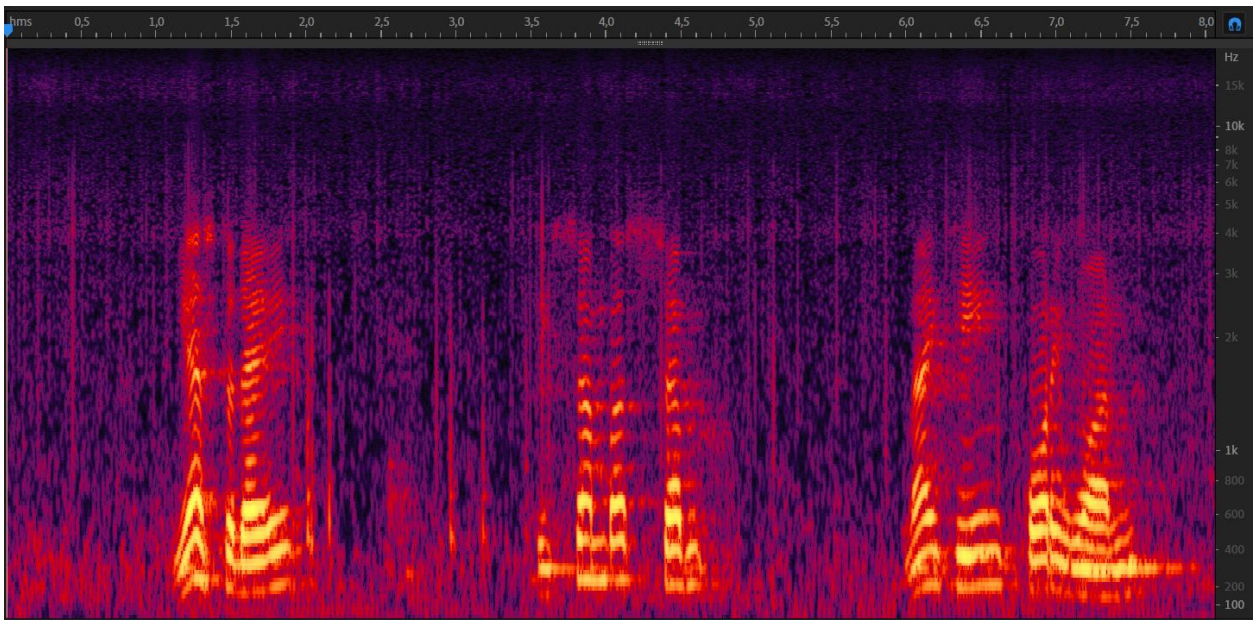

Рисунок 3.3 – Спектрограмма roosevelt_speech.wav после спектрального вычитания при $\alpha = 1$ и $\beta = 0$

Если же применить модифицированный метод, например, при $\alpha = 20$ и $\beta = 0{,}002$ мы получим результат, представленный на рисунке 3.4, и спектрограмму на рисунке 3.5.

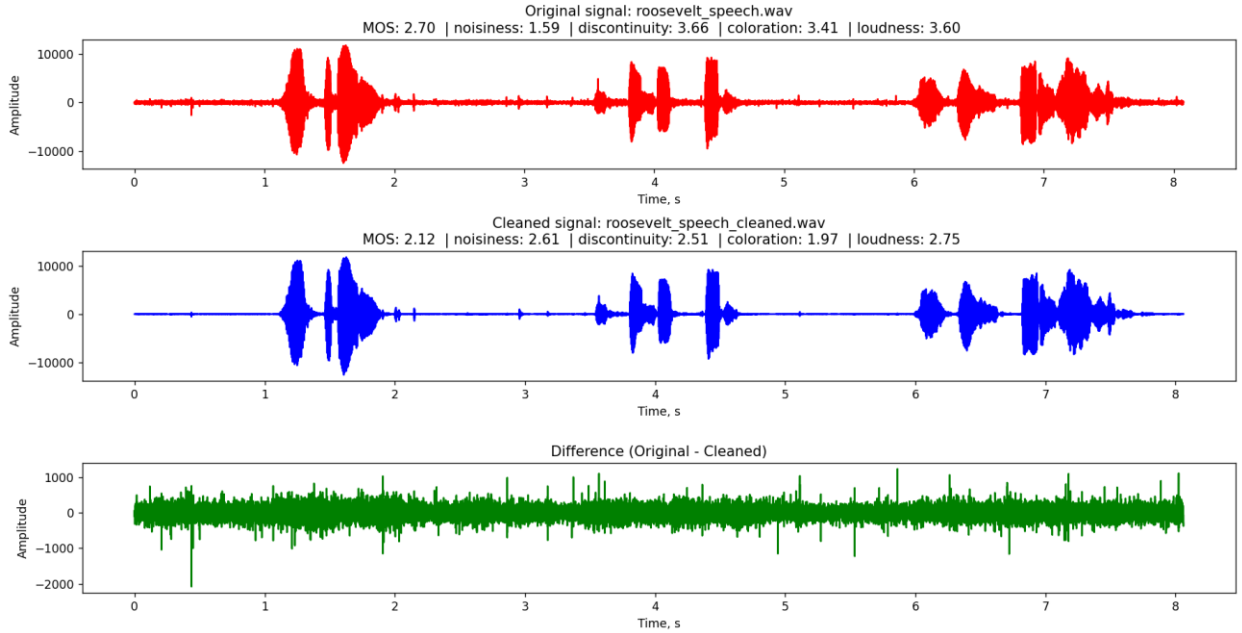

Рисунок 3.4 – Результат выполнения первого модуля программы для roosevelt_speech.wav при $\alpha = 20$ и $\beta = 0,002$

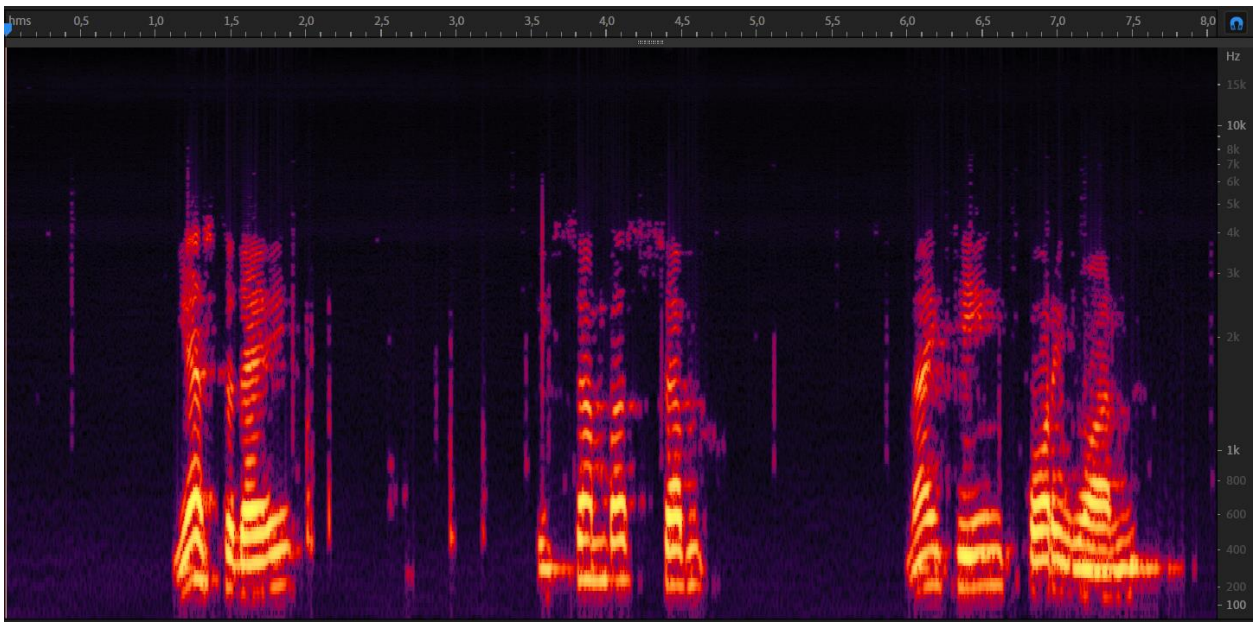

Рисунок 3.5 – Спектрограмма roosevelt_speech.wav после спектрального вычитания при $\alpha = 20$ и $\beta = 0,002$

В случае базового метода спектрального вычитания видно, что хоть небольшая часть шума и была убрана, но его, во-первых, все еще слишком много, а во-вторых, начал проявляться музыкальный шум, что делает прослушивание речевого сигнала еще более затруднительным.

В случае модифицированного метода был выставлен довольно большой параметр α, для отсечения большего количества пиков. В итоге и на диаграмме сигнала и на спектрограмме видно, что практически весь шум был убран при отсутствии возникновения музыкальных тонов.

Но в то же время не забываем и про то, что остальные метрики упали по сравнению с исходным сигналом, причем в модифицированном методе это произошло значительней. Это связано с тем, что более сильное шумоподавление урезает часть нужных для голоса частот, что влияет на его общее восприятие.

Рассмотрим еще один сигнал (churchill_report.wav). Как и в прошлом случае рассмотрим два метода, но модифицированный теперь будет работать при параметрах $\alpha = 10$ и $\beta = 0{,}001$. Графики после выполнения программы представлены на рисунках 3.6 и 3.7, а спектрограммы исходного и полученных сигналов на рисунках 3.8 – 3.10.

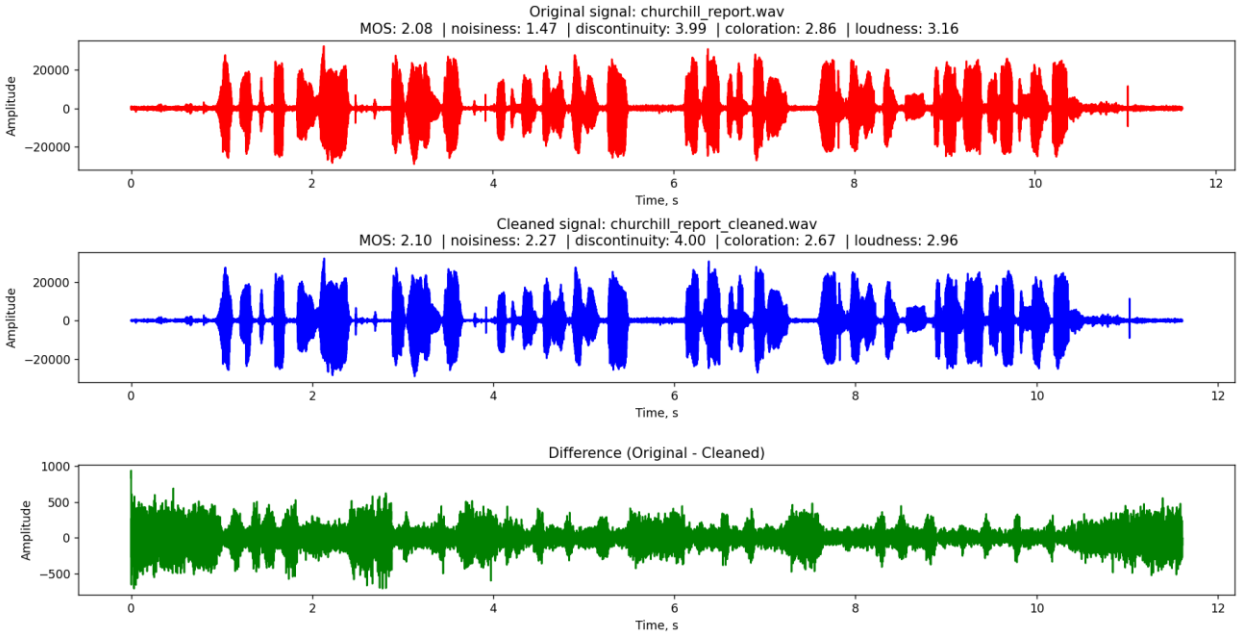

Рисунок 3.6 – Результат выполнения первого модуля программы для churchill_report.wav при $\alpha = 1$ и $\beta = 0$

