Министерство науки и высшего образования Российской Федерации

Лабораторная работа № 3

«Изучение влияния различных типов аберраций на качество оптического изображения»

Выполнил: Леко А.А..

Группа: Q4110

Проверила:

Иванова Т. В.

Санкт-Петербург 2024

Задание:

Создать программу, моделирующую влияние аберраций различного типа на ФРТ, ОПФ и произвольное изображение при некогерентном освещении.

Оптическая система:

- Зрачок круглый
- Аберрации – один из членов разложения волновой аберрации в ряд по полиномам Цернике (сферическая аберрация 3 порядка)
- Пропускание равномерно по зрачку

Величина коэффициента разложения аберрации в ряд подбирается таким образом, чтобы в итоге получились следующие результаты:

- Аберрация отсутствует
- Число Штреля ≈ 0.8
- Число Штреля ≈ 0.5
- Контраст 0.2 при частоте s ≈ 1
- Контраст 0.2 при частоте s ≈ 0.8

Результаты:

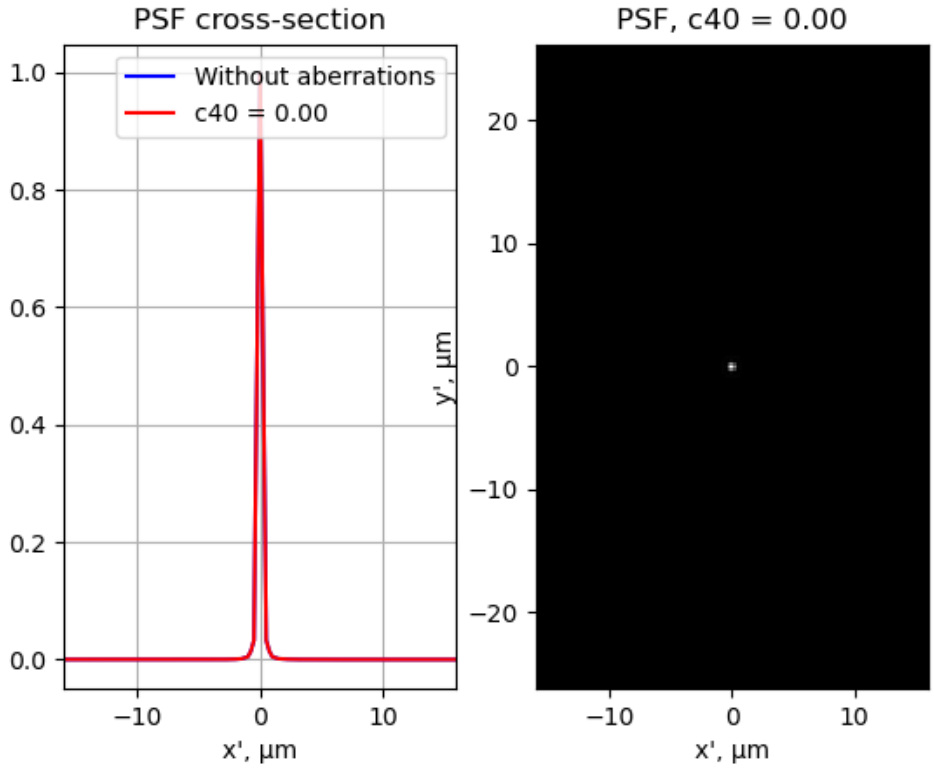

