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 1

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 разложения сигналов различного вида в ряд Фурье и моделирование сигналов различной формы с заданными
параметрами.

Задачи:
1) Выполнить разложение сигналов в ряд Фурье. Разложению подлежат следующие сигналы: последовательность прямоугольных импульсов, меандр, пилообразный сигнал и последовательность треугольных импульсов.
2) Построить графики для промежуточных стадий суммирования.
Для каждого варианта и каждого вида сигнала заданы параметры:

- для последовательности прямоугольных импульсов – амплитуда, период повторения и длительность импульсов;
- для меандра, пилообразного сигнала и последовательности треугольных импульсов – амплитуда и период повторения импульсов;
- для всех видов сигналов задано число ненулевых гармоник

Вариант 7

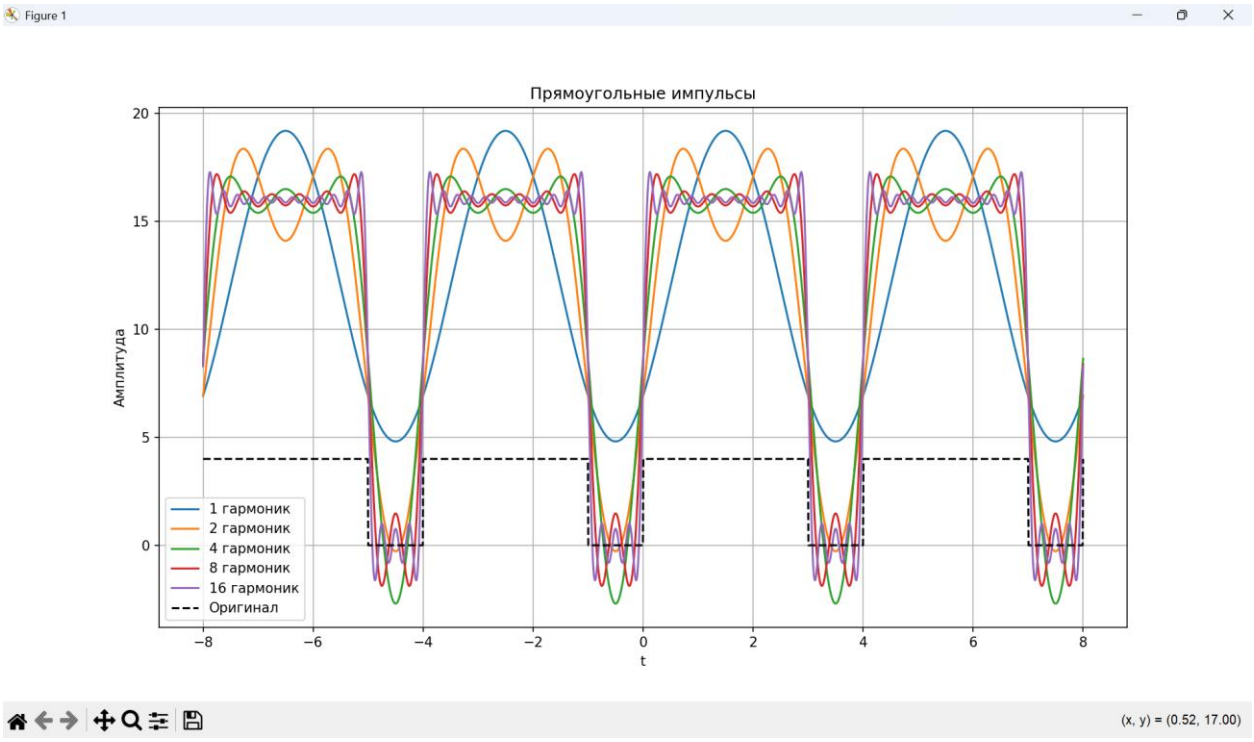
Амплитуда = 4
Период = 4
Длительность сигнала = 3
Число ненулевых гармоник = 16

