Project 3: Diffraction

```
import matplotlib    # Library used for plotting
import numpy as np   # Numerical library
import matplotlib.pyplot as plt # Plot commands
from matplotlib.colors import LinearSegmentedColormap

from scipy.fftpack import fft2, fftshift

import scipy         # Another numerical library
from scipy import integrate

CF_red = (204/255, 121/255, 167/255)
CF_vermillion = (213/255, 94/255, 0)
CF_orange = (230/255, 159/255, 0)
CF_yellow = (240/255, 228/255, 66/255)
CF_green = (0, 158/255, 115/255)
CF_sky = (86/255, 180/255, 233/255)
CF_blue = (0, 114/255, 178/255)
CF_black = (0, 0, 0)
```

Introduction

Recall from the class that when a wave goes through an aperture, it undergoes diffraction. When this diffracted wave hits a screen that is perpendicular to the aperture, the diffraction pattern can be observed. If the screen is far away from the aperture, the complex amplitude of the field on the screen is

$$E_{diff}(x,y) \approx \frac{1}{\lambda D} \int da' e^{-i\frac{k}{D}(xx'+yy')} E_{inc}(x',y'),$$

where $E_{inc}(x',y')$ is the complex field amplitude at the plane of the aperture and D is the distance between the aperture and the screen where $E_{diff}(x,y)$ is observed.

Question 1

For this project, we will be analyzing different appertures. A general aperture function will be given by `Aperture(x, y)`. So, it is useful to start by writing a function that takes `Aperture(x, y)`, k, D, the size of the screen hosting the aperture, and number of points on the 2D grid and returns the field on the screen position beyond the aperture. Start by writing `E_Fields(Aperture, k, D, screen_size, nPts)` which returns the aperture grid, the aperture signal, the target grid, and the target signal.

```
def E_Fields(Aperture, k, D, screen_size, nPts):
    vAperture = np.vectorize(Aperture)

    half_screen = screen_size / 2
```

```
    grid = np.mgrid[-half_screen:half_screen:1j * nPts, -
half_screen:half_screen:1j * nPts]

    E_Inc = vAperture(grid[0], grid[1])

    step_size = 2 * np.pi * D / (screen_size * k)
    target_size = step_size * nPts
    half_target = target_size / 2
    target_grid = np.mgrid[-half_target:half_target:1j * nPts, -
half_target:half_target:1j * nPts]

    lamda = 2 * np.pi / k
    E_Diff = np.abs(fftshift((1 / (lamda * D)) * fft2(E_Inc)))

    return grid, E_Inc, target_grid, E_Diff
```

I define the `E_Fields` function that takes in the aperture function, the wavenumber k, the distance D, the size of the screen, and the number of points on one axis the 2D grid. The function returns the aperture grid, the aperture signal, the target grid, and the target signal.

Question 2

As a start, let's make sure things work as they should. To this end, consider a green light with wavelength $\lambda = 550$nm. Let the aperture be an evenly illuminated circle of radius 1mm. Create a pair of 2D plots showing the aperture and the diffracted pattern on the target screen. It is up to you how far you position the screen, but remember that it has to be positioned at a distance that is substantially larger than $\pi R^2/\lambda$, where R is the aperture radius. **Think about reasonable values here!**. Recall that $k = 2\pi/\lambda$. **Be careful with the units!!!**

Start be writing `Aperture(x, y, R)` and then use the function you wrote above to obtain the results. Play with the scale of the final plot to make sure the results are clear.

When you plot the **absolute value** of the diffracted result, you should observe a set of concentric circles, known as the Airy pattern. Comment on your results.