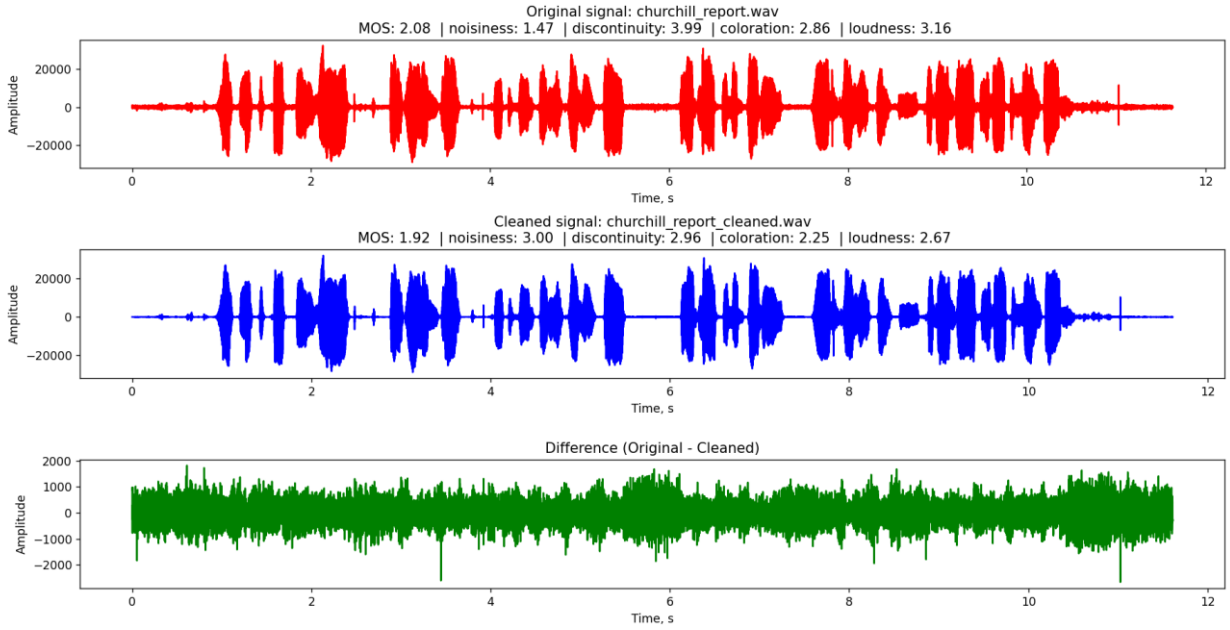

Рисунок 3.7 – Результат выполнения первого модуля программы для churchill_report.wav при $\alpha = 10$ и $\beta = 0,001$

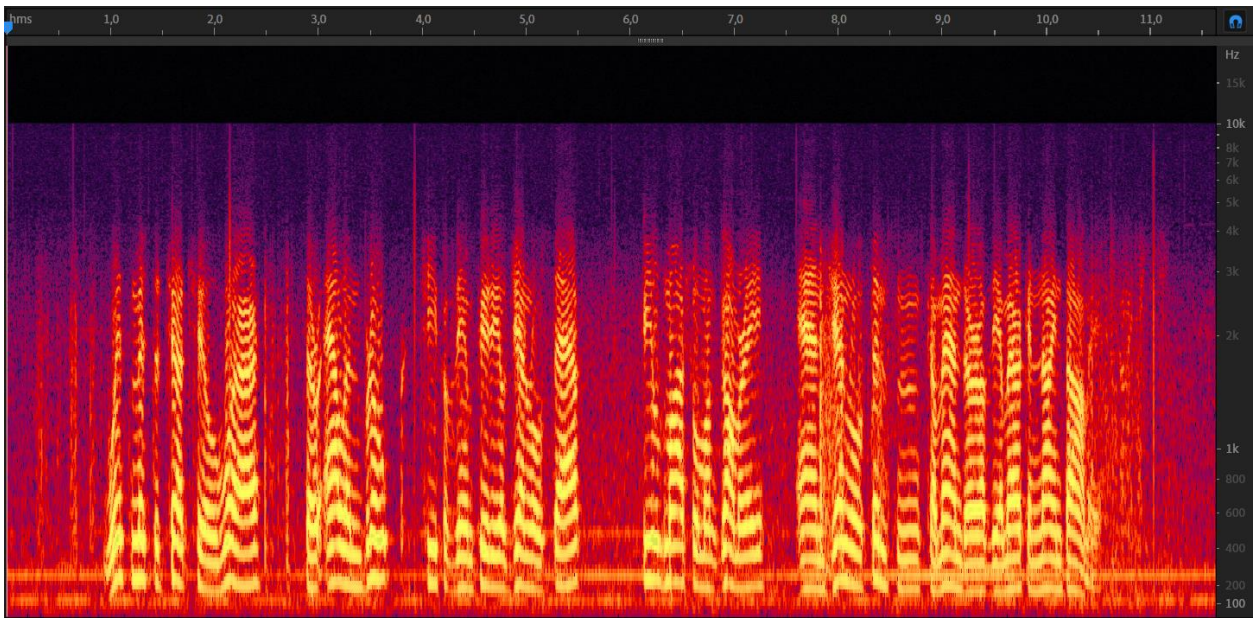

Рисунок 3.8 – Спектрограмма churchill_report.wav до обработки

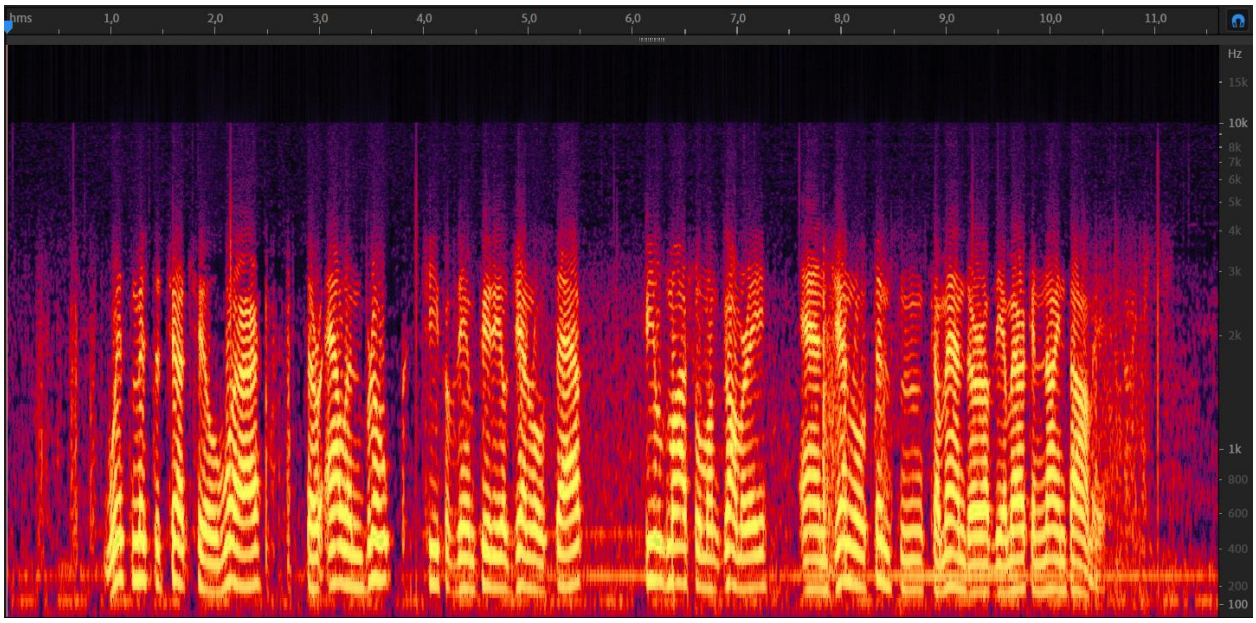

Рисунок 3.9 – Спектрограмма churchill_report.wav после спектрального вычитания при $\alpha = 1$ и $\beta = 0$

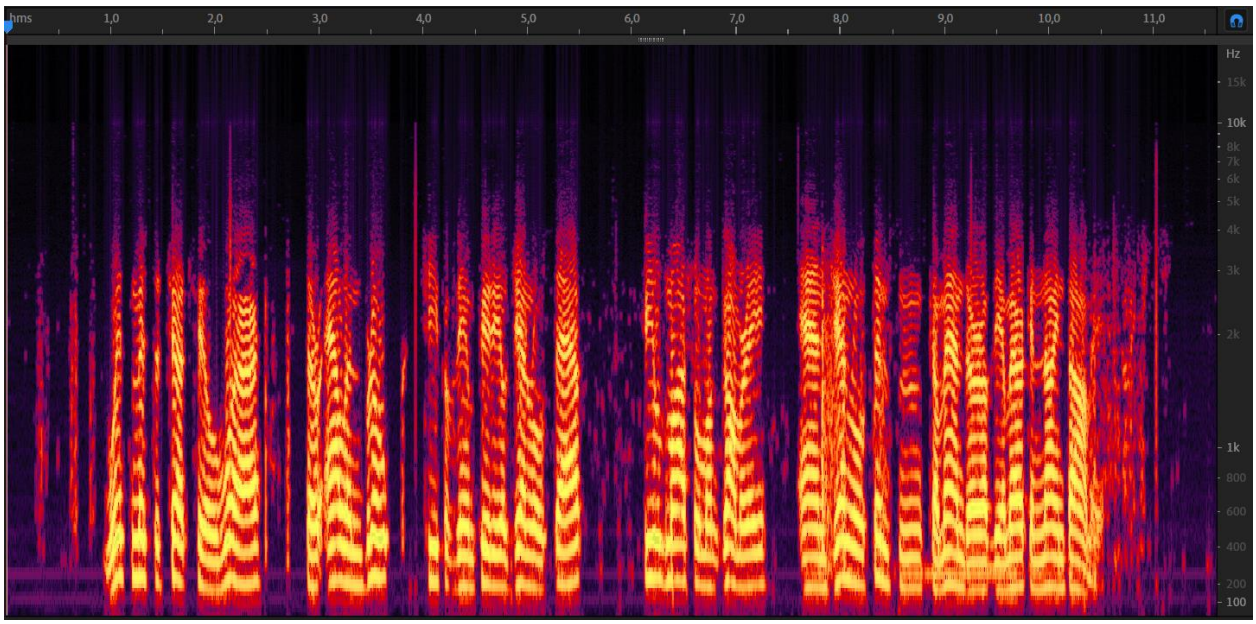

Рисунок 3.10 – Спектрограмма churchill_report.wav после спектрального вычитания при $\alpha = 10$ и $\beta = 0,001$

Здесь интересно обратить внимание на значение MOS двух оценок. В первом случае оно выше, хотя шумоподавление там не такое эффективное. Но это просто еще раз подтверждает тезис о том, что сильное шумоподавление повреждает сам речевой сигнал. В остальном результаты аналогичны результатам первого рассмотренного сигнала.

Можем сделать вывод, что модифицированный метод спектрального вычитания значительно эффективнее базового очищает шумы. Базовый метод не способен подавить достаточно шума при невысоком пороговом параметре детекции речи. При этом, если этот параметр будет значительно выше, мы можем потерять важную часть речевого сигнала. Также при очистке этим методом возникает много музыкальных шумов, которые значительно ухудшают качество сигнала. Модифицированный метод при подборе подходящих параметров эффективно справляется с задачей подавления шума, но нужно быть осторожным при выборе параметров, чтобы не ухудшить сам речевой сигнал.

