Lab 2

1. Abstract

In this lab we explore the ideas of sampling theorem, not only in the spatial domain but also in the fourier domain. We see how aliasing results in a distorted field and how to correct it. The function of a lens as a fourier transform is visualized for gaussian and vortex beams and we also look into the talbot effect for a grating.

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import imageio.v3 as io
from skimage import color
```

2. Ideas of Sampling

- Define the grids in both domains.

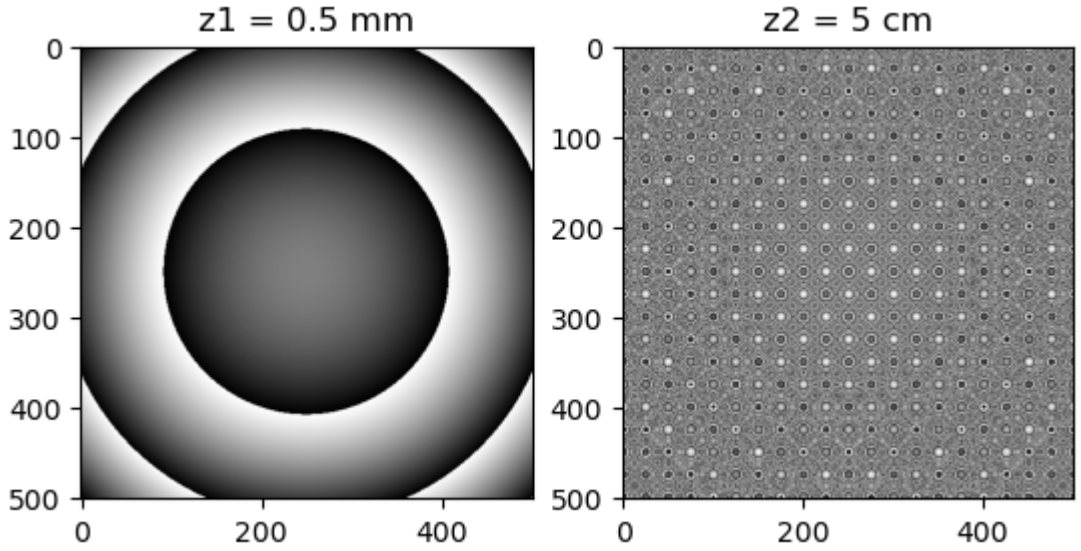
In [2]:
```
x0 = np.linspace(-250, 250, 501, endpoint = True) # Define a linspace (pixel size = 1)
x, y = np.meshgrid(x0, x0)

f0 = np.linspace(-0.5, 0.5, 501, endpoint = True) # Define a linspace in fourier domain
fx, fy = np.meshgrid(f0, f0)

p = 5e-6 # Pixel size

# Scaling the grid with spacing equals to pixel size
x, y = x*p, y*p
fx, fy = fx/p, fy/p

wavelength = 0.5e-6
k = 2*np.pi/wavelength
```

```
# Distances for propagation
z1 = 0.5e-3
z2 = 50e-2
```

- Note that the coordinates in the fourier domain are in range -0.5 to 0.5 because we took pixel size = 1 in the x0. The pixel size (i.e. spacing) becomes the total extent in the fourier domain.

In [3]:
```
# Helper functions
def ft2(x):
    return np.fft.fftshift(np.fft.fft2(np.fft.ifftshift(x))) # Fourier transform

def ift2(x):
    return np.fft.fftshift(np.fft.ifft2(np.fft.ifftshift(x))) # Inverse Fourier transform
```

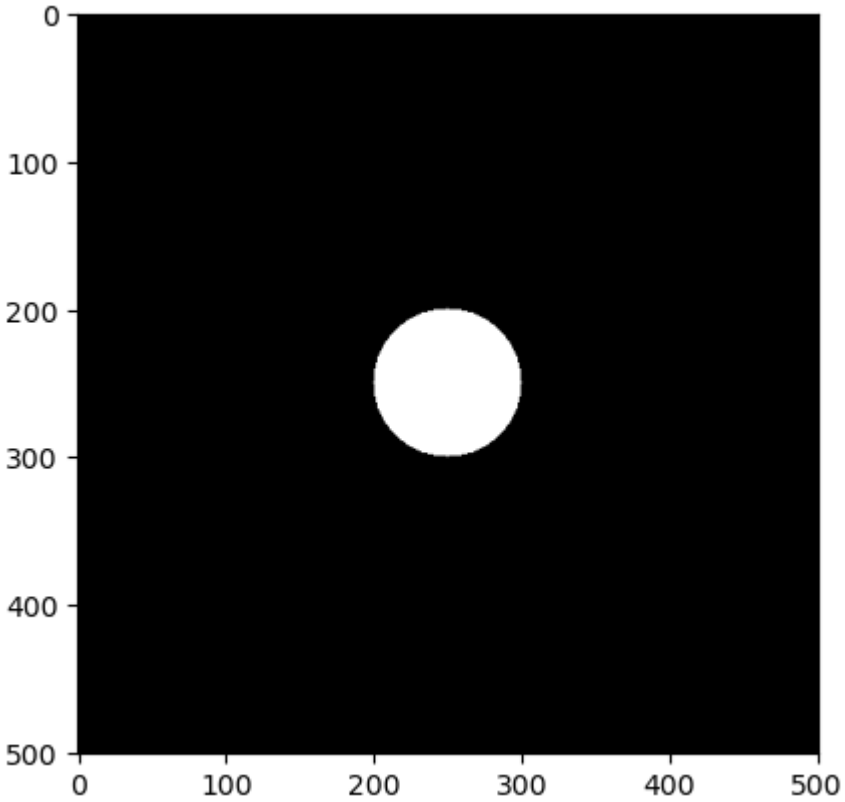
In [4]:
```
# Displaying the propagation function for 0.5mm and 50cm distance.

alpha = np.sqrt(k**2 - 4*np.pi**2*(fx**2 + fy**2))

H1 = np.exp(1j*alpha*z1) # Propagation function for z1 = 0.5mm
H2 = np.exp(1j*alpha*z2) # Propagation function for z2 = 50cm

# Display the results
plt.figure()
plt.subplot(1,2,1)
plt.imshow(np.angle(H1), cmap = "gray")
plt.title("z1 = 0.5 mm")
plt.subplot(1,2,2)
plt.imshow(np.angle(H2), cmap = "gray")
plt.title("z2 = 5 cm")
plt.show()
```



Let's see how the propagation function works when applied on a circular aperture.

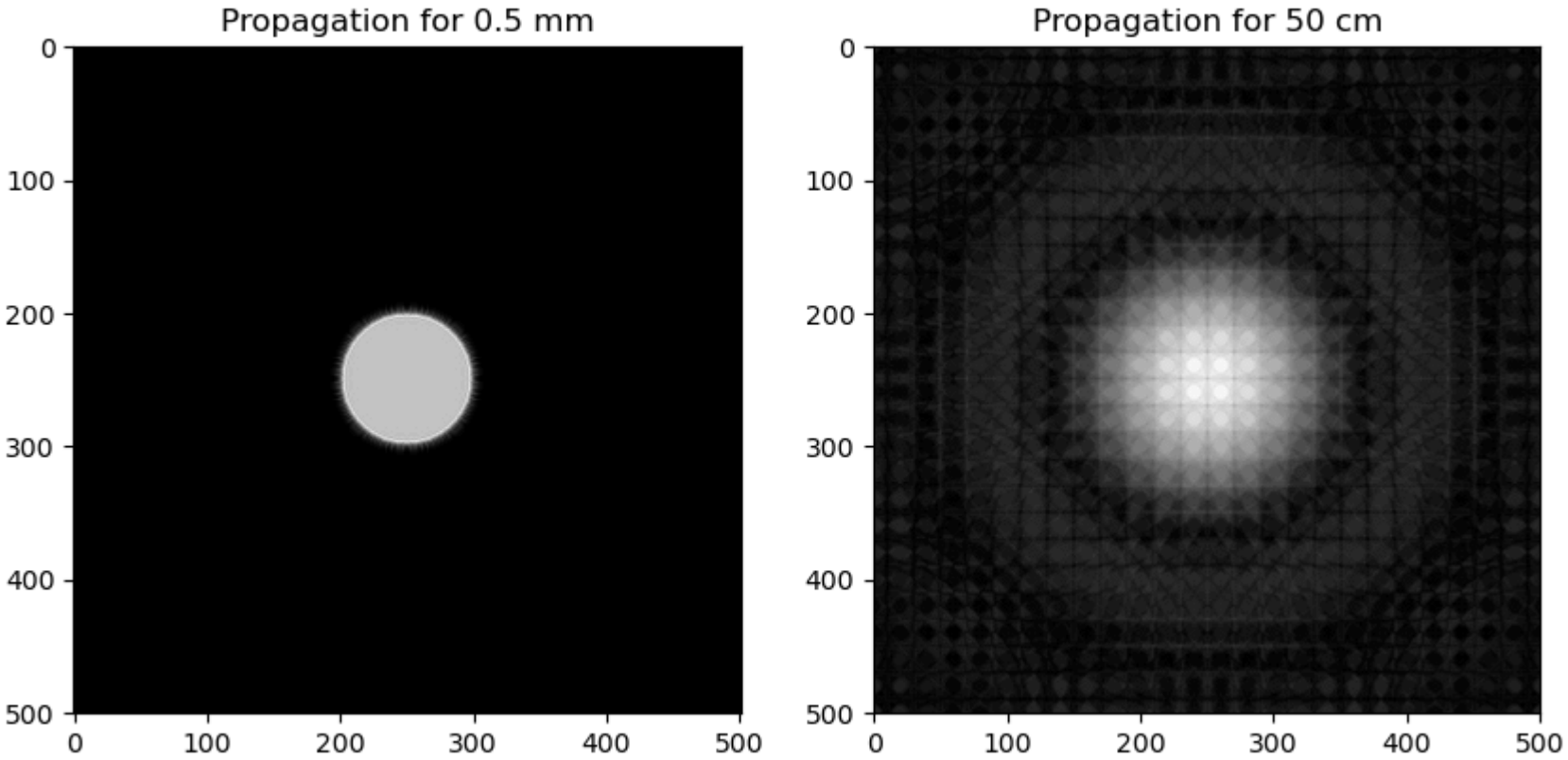
In [5]:
```
# Define a circular aperture
N = 501
circ = np.zeros((N, N))
circ[x**2 + y**2 <= (50*p)**2] = 1 # Circular aperture of radius = 50*p
plt.imshow(circ, cmap = "gray") # Display the aperture
```

Out[5]:
```
<matplotlib.image.AxesImage at 0x1f3cbe749d0>
```



In [6]:
```
# Fourier Transform the circle
circ_f = ft2(circ)

