Санкт-Петербургский Государственный Электротехнический Университет "ЛЭТИ"

кафедра физики

Задание №4 по дисциплине

"Физические основы информационных технологий"

Название: Фильтрация звукового сигнала

|  |  |
| --- | --- |
| Фамилия И.О.: | Хабибуллина А.М |
| группа: | 1303 |
| Преподаватель: | Альтмарк А.М. |
| Итоговый балл: |  |
|  |  |

Крайний срок сдачи: 5.12.23

Санкт-Петербург 2023

Условие задания 4

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.

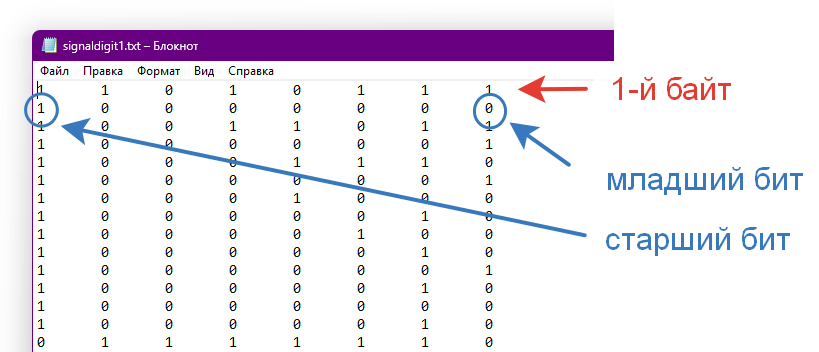


Рис.1. Структура данных в текстовом файле с сигналом

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

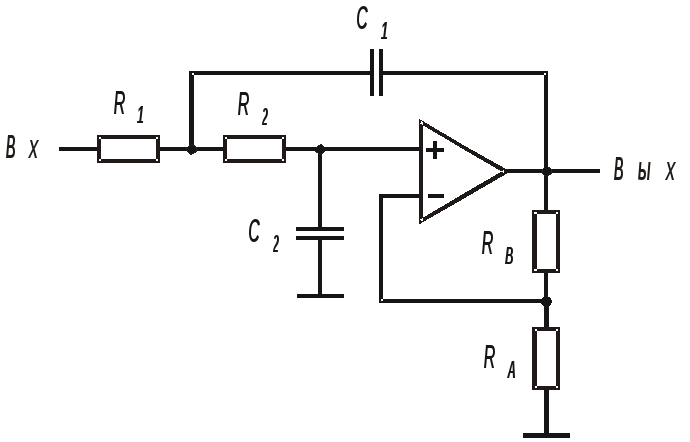
**Дано (Вариант 13):**

t = 2.5с

**Теоретические сведения.**

В лабораторной работе применялся фильтр Баттерворта для фильтрации звукового сигнала, полученного в двоичном коде. Фильтр Баттерворта был выбран из-за своих преимуществ, включая плоскую амплитудно-частотную характеристику в полосе пропускания.

Разработан пассивный фильтр Баттерворта с определенными параметрами, такими как порядок фильтра, частота среза и компоненты схемы (резисторы, конденсаторы, катушки индуктивности). Эта схема фильтра была применена к аналоговому сигналу для устранения высокочастотных помех.



**ПРИЛОЖЕНИЕ А**

**ИСХОДНЫЙ КОД ПРОГРАММЫ**

**IDZ4.py**

import numpy as np

import matplotlib.pyplot as plt

from scipy.io import wavfile

from scipy.fft import fft, ifft

import sounddevice as sd

duration = 2.5

resistor\_R1 = 50

capacitor\_C1 = 0.000005

capacitor\_C2 = 0.0000045

capacitor\_C3 = 0.0000035

capacitor\_C4 = 0.000002

capacitor\_C5 = 497.9 \* 10\*\*-9

inductor\_L1 = 0.01

inductor\_L2 = 0.012

inductor\_L3 = 0.01

inductor\_L4 = 0.007

inductor\_L5 = 0.003

with open("signaldigit13.txt", 'r') as file:

digital\_signal = [line.strip().split('\t') for line in file]

analog\_signal = [int(''.join(bits), 2) for bits in digital\_signal]

time\_step = duration / len(digital\_signal)

sample\_rate = 44100

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

def filter(w, U\_in):