Рисунок 1 – Срез ФРТ без аберраций

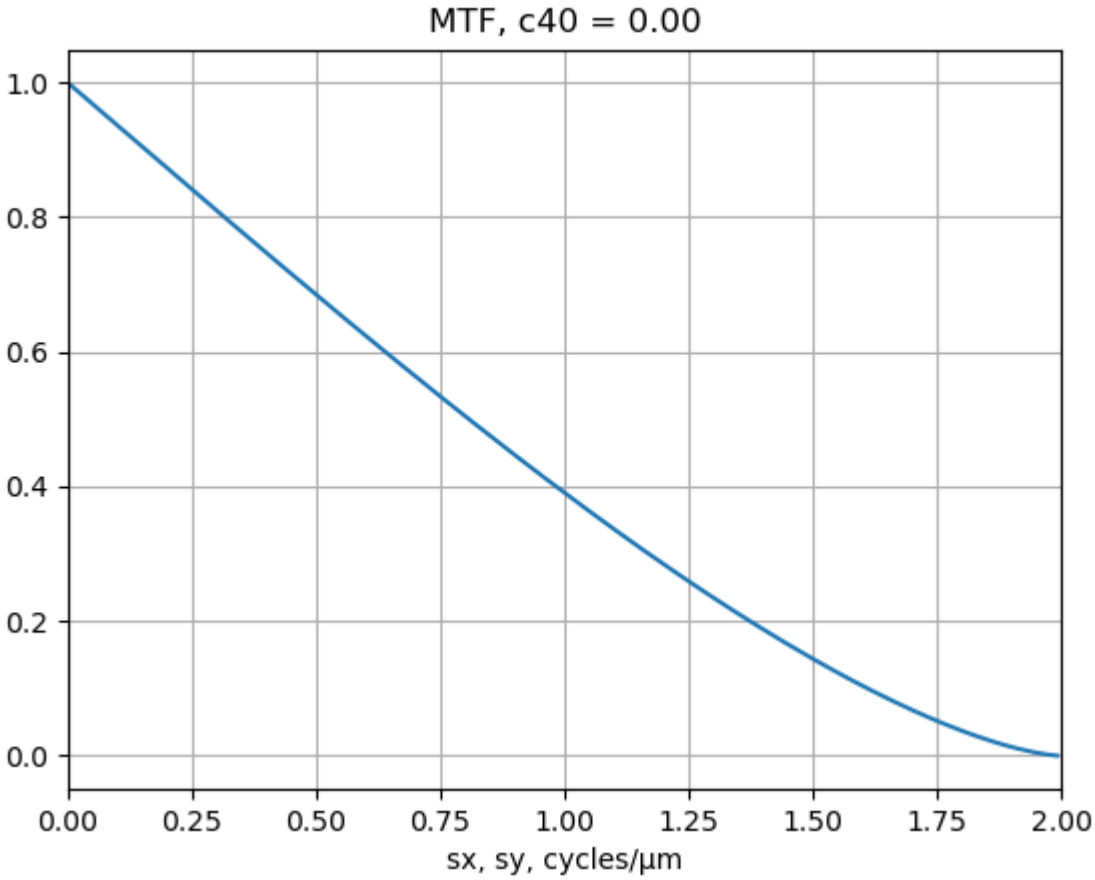

Рисунок 2 – ФПМ без аберраций

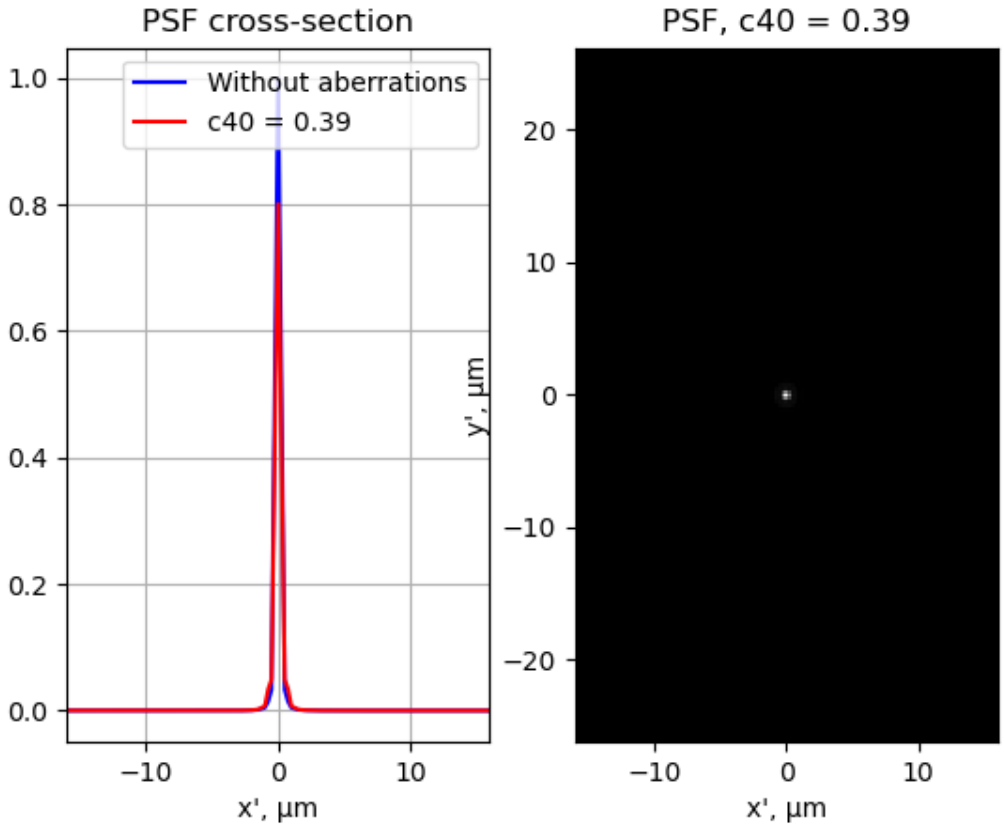

Рисунок 3 – Срез ФРТ при сферической аберрации 3-го порядка, st = 0.8

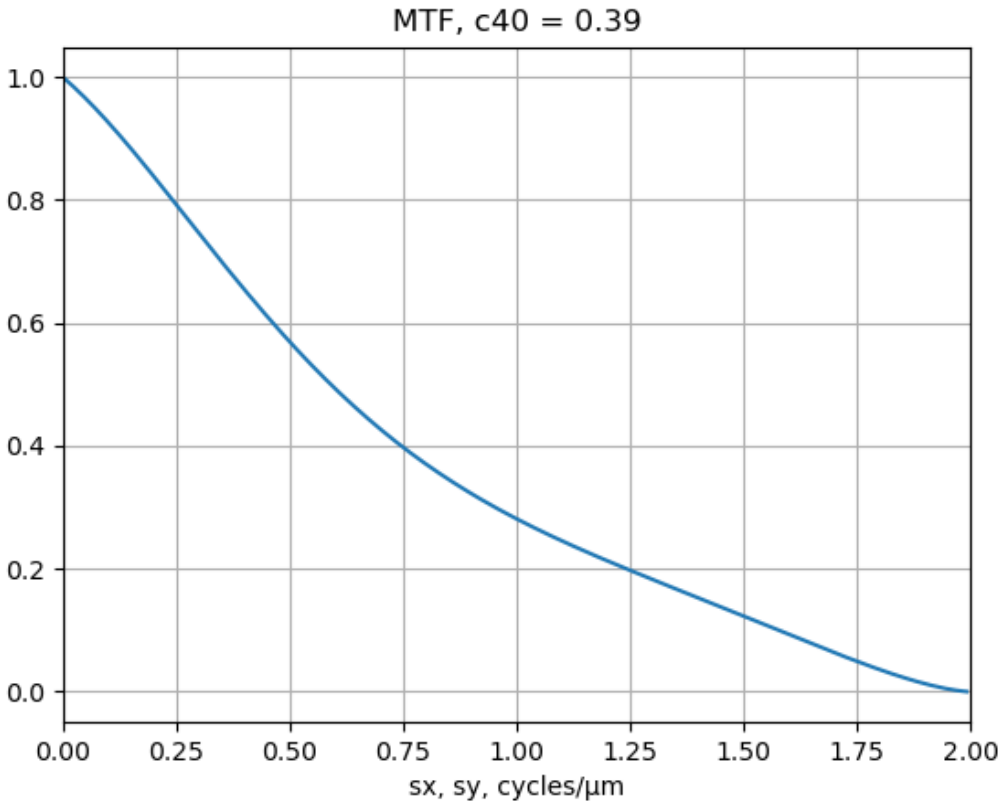

Рисунок 4 – ФПМ при сферической аберрации 3-го порядка st = 0.8