# Propagation of the circle
circ_f1 = circ_f*H1 # z1 = 0.5mm
circ_f2 = circ_f*H2 # z2 = 50cm

# Inverse transform the field at z1 and z2 distances
circ_if1 = ift2(circ_f1)
circ_if2 = ift2(circ_f2)

# Display the magnitude of the propagated fields
plt.figure(figsize = (10, 5))
plt.subplot(1,2,1)
plt.imshow(np.abs(circ_if1), cmap = "gray")
```

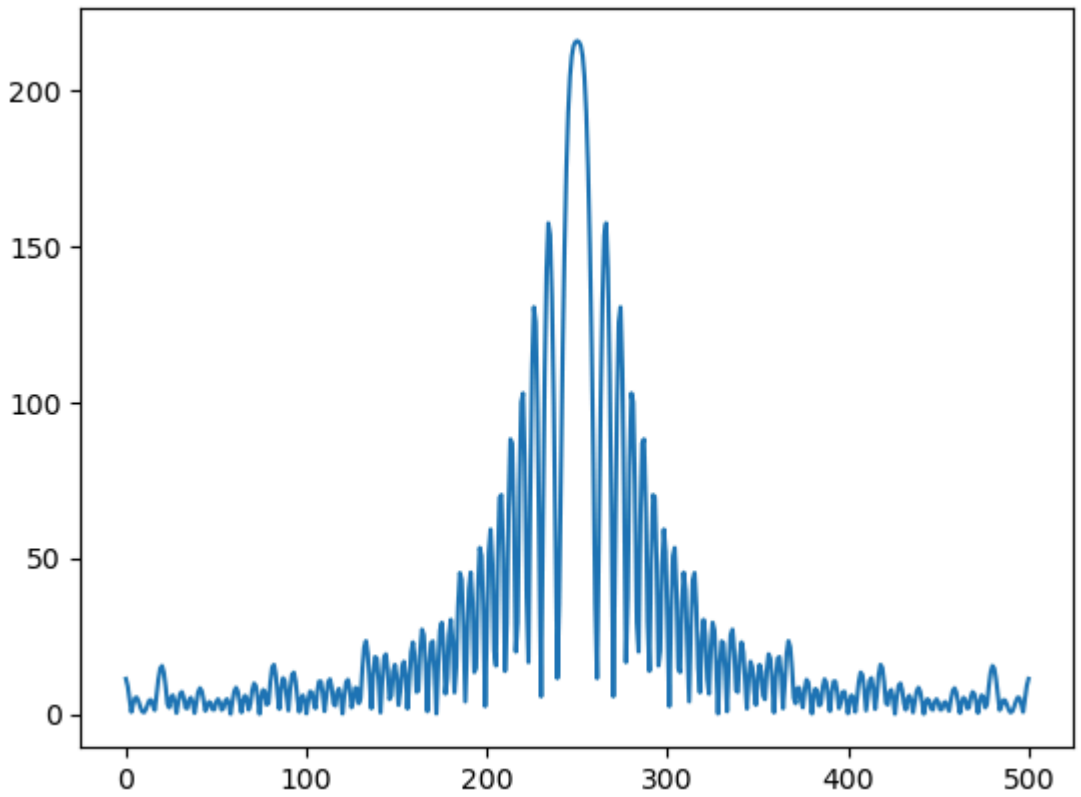
```
plt.title("Propagation for 0.5 mm")
plt.subplot(1,2,2)
plt.imshow(np.abs(circ_if2), cmap = "gray")
plt.title("Propagation for 50 cm")
plt.show()
```



- The grid like pattern at longer distances do not make sense!
- This aliasing effect is due to improper sampling. In general, to preserve the full information in the signal, it is necessary to sample at twice the maximum frequency of the signal. This is know as the Nyquist rate.
- For the proper sampling we derive a low pass filter such that the maximum phase change between two samples is equal to π.

In [7]:
```
plt.plot(np.abs(circ_f1)[226,:])
```

Out[7]: [<matplotlib.lines.Line2D at 0x1f3cdfc0710>]

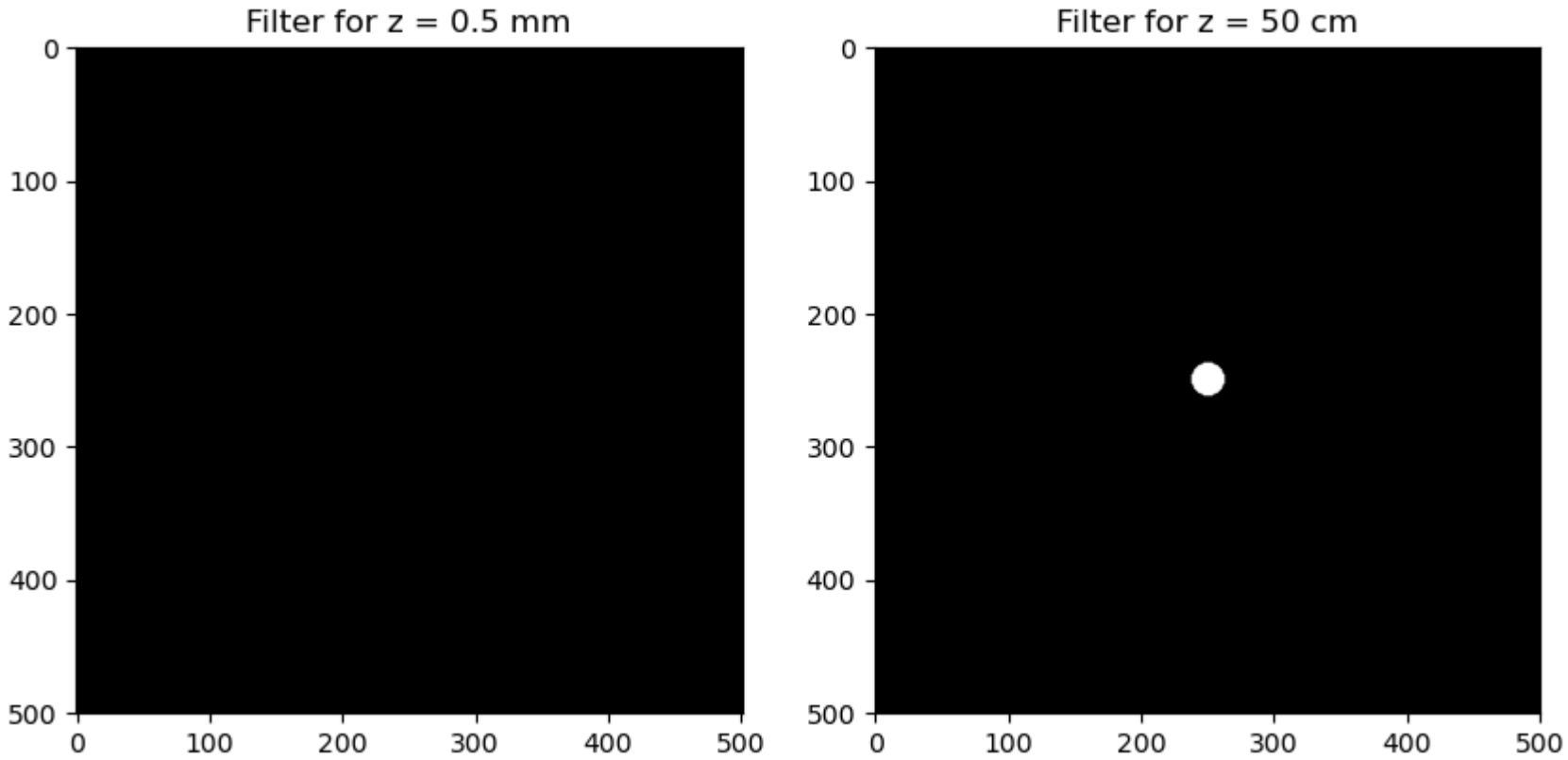


3. A low pass filter in the fourier domain

In [8]:
```
# Adding a low pass filter
lp_f1 = (1/wavelength)*(1/(np.sqrt(1 + (2*z1/(N*p))**2))) # Low pass frequency for z = z1
lp_f2 = (1/wavelength)*(1/(np.sqrt(1 + (2*z2/(N*p))**2))) # Low pass frequency for z = z2
filtr1 = np.zeros((N, N))
filtr1[fx**2 + fy**2 < lp_f1**2] = 1 # Low pass filter for z = z1
filtr2 = np.zeros((N, N))
filtr2[fx**2 + fy**2 < lp_f2**2] = 1 # Low pass filter for z = z2