Z\_parallel\_5 = 1 / (1j \* w \* capacitor\_C5 + 1 / resistor\_R1)

Z\_parallel\_4 = 1 / (1j \* w \* capacitor\_C4 + 1 / (1j \* w \* inductor\_L5 + Z\_parallel\_5))

Z\_parallel\_3 = 1 / (1j \* w \* capacitor\_C3 + 1 / (1j \* w \* inductor\_L4 + Z\_parallel\_4))

Z\_parallel\_2 = 1 / (1j \* w \* capacitor\_C2 + 1 / (1j \* w \* inductor\_L3 + Z\_parallel\_3))

Z\_parallel = 1 / (1j \* w \* capacitor\_C1 + 1 / (1j \* w \* inductor\_L2 + Z\_parallel\_2))

I1 = U\_in / (1j \* w \* inductor\_L1 + Z\_parallel)

U\_parallel = I1 \* Z\_parallel

I2 = U\_parallel / (1j \* w \* inductor\_L2 + Z\_parallel\_2)

U\_parallel\_2 = I2 \* Z\_parallel\_2

I3 = U\_parallel\_2 / (1j \* w \* inductor\_L3 + Z\_parallel\_3)

U\_parallel\_3 = I3 \* Z\_parallel\_3

I4 = U\_parallel\_3 / (1j \* w \* inductor\_L4 + Z\_parallel\_4)

U\_parallel\_4 = I4 \* Z\_parallel\_4

I5 = U\_parallel\_4 / (1j \* w \* inductor\_L5 + Z\_parallel\_5)

U\_parallel\_5 = I5 \* Z\_parallel\_5

U\_out = U\_parallel\_5

result = U\_out / U\_in

return result

plt.plot(time\_axis, analog\_signal)

plt.title('Analog Signal')

plt.xlabel('Time (s)')

plt.ylabel('Amplitude')

plt.show()

frequency\_spectrum = fft(analog\_signal)

num\_samples = len(frequency\_spectrum)

df = 1 / time\_step

frequency\_axis = np.fft.fftfreq(num\_samples, time\_step)[1:]

plt.plot(frequency\_axis, np.abs(frequency\_spectrum[1:]))

plt.title('Spectrum')

plt.xlabel('Frequency (Hz)')

plt.ylabel('Amplitude')

plt.show()

U\_in\_value = 1

frequency\_range = np.linspace(1, num\_samples/10, num\_samples - 1)

filter\_response = np.abs(filter(frequency\_range, U\_in\_value))

plt.plot(frequency\_range, filter\_response)

plt.title('Normalized Filter Frequency Response')

plt.xlabel('Frequency (Hz)')

plt.ylabel('Amplitude')

plt.show()

H\_list = filter\_response

F\_new = frequency\_spectrum[1:] \* H\_list

plt.plot(frequency\_axis, np.abs(F\_new))

plt.title('Filtered Spectrum')

plt.xlabel('Frequency (Hz)')

plt.ylabel('Amplitude')

plt.show()

filtered\_signal = ifft(np.concatenate(([0], F\_new)))

filtered\_time\_axis = np.linspace(0, duration, len(filtered\_signal))

plt.plot(filtered\_time\_axis, np.real(filtered\_signal))

plt.title('Filtered Signal')

plt.xlabel('Time (s)')

plt.ylabel('Amplitude')

plt.show()

wavfile.write('filtered\_signal.wav', sample\_rate, np.real(filtered\_signal).astype(np.int16))

sd.play(np.real(filtered\_signal), sample\_rate)

sd.wait()