Министерство науки и высшего образования Российской Федерации
Калужский филиал
федерального государственного автономного
образовательного учреждения высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(КФ МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ***ИУК «Информатика и управление»***

КАФЕДРА ***ИУК4 «Программное обеспечение ЭВМ, информационные технологии»***

ЛАБОРАТОРНАЯ РАБОТА 5

ДИСЦИПЛИНА: «Цифровая обработка сигналов»

Выполнил: студент гр. ИУК4-72Б _____ (____Губин Е.В._____)
(Подпись) (Ф.И.О.)

Проверил: _____ (____Чурилин О.И___)
(Подпись) (Ф.И.О.)

Дата сдачи (защиты):

Результаты сдачи (защиты):

- Балльная оценка:
- Оценка:

Калуга, 2025

Цель: формирование практических навыков анализа спектра дискретных сигналов с помощью дискретного преобразования Фурье (ДПФ).

Задачи:
1) используя ДПФ построить АЧХ сигналов: заданного и отфильтрованного;
2) с помощью АЧХ проверить правильность процедуры фильтрации, при необходимости скорректировать параметры фильтра.

Формулировка задания (3 вариант):

1. Изучить краткий теоретический материал.
2. Произвести для соответствующих данных дискретные преобразования Фурье и построить спектр сигналов.
3. Проверить с помощью АЧХ спектра правильности процедуры фильтрации. При необходимости скорректировать параметры фильтра.
4. В одном графическом окне отобразить:
   - полный сигнал (1 2S S□ или1 2 3S S S□ □);
   - отфильтрованный сигнал;
   - спектр полного сигнала;
   - спектр полного сигнала и спектр отфильтрованного сигнала.

Результаты выполнения:

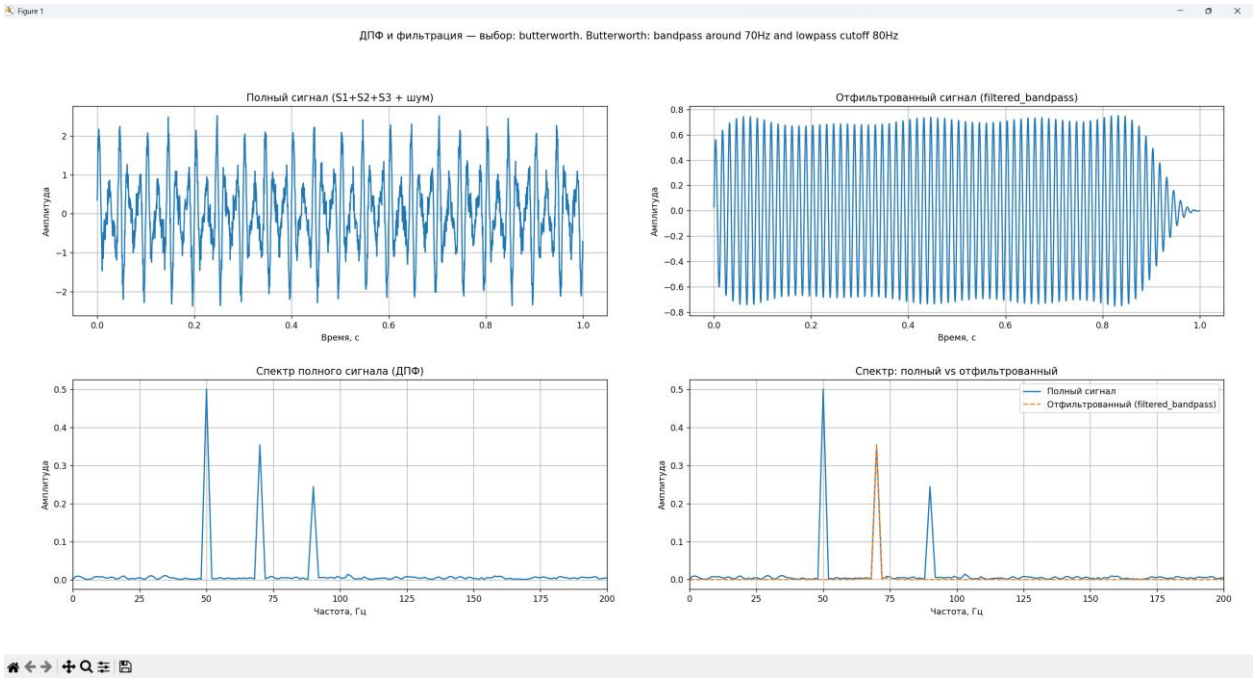

Рисунок 1 Баттерворт

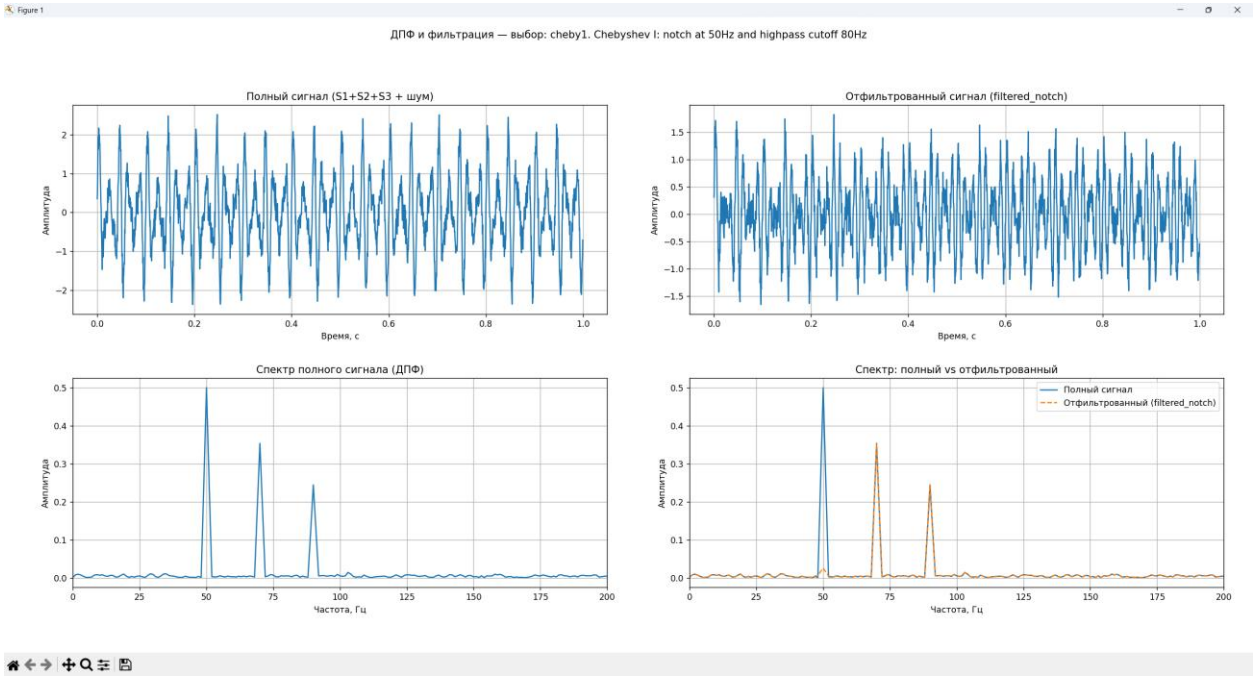

Рисунок 2 Чебышев 1-го рода

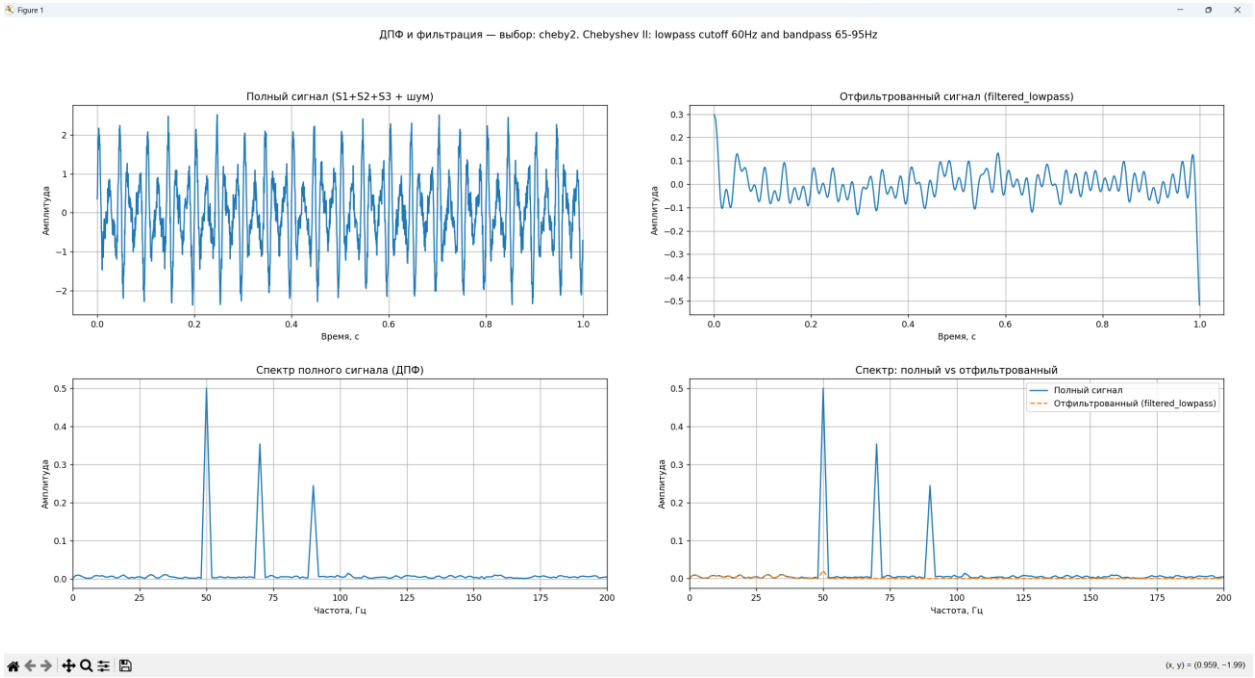

Рисунок 3 Чебышев 2-го рода

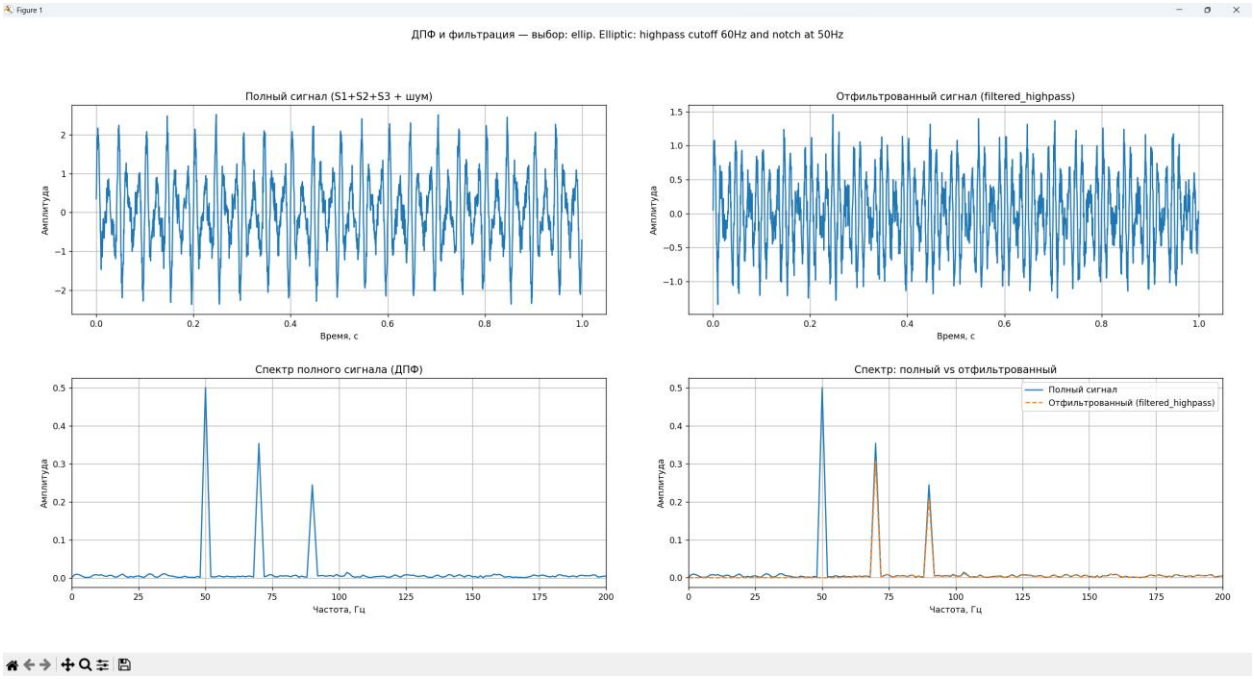

Рисунок 4 Эллиптический

Листинг программы:

```
import numpy as np
import matplotlib.pyplot as plt
from scipy import signal

fs = 2000
T = 1.0
t = np.arange(0, T, 1/fs)
f1, f2, f3 = 50, 70, 90
A1, A2, A3 = 1.0, 0.7, 0.5
noise_amp = 0.2

S1 = A1 * np.sin(2*np.pi*f1*t)
S2 = A2 * np.sin(2*np.pi*f2*t)
S3 = A3 * np.sin(2*np.pi*f3*t)
np.random.seed(0)
noise = noise_amp * np.random.randn(len(t))

full_signal = S1 + S2 + S3 + noise

def compute_spectrum(x, fs):
    N = len(x)
    X = np.fft.rfft(x * np.hanning(N))
    freqs = np.fft.rfftfreq(N, 1/fs)
    magnitude = np.abs(X) / (N/2)
    return freqs, magnitude

def notch(fs, f0, Q=30):
    w0 = f0/(fs/2)
    b, a = signal.iirnotch(w0, Q)
    return b, a

def bandpass_iir(ftype, fs, lowcut, highcut, order=4, rp=1, rs=40):
    nyq = fs/2
    low = lowcut/nyq
    high = highcut/nyq
```

```
    if ftype == 'butter':
        b, a = signal.butter(order, [low, high], btype='band')
    elif ftype == 'cheby1':
        b, a = signal.cheby1(order, rp, [low, high], btype='band')
    elif ftype == 'cheby2':
        b, a = signal.cheby2(order, rs, [low, high], btype='band')
    elif ftype == 'ellip':
        b, a = signal.ellip(order, rp, rs, [low, high], btype='band')
    else:
        raise ValueError("Unknown ftype")
    return b, a

def lowpass_iir(ftype, fs, cutoff, order=4, rp=1, rs=40):
    nyq = fs/2
    W = cutoff/nyq
    if ftype == 'butter':
        b, a = signal.butter(order, W, btype='low')
    elif ftype == 'cheby1':
        b, a = signal.cheby1(order, rp, W, btype='low')
    elif ftype == 'cheby2':
        b, a = signal.cheby2(order, rs, W, btype='low')
    elif ftype == 'ellip':
        b, a = signal.ellip(order, rp, rs, W, btype='low')
    else:
        raise ValueError("Unknown ftype")
    return b, a

def highpass_iir(ftype, fs, cutoff, order=4, rp=1, rs=40):
    nyq = fs/2
    W = cutoff/nyq
    if ftype == 'butter':
        b, a = signal.butter(order, W, btype='high')
    elif ftype == 'cheby1':
        b, a = signal.cheby1(order, rp, W, btype='high')
    elif ftype == 'cheby2':
        b, a = signal.cheby2(order, rs, W, btype='high')
    elif ftype == 'ellip':
        b, a = signal.ellip(order, rp, rs, W, btype='high')
    else:
        raise ValueError("Unknown ftype")
    return b, a

# Варианты: 'butterworth', 'cheby1', 'cheby2', 'ellip'
filter_choice = 'butterworth'

def apply_filter(choice, x, fs):
    if choice == 'butterworth':
        b_bp, a_bp = bandpass_iir('butter', fs, 65, 75, order=4)
        filtered_bp = signal.filtfilt(b_bp, a_bp, x)
        b_lp, a_lp = lowpass_iir('butter', fs, 80, order=4)
        filtered_lp = signal.filtfilt(b_lp, a_lp, x)
        return {'filtered_bandpass': filtered_bp, 'filtered_lowpass':
