Санкт-Петербургский Государственный Электротехнический Университет "ЛЭТИ"

кафедра физики

Задание №4 по дисциплине

"Физические основы информационных технологий"

Название: Фильтрация звукового сигнала

|  |  |
| --- | --- |
| Фамилия И.О.: | Коренев Д.А. |
| 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.

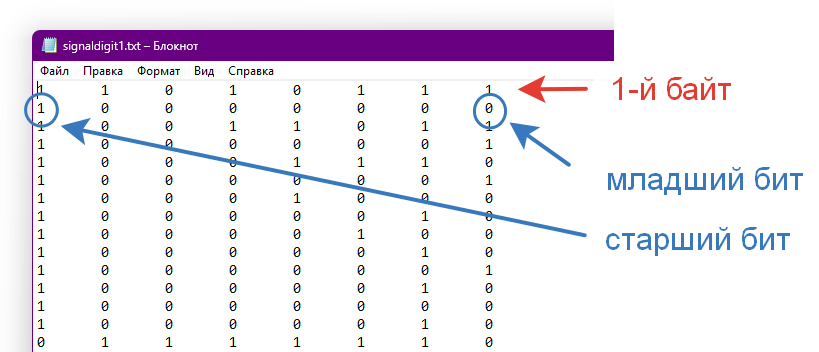


Рис.1. Структура данных в текстовом файле с сигналом

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 19

Длительность сигнала 3.75с

Файл с сигналом: signaldigit19.txt

Теоретические положения

Для отчистки сигнала от помех используются фильтры высоких и низких частот.

В данной работе используется фильтр низких частот RC.

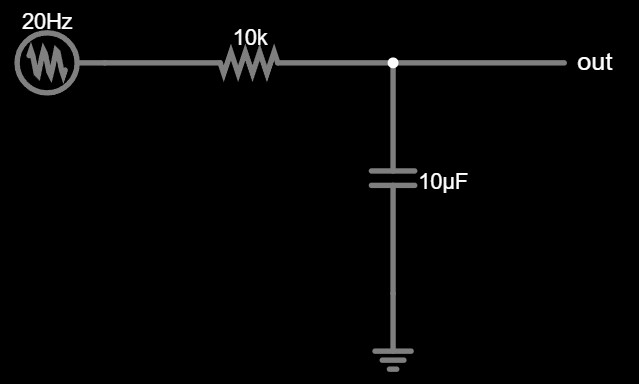


Рисунок 1. ФНЧ RC

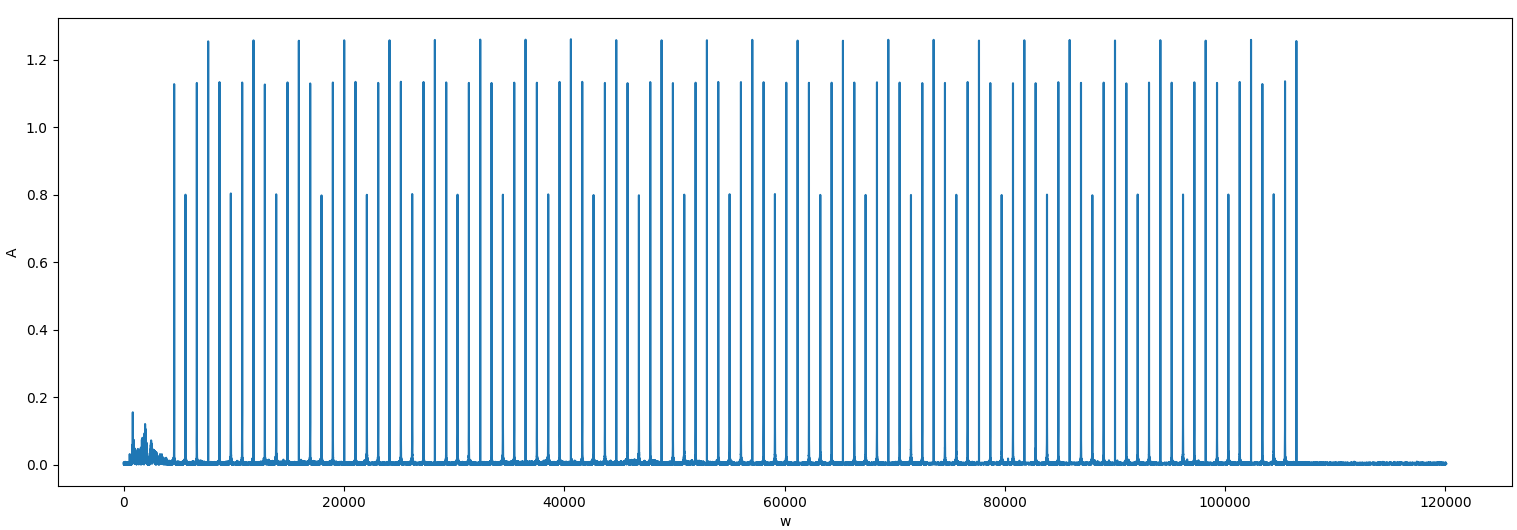


Рисунок 2. Спектр оригинального сигнала

После фильтрации сигнала получаем уже другой спектр.

