|  |  |
| --- | --- |
|  | **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 технологи»

**ДОМАШНЯЯ РАБОТА**

**«Разложение сигналов»**

**по дисциплине: «Цифровая обработка сигналов»**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Выполнил: студент группы ИУК4-72Б | |  |  | И. А. Петроченков | |
|  | | (Подпись) |  | (И.О. Фамилия) | |
| Проверил(-а): | |  |  | О. И. Чурилин | |
|  | | (Подпись) |  | (И.О. Фамилия) | |
| Дата сдачи (защиты):  Результаты сдачи (защиты): | | | | |
|  | - Балльная оценка:  - Оценка: | | | |

Калуга, 2025

**Целью** выполнения домашней работы является формирование практических навыков разложения сигналов с использованием дискретного преобразования Фурье (ДПФ).

**Основными задачами** выполнения лабораторной работы являются:

1. С помощью ДПФ построить АЧХ гармонического сигнала;
2. Из спектра сигнала определить частоты основных гармоник сигнала и осуществить фильтрацию этих гармоник с помощью фильтров любого типа, подобрав соответствующие параметры фильтров;
3. В спектральной плоскости отобразить составляющую сигнала;
4. Над каждой выделенной составляющей сигнала произвести обратное ДПФ;
5. Построить графики полученных сигналов.

**Вариант задания:**

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **№ вар.** |  | | |  | | |  | | |  | | |
|  |  |  |  |  |  |  |  |  |  |  |  |
| 19 |  |  |  |  |  |  |  |  |  |  |  |  |

