Министерство образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«Национальный исследовательский Томский политехнический Университет»



Центр цифровых

образовательных технологий

01.03.02 «Прикладная математика и информатика»

Лабораторная работа № 10

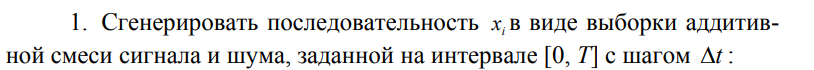
по дисциплине:

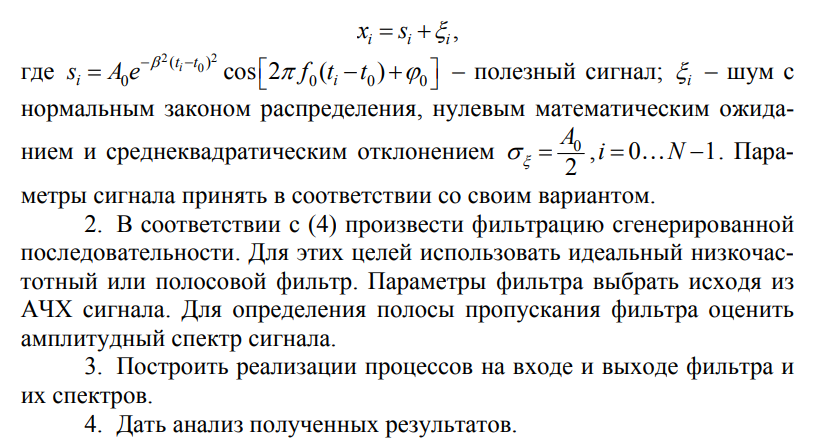
**Компьютерный анализ данных**

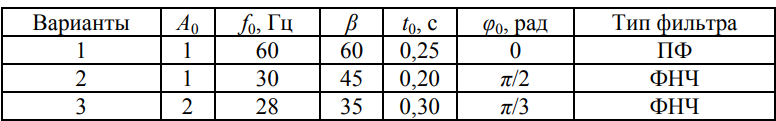
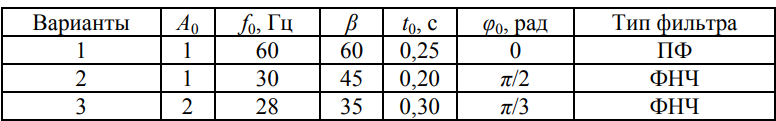
Вариант 3

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Исполнитель:** |  | | | | |
| студент группы | 0В02 |  | Редько Дмитрий Александрович |  |  |
|  |  |  |  |  |  |
| **Руководитель:** |  | | | | |
| преподаватель |  |  | Кочегуров Александр Иванович |  |  |
|  |  |  |  |  |  |

Томск — 2024



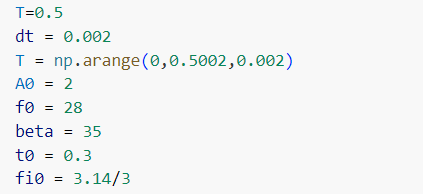


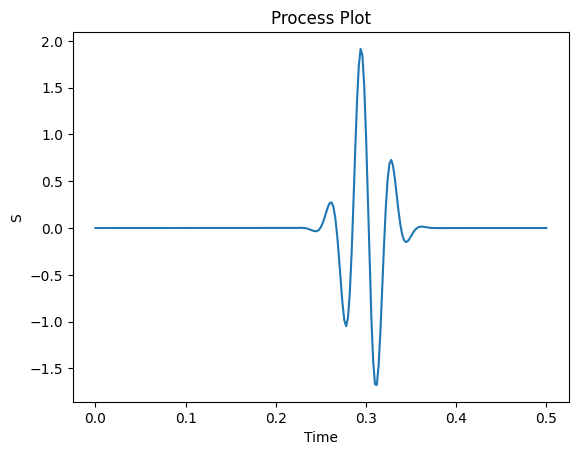
Ход работы:

1. По формуле

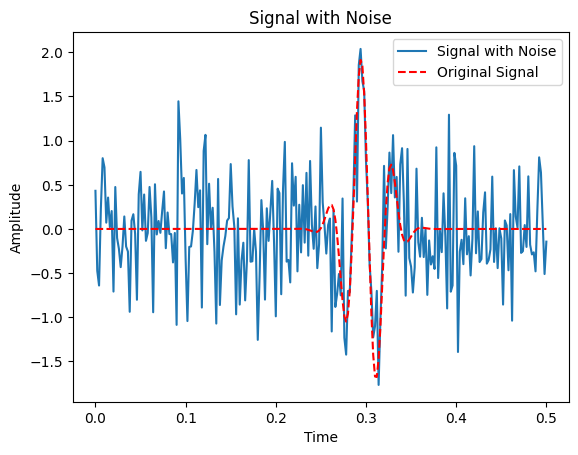
для параметров:



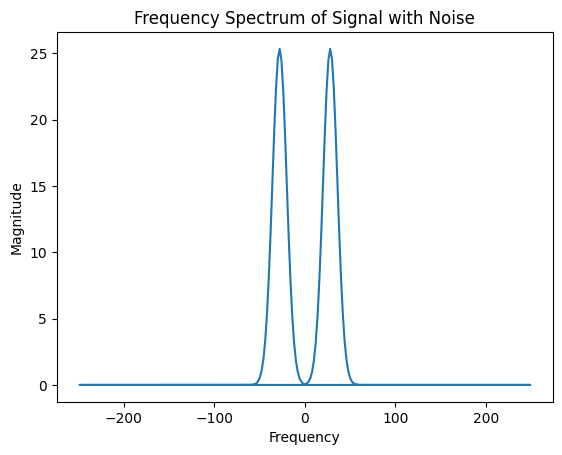
получим полезный сигнал:

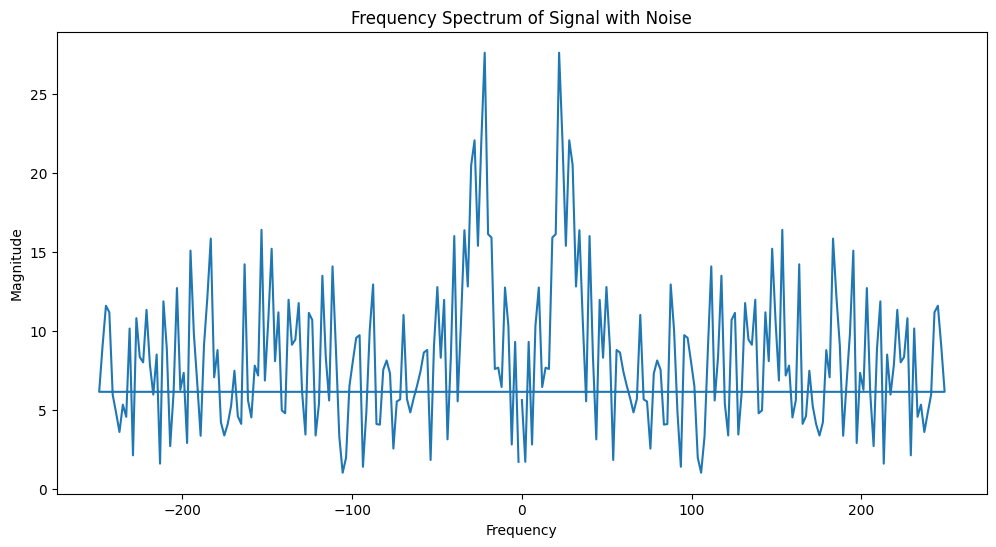


1. Добавим шум к этому сигналу

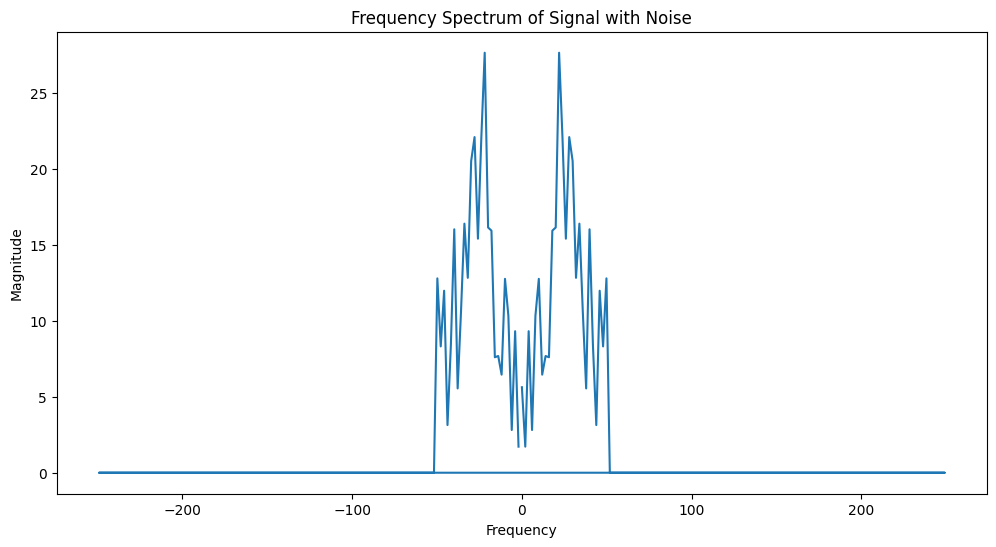


1. Построим АЧХ для чистого и зашумленного сигналов

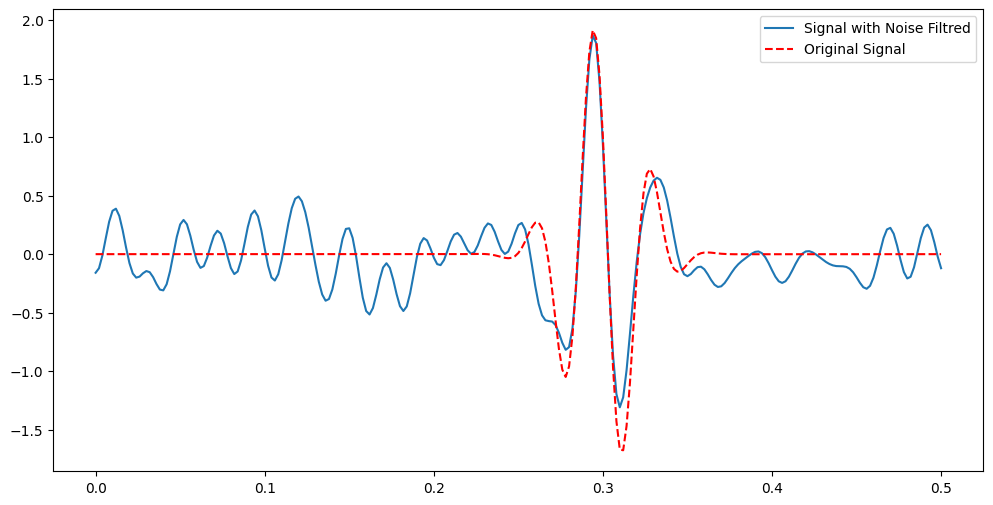




1. Проведем фильтрацию фильтром низких частот с пороговой частотой 50 и построим АЧХ после фильтрации



1. Выполним обратное преобразование Фурье:



**Вывод:**

Таким образом, получили отфильтрованный сигнал фильтром низких частот. Из полученного графика явно заметен оригинальный чистый сигнал.

**Приложение**

import numpy as np

import matplotlib.pyplot as plt

T=0.5

dt = 0.002

T = np.arange(0,0.5002,0.002)

A0 = 2

f0 = 28

beta = 35

t0 = 0.3

fi0 = 3.14/3

S = [A0\*np.exp(-beta\*\*2\*(ti - t0)\*\*2)\*np.cos(2\*3.14\*f0\*(ti-t0)+fi0) for ti in T]

plt.plot(T, S)

plt.xlabel('Time')

plt.ylabel('S')

plt.title('Process Plot')

plt.show()

X = [si + 0.5\*np.random.randn() for si in S]

plt.plot(T, X, label='Signal with Noise')

plt.plot(T, S, label='Original Signal', linestyle='dashed',color = 'red')

plt.xlabel('Time')

plt.ylabel('Amplitude')

plt.title('Signal with Noise')

