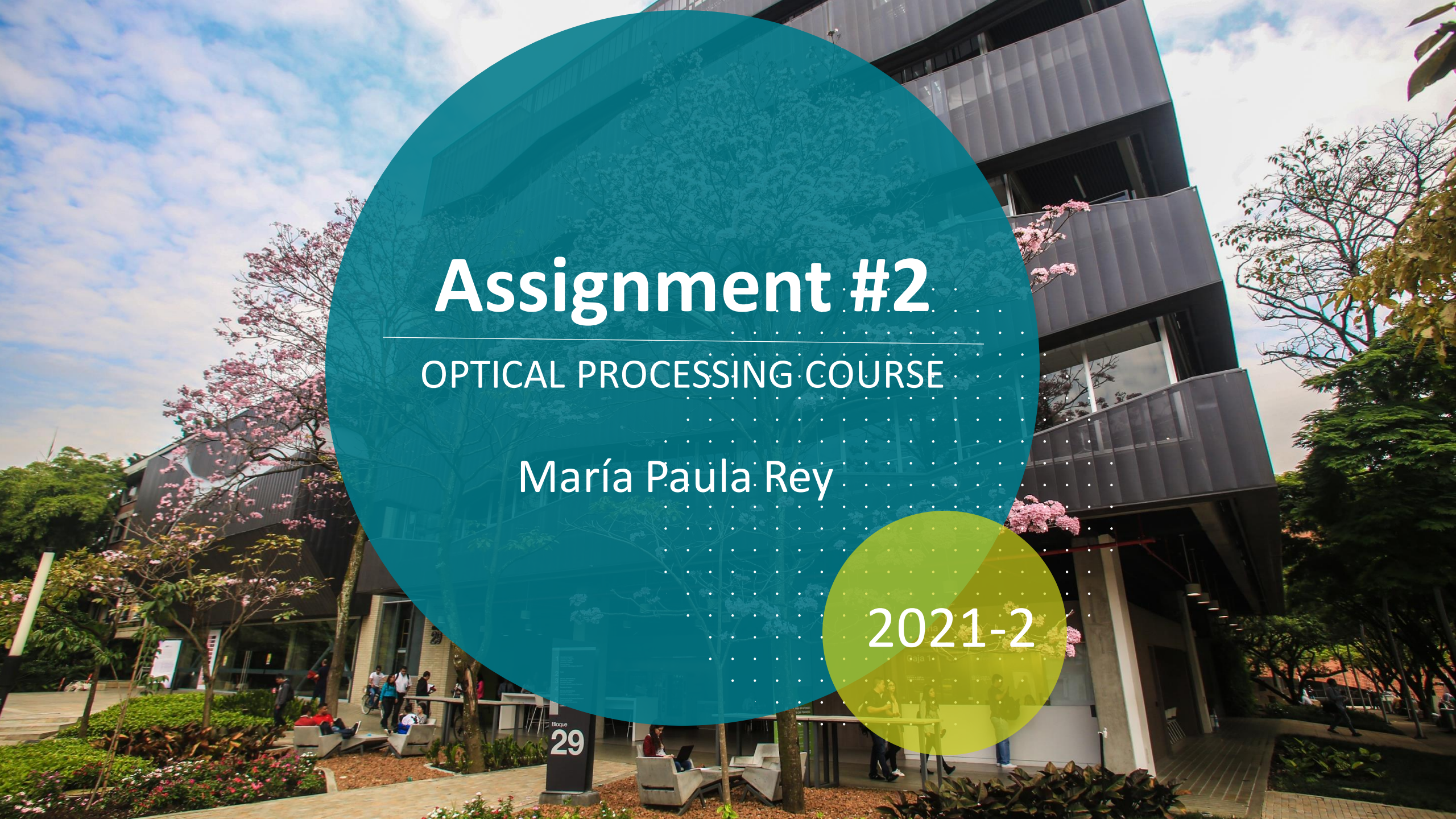
Inspira Crea Transforma

UNIVERSIDAD EAFIT



Assignment #2

OPTICAL PROCESSING COURSE

María Paula Rey

2021-2

UNIVERSIDAD EAFIT

Contents

1. Introduction
2. Code
3. Results
4. References

Introduction

Introduction

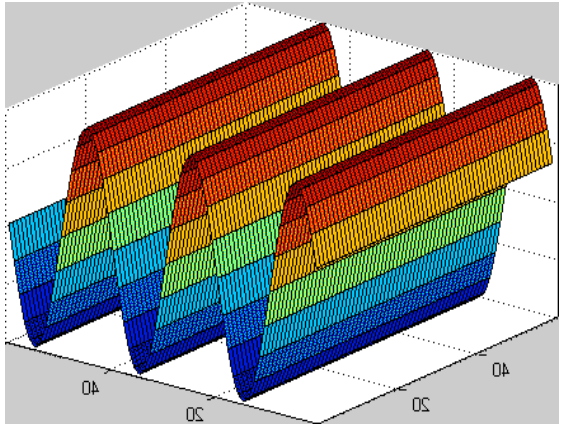
Numerical wave propagation of scalar optical fields

Numerical wave propagation of scalar optical fields consists in the discretization and calculation of the diffraction integral in any of its formalisms. This calculation allows the computing of the complex optical field on a given plane, from its known value over another plane [1].

