Министерство науки и высшего образования Российской Федерации

Лабораторная работа № 1.1.

«Преобразование Фурье. Дифракция Фраунгофера и Френеля. Интерференция. (Часть 1)»

Выполнил: Леко А.А..

Группа: Q4110

Проверила: Иванова Т. В.

Санкт-Петербург 2023

Задание: Вычислить одномерное преобразование Фурье от функций:

- Delta
- Comb
- Rect
- Tr
- sin
- cos
- Rect(x-1)
- Rect(x+1)
- Rect(x*2)
- Rect(x/2)
- 1 — Rect(x)

Графики функций, построенные при помощи Python 3:

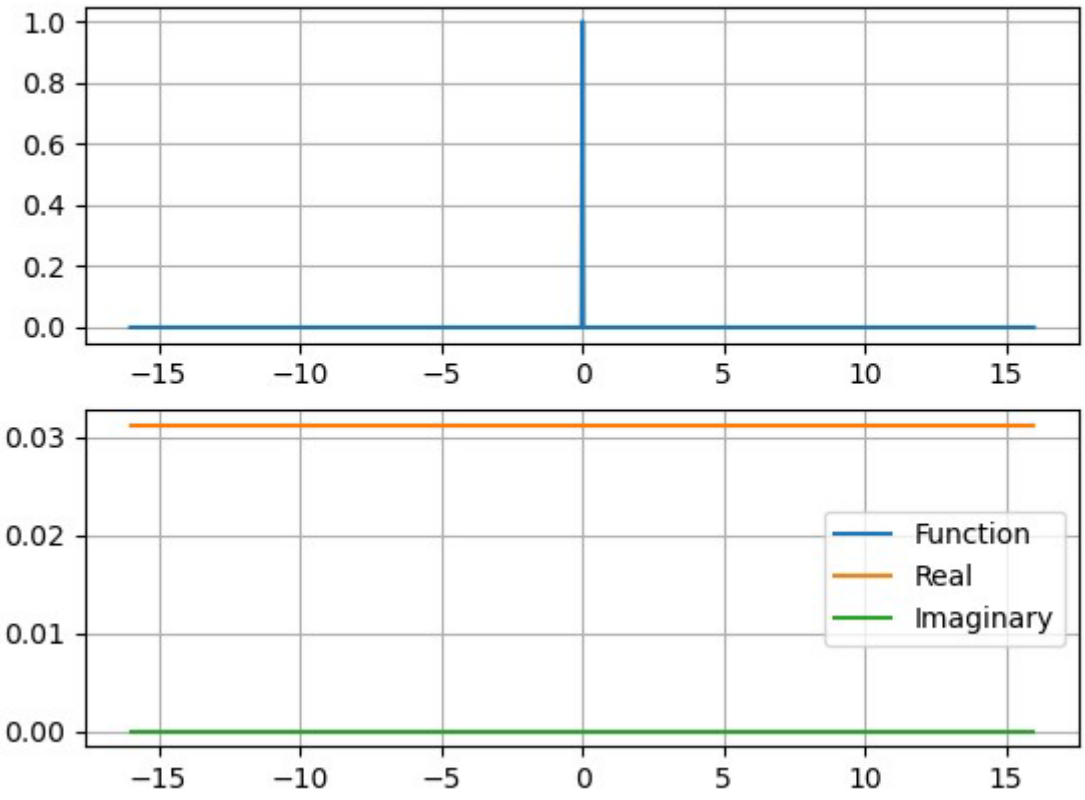

Рисунок 1 — Графики функции Delta, результата преобразования Фурье и спектра