Листинг пограммы:

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

A = 4
T = 4
D = 3
N = 16

t = np.linspace(-2*T, 2*T, 2000)
w0 = 2 * np.pi / T

def rect_pulse(t):
    return A * ((t % T) < D).astype(float)

def meander(t):
    return A * np.sign(np.sin(w0 * t))

def sawtooth(t):
    return (2*A/np.pi) * np.arctan(np.tan((w0*t)/2))

def triangle(t):
    return (2*A/np.pi) * np.arcsin(np.sin(w0*t))

def fourier_series(func, t, N):
    sums = []
    f = func(t)
    a0 = (2/T) * np.trapz(f * np.ones_like(t), t)
```

```
    partial = a0/2 * np.ones_like(t)
    sums.append(partial.copy())

    for n in range(1, N+1):
        cos_term = np.cos(n*w0*t)
        sin_term = np.sin(n*w0*t)

        an = (2/T) * np.trapz(f * cos_term, t)
        bn = (2/T) * np.trapz(f * sin_term, t)

        partial += an * cos_term + bn * sin_term
        sums.append(partial.copy())
    return sums

def plot_fourier(func, title):
    sums = fourier_series(func, t, N)
    plt.figure(figsize=(12, 8))
    for i in [1, 2, 4, 8, 16]:  # промежуточные стадии
        plt.plot(t, sums[i], label=f'{i} гармоник')
    plt.plot(t, func(t), 'k--', label='Оригинал')
    plt.title(title)
    plt.xlabel('t')
    plt.ylabel('Амплитуда')
    plt.legend()
    plt.grid()
    plt.show()