# Display the filters
plt.figure(figsize = (10,5))
```

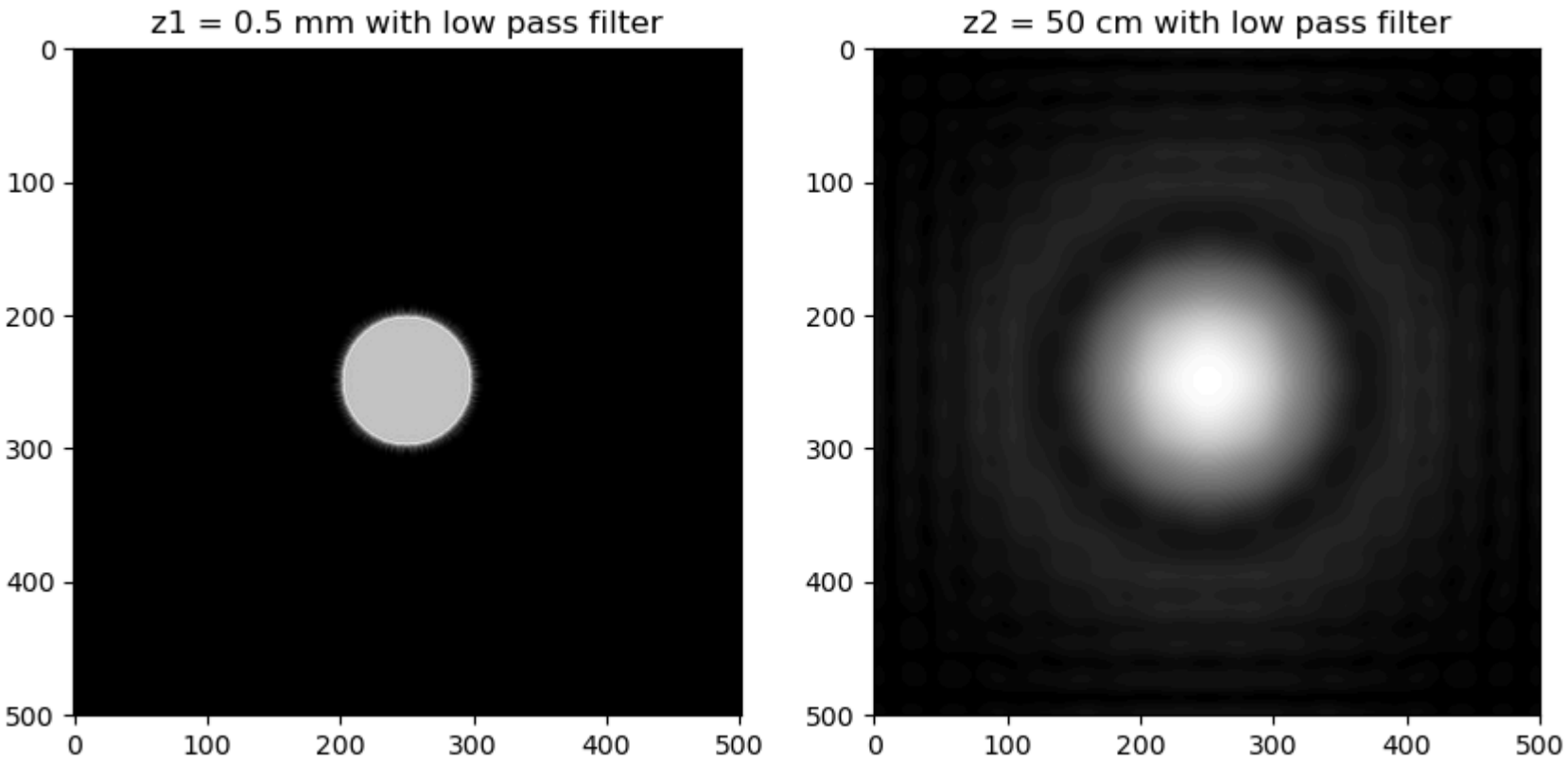
```
plt.subplot(1,2,1)
plt.imshow(filtr1, cmap = "gray")
plt.title("Filter for z = 0.5 mm")
plt.subplot(1,2,2)
plt.imshow(filtr2, cmap = "gray")
plt.title("Filter for z = 50 cm")
```

Out[8]: Text(0.5, 1.0, 'Filter for z = 50 cm')



In [9]:
```
# Applying the filters with propagation function for both distances.
circ_f1_fltrd = circ_f1*filtr1
circ_f2_fltrd = circ_f2*filtr2
circ_if1_fltrd = ift2(circ_f1_fltrd)
circ_if2_fltrd = ift2(circ_f2_fltrd)