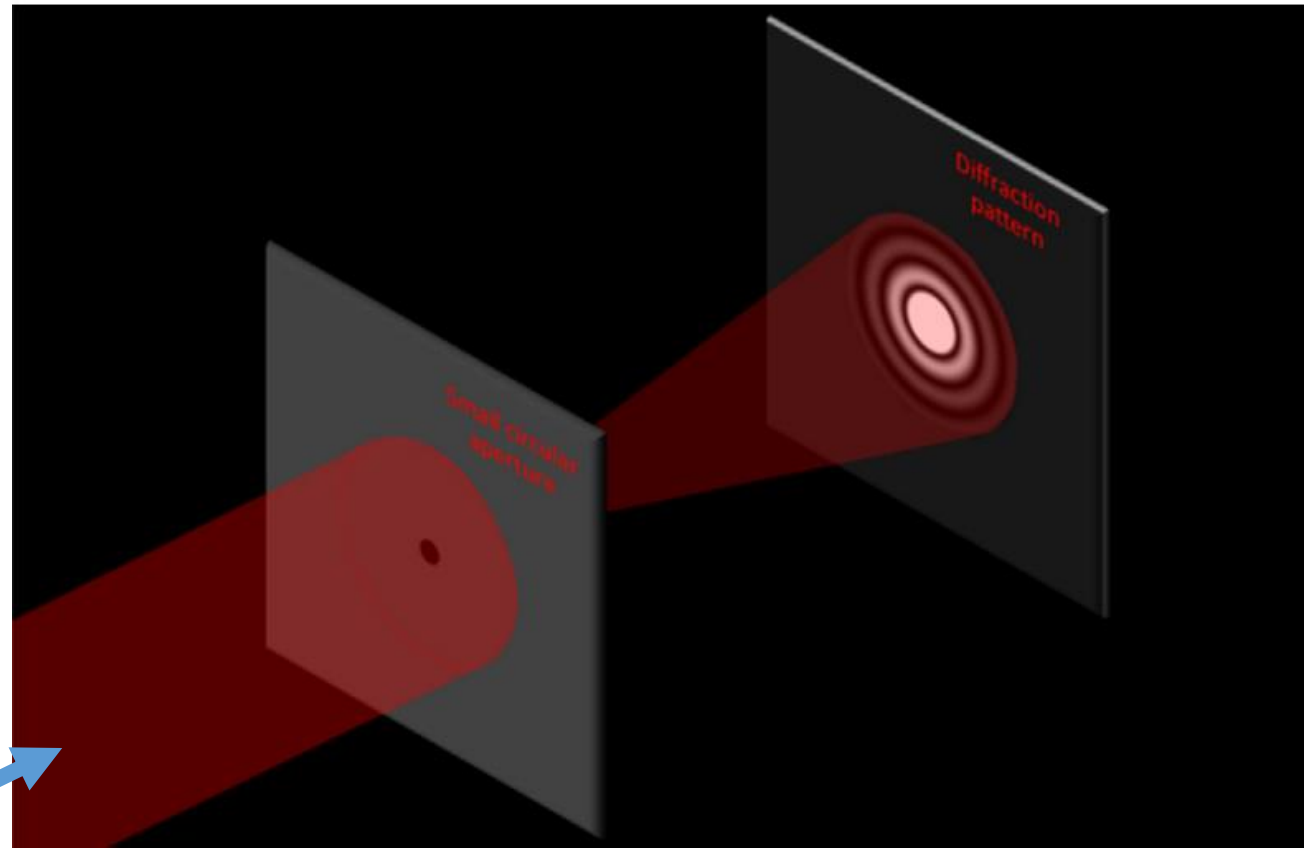
Introduction

Fresnel zones

A monochromatic optical field is propagating between an **input plane**, an opaque screen with a small circular aperture (pinhole), and an **output plane** over which the diffraction pattern of the wave through the pinhole is obtained [1].



Plane wave. Image obtained from t.ly/Y0TH.



Diffraction pattern from a small circular aperture *[1].*

Introduction

Fresnel Transform

To compute a numerical propagation via the scalar diffraction theory, we can use the schematic diagram presented in Fig. 2. [1].

The position of one particular pixel:

- Input plane → $x = \Delta x \cdot n$ and $y = \Delta y \cdot m$
- Output plane → $x' = \Delta x' \cdot n'$ and $y' = \Delta y' \cdot m'$

n, m, n' and m' are integer indexes.

$\Delta x, \Delta y, \Delta x'$ and $\Delta y'$ are the pixel size along each dimension for the input and output planes, respectively.

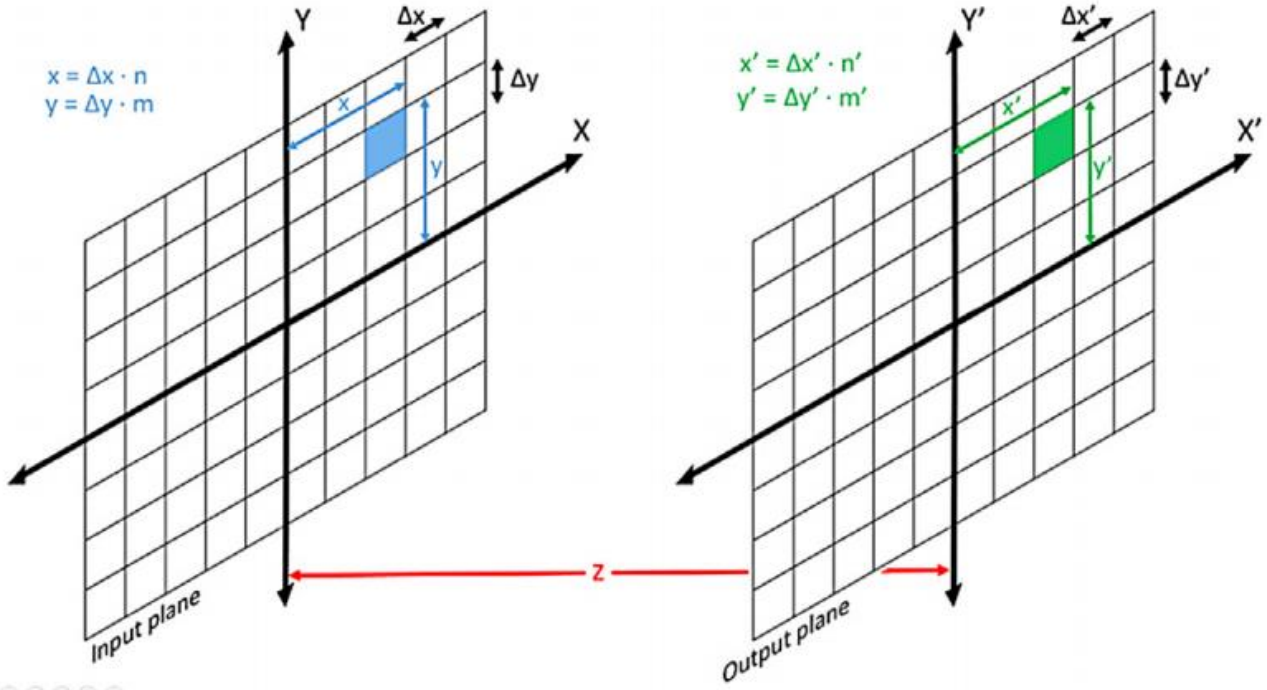


Fig.2. Schematic illustration of the diffraction calculation between two discretized parallel planes. Input plane (left) and output plane (right) [1].

Introduction

Fresnel Transform

$U(n\Delta x, m\Delta y, 0) = U(n, m, 0)$ → entry of the matrix that accounts for the input plane information
$U(n'\Delta x', m'\Delta y', z) = U(n'\Delta x', m'\Delta y', z)$ → entry of the matrix that accounts for the output plane information

The numerical propagation consists on obtaining $U(n', m', z)$ from $U(n, m, 0)$.

Introduction

Fresnel Transform

The Fresnel transform considers the computation of the propagated wavefield by Fourier transforming the input wavefield modified by the impulse response of the optical system. This approach is valid for **large distances** between the input and output planes [1].

$$U(n', m', z) = \frac{e^{ikz}}{2i\lambda} e^{\frac{ik}{2z}\left[(n'\Delta x')^2 + (m'\Delta y')^2\right]} \Im\left[U(n, m, 0) e^{\frac{ik}{2z}\left[(n\Delta x)^2 + (m\Delta y)^2\right]}\right] \quad (1)$$

FFT

$k = 2\pi/\lambda \rightarrow$ wave number

Eq. (1) allows the calculation of the complex wavefield in an output plane from the wavefield in input plane; both planes are parallel and placed z apart [1].

Introduction

Fresnel Transform

The distances z for which Eq. (1) is valid must satisfy

$$z \geq P_N \Delta x^2 [N - P_N]/\lambda$$

$P_N \rightarrow$ number of pixels needed to sample a π phase jump.

The distance between the planes must be considerably larger than their dimensions.

This formalism is widely used due to its low computational complexity, of the order of $O(N^2 lnN)$, which allows video rate executions in a wide range of applications.

Introduction

Angular Spectrum

The angular spectrum method accounts for the numerical propagation by considering the transfer function of the optical system, which is computed analytically (pure angular spectrum) or numerically (convolution method). The use of the angular spectrum approach limits its application to **small distances** between the input and output planes [1].

$$U(n', m', z) = \underbrace{\Im^{-1}}_{\text{FFTs}} \left[\Im \left[U(n, m, 0) \right] e^{iz\sqrt{k^2 - 4\pi^2(\Delta f_x^2 p^2 + \Delta f_y^2 q^2)}} \right] \quad (2)$$

$(\Delta f_x, \Delta f_y)$ → sampling intervals in the frequency domain

$k = 2\pi/\lambda$ → wave number

p and q integer variables

Eq. (2) is the typical numerical implementation of the angular spectrum formalism. This expression has a computational complexity in the order of $O(N^2 lnN)$.