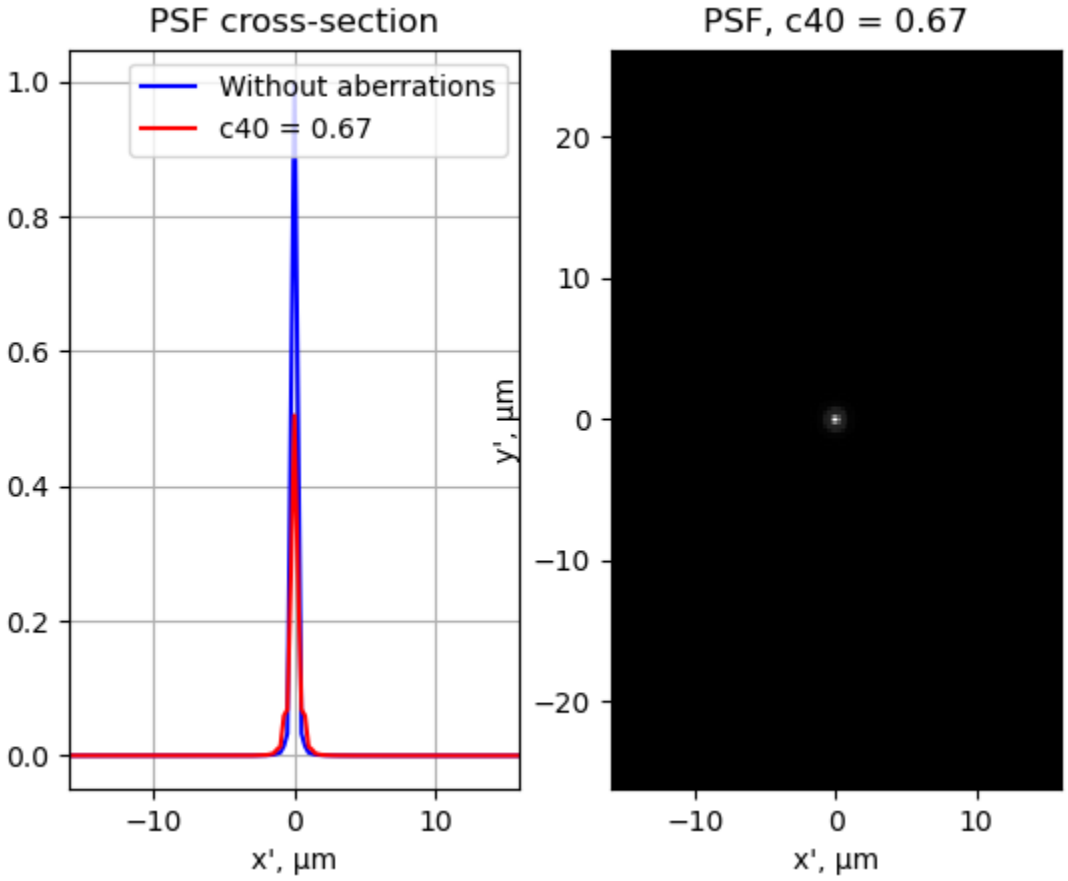

Рисунок 5 – Срез ФРТ при сферической аберрации 3-го порядка st = 0.5

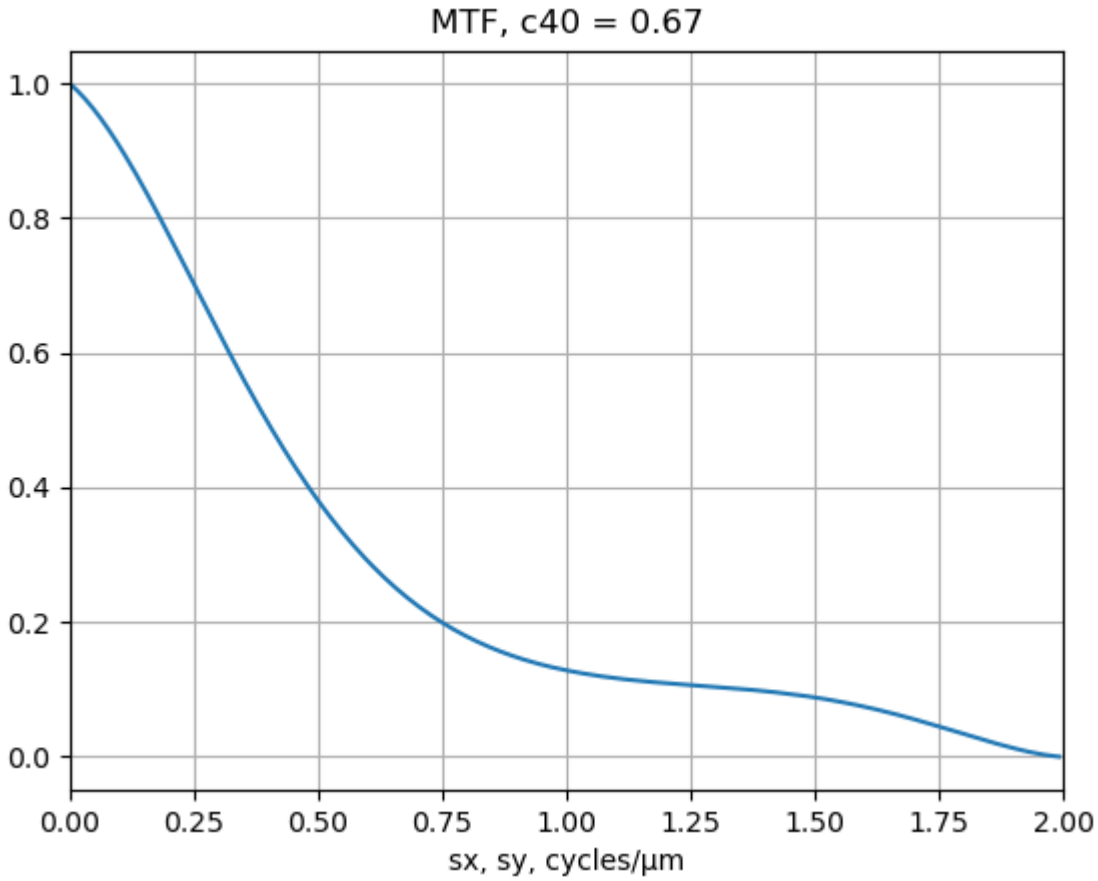

Рисунок 6 – ФПМ при сферической аберрации 3-го порядка st = 0.5

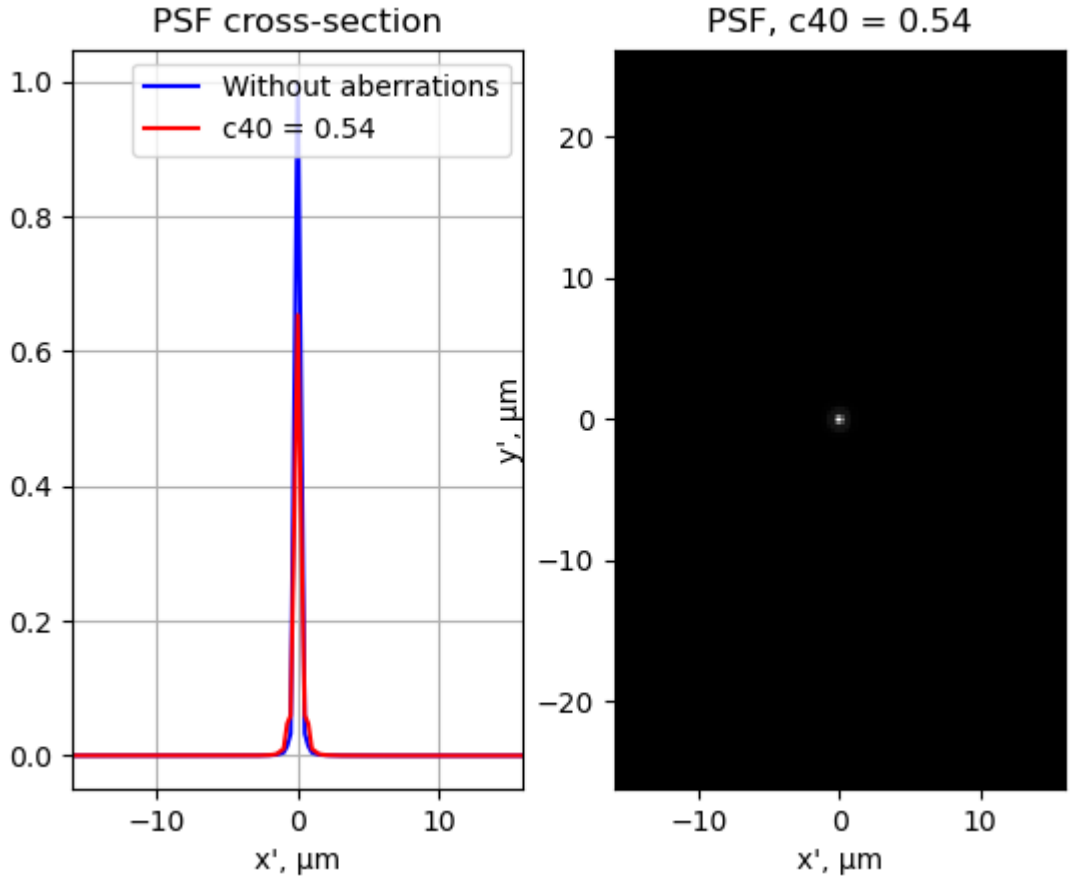

