

ГУАП

КАФЕДРА № 42

ОТЧЕТ  
ЗАЩИЩЕН С ОЦЕНКОЙ \_\_\_\_\_  
ПРЕПОДАВАТЕЛЬ

старший преподаватель \_\_\_\_\_  
должность, уч. степень, звание \_\_\_\_\_  
подпись, дата \_\_\_\_\_  
Т.А. Сутина  
ициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №7  
РАБОТА С ФАЙЛАМИ ФОРМАТА MP3

по курсу: Техника аудиовизуальных средств информации

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. № \_\_\_\_\_ 4329 \_\_\_\_\_  
подпись, дата \_\_\_\_\_  
Д.С. Шаповалова  
ициалы, фамилия

Санкт-Петербург 2025

## **1. Цель работы:**

Ознакомиться с типизированными файлами на примере формата MP3. Отработать навык работы с методами обработки данных файла.

## **2. Задание:**

Вариант = 4

Загрузить небольшой MP3 файл.

1. Считать заголовок фрейма MP3 файла для постоянного битрейта или несколько при переменном битрейте. Вывести на экран параметры и их значения. Вычислить длительность файла.
2. Выполнить расчет амплитуды и битовой глубины. Сформировать задание в соответствии с таблицей.

Вариант	1	2	3	4	5	6
Частота дискретизации	↑2	↑2	↑2	↓2	↓2	↓2
Битовая глубина	16	24	8	8	16	24

Построить: представление формы сигнала (во временной области), визуализацию частотного спектра, нормализованную энергию цветности, спектrogramму. Выполнить сравнения с исходными данными и сделать соответствующие выводы.

3. Выполнить сегментацию MP3 файла по методу Лапласа (Laplacian segmentation). В отчете отразить теоретический и математический материал, методику реализации и результаты выполнения сегментации. (можно посмотреть в документации Librosa). Сделать соответствующие выводы.

### **3. Ход работы:**

#### *Чтение заголовка фреймов*

```
Маркер фрейма: '111111111111'
Индекс версии MPEG: '11', MPEG-1
Индекс версии Layer: '01', Layer 3
Бит защиты: '1', CRC защита
Индекс битрейта, MPEG-1 Layer 3: '1011', битрейт в кбит/с: 192
Индекс частоты дискретизации: '00', для MPEG-1 частота дискретизации: 44 100 Гц
Бит смещения: '0', Нет смещения
Бит private: '0'
Индекс режима канала: '11', Режим канала: Mono
Расширение режима канала: '00'
Копирайт: '0'
Оригинал: '1'
Акцент: '00'
```

Рисунок 1 – Заголовок входного файла

Тип битрейта определён как CBR, постоянный и равный 192 кбит/с.

Длительность воспроизведения при постоянном битрейте вычисляется по формуле:

$$L = \frac{V}{B * 8}, \quad (1)$$

где L – длина воспроизведения, V – объём аудиоданных, B – битрейт.

$$L = \frac{84,4}{192 * 8},$$

Для входного файла длительность воспроизведения составила 3,52 с.

### *Изменение параметров тр3 файла*

```
Исходная частота: 44100 Гц → новая: 22050 Гц  
Битовая глубина: 8 бит (256 уровней)
```

Рисунок 2.1 – Параметры амплитуды и битовой глубины входного

```
==== АМПЛИТУДА ====  
Исходная: 0.238035  
После обработки: 0.235294  
Битовая глубина (по заданию): 8 бит
```