# Display the results after proper filter
```

```
plt.figure(figsize = (10,5))
plt.subplot(1,2,1)
plt.imshow(np.abs(circ_if1_fltrd), cmap = "gray")
plt.title("z1 = 0.5 mm with low pass filter")
plt.subplot(1,2,2)
plt.imshow(np.abs(circ_if2_fltrd), cmap = "gray")
plt.title("z2 = 50 cm with low pass filter")
plt.show()
```

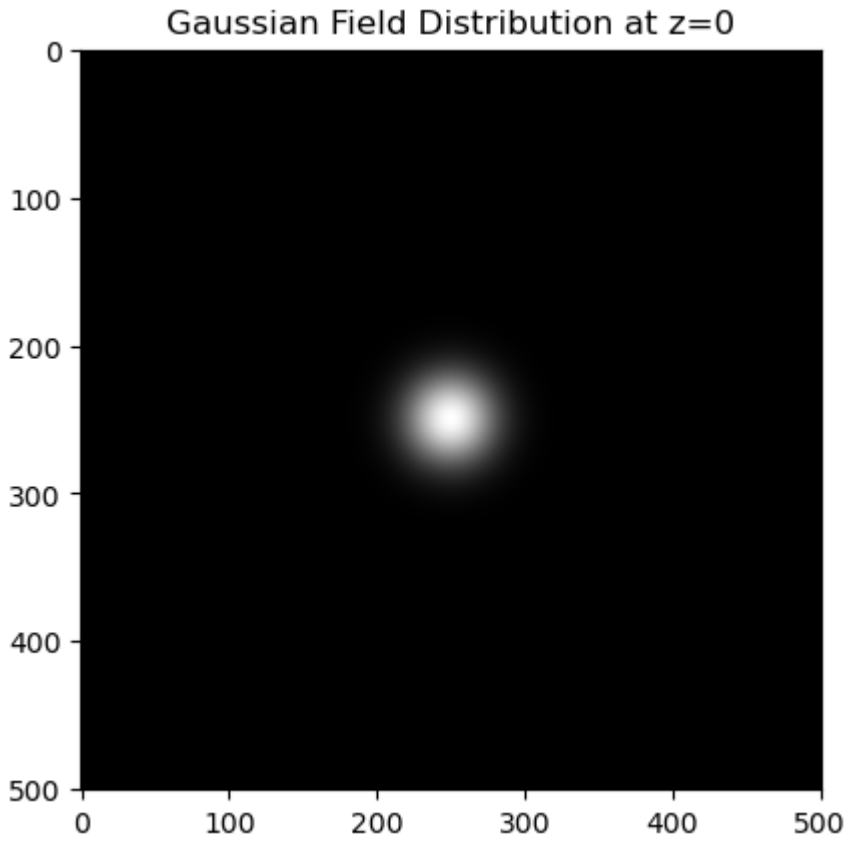


The aliasing effect is no longer visible for z = 50cm!!

4. Propagation of a Gaussian Beam

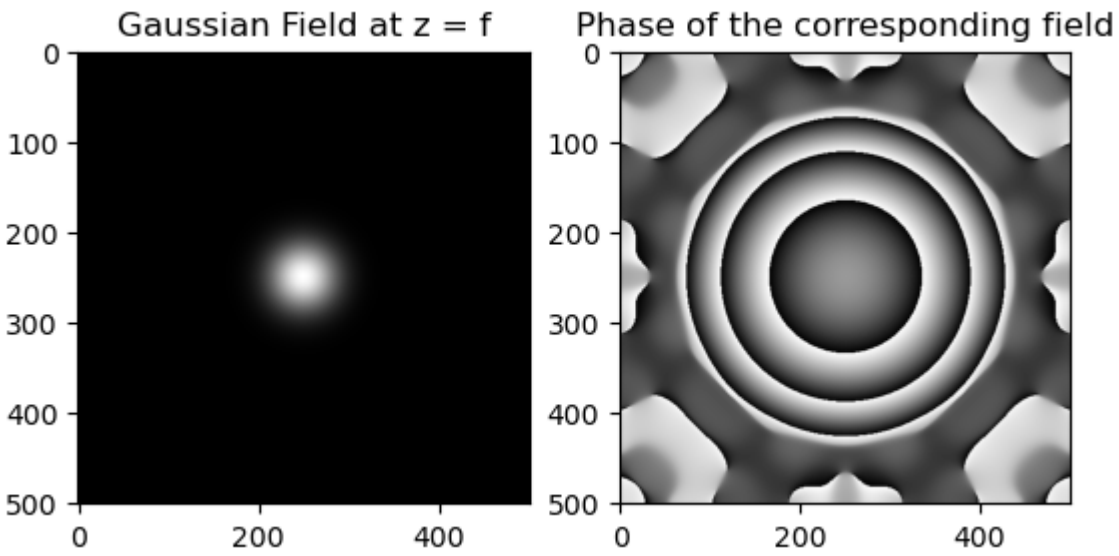
In [10]:

```
f= 0.1 # Focal length
z3 = 0 # Inpput plane
w0 = 30*p # Width of the gaussian beam

# Defining a Gaussian field at z = 0 plane
e_0 = np.exp(-(x**2 + y**2)/w0**2)

# Display the gaussian beam
plt.figure()
plt.imshow(e_0, cmap = "gray")
plt.title("Gaussian Field Distribution at z=0")
plt.show()
```



In [11]:

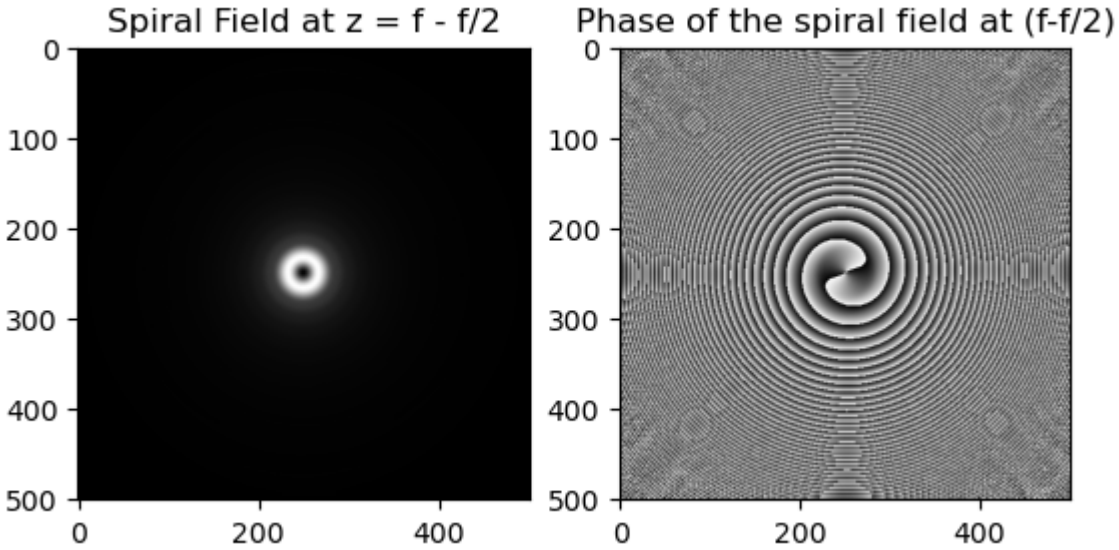
```
# Defining the lens transmittance function
circ_lens = np.zeros((N, N))
circ_lens[x**2 + y**2 < (70*p)**2] = 1

lens = np.exp(-1j*k*(x**2 + y**2)/(2*f)) * circ_lens

# Taking the fourier transform and propagating the field for z = f distance
E_0 = ft2(e_0)

# Propagation function for z = f distance
H3 = np.exp(-1j*alpha*f)

# Defining the low pass filter
lp_f3 = (1/wavelength)*(1/(np.sqrt(1 + (2*f/(N*p))**2)))
filtr3 = np.zeros((N, N))
filtr3[fx**2 + fy**2 <= lp_f3**2] = 1

E_f = E_0*H3*filtr3
e_f = ift2(E_f) # Inverse fourier transform the propagated field

# Gaussian field at the focus of the lens
plt.figure()
plt.subplot(1,2,1)
plt.imshow(np.abs(e_f), cmap = "gray")
plt.title("Gaussian Field at z = f")
plt.subplot(1,2,2)
plt.imshow(np.angle(e_f), cmap = "gray")
plt.title("Phase of the corresponding field")
plt.show()
```



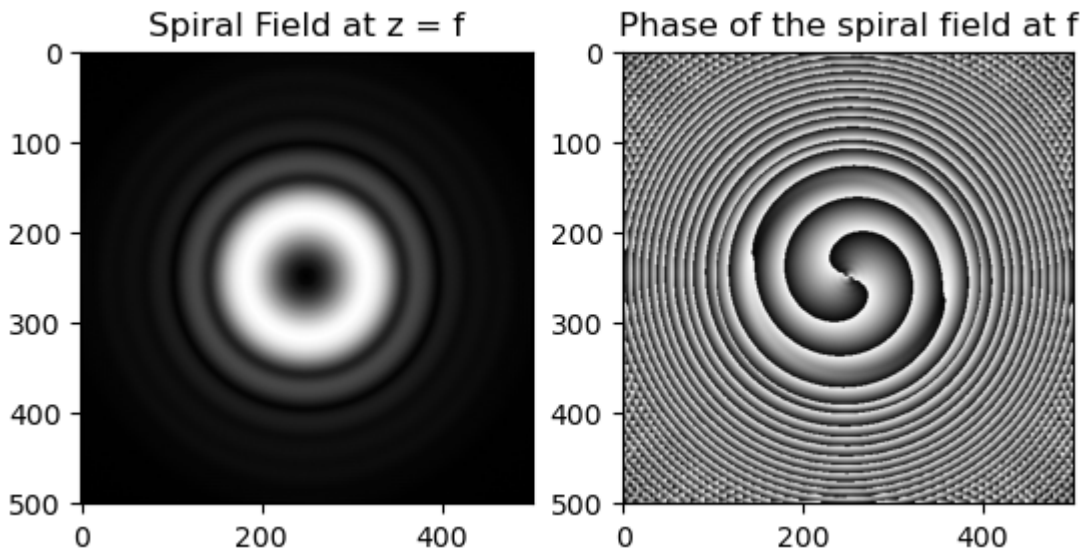
5. Introducing a spiral phase and propagating it for different distances.

In [12]:
```
# Applying the lens transmittance again with a spiral phase
theta = np.arctan2(y, x)
e_fspiral = e_f * lens * np.exp(1j*2*theta) # np.exp(1j*2*theta) is the phase function added corresponding to l=2

