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

Отчёт по лабораторной работе №3
Дисциплина: «Оптическая информатика»

РЕАЛИЗАЦИЯ ОПТИЧЕСКОГО ПРЕОБРАЗОВАНИЯ ХАНКЕЛЯ ДЛЯ РАДИАЛЬНО-ВИХРЕВЫХ ПУЧКОВ

Выполнил: Гуторов Владислав
Сергеевич
Группа: 6402-010302D
Вариант 15

Самара 2024

Задание

1. Выбрать входную функцию $f(r, \varphi) = f(r)e^{im\varphi}$ 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 $f(r)e^{im\varphi}$ 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 $F(\rho)$ $\exp(im\theta)$ и построить её изображение.
4. Реализовать двумерное преобразование Фурье через БПФ. Применить его ко входной двумерной функции $f(r)e^{im\varphi}$. 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.

15	Мода Гаусса-Лагерра	$\|GL_{3,-2}(r, \varphi)\|e^{-3i\varphi}$	$m = -3$

Результаты работы

На рисунке 1 представлена амплитуда и фаза мода Гаусса-Лагерра.

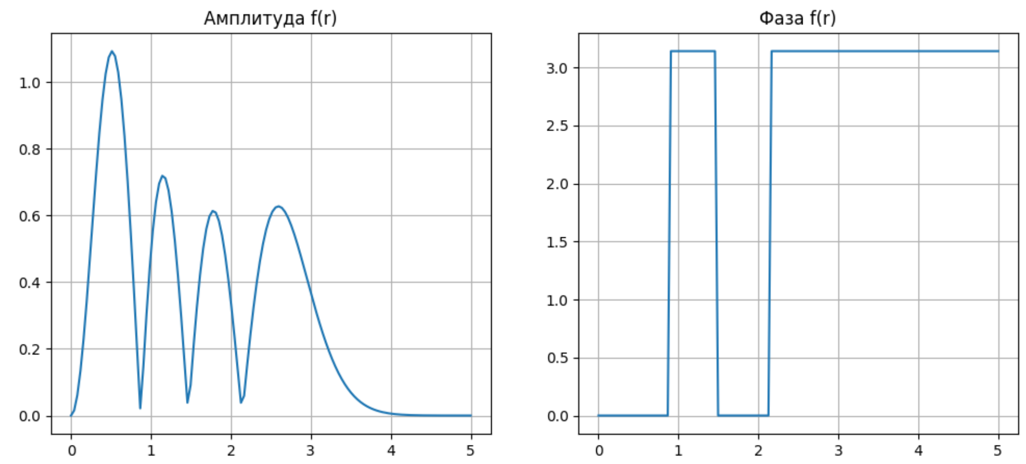

Рисунок 1 – Амплитуда и фаза входной функции

Построим теперь график восстановленного изображения, полученного при помощи раскрутки входной функции. Результат (амплитуда и фаза) представлен на рисунке 2.

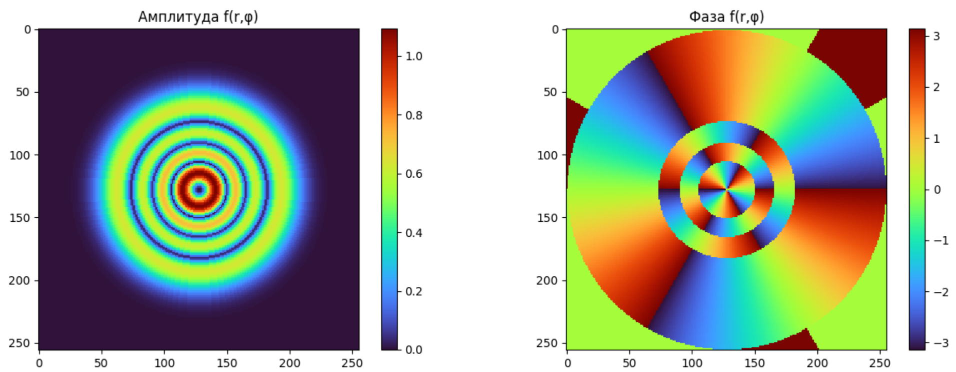

Рисунок 2 – Амплитуда и фаза восстановленного изображения

Построим результат одномерного преобразования Ханкеля (Рисунок 3).