3.3 Результаты шумоподавления при помощи вейвлет-преобразования

При анализе будем использовать те же данные, что и в случае со спектральным вычитанием.

Начнем рассмотрение с сигнала (night_forest.wav). На этой записи человек говорит на фоне довольно громкого шума ночного леса (сверчки и тому подобные звуки). При шумоподавлении, как раньше уже упоминалось, использовалась декомпозиция сигнала до 2-го уровня, в качестве материнского вейвлета – D4 из семейства вейвлетов Добеши и мягкая пороговая функция.

В результате шумоподавления, которое было осуществлено при помощи данного модуля программы, были получены результаты, представленные на рисунке 3.11.

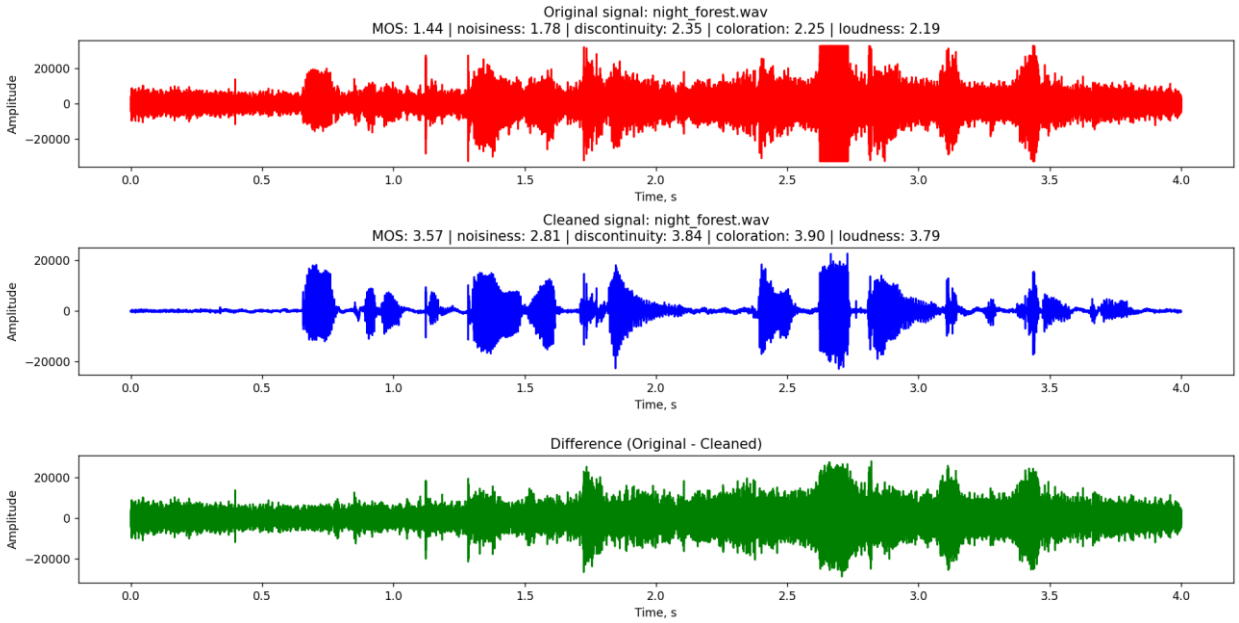

Рисунок 3.11 – Результат выполнения второго модуля программы для night_forest.wav

На графиках невооруженным глазом видны значительные изменения по сравнением с исходным зашумленным сигналом. Обращу внимание, что разностный график имеет амплитуду сопоставимую с результирующим, что говорит об очень сильном переборе в громкости у исходного сигнала. Это же подтверждает метрика loudness (так как громкость стала тише, что более нормально в данном случае, оценка выше). Да и все метрики в общем показывают значительное улучшение результата по сравнением с исходным сигналом. Рассмотрим спектрограммы исходного и обработанного сигналов, с которыми можно ознакомиться на рисунках 3.12 и 3.13 соответственно:

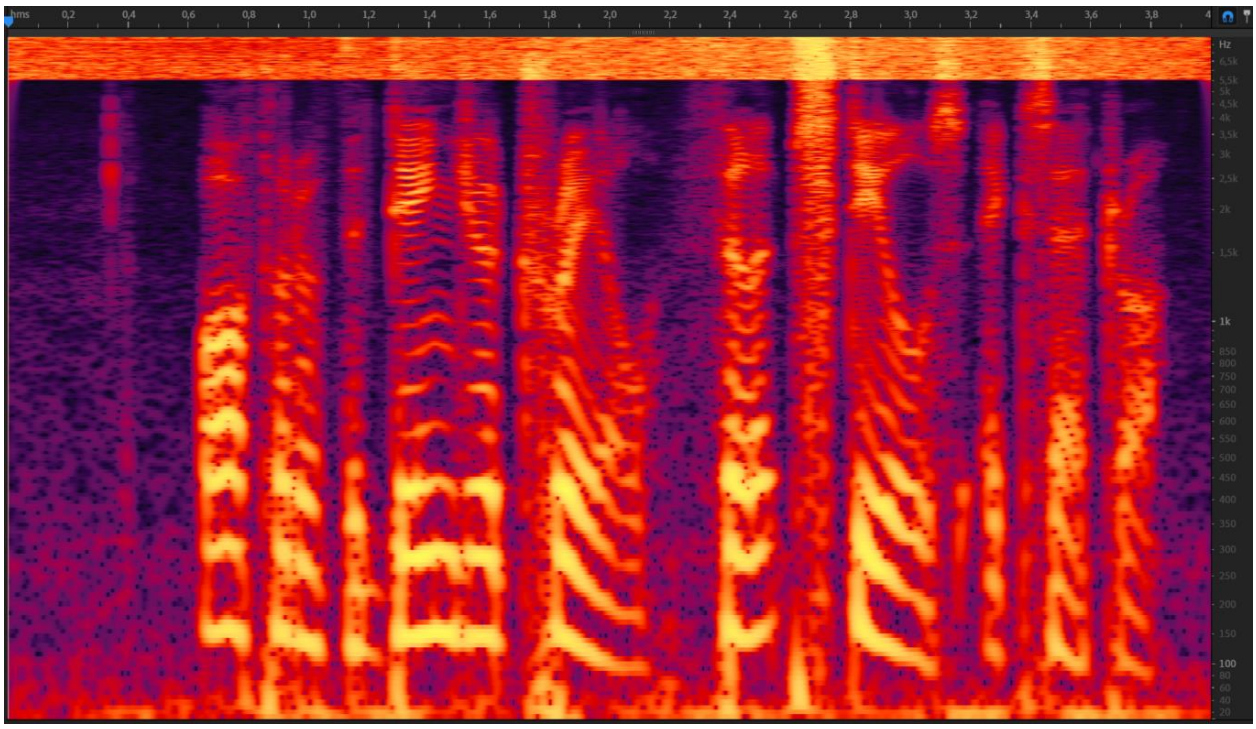

Рисунок 3.12 – Спектрограмма night_forest.wav до обработки

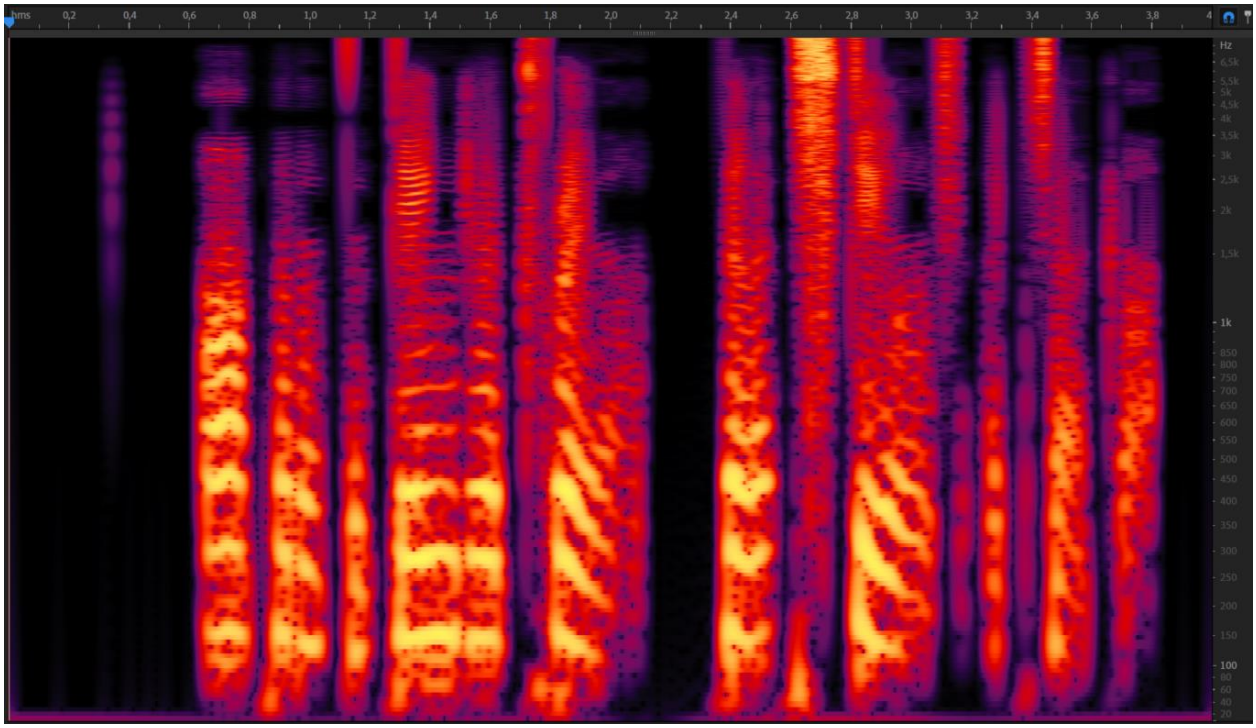

Рисунок 3.13 – Спектрограмма night_forest.wav после порогового шумоподавления

Во-первых, видно, что программа успешно очистила исходный сигнал от шума. Если анализировать сигнал на слух, шумы действительно были убраны при этом разборчивость человеческой речи была сохранена и даже улучшена.

Стоит обратить внимание на высокочастотный мощный сигнал вверху гистограммы исходного сигнала, который был полностью очищен в результате работы программы. Для сравнения посмотрим на рисунок 3.14 – гистограмму того же сигнала обработанного алгоритмом спектрального вычитания из прошлого пункта главы.