Рисунок 7 – Срез ФРТ при сферической аберрации 3-го порядка f(s=1) = 0.2

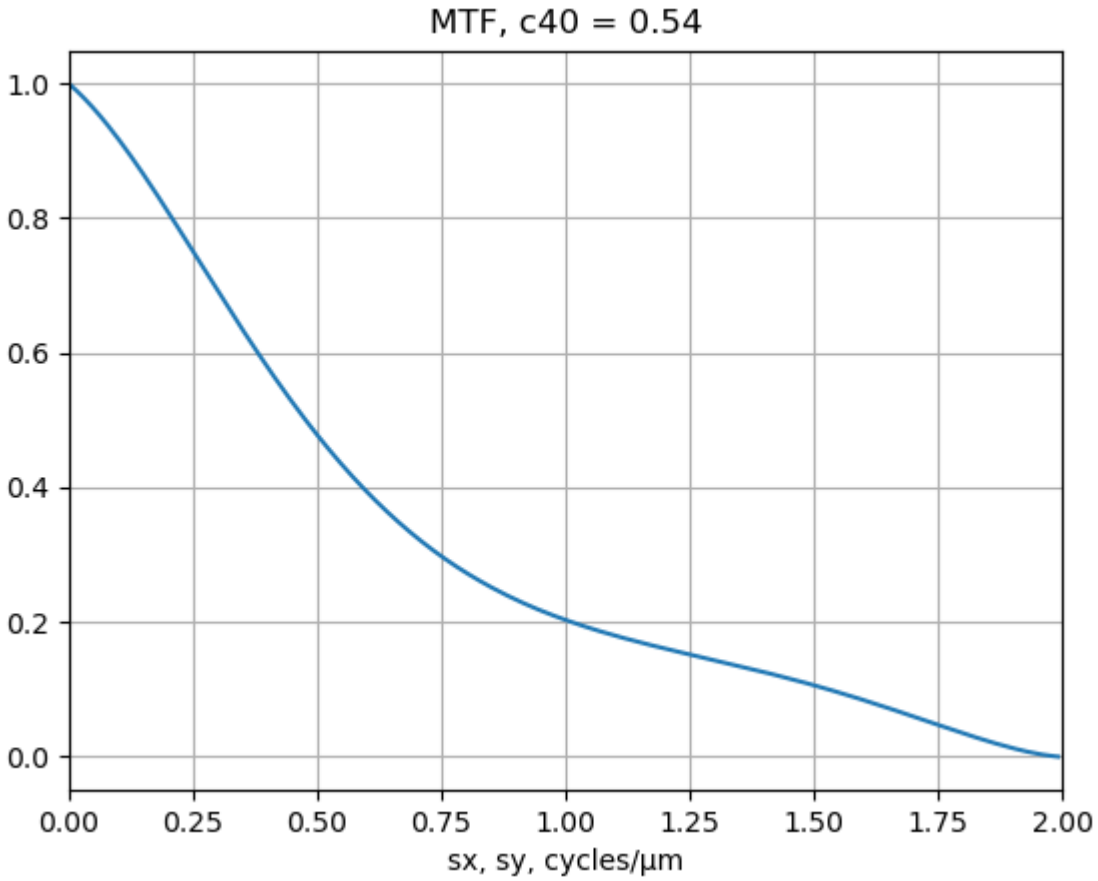

Рисунок 8 – ФПМ при сферической аберрации 3-го порядка ФПМ(s = 1) = 0.2

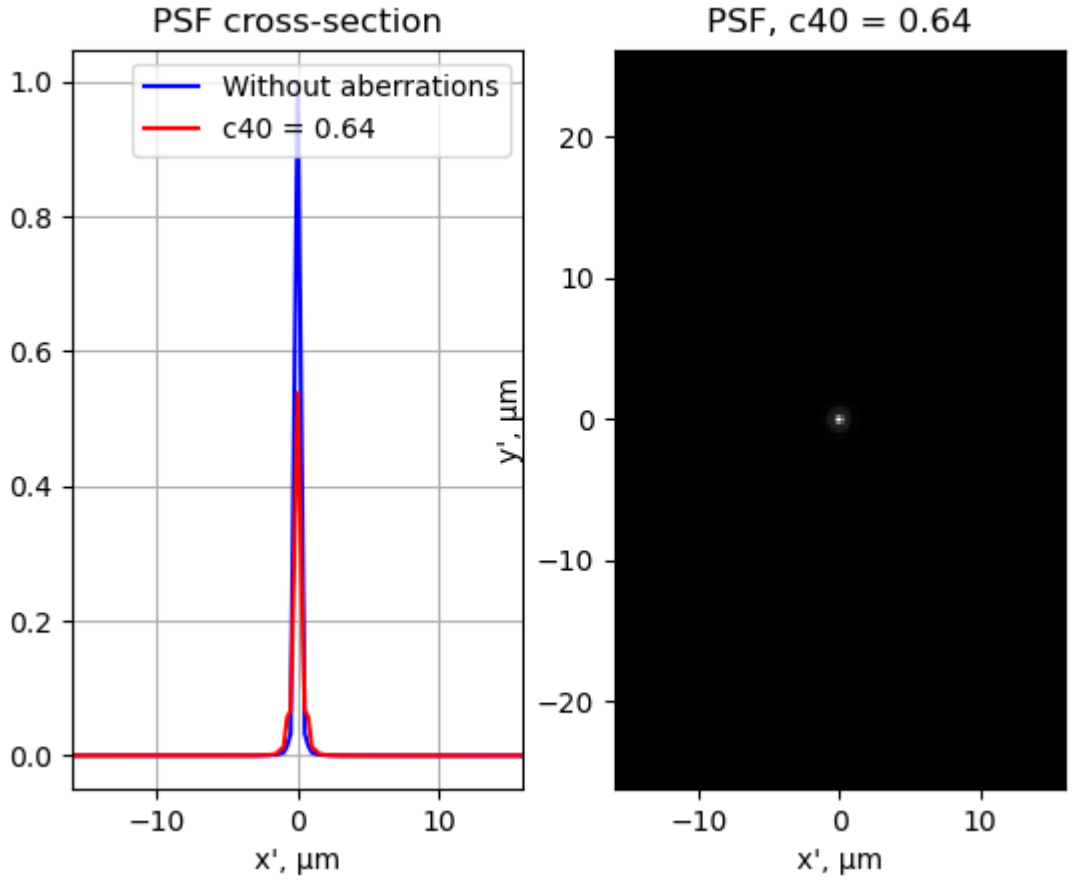

Рисунок 9 – Срез ФРТ при сферической аберрации 3-го порядка f(s=0.8) = ФПМ(s=0.8) = 0.2

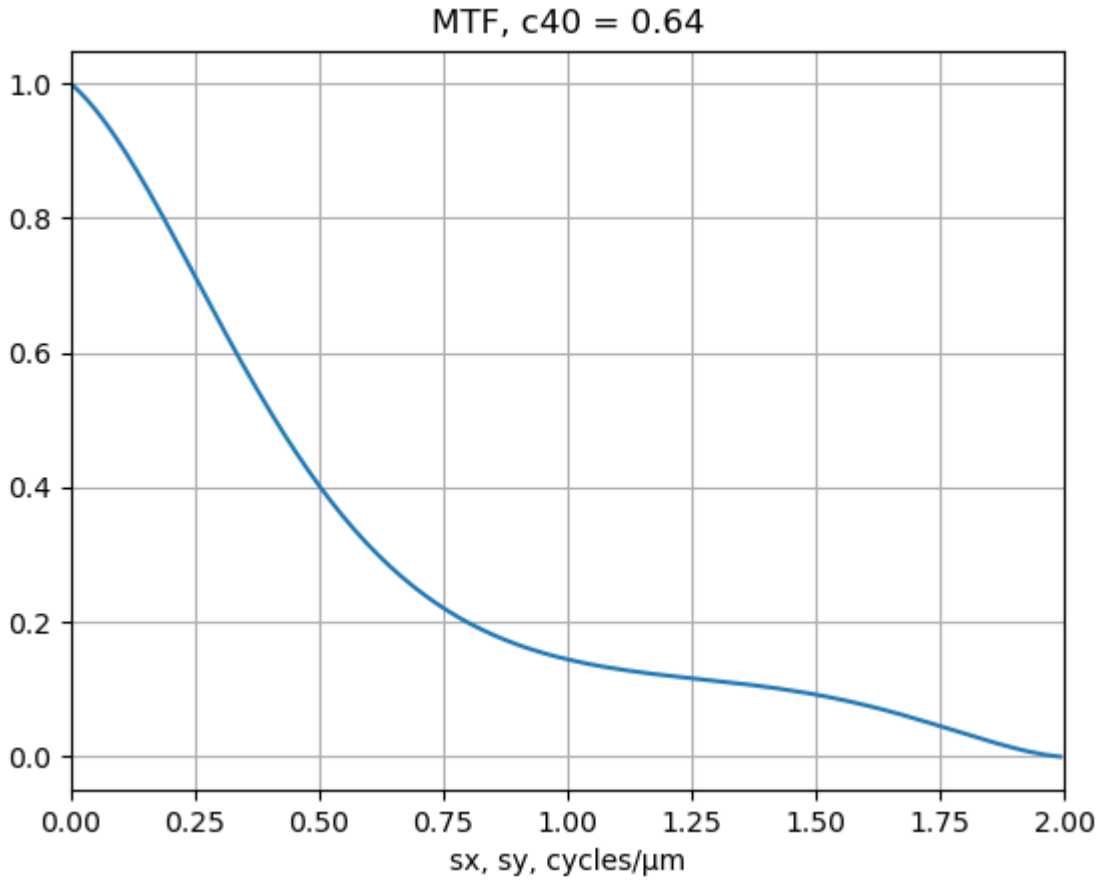

Рисунок 10 – ФПМ при сферической аберрации 3-го порядка f(s=0.8) = ФПМ(s=0.8) = 0.2

Для сложного изображения:

Рисунок 11 – Исходное изображение

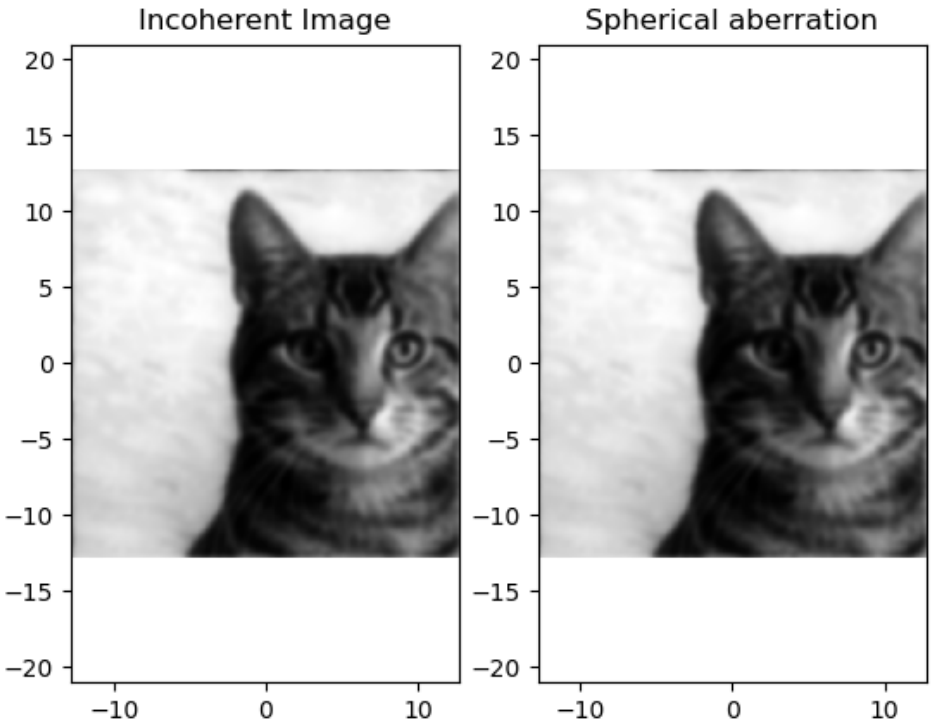

Рисунок 12 – Изображение со сферической аберрацией

Вывод:

В ходе работы разработаны программы для моделирования влияния сферической аберрации на качество изображения при различных значениях коэффициентов в полиноме Цернике.

Текст программ:

```
import numpy as np
import matplotlib.pyplot as plt


def make_abb(f, px, py, C40):
    rho = np.sqrt(px**2 + py**2)
    rho_normalized = rho / np.max(rho)
    R40 = 6 * rho_normalized**4 - 6 * rho_normalized**2 + 1
    f_abb = f * np.exp(2 * np.pi * 1j * C40 * R40)
    return f_abb

# Function to create a circular aperture
def func_Circ(x, y, N, r):
    f_circ = np.zeros((N, N))
    for i in range(N):
        for j in range(N):
            if x[j, i]**2 + y[j, i]**2 <= r**2:
                f_circ[j, i] = 1
    return f_circ


N = 512
lmbda = 0.5
Dzr = 4
A = 0.5
R_zr = N / Dzr
dp = Dzr / N
dn = 1 / (N * dp)
dx = dn * lmbda / A

n_max = dn * N / 2
p_max = dp * N / 2
x_max = dx * N / 2

nx, ny = np.meshgrid(np.arange(-n_max, n_max, dn), np.arange(-
n_max, n_max, dn))
px, py = np.meshgrid(np.arange(-p_max, p_max, dp), np.arange(-
p_max, p_max, dp))
x, y = np.meshgrid(np.arange(-x_max, x_max, dx), np.arange(-x_max,
x_max, dx))

for C40 in [0, 0.39, 0.67, 0.535, 0.64]:
    zr = func_Circ(px, py, N, 1)
    zr_abb = make_abb(zr, px, py, C40)

    psf_ = (dp / dn) *
np.fft.fftshift(np.fft.ifft2(np.fft.fftshift(zr))) * N
    psf_abs = (np.abs(psf_) * np.abs(psf_)) / (np.pi**2)
    D = (dn / dp) *
np.fft.fftshift(np.fft.fft2(np.fft.fftshift(psf_abs))) / N
    D_norm = D * np.pi
    M_abs = np.abs(D_norm)

    psf_abb = (dp / dn) *
np.fft.fftshift(np.fft.ifft2(np.fft.fftshift(zr_abb))) * N
```

```
    psf_abb_abs = (np.abs(psf_abb) * np.abs(psf_abb)) / (np.pi**2)
    D_abb = (dn / dp) *
np.fft.fftshift(np.fft.fft2(np.fft.fftshift(psf_abb_abs))) / N
    D_abb_norm = D_abb * np.pi
    M_abb_abs = np.abs(D_abb_norm)
    st = np.abs(np.max(psf_abb) / np.max(psf_))

    psf = (np.abs(psf_) * np.abs(psf_)) / (np.pi**2)
    psf_abb = (np.abs(psf_abb) * np.abs(psf_abb)) / (np.pi**2)

    # Plot PSF
    plt.figure(1)
    plt.subplot(1, 2, 1)
    plt.cla()
    plt.plot(x[N // 2, :], psf[N // 2, :], 'b')
    plt.plot(x[N // 2, :], psf_abb[N // 2, :], 'r')
    plt.xlim([-x_max / 4, x_max / 4])
    plt.grid(True)
    plt.xlabel("x', μm")
    plt.title('PSF cross-section')
    text = f'c40 = {C40:.2f}'
    plt.legend(['Without aberrations', text])

    plt.subplot(1, 2, 2)
    plt.pcolormesh(x, y, psf_abb, cmap='gray')
    plt.axis('equal')
    plt.axis([-x_max / 4, x_max / 4, -x_max / 4, x_max / 4])
    plt.xlabel("x', μm")
    plt.ylabel("y', μm")
    text = f'PSF, c40 = {C40:.2f}'
    plt.title(text)

    # Plot MTF
    plt.figure(2)
    plt.subplot(1, 1, 1)
    plt.cla()
    plt.plot(px[N // 2, :], M_abb_abs[N // 2, :])
    plt.xlim([0, 2])
    plt.grid(True)
    plt.xlabel('sx, sy, cycles/μm')
    text = f'MTF, c40 = {C40:.2f}'
    plt.title(text)
    plt.pause(0.1)

plt.show()
```

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


def make_abb(f, px, py, C40):
    rho = np.sqrt(px**2 + py**2)
    rho_normalized = rho / np.max(rho)
    R40 = 6 * rho_normalized**4 - 6 * rho_normalized**2 + 1
    f_abb = f * np.exp(2 * np.pi * 1j * C40 * R40)
```

```
    return f_abb

# 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) * step_zr, step_zr),
    np.arange(-(N/2) * step_zr, (N/2) * step_zr, step_zr)
)
a = 1
for i in range(N-1):
    for j in range(N-1):
        if (x[i, j]**2)/(a**2) + y[i, j]**2 < 1:
            pupil[i, j] = 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.fftshift
(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, :]

abb = make_abb(intensity_rasp_img, im_axis_X, im_axis_Y, 0.5)
```

```
# Plot results
#plt.figure(figsize=(12, 6))
plt.subplot(1,2,1)
plt.pcolormesh(im_axis_X, im_axis_Y, intensity_rasp_img,
cmap='gray')
plt.axis('equal')
plt.title("Incoherent Image")

plt.subplot(1, 2, 2)
plt.pcolormesh(im_axis_X, im_axis_Y, np.abs(abb), cmap='gray')
plt.axis('equal')
plt.title('Spherical aberration')
plt.show()
```