Санкт-Петербургский Государственный Электротехнический Университет "ЛЭТИ"

кафедра физики

Задание №4 по дисциплине

"Физические основы информационных технологий"

Название: Фильтрация звукового сигнала

| Фамилия И.О.: | Герасименко Я.Д. |
| --- | --- |
| группа: | 1303 |
| Преподаватель: | Альтмарк А.М. |
| Итоговый балл: |  |
|  |  |

Крайний срок сдачи: 5.12.23

.

Санкт-Петербург 2023

**Условие задания**

На входе приемника получен звуковой сигнал в двоичном коде (рис.1.). Необходимо перевести двоичный код в десятичный и затем провести над аналоговым сигналом процедуру фильтрации от высокочастотных помех. Для фильтрации необходимо использовать пассивные фильтры (фильтры без дополнительного источника питания), которые могут в себя включать, резисторы, конденсаторы и катушки индуктивности.

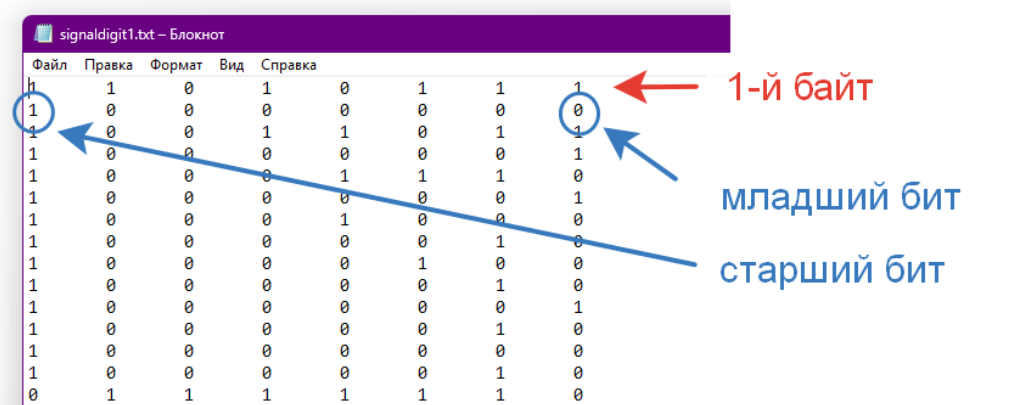


Рис.1. Структура данных в текстовом файле с сигналом

Исходные данные нужно взять в файле FOIT\_IDZ4.xlsx. В отчет нужно включить график сигнала во временной области и его спектр, схему фильтра и АЧХ его передаточной функции, спектр фильтрованного сигнала, а также график выходного сигнала во временной области. Файл IDZ4.txt должен содержать ответ на вопрос, который записан в звуком сигнале.

Помимо текстового файла IDZ4.txt в папке IDZ4 должен находиться Word-файл с отчетом, а также файл с кодом (Python, Mathcad, Mathematica). Для лучшего понимания отчетности смотрите папку “Пример организации яндекс-папки студентов”.

Пример содержания файла IDZ4.txt:

25

Вариант 20

t = 2.75

**Основные теоретические положения**

Для фильтрации сигнала в работе был использован фильтр Баттерворта. Так как он позволяет достаточно точно обрезать высокие частоты шума и распознать сообщение. Фильтр представлен на рисунке 2.

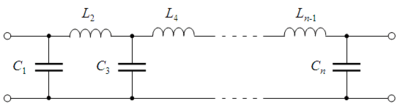


Рис.2. Фильтр Баттерворта низких частот

**Выполнение работы**

для фильтрации сигнала в своей работе я использовал фильтр из библиотеки signal. И реализвал свою функцию **def butterworth\_filter(data, sample\_rate, cutoff, order).**

**Описание функции:**

*Нормализация частоты среза:*

* + nyquist: Это половина частоты дискретизации сигнала (часто называемая частотой Найквиста). Она определяет максимальную частоту, которую можно корректно воспроизвести при данной частоте дискретизации, согласно теореме Найквиста-Шеннона.
  + normal\_cutoff: Частота среза фильтра нормализуется относительно частоты Найквиста. Это делается для того, чтобы частота среза была в пределах от 0 до 1, где 1 соответствует частоте Найквиста.
* *Создание фильтра:*
  + Функция signal.butter из библиотеки scipy.signal используется для создания фильтра Баттерворта.
  + Параметры order и normal\_cutoff определяют порядок фильтра и его частоту среза соответственно.
  + btype='low' указывает на то, что это фильтр нижних частот, который пропускает частоты ниже заданной частоты среза и подавляет частоты выше неё.
  + Фильтр Баттерворта выбирается за его плоскую амплитудную характеристику в полосе пропускания, что делает его популярным выбором во многих аудио и сигнальных приложениях.
* *Применение фильтра к сигналу:*
  + signal.filtfilt: Эта функция применяет фильтр ко входному сигналу. Она использует метод прямого и обратного фильтрования, что позволяет избежать фазовых сдвигов, обычно связанных с фильтрацией.
  + b и a - коэффициенты фильтра, полученные от signal.butter.

