МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение
высшего образования «Самарский национальный исследовательский
университет имени академика С.П. Королева»
(Самарский университет)

Институт информатики и кибернетики
Кафедра технической кибернетики

ОТЧЕТ

по Лабораторной работе № 3

Реализация оптического преобразования Ханкеля для радиально-вихревых пучков

Вариант 6

Выполнил студент
группы 6402-010302D
Янкин И.Ю.

Самара 2024

1 Задание на лабораторную работу

6	Полином Цернике	$Z_5^{-3}(r,\varphi)$	$m = -3$

1. Выбрать входную функцию $f(r,\varphi) = f(r)e^{\mathrm{im}\varphi}$ и число m, исходя из варианта. Построить график $f(r)$. Здесь и далее для каждого графика следует строить отдельно графики/изображения амплитуды и фазы. Входную область ограничить радиусом $R = 5$.
2. Восстановить изображение $f(r)e^{\mathrm{im}\varphi}$ в двумерный массив и построить это изображение.
3. Реализовать преобразование Ханкеля методом численного интегрирования (например, методом левых прямоугольников). Размеры входной и выходной областей должны совпадать. Применить преобразование ко входной функции и получить выходную $F(\rho)$. Построить её график, а также восстановить двумерную функцию $F(\rho)e^{im\theta}$ и построить её изображение.
4. Реализовать двумерное преобразование Фурье через БПФ. Применить его ко входной двумерной функции $f(r)e^{\mathrm{im}\varphi}$. Построить изображение выходной функции, сравнить его с результатом, полученным для преобразования Ханкеля. Если изображения амплитуд сильно отличаются, попытаться увеличить число точек дискретизации.
5. Исследовать скорость выполнения двумерного БПФ и преобразования Ханкеля, варьируя число точек дискретизации. Сделать выводы.

2 Результаты работы программы

2.1 График входной функции

Входная функция в предложенном варианте работы имеет вид:

$$Z_5^{-3}(r,\varphi) = R_5^{|-3|}(r)e^{ip\varphi},$$

где $R_5^{|-3|}$ – радиальные полиномы Цернике, которые определяются по формуле:

$$R_n^{|p|} = \sum_{k=0}^{\frac{n-p}{2}} \frac{(-1)^k (n-k)!}{k!\left(\frac{n+p}{2}-k\right)!\left(\frac{n-p}{2}-k\right)} r^{n-2k}.$$

На рисунке 1 представлены графики амплитуды и фазы полинома Цернике.

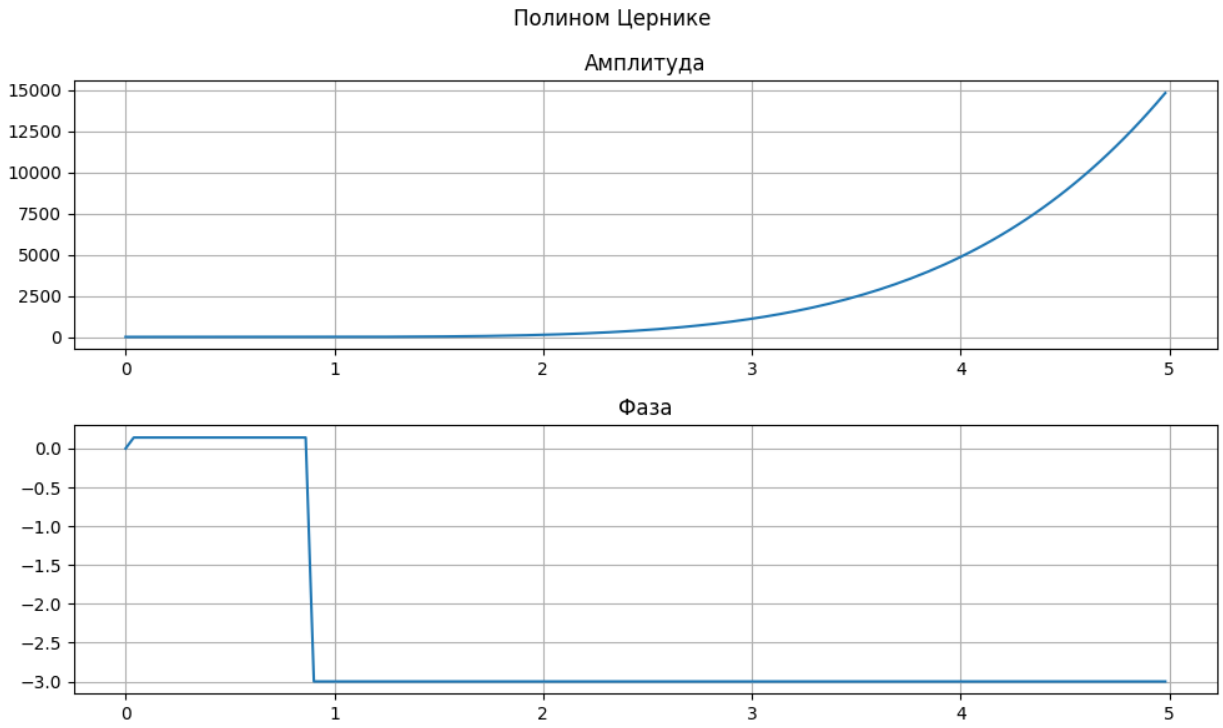

Рисунок 1 – Амплитуда и фаза полинома Цернике

2.2 Восстановление изображения

Построим теперь график восстановленного изображения. Результат амплитуды и фазы восстановленного изображения представлен на рисунке 2.

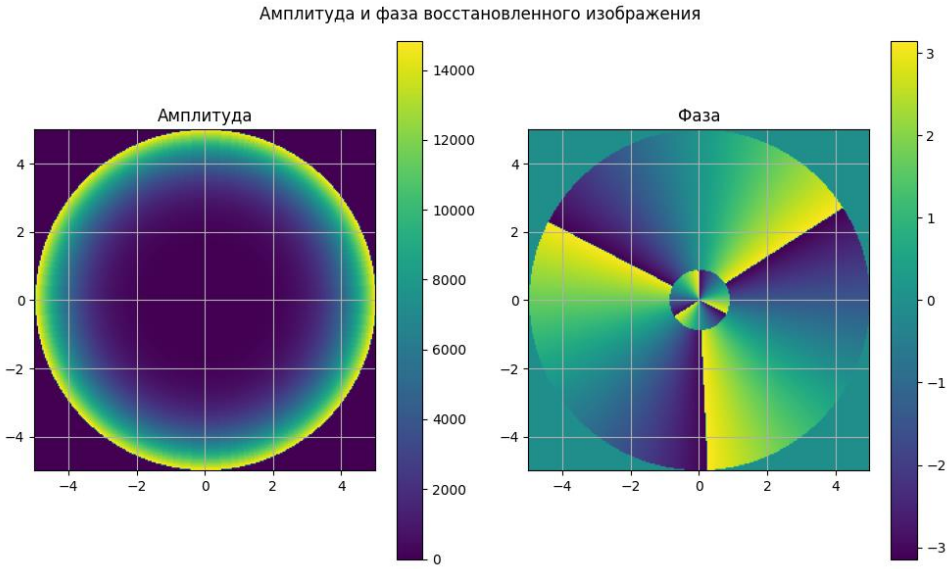

Рисунок 2 – Графики амплитуды и фазы восстановленного изображения полинома Цернике

2.3 Преобразование Ханкеля

Далее будем строить результат одномерного преобразования Ханкеля над входной функцией, используя следующую формулу:

$$F(\rho) = \frac{2\pi}{i^m} \int_0^R f(r) J_m(2\pi r p) r dr.$$

Графики амплитуды и фазы одномерного преобразования Ханкеля над входной функцией приведены на рисунке 3.

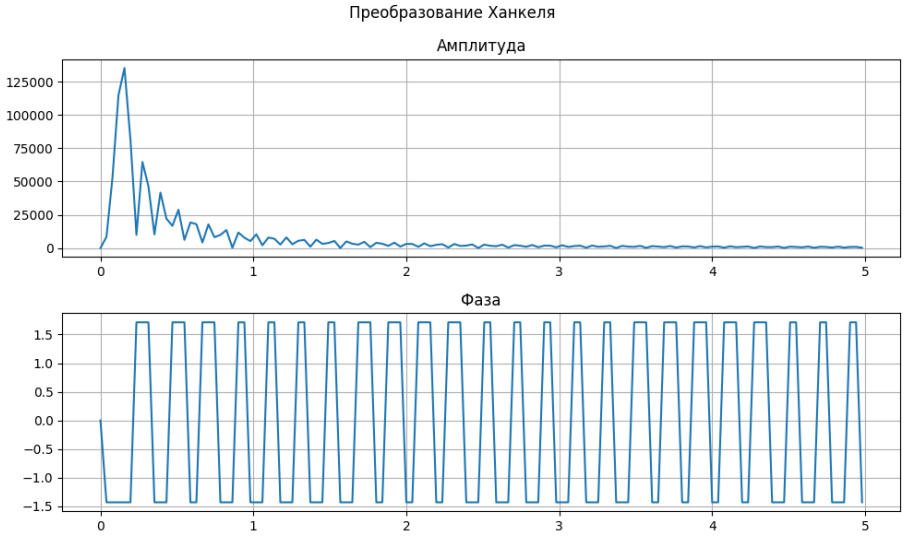

Рисунок 3 - Графики амплитуды и фазы преобразования Ханкеля

Далее по той же формуле восстанавливаем приведенное преобразование Ханкеля в двумерную область. Графики амплитуды и фазы двумерного преобразования Ханкеля над входной функцией приведены на рисунке 4.

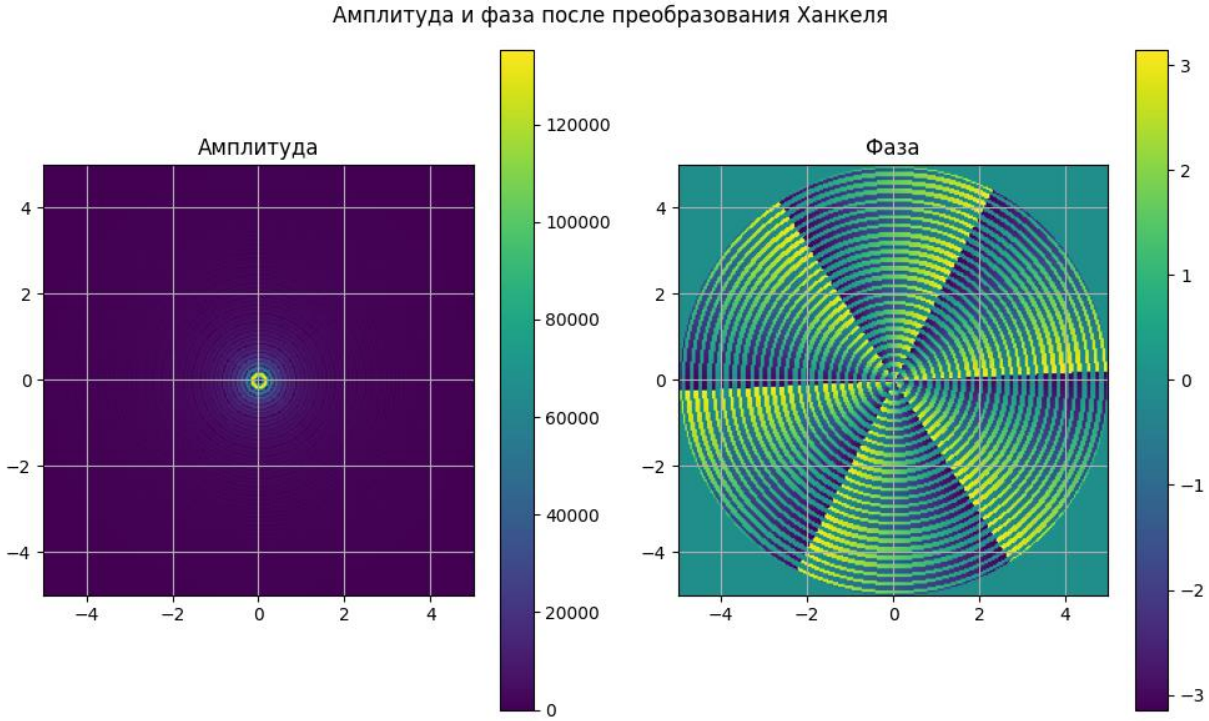

Рисунок 4 – Графики амплитуды и фазы восстановленного изображения преобразования Ханкеля

2.4 Двумерное БПФ

Теперь воспользуемся двумерным финитным преобразованием Фурье, реализованным через быстрое преобразование Фурье. Результат преобразования Фурье восстановленного изображения амплитуды и фазы представлен на рисунке 5.

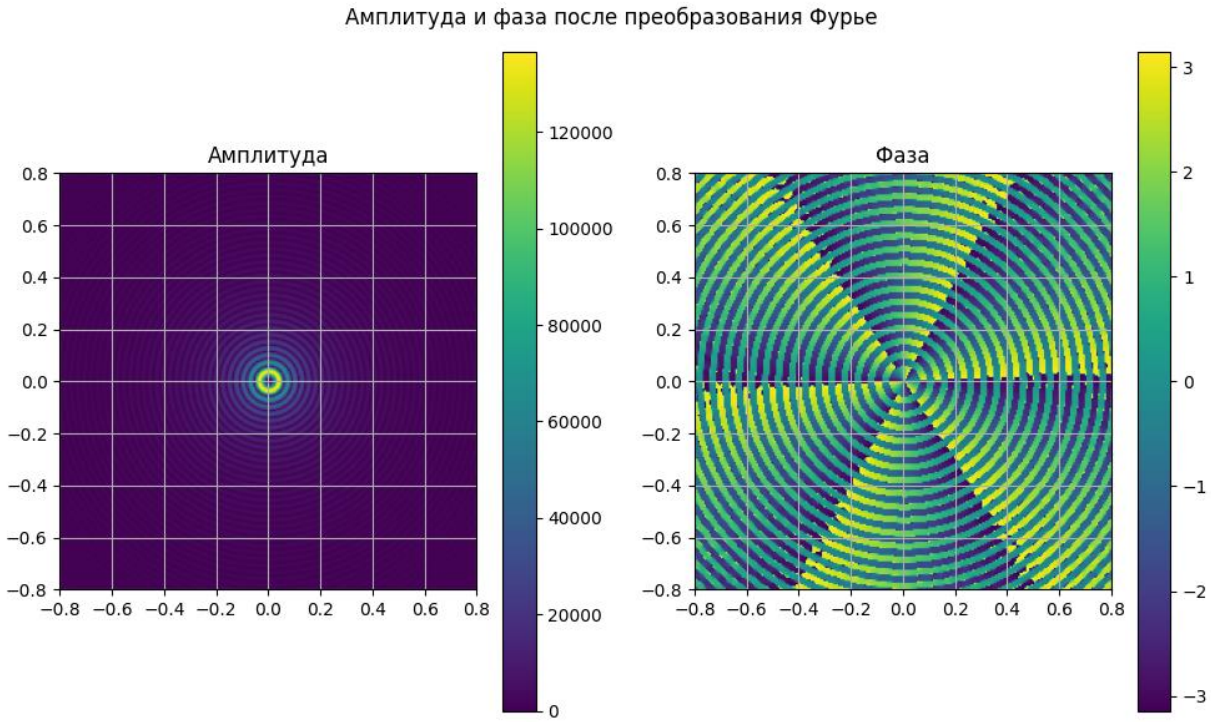

Рисунок 5 – Графики амплитуды и фазы БПФ восстановленного изображения

2.5 Сравнение преобразования Ханкеля и БПФ

В программе необходимо посчитать время работы восстановленного преобразования Ханкеля и преобразования Фурье. Сравним скорости работы двух преобразований.

При получении численных результатов времени вычисления преобразования Фурье параметр M был взят равным 1024. Результаты исследования представлены в таблице 1.

Таблица 1 – Результаты исследования времени работы преобразования Ханкеля и преобразования Фурье.

N	**пр. Ханкеля, с**	**БПФ, с**
64	0,0194	0,0087
128	0,0397	0,0298
256	0,0739	0,1157
512	0,161	0,4657

На рисунке 6 представлено сравнение времени двух преобразований.

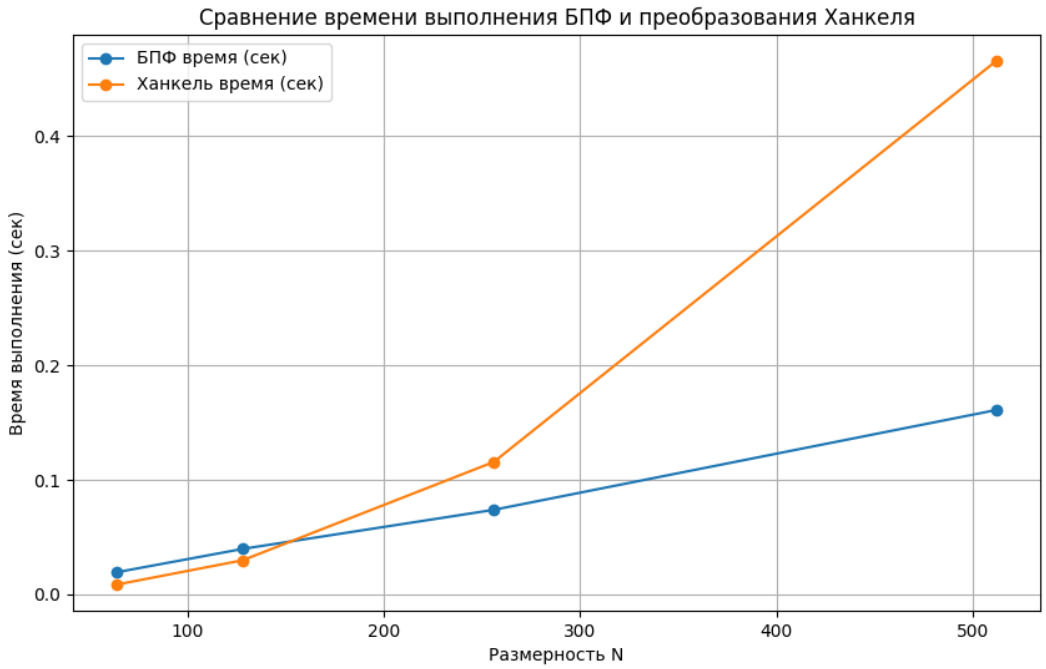

Рисунок 6 – Сравнение времени исполнения преобразования Ханкеля и БПФ

На основе полученных результатов можно сделать вывод о том, что при небольших параметрах N, где N – это число разбиений входной функции, преобразование Ханкеля работает быстрее, чем преобразование Фурье. Однако при увеличении числа точек разбиения, БПФ выполняется в разы быстрее преобразования Ханкеля. Это происходит потому, что преобразование Фурье выигрывает за счёт алгоритмической сложности $O(NlogN)$ в сравнении со сложностью алгоритма Ханкеля $O(N^2)$.

Код программы на языке Python

```
import time
from datetime import datetime
import matplotlib.pyplot as plt
import numpy as np
from scipy.fftpack import fft, fftshift
from scipy.special import factorial, jv
from tabulate import tabulate

def plot_data(data, title, x_values=None, extent=None,
layout='vertical'):
    """Отображение амплитуды и фазы данных с использованием
Matplotlib."""
    fig, axes = plt.subplots(2 if layout == 'vertical' else 1,
if layout == 'vertical' else 2,
                             figsize=(10, 6))
    axes = np.atleast_1d(axes)
    labels = ["Амплитуда", "Фаза"]

    for ax, component, label in zip(axes, [np.abs(data),
np.angle(data)], labels):
        if extent is not None:
            img = ax.imshow(component, extent=extent)
            fig.colorbar(img, ax=ax)
        else:
            ax.plot(x_values, component)
        ax.set_title(label)
        ax.grid(True)

    fig.suptitle(title)
    plt.tight_layout()
    plt.show()

def fft_func(func, M, hx):
    """Выполнение быстрого преобразования Фурье (БПФ)."""
    N = len(func)
    padded_func = np.pad(func, (int((M - N) / 2), int((M - N) /
2)), 'constant')
    transformed_func = fftshift(fft(fftshift(padded_func))) * hx
    return transformed_func[int(M / 2 - N / 2): int(M / 2 + N /
2)]

def fft_2d_func(field, M, hx):
    """обработка двумерного поля через БПФ"""
    field = np.apply_along_axis(fft_func, axis=0, arr=field, M=M,
hx=hx)
    field = np.apply_along_axis(fft_func, axis=1, arr=field, M=M,
hx=hx)
    return field

def radial_p(n, p, r):
    """Расчет радиальных полиномов Цернике."""
    R_np = 0
    for k in range(int((n - p) / 2) + 1):
        R_np += ((-1) ** k * factorial(n - k) /
                 (factorial(k) * factorial((n + p) // 2 - k) *
factorial((n - p) // 2 - k))
                 ) * r ** (n - 2 * k)
    return R_np

def zernike_func(n, m, p, r):
    """Расчет полинома Цернике."""
    return radial_p(n, abs(p), r) * np.exp(1j * m)

def generate_image_from_zernike(zernike, N, m):
    """Восстановление двумерного изображения на основе полинома
Цернике."""
    image = np.zeros((2 * N, 2 * N), dtype=complex)
    for row in range(2 * N):
        for col in range(2 * N):
            alpha = int(round(np.sqrt((row - N) ** 2 + (col - N)
** 2)))
            if alpha < N:
                image[row, col] = zernike[alpha] * np.exp(1j * m *
np.arctan2(col - N, row - N))
    return image

def hankel_transform(zernike, r, hr, m):
    """Выполнение преобразования Ханкеля."""
    start_time = datetime.now()
    X, XI = np.meshgrid(r, r)
    A = (2 * np.pi / (1j ** m)) * jv(m, 2 * np.pi * X * XI) * X
    H = A.dot(zernike) * hr
    print(f'Время выполнения преобразования Ханкеля:
{datetime.now() - start_time} сек')
    return H

def experiment(N_values, m):
    """Измерение времени выполнения для БПФ и преобразования
Ханкеля с выводом в виде таблицы и графика."""
    results = []
    fft_times = []
    hankel_times = []

    for N in N_values:
        # Генерация случайного двумерного массива комплексных
чисел
        image = np.random.rand(N, N) + 1j * np.random.rand(N, N)
        r = np.linspace(0, 5, N)
        hr = 5 / N

        # Время выполнения двумерного БПФ
        start_time = time.perf_counter()
        fft_image = fft_2d_func(image, M=1024, hx=hr)
        fft_time = time.perf_counter() - start_time

        # Время выполнения преобразования Ханкеля
        zernike = zernike_func(5, m, -3, r)
        start_time = time.perf_counter()
        H = hankel_transform(zernike, r, hr, m)
        hankel_time = time.perf_counter() - start_time

        # Сохранение результатов для таблицы
        results.append([N, f"{fft_time:.4f}",
f"{hankel_time:.4f}"])
```

```
        fft_times.append(fft_time)
        hankel_times.append(hankel_time)

    # Выводим результаты в виде таблицы
    print(tabulate(results, headers=["N", "БПФ время (сек)",
"Ханкель время (сек)"], tablefmt="grid"))

    # Построение графика
    plot_results(N_values, fft_times, hankel_times)

def plot_results(N_values, fft_times, hankel_times):
    """Построение графика времени выполнения для БПФ и
преобразования Ханкеля."""
    plt.figure(figsize=(10, 6))
    plt.plot(N_values, fft_times, label="БПФ время (сек)",
marker="o")
    plt.plot(N_values, hankel_times, label="Ханкель время (сек)",
marker="o")
    plt.xlabel("Размерность N")
    plt.ylabel("Время выполнения (сек)")
    plt.title("Сравнение времени выполнения БПФ и преобразования
Ханкеля")
    plt.legend()
    plt.grid()
    plt.show()

def main():
    m = -3  # Порядок полинома Цернике
    N = 128  # число точек в радиусе
    R = 5  # Радиус
    hr = R / N  # Шаг по радиусу
    r = np.linspace(0, R - hr / 2, N)  # Радиусные точки

    # Расчет полинома Цернике и отображение амплитуды и фазы
    zernike = zernike_func(5, m, -3, r)
    plot_data(zernike, "Полином Цернике", x_values=r)

    # Восстановление изображения в двумерный массив
    image = generate_image_from_zernike(zernike, N, m)
    plot_data(image,
              "Амплитуда и фаза восстановленного изображения",
extent=[-R, R, -R, R], layout='horizontal')

    # Преобразование Ханкеля и его отображение
    H = hankel_transform(zernike, r, hr, m)
    plot_data(H, "Преобразование Ханкеля", x_values=r)

    # Восстановление изображения из преобразования Ханкеля
    image_hankel = generate_image_from_zernike(H, N, m)
    plot_data(image_hankel,
              "Амплитуда и фаза после преобразования Ханкеля",
extent=[-R, R, -R, R], layout='horizontal')

    # Двумерное преобразование Фурье через БПФ
    M = 1024
    b = N ** 2 / (4 * R * M)
    start_time = datetime.now()

    # Преобразование по строкам и столбцам
    for row in range(image.shape[0]):
        image[row, :] = fft_func(image[row, :], M, hr)
    for col in range(image.shape[1]):
        image[:, col] = fft_func(image[:, col], M, hr)

    print(f'Время выполнения преобразования Фурье: {datetime.now()
- start_time} сек')

    # Отображение результатов преобразования Фурье
    plot_data(image,
              "Амплитуда и фаза после преобразования Фурье",
extent=[-b, b, -b, b], layout='horizontal')

    N_values = [64, 128, 256, 512]
    experiment(N_values, m)

if __name__ == "__main__":
    main()
```