Introduction

Fresnel Zones

We can verify that the propagation is executed properly if the number of Fresnel's zones for a particular propagation distance agrees with the spot in the center of the obtained diffraction pattern [1]

$$N_F = \frac{R^2}{\lambda d}$$

N_F	Fresnel Number
R	Circular aperture radius
λ	Wavelenght
d	Distance of the screen from the aperture

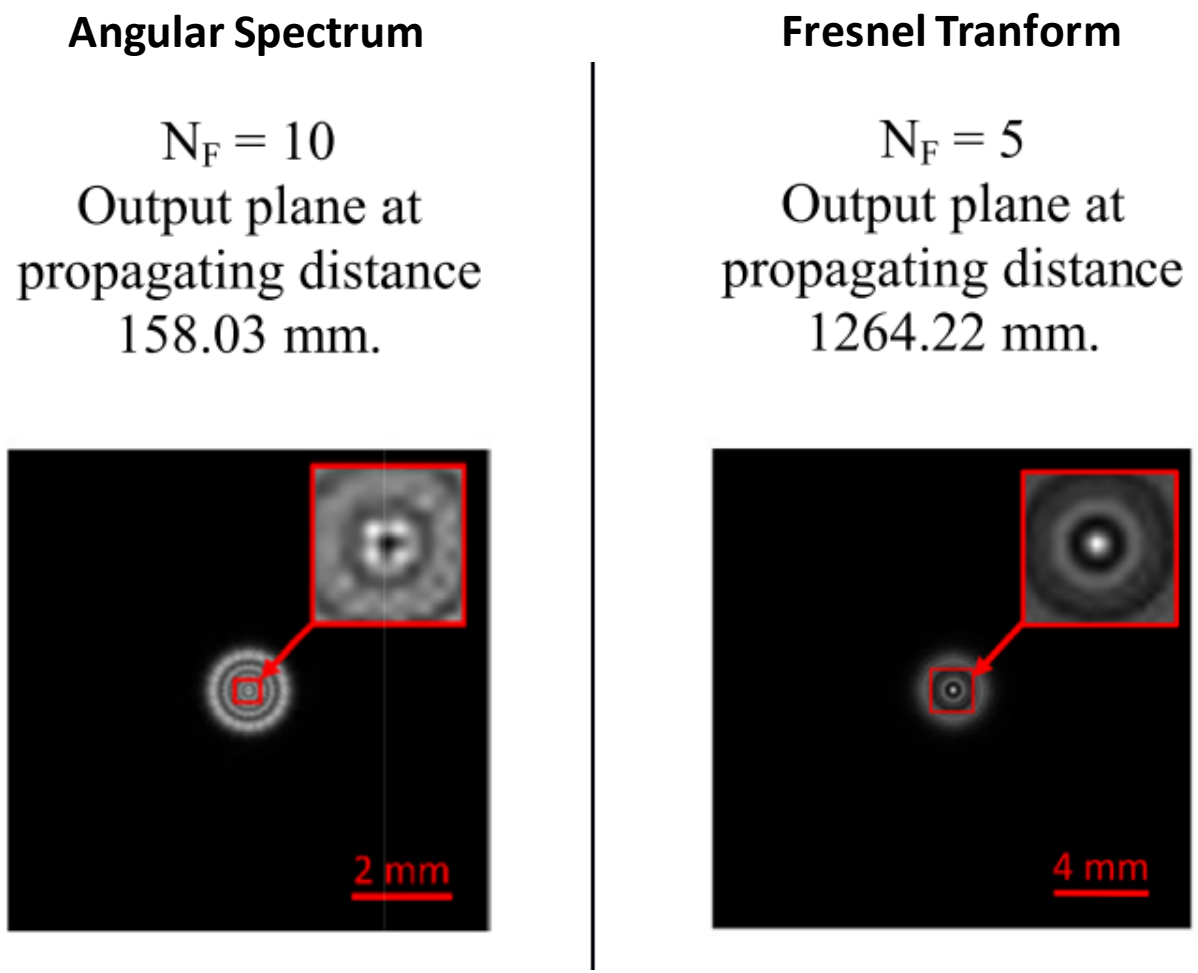
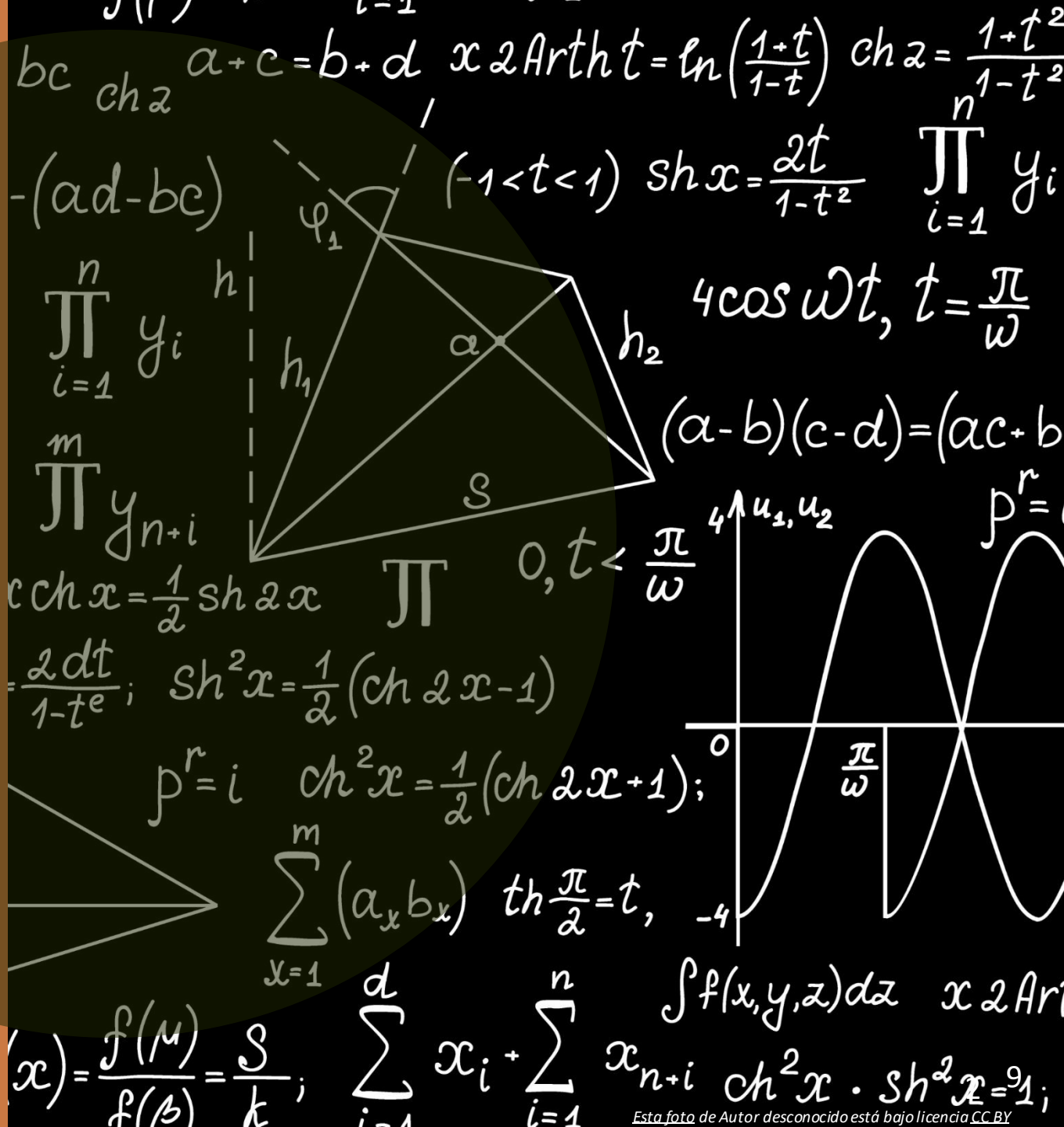


Fig. 3. Modeling of the numerical propagation of a plane wave that illuminates a circular aperture [1].