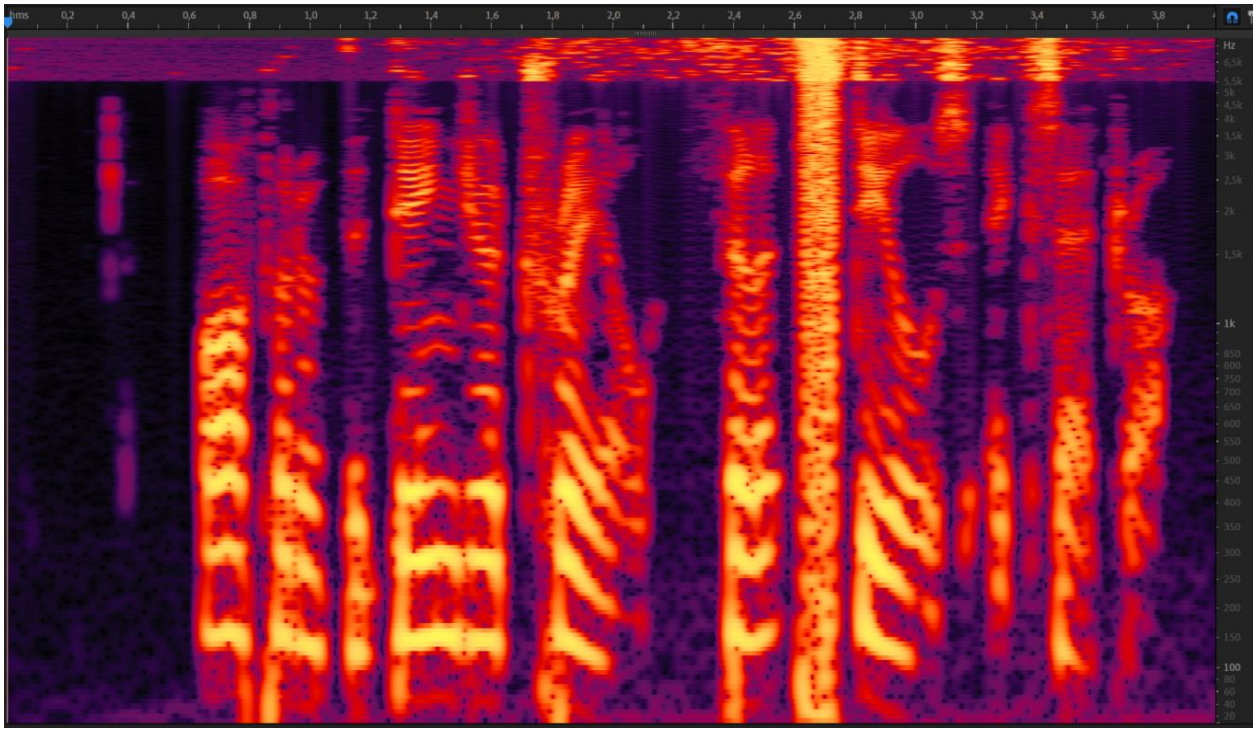

Рисунок 3.14 – Спектрограмма night_forest.wav после спектрального вычитания

Помимо в целом недостаточно хорошо очищенного сигнала, при его прослушивании можно услышать остатки фоновых звуков и некоторые звоны – музыкальные шумы, которые возникают при попытке еще сильнее очистить сигнал. Но самое главное – это не очищенный высокочастотный шум, на котором раньше мы уже акцентировали внимание. Вследствие не очень эффективной работы оконного преобразования Фурье с локальными особенностями спектра, такой алгоритм не смог очистить подобный сигнал.

Рассмотрим еще один пример. На это раз возьмем сигнал (birds.wav) с нестационарными шумами на фоне: пение птиц. При применении к такому сигналу спектрального вычитания для очистки от шума, пение птиц убрать не удастся, так как алгоритм не успевает адаптироваться к постоянно изменяемому шуму. Можем убедиться в этом. Результат выполнения первого модуля с этим сигналом на рисунке 3.15.

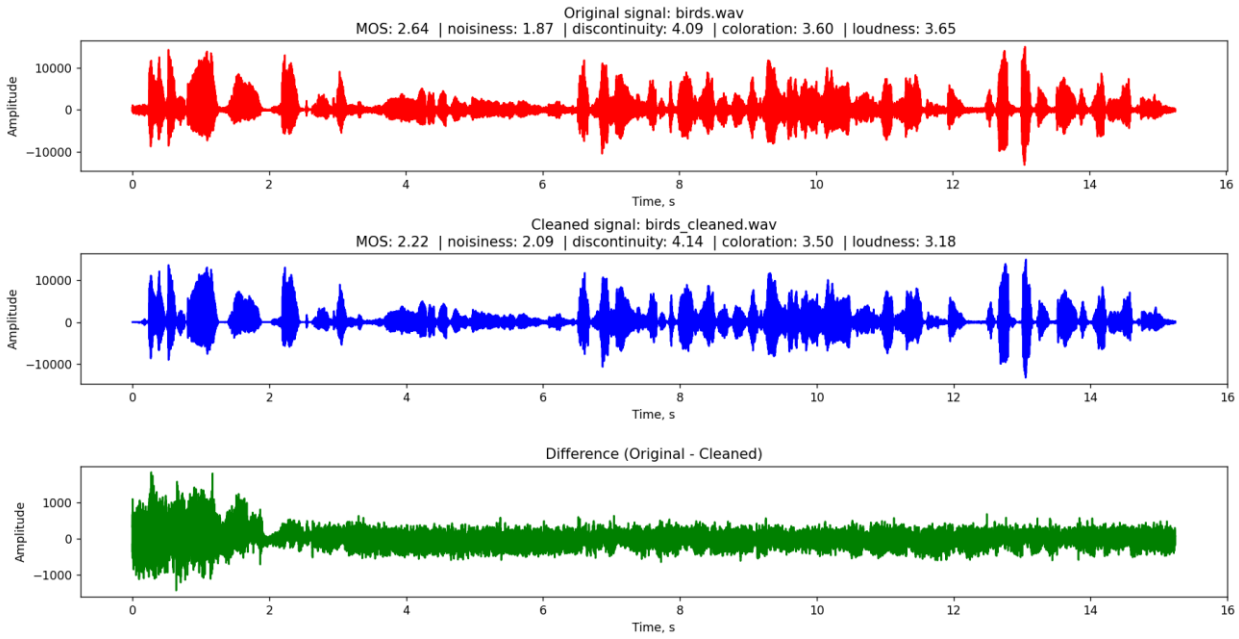

Рисунок 3.15 – Результат выполнения первого модуля для birds.wav

Декомпозицию снова проводим только до второго уровня, так как большая приводит к повреждению речевого сигнала. И рассмотрим тот же сигнал после обработки вторым модулем на рисунке 3.16.

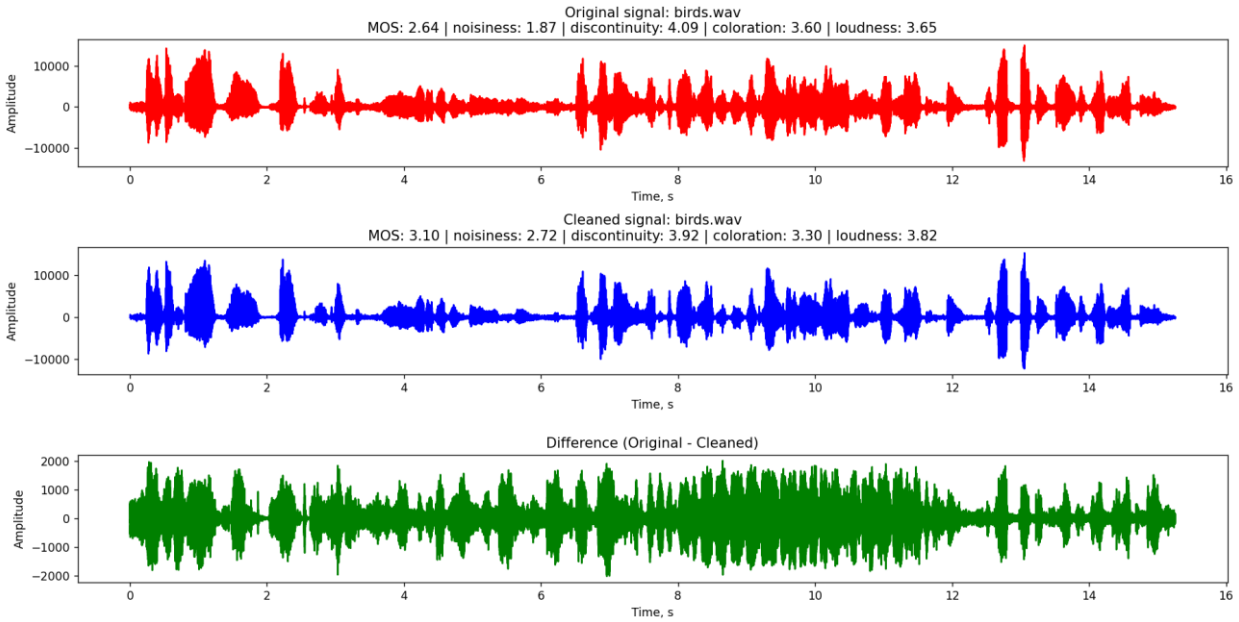

Рисунок 3.16 – Результат выполнения второго модуля для birds.wav

Видим огромную разницу в результатах. Чтобы еще лучше рассмотреть разницу, посмотрим на спектрограммы исходного сигнала и двух спектрограмм с шумоподавлением (спектральное вычитание и пороговое). Они представлены на рисунках 3.17 – 3.19.