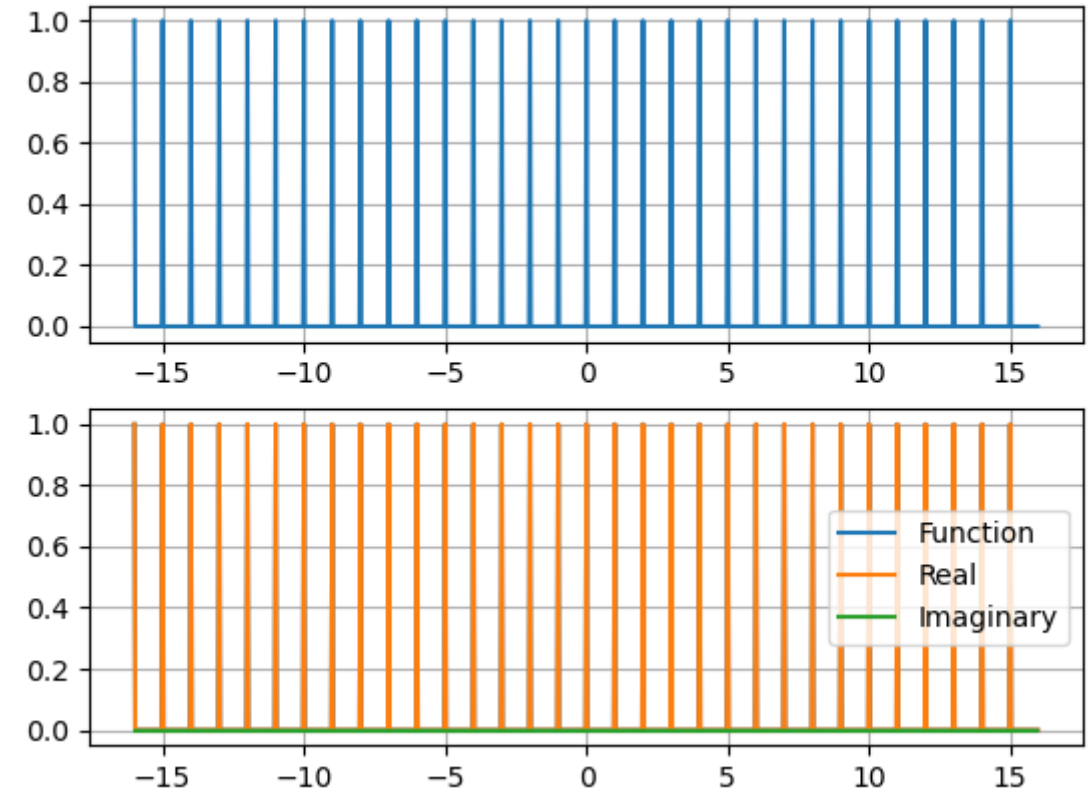

Рисунок 2 — Графики функции Comb, результата преобразования Фурье и спектра

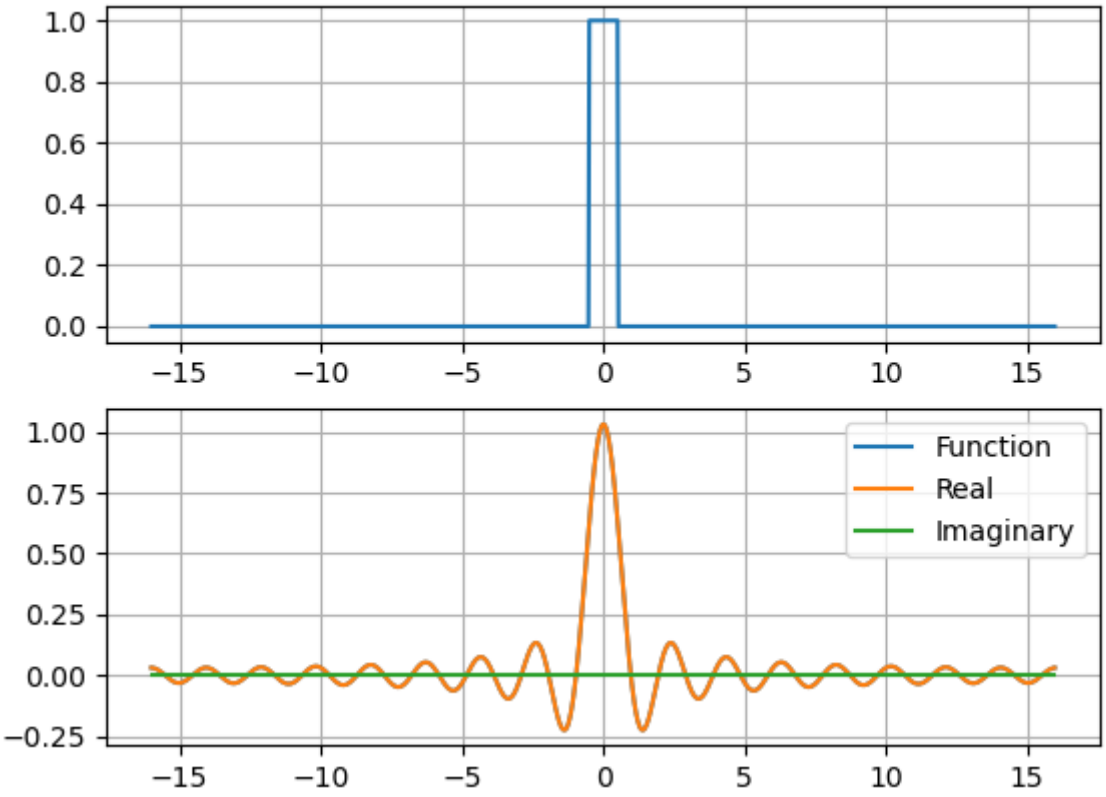

Рисунок 3 — Графики функции Rect, результата преобразования Фурье и спектра

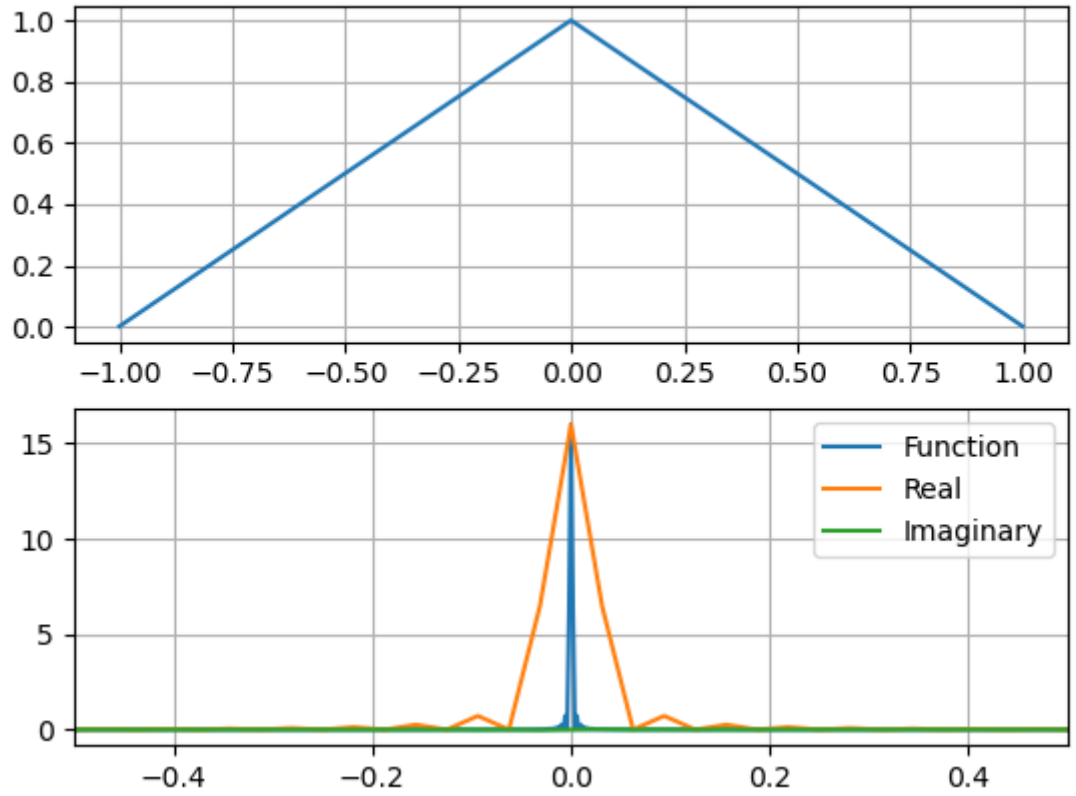