UNIVERSIDAD EAFIT

Code



Code

Propagation parameters

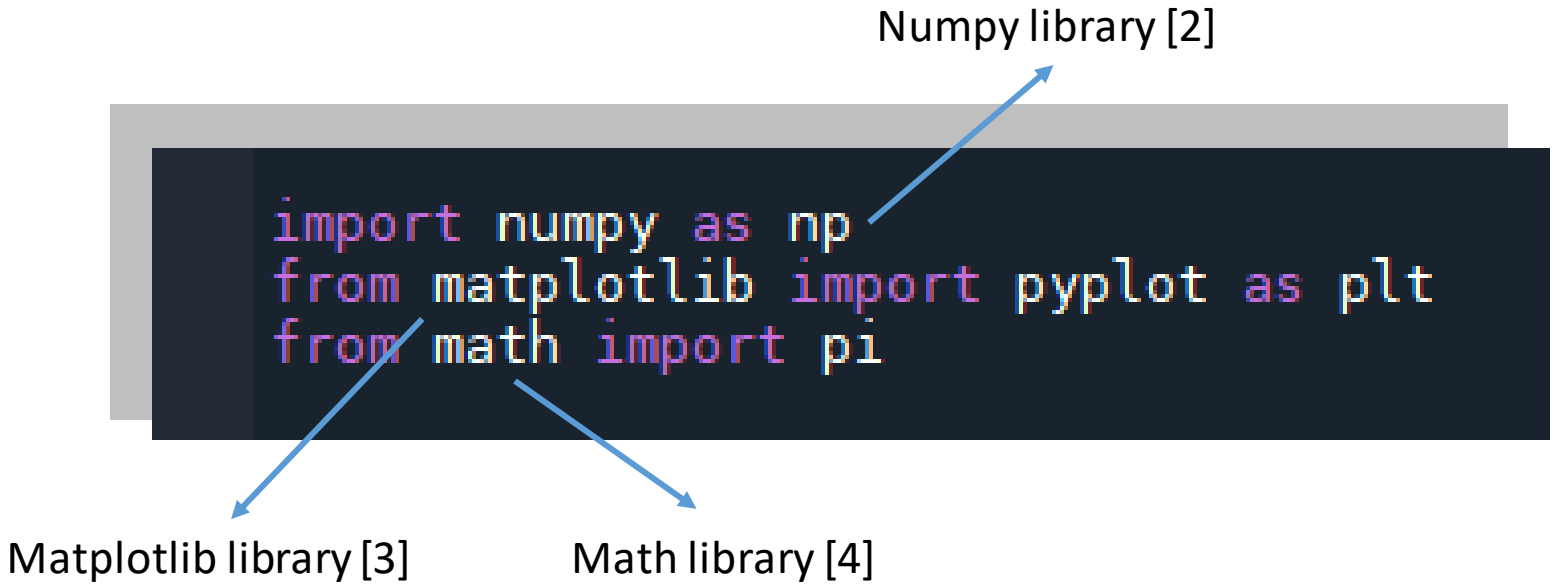
Parameter	Value
Input pitch ***dx*** [mm]	0.039
Input pitch ***dy*** [mm]	0.039
Wavelength [mm]	0.0006328
Number of horizontal data points (***M***)	256
Number of horizontal data points (***N***)	256
Width and height of input planes [mm]	10

	Propagation distance (z) [mm]	Expected Fresnel Number	Pinhole radius
Angular Spectrum	143.66	11	1mm
	158.03	10	
Fresnel Transform	1053.52	6	2mm
	1254.22	5	

Code

Necessary functions

Declaring libraries



Code

Necessary functions

Function to display an image

```python
def imageShow (inp, title):
    '''
    # Function to display an image
    # Inputs:
    # inp - The input complex field
    # title - The title of the displayed image
    '''
    plt.imshow(inp, cmap='gray'), plt.title(title)
    plt.show()

    return
```

Image in gray scale

Show image

Code

Necessary functions

Pinhole

mm to px conversion

```
#-------------------------------------Propagating-------------------------------------

M = N = 256

# Defining the pinhole (circular aperture)
x0 = M/2                        #M/2
y0 = N/2                        #N/2
radius = 51.4                   #Radius = 2mm/(10mm/256) [px].  Diameter = 4mm
pinhole = np.zeros((M,N))       # Array filled with zeros

for j in range (M):
    for i in range (N):
        # Defining = 1 for everything inside the radius (this is the circular aperture)
        if np.power(j-x0, 2) + np.power(i-y0, 2) < np.power(radius, 2):
            pinhole[i,j] = 1
```

Number of horizontal and vertical data points

Pinhole radius in px

$$R = \frac{2mm}{(10mm/256)}$$

Creating an MxN matrix filled with zeros

Creating the circular aperture (a circle within the redius where all values are 1)

Code

Necessary functions

All parameters where taken from [1]

Plane wave

```
# Propagating distance (distance between the input and the output planes)
# 1264.22mm (Fresnel Number = 5) or 1053.52mm (Fresnel Number = 6)
z = 1264.22
#z = 1053.52

# Pinhole illuminated by a plane wave
input_wave = np.exp2(1j*k*z)*pinhole
```

z is the propagating distance

Pinhole illuminated by a plane wave

Code

Fresnel Transform

1

```
#----------------------------------------Fresnel Transform----------------------------------------
def fresnel(field, z, wavelength, dx, dy):
    """
    # Function to diffract a complex field using Fresnel approximation with
    # Fourier method
    # Inputs:
    # field - complex field
    # z - propagation distance
    # wavelength - wavelength
    # dx/dy - sampling pitches
    """

    x = np.arange(0, N, 1)  # array x
    y = np.arange(0, M, 1)  # array y
    X, Y = np.meshgrid(x - (N / 2), y - (M / 2), indexing='xy')

    dxout = (wavelength * z) / (M * dx)
    dyout = (wavelength * z) / (N * dy)

    k =  (2 * pi ) / wavelength

    z_phase = np.exp2((1j * k * z )/ (1j * wavelength * z))
    out_phase = np.exp2((1j * pi / (wavelength * z)) * (np.power(X * dxout, 2) + np.power(Y * dyout
    in_phase = np.exp2((1j * pi / (wavelength * z)) * (np.power(X * dx, 2) + np.power(Y * dy, 2)))

    tmp = (field * in_phase)
    tmp = np.fft.fftshift(tmp)
    tmp = np.fft.fft2(tmp)
    tmp = np.fft.fftshift(tmp)

    out = z_phase * out_phase * dx * dy * tmp

    return out
```

Wave number k

FFTs

Code

Fresnel Transform

Amplitude function

2

Input complex field

```
#-----------------------------------Amplitude definition---------------------------
def amplitude (inp, log):
    '''
    # Function to calcule the amplitude representation of a given complex field
    # Inputs:
    # inp - The input complex field
    # log - boolean variable to determine if a log representation is applied
    '''
    out = np.abs(inp)
    if log == True:
        out = 20 * np.log(out)
    return out
```

Code

Fresnel Transform

Pinhole

3

```
#-------------------------------------Propagating-------------------------------------

M = N = 256

# Defining the pinhole (circular aperture)
x0 = M/2                    #M/2
y0 = N/2                    #N/2
radius = 51.4               #Radius = 2mm/(10mm/256) [px].  Diameter = 4mm
pinhole = np.zeros((M,N))   # Array filled with zeros

for j in range (M):
    for i in range (N):
        # Defining = 1 for everything inside the radius (this is the circular aperture)
        if np.power(j-x0, 2) + np.power(i-y0, 2) < np.power(radius, 2):
            pinhole[i,j] = 1
```