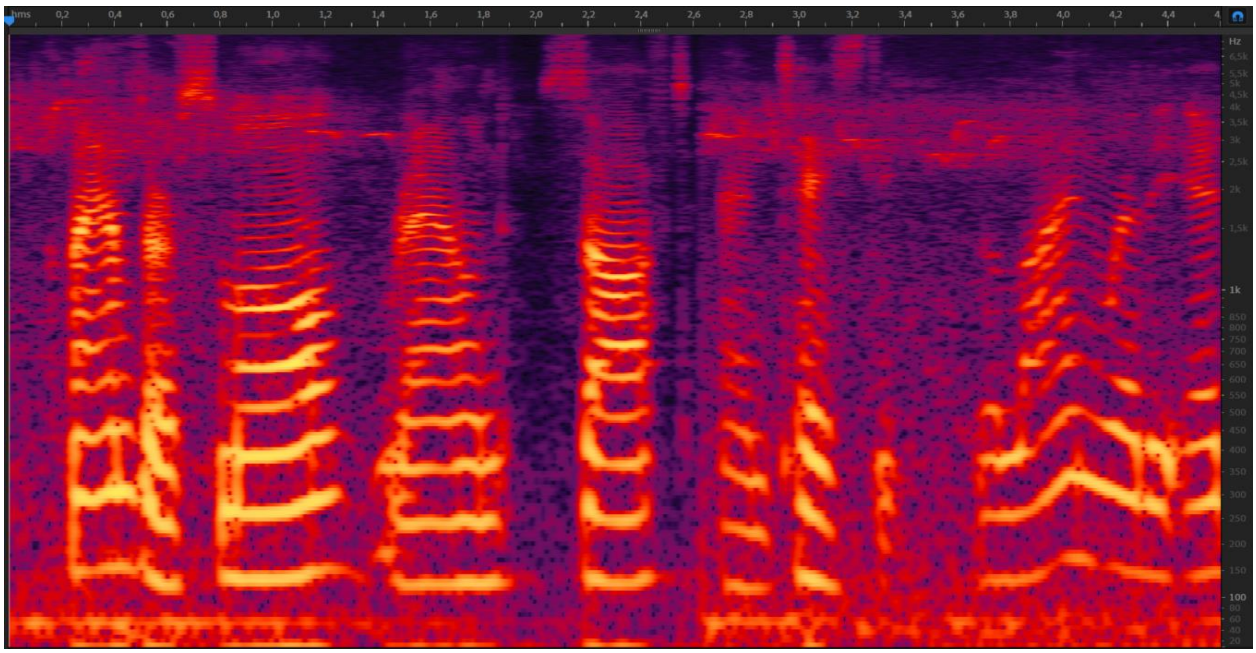

Рисунок 3.17 – Спектрограмма birds.wav до обработки

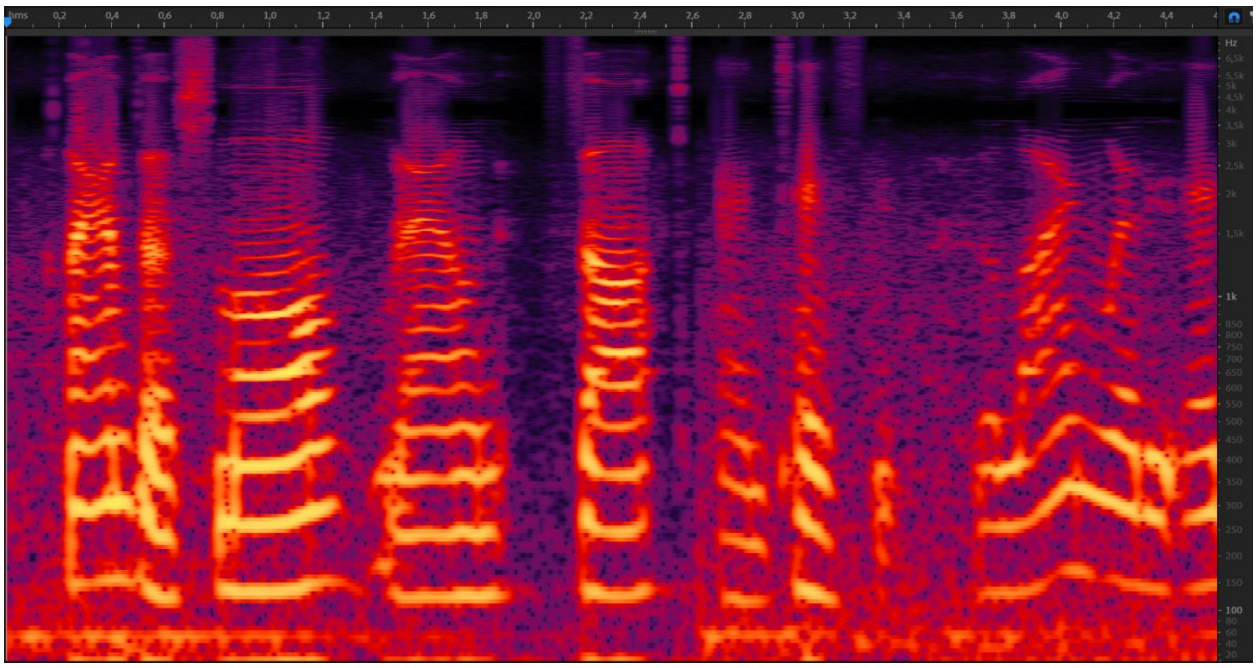

Рисунок 3.18 – Спектрограмма birds.wav после спектрального вычитания

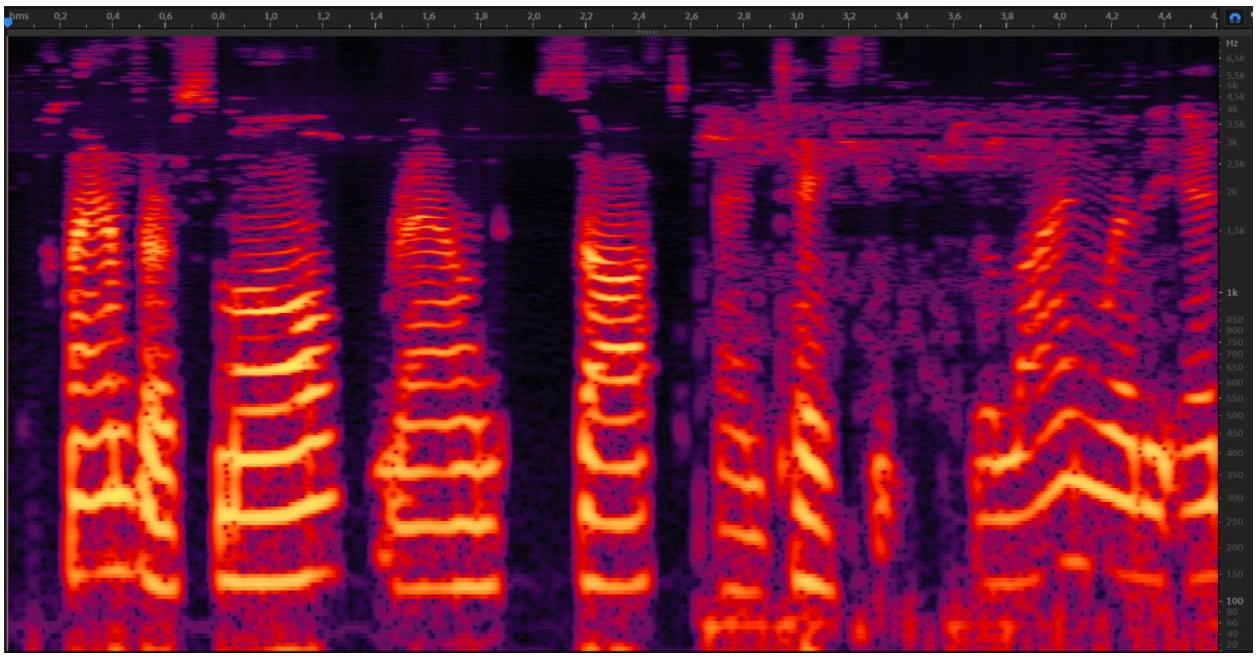

Рисунок 3.19 – Спектрограмма birds.wav после порогового шумоподавления

Видно, что спектральное вычитание убрало значительно больше шума, но при этом именно пение птиц все еще остается (бобовидные красные пятна на частоте 4k Гц). В это же время пороговое шумоподавление локально очистило именно зону со звуками птиц, убрав мешающий для восприятия речи шум.

Таким образом, благодаря свойствам локальности метод подавления шума, основанный на вейвлет преобразовании, имеет ряд преимуществ перед альтернативными методами, разработанными на базе преобразования Фурье. Основное из таких преимуществ – возможность подавлять нестационарный шум. При этом настройка такого преобразования может быть сложнее из-за большого количество альтернатив при выборе параметров подавления. Так же сложнее добиться полного сохранения изначального качества речевого сигнала, из-за чего при подавлении шума в стационарном сигнале предпочтительнее выбрать метод спектрального вычитания.

3.4 Результаты передискретизации речевых сигналов

Для анализа этого модуля программы будем рассматривать получаемые графики с метриками, а также посмотрим на данные о частоте дискретизации и размере файлов до и после выполнения программы (произведя децимацию, мы уменьшаем размер аудиофайла, а при интерполяции – увеличиваем). Для этого снова будем использовать AdobeAudition.

Для начала рассмотрим работу модуля программы на примере речевого сигнала, который мы очистили от шумов в одном из прошлых пунктов главы: roosevelt_speech_cleaned.wav. Изначально этот сигнал имеет частоту дискретизации 44,1 кГц и весит 294,23 KB, что можно узнать из свойств, которые представлены на рисунке 3.20.

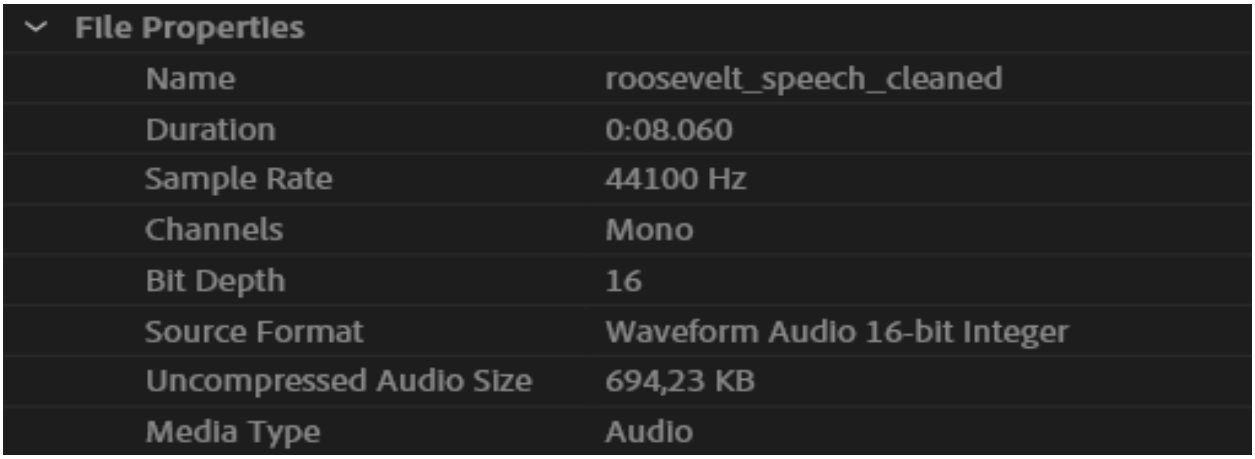

Рисунок 3.20 – Свойства файла roosevelt_speech_cleaned.wav

Теперь увеличим частоту дискретизации данного сигнала до 96 кГц при помощи модуля передискретизации нашей программы. В результате ее выполнения мы получим аудиофайл resampled_roosevelt_speech_cleaned.wav с измененной частотой дискретизации и амплитудные графики сигналов до и после передискретизации. Графики с метриками представлены на рисунке 3.21.

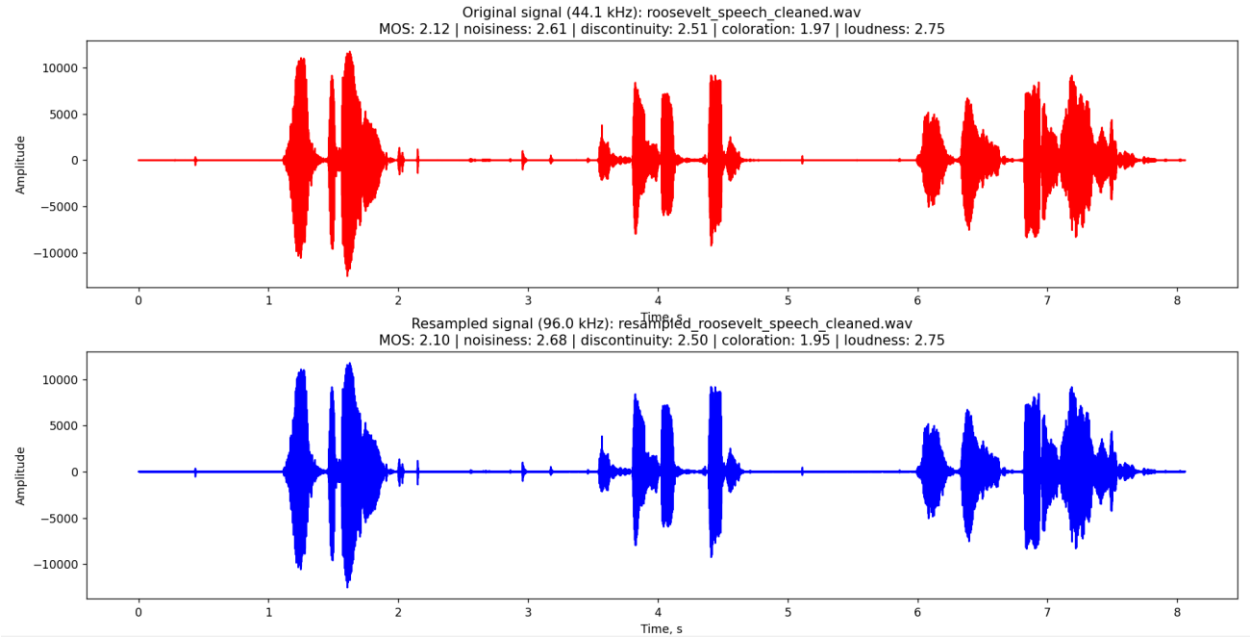

Рисунок 3.21 – Амплитудные графики до и после интерполяции аудиофайла roosevelt_speech_cleaned.wav

Можно заметить, что графики выглядят почти идентично, но некоторые отличия находят оценки метрик. Это можно связать с тем, что интерполяция так или иначе предполагает прогнозирование промежуточных значений сигнала, что уже не может гарантировать идентичность исходного и обработанного сигналов. Тем не менее отличия остаются незначительными, так что можно считать, что алгоритм сработал успешно. Теперь рассмотреть свойства полученного файла, которые продемонстрированы на рисунке 3.22.