Рисунок 4 — Графики функции Tr, результата преобразования Фурье и спектра

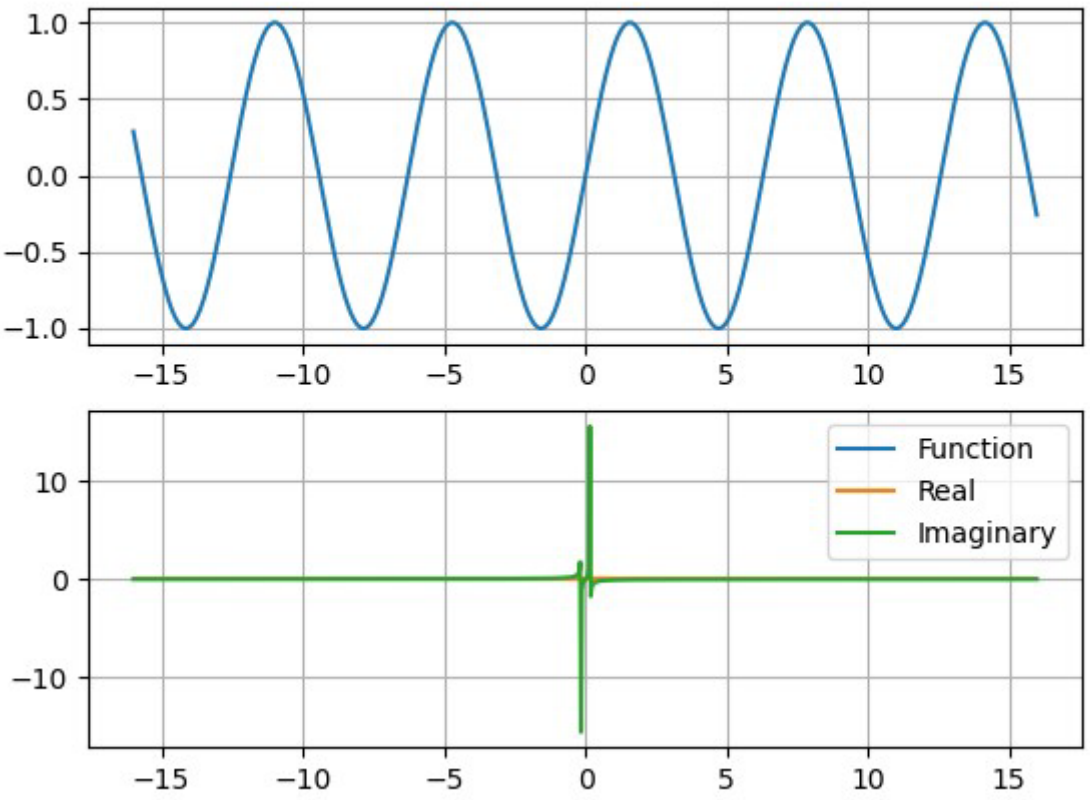

Рисунок 5 — Графики функции sin, результата преобразования Фурье и спектра

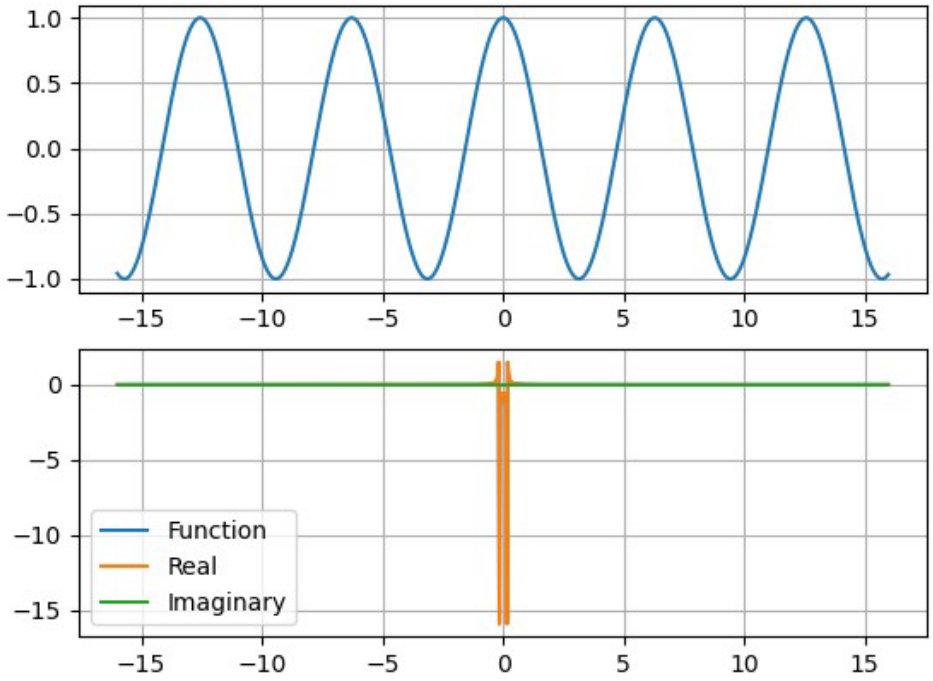

Рисунок 6 — Графики функции cos, результата преобразования Фурье и спектра