Code

Fresnel Transform

4

```
#All units are calculated in mm
dx = 0.039                          # Input pitch
dy = 0.039                          # Input pitch
wavelength = 0.0006328              # Monochromatic light (red)
k =  (2 * pi ) / wavelength         # Wave number modulus

# Propagating distance (distance between the input and the output planes)
# 1264.22mm (Fresnel Number = 5) or 1053.52mm (Fresnel Number = 6)
z = 1264.22
#z = 1053.52

# Pinhole illuminated by a plane wave
input_wave = np.exp2(1j*k*z)*pinhole
```

Propagation parameters

Pinhole illuminated by plane wave

Code

Fresnel Transform

5

```
#Propagation from input plane (where the pinhole is) to output plane
temp = fresnel(input_wave, wavelength, z, dx, dx)
out = amplitude(temp, False)

#Display an gray value image with the given title
imageShow (out , 'Output Image with Fresnel Transform')
```

Calling our function (Fresnel Transform)

Using amplitude representation

Finally displaying output image

Code

Angular Spectrum

1

```python
#----------------------------------------Angular Spectrum-----------------------------------------
def angularSpectrum(field, z, wavelength, dx, dy):
    '''
    # Function to diffract a complex field using the angular spectrum approach
    # Inputs:
    # field - complex field
    # z - propagation distance
    # wavelength - wavelength
    # dx/dy - sampling pitches
    '''
    M, N = field.shape      # Number of horizontal and vertical data points
    x = np.arange(0, N, 1)  # array x
    y = np.arange(0, M, 1)  # array y
    X, Y = np.meshgrid(x - (N / 2), y - (M / 2), indexing='xy')

    dfx = 1 / (dx * M)
    dfy = 1 / (dy * N)

    field_spec = np.fft.fftshift(field)
    field_spec = np.fft.fft2(field_spec)
    field_spec = np.fft.fftshift(field_spec)

    phase = np.exp2(1j * z * pi * np.sqrt(np.power(1/wavelength, 2) - (np.power(X * dfx, 2) + np.power(Y * dfy, 2))))

    tmp = field_spec*phase

    out = np.fft.ifftshift(tmp)
    out = np.fft.ifft2(out)
    out = np.fft.ifftshift(out)

    return out
```

Code

Angular Spectrum

Intensity function

2

```
#-------------------------------Intensity definition-----------------------------
def intensity (inp, log):
    '''
    # Function to calcule the intensity representation of a given complex field
    # Inputs:
    # inp - The input complex field
    # log - boolean variable to determine if a log representation is applied
    '''
    out = np.abs(inp)
    out = out*out
    if log == True:
        out = 20 * np.log(out)
    return out
```

Luz circularmente polarizada hacia la izquierda. Obtenido de t.ly/JgzA

Code

Fresnel Transform

Pinhole

3

```
M = N = 256

# Defining the pinhole (circular aperture)
x0 = M/2
y0 = N/2
radius = 25.6                # Radius = 1mm/(10mm/256) [px] Diameter = 2mm.
pinhole = np.zeros((M,N))    # Array filled with zeros

for j in range (M):
    for i in range (N):
        # Defining = 1 for everything inside the radius (this is the circular aperture)
        if np.power(j-x0, 2) + np.power(i-y0, 2) < np.power(radius, 2):
            pinhole[i,j] = 1
```

Code

Angular Spectrum

4

```
#All units are calculated in mm
dx = 0.039                          # Input pitch
dy = 0.039                          # Input pitch
wavelength = 0.0006328              # Monochromatic light (red)
k =  (2 * pi ) / wavelength         # Wave number modulus

# Propagating distance (distance between the input and the output planes)
# 158.03mm (Fresnel Number = 10) or 143.66mm (Fresnel Number = 11)
#z = 158.03
#z = 143.66
z = 300

# Pinhole illuminated by a plane wave
input_wave = np.exp2(1j*k*z)*pinhole
```

Propagation parameters

Pinhole illuminated by plane wave

Code

Angular Spectrum

5

```
#Propagation from input plane (where the pinhole is) to output plane
complexfield = angularSpectrum(input_wave, z, wavelength, dx, dx)

#This function calculates the amplitude representation of a given complex fi
out = intensity(complexfield, False)

#Display a gray-value image with the given title
imageShow (out,'Output Image with Angular Spectrum' )
```

Calling our function (Angular Spectrum)

Using intensity representation

Finally displaying output image

Code

Verifying the propagation was done properly

6

```
#Verifying if the propagation was executed properly
Fresnel_num = 2**2 / (wavelength * z)
print('Fresnel number: ',Fresnel_num)
round_Fresnel = round(Fresnel_num)
if (round_Fresnel % 2) == 0:
   print("{0} is even. There should be a dark spot in the center of the diffraction pattern".format(round_Fresnel))
else:
   print("{0} is odd. There should be a bright spot in the center of the diffraction pattern".format(round_Fresnel))
```

Computing the Fresnel number

Rounding up Fresnel number

Checking if Fresnel number is even or odd.

If Fresnel number is **even** → Dark spot in center
If Fresnel number is **odd** → Bright sport in center

