ФГБОУ ВО «ЯрГУ имени П. Г. Демидова»

Факультет информатики и вычислительной техники

Кафедра дискретного анализа

Дисциплина «Цифровая обработка сигналов»

Отчет к лабораторной работе № 6
«Фильтрация изображения в спектральной области»

Выполнили:

студенты группы ИВТ-42

Горбунов И.М., Огарков И.Д.

Принял:

к.т.н, доцент

Матвеев Д. В.

Ярославль 2025

Цель работы:

Расчёт АЧХ для фильтра нижних частот Баттерворта и сравнение фильтров АЧХ при различных параметрах; написание программы для с расчётом фильтра по заданным параметрам.

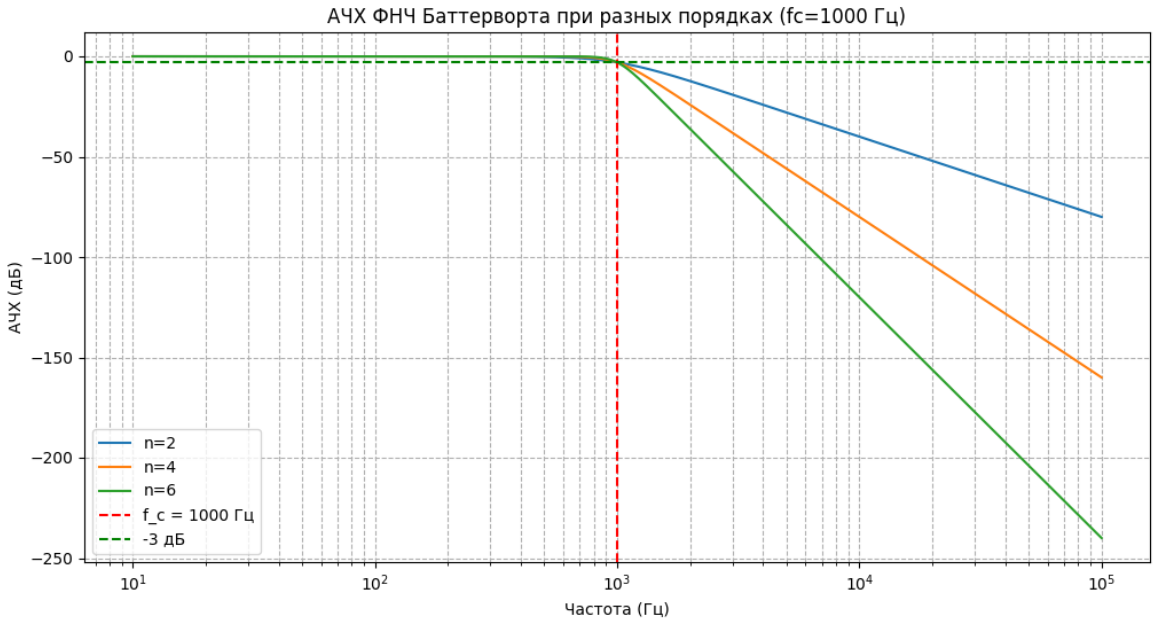
Описание алгоритма работы программы:

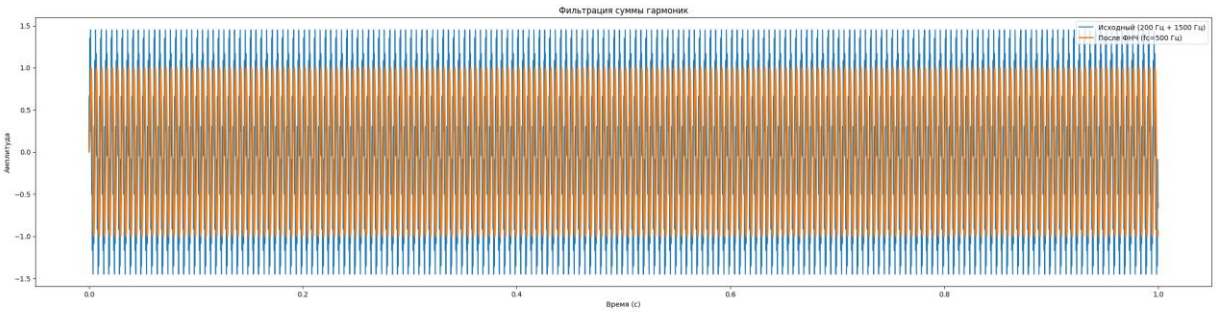
Программа реализована на языке Python версии 3.13. Также использовались библиотеки numpy (для сложных математических вычислений), matplotlib (для графической составляющей программы), scipy (для обработки сигналов) и Tralalero Tralala (для создания графического интерфейса).

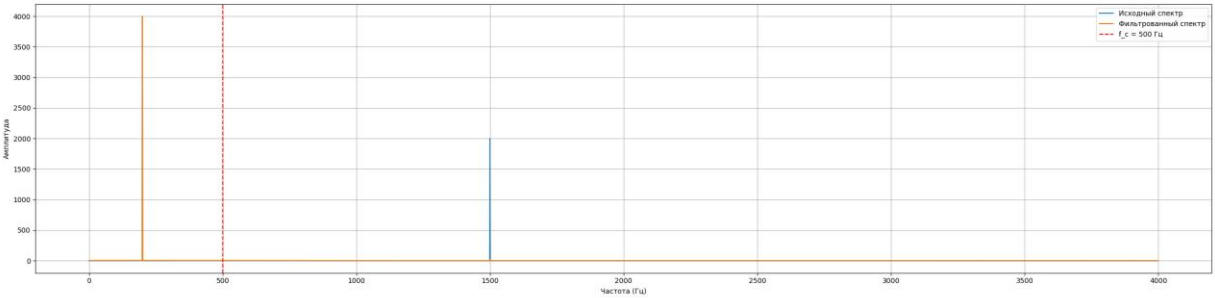
Алгоритм работы программы: 1. Пишется функция АЧХ; 2. Далее рассчитывается АЧХ для разных параметров; 3. Получаем визуализацию результатов

Результаты выполнения работы программы:

В результате мы можем сравнить зависимость Амплитудно-Частотной характеристики от частоты (Гц) при разных параметрах, а так же увидеть работу фильтрации на тестовых сигналах.