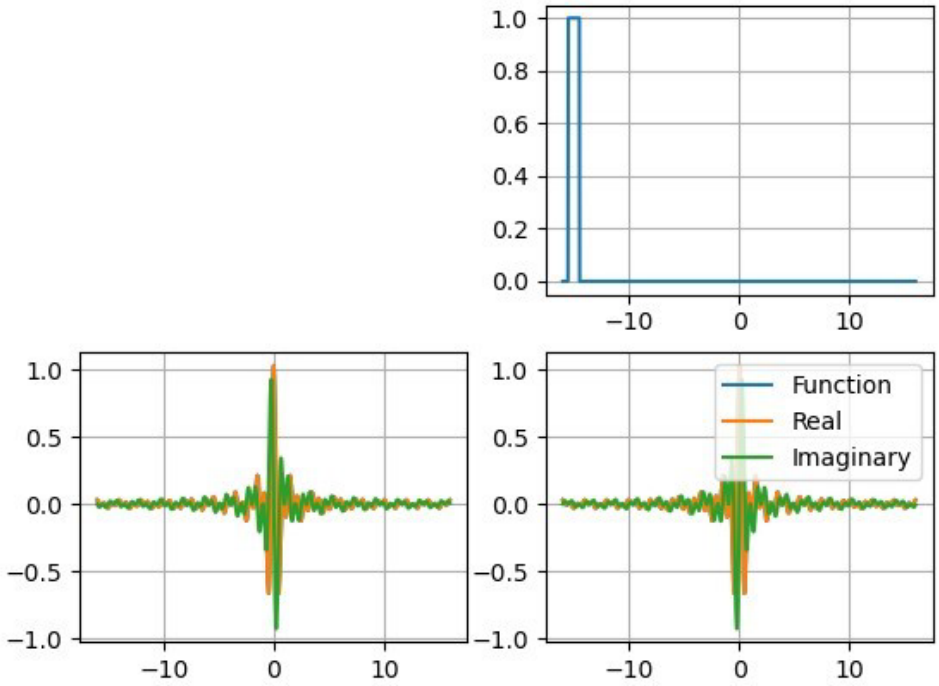

Рисунок 7 — Графики функции Rect(x-1), результатов прямого и обратного преобразования Фурье и спектра

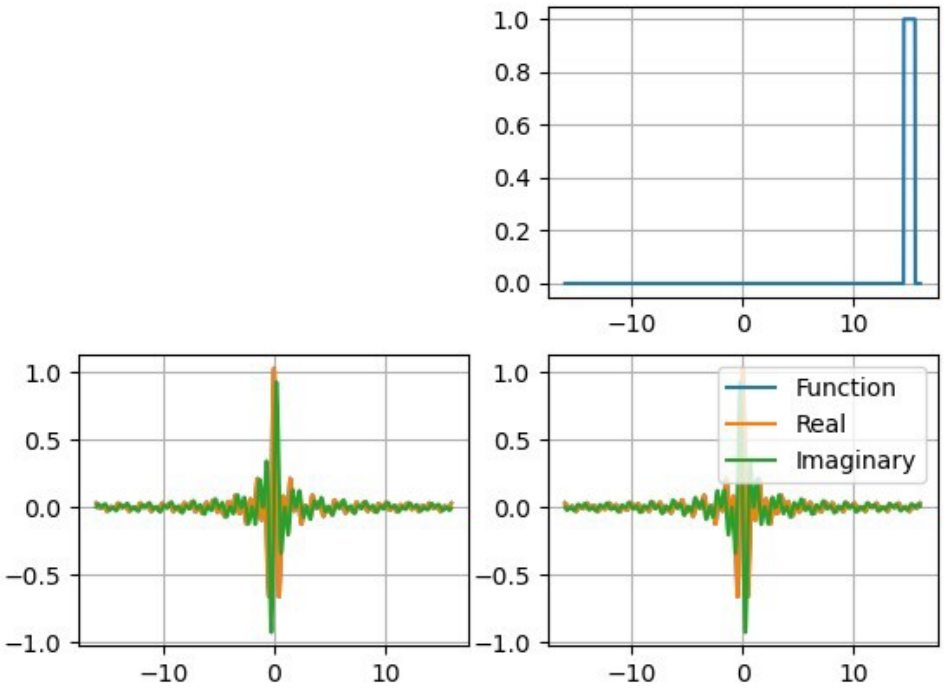

Рисунок 8 — Графики функции Rect(x+1), результатов прямого и обратного преобразования Фурье и спектра