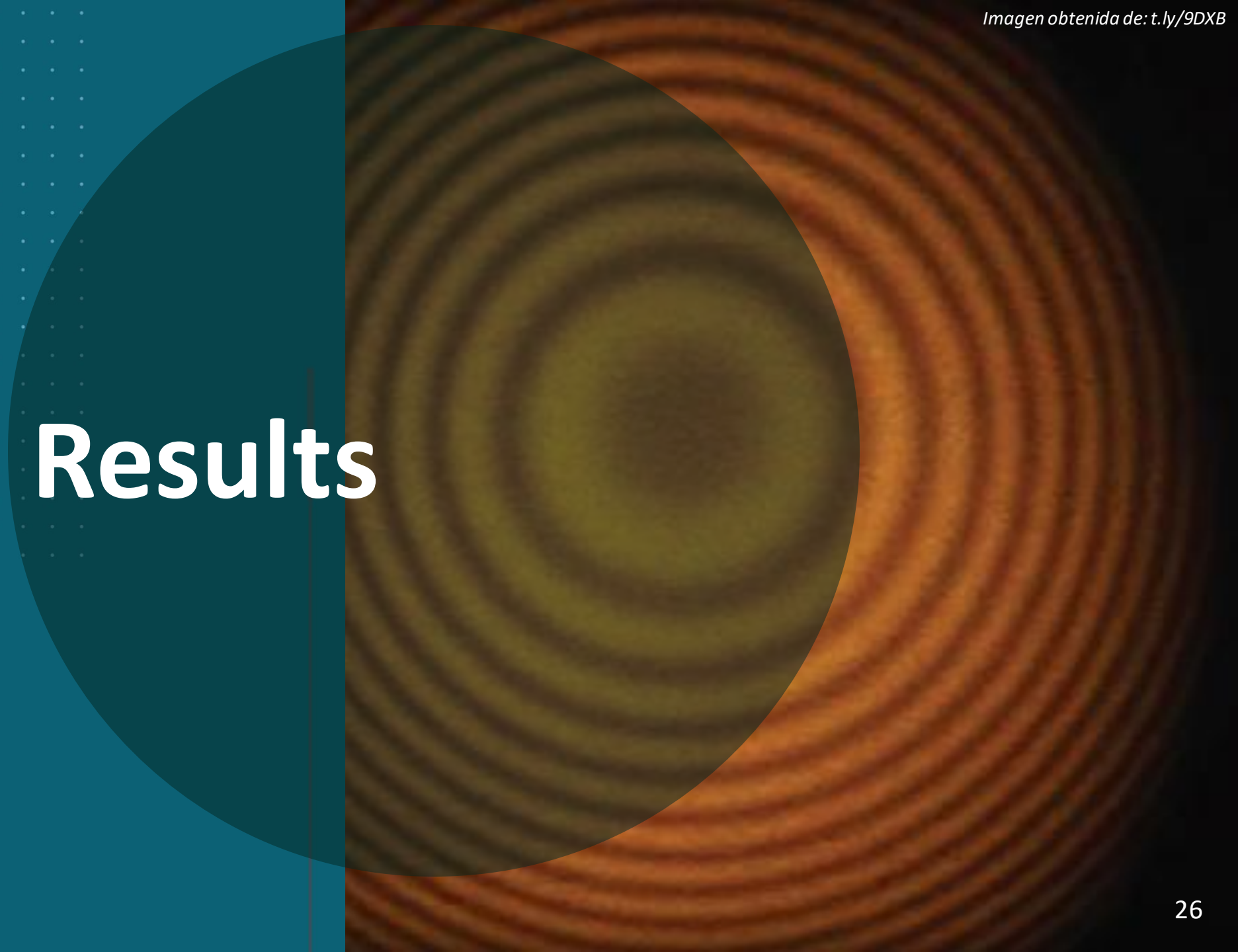
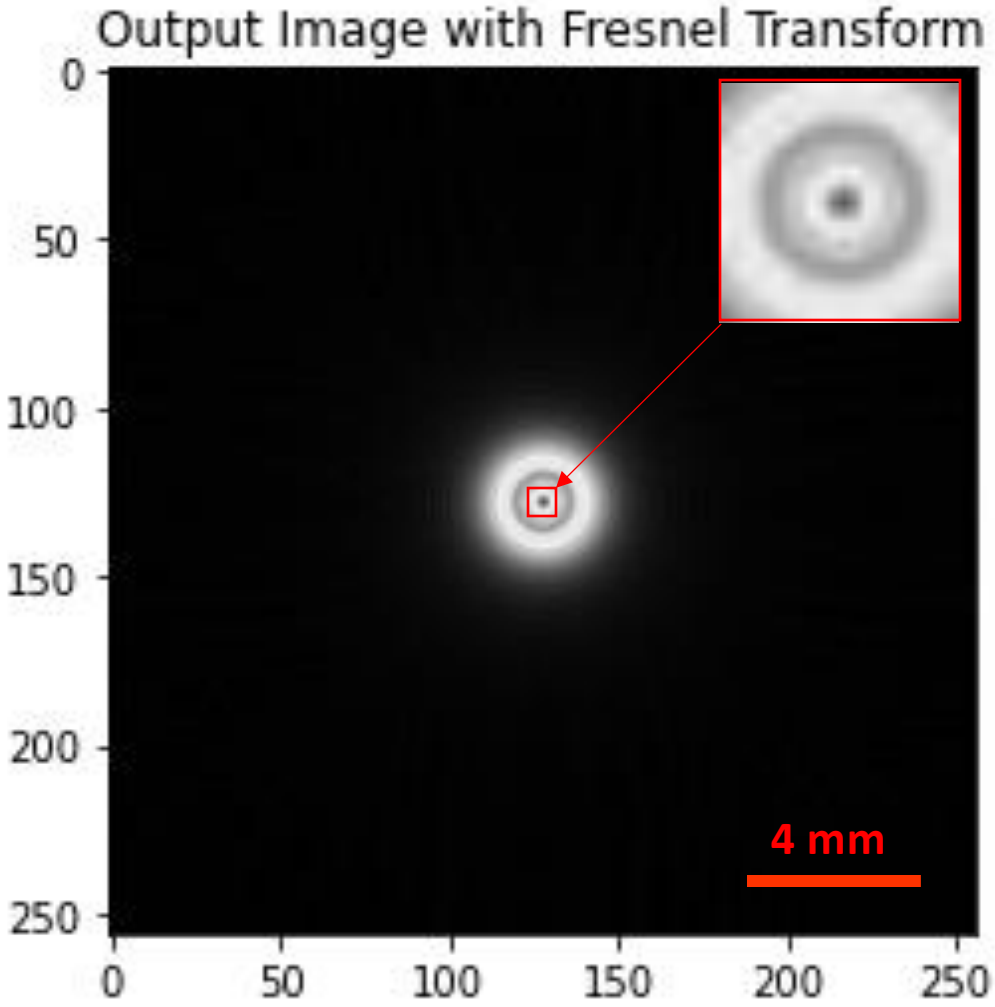


Imagen obtenida de: t.ly/9DXB

Results

Results

Fresnel Transform



Output image with Fresnel Tranform for 1053.52 mm propagation distance.

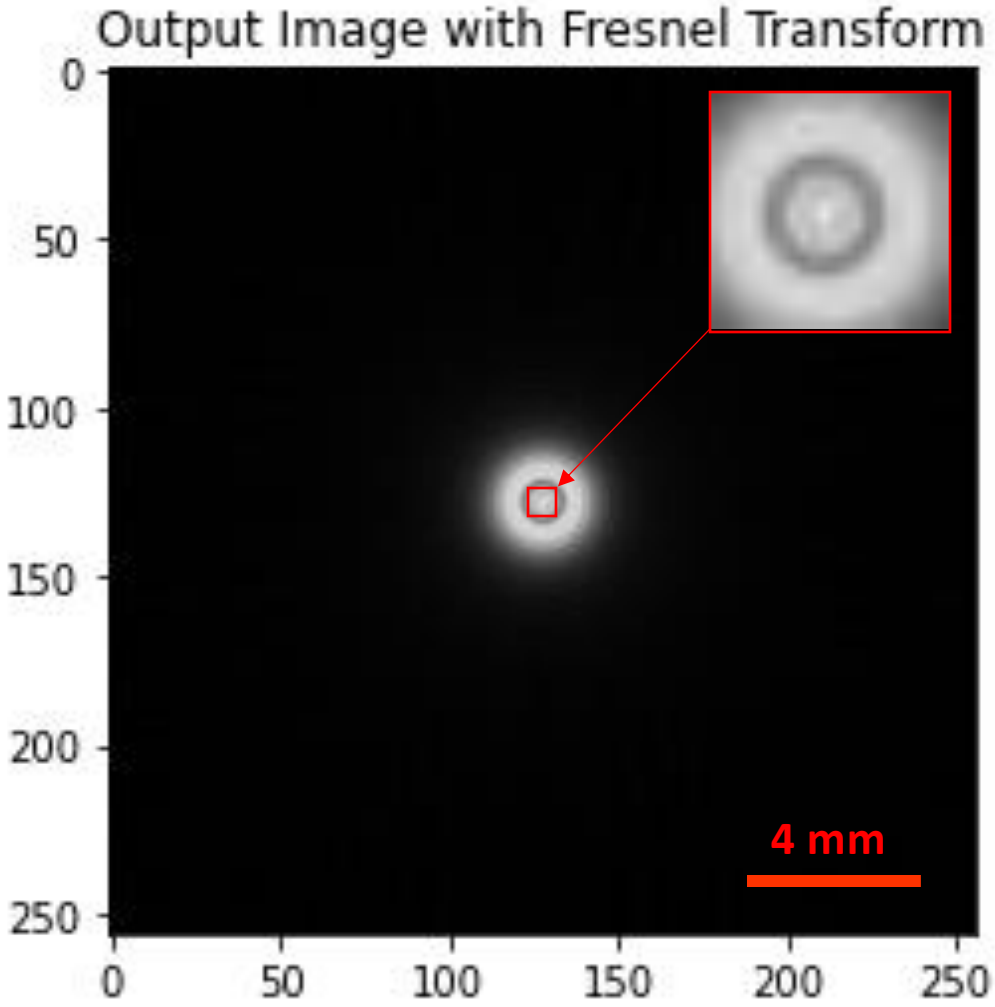
```
Fresnel number:  5.999992896008411
6 is Even. There should be a dark spot in the
center of the diffraction pattern
```

Fresnel number ≈ 6 → According to [1], it should be 6 for this propagation distance.