plot_fourier(rect_pulse, 'Прямоугольные импульсы')
plot_fourier(meander, 'Меандр')
plot_fourier(sawtooth, 'Пилообразный сигнал')
plot_fourier(triangle, 'Треугольный сигнал')
```

Результаты выполнения программы:



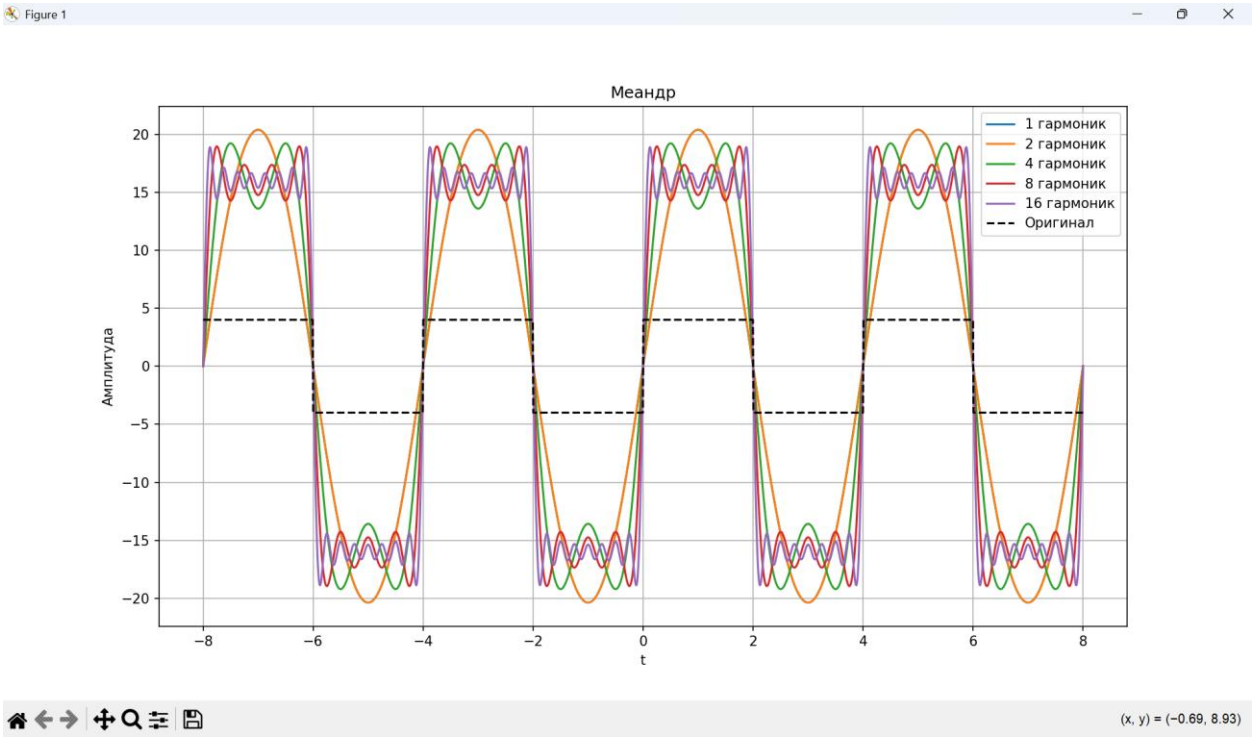


Figure 1
Меандр
1 гармоник
2 гармоник
4 гармоник
8 гармоник
16 гармоник
Оригинал
