Министерство образования и науки Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Самарский национальный исследовательский университет
имени академика С.П.Королёва»

Институт информатики, математики и электроники
Факультет информатики
Кафедра технической кибернетики

Отчет по лабораторной работе No2
по курсу «Оптоинформационные технологии и системы»

Вариант 9

Выполнил студент: Белоусов А. А.
Группа: 6409
Преподаватель: Кириленко М.С.

Самара 2019

ЗАДАНИЕ

1. Реализовать одномерное финитное преобразование Фурье с помощью применения алгоритма БПФ.

2. Построить график гауссова пучка e^{-x^2}. Здесь и далее для каждого графика следует строить отдельно графики амплитуды и фазы. Амплитуда находится как модуль каждого значения функции, фаза – как аргумент (или с помощью функции atan2).

3. Убедиться в правильности реализации преобразования, подав на вход гауссов пучок e^{-x^2} – собственную функцию преобразования Фурье. На выходе тоже должен получиться гауссов пучок (построить график на правильной области определения $[-\tilde{b}, \tilde{b}]$). Рекомендуемая входная область: $[-a, a] = [-5, 5]$.

4. Реализовать финитное преобразование Фурье стандартным методом численного интегрирования (например, методом прямоугольников). Важно: необходимо вычислить интеграл для каждого дискретного значения u, чтобы получить результат в виде вектора. На вход преобразования вновь следует подавать гауссов пучок.

5. Построить результаты двух разных реализаций преобразования на одном изображении (одно для амплитуды, одно для фазы) и убедиться, что они совпадают.

6. Используя первую реализацию преобразования, подать на вход световое поле, отличное от гауссова пучка, в соответствии со своим вариантом. Построить графики самого пучка и результата преобразования.

7. Рассчитать аналитически результат преобразования своего варианта поля и построить график на одной системе координат с результатом, полученным в предыдущем пункте.

АЛГОРИТМ РЕАЛИЗАЦИИ ОПТИЧЕСКОГО ФИНИТНОГО ПРЕОБРАЗОВАНИЯ ФУРЬЕ ЧЕРЕЗ ИСПОЛЬЗОВАНИЕ БПФ

1. Провести дискретизацию входной функции $f(x)$ в вектор f с размерностью N.
2. Дополнить вектор f и слева, и справа необходимым числом нулей до размерности M.
3. Разбить вектор f на две половины и поменять их местами.
4. Выполнить БПФ от f и умножить результат на шаг h_x , получив вектор F.
5. Разбить вектор F на две половины и поменять их местами.
6. «Вырезать» центральную часть вектора F, оставив N элементов.

Выполнение задания

1. Одномерное финитное преобразование Фурье записывается в виде (1);

$$F_a(u) = \Phi_a\left[f(x)\right](u) = \int\limits_{-a}^{a} f(x)e^{-2\pi ixu}\,dx, \tag{1}$$

где $f(x)$ – финитная функция, $F_a(u)$ – спектр, Φ_a – оператор финитного преобразования Фурье.

Приведём реализацию одномерного финитного преобразования Фурье (полный код представлен в приложении А):

2. На следующих рисунках изображены графики гауссова пучка, его амплитуды и фазы.