Fresnel number ≈ 6 → Even → Dark spot in the center

Results

Fresnel Transform



Output image with Fresnel Tranform for 1264.22 mm propagation distance.

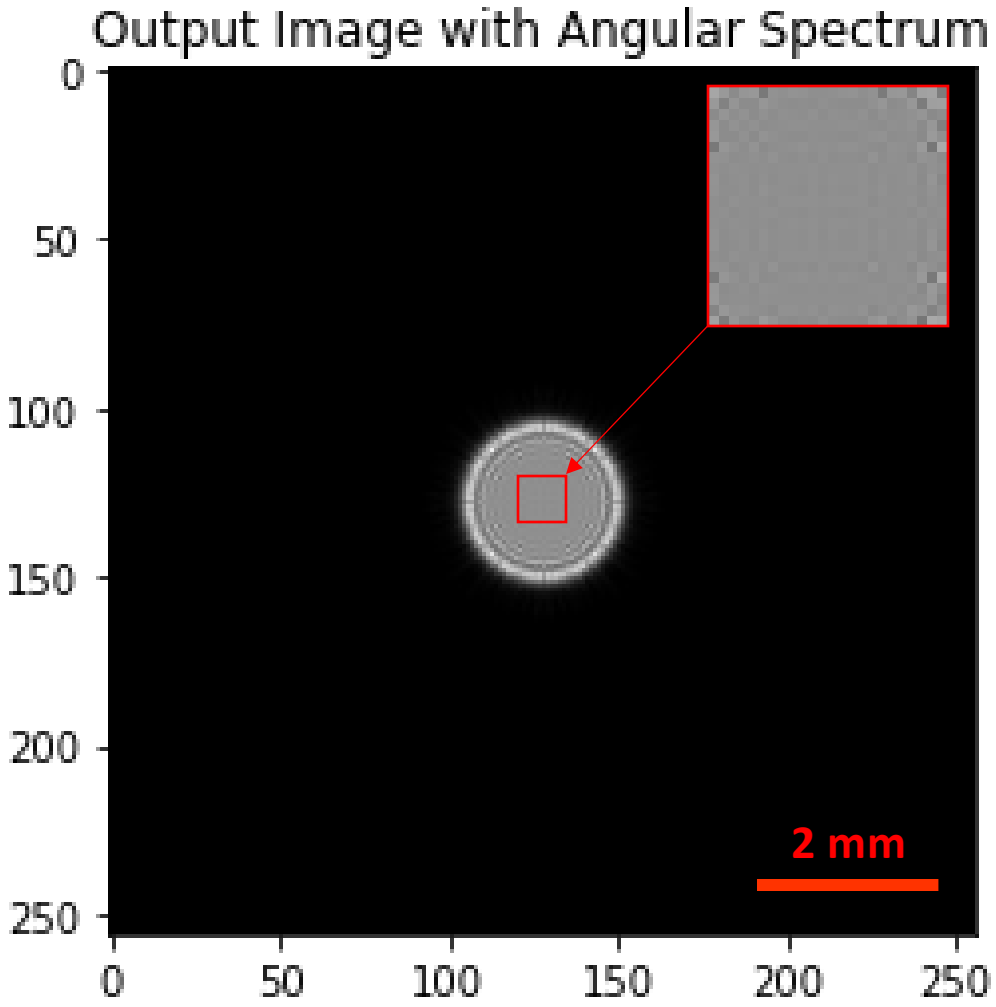
```
Fresnel number:  5.000009900019602
5 is odd. There should be a bright spot in the
center of the diffraction pattern
```

Fresnel number ≈ 5 → According to [1], it should be 5 for this propagation distance.

Fresnel number ≈ 5 → Odd → Bright spot in the center

Results

Angular Spectrum



Output image with Angulas Spectrum for 143.66 mm propagation distance.

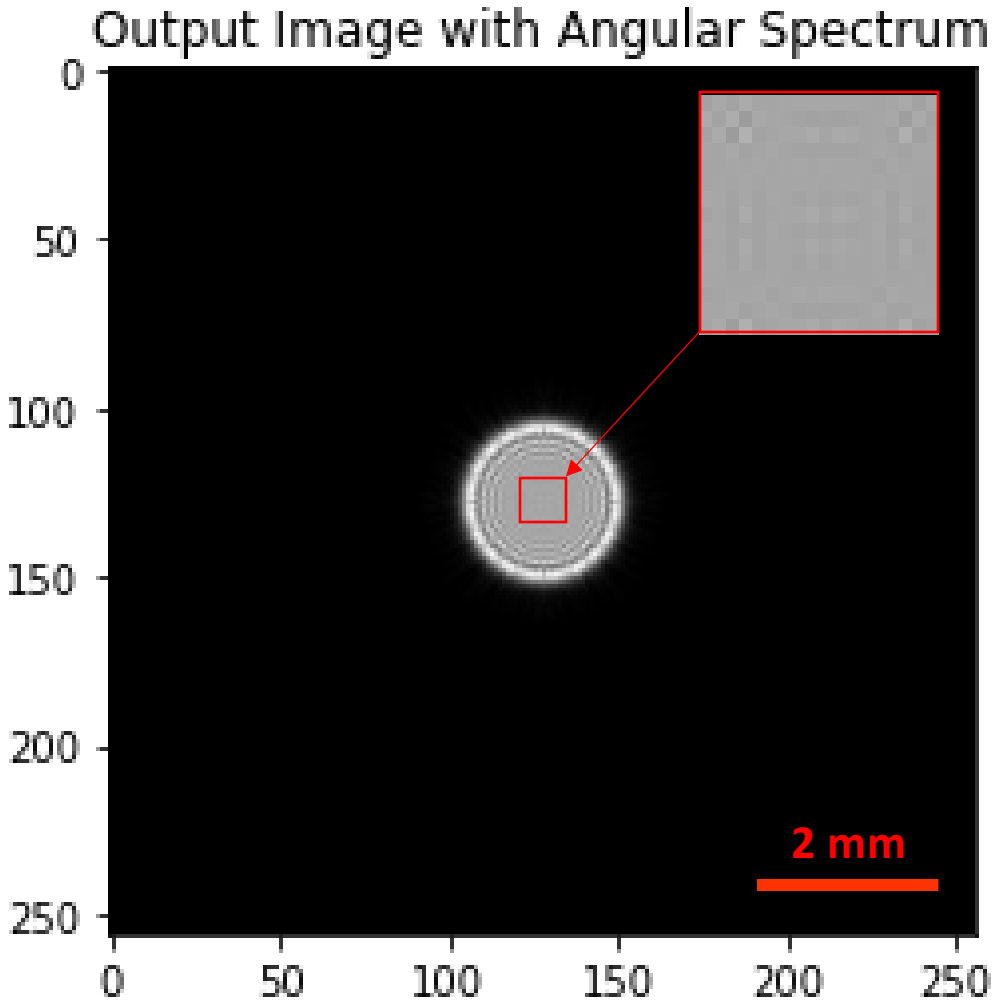
```
Fresnel number:  11.000126193447691
11 is odd. There should be a bright spot in the
center of the diffraction pattern
```

Fresnel number ≈ 11 → According to [1], it should be 11 for this propagation distance.

Fresnel number ≈ 11 → Odd → Bright spot in the center

Results

Angular Spectrum



Output image with Angulas Spectrum for 158.03 mm propagation distance.