Формула:

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

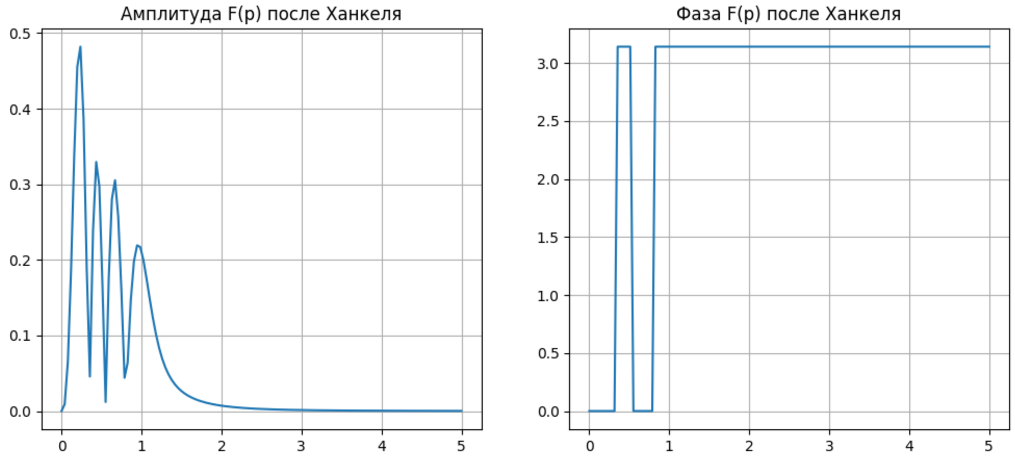

Рисунок 3 – Амплитуда и фаза после преобразования Ханкеля

Восстановим преобразование Ханкеля в двумерную область (Рисунок 4).

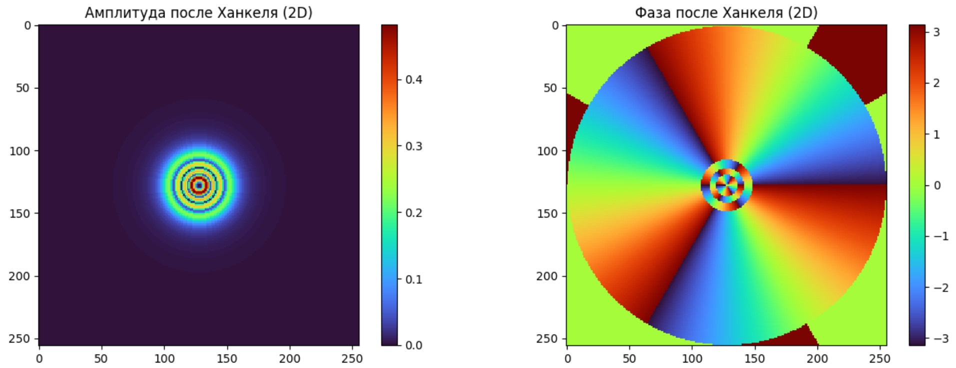

Рисунок 4 – Амплитуда и фаза восстановленного преобразования Ханкеля

Воспользуемся двумерным финитным преобразованием Фурье, реализованным через быстрое преобразование Фурье (Рисунок 5).

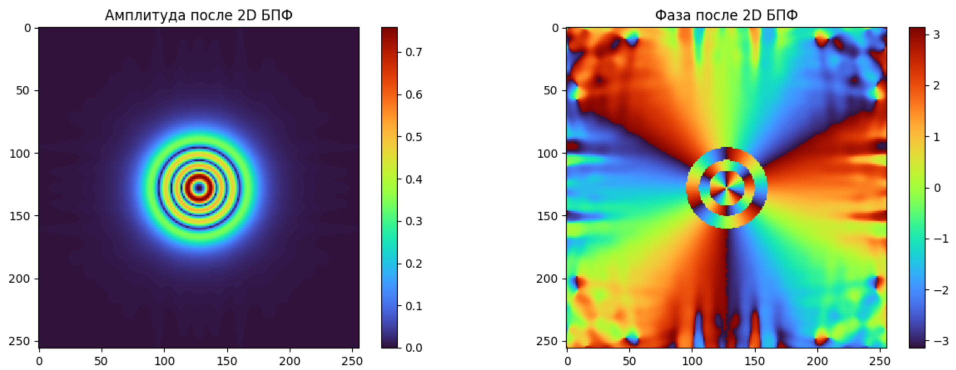

Рисунок 5 – Амплитуда и фаза преобразования Фурье

Таблица 1 – Сравнение времени выполнения преобразования Ханкеля и Фурье

N	M	Время выполнения преобразования Ханкеля, с	Время выполнения преобразования Фурье, с
64	1024	0.0034	0.0071
128	1024	0.0126	0.0152
256	1024	0.0481	0.0329
512	1024	0.1881	0.0846

Выводы

При малых размерах N Ханкель-преобразование может быть выполнено быстрее, однако при увеличении разрешения (росте N) преимущество переходит к 2D-преобразованию Фурье. Это связано с лучшей асимптотической сложностью FFT-алгоритмов по сравнению с численным интегрированием для преобразования Ханкеля. Так же стоит отметить, что чем больше N, тем более точным будет численное интегрирование и тем лучше будет разрешение в радиальном пространстве.

Код

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.special import genlaguerre, jv
import time

# Параметры задачи для варианта №15
# Мода Гаусса-Лагерра GL_{3,-2}(r,φ) * exp(-3iφ)
# Здесь p=3, l=-2, m=-3 (угловой момент)
p = 3
l = -2
m = -3
R = 5      # Радиус ограниченной области
N = 64   # Число точек дискретизации в радиальном направлении
M = 1024  # Число точек для дополнения при БПФ
w = 1.0   # Параметр гауссовой моды (плечо пучка), можно взять равным 1

# Определение радиальной части Гаусса-Лагерра GL_{p,l}(r).
# Здесь L_p^{α}(x) - полином Лагерра, реализуем через genlaguerre(p, α)
def GL_radial(r, p, l, w=1.0):
    alpha = abs(l)
    L = genlaguerre(p, alpha)
    return (r/w)**alpha * L(2*(r**2)/(w**2)) * np.exp(-(r**2)/(w**2))

# Исходная функция f(r, φ) = GL_{3,-2}(r) * exp(-3 i φ)
# В данной реализации мы сначала формируем одномерный профиль f(r) = GL_{3,-
2}(r),
# а угловая часть будет учитываться при восстановлении 2D изображения.
def func(r):
    return GL_radial(r, p, l, w)

# Восстановление двумерного массива из радиального профиля
# Здесь учитываем угловой множитель exp(i*m*φ)
def to2d(y):
    n = len(y)
    arr_2d = np.zeros((2 * n, 2 * n), dtype=complex)
    xs, ys = np.meshgrid(np.arange(0, 2 * n), np.arange(0, 2 * n))
    xs = xs - n
    ys = ys - n
    dist = np.round(np.sqrt(xs**2 + ys**2)).astype(int)
    mask = dist < n
    arr_2d[mask] = y[dist[mask]]
    fi = np.arctan2(ys, xs)
    # Добавляем угловой множитель exp(i*m*φ)
    return arr_2d * np.exp(1j * m * fi)

# Одномерное преобразование Фурье с конечной областью (аналог БПФ)
def finite_fft(y, b, a, M, N):
    h = (b - a) / (N - 1)
    zeros = np.zeros(int((M - N) / 2))
    y = np.concatenate((zeros, y, zeros), axis=None)
    middle = int(len(y) / 2)
    y = np.concatenate((y[middle:], y[:middle]))
    Y = np.fft.fft(y, axis=-1) * h
    middle = int(len(Y) / 2)
    Y = np.concatenate((Y[middle:], Y[:middle]))
    Y = Y[int((M - N) / 2): int((M - N) / 2 + N)]
    interval = abs(N ** 2 / (4 * b * M))
    return Y, interval

# Двумерное преобразование (аналог БПФ 2D)
def finite_fft_2d(Z, a, b, N, M):
```

```
    # Применяем преобразование по каждой строке
    for i in range(N):
        Z[:, i], area = finite_fft(Z[:, i], b, a, M, N)
    # Применяем преобразование по каждому столбцу
    for i in range(N):
        Z[i, :], area = finite_fft(Z[i, :], b, a, M, N)
    return Z, area

# Численное преобразование Ханкеля для m-ого порядка
# Здесь используем численное интегрирование методом левых прямоугольников
def hankel_transform(x, y, m):
    # Шаг интегрирования
    dr = x[1] - x[0]
    Y = np.zeros(len(x), dtype=np.complex128)
    for j in range(len(x)):
        p_val = x[j]
        # Интегрирование: sum(y(r)*J_m(2п p r)*r*dr)
        integrand = y * jv(m, 2 * np.pi * p_val * x) * x
        Y[j] = np.sum(integrand) * dr
    return Y

if __name__ == '__main__':
    # Шаг 1: Формируем сетку по r
    x = np.linspace(0, R, N)
    f_r = func(x)

    # Построение графиков амплитуды и фазы f(r) (задание 1)
    plt.figure(figsize=(12,5))
    plt.subplot(1,2,1)
    plt.plot(x, np.abs(f_r))
    plt.grid(True)
    plt.title('Амплитуда f(r)')
    plt.subplot(1,2,2)
    plt.plot(x, np.angle(f_r))
    plt.grid(True)
    plt.title('Фаза f(r)')
    plt.show()

    # Восстановление изображения (задание 2)
    f_image = to2d(f_r)
    fig, arr = plt.subplots(1, 2, figsize=(15, 5))
    amp = arr[0].imshow(np.abs(f_image), cmap='turbo',
interpolation='nearest')
    arr[0].set_title('Амплитуда f(r,φ)')
    fig.colorbar(amp, ax=arr[0])
    phase = arr[1].imshow(np.angle(f_image), cmap='turbo',
interpolation='nearest')
    arr[1].set_title('Фаза f(r,φ)')
    fig.colorbar(phase, ax=arr[1])
    plt.show()

    # Преобразование Ханкеля (задание 3)
    start = time.time()
    f_hankel = hankel_transform(x, f_r, m)
    end = time.time()
    print("Время выполнения преобразования Ханкеля: {} сек".format(end -
start))

    # Графики амплитуды и фазы F(p)
    plt.figure(figsize=(12,5))
    plt.subplot(1,2,1)
    plt.plot(x, np.abs(f_hankel))
    plt.grid(True)
    plt.title('Амплитуда F(p) после Ханкеля')
```

```
    plt.subplot(1,2,2)
    plt.plot(x, np.angle(f_hankel))
    plt.grid(True)
    plt.title('Фаза F(p) после Ханкеля')
    plt.show()

    # Восстановление двумерной функции F(p) e^{i m φ}
    F_image = to2d(f_hankel)
    fig, arr = plt.subplots(1, 2, figsize=(15, 5))
    amp = arr[0].imshow(np.abs(F_image), cmap='turbo',
interpolation='nearest')
    arr[0].set_title('Амплитуда после Ханкеля (2D)')
    fig.colorbar(amp, ax=arr[0])
    phase = arr[1].imshow(np.angle(F_image), cmap='turbo',
interpolation='nearest')
    arr[1].set_title('Фаза после Ханкеля (2D)')
    fig.colorbar(phase, ax=arr[1])
    plt.show()

    # Задание 4: Реализация 2D-преобразования Фурье через БПФ
    # Увеличиваем число точек в связи с расширением области 2D массива
    N2 = f_image.shape[0]
    start = time.time()
    f_bpf, _ = finite_fft_2d(np.copy(f_image), 0, R, N2, M)
    end = time.time()
    print("Время выполнения двухмерного БПФ: {} сек".format(end - start))

    fig, arr = plt.subplots(1, 2, figsize=(15, 5))
    amp = arr[0].imshow(np.abs(f_bpf), cmap='turbo', interpolation='nearest')
    arr[0].set_title('Амплитуда после 2D БПФ')
    fig.colorbar(amp, ax=arr[0])
    phase = arr[1].imshow(np.angle(f_bpf), cmap='turbo',
interpolation='nearest')
    arr[1].set_title('Фаза после 2D БПФ')
    fig.colorbar(phase, ax=arr[1])
    plt.show()
```