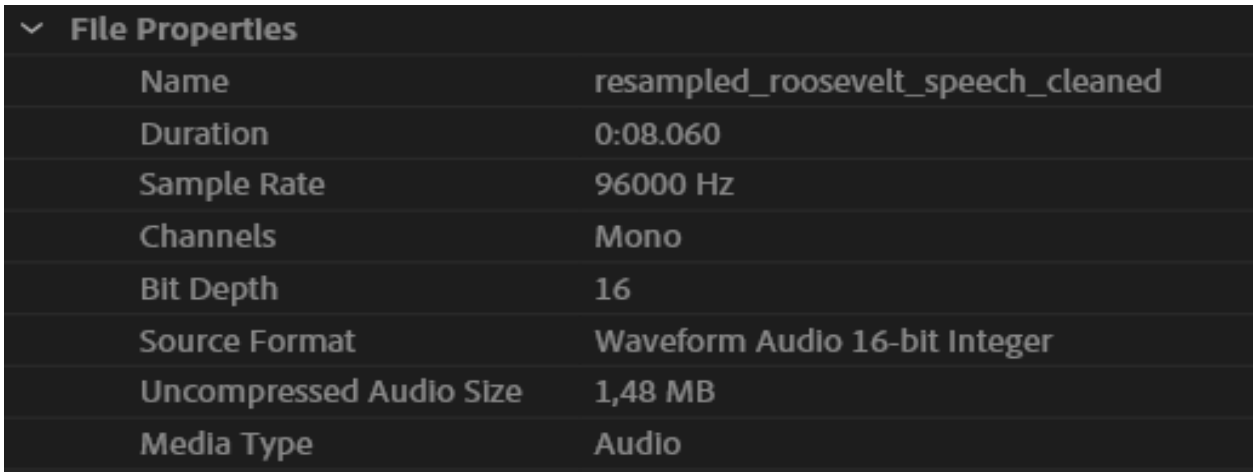

Рисунок 3.22 – Свойства файла resampled_roosevelt_speech_cleaned.wav

Видим, что у нового сигнала, как и ожидалось, частота дискретизации 96 кГц. При этом размер файла увеличился, причем $\frac{F_S'}{F_S} \approx \frac{1,48}{0,69}$ (здесь равенства нет, так как файл помимо самих данных о сигнале, также содержит информацию о типе файла, его размере и о некоторых других параметрах, которая также занимает некоторую часть от общего веса файла). Еще отметим, что длительность аудиофайла до и после изменения не поменялось, что также свидетельствует о корректной работе программы.

Теперь рассмотрим уменьшение частоты дискретизации на примере другого сигнала. Это также будет речевой сигнал, который был нами ранее очищен: churchill_report_cleaned.wav. Его изначальные свойства – частота дискретизации 44,1 кГц и размер 1000 KB – представлены на рисунке 3.23.

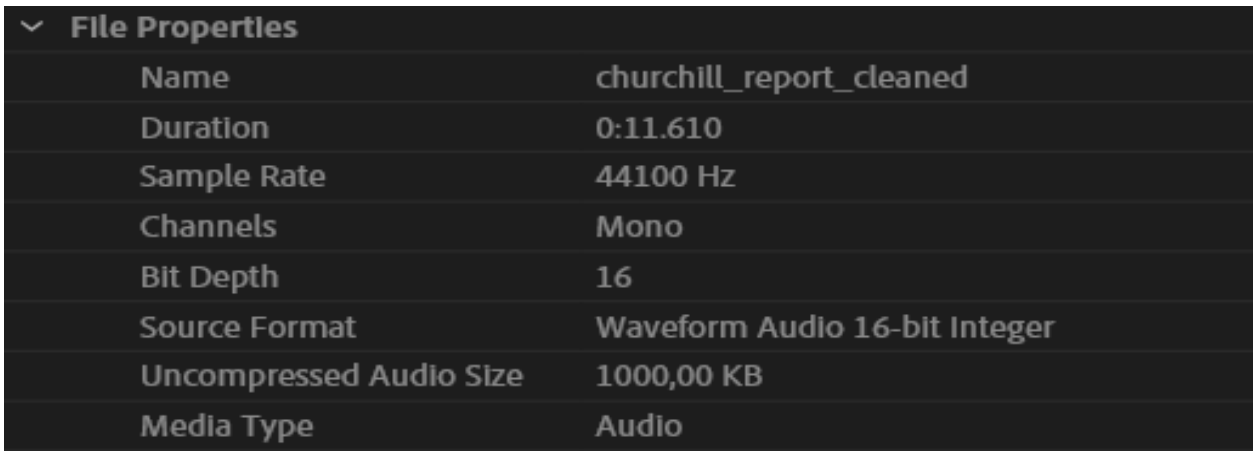

Рисунок 3.23 – Свойства файла churchill_report_cleaned.wav

Уменьшим частоту дискретизации аудиосигнала до 32 кГц. В результате выполнения программы получим графики, изображенные на рисунке 3.24.

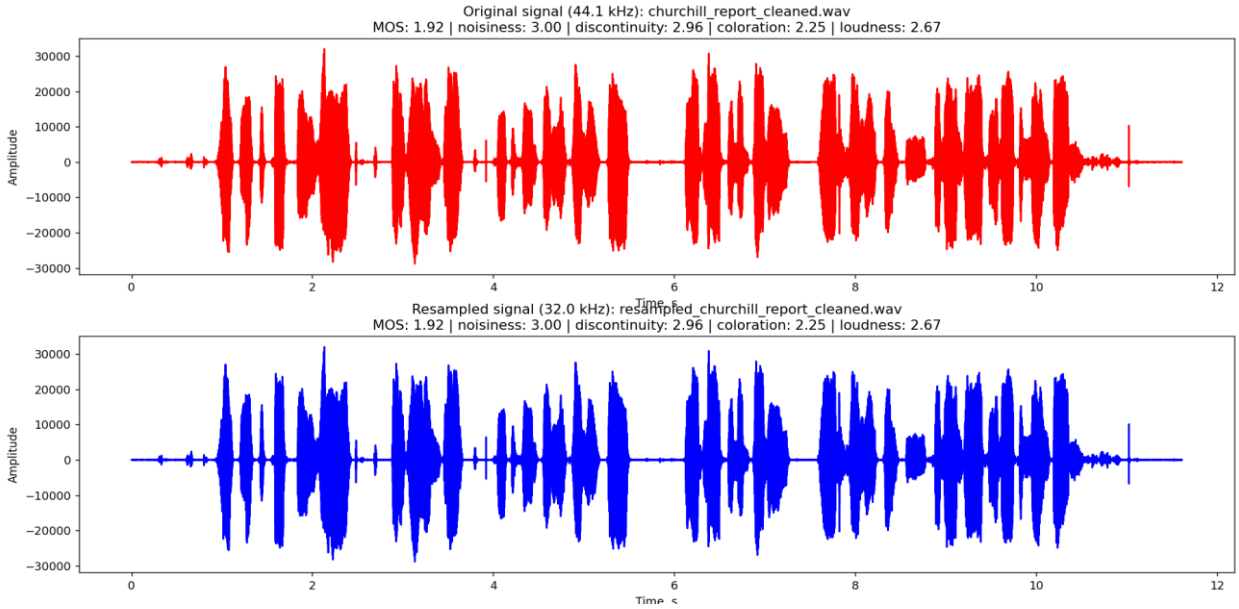

Рисунок 3.24 – Амплитудные графики до и после децимации аудиофайла churchill_report_cleaned.wav

На этот раз даже оценки не находят различий, что и ожидалось с учетом особенности алгоритма понижения частоты дискретизации. При прослушивании также невозможно найти различия в аудиофайлах. Рассмотрим свойства полученного файла, которые представлены на рисунке 3.25.

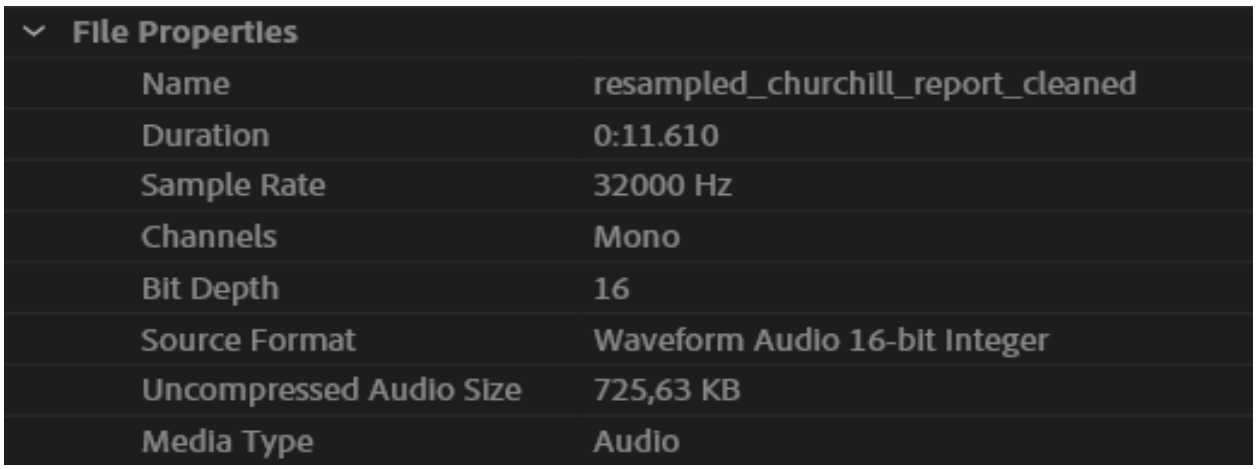

Рисунок 3.25 – Свойства файла resampled_churchill_report_cleaned.wav

Частота дискретизации файла уменьшилась до нужного значения. Размер файла уменьшился пропорционально коэффициенту масштабирования частоты дискретизации. А вот длина аудиофайла осталась прежней. Так что можем констатировать успешное уменьшение частоты аудиосигнала.

На основе двух вышерассмотренных примеров работы данного модуля программы, можем сделать вывод, что алгоритмы интерполяции и децимации с изменением размера входных данных успешно справляются с передискретизацией речевых сигналов.

ЗАКЛЮЧЕНИЕ

Дипломная работа направлена на разработку алгоритмов подавления шума в речевых сигналах, что позволяет повысить их качество путем уменьшения фонового шума, а также алгоритма передискретизации речевого сигнала, что позволяет повысить или понизить частоту дискретизации сигнала.

Разработанные алгоритмы шумоподавления могут активно использоваться в системах голосовой связи: мобильная телефония, видеоконференции и VoIP-сервисы (Zoom, Skype). Алгоритмы позволяют эффективно подавлять фоновые шумы, значительно улучшая разборчивость речи без искажения полезного сигнала. В системах звукозаписи (подкасты, аудиокниги, запись музыки) эти алгоритмы используются для постобработки материала, чтобы удалить возможные шумы при снятии звука микрофонами или акустические артефакты. Особенно важную роль шумоподавление играет в слуховых аппаратах и системах распознавания речи (голосовые помощники, call-центры), где от качества входного сигнала напрямую зависит точность его обработки.

Алгоритм передискретизации, который был разобран в данной работе, активно применяется при обеспечения совместимости аудиоустройств с разными техническими характеристиками. Он может использоваться при конвертации аудиофайлов между форматами (например, при преобразовании студийных записей 96 кГц в стандартный формат 44.1 кГц для CD), в системах цифровой телефонии для согласования частот дискретизации между оборудованием различных производителей, а также в мультимедийных процессорах для адаптации сигнала под конкретные условия воспроизведения.