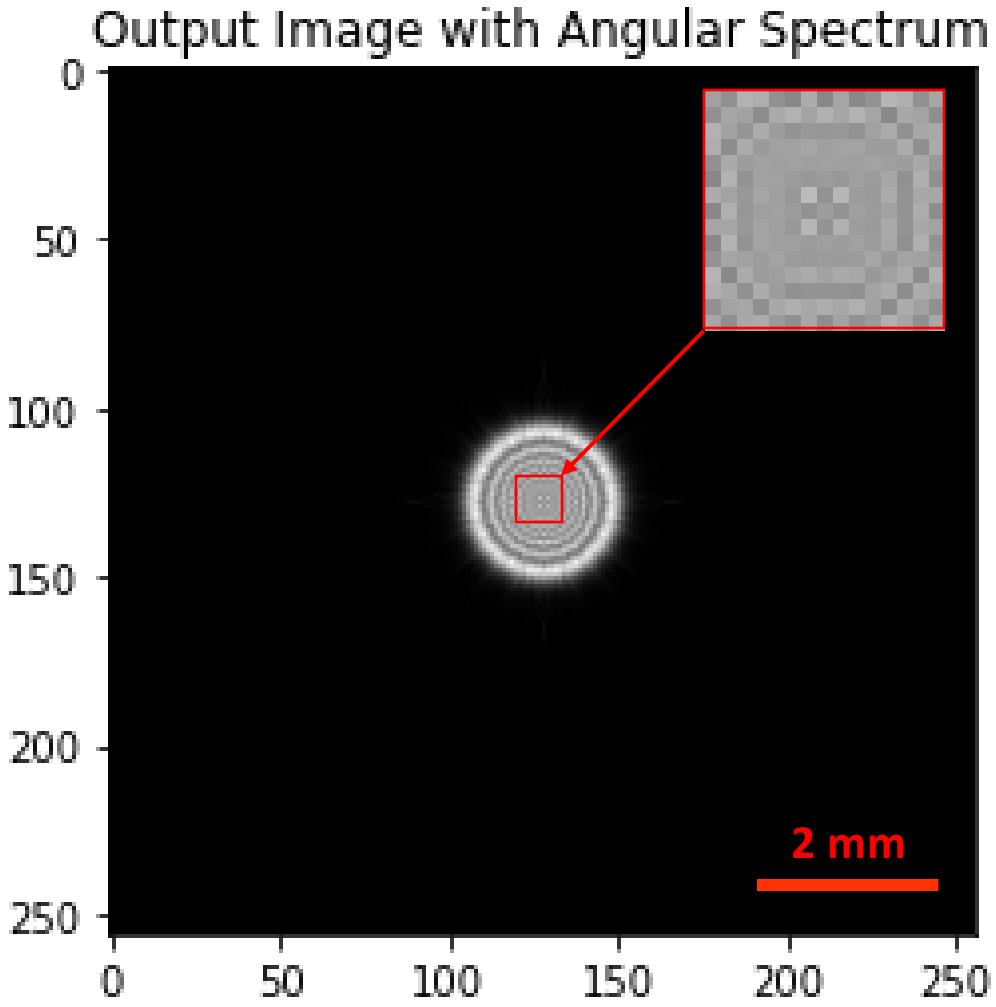
```
Fresnel number:  9.99986160191543
10 is even. There should be a dark spot in the
center of the diffraction pattern
```

Fresnel number ≈ 10 → According to [1], it should be 10 for this propagation distance.

Fresnel number ≈ 10 → Even → Dark spot in the center

Results

Angular Spectrum



Output image with Angulas Spectrum for 263 mm propagation distance.

```
Fresnel number:  6.008662087265001
6 is even. There should be a dark spot in the
center of the diffraction pattern
```

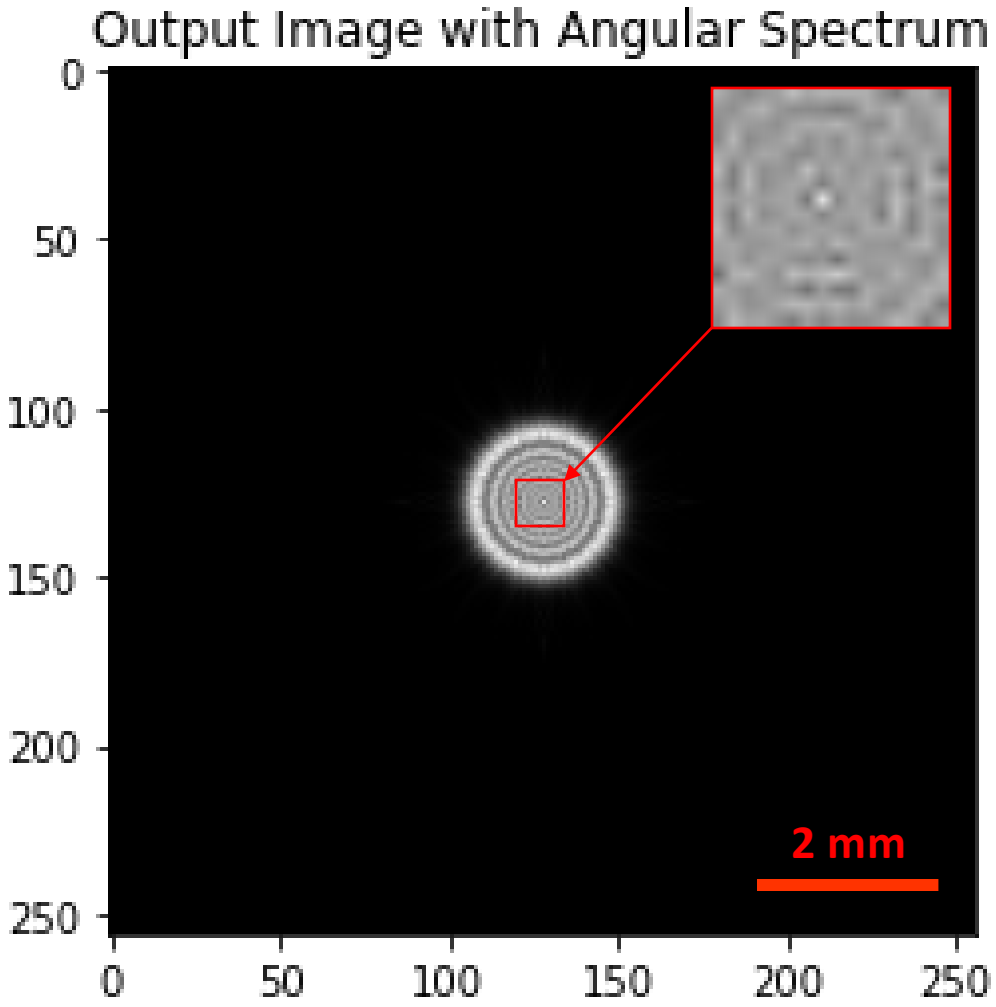
Fresnel number ≈ 6

↓

Even→ Dark spot in the center

Results

Angular Spectrum



Output image with Angulas Spectrum for 300 mm propagation distance.

```
Fresnel number:  5.267593763168984
5 is odd. There should be a bright spot in the
center of the diffraction pattern
```

Fresnel number ≈ 5

↓

Odd → Bright spot in the center

References

Bibliography

[1] P. Piedrahita-Quintero, C. Trujillo & J. Garcia-Sucerquia, «JDiffraction: A GPGPU-accelerated JAVA library for numerical propagation of scalar wave fields,» *Elsevier,* pp. 128-139, 2017

[2] NumPy, «NumPy,» NumPy v1.21 Manual, 22 Junio 2021. [Online]. Available: t.ly/6vbk. [Último acceso: 9 August 2021].

[3] matplotlib, «Matplotlib: Visualization with Python,» 30 Agosto 2021. [Online]. Available: t.ly/pNpS.

[4] The Python Standard Library, «math — Mathematical functions,» August 2021. [Online]. Available: t.ly/BtMO.

Thank you!

Inspira Crea Transforma