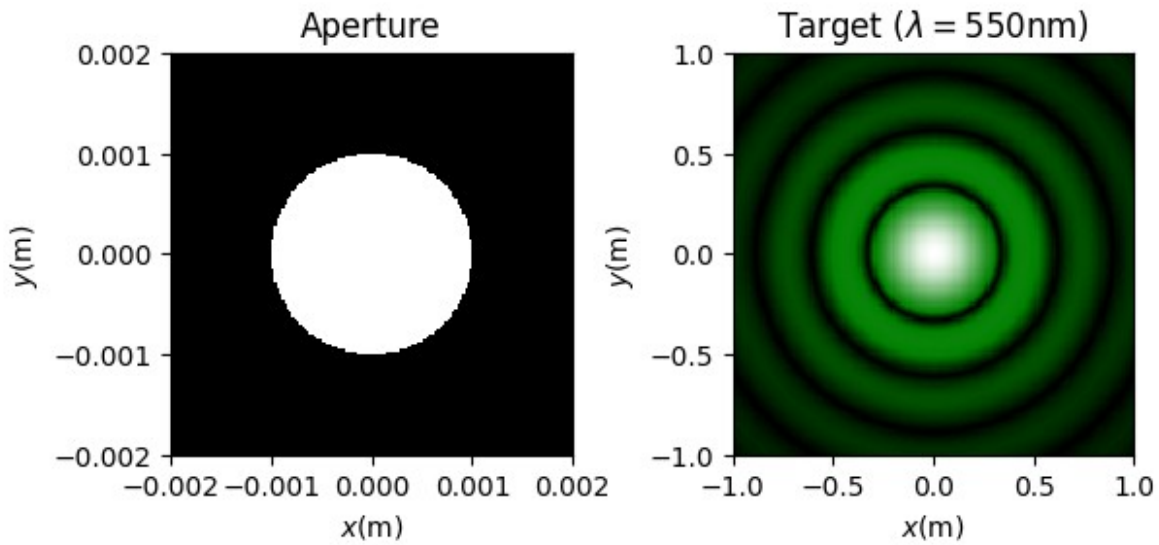
SUGGESTION: Divide your diffracted pattern by `nPts**2` to prevent the numbers from getting huge.

```
def Aperture(x, y, R):
    return 1 if np.sqrt(x**2 + y**2) < R else 0

R = 1e-3
lamda = 550e-9
k = 2 * np.pi / lamda
D = 1e3

screen_size = 5e-2
nPts = 2000

Aperture_with_R = lambda x, y: Aperture(x, y, R)
```

```
aperture_grid, E_Inc, target_grid, E_Diff = E_Fields(Aperture_with_R,
k, D, screen_size, nPts)
```

I define the `Aperture` function that creates a signal within a circle of the given radius. Then, I set the parameters of the problem. R and λ are based on the given values. k is calculate from the wavelength, and D is set to 1000 which is much larger than $\pi R^2/\lambda$. For the screen size, I use 0.05 and simulate with 2000 points on each axis. Finally, I calculate the fields and signals using these parameters.

```
fig, (ax1, ax2) = plt.subplots(1, 2)

green_colors = [(0, "black"), (0.1, "green"), (1, "white")]
green_cmap = LinearSegmentedColormap.from_list("green", green_colors)

ax1_lim = 2 * R
ax1.pcolormesh(aperture_grid[0], aperture_grid[1], E_Inc, cmap =
green_cmap)
ax1.set(title = "Aperture", xlabel = "$x$(m)", ylabel = "$y$(m)",
        xlim = [-ax1_lim, ax1_lim], ylim = [-ax1_lim, ax1_lim])
ax1.set_aspect("equal")

ax2_lim = 1
pc = ax2.pcolormesh(target_grid[0], target_grid[1], E_Diff / nPts**2,
cmap = green_cmap)
ax2.set(title = "Target ($\lambda = $550nm)", xlabel = "$x$(m)",
ylabel = "$y$(m)",
        xlim = [-ax2_lim, ax2_lim], ylim = [-ax2_lim, ax2_lim])
ax2.set_aspect("equal")

plt.subplots_adjust(wspace = 0.4)
```