filtered_lp}, "Butterworth: bandpass around 70Hz and lowpass cutoff 80Hz"
    elif choice == 'cheby1':
        b_notch, a_notch = notch(fs, f1, Q=30)
        filtered_notch = signal.filtfilt(b_notch, a_notch, x)
        b_hp, a_hp = highpass_iir('cheby1', fs, 80, order=4, rp=1)
        filtered_hp = signal.filtfilt(b_hp, a_hp, x)
        return {'filtered_notch': filtered_notch, 'filtered_highpass':
filtered_hp}, "Chebyshev I: notch at 50Hz and highpass cutoff 80Hz"
    elif choice == 'cheby2':
        b_lp, a_lp = lowpass_iir('cheby2', fs, 60, order=6, rs=40)
        filtered_lp = signal.filtfilt(b_lp, a_lp, x)
        b_bp, a_bp = bandpass_iir('cheby2', fs, 65, 95, order=6, rs=40)
```

```
        filtered_bp = signal.filtfilt(b_bp, a_bp, x)
        return {'filtered_lowpass': filtered_lp, 'filtered_bandpass':
filtered_bp}, "Chebyshev II: lowpass cutoff 60Hz and bandpass 65-95Hz"
    elif choice == 'ellip':
        b_hp, a_hp = highpass_iir('ellip', fs, 60, order=6, rp=1, rs=60)
        filtered_hp = signal.filtfilt(b_hp, a_hp, x)
        b_notch, a_notch = notch(fs, f1, Q=30)
        filtered_notch = signal.filtfilt(b_notch, a_notch, x)
        return {'filtered_highpass': filtered_hp, 'filtered_notch':
filtered_notch}, "Elliptic: highpass cutoff 60Hz and notch at 50Hz"
    else:
        raise ValueError("Unknown filter choice")

filtered_dict, description = apply_filter(filter_choice, full_signal, fs)

plt.figure(figsize=(12, 8))

ax1 = plt.subplot(2,2,1)