**ПРИЛОЖЕНИЕ А**

**ИСХОДНЫЙ КОД ПРОГРАММЫ**

import numpy as np

import sounddevice as sd

import scipy.signal as signal

import matplotlib.pyplot as plt

# Функция для чтения цифрового сигнала из файла

def read\_signal(file\_path):

with open(file\_path, 'r') as file:

signal = []

for line in file:

# Разделяем строку на биты и преобразуем их в целые числа

bits = [int(bit) for bit in line.split()]

# Преобразуем список битов в десятичное число

number = int("".join(map(str, bits)), 2)

signal.append(number)

return signal

# Функция фильтра Баттерворта

def butterworth\_filter(data, sample\_rate, cutoff, order):

# Нормализация частоты среза

nyquist = 0.5 \* sample\_rate

normal\_cutoff = cutoff / nyquist # Нормализованная частота среза

# Создание фильтра

b, a = signal.butter(order, normal\_cutoff, btype='low', analog=False) # Фильтр Баттерворта низких частот 6 порядка

# Применение фильтра к сигналу

filtered\_data = signal.filtfilt(b, a, data)

return filtered\_data

# Визуализация сигнала

def plot\_signal(signal, time\_axis, title):

plt.plot(time\_axis, signal)

plt.xlabel('Время, с')

plt.ylabel('Амплитуда')

plt.title(title)

plt.grid()

plt.show()

# Визуализация спектра сигнала

def fftPlot(signal, str='Спектр сигнала до'):

fsig = np.fft.fft(signal)

out\_n = len(fsig)

freq\_axis = np.fft.fftfreq(out\_n, dt)[1:]

plt.plot(freq\_axis, np.abs(fsig[1:])) # Первый элемент - постоянная составляющая

plt.title(str)

plt.xlabel('Частота, Гц')

plt.ylabel('Амплитуда')

plt.show()

def plot\_butterworth\_response(sample\_rate, cutoff, order):

# Получаем коэффициенты фильтра

b, a = signal.butter(order, cutoff / (0.5 \* sample\_rate), btype='low')

# Получаем частотную характеристику

w, h = signal.freqz(b, a, worN=8000)

# Рисуем АЧХ

plt.semilogx(w \* sample\_rate / (2 \* np.pi), np.abs(h)) # Переводим частоту в Гц

plt.title('АЧХ фильтра Баттерворта')

plt.xlabel('Частота, Гц')

plt.ylabel('Амплитуда')

plt.grid()

plt.show()

t = 2.75

# Путь к файлу с вашим сигналом

file\_path = "signaldigit20.txt"

# Чтение и преобразование сигнала

digital\_signal = read\_signal(file\_path)

# Расчет частоты дискретизации

dt = t / len(digital\_signal) # Время дискретизации

sample\_rate = 1 / dt # Частота дискретизации

# Нормализация сигнала к диапазону от -1 до 1

analog\_signal = np.array(digital\_signal)

analog\_signal = (analog\_signal - np.min(analog\_signal)) / (

np.max(analog\_signal) - np.min(analog\_signal)) # Нормализация

analog\_signal = 2 \* analog\_signal - 1 # Приведение к диапазону от -1 до 1

# Визуализация сигнала

time\_axis = np.linspace(0, t, len(analog\_signal))

plot\_signal(analog\_signal, time\_axis, 'Аналоговый сигнал')

fftPlot(analog\_signal)

# Применение фильтра Баттерворта

cutoff\_frequency = 1200 # Частота среза в Гц

filter\_order = 6 # Порядок фильтра

filtered\_signal = butterworth\_filter(analog\_signal, sample\_rate, cutoff\_frequency, filter\_order)

# Визуализация отфильтрованного сигнала

plot\_signal(filtered\_signal, time\_axis, 'Отфильтрованный сигнал')

fftPlot(filtered\_signal, 'Спектр сигнала после')

# ачх фильтра

plot\_butterworth\_response(sample\_rate, cutoff\_frequency, filter\_order)

sd.play(analog\_signal, samplerate=int(sample\_rate))

sd.wait()

sd.play(filtered\_signal, samplerate=int(sample\_rate))

sd.wait()