Санкт-Петербургский Государственный Электротехнический

Университет "ЛЭТИ"

кафедра физики

Задание №4 по дисциплине

"Физические основы информационных технологий"

Название: Фильтрация звукового сигнала

|  |  |
| --- | --- |
| Фамилия И.О.: | Бутыло Е. А. |
| группа: | 1303 |
| Преподаватель: | Альтмарк А. М. |
| Итоговый балл: |  |
| Крайний срок сдачи: | 5.12.23 |

Санкт-Петербург 2023

Условие задания

На входе приёмника 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.

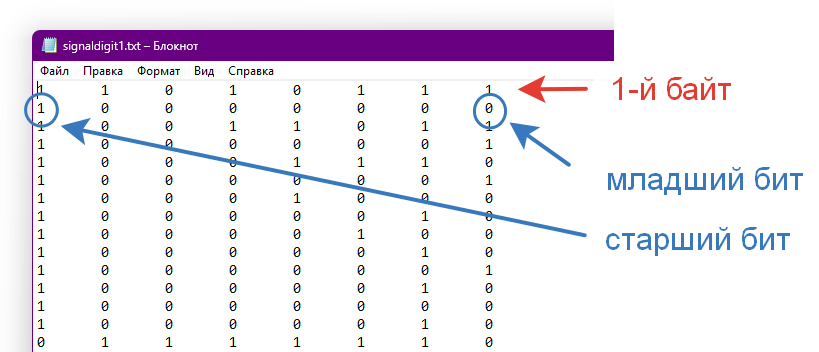


Рис.1. Структура данных в текстовом файле с сигналом

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”.

Исходные данные

|  |  |  |
| --- | --- | --- |
| Вар | длительность сигнала, с | Файл с сигналом |
| 3 | 4 | Signaldigit3.txt |

**Теоретические сведения**

В качестве фильтра низких частот в работе использовался фильтр Баттерворда. Фильтр Баттерворта проектируется так, чтобы его [амплитудно-частотная характеристика](https://ru.wikipedia.org/wiki/%D0%90%D0%BC%D0%BF%D0%BB%D0%B8%D1%82%D1%83%D0%B4%D0%BD%D0%BE-%D1%87%D0%B0%D1%81%D1%82%D0%BE%D1%82%D0%BD%D0%B0%D1%8F_%D1%85%D0%B0%D1%80%D0%B0%D0%BA%D1%82%D0%B5%D1%80%D0%B8%D1%81%D1%82%D0%B8%D0%BA%D0%B0" \o "Амплитудно-частотная характеристика) была максимально гладкой на частотах [полосы пропускания](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D0%BE%D1%81%D0%B0_%D0%BF%D1%80%D0%BE%D0%BF%D1%83%D1%81%D0%BA%D0%B0%D0%BD%D0%B8%D1%8F" \o "Полоса пропускания). Он позволяет эффективно подавить высокие частоты шума и распознать сообщение. Изображение фильтра представлено на рисунке 2.

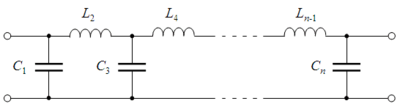


Рис. 2. Фильтр Баттерворта

**ПРИЛОЖЕНИЕ А**

**ПРОГРАММА MAIN.PY**

import matplotlib.pyplot as plt

import sounddevice as sd

from scipy.fft import fft, ifft

import numpy as np

def play\_sound(signal):

sd.play(np.real(signal), 45000)

sd.wait()

def draw\_plot(\*args, title) -> None:

if len(args) == 2:

plt.plot(args[0], args[1])

plt.title(title)

plt.show()

def initialization(filename: str, dur\_time: float) -> tuple[list[int], list[float], float]:

with open(filename, 'r') as signal\_file:

content = signal\_file.read()

content = ''.join(content.split('\t'))

content = content.split('\n')

signal = [int(elem, 2) for elem in content]

n = len(signal)

delta\_t = dur\_time / n

timeline = [i \* delta\_t for i in range(n)]

dt = 1 / n

return signal, timeline, dt

def filter\_freq(freq\_range):

coefficient = 2

L1 = 12.4 \* pow(10, -3) \* coefficient

L2 = 14.4 \* pow(10, -3) \* coefficient

L3 = 12 \* pow(10, -3) \* coefficient

L4 = 8.3 \* pow(10, -3) \* coefficient

L5 = 3.7 \* pow(10, -3) \* coefficient

C1 = 5.9 \* pow(10, -6) \* coefficient

C2 = 5.4 \* pow(10, -6) \* coefficient

C3 = 4.1 \* pow(10, -6) \* coefficient

C4 = 2.4 \* pow(10, -6) \* coefficient

C5 = 497.9 \* pow(10, -9) \* coefficient

R = 50

def ZL(L, omega):

return 1j \* omega \* L

def ZC(C, omega):

return 1 / (1j \* omega \* C)

def H(omega):

Z5par = 1 / ((1 / R) + 1 / (ZC(C5, omega)))

Z4par = 1 / ((1 / ZC(C4, omega)) + 1 / (Z5par + ZL(L5, omega)))

Z3par = 1 / (1 / (ZC(C3, omega)) + 1 / (Z4par + ZL(L4, omega)))

Z2par = 1 / (1 / (ZC(C2, omega)) + 1 / (Z3par + ZL(L3, omega)))

Z1par = 1 / (1 / (ZC(C1, omega)) + 1 / (Z2par + ZL(L2, omega)))

ZL1 = ZL(L1, omega)

Zall = ZL1 + Z1par

Iin = 1 / Zall

Upar1 = Iin \* Z1par

I1top = Upar1 / (Z2par + ZL(L2, omega))

Upar2 = I1top \* Z2par

I2top = Upar2 / (Z3par + ZL(L3, omega))

Upar3 = I2top \* Z3par

I3top = Upar3 / (Z4par + ZL(L4, omega))

Upar4 = I3top \* Z4par

I4top = Upar4 / (Z5par + ZL(L5, omega))

Upar5 = I4top \* Z5par

return Upar5

return H(freq\_range)

def main() -> None:

duration: float = 4

file: str = 'signaldigit3.txt'

signal, timeline, dt = initialization(file, duration)

draw\_plot(timeline, signal, title='Input signal')

sign\_ampl = fft(signal)

n = len(sign\_ampl)

freq = np.fft.fftfreq(n, dt)[1:]

draw\_plot(freq, np.abs(sign\_ampl[1:]), title='Input spectrum')

freq\_range = np.linspace(1, n, n - 1)

filtered = np.abs(filter\_freq(freq\_range))

plt.xlim(0, 10000)

draw\_plot(freq\_range, filtered, title='Filter frequency response')

H = filtered

F = sign\_ampl[1:] \* H

draw\_plot(freq, np.abs(F), title='Result spectrum')

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

filtered\_time\_axis = np.linspace(0, 4, len(changed\_signal))

draw\_plot(filtered\_time\_axis, np.real(changed\_signal), title='Result signal')

# play\_sound(changed\_signal)

if \_\_name\_\_ == "\_\_main\_\_":

main()