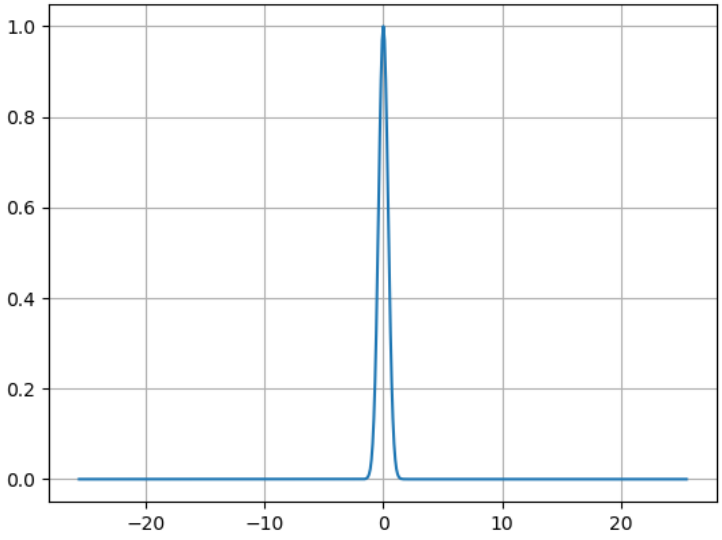

Рисунок 1 – График гауссова пучка

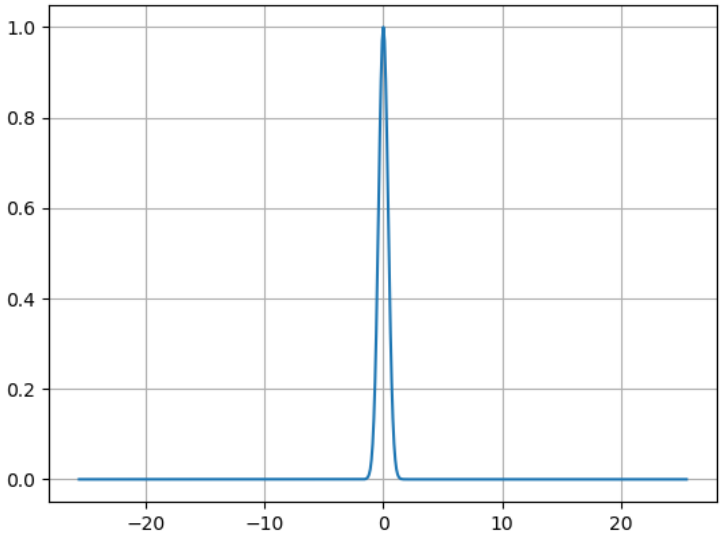

Рисунок 2 – График амплитуды гауссова пучка

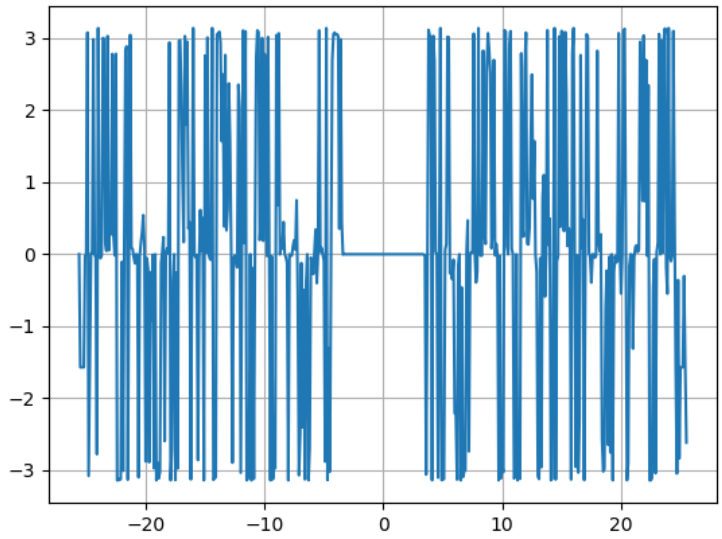

Рисунок 3 – График фазы гауссова пучка

3. Проверка правильности реализации преобразования Фурье. Наложим график исходной функции гауссова пучка на график амплитуды. При корректной реализации графики должны совпадать, что мы можем наблюдать на следующем рисунке.