Рисунок 2.2 – Параметры амплитуды и битовой глубины файла после обработки

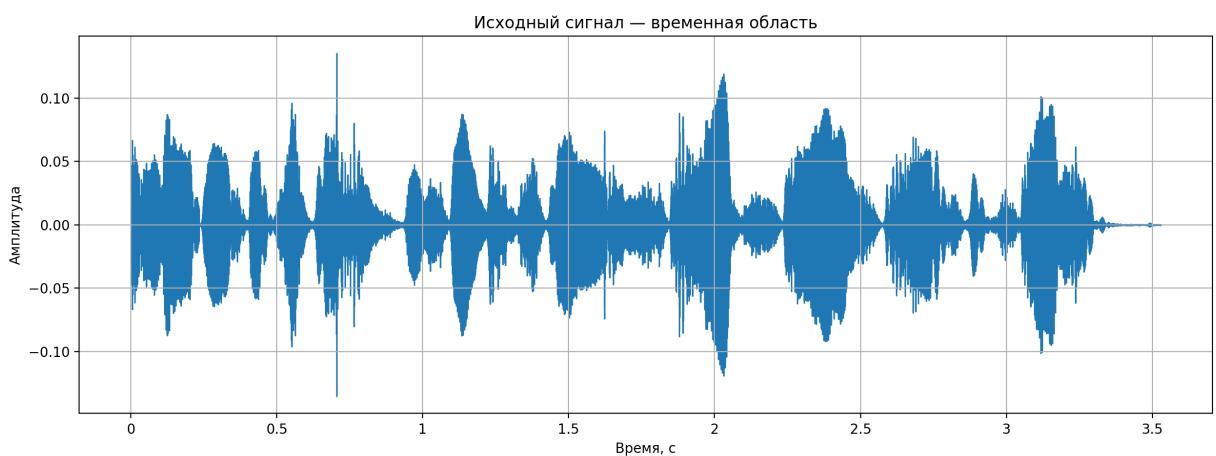


Рисунок 3 – График формы исходного сигнала во временной области

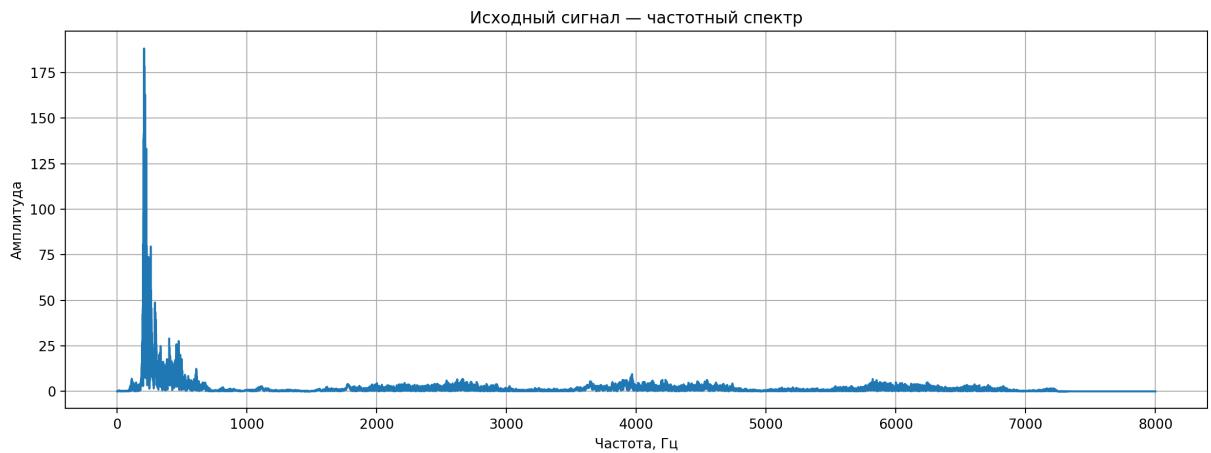


Рисунок 4 – Частотный спектр исходного сигнала

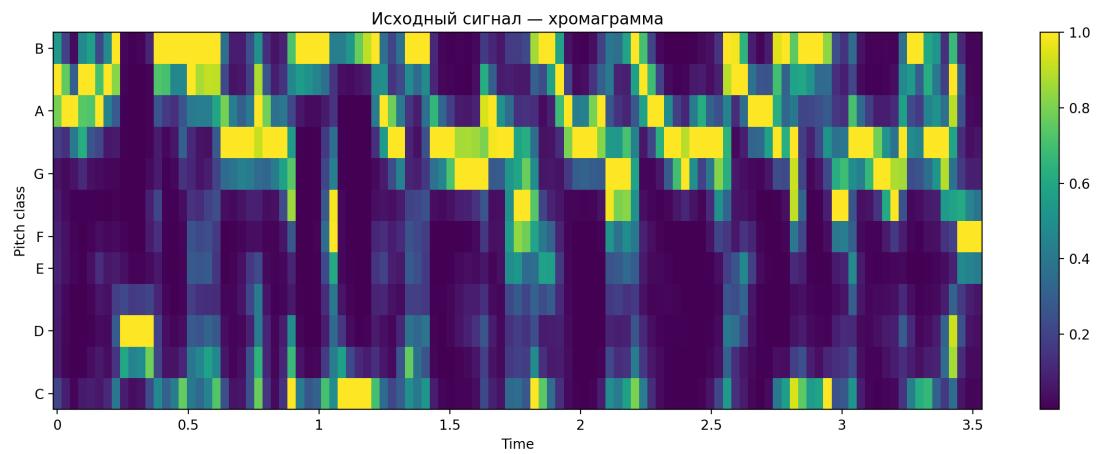


Рисунок 5 – Хромаграмма исходного сигнала

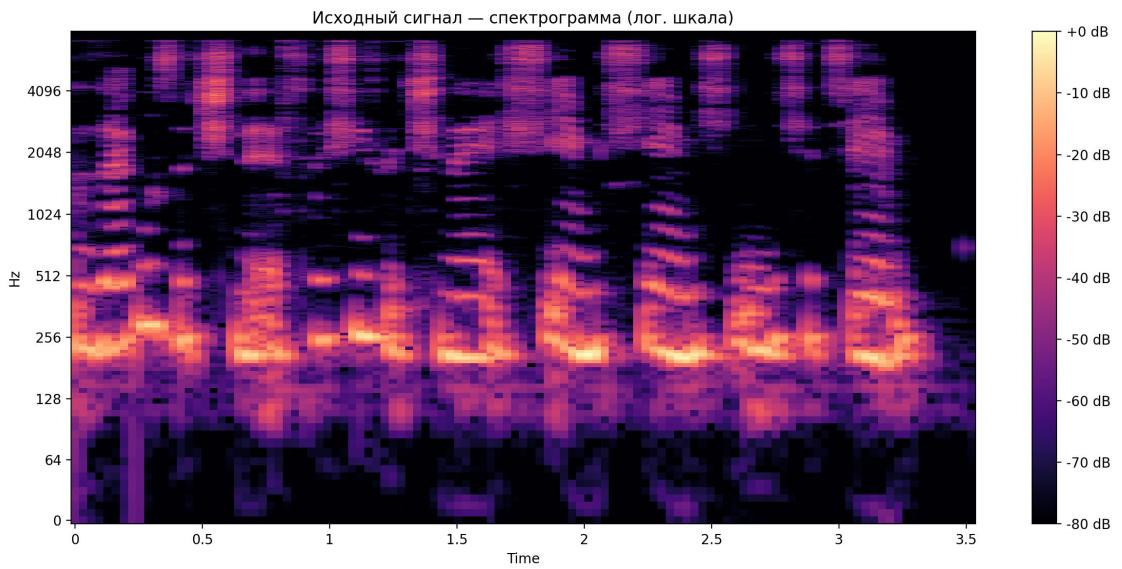


Рисунок 6 – Спектрограмма исходного сигнала

Далее был сохранён обработанный mp3 файл с новыми параметрами:

- Частота дискретизации: 22050 Гц
- Битовая глубина: 8

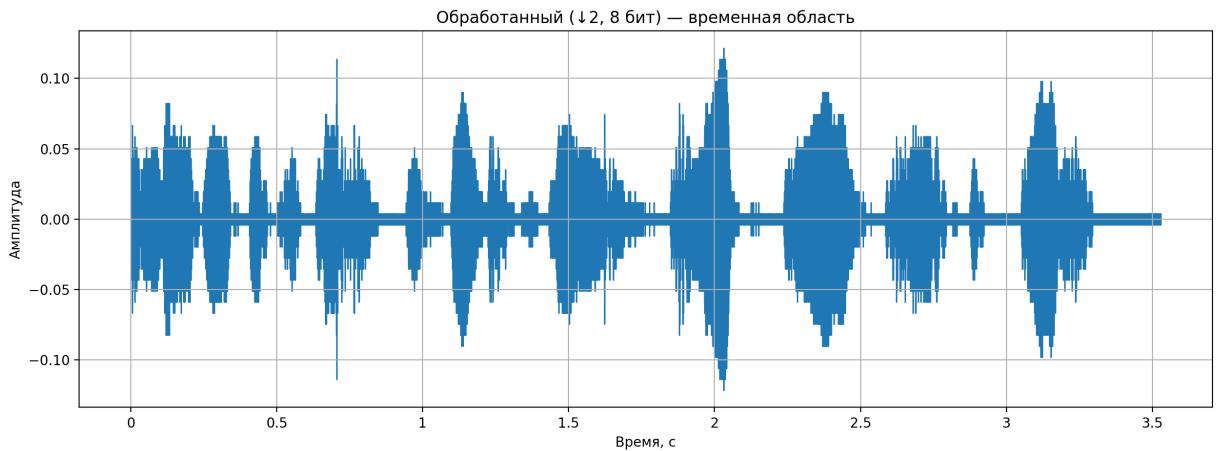


Рисунок 7 – График обработанного сигнала во временной области



Рисунок 8 – Частотный спектр обработанного сигнала

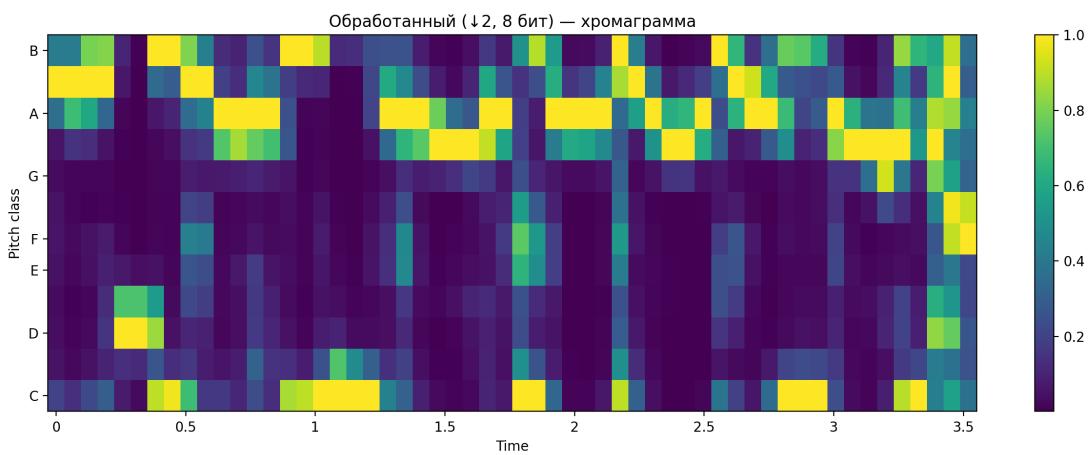


Рисунок 9 – Хромаграмма обработанного сигнала

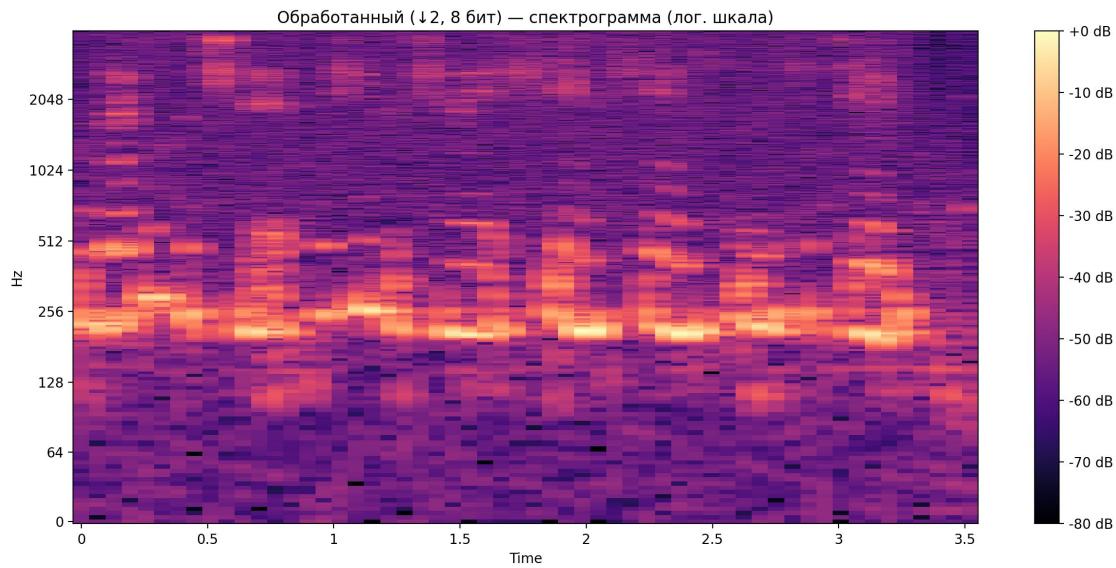


Рисунок 10 – Спектrogramма обработанного сигнала

Обработанный сигнал был прослушан и сравнён с исходным: звук стал более шипящим и содержащим посторонние звенящие сигналы, шумы. Также уменьшился объём файла с 84,4 кБайт до 16,5 кБайт.

