Министерство образования Республики Беларусь

Учреждение образования
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОИЭЛЕКТРОНИКИ»

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

Лабораторная работа № 4
на тему
«Проектирование цифровых фильтров»

Выполнили: Басько А. С.
Какадей С. В.

Проверил: Третьяков А. Г.

МИНСК 2021

Задание:

1. Для сигнала, заданного в лабораторной работе №1, реализовать КИХ и БИХ фильтр.

1.1 На вход фильтра, применяемых для сглаживания, подавать сигнал искаженный аддитивной шумовой помехой.

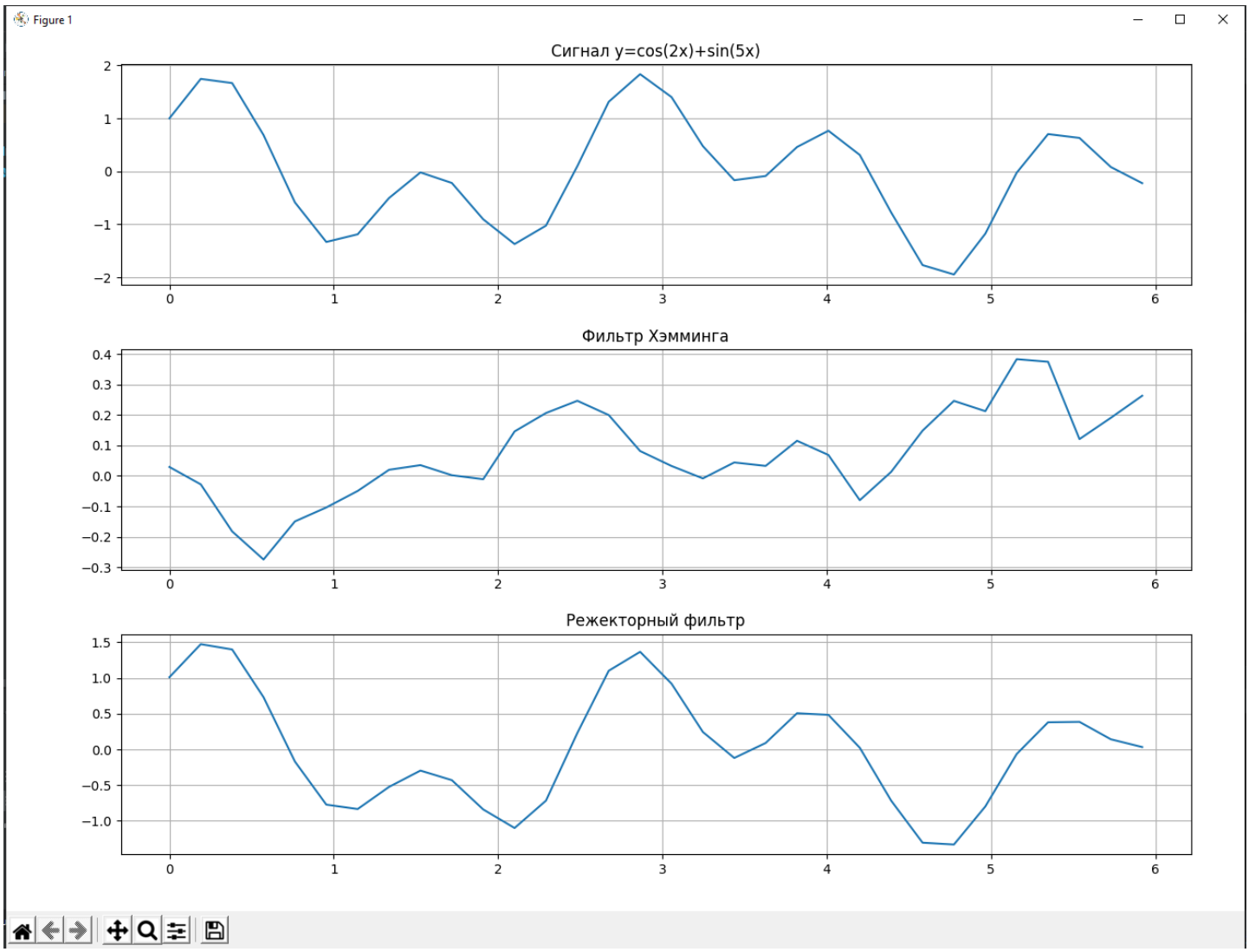
1.2 На вход фильтра, применяемых для частотной селекции, входной сигнал необходима представить в частотной области.