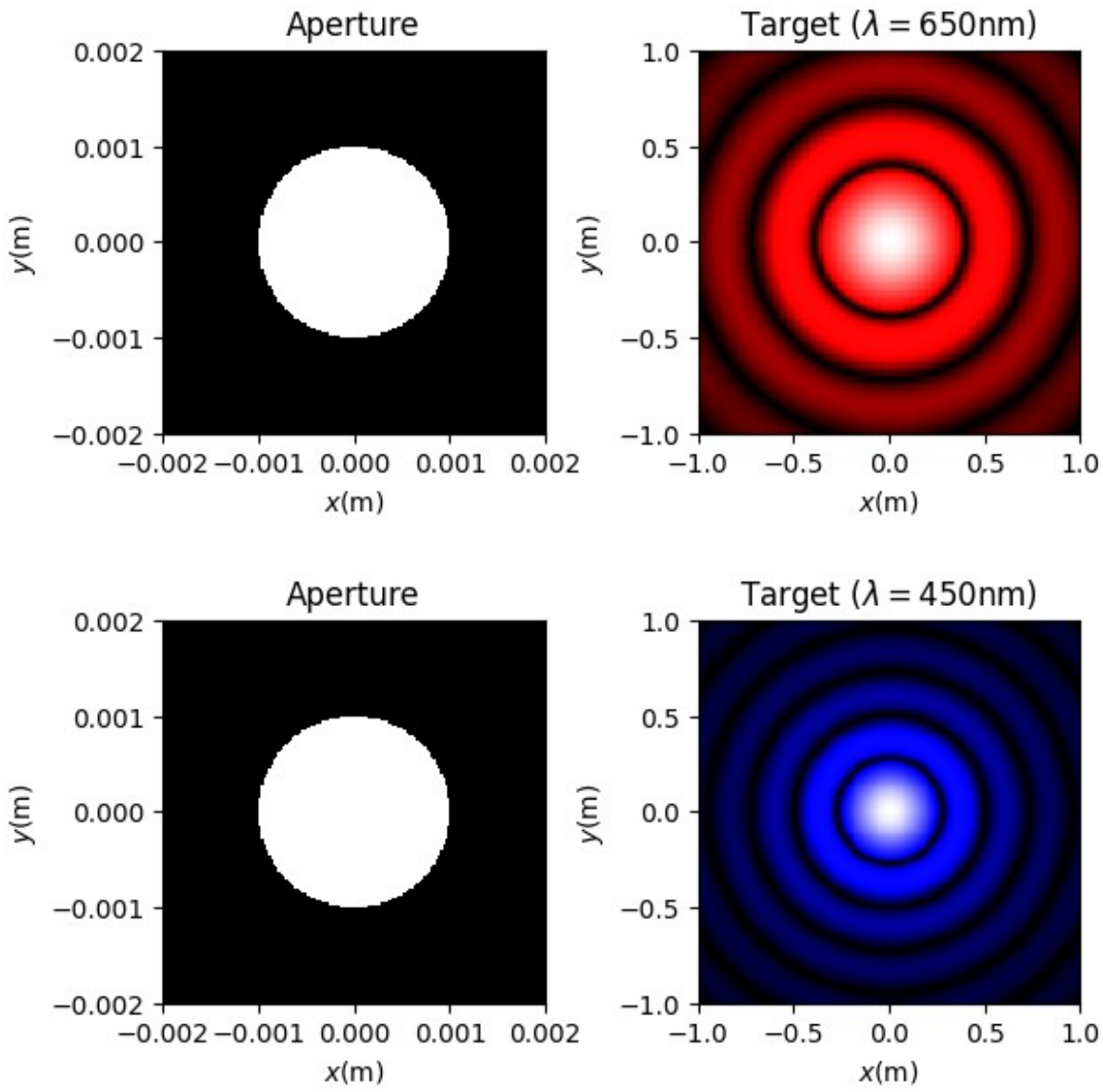


Plotting the diffraction of the green light with the given parameters, I observe the Airy pattern. The concentric circles are clearly visible. The original plot is too zoomed out, so I set limits to the plot to make the pattern more visible.

Question 3

Repeat the plots above for $\lambda = 650$ nm and $\lambda = 450$ nm. Keep D the same as you used above. What colors do these wavelengths correspond to? How does your diffracted pattern change?

```
lambdas = np.array([650e-9, 450e-9])
ks = 2 * np.pi / lambdas

lambda_headings = ["650nm", "450nm"]
red_colors, blue_colors = [(0, "black"), (0.1, "red"), (1, "white")],
[(0, "black"), (0.1, "blue"), (1, "white")]
red_blue_cmaps = LinearSegmentedColormap.from_list("red", red_colors),
LinearSegmentedColormap.from_list("blue", blue_colors)

for i, k in enumerate(ks):
    aperture_grid, E_Inc, target_grid, E_Diff =
E_Fields(Aperture_with_R, k, D, screen_size, nPts)

    fig, (ax1, ax2) = plt.subplots(1, 2)

    ax1_lim = 2 * R
    ax1.pcolormesh(aperture_grid[0], aperture_grid[1], E_Inc, cmap =
red_blue_cmaps[i])
    ax1.set(title = "Aperture", xlabel = "$x$(m)", ylabel = "$y$(m)",
            xlim = [-ax1_lim, ax1_lim], ylim = [-ax1_lim, ax1_lim])
    ax1.set_aspect("equal")

    ax2_lim = 1
    pc = ax2.pcolormesh(target_grid[0], target_grid[1], (E_Diff /
nPts**2), cmap = red_blue_cmaps[i])
    ax2.set(title = f"Target ($\lambda = ${lambda_headings[i]})",
xlabel = "$x$(m)", ylabel = "$y$(m)",
            xlim = [-ax2_lim, ax2_lim], ylim = [-ax2_lim, ax2_lim])
    ax2.set_aspect("equal")

    plt.subplots_adjust(wspace = 0.4)
```



The longer wavelength of light corresponds to the color red and the shorter wavelength corresponds to the color blue. Although the pattern is the same, the diffraction pattern is more spread out for the red light and more compressed for the blue light. When the wavelength is increased, there is more diffraction and radius of the central spot becomes wider. This suggests that diffraction is dependent on the wavelength of the light.

Question 4

Let us now calculate how the radius of the central bright spot of the diffracted pattern depends on λ. To establish this

- Choose a number of λ's in the range from 400 to 700nm.
- Keeping the same setup parameters as above, calculate the corresponding diffraction patterns.

- Take a slice across $y=0$ for each λ. Find the position of the first minimum in the signal for $x>0$ (call it x_i) and record it. Plot x_i vs λ. What does the plot look like?

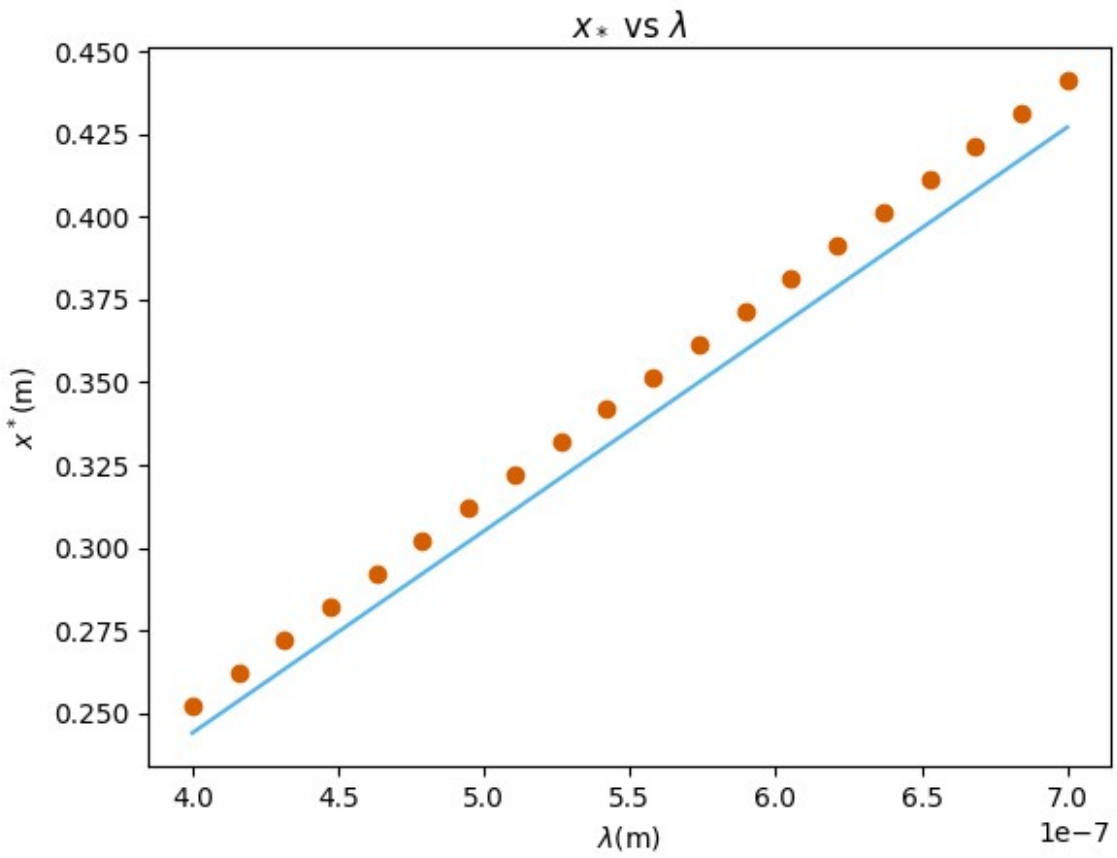
As it turns out, there is a formula that gives the radius of the spot for this kind of diffraction: $\rho = 1.22\lambda D/(2R)$. Do you results agree with this? Modify R and D to confirm. **Be careful what values you use! Don't just put random ones: make sure that the necessary relations hold!**

```
def min_index_from_middle(ls):
    middle = len(ls) // 2
    for i in range (middle, len(ls) - 1):
        if ls[i + 1] > ls[i]:
            return i

lambdas = np.linspace(400e-9, 700e-9, 20)
ks = 2 * np.pi / lambdas

mid = nPts // 2
x_stars = []
for k in ks:
    _, _, target_grid, E_Diff = E_Fields(Aperture_with_R, k, D,
screen_size, nPts)
    min_index = min_index_from_middle(E_Diff[mid])
    x_stars.append(target_grid[1][mid][min_index])
```

Here, I define the functions to get the radius of the airy spot for different wavelengths. I set the parameters of the problem as before. However, I use 20 different wavelength ranging from 400 to 700 nm and calculate the corresponding ks. For each k, I calculate the fields and signals and take a slice across $y=0$. I find the first position of the minimum value from the middle of the signal and record it.

```
formula = lambda D, R: 1.22 * lambdas * D / (2 * R)

fig, ax = plt.subplots()
ax.scatter(lambdas, x_stars, color = CF_vermillion)
ax.plot(lambdas, formula(D, R), color = CF_sky)
ax.set(xlabel = "$\lambda$(m)", ylabel = "$x^*$(m)", title = "$x_*$ vs
$\lambda$")
```

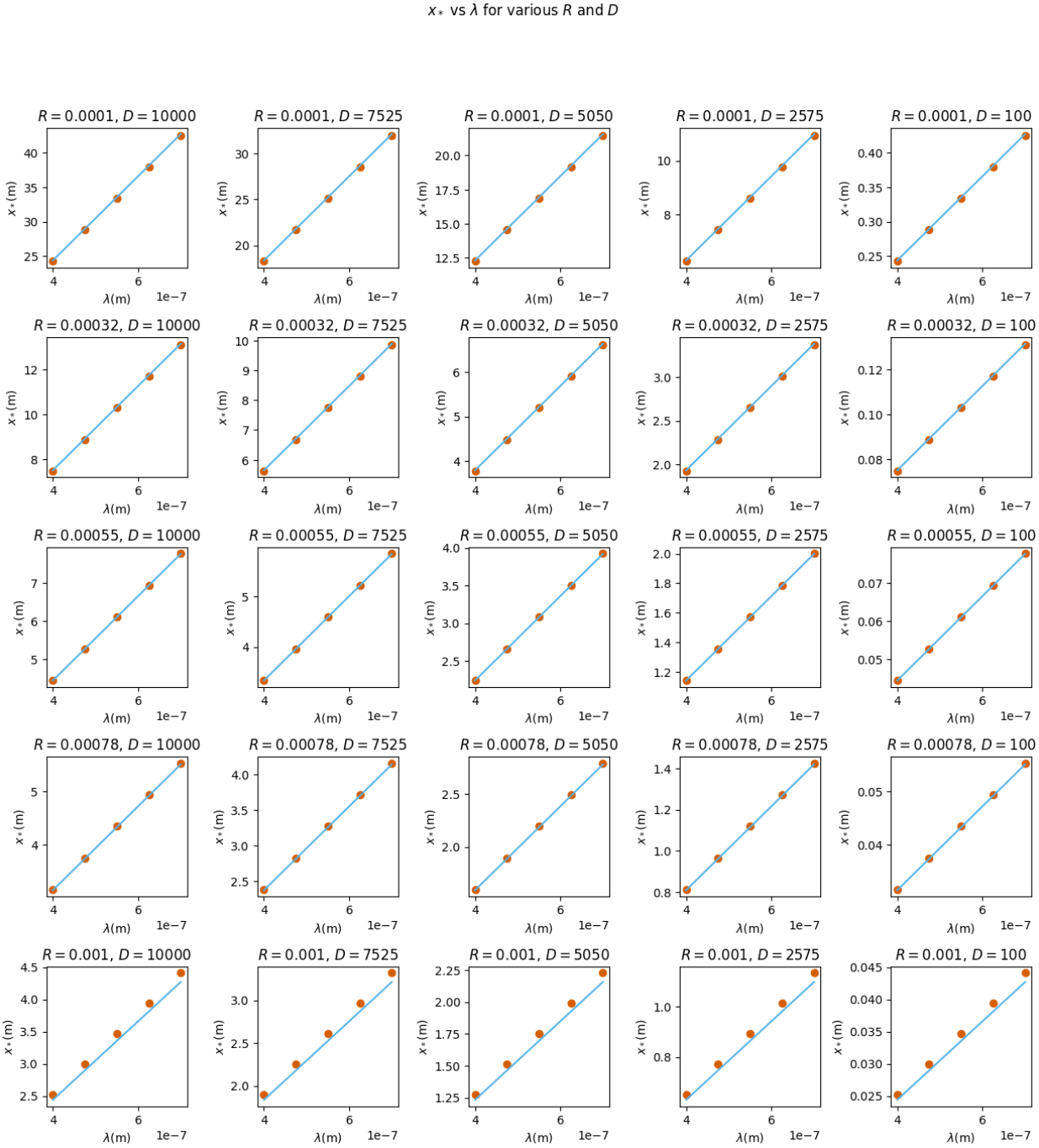
```
[Text(0.5, 0, '$\\lambda$(m)'),
 Text(0, 0.5, '$x^*$(m)'),
 Text(0.5, 1.0, '$x_*$ vs $\\lambda$')]
```



I plot the $x_¿$ values against the wavelengths as a scatter plot. Additionally, I plot the theoretical values of the radius of the central spot using the given formula. The plot shows that the calculated values are close to the theoretical values.

```
Rs = np.linspace(1e-4, 1e-3, 5)
Ds = np.linspace(1e4, 1e2, 5)

fig, axs = plt.subplots(5, 5, figsize = (15, 15))

for i, R in enumerate(Rs):
    for j, D in enumerate(Ds):
        ax = axs[i,j]
        lambdas = np.linspace(400e-9, 700e-9, 5)
        ks = 2 * np.pi / lambdas

        mid = nPts // 2
        mins = []
        for k in ks:
            _, _, target_grid, E_Diff = E_Fields(Aperture_with_R, k,
D, screen_size, nPts)
            min_index = min_index_from_middle(E_Diff[mid])
            mins.append(target_grid[1][mid][min_index])
```

```
        ax.scatter(lambdas, mins, color = CF_vermillion)
        ax.plot(lambdas, formula(D, R), color = CF_sky)
        ax.set(xlabel = "$\lambda$(m)", ylabel = "$x_*$(m)", title =
f"$R = ${round(R, 5)}, $D = ${round(D)}")

fig.suptitle("$x_*$ vs $\lambda$ for various $R$ and $D$")
plt.subplots_adjust(wspace = 0.5, hspace = 0.5)
```



I repeat the simulations with different values of R and D to confirm the formula. I take 5 points for R and D ranging from 0.0001 to 0.001 and 10000 to 100, respectively. I use the same range of wavelengths as before, but only use 5 wavelength to reduce computation time. The plots show that the simulation results agree with the theoretical results for different values of R and D.