## *Сегментация по методу Лапласа*

Ниже приведены результаты сегментации по методу Лапласа:

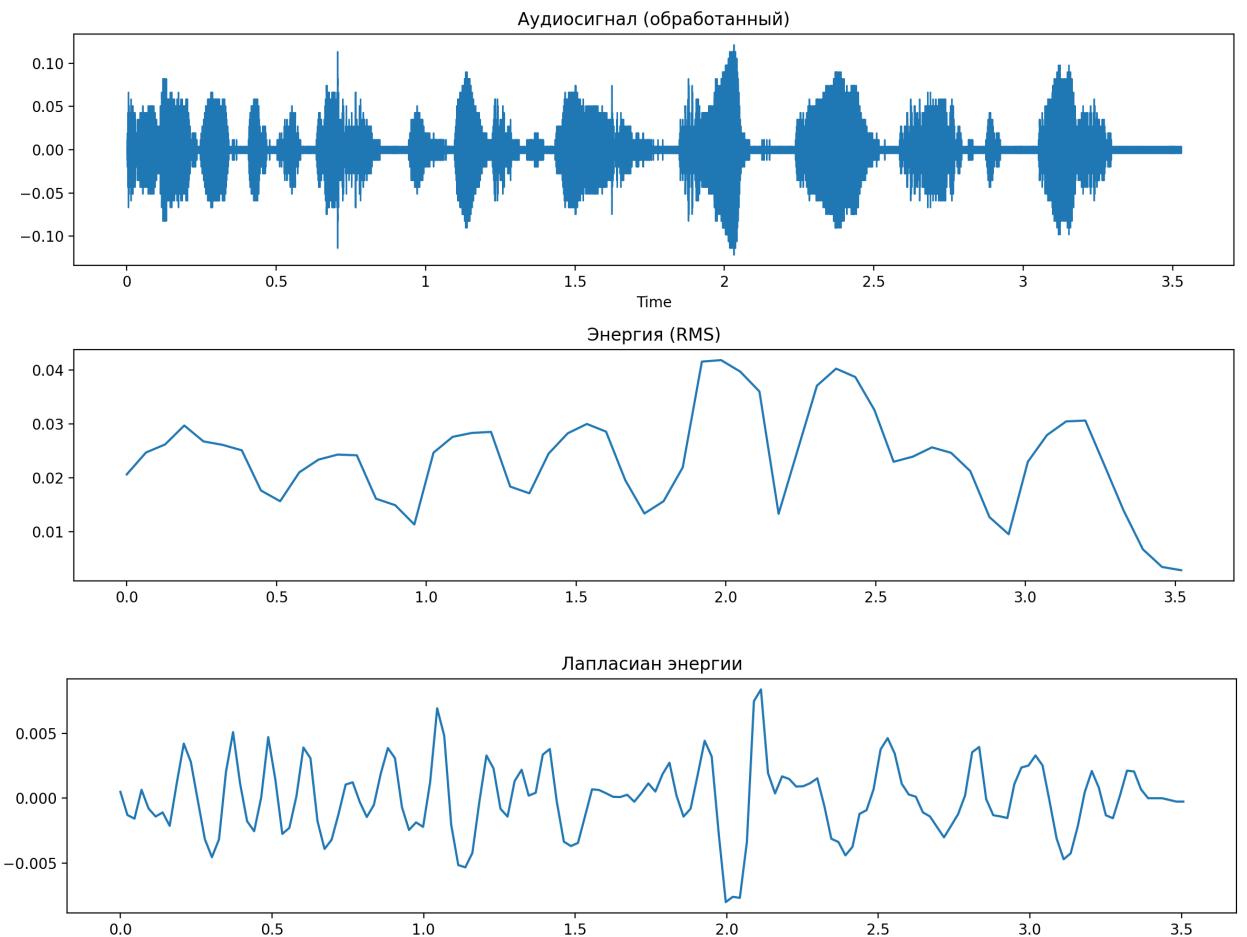


Рисунок 11.1 – График Лапласиана энергии

```
Временные метки (сек): [0.023 0.046 0.232 0.244 0.279 0.29 0.302 0.313 0.36 0.372 0.383 0.395  
0.418 0.43 0.441 0.476 0.488 0.499 0.511 0.534 0.546 0.557 0.569 0.592  
0.604 0.615 0.662 0.673 0.685 0.697 0.731 0.743 0.917 0.964 0.975 1.068  
1.08 1.115 1.126 1.138 1.149 1.161 1.184 1.196 1.207 1.242 1.254 1.312  
1.405 1.416 1.428 1.463 1.474 1.486 1.498 1.823 1.834 1.869 1.927 1.939  
1.95 2.009 2.02 2.032 2.043 2.055 2.067 2.078 2.09 2.206 2.218 2.357  
2.368 2.38 2.438 2.45 2.473 2.566 2.844 2.856 2.879 2.891 2.902 3.03  
3.1 3.111 3.135 3.181 3.193]
```