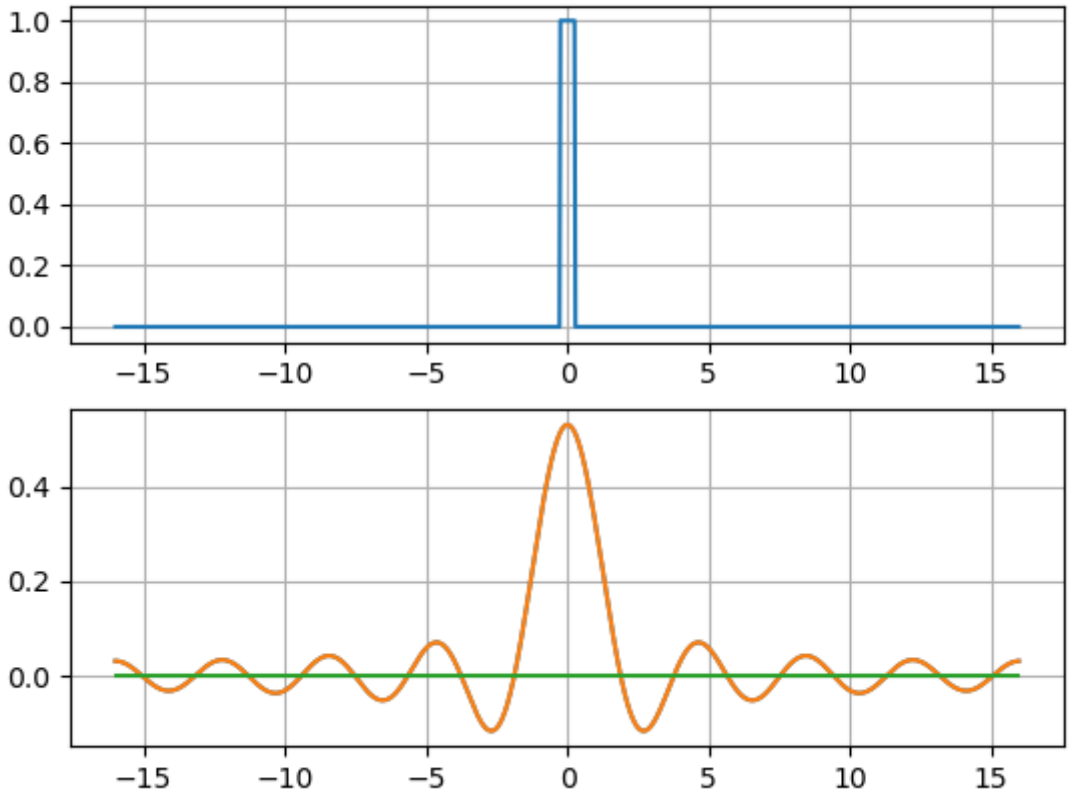

Рисунок 9 — Графики функции Rect(x*2), результата преобразования Фурье и спектра

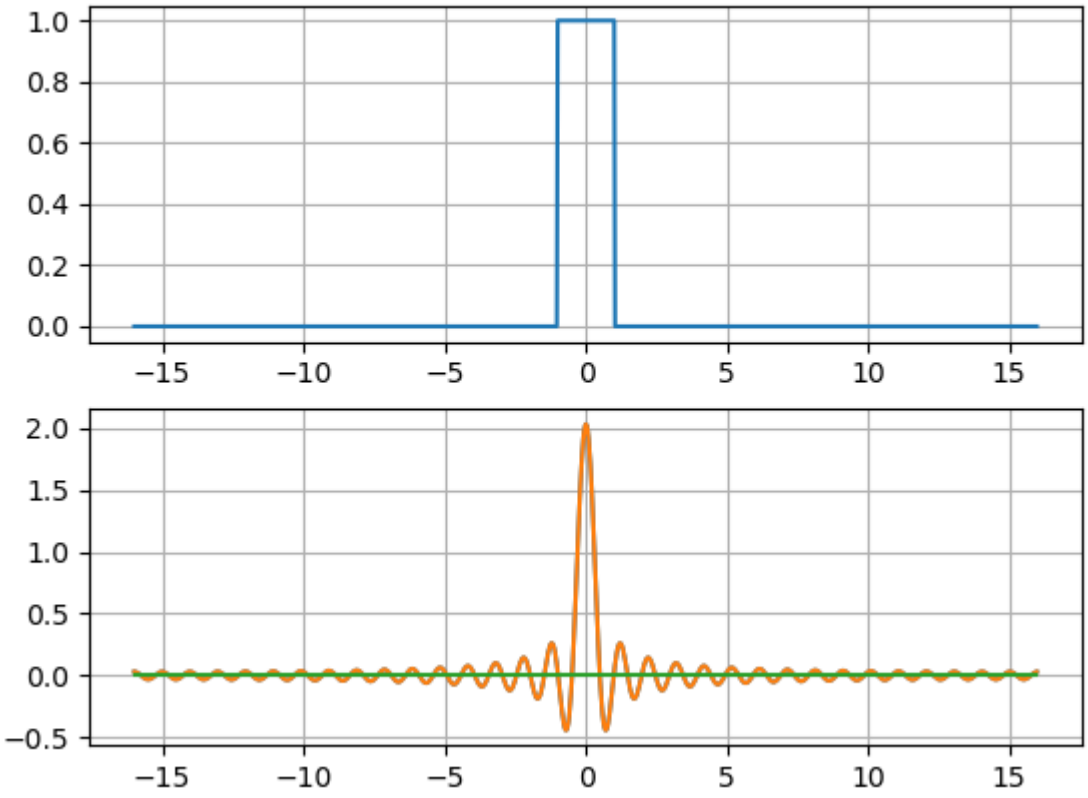

Рисунок 10 — Графики функции Rect(x/2), результата преобразования Фурье и спектра