# Again use the angular spectrum propagation
E_fspiral = ft2(e_fspiral)

z4 = f - (f/2) # Before the focus
z5 = f + (f/2) # After the focus

H4 = np.exp(1j*alpha*z4) # Propagation function for z = z4 = f - f/2

lp_f4 = (1/wavelength)*(1/(np.sqrt(1 + (2*z4/(N*p))**2)))
filtr4 = np.zeros((N, N))
filtr4[fx**2 + fy**2 <= lp_f4**2] = 1 # Low pass filter

E_z4 = E_fspiral * H4 * filtr4 # Propagation of the spiral field
e_z4 = ift2(E_z4) # Field at z = (f - f/2)
```

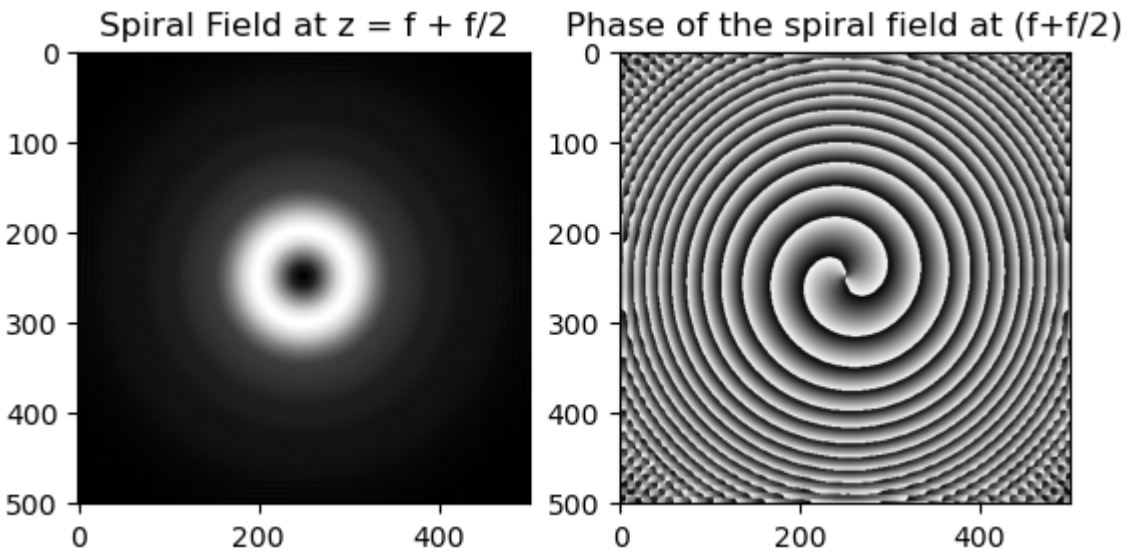
```
# Display the field at z = f - f/2
plt.figure()
plt.subplot(1,2,1)
plt.imshow(np.abs(e_z4), cmap = "gray")
plt.title("Spiral Field at z = f - f/2")
plt.subplot(1,2,2)
plt.imshow(np.angle(e_z4), cmap = "gray")
plt.title("Phase of the spiral field at (f-f/2)")
plt.show()
```



In [13]:
```
# Propagation for z = f distance
E_zf = E_fspiral * H3 * filtr3
e_zf = ift2(E_zf)

plt.figure()
plt.subplot(1,2,1)
plt.imshow(np.abs(e_zf), cmap = "gray")
plt.title("Spiral Field at z = f")
plt.subplot(1,2,2)
plt.imshow(np.angle(e_zf), cmap = "gray")
plt.title("Phase of the spiral field at f")
plt.show()
```

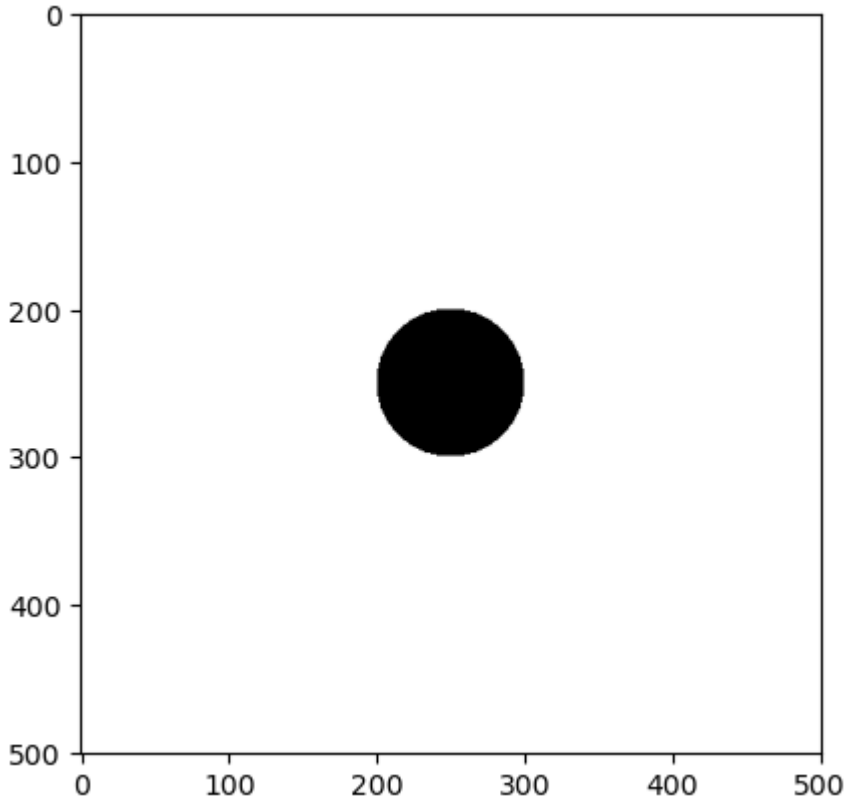


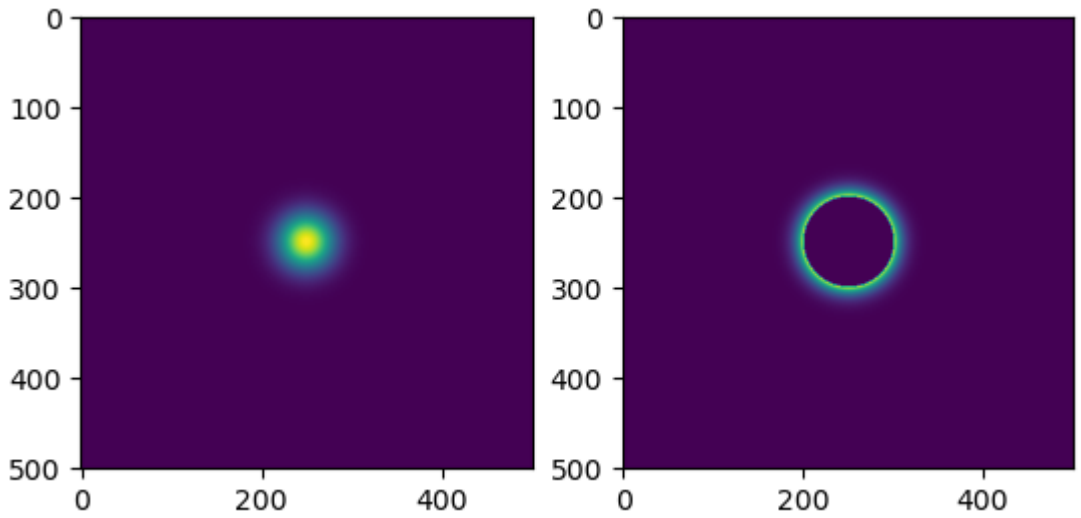
In [14]:
```
# Propagation function for z = z5 = f + f/2
H5 = np.exp(1j * alpha * z5)

lp_f5 = (1/wavelength)*(1/(np.sqrt(1 + (2*z5/(N*p))**2)))
filtr5 = np.zeros((N, N))
filtr5[fx**2 + fy**2 <= lp_f5**2] = 1