Амплитуда
t
(x, y) = (−0.69, 8.93)

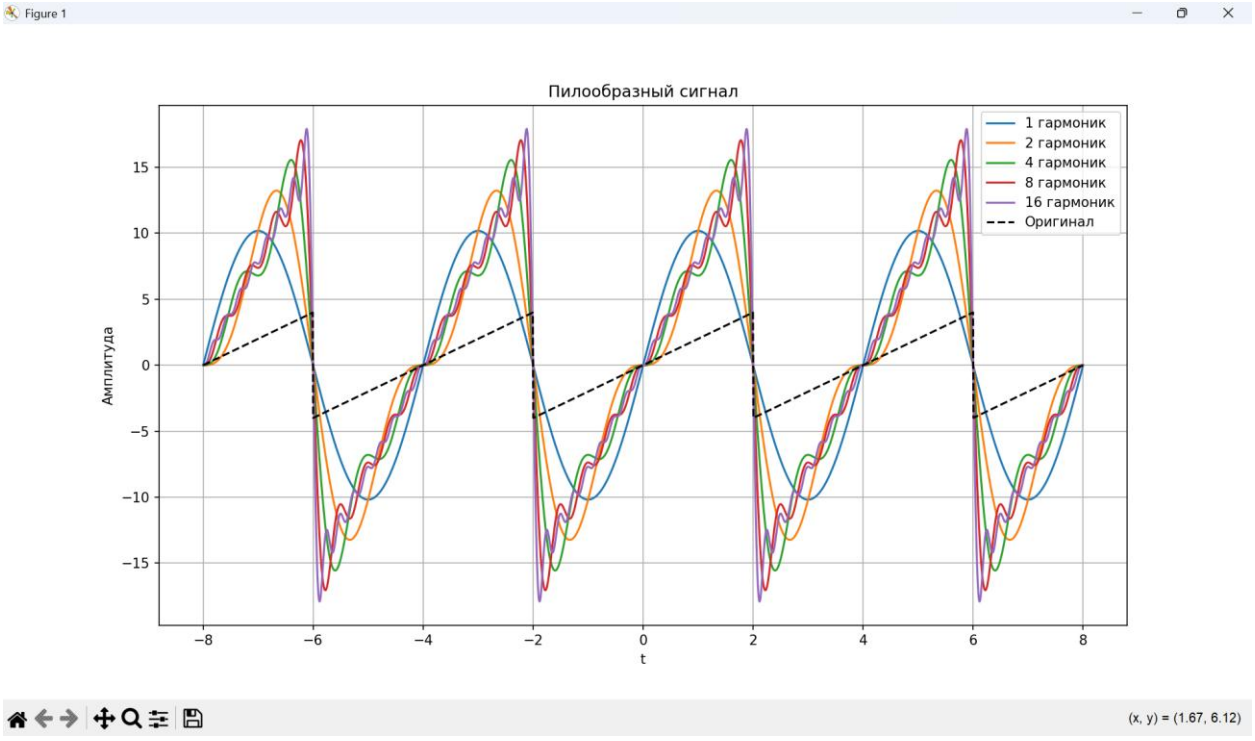
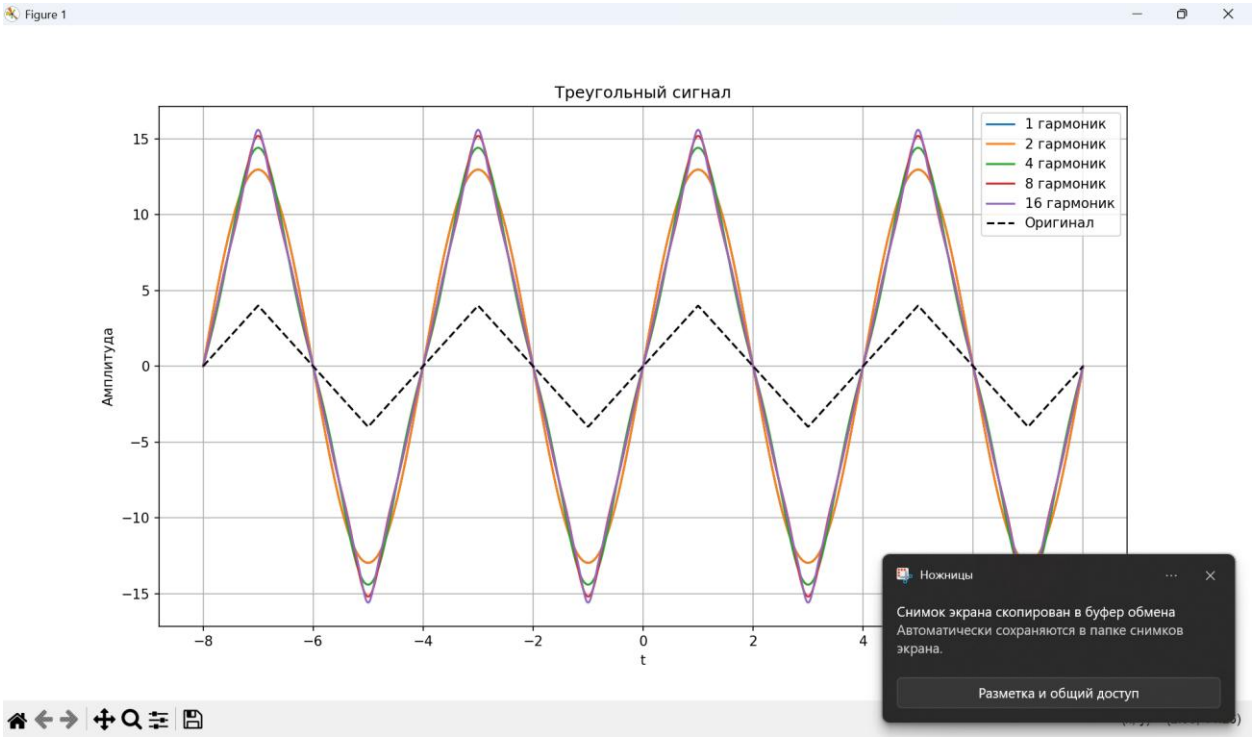


Figure 1
Пилообразный сигнал
1 гармоник
2 гармоник
4 гармоник
8 гармоник
16 гармоник
Оригинал
Амплитуда
t
(x, y) = (1.67, 6.12)



Вывод: в ходе лабораторной работы я получил навыки по раскладыванию сигнала в ряд Фурье.