Санкт-Петербургский Государственный Электротехнический Университет "ЛЭТИ"

кафедра физики

Задание №4 по дисциплине

"Физические основы информационных технологий"

Название: Фильтрация звукового сигнала

|  |  |
| --- | --- |
| Фамилия И.О.: | Депрейс А.С. |
| 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.

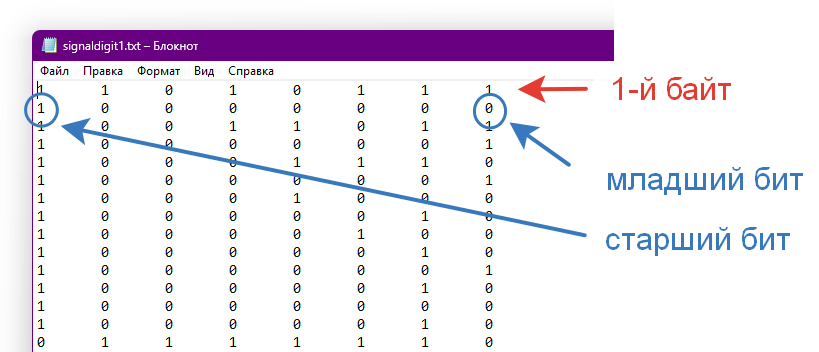


Рис.1. Структура данных в текстовом файле с сигналом

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”.

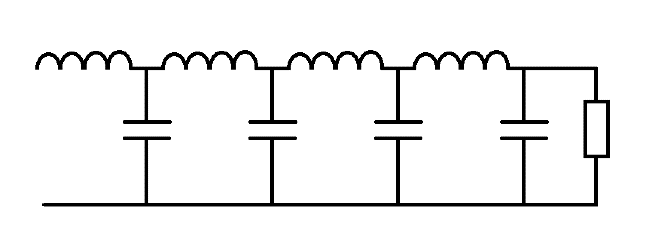
Пример содержания файла IDZ4.txt:

25

Вариант 24 Длительность сигнала 3.25с

Теоретические положения

Для отчистки сигнала от помех используются фильтры.

В данной работе используется фильтр Баттерворта.

АЧХ его передаточной функции имеет вид ступеньки, что позволяет достаточно точно обрезать высокие частоты шума, почти не ослабив основной сигнал. Схема фильтра представлена на рисунке.

**ПРИЛОЖЕНИЕ А**

**ПРОГРАММА foit\_4.py**

import struct

import wave

import scipy

import sympy

import numpy as np

import matplotlib.pyplot as plt

file = open('signaldigit24.txt', 'r')

signal = []

for num in file:

signal.append(int(num.replace("\t", ""),2))

duration = 3.25

deltaT = duration / len(signal)

timeline = [i \* deltaT for i in range(len(signal))]

def create\_wav\_file(signal, filename, end = False):

with wave.open(filename, mode="wb") as audio:

audio.setnchannels(1)

audio.setsampwidth(1)

audio.setframerate((4 if end else 2) \* len(signal) / duration)

for s in signal:

packed\_value = struct.pack(("f" if end else"h"),s)

audio.writeframes(packed\_value)

# create\_wav\_file(signal, "raw\_signal.wav")

plt.plot(timeline, signal)

plt.show()

coefficient = 1.6

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

F = np.fft.fft(signal)

Fabs = list(map(abs, F[1:]))

freqs = [(i + 1) \* 2 \* np.pi/(duration) for i in range(len(Fabs))]

Habs = [abs(H(freqs[i])) for i in range(len(freqs))]

plt.plot(freqs, Habs)

plt.show()

plt.plot(freqs, Fabs)

plt.show()

Fout = []

df = 1/duration

for i in range(len(F) - 1):

Fouti = F[i + 1] \* abs(H(freqs[i]))

Fout.append(Fouti)

plt.plot(freqs, list(map(abs,Fout)))

plt.show()

Sout = np.fft.ifft(Fout)

Filtered = []

for i in range(len(Sout)):

Filtered.append(Sout[i].real)

plt.plot(timeline[1:], Filtered)

plt.show()

# create\_wav\_file(Filtered, "filtered\_signal.wav", True)