**Порядок выполнения работы:**

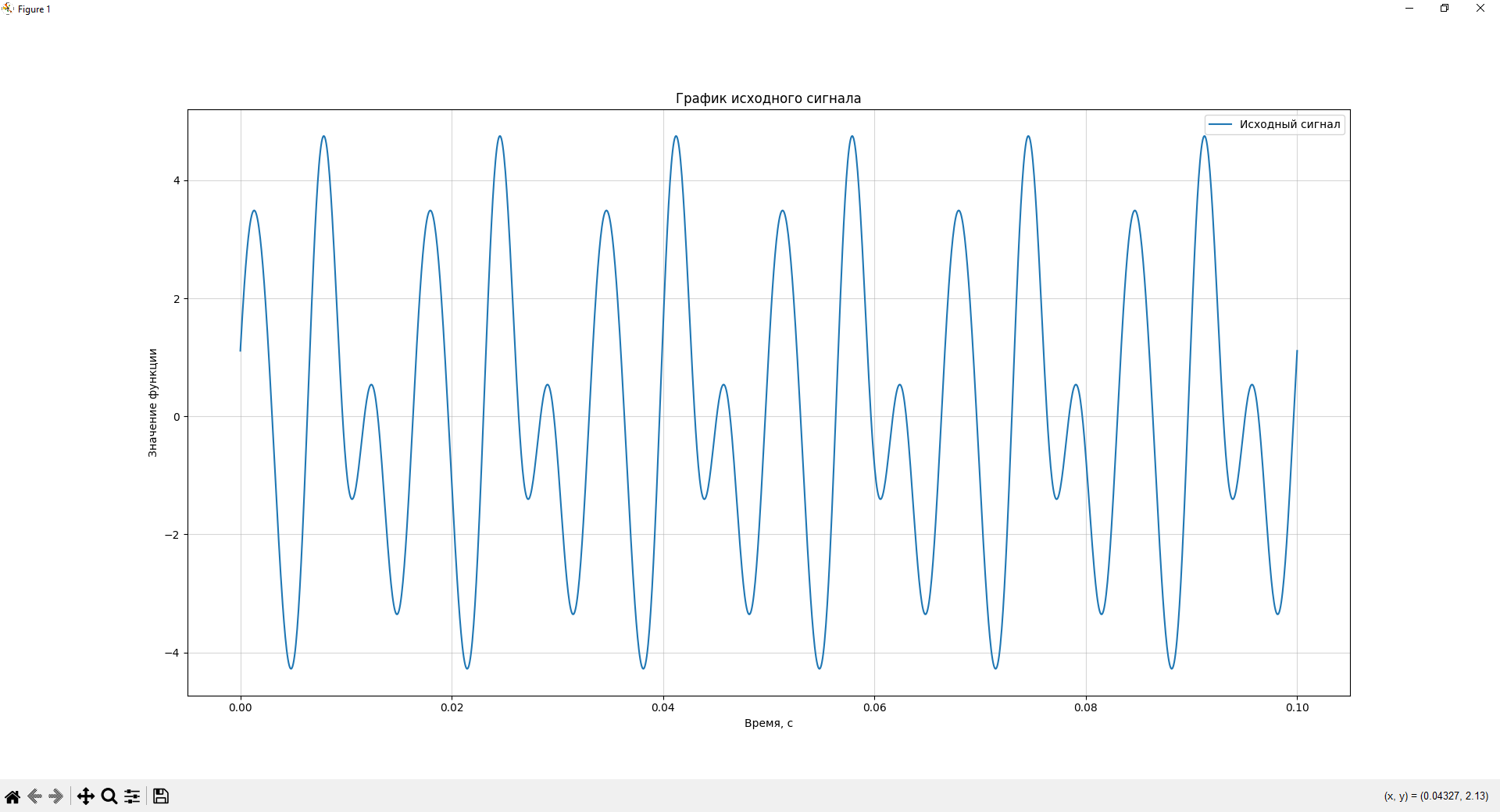


Рис. 1 График исходного сигнала

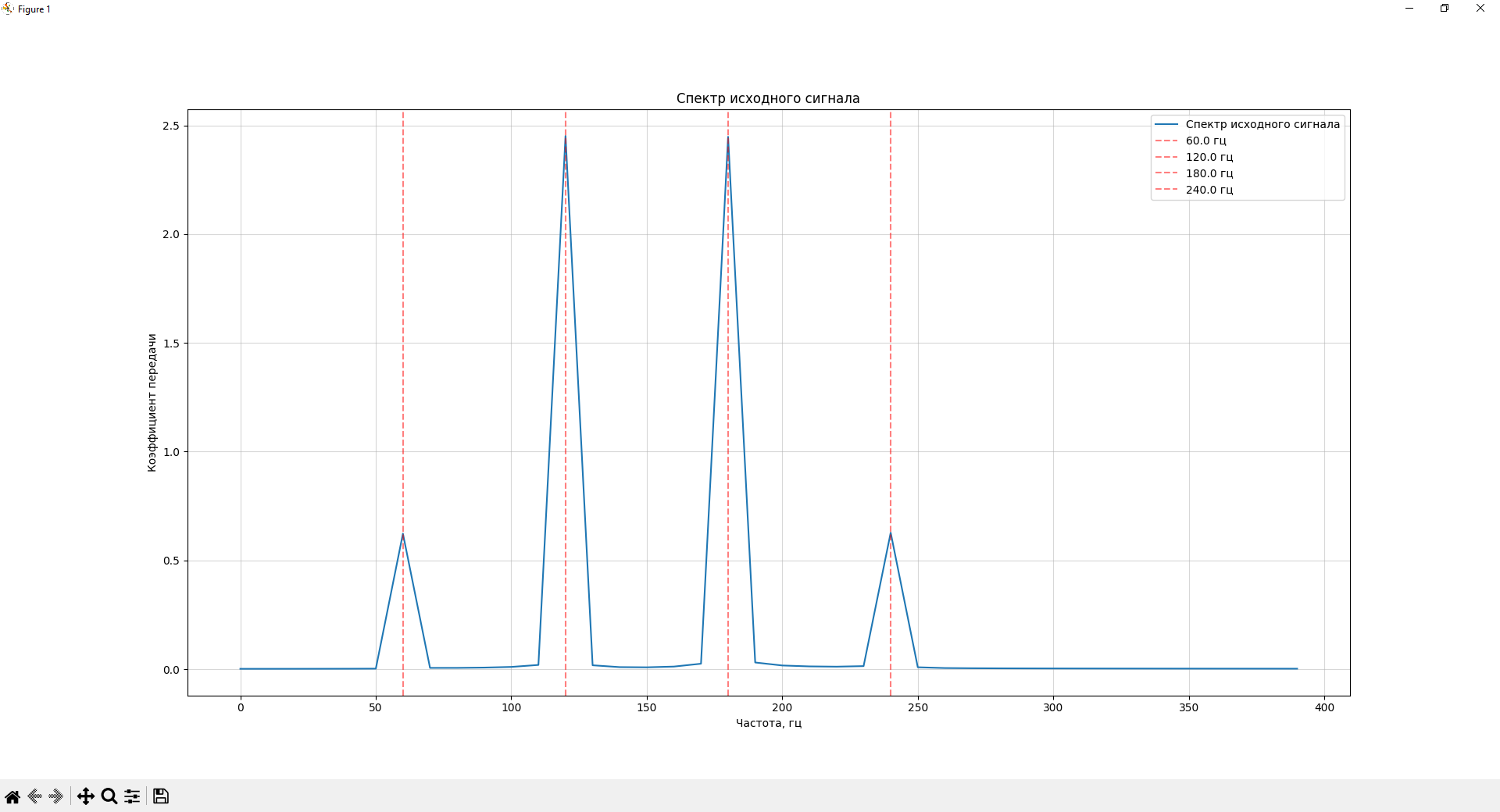


Рис. 2 Результаты анализа спектра исходного сигнала

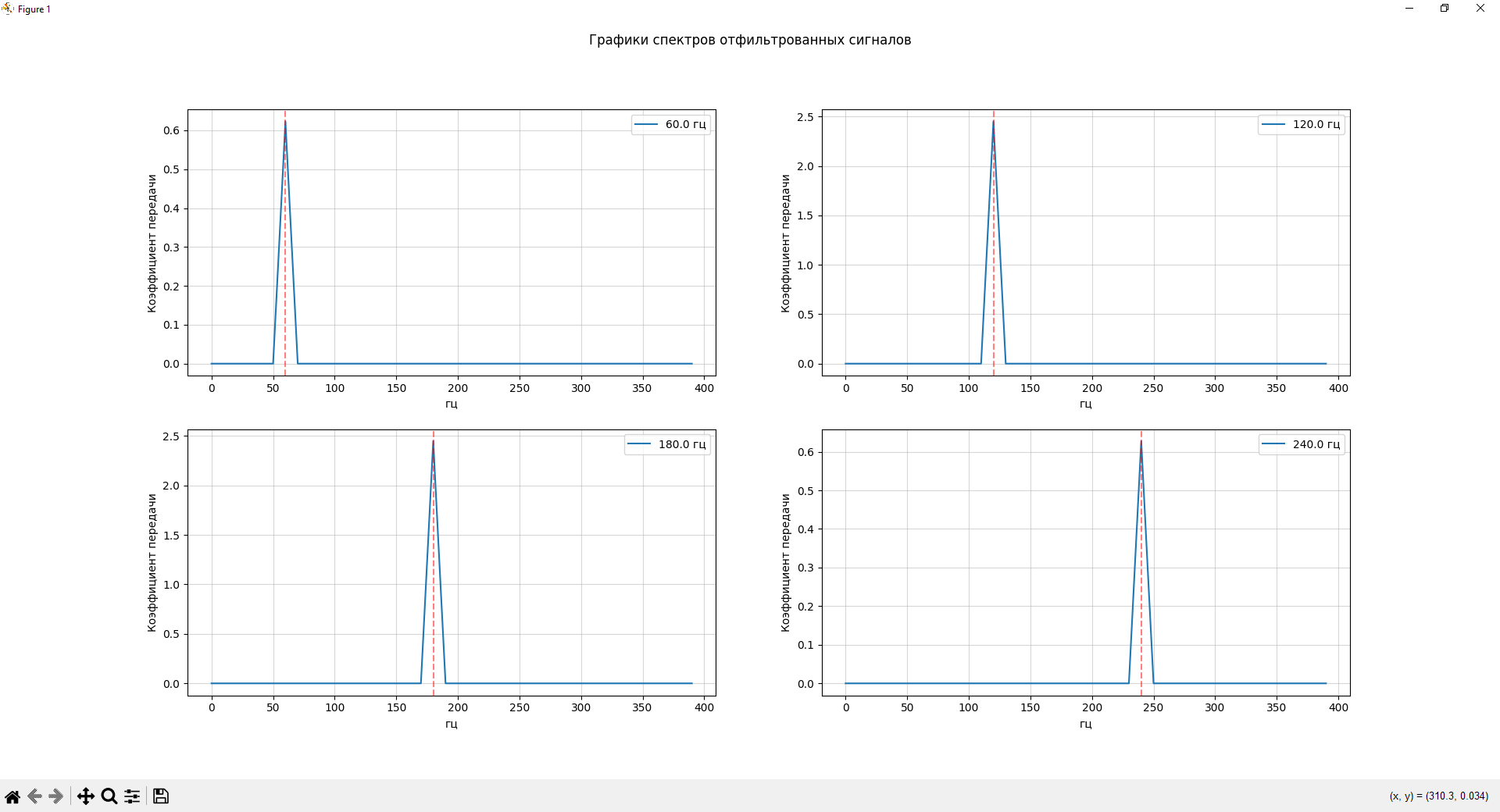


Рис. 3 Результаты фильтрации основных гармоник исходного сигнала

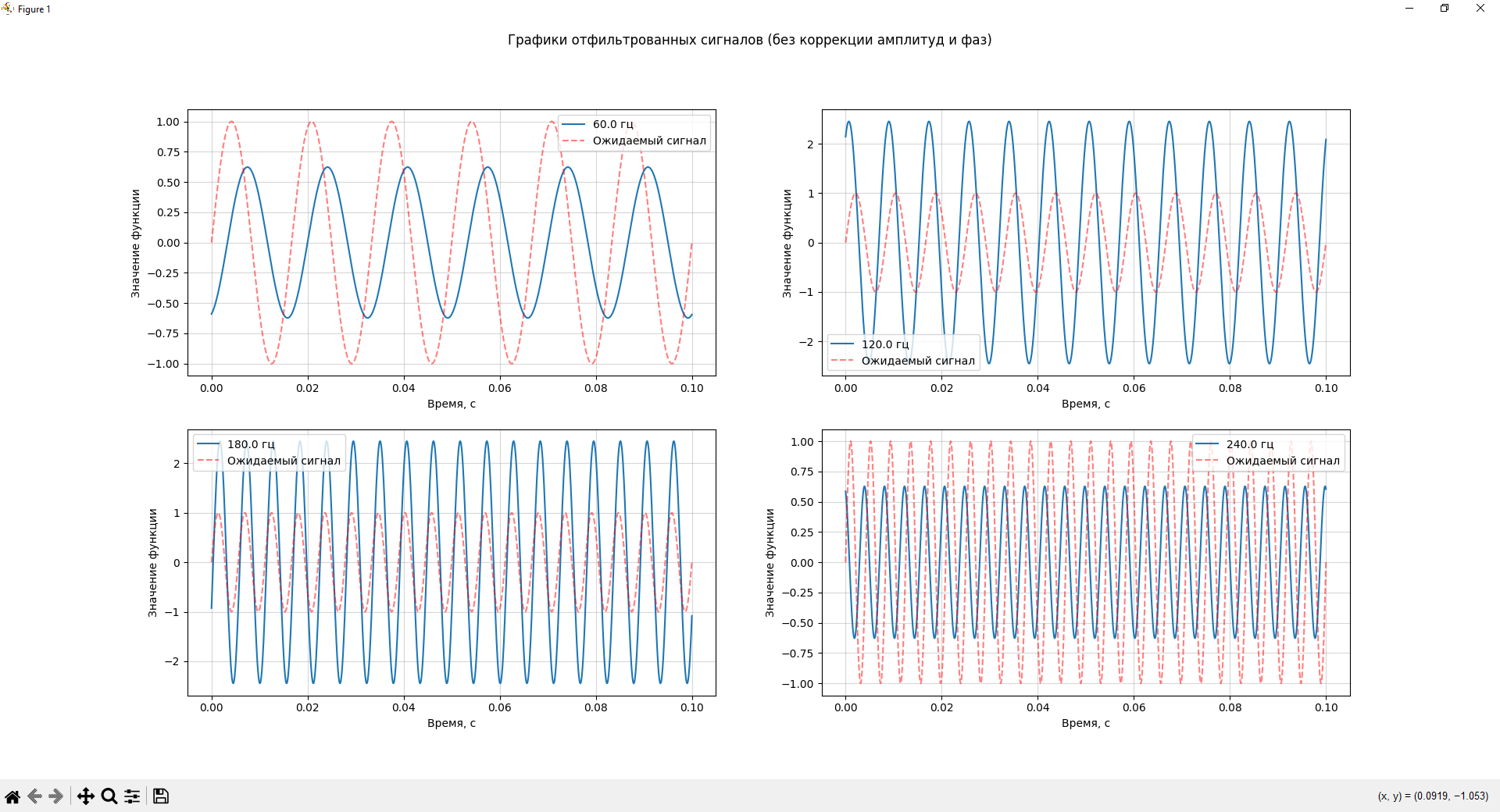


Рис. 4 Результат применения обратного преобразования Фурье к отфильтрованным спектральным функциям

**Листинг кода:**

**hw1.py**

import numpy as np

import matplotlib.pyplot as plt

from numpy.fft import fftfreq

import scipy.signal as sgl

import scipy.fft as fft

A1 = 0.5

A2 = 2.0

A3 = 1.7

A4 = 2.5

omega1 = 90

omega2 = 30

omega3 = 30

omega4 = 150

phi1 = 0

phi2 = 30

phi3 = -60

phi4 = 60

func = lambda A, omega, phi: (lambda x: A \* np.sin(2 \* np.pi \* x \* omega + phi))

s1 = func(A1, omega1, phi1)

s2 = func(A2, omega2, phi2)

s3 = func(A3, omega3, phi3)

s4 = func(A4, omega4, phi4)

fs = 16000

t = 1.0 / fs

T = 0.1

N = int(fs \* T)

x = np.linspace(0, T, N)

y1 = s1(x)

y2 = s2(x)

y3 = s3(x)

y4 = s4(x)

y = (y1 + y2 + y3) \* y4

yf = fft.fft(y)

yf\_full = np.array(yf)

xf\_full = fft.fftfreq(N, t)

yf = (2.0 / N) \* np.abs(yf[0:N//2])

xf = xf\_full[:N//2]

harmonics\_freq\_idx = sgl.argrelextrema(yf, np.greater)[0]

harmonics\_freq\_vals = xf[harmonics\_freq\_idx]

#freq\_delta = 3.0

#Rp = 0.1

#Rs = 40

#n = 5

#clamp = lambda x, mn, mx: int(mn) if x < mn else int(mx) if x > mx else int(x)

#clamp\_yf = lambda x: clamp(x, 0, yf\_full.size)

#epsilon = 3

def extract\_harmonic(freq, bandwidth=3.0):

mask\_pos = ((xf\_full >= freq - bandwidth) & (xf\_full <= freq + bandwidth))

mask\_neg = ((xf\_full >= -freq - bandwidth) & (xf\_full <= -freq + bandwidth))

yf\_filtered = np.zeros\_like(yf\_full)

yf\_filtered[mask\_pos] = yf\_full[mask\_pos]

yf\_filtered[mask\_neg] = yf\_full[mask\_neg]

return yf\_filtered

#filter\_func = lambda freq: sgl.filtfilt(\*sgl.ellip(n, Rp, Rs, [freq - freq\_delta, freq + freq\_delta], 'bandpass', fs=fs), y)

#filter\_func = lambda freq\_idx: np.pad(yf\_cplx[clamp\_yf(freq\_idx-epsilon):clamp\_yf(freq\_idx+epsilon)], (clamp\_yf(freq\_idx-epsilon), yf\_cplx.size - clamp\_yf(freq\_idx+epsilon)), 'constant', constant\_values=0)

y1f\_filtered = extract\_harmonic(harmonics\_freq\_vals[0])

y2f\_filtered = extract\_harmonic(harmonics\_freq\_vals[1])

y3f\_filtered = extract\_harmonic(harmonics\_freq\_vals[2])

y4f\_filtered = extract\_harmonic(harmonics\_freq\_vals[3])

y1\_filtered = fft.ifft(y1f\_filtered)

y2\_filtered = fft.ifft(y2f\_filtered)

y3\_filtered = fft.ifft(y3f\_filtered)

y4\_filtered = fft.ifft(y4f\_filtered)

# Исходный сигнал

plt.figure()

plt.plot(x, y, label='Исходный сигнал')

plt.grid(True, alpha=0.5)

plt.xlabel('Время, с')

plt.ylabel('Значение функции')

plt.title('График исходного сигнала')

plt.legend()

plt.show()

mx\_display\_freq = 400

# Спектр основного сигнала

xf = xf[xf < mx\_display\_freq]

yf = yf[:xf.size]

plt.figure()

plt.plot(xf, yf, label='Спектр исходного сигнала')

plt.grid(True, alpha=0.5)

plt.xlabel('Частота, гц')

plt.ylabel('Коэффициент передачи')

for harmonic in harmonics\_freq\_vals:

plt.axvline(harmonic, alpha=0.5, color='r', linestyle='--', label=f'{harmonic} гц')

plt.legend()

plt.title('Спектр исходного сигнала')

plt.show()

# Спектры

plt.figure()

plt.subplot(2, 2, 1)

plt.plot(xf, 2.0 / N \* np.abs(y1f\_filtered[:xf.size]), label=f'{harmonics\_freq\_vals[0]} гц')

plt.axvline(harmonics\_freq\_vals[0], alpha=0.5, color='r', linestyle='--')

plt.grid(True, alpha=0.5)

plt.xlabel('гц')

plt.ylabel('Коэффициент передачи')

plt.legend()

plt.subplot(2, 2, 2)

plt.plot(xf, 2.0 / N \* np.abs(y2f\_filtered[:xf.size]), label=f'{harmonics\_freq\_vals[1]} гц')

plt.axvline(harmonics\_freq\_vals[1], alpha=0.5, color='r', linestyle='--')

plt.grid(True, alpha=0.5)

plt.xlabel('гц')

plt.ylabel('Коэффициент передачи')

plt.legend()

plt.subplot(2, 2, 3)

plt.plot(xf, 2.0 / N \* np.abs(y3f\_filtered[:xf.size]), label=f'{harmonics\_freq\_vals[2]} гц')

plt.axvline(harmonics\_freq\_vals[2], alpha=0.5, color='r',linestyle='--')

plt.grid(True, alpha=0.5)

plt.xlabel('гц')

plt.ylabel('Коэффициент передачи')

plt.legend()

plt.subplot(2, 2, 4)

plt.plot(xf, 2.0 / N \* np.abs(y4f\_filtered[:xf.size]), label=f'{harmonics\_freq\_vals[3]} гц')

plt.axvline(harmonics\_freq\_vals[3], alpha=0.5, color='r', linestyle='--')

plt.grid(True, alpha=0.5)

plt.xlabel('гц')

plt.ylabel('Коэффициент передачи')

plt.legend()

plt.suptitle('Графики спектров отфильтрованных сигналов')

plt.show()

# Графики отфильтрованных сигналов

# def get\_signal\_phase(y\_filtered, freq, fs):

# yf = fft.fft(y\_filtered)

# freqs = fft.fftfreq(len(y\_filtered), 1/fs)

# idx = np.argmin(np.abs(freqs - freq))

# phase = np.angle(yf[idx])

# return phase

# phase1 = get\_signal\_phase(y1\_filtered, harmonics\_freq\_vals[0], fs)

# phase2 = get\_signal\_phase(y2\_filtered, harmonics\_freq\_vals[1], fs)

# phase3 = get\_signal\_phase(y3\_filtered, harmonics\_freq\_vals[2], fs)

# phase4 = get\_signal\_phase(y4\_filtered, harmonics\_freq\_vals[3], fs)

plt.figure()

plt.subplot(2, 2, 1)

plt.plot(x[:y1\_filtered.size], y1\_filtered, label=f'{harmonics\_freq\_vals[0]} гц')

plt.plot(x[:y1\_filtered.size], np.sin(2 \* np.pi \* harmonics\_freq\_vals[0] \* x[:y1\_filtered.size]), alpha = 0.5, color='r', linestyle='--', label=f'Ожидаемый сигнал')

plt.grid(True, alpha=0.5)

plt.xlabel('Время, с')

plt.ylabel('Значение функции')

plt.legend()

plt.subplot(2, 2, 2)

plt.plot(x[:y2\_filtered.size], y2\_filtered, label=f'{harmonics\_freq\_vals[1]} гц')

plt.plot(x[:y2\_filtered.size], np.sin(2 \* np.pi \* harmonics\_freq\_vals[1] \* x[:y1\_filtered.size]), alpha = 0.5, color='r', linestyle='--', label=f'Ожидаемый сигнал')

plt.grid(True, alpha=0.5)

plt.xlabel('Время, с')

plt.ylabel('Значение функции')

plt.legend()

plt.subplot(2, 2, 3)

plt.plot(x[:y3\_filtered.size], y3\_filtered, label=f'{harmonics\_freq\_vals[2]} гц')

plt.plot(x[:y3\_filtered.size], np.sin(2 \* np.pi \* harmonics\_freq\_vals[2] \* x[:y1\_filtered.size]), alpha = 0.5, color='r', linestyle='--', label=f'Ожидаемый сигнал')

plt.grid(True, alpha=0.5)

plt.xlabel('Время, с')

plt.ylabel('Значение функции')

plt.legend()

plt.subplot(2, 2, 4)

plt.plot(x[:y4\_filtered.size], y4\_filtered, label=f'{harmonics\_freq\_vals[3]} гц')

plt.plot(x[:y4\_filtered.size], np.sin(2 \* np.pi \* harmonics\_freq\_vals[3] \* x[:y1\_filtered.size]), alpha = 0.5, color='r', linestyle='--', label=f'Ожидаемый сигнал')

plt.grid(True, alpha=0.5)

plt.xlabel('Время, с')

plt.ylabel('Значение функции')

plt.legend()

plt.suptitle('Графики отфильтрованных сигналов (без коррекции амплитуд и фаз)')

plt.show()

**Вывод:** в ходе выполнения домашней работы были сформированы практические навыки разложения сигналов с использованием дискретного преобразования Фурье (ДПФ).