Все вычислительные алгоритмы в данной работе активно используют интегральные преобразования, которые позволяют получить спектр сигнала и произвести его дальнейшую обработку в спектральной области. При этом один из алгоритмов шумоподавления построен на основе модифицированного метода спектральных вычитаний, другой производит более детальную очистку в некоторых случаях при помощи порогового шумоподавления сигнала. В случае же алгоритма передискретизации применялись методы интерполяции и децимации.

На основе разработанных алгоритмов была реализована программа на языке Python с отдельным программным модулем под каждый из алгоритмов: модуль шумоподавления при помощи преобразования Фурья, модуль шумоподавления при помощи вейвлет-преобразования и модуль передискретизации речевых сигналов.

В результате выполнения обоих программных модулей шумоподавления, написанных на основе соответствующих алгоритмов, мы получаем очищенный от шума речевой сигнал, который в дальнейшем может использоваться для простого прослушивания человеком или для дальнейшей обработки и анализа сигнала в ранее упомянутых сферах, например, в системах распознавания речи. Модуль передискретизации в результате выполнения дает на выход речевой сигнал с измененной частотой дискретизации, который в последствии может быть воспроизведен или обработан устройством или программой, работающей с сигналами этой частоты дискретизации.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Архипов, Е. Д. Методы оценивания качества передаваемой речевой информации. Объективные интрузивные методы / Е. Д. Архиров // Меридиан. – 2022. – Т. 63, № 1. – С. 161-165.
2. Ахмад, Х. М. Введение в цифровую обработку речевых сигналов : учебное пособие / Х. М. Ахмад, В. Ф. Жирков ; Владим. гос. ун-т. – Владимир : Изд-во Владим. гос. ун-та, 2007. – 192 с.
3. Бельхеева, Р. К. Преобразование Фурье в примерах и задачах : учебное пособие / Р. К. Бельхеева ; Новосиб. гос. ун-т. — Новосибирск : РИЦ НГУ, 2014. – 81 с.
4. Вадутов, О. С. Математические основы обработки сигналов. Практикум: учебное пособие / О. С. Вадутов; Томский политехнический университет. – 3-е изд., испр. и доп. – Томск : Изд-во Томского политехнического университета, 2014. – 102 с.
5. Вишняков, И. Э. Методы и алгоритмы шумоочистки звука в реальном времени / И. Э. Вишняков, М. М. Масягин, О. А. Одинцов, В. В. Слюсарь // Изв. Вузов. Электроника. – 2021. – Т. 26, № 2. – С. 184 – 194.
6. Внуков, Ю. Н. Моделирование износа инструмента по результатам вейвлет-преобразования звукового сигнала / Ю.Н. Внуков [и др.] // Искусственный интеллект. – 2007. – № 1. — С. 73-79.
7. Воробьев, Н. Н. Теория рядов. 4 изд. : учеб. для вузов / Н. Н. Воробьев – М. : Главная редакция физико-математической литературы, 1979. – 408 с.
8. Дремин, И. М. Вейвлеты и их использование / И. М. Дремин, О. В. Иванов // Успехи физических наук. – 2001. – Т. 171, № 5. – С. 465–501.
9. Мамедов, Т. Т. Применение вейвлет-преобразований для анализа и сжатия телеметрических данных вибрационных процессов / Т. Т. Мамедов [и др.] // Ракетно-космическое приборостроение и инф. системы. – 2019. – Т. 6, № 4. – С. 59 – 65.
10. Обидин, М. В. Очистка сигнала от шумов с использованием вейвлет преобразования и фильтра Калмана / М. В. Обидин, А. П. Серебровский // Информационные процессы. – 2013. – Т. 13, № 3. – С. 198-205.
11. Петровский, А. А. Цифровая обработка аудио- и видеоданных : учебное пособие / А. А. Петровский, М. И. Вашкевич, И. С. Азаров. — Минск : БГУИР, 2017. – 64 с.
12. Приоров, А. Л. Обработка сигналов на основе вейвлетпреобразования: методические указания / А. Л. Приоров, В. А. Волохов, И. В. Апальков; Яросл. гос. ун-т им. П. Г. Демидова. – Ярославль : ЯрГУ, 2011. – 44 с.

13. Рабинер, Л. Р. Цифровая обработка речевых сигналов : пер. с англ. / Л. Р. Рабинер, Р. В. Шафер ; под ред. М. В. Назарова и Ю. Н. Прохорова. – М. : Радио и связь, 1981. – 496 с.

14. Сергиенко, А. Б. Цифровая обработка сигналов : учеб. для вузов / А. Б. Сергиенко. – СПб. : Питер, 2003. – 604 с.

15. Столбов, М. Б. Основы анализа и обработки речевых сигналов : учебное пособие / М. Б. Столбов. – СПб. : НИУ ИТМО, 2021. – 101 с.

16. Чеб, Е. С. Интегральные преобразования: методические указания и задания. В 2 ч. Ч. 2 : учебное пособие / Е. С. Чеб. – Минск : БГУ, 2022. – 60 с.

17. Arar, S. FIR Filter Design by Windowing: Concepts and the Rectangular Window / S. Arar // All About Circuits, articles for electrical engineers. – URL: https://www.allaboutcircuits.com/technical-articles/finite-impulse-response-filter-design-by-windowing-part-i-concepts-and-rect (date of access: 20.05.2025).

18. He, C. A New Wavelet Threshold Determination Method Considering Interscale Correlation in Signal Denoising / C. He // Mathematical Problems in Engineering, Wiley. – URL: https://onlinelibrary.wiley.com/doi/10.1155/2015/280251 (date of access: 20.05.2025).

19. Lyons, R. How to Interpolate in the Time-Domain by Zero-Padding in the Frequency Domain / R. Lyons // DSPGuru, digital signal processing articles. – URL: https://dspguru.com/dsp/howtos/how-to-interpolate-in-time-domain-by-zero-padding-in-frequency-domain (date of access: 20.05.2025).

20. Mittag, G. NISQA: A Deep CNN-Self-Attention Model for Multidimensional Speech Quality Prediction with Crowdsourced Datasets / G. Mittag, Babak Naderi, Assmaa Chehadi, Sebastian Möller // Cornell University. ArXiv, open-access archive for scholarly articles. – URL: https://arxiv.org/abs/2104.09494 (date of access: 20.05.2025).

21. ODS AI Ru. Иван Бескровный | NISQA-s: оценка качества для потокового аудио // YouTube. – URL: https://www.youtube.com/watch?v=AtGyrKGxC4Y (date of access: 20.05.2025).

22. Smith, J. Mathematics of the Discrete Fourier Transform (DFT), with Audio Applications, Second Edition / J. Smith // Center for Computer Research in Music and Acoustics, Stanford University. – URL: https://ccrma.stanford.edu/~jos/mdft/Spectrograms.html (date of access: 20.05.2025).

23. Tan, L. Multirate DSP, part 1: Upsampling and downsampling / L. Tan // Electronic Engineering Times. – URL: https://www.eetimes.com/multirate-dsp-

part-1-upsampling-and-downsampling/?_ga=page_number%3D3 (date of access: 20.05.2025).

24. Valimaki, V. Giant FFTs for Sample Rate Conversion / V. Valimaki, S. Bilbao // Journal of the Audio Engineering Society. – 2023. – Vol. 71, № 3. – P. 88–99.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД МОДУЛЯ СПЕКТРАЛЬНОГО ВЫЧИТАНИЯ

```
import math
import numpy as np
import matplotlib
import os
from scipy.io.wavfile import read, write
import matplotlib.pyplot as plt
from nisqa.NISQA_model import nisqaModel

matplotlib.use('TkAgg')


class AudioFourierDenoise:
    def __init__(self, input_file):
        self.__input_file = input_file
        self.frame_duration = 0.02  # Размер фрейма
        self.threshold_snr = 1.1  # Пороговое значение отношения сигнал-
шум
        self.overlap_rate = 50  # Процент перекрытия фреймов
        self.beta = 0.001
        self.gamma = 0.9
        self.alpha = 10

    @staticmethod
    def __next_pow2(numb):
        k = 1
        power = 0
        while k < numb:
            k *= 2
            power += 1
        return power

    @staticmethod
    def __evaluate_with_nisqa(filename):
        args = {
```

```
        'mode': 'predict_file',
        'pretrained_model': 'nisqa.tar',
        'deg': filename,
        'output_dir': None,
        'bs': 1,
        'ms_channel': 1
    }
    nisqa = nisqaModel(args)
    return nisqa.predict()

@staticmethod
def __write_metrics(metrics, title):
    if metrics is not None:
        title += (f"\nMOS: {metrics['mos_pred'].values[0]:.2f} | "
                  f"noisiness: {metrics['noi_pred'].values[0]:.2f} | "
                  f"discontinuity: {metrics['dis_pred'].values[0]:.2f} | "
                  f"coloration: {metrics['col_pred'].values[0]:.2f} | "
                  f"loudness: {metrics['loud_pred'].values[0]:.2f}")

def __plot_results(self, original_file, cleaned_file, output_file):
    sample_rate, original_signal = read(original_file)
    sample_rate2, result_signal = read(cleaned_file)
    original_metrics = self.__evaluate_with_nisqa(original_file)
    cleaned_metrics = self.__evaluate_with_nisqa(cleaned_file)
    time_original = np.arange(len(original_signal)) / sample_rate
    time_result = np.arange(len(result_signal)) / sample_rate2
    plt.figure(figsize=(12, 8))

    # Оригинальный сигнал
    plt.subplot(3, 1, 1)
    plt.plot(time_original, original_signal, 'r')
    title = f"Original signal: {os.path.basename(self.__input_file)}"
    self.__write_metrics(original_metrics, title)
    plt.title(title)
    plt.xlabel("Time, s")
    plt.ylabel("Amplitude")

    # Очищенный сигнал
    plt.subplot(3, 1, 2)
    plt.plot(time_result, result_signal, 'b')
```

```
            title = f"Cleaned signal: {os.path.basename(output_file)}"
            self.__write_metrics(cleaned_metrics, title)
            plt.title(title)
            plt.xlabel("Time, s")
            plt.ylabel("Amplitude")

            # Разница сигналов
            plt.subplot(3, 1, 3)
            plt.plot(time_original[:len(result_signal)],
original_signal[:len(result_signal)] - result_signal, 'g')
            plt.title("Difference (Original - Cleaned)")
            plt.xlabel("Time, s")
            plt.ylabel("Amplitude")

            plt.tight_layout()
            plt.show()

        def __denoise_frame(self, normed_signal, sample_rate):
            frame_samples = math.floor(self.frame_duration * sample_rate)
            if frame_samples % 2 == 1:
                frame_samples += 1

            overlapped_samples = math.floor(frame_samples * self.overlap_rate /
100)
            free_samples = frame_samples - overlapped_samples

            window = np.hamming(frame_samples)
            window_norm = free_samples / sum(window)

            frames_count = math.floor(len(normed_signal) / free_samples) - 1
            frame_fft_len = 2 * pow(2, self.__next_pow2(frame_samples))

            # Инициализация спектра шума
            noise_spect = np.zeros(frame_fft_len)
            for n in range(0, 5 * frame_samples, frame_samples):
                noise_spect += abs(np.fft.fft(window * normed_signal[n: n +
frame_samples], frame_fft_len))
            noise_spect = noise_spect / 5
            noise_spect_power = pow(noise_spect, 2)
```

```
            # Обработка сигнала
            signal_before_overlapping = np.zeros(overlapped_samples)
            result_signal_norm = np.zeros(frames_count * free_samples)

            for n in range(0, frames_count * free_samples, free_samples):
                frame_spect = np.fft.fft(window * normed_signal[n: n +
frame_samples], frame_fft_len)
                frame_spect_power = pow(abs(frame_spect), 2)
                frame_spect_phase = np.angle(frame_spect)

                # Спектральное вычитание
                cleaned_frame_spect = frame_spect_power - self.alpha *
noise_spect_power
                zeros_ind = np.asarray(cleaned_frame_spect - self.beta *
noise_spect_power < 0).nonzero()
                if zeros_ind:
                    cleaned_frame_spect[zeros_ind] = self.beta *
noise_spect_power[zeros_ind]

                # Детектор речевой активности
                frame_snr = 10 * math.log(sum(frame_spect_power) /
sum(noise_spect_power), 10)
                if frame_snr < self.threshold_snr:
                    noise_spect_power = self.gamma * noise_spect_power + (1 -
self.gamma) * frame_spect_power

                # Обратное ДПФ
                cleaned_frame = np.real(np.fft.ifft(np.sqrt(cleaned_frame_spect) *
(np.exp(1j * frame_spect_phase))))
                result_signal_norm[n: n + overlapped_samples] = \
                    signal_before_overlapping + cleaned_frame[0:
overlapped_samples]
                signal_before_overlapping = cleaned_frame[overlapped_samples:
frame_samples]

            return result_signal_norm * window_norm

        def denoise(self, output_file):
            # Чтение и нормализация сигнала
            sample_rate, original_signal = read(self.__input_file)
```

```
            signal_norm = max(abs(original_signal.max()),
abs(original_signal.min()))
            normed_signal = original_signal.astype(np.double) / signal_norm

            original_temp = 'original_temp.wav'
            cleaned_temp = 'cleaned_temp.wav'
            write(original_temp, sample_rate, original_signal.astype(np.int16))

            # Обработка сигнала
            result_signal_norm = self.__denoise_frame(normed_signal,
sample_rate)
            result_signal = result_signal_norm * signal_norm

            write(cleaned_temp, sample_rate, result_signal.astype(np.int16))
            write(output_file, sample_rate, result_signal.astype(np.int16))
            self._plot_results(original_temp, cleaned_temp, output_file)

            os.remove(original_temp)
            os.remove(cleaned_temp)


        if __name__ == "__main__":
            denoiser = AudioFourierDenoise("input_wav/birds.wav")
            denoiser.denoise("output_wav/birds_cleaned.wav")
```