Рисунок 11.2 - Сегменты

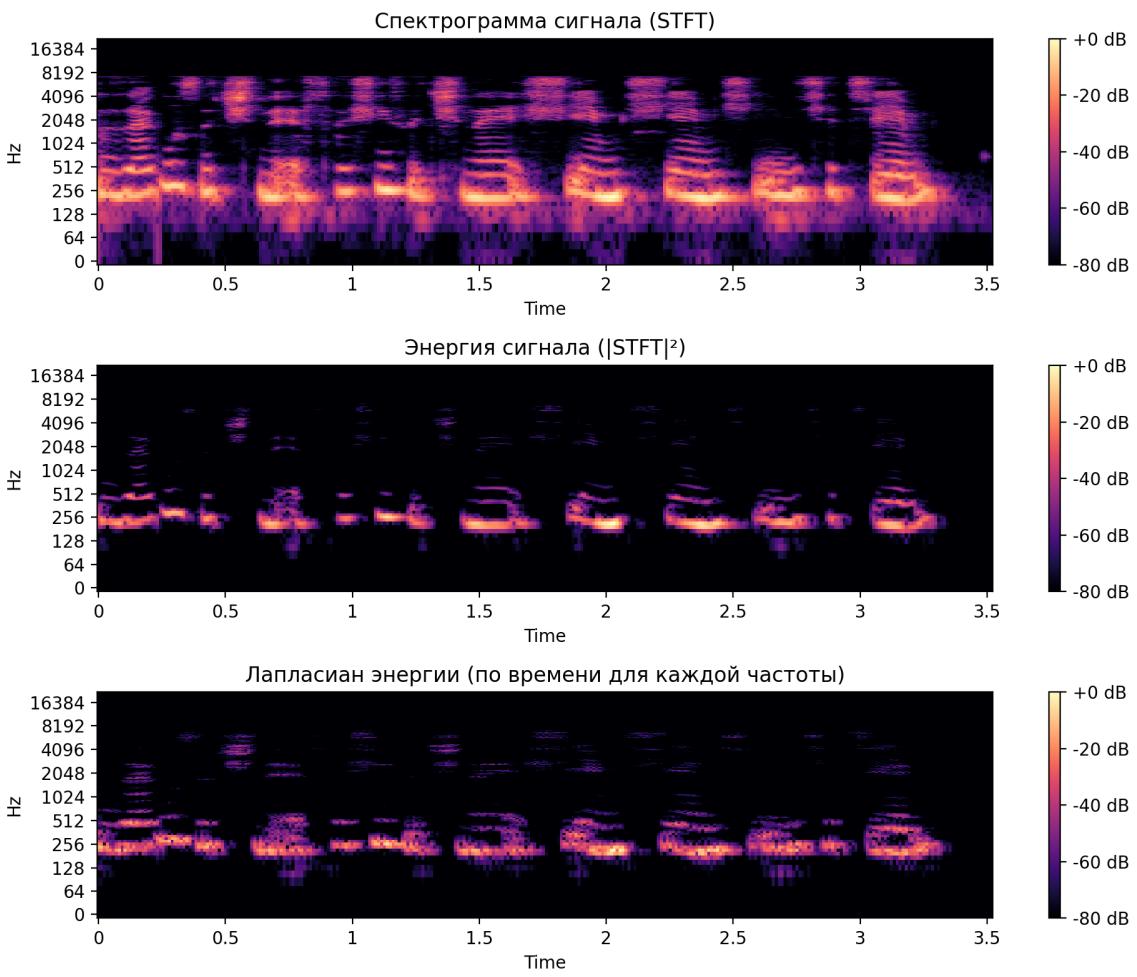


Рисунок 12 – Спектрограммы Лапласиана энергии

График аудиосигнала отражает его форму во временной области – как амплитуда звуковой волны изменяется от момента к моменту.

График энергии сигнала показывает, насколько громким является звук в каждый момент времени. Он рассчитывается как среднеквадратичное значение (RMS) амплитуды на коротких временных окнах.

Пики на этом графике соответствуют громким фрагментам, а провалы — тишине или слабому звучанию.

График лапласиана энергии, в свою очередь, отображает темп изменения энергии: он выявляет те моменты, когда громкость резко нарастает или спадает.

Математически лапласиан представляет собой вторую производную энергетической огибающей сигнала.

Благодаря этому свойству, лапласиан энергии эффективно используется для обнаружения границ сегментов — например, начала или окончания речи, удара по барабану или смены музыкальной фразы.

Суть метода заключается в том, что наибольшие по модулю значения второй производной возникают именно в тех точках, где энергия меняется наиболее стремительно, что и указывает на возможную границу между аудиосегментами.

Используемый математический аппарат:

Среднеквадратичное значение:

$$RMS = \sqrt{\frac{\sum_{i=1}^n x_i^2}{n}}, \quad (2)$$

По умолчанию используется  $n = 512$ .

Используется оператор Лапласа (двойной градиент):

$$\Delta u = \sum_{i=1}^n \frac{\partial^2 u}{\partial x_i^2}, \quad (3)$$

Матрица смежности:

$$W_{ij} = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right), \quad (4)$$

Матрица Степеней:

$$D_{ii} \sum_j W_{ij}, \quad (5)$$

Графовый Лапласиан:

$$L = D - W, \quad (6)$$

Собственные векторы:

$$Lv = \lambda v, \quad (7)$$

$v$  – собственный вектор.