Изображение выглядит как снимок экрана, текст, линия, Прямоугольник

Автоматически созданное описание

Рисунок 3. Спектр отфильтрованного сигнала

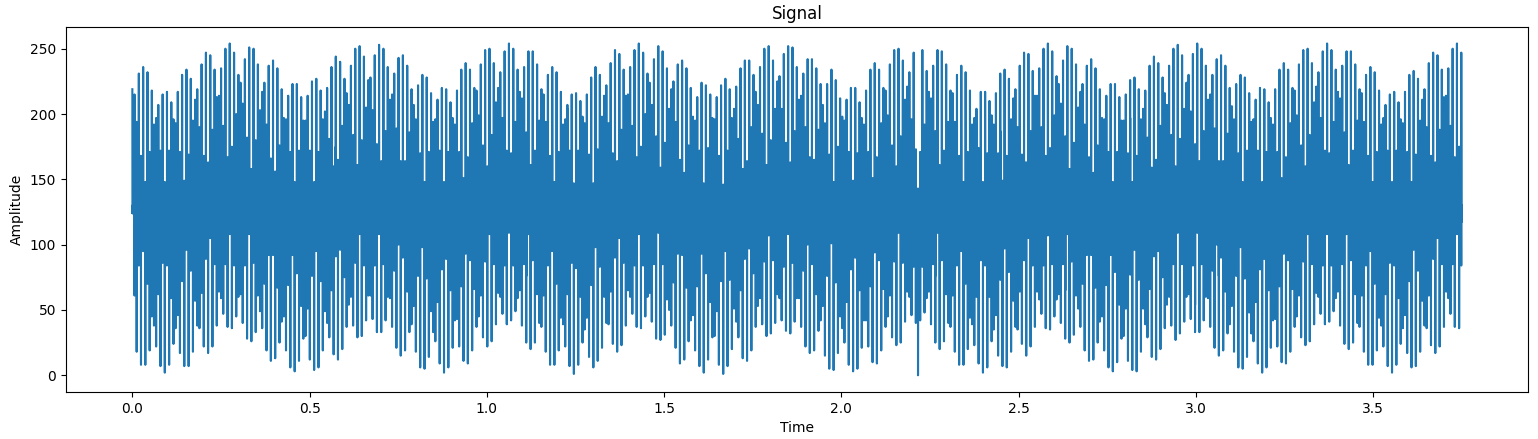


Рисунок 4 Оригинальный сигнал

Изображение выглядит как линия, График, снимок экрана, диаграмма

Автоматически созданное описание

Рисунок 5 Измененный сигнал

Изображение выглядит как текст, снимок экрана, дисплей, Прямоугольник

Автоматически созданное описание

Рисунок 6 АЧХ

**ПРИЛОЖЕНИЕ А**

**ПРОГРАММА IDZ4.py**

import numpy as np

import matplotlib.pyplot as plt

from scipy.io.wavfile import write

filename = "signaldigit19.txt"

t = 3.75

signal\_length = None

dt = None

F = None

H\_array = []

w = []

C = 0.00001

R = 100000

U = 5

def dt():

return t / signal\_length

def ZC(w, C):

if 1j \* w \* C == 0:

return np.inf

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

def R\_in(w):

if ZC(w, C) == np.inf:

return np.inf

return R + ZC(w, C)

def I\_in(w, U\_in):

return U\_in / R\_in(w)

def H(w, U\_in):

U\_out = I\_in(w, U\_in) \* ZC(w, C)

return U\_out / U\_in

def bytes\_to\_decimal(filename):

discret\_signal = np.loadtxt(filename, dtype=int)

return [int(''.join(map(str, row)), 2) for row in discret\_signal]

def scale(signal):

return np.float32(np.real(signal) / np.max(np.abs(np.real(signal))) \* 1.0)

def write\_audio(filename, signal):

write(f'{filename}.wav', 44100, (signal \* 32767).astype(np.int16))

def show\_signal(signal):

time\_axis = np.arange(0, t, dt)[:len(signal)]

plt.plot(time\_axis, signal)

plt.xlabel('Time')

plt.ylabel('Amplitude')

plt.title('Signal')

plt.show()

def show\_input\_spectre():

a = [np.abs(i) \* 2 / signal\_length for i in F[1:]]

w = [i \* 2 \* np.pi / t for i in range(signal\_length)]

plt.plot(w[:int(len(w) / 2)], a[:int(len(a) / 2)])

plt.xlabel('w')

plt.ylabel('A')

plt.show()

def show\_afc():

plt.plot(w[:450], H\_array[:450])

plt.xlabel('Hz')

plt.ylabel('H')

plt.show()

def show\_output\_spectre():

F\_new = []

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

F\_new.append(F[i + 1] \* H\_array[i + 1])

A = [np.abs(i) \* 2 / signal\_length for i in F\_new]

plt.plot(w[:int(len(w) / 2)], A[:int(len(A) / 2)])

plt.xlabel('w')

plt.ylabel('A')

plt.show()

def get\_output\_signal():

F\_new = []

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

F\_new.append(F[i + 1] \* H\_array[i + 1])

return np.fft.ifft(F\_new)

if \_\_name\_\_ == '\_\_main\_\_':

analog\_signal = bytes\_to\_decimal(filename)

signal\_length = len(analog\_signal)

dt = dt()

scaled\_signal = scale(analog\_signal)

write\_audio("original", scaled\_signal)

show\_signal(analog\_signal)

F = np.fft.fft(analog\_signal)

w = [i \* 2 \* np.pi / t for i in range(signal\_length)]

H\_array = [np.abs(H(w\_elem, U)) for w\_elem in w]

show\_input\_spectre()

show\_afc()

show\_output\_spectre()

signal\_out = get\_output\_signal()

show\_signal(signal\_out)

signal\_out\_scale = scale(signal\_out)

write\_audio("changed\_signal", signal\_out\_scale)