ПРИЛОЖЕНИЕ Б

ИСХОДНЫЙ КОД МОДУЛЯ ПОРОГОВОГО ШУМОПОДАВЛЕНИЯ

```
import numpy as np
import pywt
import soundfile
import matplotlib
import os
import matplotlib.pyplot as plt
from scipy.io.wavfile import read, write
from nisqa.NISQA_model import nisqaModel

matplotlib.use('TkAgg')


class AudioWaveletDenoise:
    def __init__(self, input_file):
        self.__inputFile = input_file

    @staticmethod
    def __mad(arr):
        return np.median(np.abs(arr)) / 0.6745

    @staticmethod
    def __evaluate_metrics_with_nisqa(filename):
        args = {
            'mode': 'predict_file',
            'pretrained_model': 'nisqa.tar',
            'deg': filename,
            'output_dir': None,
            'bs': 1,
            'ms_channel': 1
        }
        nisqa = nisqaModel(args)
        return nisqa.predict()

    @staticmethod
```

```
        def __write_metrics(metrics, title):
            if metrics is not None:
                title += (f"\nMOS: {metrics['mos_pred'].values[0]:.2f} | "
                          f"noisiness: {metrics['noi_pred'].values[0]:.2f} | "
                          f"discontinuity: {metrics['dis_pred'].values[0]:.2f} | "
                          f"coloration: {metrics['col_pred'].values[0]:.2f} | "
                          f"loudness: {metrics['loud_pred'].values[0]:.2f}")

        def __plot_results(self, original_file, cleaned_file, output_file):
            sample_rate, original_signal = read(original_file)
            sample_rate2, result_signal = read(cleaned_file)
            original_metrics = self.__evaluate_metrics_with_nisqa(original_file)
            cleaned_metrics = self.__evaluate_metrics_with_nisqa(cleaned_file)
            time_original = np.arange(len(original_signal)) / sample_rate
            time_result = np.arange(len(result_signal)) / sample_rate2
            plt.figure(figsize=(12, 8))

            # Оригинальный сигнал
            plt.subplot(3, 1, 1)
            plt.plot(time_original, original_signal, 'r')
            title = f"Original signal: {os.path.basename(self.__inputFile)}"
            self.__write_metrics(original_metrics, title)
            plt.title(title)
            plt.xlabel("Time, s")
            plt.ylabel("Amplitude")

            # Очищенный сигнал
            plt.subplot(3, 1, 2)
            plt.plot(time_result, result_signal, 'b')
            title = f"Cleaned signal: {os.path.basename(output_file)}"
            self.__write_metrics(cleaned_metrics, title)
            plt.title(title)
            plt.xlabel("Time, s")
            plt.ylabel("Amplitude")

            # Разница сигналов
            plt.subplot(3, 1, 3)
            plt.plot(time_original[:len(result_signal)],
original_signal[:len(result_signal)] - result_signal[:], 'g')
            plt.title("Difference (Original - Cleaned)")
```

```
            plt.xlabel("Time, s")
            plt.ylabel("Amplitude")

            plt.tight_layout()
            plt.show()

        def denoise(self, output_file):
            info = soundfile.info(self.__inputFile)
            rate = info.samplerate
            original_temp = 'original_temp.wav'
            cleaned_temp = 'cleaned_temp.wav'

            with soundfile.SoundFile(cleaned_temp, "w", samplerate=rate,
channels=info.channels) as of:
                data, samplerate = soundfile.read(self.__inputFile)
                soundfile.write(original_temp, data, samplerate)
                coefficients = pywt.wavedec(data, 'db4', level=2, mode='per')
                for i in range(1, len(coefficients)):
                    level = coefficients[i]
                    sigma = self.__mad(level)
                    thresh = sigma * np.sqrt(2 * np.log(len(data)))
                    coefficients[i] = pywt.threshold(coefficients[i], value=thresh,
mode='soft')
                clean = pywt.waverec(coefficients, 'db4', mode='per')
                of.write(clean)

            _, result_signal = read(cleaned_temp)
            write(output_file, rate, result_signal)
            self.__plot_results(original_temp, cleaned_temp, output_file)

            os.remove(original_temp)
            os.remove(cleaned_temp)


    if __name__ == "__main__":
        audioDenoiser = AudioWaveletDenoise("input_wav/night_forest.wav")
        audioDenoiser.denoise("output_wav/night_forest.wav")
```

ПРИЛОЖЕНИЕ В

ИСХОДНЫЙ КОД МОДУЛЯ ПЕРЕДИСКРЕТИЗАЦИИ

```
import math
import numpy as np
import matplotlib
import os
import matplotlib.pyplot as plt
from nisqa.NISQA_model import nisqaModel
from scipy.io.wavfile import read, write

matplotlib.use('TkAgg')


class AudioResampler:
    def __init__(self, input_file):
        self.__inputFile = input_file

    @staticmethod
    def __evaluate_metrics_with_nisqa(filename):
        args = {
            'mode': 'predict_file',
            'pretrained_model': 'nisqa.tar',
            'deg': filename,
            'output_dir': None,
            'bs': 1,
            'ms_channel': 1
        }
        nisqa = nisqaModel(args)
        return nisqa.predict()

    def __plot_results(self, original_file, resampled_file, output_file, fs,
new_fs):
        sample_rate, original_signal = read(original_file)
        sample_rate2, result_signal = read(resampled_file)
        original_metrics = self.__evaluate_metrics_with_nisqa(original_file)
```

```
            resampled_metrics                                                    =
self.__evaluate_metrics_with_nisqa(resampled_file)
            time_original = np.arange(len(original_signal)) / fs
            time_result = np.arange(len(result_signal)) / new_fs
            plt.figure(figsize=(12, 8))

            # Оригинальный сигнал
            plt.subplot(2, 1, 1)
            plt.plot(time_original, original_signal, 'r')
            title = f"Original signal ({(fs / 1000):.1f} kHz): {original_file}"
            if original_metrics is not None:
                title += (f"\nMOS: {original_metrics['mos_pred'].values[0]:.2f} | "
                          f"noisiness: {original_metrics['noi_pred'].values[0]:.2f} | "
                          f"discontinuity: {original_metrics['dis_pred'].values[0]:.2f} | "
                          f"coloration: {original_metrics['col_pred'].values[0]:.2f} | "
                          f"loudness: {original_metrics['loud_pred'].values[0]:.2f}")
            plt.title(title)
            plt.xlabel("Time, s")
            plt.ylabel("Amplitude")

            # Ресемплированный сигнал
            plt.subplot(2, 1, 2)
            plt.plot(time_result, result_signal, 'b')
            title = f"Resampled signal ({(new_fs / 1000):.1f} kHz): {output_file}"
            if resampled_metrics is not None:
                title += (f"\nMOS: {original_metrics['mos_pred'].values[0]:.2f} | "
                          f"noisiness: {original_metrics['noi_pred'].values[0]:.2f} | "
                          f"discontinuity: {original_metrics['dis_pred'].values[0]:.2f} | "
                          f"coloration: {original_metrics['col_pred'].values[0]:.2f} | "
                          f"loudness: {original_metrics['loud_pred'].values[0]:.2f}")
            plt.title(title)
            plt.xlabel("Time, s")
            plt.ylabel("Amplitude")

            plt.tight_layout()
            plt.show()

        def __resampling(self, signal, fs, new_fs, new_n):
            n = len(signal)
            spect = np.fft.fft(signal)
```

```
            if new_fs > fs:
                new_spect = np.concatenate((spect[: n // 2], [spect[n // 2] / 2],
                                            np.tile([0 + 0j], new_n - n - 1),
                                            [spect[n // 2] / 2], spect[n // 2 + 1:]))
            else:
                new_spect = np.concatenate((spect[: new_n // 2], [0 + 0j],
                                            spect[n - new_n // 2 + 1:]))

            new_spect_tapering = new_spect
            result_signal = np.real(np.fft.ifft(new_spect_tapering))
            result_signal = result_signal * new_fs / fs
            return result_signal

        def resample(self, output_file, new_fs):
            fs, original_signal = read(self.__inputFile)

            # Создаем временные файлы для оценки
            original_temp = 'original_temp.wav'
            resampled_temp = 'resampled_temp.wav'

            # Количество каналов сигнала
            channels = len(original_signal.shape)
            if channels == 2:
                original_signal = [original_signal[i][0] for i in
range(len(original_signal))]

            n_in = len(original_signal)
            gcd = np.gcd(fs, new_fs)
            p = new_fs // gcd
            q = fs // gcd

            m = 2 * math.ceil(n_in / 2 / q)
            n = q * m
            original_signal_pad = np.concatenate((original_signal, np.tile(0, n -
n_in)))
            new_n = p * m

            resampled_signal = self.__resampling(original_signal_pad, fs, new_fs,
new_n)
```

```
            result_signal = resampled_signal[: math.ceil(n_in * new_fs / fs)]

            # Сохраняем временные файлы для оценки
            write(original_temp, fs, np.array(original_signal).astype(np.int16))
            write(resampled_temp, new_fs, result_signal.astype(np.int16))

            # Сохранение итогового файла
            if channels == 2:
                result_signal = np.array([[result_signal[i], result_signal[i]] for i in
range(len(result_signal))])
                write(output_file, new_fs, result_signal.astype((np.int16, np.int16)))
            else:
                write(output_file, new_fs, result_signal.astype(np.int16))

            self.__plot_results(original_temp, resampled_temp, output_file, fs,
new_fs)

            # Удаление временных файлов
            os.remove(original_temp)
            os.remove(resampled_temp)


        if __name__ == "__main__":
            resampler = AudioResampler("input_wav/churchill_report_cleaned.wav")
            resampler.resample("output_wav/churchill_report_cleaned.wav", 32000)
```