plt.legend()

plt.show()

fft\_result\_S = np.fft.fft(S)

freq = np.fft.fftfreq(len(T), T[1] - T[0])

plt.plot(freq, np.abs(fft\_result\_S))

plt.xlabel('Frequency')

plt.ylabel('Magnitude')

plt.title('Frequency Spectrum of Signal with Noise')

plt.show()

fft\_result\_X = np.fft.fft(X)

freq = np.fft.fftfreq(len(T), T[1] - T[0])

plt.figure(figsize=(12, 6))

plt.plot(freq, np.abs(fft\_result\_X))

plt.xlabel('Frequency')

plt.ylabel('Magnitude')

plt.title('Frequency Spectrum of Signal with Noise')

plt.show()

def filter(x, y, f0):

    if(x> f0 or x < -f0):

        return 0

    else:

        return y

X\_filtred = [filter(freq[i],fft\_result\_X[i],50) for i in range(len(freq))]

X\_filtred

plt.figure(figsize=(12, 6))

plt.plot(freq, np.abs(X\_filtred))

plt.xlabel('Frequency')

plt.ylabel('Magnitude')

plt.title('Frequency Spectrum of Signal with Noise')

plt.show()

x\_filtred = np.fft.ifft(X\_filtred).real

plt.figure(figsize=(12, 6))

plt.plot(T, x\_filtred)

plt.show()

plt.figure(figsize=(12, 6))

plt.plot(T, x\_filtred, label='Signal with Noise Filtred')

plt.plot(T, S, label='Original Signal', linestyle='dashed',color = 'red')

plt.legend()

plt.show()