2. получить график заданной функции, график по результатам КИХ фильтра, график по результатам БИХ фильтра;

Исходные данные:

1. Номер варианта: 3.
2. Функция сигнала: y=cos(2x) + sin(5x).
3. N: 32.
4. Фильтр №1: ВЧ оконный фильтр. Окно Хэмминга.
5. Фильтр №2: Режекторный узкополосный фильтр

Результат работы программы:



Листинг кода:

(main.py)

```
import matplotlib.pyplot as plt
import scipy

from scipy.signal import filtfilt, iirnotch
import numpy as np


N = 32

def _fun(x):
    y = np.cos(2*x) + np.sin(5*x)
    return y



def Hamming_filter(signalIn):

    Fs = 2.8 / (np.pi)
    T  = 1/Fs
    t  = np.arange(0,(1024*T),T)




    F_low  = 25000 #  25kHz
    F_high = 1000000  #  1MHz
    ip = np.sin(2*np.pi*F_low*t) + np.sin(2*np.pi*F_high*t)

    op = [0]*len(ip)


    Fc = 900000
    Ft = Fc/Fs
    Length       = 101
    M            = Length - 1
    Weight       = [0] * N
    for n in range(0, N):
        if( n != (M/2) ):
            Weight[n] = ( -np.sin(2*np.pi*Ft*(n-(M/2))) / (np.pi*(n-(M/2))) )
        else:
            Weight[n] = ( 1-2*Ft )



    for n in range(len(Weight), len(ip)):
      y = 0
      for i in range(0, len(Weight)):
          op[i] += Weight[i]*ip[n-i]

      filter=op/np.sum(op)
    signalOut=[0]*len(signalIn)


    for i in range(len(signalIn)):
        for j in range(len(signalIn) - 1):
            if i >= j:
               signalOut[i] += filter[j] * signalIn[i - j]
    return signalOut

def Reject_filter(signal):
    for i in np.arange(50,500,50):
        fs = 1000.0  # Sample frequency (Hz)
        f0 = i  # Frequency to be removed from signal (Hz)
```

```
        w0 = f0 / (fs / 2)  # Normalized Frequency
        Q= 30
        b, a = iirnotch(w0, Q)
        signal = scipy.signal.filtfilt(b, a, signal)
    return(signal)

def _main():
    t = [0] * N
    step = 0.6 / (np.pi)

    signal = []

    for i in range(N):
        signal.append(_fun(t[i]))
        if i < N - 1:
            t[i + 1] = t[i] + step



    result1 = Hamming_filter(signal)
    f, axarr = plt.subplots(3, 1, figsize=(6, 6))
    plt.tight_layout()

    axarr[0].plot(t, signal)
    axarr[0].set_title('Сигнал y=cos(2x)+sin(5x)')
    axarr[0].grid(True)

    axarr[1].plot(t, result1)
    axarr[1].set_title('Фильтр Хэмминга')
    axarr[1].grid(True)



    signal2 = []

    for i in range(N):
        signal2.append(_fun(t[i]))
        if i < N - 1:
            t[i + 1] = t[i] + step

    result2 = Reject_filter(signal2)

    axarr[2].plot(t, result2)
    axarr[2].set_title('Режекторный фильтр')
    axarr[2].grid(True)

    plt.show()


if __name__ == "__main__":
    _main()
```