Санкт-Петербургский Государственный Электротехнический Университет "ЛЭТИ"

кафедра физики

Задание №4 по дисциплине

"Физические основы информационных технологий"

Название: Фильтрация звукового сигнала

|  |  |
| --- | --- |
| Фамилия И.О.: | Андреева Е.А. |
| 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.

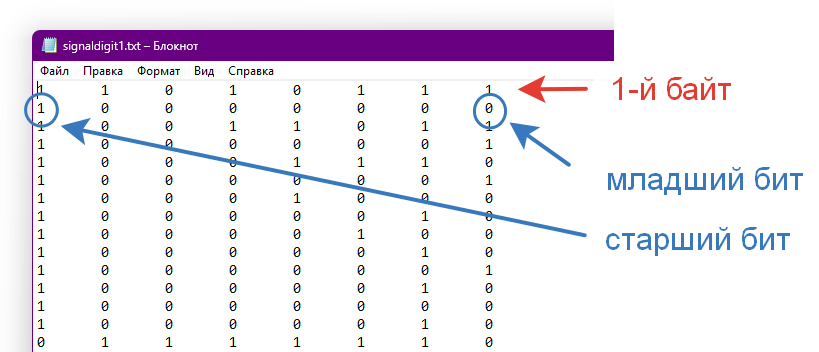


Рис.1. Структура данных в текстовом файле с сигналом

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

Исходные данные

|  |  |  |
| --- | --- | --- |
| Вар | длительность сигнала, с | Файл с сигналом |
| 1 | 3,75 | signaldigit1.txt |

Основные теоретические положения

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) 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). Он позволяет эффективно подавить высокие частоты шума и распознать сообщение. Изображение фильтра представлено на рисунке 2.

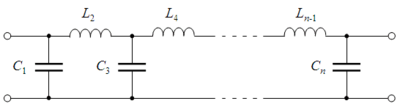


Рис. 2. Фильтр Баттерворта

**ПРИЛОЖЕНИЕ А**

**ФАЙЛ IDZ4.PY**

import numpy as np  
import matplotlib.pyplot as plt  
from scipy.fft import fft, ifft  
import sounddevice as sd  
  
  
def read\_analog\_signal(*filename*):  
 with *open*(*filename*, 'r') as file:  
 discret\_signal = [line.strip().split('\t') for line in file]  
 return [*int*(''.join(b), 2) for b in discret\_signal]  
  
  
def plot\_time\_domain(*signal*, *t*):  
 time\_axis = np.linspace(0, *t*, *len*(*signal*))  
 plt.plot(time\_axis, *signal*)  
 plt.title('Input Signal')  
 plt.xlabel('Time (s)')  
 plt.ylabel('Amplitude')  
 plt.show()  
  
  
def plot\_frequency\_spectrum(*signal*):  
 fsig = fft(*signal*)  
 out\_n = *len*(fsig)  
 dt = 1 / *len*(*signal*)  
 freq\_axis = np.fft.fftfreq(out\_n, dt)[1:]  
 plt.plot(freq\_axis, np.abs(fsig[1:]))  
 plt.title('Frequency Spectrum')  
 plt.xlabel('Frequency (Hz)')  
 plt.ylabel('Amplitude')  
 plt.show()  
  
  
def butterworth\_filter(*w*, *Uin*):  
 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  
 R1 = 50  
 Zpar5 = 1 / (1j \* *w* \* C5 + 1 / R1)  
 Zpar4 = 1 / (1j \* *w* \* C4 + 1 / (1j \* *w* \* L5 + Zpar5))  
 Zpar3 = 1 / (1j \* *w* \* C3 + 1 / (1j \* *w* \* L4 + Zpar4))  
 Zpar2 = 1 / (1j \* *w* \* C2 + 1 / (1j \* *w* \* L3 + Zpar3))  
 Zpar = 1 / (1j \* *w* \* C1 + 1 / (1j \* *w* \* L2 + Zpar2))  
 I1 = *Uin* / (1j \* *w* \* L1 + Zpar)  
 Upar = I1 \* Zpar  
 I2 = Upar / (1j \* *w* \* L2 + Zpar2)  
 Upar2 = I2 \* Zpar2  
 I3 = Upar2 / (1j \* *w* \* L3 + Zpar3)  
 Upar3 = I3 \* Zpar3  
 I4 = Upar3 / (1j \* *w* \* L4 + Zpar4)  
 Upar4 = I4 \* Zpar4  
 I5 = Upar4 / (1j \* *w* \* L5 + Zpar5)  
 Upar5 = I5 \* Zpar5  
 Uout = Upar5  
 return Uout / *Uin*def apply\_butterworth\_filter(*signal*):  
 fsig = fft(*signal*)  
 out\_n = *len*(fsig)  
 dt = 1 / *len*(*signal*)  
 freq\_axis = np.fft.fftfreq(out\_n, dt)[1:]  
 freq\_range = np.linspace(1, out\_n, out\_n - 1)  
 filter\_response = np.abs(butterworth\_filter(freq\_range, 1))  
  
 plt.plot(freq\_range, filter\_response)  
 plt.title('Normalized Filter Frequency Response')  
 plt.xlabel('Frequency (Hz)')  
 plt.ylabel('Amplitude')  
 plt.xlim(0, 10000)  
 plt.show()  
  
 H\_list = filter\_response  
 F\_new = fsig[1:] \* H\_list  
  
 plt.plot(freq\_axis, np.abs(F\_new))  
 plt.title('Filtered Spectrum')  
 plt.xlabel('Frequency (Hz)')  
 plt.ylabel('Amplitude')  
 plt.show()  
  
 changed\_signal = ifft(np.concatenate(([0], F\_new)))  
 return np.real(changed\_signal)  
  
  
def plot\_filtered\_signal(*filtered\_signal*, *t*):  
 filtered\_time\_axis = np.linspace(0, *t*, *len*(*filtered\_signal*))  
 plt.plot(filtered\_time\_axis, *filtered\_signal*)  
 plt.title('Filtered Signal')  
 plt.xlabel('Time (s)')  
 plt.ylabel('Amplitude')  
 plt.show()  
  
  
def play\_audio(*signal*, *sample\_rate*):  
 sd.play(*signal*, *sample\_rate*)  
 sd.wait()  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 file\_path = "signaldigit1.txt"  
 analog\_signal = read\_analog\_signal(file\_path)  
  
 t = 3.75  
  
 plot\_time\_domain(analog\_signal, t)  
 plot\_frequency\_spectrum(analog\_signal)  
  
 filtered\_signal = apply\_butterworth\_filter(analog\_signal)  
  
 plot\_filtered\_signal(filtered\_signal, t)  
 play\_audio(filtered\_signal, 44100)

if \_\_name\_\_ == "\_\_main\_\_":  
 file\_path = "signaldigit1.txt"  
 analog\_signal = read\_analog\_signal(file\_path)  
  
 t = 3.75  
  
 plot\_time\_domain(analog\_signal, t)  
 plot\_frequency\_spectrum(analog\_signal)  
  
 filtered\_signal = apply\_butterworth\_filter(analog\_signal)  
  
 plot\_filtered\_signal(filtered\_signal, t)  
 play\_audio(filtered\_signal, 44100)