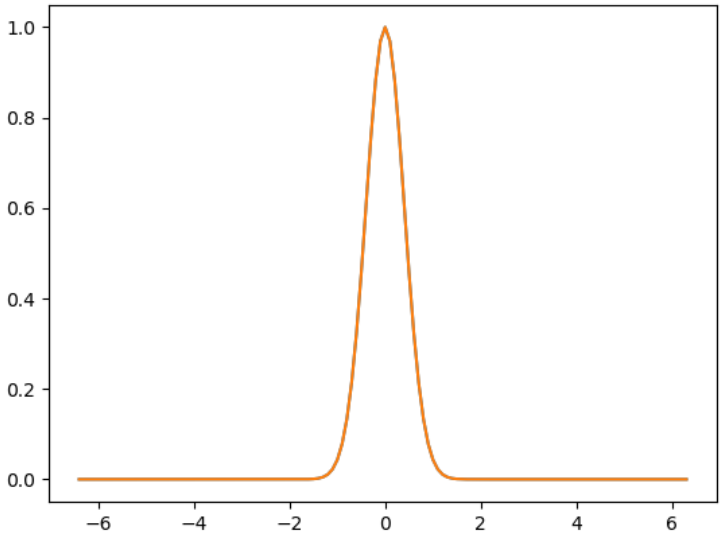

Рисунок 4 – Совмещенные графики гаусоова пучка и амплитуды

4. Реализуем финитное преобразование Фурье стандартным методом численного интегрирования – методом прямоугольников (2).

$$\int_a^b f(x)dx = \sum_{k=1}^{n} f(\frac{x_{i-1} + x_i}{2}(x_i - x_{i-1}). \tag{2}$$

Приведём реализацию метода численного интегрирования (полный код представлен в приложении А):

5. Сравнение преобразований Фурье: стандартным методом численного интегрирования и с помощью применения алгоритма БПФ. Результаты изображены на рисунках 5, 6.

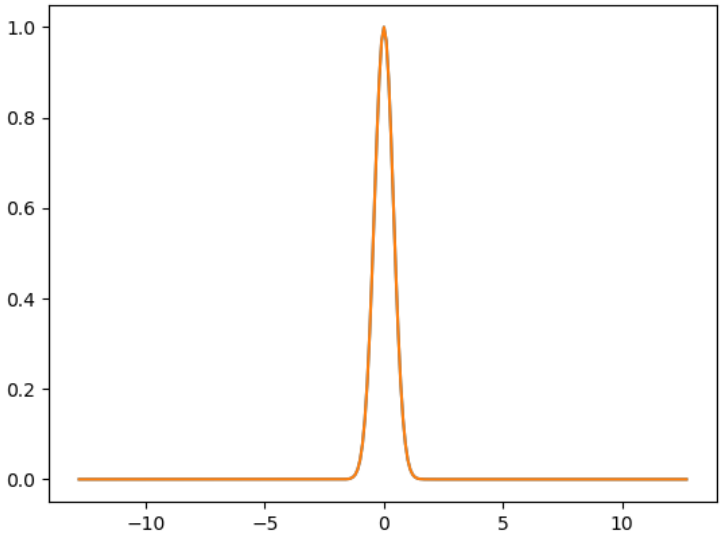

Рисунок 5 – Сравнение амплитуд преобразований Фурье

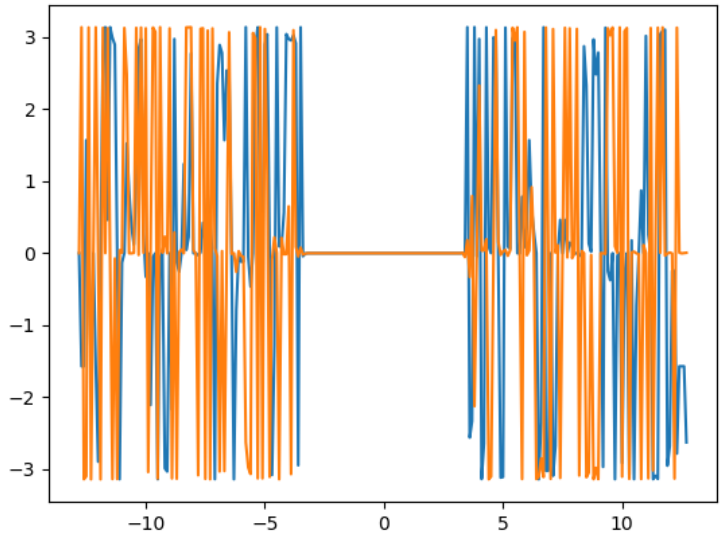

Рисунок 6 – Сравнение фаз преобразований Фурье

Как можно видеть, численное преобразование Фурье, посчитанное с помощью метода прямоугольников совпадает с преобразованием Фурье, вычисленным с помощью алгоритма БПФ.

6. На вход подается функция: $rect((x-1)/4)$, используется реализация преобразования Фурье через БПФ. Графики результатов преобразования изображены на

рисунках 7, 8.

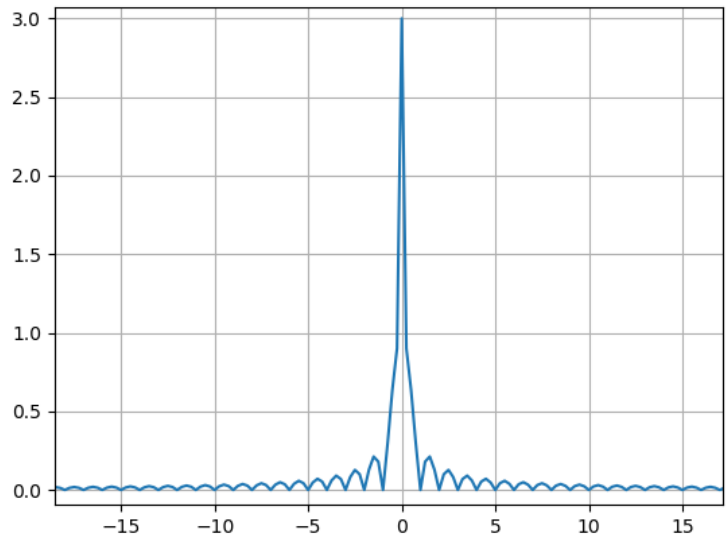

Рисунок 7 – График амплитуды функции $rect((x-1)/4)$

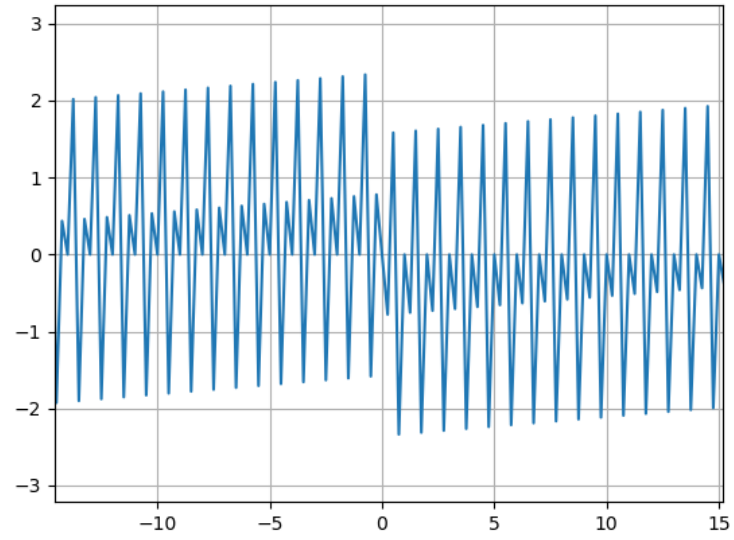

Рисунок 8 – График фазы функции $rect((x-1)/4)$

7. Найдём аналитическую форму для преобразования Фурье функции $rect((x-$

1)/4).

$$F\{rect((x-1)/4)\} = \int_{-1}^{3} rect(x)e^{-2\pi ix\xi}\,dx = \frac{1}{-2\pi ix\xi}\left[e^{-2\pi ix\xi}\Big|_{-1}^{3}\right] =$$
$$= \frac{1}{-2\pi ix\xi}\left[e^{-6\pi ix\xi} - e^{2\pi ix\xi}\right].$$

Результаты сравнения аналитического преобразования Фурье и преобразования Фурье через БПФ представлены на следующих рисунках.

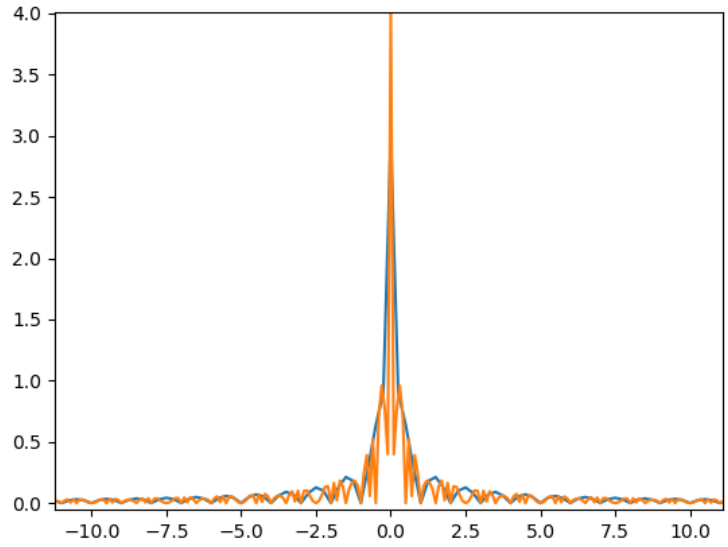

Рисунок 9 – Сравнение амплитуды аналитического решения и преобразования Фурье через БПФ

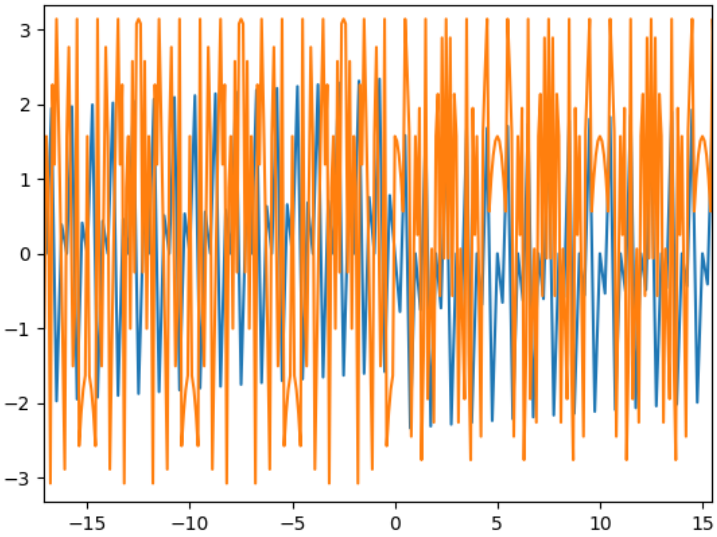

Рисунок 10 – Сравнение фазы аналитического решения и преобразования Фурье через БПФ

ЗАКЛЮЧЕНИЕ

В данной лабораторной работе было реализовано одномерное финитное преобразование Фурье с помощью метода БПФ, а так же с помощью метода численного интегрирования. Был рассчитан теоретический результат преобразования Фурье для функции $rect((x-1)/4)$. Построены графики для сравнения результатов различных преобразований. Было выяснено, что результаты аналитического преобразования Фурье и преобразования Фурье через БПФ достаточно близки.

ПРИЛОЖЕНИЕ А

Код программы

```
from typing import List
import numpy as np
import matplotlib.pyplot as plt


def gauss_beam(x):
    return np.exp(-np.pi * (x**2))


def custom_func(x: float):
    # if abs(x) > 0.5:
    # return 0
    # return (x - 1) / 4
    if x > 3 or x < -1:
        return 0
    return 1


def custom_func_fourier(x: float, xi: float):
    # return (2 * np.sqrt(2 / np.pi) * np.sin(xi) * np.cos(xi) * (np.cos(xi) + 1j * np.
        ↪ sin(xi))) / xi
    # return np.exp(-2*np.pi*1j*xi) * np.sin(4 * np.pi * xi) / np.pi * xi

    flags = np.abs(xi) < 1e-6
    around_zero = 4 * 1j
    #around_zero = 1j
    others = -1/(2 * np.pi * xi) * (np.exp(-6 * np.pi * x * 1j * xi) - (np.exp(2 * np.
        ↪ pi * x * 1j * xi)))

    return others * (1-flags) + around_zero * flags


def fourier_builitn(f, step: float, xs: List[float]) -> List[float]:
    fs = list(map(lambda x: f(x), xs))
    fs = np.fft.fftshift(fs)
    fourier = list(map(lambda x: x * step, np.fft.fft(fs)))
    return np.fft.fftshift(fourier)
```

```
def fourier_exp(u: float, x: float) -> float:
    return np.exp(-2 * np.pi * 1j * u * x)


def fourier_calc(f, step: float, xs: [float],
                 us: List[float]) -> List[float]:
    fourier = []
    for u in us:
        newx = 0
        for x in xs:
            newx += f(x) * fourier_exp(u, x)
        newx *= step
        fourier.append(newx)
    return fourier


def plot_calculus(f, a: float, b: float, step_count: int, fig_name):
    step_src = abs(b - a) / step_count
    c = step_count / (2 * abs(b - a))
    step = (2 * abs(c)) / step_count
    us = np.arange(-c, c, step)
    xs = np.arange(a, b, step_src)
    fourier = fourier_calc(f, step_src, xs, us)
    phase = np.angle(fourier)
    amplitude = np.abs(fourier)
    #print(f"Calculus => {fourier[:4]}")
    plt.figure(f"Phase {fig_name}")
    plt.plot(us, phase)
    plt.grid()
    plt.figure(f"Amplitude {fig_name}")
    plt.plot(us, amplitude)
    plt.grid()


def plot_fft(f, a: float, b: float, step_count: int, fig_name):
    step_src = (b - a) / step_count
    c = step_count / (2 * (abs(a) + abs(b)))
    step = (2 * abs(c)) / step_count
```

```
    # step = (abs(a) + abs(b)) / step_count
    xs = np.arange(a, b, step_src)
    nxs = np.arange(-c, c, step)
    fourier = fourier_builitn(f, step_src, xs)
    phase = np.angle(fourier)
    amplitude = np.abs(fourier)
    plt.figure(f"Phase {fig_name}")
    plt.plot(nxs, phase)
    plt.grid()
    plt.figure(f"Amplitude {fig_name}")
    plt.plot(nxs, amplitude)
    plt.grid()


def plot_complex_function(f, a, b, step_count, fig_name):
    c = step_count / (2 * (abs(a) + abs(b)))
    step = (2 * abs(c)) / step_count
    nxs = np.arange(-c, c, step)
    us = np.arange(-c, c, step)
    ys = f(nxs, us)
    #ys = f(nxs)
    amplitude = np.abs(ys)
    phase = np.angle(ys)
    plt.figure(f"Phase {fig_name}")
    plt.plot(nxs, phase)
    plt.grid()
    plt.figure(f"Amplitude {fig_name}")
    plt.plot(nxs, amplitude)
    plt.grid()


if __name__ == "__main__":
    #plot_complex_function(gauss_beam, -5, 5, 64, "gauss")
    #plot_fft(gauss_beam, -5, 5, 256, "gauss")
    #plot_calculus(gauss_beam, -5, 5, 256, "gauss")
    plot_fft(custom_func, -2, 2, 512, "custom")
    #plot_calculus(custom_func, -5, 5, 512, "custom")
    #plot_complex_function(custom_func_fourier, -5, 5, 512, "custom")

    plt.show()
```