ax1.plot(t, full_signal)
ax1.set_title("Полный сигнал (S1+S2+S3 + шум)")
ax1.set_xlabel("Время, с")
ax1.set_ylabel("Амплитуда")
ax1.grid(True)

first_key = list(filtered_dict.keys())[0]
ax2 = plt.subplot(2,2,2)
ax2.plot(t, filtered_dict[first_key])
ax2.set_title(f"Отфильтрованный сигнал ({first_key})")
ax2.set_xlabel("Время, с")
ax2.set_ylabel("Амплитуда")
ax2.grid(True)

ax3 = plt.subplot(2,2,3)
freqs_full, mag_full = compute_spectrum(full_signal, fs)
ax3.plot(freqs_full, mag_full)
ax3.set_xlim(0, 200)
ax3.set_title("Спектр полного сигнала (ДПФ)")
ax3.set_xlabel("Частота, Гц")
ax3.set_ylabel("Амплитуда")
ax3.grid(True)

ax4 = plt.subplot(2,2,4)
freqs_filt, mag_filt = compute_spectrum(filtered_dict[first_key], fs)
ax4.plot(freqs_full, mag_full, label='Полный сигнал')
ax4.plot(freqs_filt, mag_filt, label=f'Отфильтрованный ({first_key})',
linestyle='--')
ax4.set_xlim(0, 200)
ax4.set_title("Спектр: полный vs отфильтрованный")
ax4.set_xlabel("Частота, Гц")
ax4.set_ylabel("Амплитуда")
ax4.legend()
ax4.grid(True)

plt.suptitle(f"ДПФ и фильтрация — выбор: {filter_choice}. {description}",
fontsize=12)
plt.tight_layout(rect=[0, 0.03, 1, 0.95])
plt.show()

print("Выбранный фильтр:", filter_choice)
print("Описание:", description)
print("Доступные отфильтрованные сигналы:", list(filtered_dict.keys()))
```

Вывод: в ходе выполнения лабораторной работы были получены практически навыки по анализу спектра дискретных сигналов с помщью дискретного преобразования Фурье.