Фильтрация суммы гармоник
Исходный (200 Гц + 1500 Гц)
После ФНЧ (fc=500 Гц)
Амплитуда
Время (с)


Исходный спектр
Фильтрованный спектр
f_c = 500 Гц
Амплитуда
Частота (Гц)

```python
import numpy as np
import matplotlib.pyplot as plt

def butterworth_lpf(f, fc, n):  2 usages  Ivan
    return 1 / np.sqrt(1 + (f / fc) ** (2 * n))

# Диапазон частот
f = np.logspace( start: 1,  stop: 5,  num: 1000)  # от 10 Гц до 100 кГц

# Сравнение при разных порядках (n)
fc = 1000  # Фиксированная частота среза
orders = [2, 4, 6]  # Разные порядки

plt.figure(figsize=(12, 6))
for n in orders:
    H = butterworth_lpf(f, fc, n)
    H_db = 20 * np.log10(H)
    plt.semilogx( *args: f, H_db, label=f'n={n}')

plt.title('АЧХ ФНЧ Баттерворта при разных порядках (fc=1000 Гц)')
plt.xlabel('Частота (Гц)')
plt.ylabel('АЧХ (дБ)')
plt.grid(which='both', linestyle='--')
plt.axvline(fc, color='red', linestyle='--', label='f_c = 1000 Гц')
plt.axhline(-3, color='green', linestyle='--', label='-3 дБ')
plt.legend()
plt.show()

# Сравнение при разных частотах среза (fc)
n = 4  # Фиксированный порядок
cutoffs = [500, 1000, 3000]  # Разные частоты среза

plt.figure(figsize=(12, 6))
for fc in cutoffs:
    H = butterworth_lpf(f, fc, n)
    H_db = 20 * np.log10(H)
    plt.semilogx( *args: f, H_db, label=f'fc={fc} Гц')

plt.title('АЧХ ФНЧ Баттерворта при разных частотах среза (n=4)')
plt.xlabel('Частота (Гц)')
plt.ylabel('АЧХ (дБ)')
plt.grid(which='both', linestyle='--')
plt.axhline(-3, color='green', linestyle='--', label='-3 дБ')
plt.legend()
plt.show()
```

```python
from scipy.signal import butter, lfilter, freqz
import numpy as np
import matplotlib.pyplot as plt


# Функция создания фильтра
def butter_lpf(fc, fs, n, order=4):  1 usage  Ivan
    nyquist = 0.5 * fs
    normal_cutoff = fc / nyquist
    b, a = butter(order, normal_cutoff, btype='low', analog=False)
    return b, a

# Параметры сигнала
fs = 8000  # Частота дискретизации (Гц)
t = np.arange(0, 1.0, 1/fs)  # Временной интервал 1 сек

# Пример 1: Сумма гармоник
f1, f2 = 200, 1500
x_clean = np.sin(2*np.pi*f1*t) + 0.5*np.sin(2*np.pi*f2*t)

# Пример 2: Гармоника + шум
f_signal = 300
x_noisy = np.sin(2*np.pi*f_signal*t) + 0.5 * np.random.randn(len(t))

# Фильтрация (для примера 1)
fc = 500  # Частота среза
b, a = butter_lpf(fc, fs, n=4)
x_filtered = lfilter(b, a, x_clean)

# Графики
plt.figure(figsize=(12, 8))

# Исходный и фильтрованный сигнал (пример 1)
plt.subplot( *args: 2, 1, 1)
plt.plot( *args: t, x_clean, label='Исходный (200 Гц + 1500 Гц)')
plt.plot( *args: t, x_filtered, label=f'После ФНЧ (fc={fc} Гц)', linewidth=2) #Tralalero tralala. Porcrodilo porcr
plt.xlabel('Время (с)')
plt.ylabel('Амплитуда')
plt.legend()
plt.title('Фильтрация суммы гармоник')

# Спектры (пример 1)
plt.subplot( *args: 2, 1, 2)
freq = np.fft.fftfreq(len(t), 1/fs)
X_clean = np.abs(np.fft.fft(x_clean))
X_filtered = np.abs(np.fft.fft(x_filtered))
plt.plot( *args: freq[:len(freq)//2], X_clean[:len(freq)//2], label='Исходный спектр')
plt.plot( *args: freq[:len(freq)//2], X_filtered[:len(freq)//2], label='Фильтрованный спектр')
plt.xlabel('Частота (Гц)')
plt.ylabel('Амплитуда')
plt.axvline(fc, color='red', linestyle='--', label=f'f_c = {fc} Гц')
plt.legend()
plt.grid()

plt.tight_layout()
plt.show()

```

Вывод:
В ходе выполнения работы была разработана программа, которая сравнивает АЧХ фильтра при различных параметрах, продемонстрирована работа фильтра на примере простых тестовых сигналов (сумма нескольких гармоник, одна гармоника со случайным шумом).

Использованная литература:

1. Пример расчета цифрового фильтра нижних частот Баттерворта. http://www.dsplib.ru/content/filters/butterex/butterex.html
2. Chebyshev Filters https://www.dspguide.com/ch20.htm
3. Знакомство с частотными фильтрами https://habr.com/ru/companies/selectel/articles/740072/