E_z5 = E_fspiral * H5 * filtr5 # Propagation for z = (f + f/2) distance
e_z5 = ift2(E_z5)

plt.figure()
plt.subplot(1,2,1)
plt.imshow(np.abs(e_z5), cmap = "gray")
plt.title("Spiral Field at z = f + f/2")
plt.subplot(1,2,2)
plt.imshow(np.angle(e_z5), cmap = "gray")
plt.title("Phase of the spiral field at (f+f/2)")
plt.show()
```



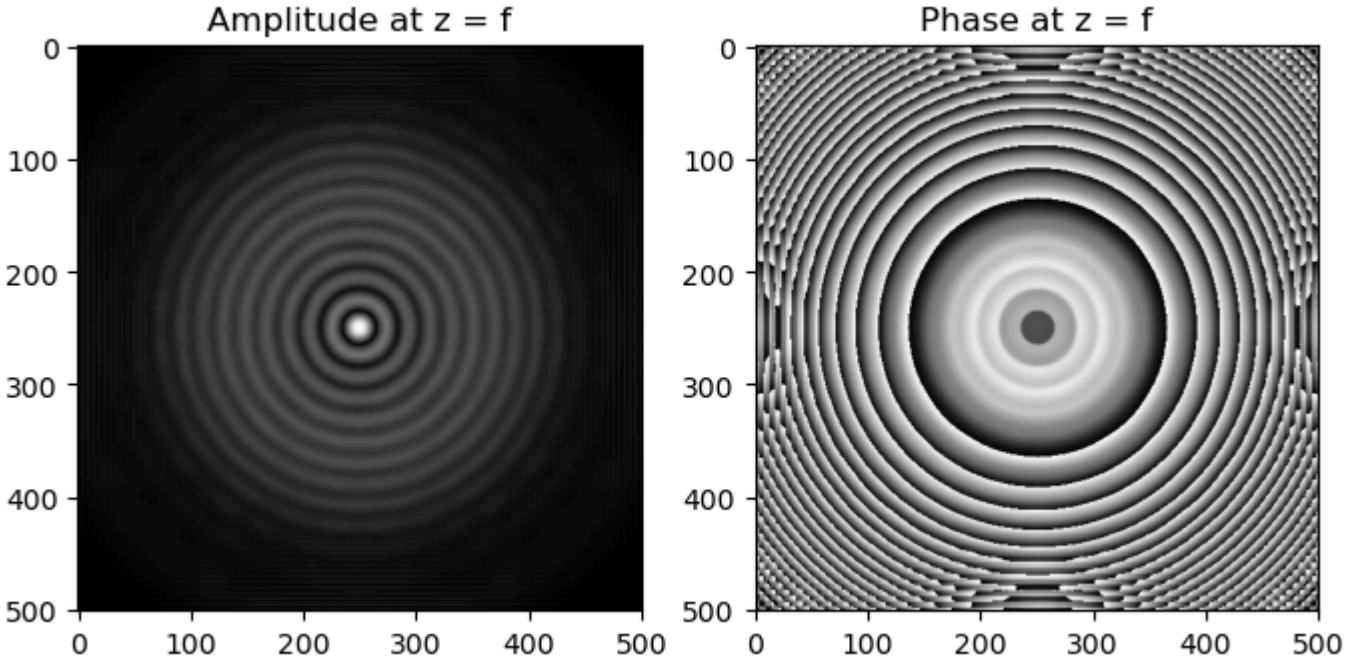
> **Exercise -** When we put an opaque disk in front of a gaussian beam it produces a bright spot at the center after propagation.

In [15]:
```
# Creating a mask
circ1 = np.zeros((N, N))
circ1[x**2 + y**2 >= (50*p)**2] = 1

plt.figure()
plt.imshow(circ1, cmap = "gray")
plt.show()

e_01 = e_0*circ1
plt.figure()
plt.subplot(1,2,1)
plt.imshow(np.abs(e_0))
plt.subplot(1,2,2)
plt.imshow(np.abs(e_01))
plt.show()
```


0
100
200
300
400
500
0
100
200
300
400
500



In [16]:
```
# Lets propagate this field through a lens
E_01 = ft2(e_01*lens)
E_01_prop = E_01*H3*filtr3
e_01_prop = ift2(E_01_prop)

E_02_prop = E_01*H2*filtr2
e_02_prop = ift2(E_02_prop)

plt.figure(figsize = (8,8))
plt.subplot(1,2,1)
plt.imshow(np.abs(e_01_prop), cmap = "gray")
plt.title("Amplitude at z = f")
plt.subplot(1,2,2)
plt.imshow(np.angle(e_01_prop), cmap = "gray")
plt.title("Phase at z = f")
plt.show()
```



We can see that the centre is a bright spot when we put an opaque disk in front of a gaussian beam.

6. Talbot effect -

The Talbot effect is a self-imaging phenomenon where the diffraction pattern from a periodic structure reforms a copy of the structure at downstream planes. the Talbot planes or the planes where the object is reproduced are located at

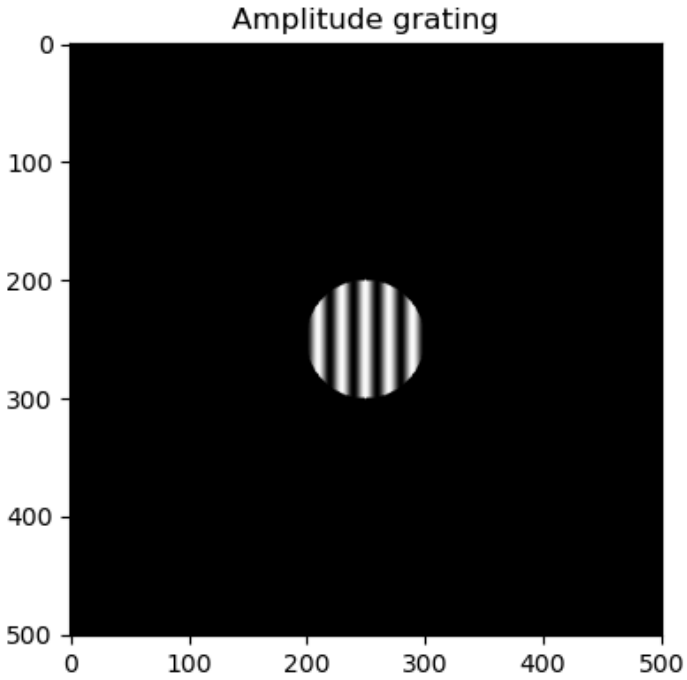
$$z_m = \frac{2mL^2}{\lambda}$$

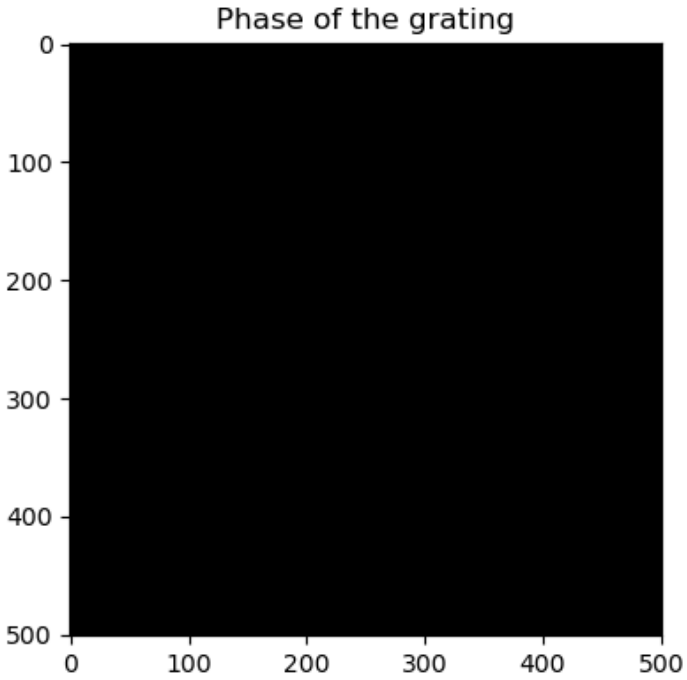
where m is an integer, and L is the grating period.

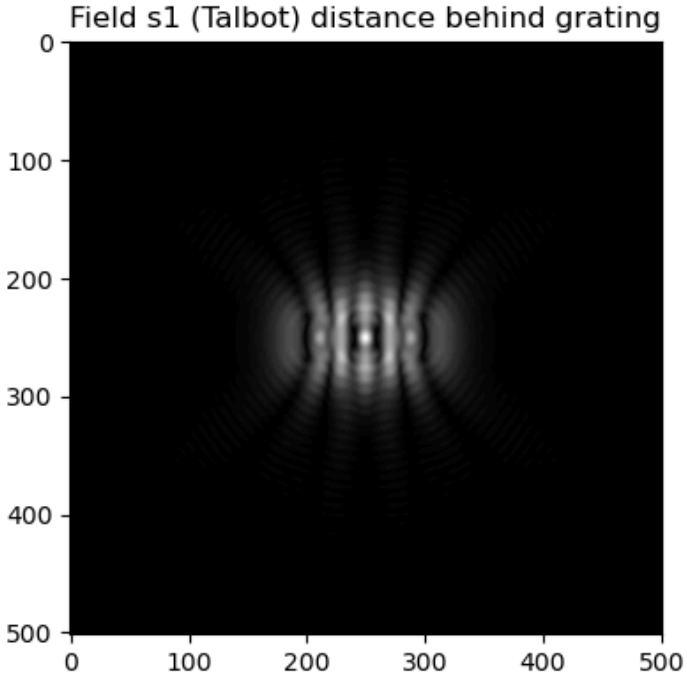
In [23]:
```
L = 1e-4 # Grating period
transmittance = np.cos(2*np.pi*x/L)
grat = 0.5*(1+transmittance) # Grating transmittance

