Министерство науки и высшего образования Российской Федерации

Лабораторная работа № 2

«Моделирование формирования изображения при когерентном и некогерентном освещении для идеальной оптической системы»

Выполнил: Леко А.А..

Группа: Q4110

Проверила:

Иванова Т. В.

Санкт-Петербург 2023

Задание:

Создать программу для моделирования формирования изображения при когерентном и некогерентном освещении.

Для периодической решетки:

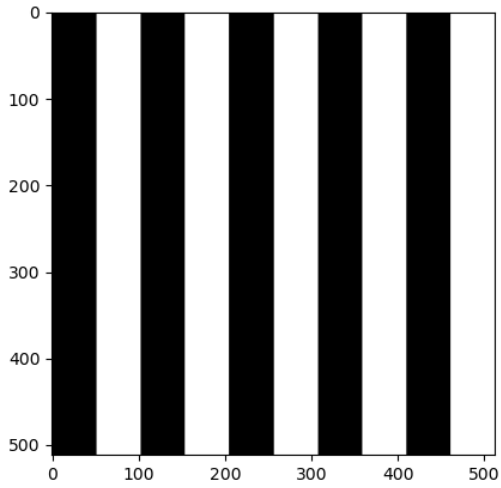

Рисунок 1 – Объект (периодическая решетка)