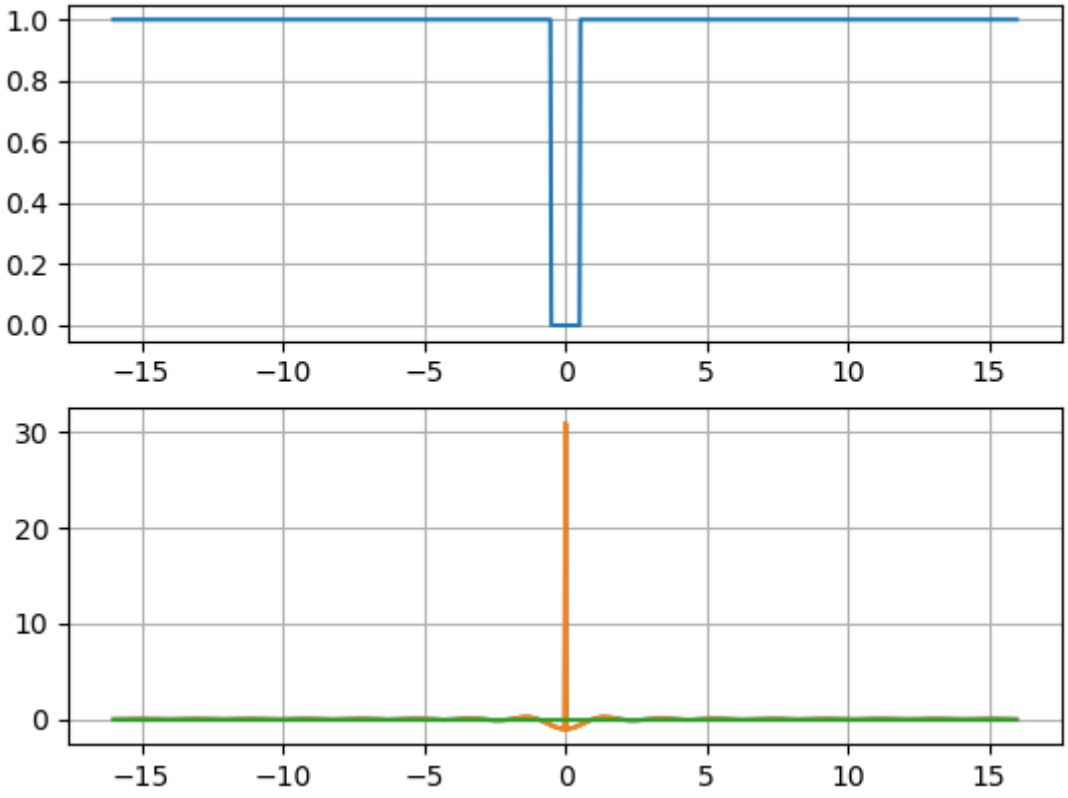

Рисунок 11 — Графики функции 1 - Rect(x), результата преобразования Фурье и спектра

Выводы: В ходе работы было освоено выполнение преобразования Фурье от различных

функций с использованием языка программирования Python 3, а также библиотек scipy, numpy и matplotlib.

Текст программы:

```
import matplotlib.pyplot as plt   #to run this program libraries
must be installed
from scipy.fft import fft, ifft, fftshift
from scipy.signal.windows import triang
import numpy as np


def delta(n):  # needs length N
    delta_function = [0 for i in range(N)]  # array of zeros
    delta_function[len(delta_function) // 2] = 1
    delta_function = np.array(delta_function)
    return delta_function


def comb(x, n):   #needs list of x and length N
    f_comb = np.zeros(n)
    for i in range(n):
        if x[i] % 1 == 0:
            f_comb[i] = 1
    return f_comb


def rect(x):  # needs list x
    return np.where(abs(x) <= 0.5, 1, 0)


def tr(n):
    return triang(n)


def sin(x):
    return np.sin(x)


def cos(x):
    return np.cos(x)


N = 1024
step = (1 / N) ** (1 / 2)
x_max = step * (N / 2)
x = np.arange(-x_max, x_max, step)
x_tr = np.arange(-1, 1, 1 / 512)

delta_function = delta(N)
```

```
y_delta = fftshift(fft(fftshift(delta_function)))/(N ** 0.5)

comb = comb(x, N)
y_comb = fft(comb)/(N ** 0.5)

rect_func = rect(x)
rect_func1 = fftshift(rect_func)
y_rect = fftshift(fft(rect_func1)) / (N ** 0.5)

triangular = tr(N)
y_tr = fftshift(fft(fftshift(triangular))) / (N ** 0.5)

sin = sin(x)
y_sin = fftshift(fft(sin))/(N ** 0.5)

cos = cos(x)
y_cos = fftshift(fft(cos))/(N ** 0.5)

plt.figure(1)
plt.subplot(211)
plt.plot(x, delta_function)
plt.grid(True)
plt.subplot(212)
plt.plot(x, y_delta, label='Function')
plt.plot(x, np.real(y_delta), label='Real')
plt.plot(x, np.imag(y_delta), label='Imaginary')
plt.grid(True)
plt.legend()

plt.figure(2)
plt.subplot(211)
plt.plot(x, comb)
plt.grid(True)
plt.subplot(212)
plt.plot(x, y_comb, label='Function')
plt.plot(x, np.real(y_comb), label='Real')
plt.plot(x, np.imag(y_comb), label='Imaginary')
plt.grid(True)
plt.legend()


plt.figure(3)
plt.subplot(211)
plt.plot(x, rect_func)
plt.grid(True)
plt.subplot(212)
plt.plot(x, y_rect, label='Function')
plt.plot(x, np.real(y_rect), label='Real')
plt.plot(x, np.imag(y_rect), label='Imaginary')
plt.grid(True)
plt.legend()

plt.figure(4)
```

```
plt.subplot(211)
plt.plot(x_tr, triangular)
plt.grid(True)
plt.subplot(212)
plt.plot(x_tr, y_tr, label='Function')
plt.plot(x, np.real(y_tr), label='Real')
plt.plot(x, np.imag(y_tr), label='Imaginary')
plt.xlim([-0.5, 0.5])
plt.grid(True)
plt.legend()

plt.figure(5)
plt.subplot(211)
plt.plot(x, sin)
plt.grid(True)
plt.subplot(212)
plt.plot(x, y_sin, label='Function')
plt.plot(x, np.real(y_sin), label='Real')
plt.plot(x, np.imag(y_sin), label='Imaginary')
plt.grid(True)
plt.legend()

plt.figure(6)
plt.subplot(211)
plt.plot(x, cos)
plt.grid(True)
plt.subplot(212)
plt.plot(x, y_cos, label='Function')
plt.plot(x, np.real(y_cos), label='Real')
plt.plot(x, np.imag(y_cos), label='Imaginary')
plt.grid(True)
plt.legend()
plt.show()

rect1 = rect(x-1)
rect11 = fftshift(rect1)
y_rect1 = fftshift(fft(rect11)) / (N ** 0.5)
yi_rect1 = fftshift(ifft(rect11)) * (N ** 0.5)

plt.figure(7)
plt.subplot(222)
plt.plot(x, rect1)
plt.grid(True)
plt.subplot(223)
plt.plot(x, y_rect1, label='Function')
plt.plot(x, np.real(y_rect1), label='Real')
plt.plot(x, np.imag(y_rect1), label='Imaginary')
plt.grid(True)
plt.subplot(224)
plt.plot(x, yi_rect1, label='Function')
plt.plot(x, np.real(yi_rect1), label='Real')
plt.plot(x, np.imag(yi_rect1), label='Imaginary')
plt.grid(True)
```

```
plt.legend()

rect2 = rect(x+1)
rect21 = fftshift(rect2)
y_rect2 = fftshift(fft(rect21)) / (N ** 0.5)
yi_rect2 = fftshift(ifft(rect21)) * (N ** 0.5)

plt.figure(8)
plt.subplot(222)
plt.plot(x, rect2)
plt.grid(True)
plt.subplot(223)
plt.plot(x, y_rect2, label='Function')
plt.plot(x, np.real(y_rect2), label='Real')
plt.plot(x, np.imag(y_rect2), label='Imaginary')
plt.grid(True)
plt.subplot(224)
plt.plot(x, yi_rect2, label='Function')
plt.plot(x, np.real(yi_rect2), label='Real')
plt.plot(x, np.imag(yi_rect2), label='Imaginary')
plt.grid(True)
plt.legend()

rect3 = rect(x*2)
rect31 = fftshift(rect3)
y_rect3 = fftshift(fft(rect31)) / (N ** 0.5)
plt.figure(9)
plt.subplot(211)
plt.plot(x, rect3)
plt.grid(True)
plt.subplot(212)
plt.plot(x, y_rect3, label='Function')
plt.plot(x, np.real(y_rect3), label='Real')
plt.plot(x, np.imag(y_rect3), label='Imaginary')
plt.grid(True)

rect4 = rect(x/2)
rect41 = fftshift(rect4)
y_rect4 = fftshift(fft(rect41)) / (N ** 0.5)
plt.figure(10)
plt.subplot(211)
plt.plot(x, rect4)
plt.grid(True)
plt.subplot(212)
plt.plot(x, y_rect4, label='Function')
plt.plot(x, np.real(y_rect4), label='Real')
plt.plot(x, np.imag(y_rect4), label='Imaginary')
plt.grid(True)

rect5 = 1 - rect(x)
rect51 = fftshift(rect5)
y_rect5 = fftshift(fft(rect51)) / (N ** 0.5)
plt.figure(11)
```

```
plt.subplot(211)
plt.plot(x, rect5)
plt.grid(True)
plt.subplot(212)
plt.plot(x, y_rect5, label='Function')
plt.plot(x, np.real(y_rect5), label='Real')
plt.plot(x, np.imag(y_rect5), label='Imaginary')
plt.grid(True)
plt.show()
```