s1 = 2*L**2/wavelength # First talbot distance
```

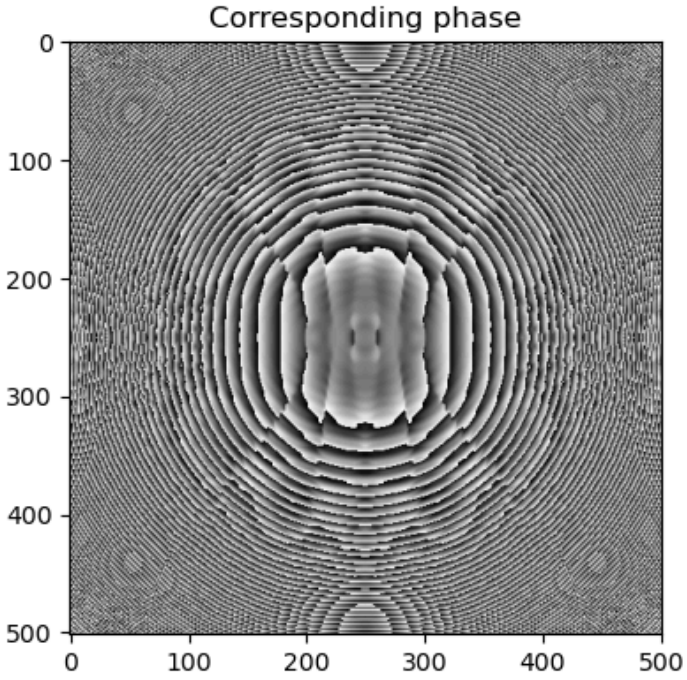
```
s2 = s1/2
s3 = s1 + s2

H_s1 = np.exp(1j*alpha*s1)
H_s2 = np.exp(1j*alpha*s2)
H_s3 = np.exp(1j*alpha*s3)

lp_s1 = (1/wavelength)*(1/(np.sqrt(1 + (2*s1/(N*p))**2)))
lp_s2 = (1/wavelength)*(1/(np.sqrt(1 + (2*s2/(N*p))**2)))
lp_s3 = (1/wavelength)*(1/(np.sqrt(1 + (2*s3/(N*p))**2)))

filtr_s1 = np.zeros((N, N))
filtr_s1[fx**2 + fy**2 <= lp_s1**2] = 1
filtr_s2 = np.zeros((N, N))
filtr_s2[fx**2 + fy**2 <= lp_s2**2] = 1
filtr_s3 = np.zeros((N, N))
filtr_s3[fx**2 + fy**2 <= lp_s3**2] = 1

grating = circ*grat

Grating = ft2(grating)

Grating_prop_s1 = Grating*H_s1*filtr_s1
grating_prop_s1 = ift2(Grating_prop_s1)

Grating_prop_s2 = Grating*H_s2*filtr_s2
grating_prop_s2 = ift2(Grating_prop_s2)

Grating_prop_s3 = Grating*H_s3*filtr_s3
grating_prop_s3 = ift2(Grating_prop_s3)