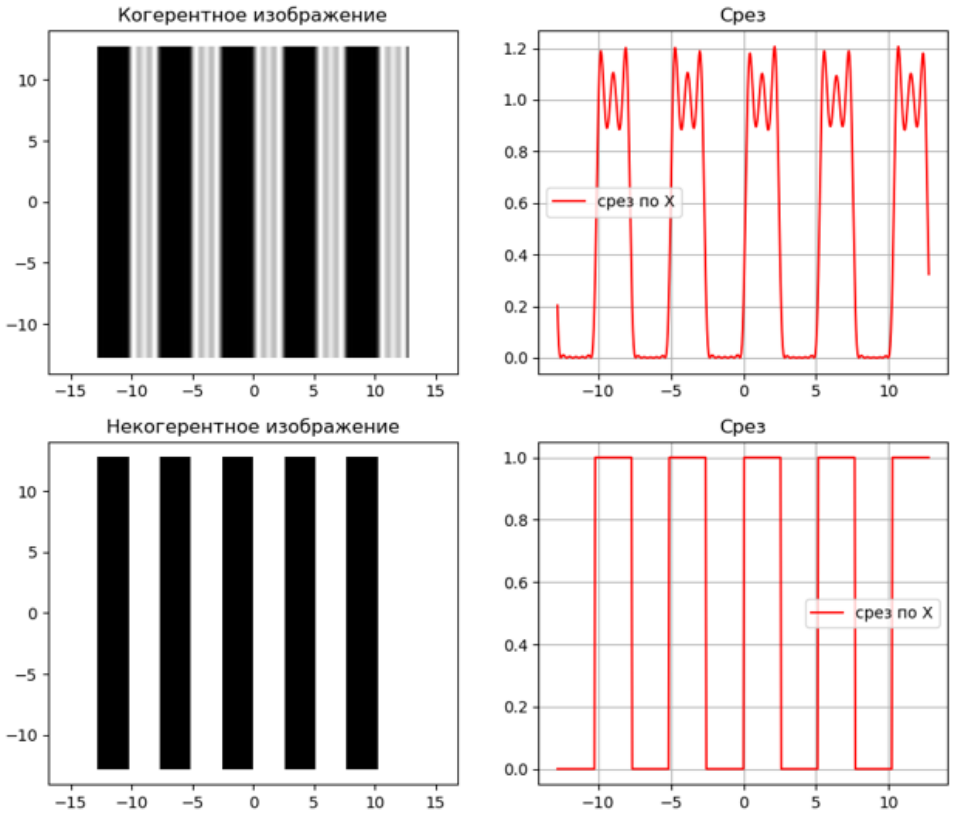

Рисунок 2 – Когерентное и некогерентное изображение и срезы по x

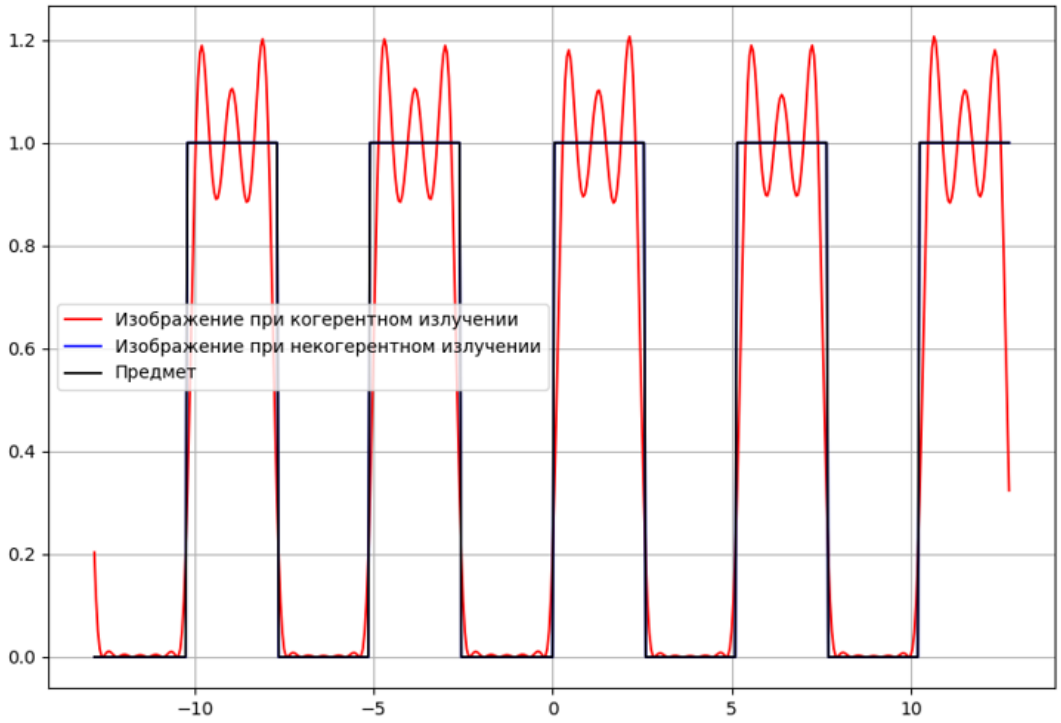

Рисунок 3 – Срезы по x

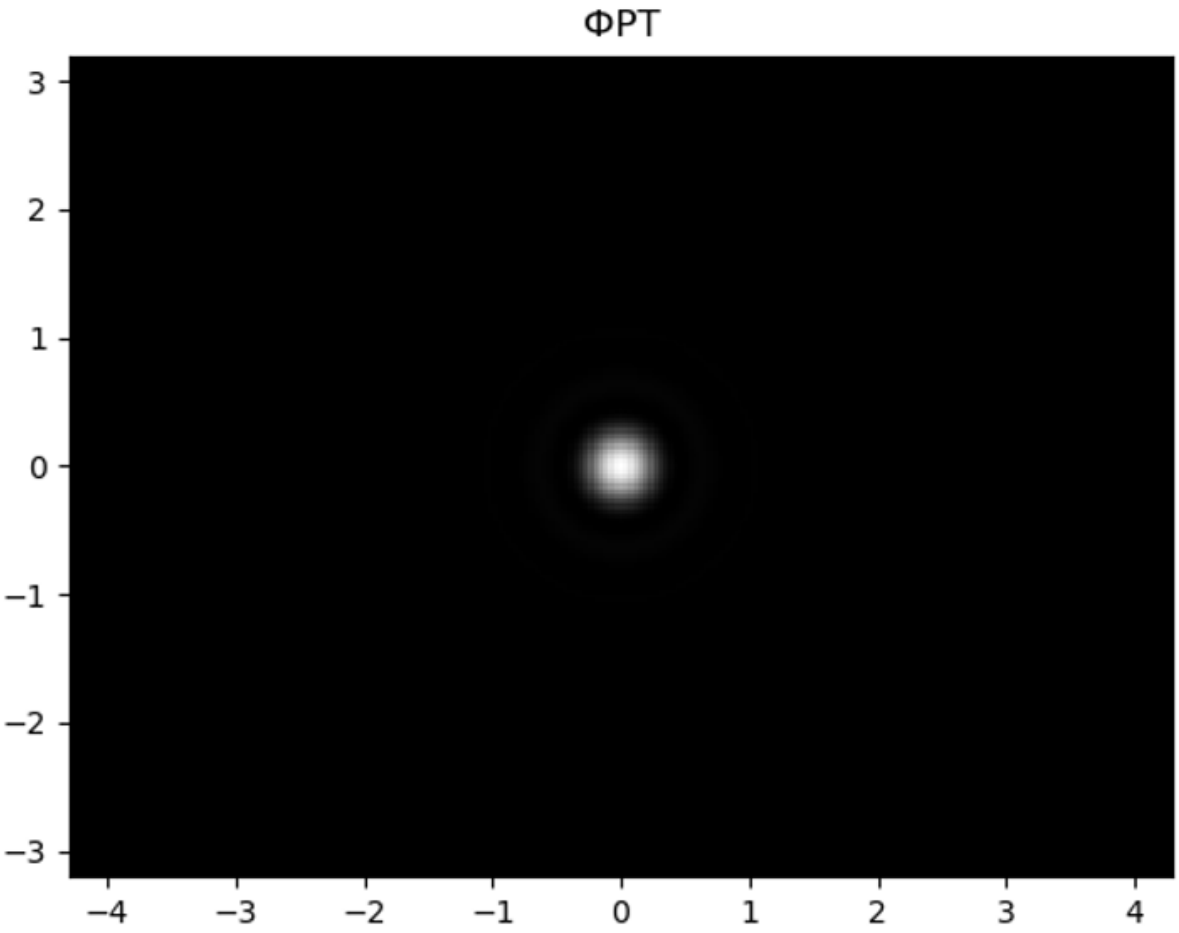

Рисунок 4 – ФРТ

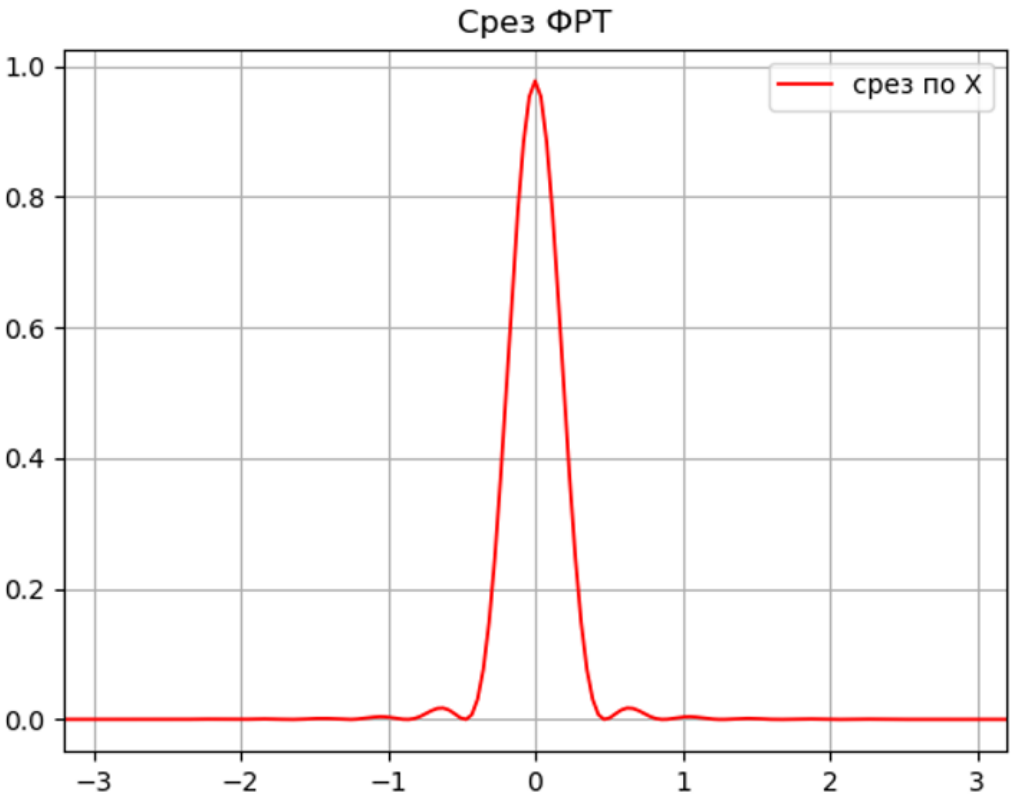

Рисунок 5 – Срез ФРТ

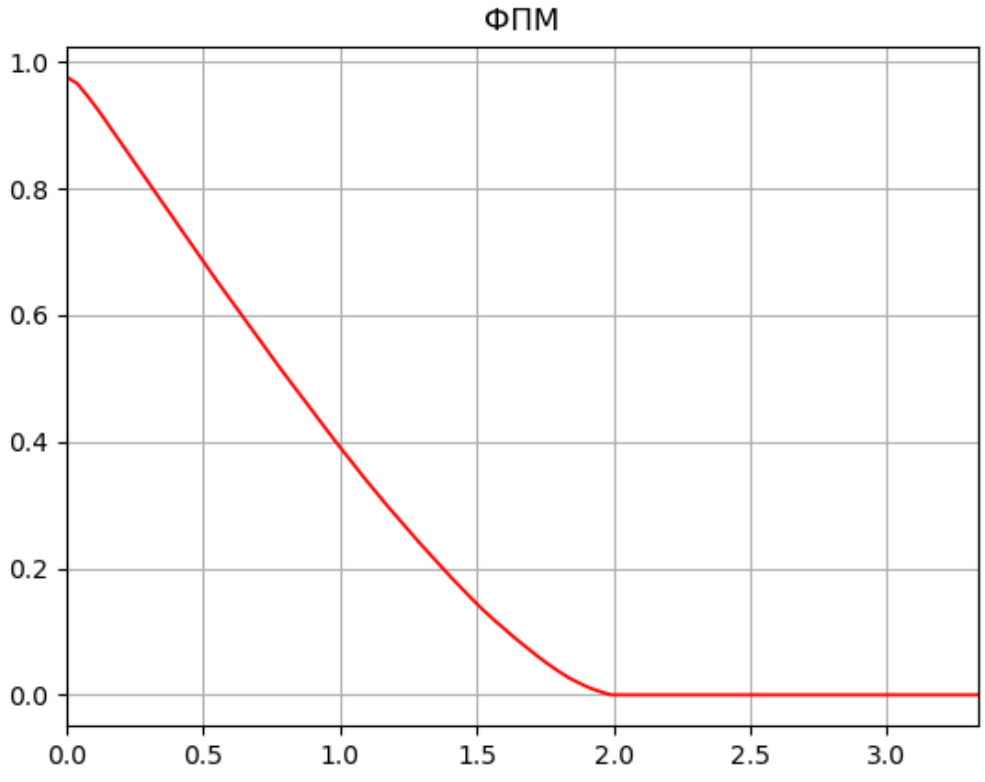

Рисунок 6 – ФПМ

Для сложного изображения:

Рисунок 7 – Исходное изображение

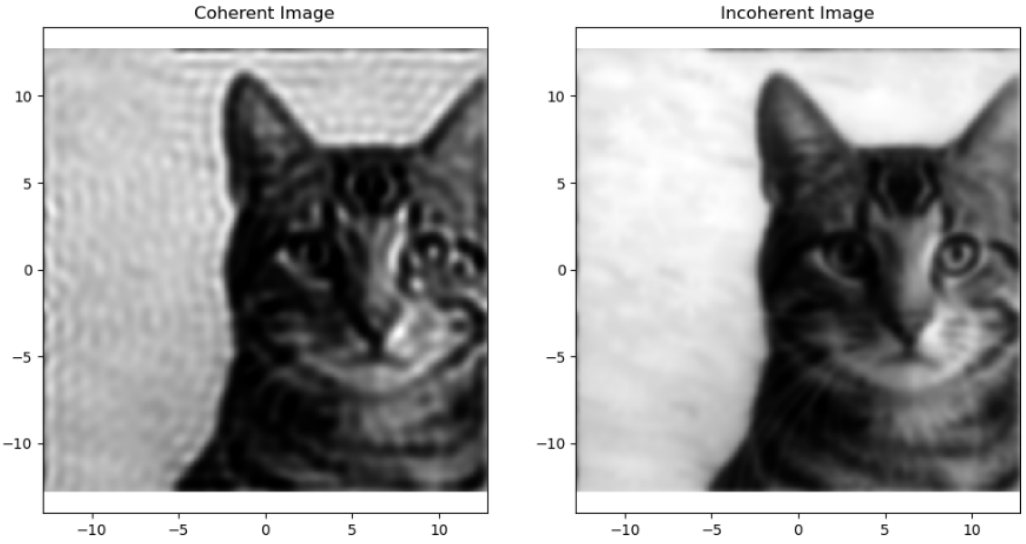

Рисунок 8 – Изображение в когерентном и некогерентном излучении

Вывод:

В ходе работы разработаны программы для моделирования когерентного и некогерентного освещения изображения. Для периодической решетки вычислены ФРТ и ФПМ.

Текст программ:

```
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image




# Load the image
image = plt.imread('cat.jpeg')

# Determine the image dimensions
height, width, _ = image.shape

# Determine the minimum size of the image
minSize = min(height, width)

# Crop the image to a square size
croppedImage = image[:minSize-1, :minSize-1, :]

# Resize the image to [512, 512]
resizedImage =
np.array(Image.fromarray(croppedImage).resize((512, 512)))
item = np.mean(resizedImage, axis=-1)  # Convert to grayscale

N = 512
A = 0.5
lambda_val = 0.5
D_zr = 20
step_zr = D_zr / N
step_it = 1 / (N * step_zr)
step_im = step_it * lambda_val / A
[im_axis_X, im_axis_Y] = np.meshgrid(
    np.arange(-(N/2) * step_im, (N/2) * step_im, step_im),
    np.arange(-(N/2) * step_im, (N/2) * step_im, step_im)
)

pupil = np.zeros((N, N))
x, y = np.meshgrid(
    np.arange(-(N/2) * step_zr, (N/2-1) * step_zr, step_zr),
    np.arange(-(N/2) * step_zr, (N/2-1) * step_zr, step_zr)
)

for i in range(N-1):
    for j in range(N-1):
        if x[i, j]**2 + y[i, j]**2 < 1:
            pupil[i, j] = 1

# Coherent image
fft_item = 1/N *
```

```
np.fft.fftshift(np.fft.fft2(np.fft.fftshift(item)))
res = N * np.fft.fftshift(np.fft.ifft2(np.fft.fftshift(fft_item
* pupil)))
res = np.abs(res)**2
res = res[::-1, :]


# Incoherent image
fft_intensity = 1 / N *
np.fft.fftshift(np.fft.fft2(np.fft.fftshift(np.abs(item))))
fft_func_rasp = 0.25 * N *
np.fft.fftshift(np.fft.fft2(np.fft.fftshift(np.abs(np.fft.fftshi
ft(np.fft.ifft2(np.fft.fftshift(pupil)))) ** 2)))

func_rasp_img = fft_intensity * fft_func_rasp
intensity_rasp_img = N *
np.fft.fftshift(np.fft.ifft2(np.fft.fftshift(func_rasp_img)))
intensity_rasp_img = np.abs(intensity_rasp_img) ** 2
intensity_rasp_img = intensity_rasp_img[::-1, :]

# Plot results
plt.figure(figsize=(12, 6))

plt.subplot(1, 2, 1)
plt.pcolormesh(im_axis_X, im_axis_Y, res, cmap='gray')
plt.axis('equal')
plt.title("Coherent Image")

plt.subplot(1, 2, 2)
plt.pcolormesh(im_axis_X, im_axis_Y, intensity_rasp_img,
cmap='gray')
plt.axis('equal')
plt.title("Incoherent Image")

plt.show()
import numpy as np
import matplotlib.pyplot as plt



def create_striped_image(count):
    N = 512
    w = N / (count*2)
    image = np.ones((N, N))  # Initialize with white background
    for i in range (N):
        for j in range (N):
            if (j // w) % 2 == 0:
                image[i, j] = 0

    return image


N = 512
```

```
item = create_striped_image(5)
plt.figure(2)
plt.imshow(item, cmap='gray')
A = 0.5
lambda_val = 0.5
D_zr = 20
step_zr = D_zr / N
step_it = 1 / (N * step_zr)
step_im = step_it * lambda_val / A
im_axis_X, im_axis_Y = np.meshgrid(
    np.arange(-(N / 2) * step_im, (N / 2) * step_im, step_im),
    np.arange(-(N / 2) * step_im, (N / 2) * step_im, step_im)
)

zrachok = np.zeros((N, N))
x, y = np.meshgrid(
    np.arange(-(N / 2) * step_zr, (N / 2) * step_zr, step_zr),
    np.arange(-(N / 2) * step_zr, (N / 2) * step_zr, step_zr)
)

for i in range(N):
    for j in range(N):
        if x[i, j]**2 + y[i, j]**2 < 1:
            zrachok[i, j] = 1

# когерентное изображение
fft_item = 1 / N *
np.fft.fftshift(np.fft.fft2(np.fft.fftshift(item)))
res = N * np.fft.fftshift(np.fft.ifft2(np.fft.fftshift(fft_item
* zrachok)))
res = np.abs(res)**2
res = np.flipud(res)

# некогерентное изображение
# для периодической решетки - раскомментировать эту строку:
# для периодической решетки - раскомментировать эту строку:
fft_intens = 1 / N *
np.fft.fftshift(np.fft.fft2(np.fft.fftshift(np.abs(item)**2)))


func_rasp_img = fft_intens
intens_rasp_img = N *
np.fft.fftshift(np.fft.ifft2(np.fft.fftshift(func_rasp_img)))
intens_rasp_img = np.flipud(intens_rasp_img)

# Ensure intens_rasp_img is real-valued
intens_rasp_img = np.abs(intens_rasp_img)


plt.figure(3)
plt.subplot(2, 3, 1)
plt.pcolormesh(im_axis_X, im_axis_Y, res, cmap='gray')
plt.axis('equal')
```

```
plt.title("Когерентное изображение")
plt.subplot(2, 3, 2)
plt.plot(im_axis_X[N // 2 + 1, :], res[N // 2 + 1, :], 'r',
linewidth=1.3)
plt.grid(True)
plt.legend(['срез по X'])
plt.title("Срез")
plt.subplot(2, 3, 4)
plt.pcolormesh(im_axis_X, im_axis_Y, intens_rasp_img,
cmap='gray')
plt.axis('equal')
plt.title("Некогерентное изображение")
plt.subplot(2, 3, 5)
plt.plot(im_axis_X[N // 2 + 1, :], intens_rasp_img[N // 2 + 1,
:], 'r', linewidth=1.3)
plt.legend(['срез по X'])
plt.grid(True)
plt.title("Срез")

[p_x, p_y] = np.meshgrid(
    np.arange(-(N / 2) * step_zr, (N / 2) * step_zr, step_zr),
    np.arange(-(N / 2) * step_zr, (N / 2) * step_zr, step_zr)
)

n_max = step_it * N / 2
x_max = step_im * N / 2
p_max = step_zr * N / 2

FRT_ = (step_zr / step_it) *
(np.fft.fftshift(np.fft.ifft2(np.fft.fftshift(zrachok))) * N)
FRT_abs = (np.abs(FRT_) * np.abs(FRT_)) / (np.pi ** 2)  # функ-
ция рассеяния точки
D = (step_it / step_zr) *
(np.fft.fftshift(np.fft.fft2(np.fft.fftshift(FRT_abs))) / N)  #
ОПФ
D_norm = D * np.pi
D_abs = np.abs(D_norm)
# ФРТ
FRT = (np.abs(FRT_) * np.abs(FRT_)) / (np.pi ** 2)

# Для решетки - раскомментировать вывод всех последующих графи-
ков:
# % Срез ФРТ, ФРТ
plt.figure(5)
plt.plot(x[N // 2 + 1, :], FRT[N // 2 + 1, :], color='r',
linewidth=1.3)
plt.xlim([-x_max / 4, x_max / 4])
plt.grid(True)
plt.legend(['срез по X'])
plt.title('Срез ФРТ')

plt.figure(6)
plt.pcolormesh(x, y, FRT, cmap='gray')
```

```
plt.axis('equal')
plt.axis([-x_max / 4, x_max / 4, -x_max / 4, x_max / 4])
plt.title('ФРТ')

# ФПМ
plt.figure(7)
plt.plot(p_x[N // 2 + 1, :], D_abs[N // 2 + 1, :], color='r',
linewidth=1.3)
plt.xlim([0, p_max / 3])
plt.grid(True)
plt.title('ФПМ')

plt.figure(8)
plt.plot(im_axis_X[N // 2 + 1, :], res[N // 2 + 1, :], 'r',
linewidth=1.3)
#plt.hold(True)
plt.plot(im_axis_X[N // 2 + 1, :], intens_rasp_img[N // 2 + 1,
:], 'b', linewidth=1.3)
#plt.hold(True)
plt.plot(im_axis_X[N // 2 + 1, :], item[N // 2 + 1, :], 'black',
linewidth=1.3)
plt.grid(True)
plt.legend(['Изображение при когерентном излучении', 'Изображе-
ние при некогерентном излучении', 'Предмет'])
plt.show()
```