Границам аудио соответствуют:

- Собственные векторы с малыми  $\lambda$
- Разрывы и скачки в этих векторах

## ВЫВОД

В ходе выполнения лабораторной работы были изучены структура и особенности формата MP3, а также отработаны навыки анализа и обработки аудиоданных с использованием библиотек Python (*librosa*, *mutagen*, *pydub*, *matplotlib* и др.).

В пункте 1 был успешно считан заголовок первого фрейма MP3-файла. На основе анализа битовых полей определены ключевые параметры: версия стандарта (MPEG-1), уровень сжатия (Layer III), битрейт, частота дискретизации и режим канала. Длительность файла рассчитана двумя способами – по метаданным и по формуле для CBR – с совпадающими результатами, что подтверждает корректность анализа.

В пункте 2 выполнена обработка аудиосигнала в соответствии с вариантом 4: частота дискретизации уменьшена в 2 раза, а сигнал проквантован до 8 бит (256 уровней). Это позволило смоделировать влияние пониженного качества аудио на его восприятие и анализ. По результатам построения осциллограммы, частотного спектра, хромаграммы и спектрограммы можно сделать следующие наблюдения:

- При уменьшении частоты дискретизации в 2 раза происходят потери высокочастотных компонентов, что ограничивает воспроизводимый частотный диапазон и ухудшает тембр звучания.
- Квантование до 8 бит приводит к появлению шума квантования, особенно заметного на тихих участках сигнала, что проявляется как «ступенчатость» формы волны и фоновый шум на спектрограмме.
- Хромаграмма становится менее чёткой, что снижает точность анализа гармонической структуры музыкального сигнала.

В пункте 3 выполнена сегментация исходного аудиофайла методом Лапласа. На основе второй производной энергии сигнала (RMS) были обнаружены границы сегментов, соответствующие резким изменениям громкости – например, началу или окончанию звучания. Метод показал высокую чувствительность к динамическим переходам, что делает его эффективным для задач автоматической разметки речи или музыки.

Таким образом, лабораторная работа позволила не только освоить формат MP3 на уровне заголовков и параметров, но и практически исследовать влияние цифровых преобразований на качество и анализируемость аудиосигнала, а также применить методы сегментации на основе физических свойств звука. Полученные навыки могут быть полезны при разработке систем обработки аудио, распознавания речи и музыкального информационного поиска.

## ПРИЛОЖЕНИЕ А

### Листинг Программы

```
import os
import numpy as np
import matplotlib.pyplot as plt
from scipy.fft import fft
from mutagen.mp3 import MP3
import librosa
import soundfile as sf
from pydub import AudioSegment

# === ПАРАМЕТРЫ ===
INPUT_MP3 = r"закусочная-сосисочная-4.mp3"
OUTPUT_MP3 = r"output_variant4.mp3"

# Если ffmpeg не в PATH - указать путь:
# AudioSegment.converter = r"C:\ffmpeg\ffmpeg-7.1.1-full_build\bin\ffmpeg.exe"
# AudioSegment.ffprobe = r"C:\ffmpeg\ffmpeg-7.1.1-full_build\bin\ffprobe.exe"

# === 1. ЧТЕНИЕ И АНАЛИЗ ЗАГОЛОВКА MP3 ===
def read_header(header: str) -> None:
    # header = b"ffffbe204"

    bits = bin(int(header, 16)).removeprefix("0b")

    print()
    print(f"Маркер фрейма: {repr(bits[0:11])}")

    match bits[11:13]:
        case "00":
            print(f"Индекс версии MPEG: {repr(bits[11:13])}, MPEG-2.5")
        case "01":
            print(f"Индекс версии MPEG: {repr(bits[11:13])}, Не используется")
        case "10":
            print(f"Индекс версии MPEG: {repr(bits[11:13])}, MPEG-2")
        case "11":
            print(f"Индекс версии MPEG: {repr(bits[11:13])}, MPEG-1")

    match bits[13:15]:
        case "00":
            print(f"Индекс версии Layer: {repr(bits[13:15])}, Не используется")
        case "01":
            print(f"Индекс версии Layer: {repr(bits[13:15])}, Layer 3")
        case "10":
            print(f"Индекс версии Layer: {repr(bits[13:15])}, Layer 2")
        case "11":
            print(f"Индекс версии Layer: {repr(bits[13:15])}, Layer 1")

    print(f"Бит защиты: {repr(bits[15])}, {'Нет защиты' if bits[15] == '0' else 'CRC защита'}")

    print(f"Индекс битрейта, MPEG-1 Layer 3: {repr(bits[16:20])}, битрейт в кбит/с: ", end="")
    match bits[16:20]:
        case "0000":
            print("Не используется")
        case "0001":
            print(32)
```

```

        case "0010":
            print(40)
        case "0011":
            print(48)
        case "0100":
            print(56)
        case "0101":
            print(64)
        case "0110":
            print(80)
        case "0111":
            print(96)
        case "1000":
            print(112)
        case "1001":
            print(128)
        case "1010":
            print(160)
        case "1011":
            print(192)
        case "1100":
            print(224)
        case "1101":
            print(256)
        case "1110":
            print(320)
        case "1111":
            print("Не используется")

    print(f"Индекс частоты дискретизации: {repr(bits[20:22])}, для MPEG-
1частота дискретизации: ", end="")
    match bits[20:22]:
        case "00":
            print("44 100 Гц")
        case "01":
            print("48 000 Гц")
        case "10":
            print("32 000 Гц")
        case "11":
            print("Не используется")

    print(f"Бит смещения: {repr(bits[22])}, {'Нет смещения' if bits[22] ==
'0' else 'Смещение данных на 1 байт'}")

    print(f"Бит private: {repr(bits[23])}")

    print(f"Индекс режима канала: {repr(bits[24:26])}, Режим канала: ",
end="")
    match bits[24:26]:
        case "00":
            print("Stereo")
        case "01":
            print("Joint stereo")
        case "10":
            print("Dual Channel")
        case "11":
            print("Mono")

    print(f"Расширение режима канала: {repr(bits[26:28])}")
    print(f"Копирайт: {repr(bits[28])}")
    print(f"Оригинал: {repr(bits[29])}")
    print(f"Акцент: {repr(bits[30:])}")

```

```

def analyze_mp3_header_and_duration(path: str):
    audio = MP3(path)
    print(f"\n==== ОБЩИЕ ПАРАМЕТРЫ ФАЙЛА ====")
    print(f"Длительность: {audio.info.length:.2f} сек")
    print(f"Частота дискретизации: {audio.info.sample_rate} Гц")
    print(f"Битрейт: {audio.info.bitrate // 1000} кбит/с")

    with open(path, "rb") as f:
        if f.read(3) == b"ID3":
            f.seek(0)
            id3_header = f.read(10)
            id3_size = (id3_header[6] << 21) | (id3_header[7] << 14) |
(id3_header[8] << 7) | id3_header[9]
            f.seek(10 + id3_size)

            while True:
                hdr = f.read(4)
                if len(hdr) < 4:
                    print("Заголовок не найден!")
                    return
                if hdr[0] == 0xFF and (hdr[1] & 0xE0) == 0xE0:
                    read_header(hdr.hex())
                    break
            f.seek(f.tell() - 3)

# === 2. ОБРАБОТКА: ↓2 + 8 БИТ ===
def process_variant4(input_path: str, output_path: str):
    y, sr = librosa.load(input_path, sr=None, mono=True)
    print(f"\n==== ОБРАБОТКА ПО ВАРИАНТУ 4 ====")
    print(f"Исходная частота: {sr} Гц → новая: {sr // 2} Гц")
    print("Битовая глубина: 8 бит (256 уровней)")

    # ↓2
    y_res = librosa.resample(y, orig_sr=sr, target_sr=sr // 2)
    new_sr = sr // 2

    # 8-битное квантование
    levels = 256
    y_norm = (y_res + 1) / 2
    y_quant = np.round(y_norm * (levels - 1)) / (levels - 1)
    y_quant = y_quant * 2 - 1

    # Сохранение в MP3
    y_int16 = (y_quant * 32767).astype(np.int16)
    audio = AudioSegment(
        y_int16.tobytes(),
        frame_rate=new_sr,
        sample_width=2,
        channels=1
    )
    audio.export(output_path, format="mp3")
    print(f"☑ Файл сохранён: {output_path}")
    return y_quant, new_sr

# === 3. ВИЗУАЛИЗАЦИИ ===
def plot_comparison(original_path: str, processed_y: np.ndarray,
processed_sr: int):
    y_orig, sr_orig = librosa.load(original_path, sr=None)

    # Амплитуды

```

```

amp_orig = np.max(y_orig) - np.min(y_orig)
amp_proc = np.max(processed_y) - np.min(processed_y)
print(f"\n==== АМПЛИТУДА ===")
print(f"Исходная: {amp_orig:.6f}")
print(f"После обработки: {amp_proc:.6f}")
print("Битовая глубина (по заданию): 8 бит")

titles = ["Исходный сигнал", "Обработанный (↓2, 8 бит)"]
signals = [y_orig, processed_y]
srs = [sr_orig, processed_sr]

for i, (y, sr, title) in enumerate(zip(signals, srs, titles)):
    # 1. Осциллограмма
    plt.figure(figsize=(10, 2))
    librosa.display.waveform(y, sr=sr)
    plt.title(f"{title} — временная область")
    plt.xlabel("Время, с")
    plt.ylabel("Амплитуда")
    plt.grid(True)
    plt.tight_layout()
    plt.show()

    # 2. Спектр
    spec = fft(y)
    freqs = np.fft.fftfreq(len(spec), 1 / sr)
    plt.figure(figsize=(10, 2))
    plt.plot(freqs[:len(freqs)//2], np.abs(spec[:len(spec)//2]))
    plt.title(f"{title} — частотный спектр")
    plt.xlabel("Частота, Гц")
    plt.ylabel("Амплитуда")
    plt.grid(True)
    plt.tight_layout()
    plt.show()

    # 3. Хромограмма
    chroma = librosa.feature.chroma_stft(y=y, sr=sr)
    chroma_norm = chroma / (chroma.max(axis=0, keepdims=True) + 1e-8)
    plt.figure(figsize=(10, 2))
    librosa.display.specshow(chroma_norm, sr=sr, x_axis='time',
    y_axis='chroma', cmap='viridis')
    plt.colorbar()
    plt.title(f"{title} — хромограмма")
    plt.tight_layout()
    plt.show()

    # 4. Спектрограмма
    X = librosa.stft(y)
    Xdb = librosa.amplitude_to_db(np.abs(X), ref=np.max)
    plt.figure(figsize=(10, 3))
    librosa.display.specshow(Xdb, sr=sr, x_axis='time', y_axis='log',
    cmap='magma')
    plt.colorbar(format='%.2f dB')
    plt.title(f"{title} — спектрограмма (лог. шкала)")
    plt.tight_layout()
    plt.show()

# === 4. СЕГМЕНТАЦИЯ ПО ЛАПЛАСУ ===
def laplacian_segmentation(y: np.ndarray, sr: int, threshold: float = 0.08):
    print("\n==== СЕГМЕНТАЦИЯ ПО ЛАПЛАСУ ===")
    rms = librosa.feature.rms(y=y, frame_length=2048, hop_length=512)[0]
    lap = np.gradient(np.gradient(rms))
    times = librosa.frames_to_time(np.arange(len(rms)), sr=sr,

```

```

hop_length=512)

    seg_points = times[np.abs(laplacian := lap) > threshold]
    print(f"Найдено {len(seg_points)} сегментов")
    print(f"Временные метки: {seg_points}")

    plt.figure(figsize=(12, 9))
    plt.subplot(3, 1, 1)
    librosa.display.waveshow(y, sr=sr)
    plt.title("Аудиосигнал (обработанный)")

    plt.subplot(3, 1, 2)
    plt.plot(times, rms)
    plt.title("Энергия (RMS)")

    plt.subplot(3, 1, 3)
    plt.plot(times, lap)
    for t in seg_points:
        plt.axvline(x=t, color='red', linestyle='--', alpha=0.7)
    plt.title("Лапласиан энергии")
    plt.tight_layout()
    plt.show()

import numpy as np
import matplotlib.pyplot as plt
import librosa

def laplacian_segmentation_spectrograms(file_path: str, threshold: float = 0.08):
    """
    Выполняет сегментацию исходного MP3-файла методом Лапласа.
    Визуализирует:
        - Аудиосигнал, энергию, лапласиан во временной области
        - Спектрограммы сигнала, энергии, лапласиана энергии
    """
    print("\n==== СЕГМЕНТАЦИЯ ЛАПЛАСА (ИСХОДНЫЙ СИГНАЛ) ====")

    # 1. Загрузка сигнала
    y, sr = librosa.load(file_path, sr=None, mono=True)
    print(f"Частота дискретизации: {sr} Гц")
    print(f"Длительность: {len(y) / sr:.2f} сек")

    # 2. Параметры STFT
    n_fft = 2048
    hop_length = 512
    frame_length = n_fft

    # 3. Энергия сигнала (RMS)
    rms = librosa.feature.rms(y=y, frame_length=frame_length,
                               hop_length=hop_length)[0]
    times_rms = librosa.frames_to_time(np.arange(len(rms)), sr=sr,
                                        hop_length=hop_length)

    # 4. Лапласиан энергии (вторая производная по времени)
    laplacian_rms = np.gradient(np.gradient(rms))
    seg_points = times_rms[np.abs(laplacian_rms) > threshold]

    # 5. STFT и спектрограммы
    stft = librosa.stft(y, n_fft=n_fft, hop_length=hop_length)
    spectrogram_energy = np.abs(stft) ** 2 # |STFT|^2 – энергия
    freqs = librosa.fft_frequencies(sr=sr, n_fft=n_fft)
    times_stft =
        librosa.frames_to_time(np.arange(spectrogram_energy.shape[1])), sr=sr,

```

```

hop_length=hop_length)

# 6. Лапласиан энергии в частотно-временной области
# Для каждой частотной полосы вычисляем лапласиан по времени
laplacian_spectrogram = np.zeros_like(spectrogram_energy)
for i in range(spectrogram_energy.shape[0]):
    laplacian_spectrogram[i] =
        np.gradient(np.gradient(spectrogram_energy[i]))

# 7. Визуализация
# --- Спектрограммы ---
plt.figure(figsize=(10, 8))
ax4 = plt.subplot(3, 1, 1)
img4 = librosa.display.specshow(
    librosa.amplitude_to_db(np.abs(stft), ref=np.max),
    sr=sr,
    x_axis='time',
    y_axis='log',
    cmap='magma'
)
ax4.set_title("Спектрограмма сигнала (STFT)")
plt.colorbar(img4, format='%+2.0f dB', ax=ax4)

ax5 = plt.subplot(3, 1, 2)
img5 = librosa.display.specshow(
    librosa.amplitude_to_db(spectrogram_energy, ref=np.max),
    sr=sr,
    x_axis='time',
    y_axis='log',
    cmap='magma'
)
ax5.set_title("Энергия сигнала ( $|STFT|^2$ )")
plt.colorbar(img5, format='%+2.0f dB', ax=ax5)

ax6 = plt.subplot(3, 1, 3)
img6 = librosa.display.specshow(
    librosa.amplitude_to_db(np.abs(laplacian_spectrogram), ref=np.max),
    sr=sr,
    x_axis='time',
    y_axis='log',
    cmap='magma'
)
ax6.set_title("Лапласиан энергии (по времени для каждой частоты)")
plt.colorbar(img6, format='%+2.0f dB', ax=ax6)

plt.tight_layout()
plt.show()

# 8. Вывод результатов
print(f"\n\N{checkbox} Найдено {len(seg_points)} границ сегментов:")
print(f"Временные метки (сек): {np.round(seg_points, 3)}")

return seg_points

# === ГЛАВНАЯ ФУНКЦИЯ ===
def main():
    if not os.path.exists(INPUT_MP3):
        raise FileNotFoundError(f"Файл не найден: {INPUT_MP3}")

    # 1. Анализ заголовка и длительности
    analyze_mp3_header_and_duration(INPUT_MP3)

    # 2. Обработка по варианту 4

```

```
y_proc, sr_proc = process_variant4(INPUT_MP3, OUTPUT_MP3)

# 3. Визуализации
plot_comparison(INPUT_MP3, y_proc, sr_proc)

# 4. Сегментация
laplacian_segmentation(y_proc, sr_proc)
laplacian_segmentation_spectrograms(INPUT_MP3, threshold=0.08)

print("\n\ufe0f Лабораторная работа выполнена!")

if __name__ == "__main__":
    main()
```