plt.figure(figsize = (15,20))
plt.subplot(4,2,1)
plt.imshow(np.abs(grating), cmap = "gray")
plt.title("Amplitude grating")
plt.subplot(4,2,2)
plt.imshow(np.angle(grating), cmap = "gray")
plt.title("Phase of the grating")
plt.subplot(4,2,3)
plt.imshow(np.abs(grating_prop_s1), cmap = "gray")
plt.title(f"Field s1 (Talbot) distance behind grating")
```

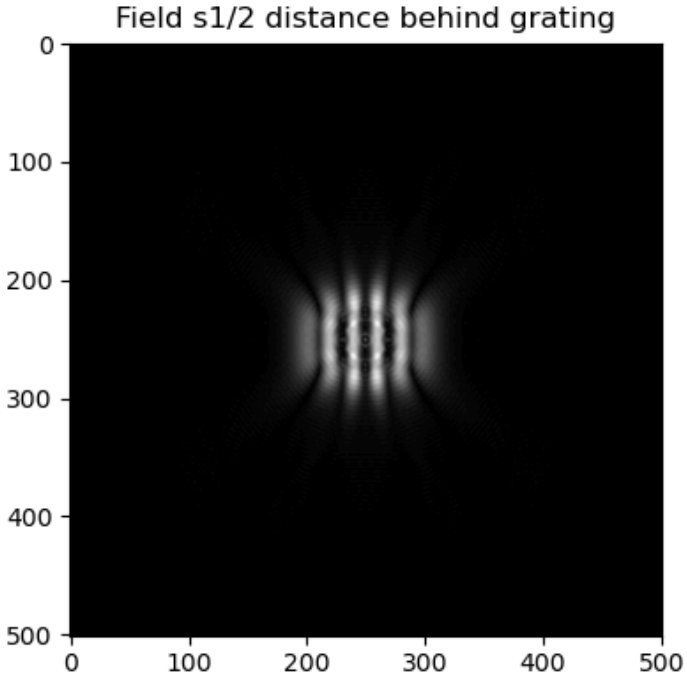
```
plt.subplot(4,2,4)
plt.imshow(np.angle(grating_prop_s1), cmap = "gray")
plt.title("Corresponding phase")
plt.subplot(4,2,5)
plt.imshow(np.abs(grating_prop_s2), cmap = "gray")
plt.title(f"Field s1/2 distance behind grating")
plt.subplot(4,2,6)
plt.imshow(np.angle(grating_prop_s2), cmap = "gray")
plt.title("Corresponding phase")
plt.subplot(4,2,7)
plt.imshow(np.abs(grating_prop_s3), cmap = "gray")
plt.title(f"Field 3*s1/2 distance behind grating")
plt.subplot(4,2,8)
plt.imshow(np.angle(grating_prop_s3), cmap = "gray")
plt.title("Corresponding phase")
plt.show()
```

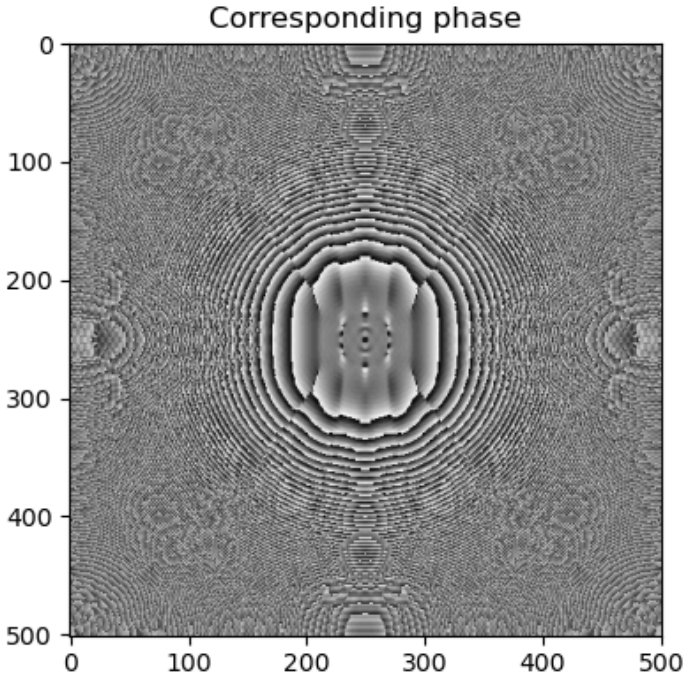

Amplitude grating
0
100
200
300
400
500
0
100
200
300
400
500

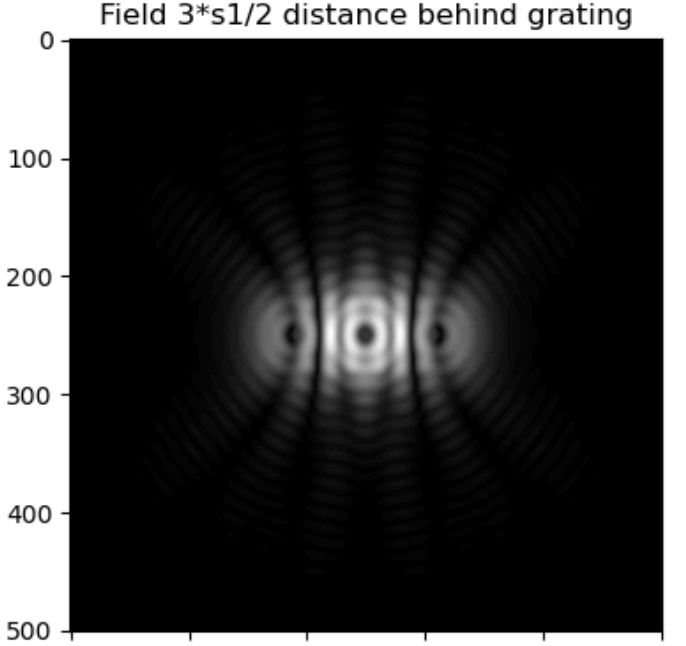

Phase of the grating
0
100
200
300
400
500
0
100
200
300
400
500

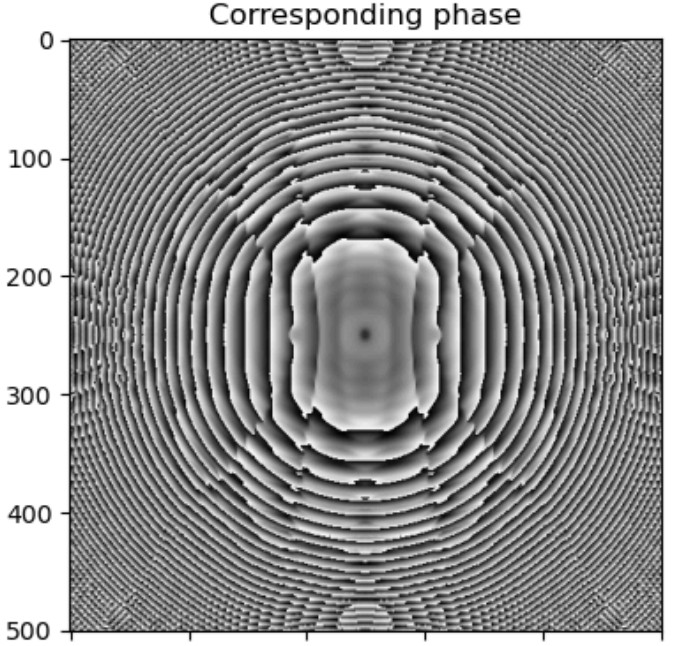

Field s1 (Talbot) distance behind grating
0
100
200
300
400
500
0
100
200
300
400
500


Corresponding phase
0
100
200
300
400
500
0
100
200
300
400
500


Field s1/2 distance behind grating
0
100
200
300
400
500
0
100
200
300
400
500


Corresponding phase
0
100
200
300
400
500
0
100
200
300
400
500


Field 3*s1/2 distance behind grating
0
100
200
300
400
500


Corresponding phase
0
100
200
300
400
500

6. Bessel Beam

2d ft of delta function $\delta(\rho - \rho_0) = 2\pi\rho_0 J_0(2\pi r\rho_0) e^{(\iota * z * \sqrt{(k^2 - 4 * \pi^2 * \rho^2)})}$

Bessel Beam doesnt change its ring shape.

Axicon -> Conical lens

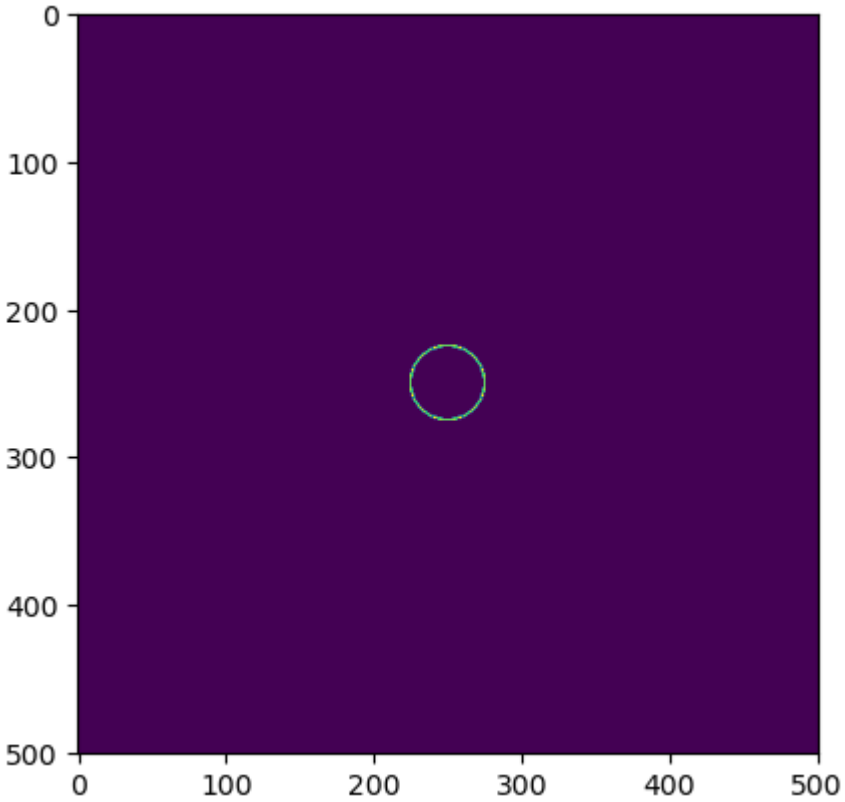
- Self Learn

  Other beam of this type is an Airy beam - cubic phase - Better depth of focus- Self learn Radially polarized Bessel Beam - how to propagate? The focal point of this beam is better than the diffraction limit.

  Reflection of polarized beam, Gaussian polarized beam

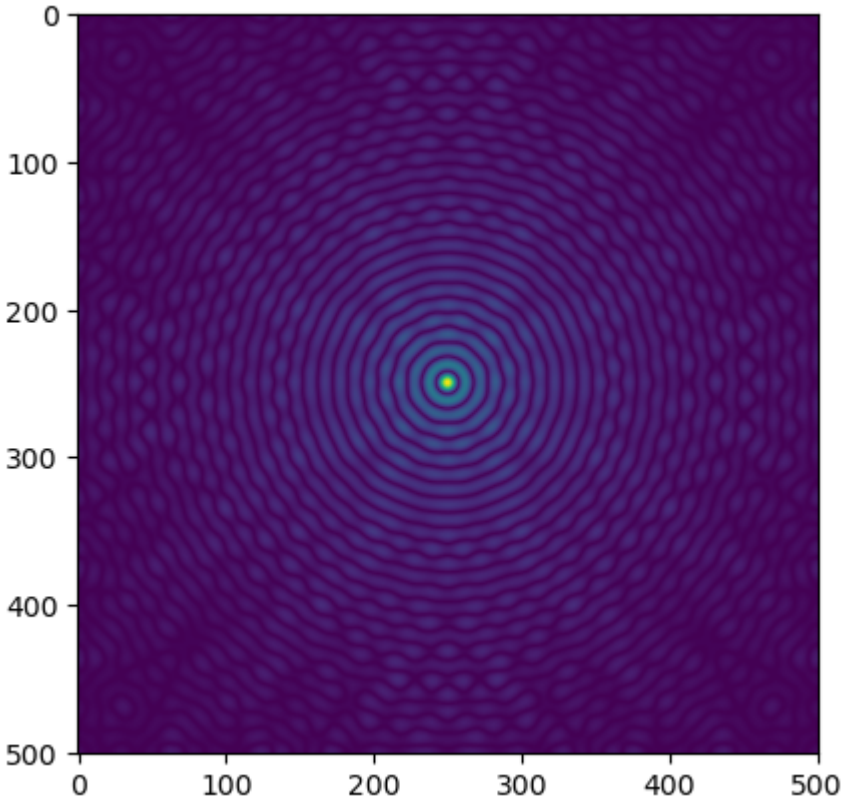
In [18]:
```
rho = np.sqrt(fx**2 + fy**2)
rho_0 = 1e4

width = 600

annular_ring = np.zeros_like(rho)
annular_ring[(rho >= rho_0 - width/2) & (rho <= rho_0 + width/2)] = 1

plt.imshow(annular_ring)
```

Out[18]:
```
<matplotlib.image.AxesImage at 0x1f3d1bd1590>
```



In [19]:
```
bessel = ift2(annular_ring)
plt.imshow(np.abs(bessel))
```

Out[19]: <matplotlib.image.AxesImage at 0x1f3d3898e90>



7. Conclusion

In this report we saw how the extent and pixel size (spacing) varies between the normal and fourier domain. We also find the condition in the fourier domain for proper sampling of the frequencies such that the Nyquist condition is satisfied. In section 5 we see the propagation of vortex beam and propagation of gaussian beam through an obstacle which shows the poisson arago spot. We also visualized the talbot effect which shows the multiplicity of perfect intensity images of an object appearing behind a grating at fixed distances without the help of lenses. We also see that the bessel beam is a fourier transform of a radial delta function, this beam holds a special property that the field remains the smae after propagation.

In [ ]: