Wave Optics based Numerical Simulation of Wave Propagation

Shamminuj Aktar

Overview

- ❑ Wave Optics
- ❑ Wave Optics Simulation
- ❑ Application of Wave Optics Simulation
- ❑ Problem Statement
- ❑ Transmittance Function of a Lens
- ❑ Fresnel Transfer Function (TF) Approach
- ❑ Python Code & Plots for Focusing with Lens
- ❑ Comparison between Theoretical & Simulation Result
- ❑ Conclusion

Wave Optics

- Area of optics in which the wave nature of light is essential when defining its propagation
- Deals with the study of various phenomenal behaviors of light like reflection, refraction, interference, diffraction, polarization etc
- Also known as physical optics

Wave Optics Simulation & Application

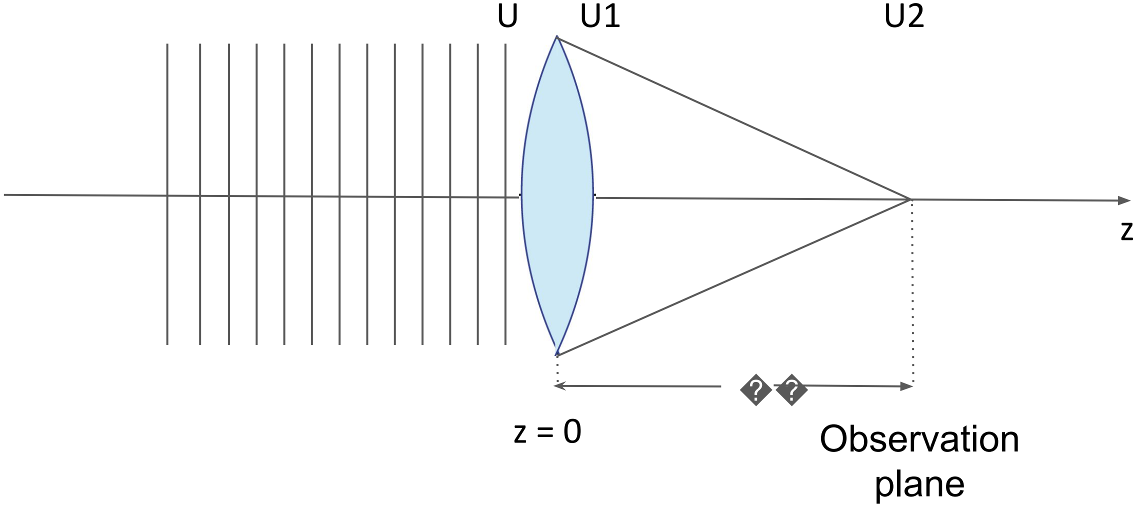
Numerical propagation is combined with the concept of the transmittance function for optical elements to model an application.

<u>Applications of Wave Optics Simulation</u>:

- ❑ Atmospheric imaging
- ❑ Astronomy
- ❑ Adaptive optics
- ❑ Free-space optical communications
- ❑ LADAR (LAser Detection And Ranging)

Problem Statement

- A plane wave is transmitted through a lens of focal length f
- The lens should converge the wave to a point at a distance f from the lens
- We will observe the resulting point image at the observation plane.



$$U = A\, e^{-jkz} = A$$

$$U_1 = t * U$$

Fig: Focusing of a plane wave by a lens

Transmittance Function

The transmittance function for an ideal, simple lens is given by

$$t_A(x, y) = P(x, y)\exp\left[-j\frac{k}{2f}\left(x^2 + y^2\right)\right]$$

where f is the focal length of lens, $P(x,y)$ is the pupil function

$$P(x, y) = \mathrm{circ}\left(\frac{\sqrt{x^2 + y^2}}{w_L}\right)$$

DG Voelz. Computational fourier optics: a matlab tutorial 2011

Fresnel Transfer Function (TF) Approach

Fresnel Transfer Function (TF) Approach is applied to propagate the field *U1* at a distance f from the lens.

Propagation of light between two planes is regarded as a linear system with impulse response h. Transfer function H is defined as,

$$H(f_X, f_Y) = e^{jkz} \exp\left[-j\pi\lambda z\left(f_X^2 + f_Y^2\right)\right]$$

At observation plane U2 is given by

$$U_2(x, y) = \Im^{-1}\left\{\Im\left\{U_1(x, y)\right\}H(f_X, f_Y)\right\}$$

DG Voelz. Computational fourier optics: a matlab tutorial 2011

Critical Sampling

To minimize artifacts in the simulation, we consider critical sampling.

The critical sampling expression is defined as

$$\Delta x = \lambda z / L$$

where Δx is sample interval, λ is the wavelength, z is the propagation distance, L is the side length.

We can rearrange the critical sampling expression as

$$L = \sqrt{M \lambda z}$$ where M = number of samples

DG Voelz. Computational fourier optics: a matlab tutorial 2011

Theoretical Result

Focusing with lens creates **Fraunhofer intensity pattern**.

The intensity is given by,

$$I_2(x, y) = \left(\frac{w^2}{\lambda z}\right)^2 \left[\frac{\mathrm{J}_1\left(2\pi \frac{w}{\lambda z}\sqrt{x^2 + y^2}\right)}{\frac{w}{\lambda z}\sqrt{x^2 + y^2}}\right]^2$$

where J1 is the Bessel function of the first kind, order 1.

DG Voelz. Computational fourier optics: a matlab tutorial 2011

Python code for Focusing with Lens

```
N=500                             #samples 1D
lambda_= 0.5e-6                   #wavelength
k= (2*pi)/lambda_                 #wavenumber
f=1                               #Focal length of lens
L=sqrt(N*lambda_*f)               #spatial grid side length

dx=L/N                            #sample interval
x = np.linspace(-L/2,L/2-dx,N)  #linear coordinates
y = x
X, Y= np.meshgrid(x,x)             #2D cordinates

z=0
A=np.ones(N)
U=A*np.exp(-1j*k*z);               #plane wave field
```

```
w=L/10                       #lens radius
P=circ(np.sqrt(np.square(X)+np.square(Y))/w) #pupil function

t_lens=np.multiply(P,np.exp(-1j*k/(2*f)*np.add(np.square(X),np.square(Y)))) #transmittance function
U1=np.multiply(U,t_lens)      #Field after lens
```

Python code for Focusing with Lens

```
U2=propTF(U1,L,lambda_,f)        # Field at a distance f from lens
I_b=np.square(abs(U2))           # Intensity at a distance f from lens
```

```
fig, axes = plt.subplots(nrows=1, ncols=2,figsize=(22, 6))
img1 = axes[0].imshow(I_b, extent = [min(x),max(x),min(y),max(y)],origin='lower') #display intensity
axes[0].set_xlabel("x (m)",fontsize=16)
axes[0].set_ylabel("y (m)",fontsize=16)

slice_ = int(N/2)
y_ = I_b[slice_,:]
axes[1].plot(x,y_)
axes[1].set_xlabel("x (m)",fontsize=16)
axes[1].set_ylabel("y (m)",fontsize=16)

plt.show()
```

Python code for Fresnel Transfer Function (TF) Approach

```
def propTF(u1,L,lambda_,z):
    # propagation - transfer function approach
    # assumes same x and y side lengths and
    # uniform sampling
    # u1 - source plane field
    # L - source and observation plane side length
    # lambda - wavelength
    # z - propagation distance
    # u2 - observation plane field
    M,N =u1.shape                              #get input field array size
    dx=L/M                                     #sample interval
    k=(2*pi)/lambda_                           #wavenumber
    fx=np.arange(-1/(2*dx),-1/(2*dx)+(M)*1/L,1/L) #freq coords
    FX,FY=np.meshgrid(fx,fx)
    H=np.exp(-1j*pi*lambda_*z*np.add(np.square(FX),np.square(FY)))  #trans func
    H=np.fft.fftshift(H)                        #shift trans func
    U1=np.fft.fft2(np.fft.fftshift(u1))         #shift, fft src field
    U2=np.multiply(H,U1)                        #multiply
    u2=np.fft.ifftshift(np.fft.ifft2(U2))       #inv fft, center obs field
    return u2
```

Python code for pupil & jinc function

```
def circ(r):
    out=(np.where(abs(r)<=1, 1, 0))
    return out
```

```
def jinc(x):
    #jinc function
    #evaluates J1(2*pi*x)/x with divide by zero fix
    #locate non-zero elements of x
    index = np.nonzero(x == 0)
    x1=(np.where(x==0, -1, x))
    #compute output values for all other x
    out = np.divide(special.jv(1, 2*pi*x1),x1)
    out[index] = pi
    return out
```

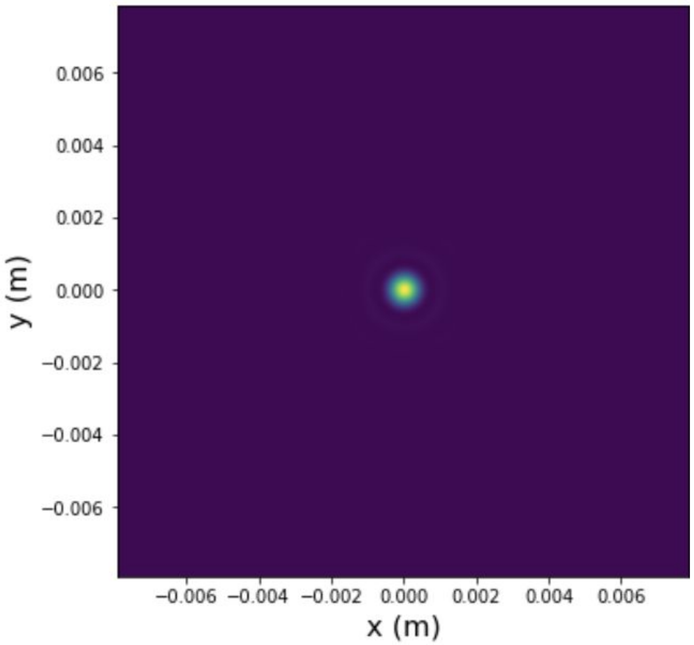
Python code for Theoretical Result

```
a1 = np.square(w)/(lambda_*f)
a2_ = (w/(lambda_*f))*np.sqrt(np.add(np.square(X),np.square(Y)))
a2 = jinc(a2_)
I2_th = np.square(a1)*np.square(a2)
```

```
fig, axes = plt.subplots(nrows=1, ncols=2,figsize=(22, 6))

img1 = axes[0].imshow(I2_th, extent = [min(x),max(x),min(y),max(y)],origin='lower') #display intensity
axes[0].set_xlabel("x (m)",fontsize=16)
axes[0].set_ylabel("y (m)",fontsize=16)

slice_ = int(N/2)
y1_ = I_b[slice_,:]
y_ = I2_th[slice_,:]
axes[1].plot(x,y1_)
axes[1].plot(x,y_,'.')
axes[1].set_xlabel("x (m)",fontsize=16)
axes[1].set_ylabel("y (m)",fontsize=16)
axes[1].legend(['Simulation', 'Theoretical'],fontsize = 12)

plt.show()
```

Result: Intensity Pattern for w = L/40

- Point image at the observation plane



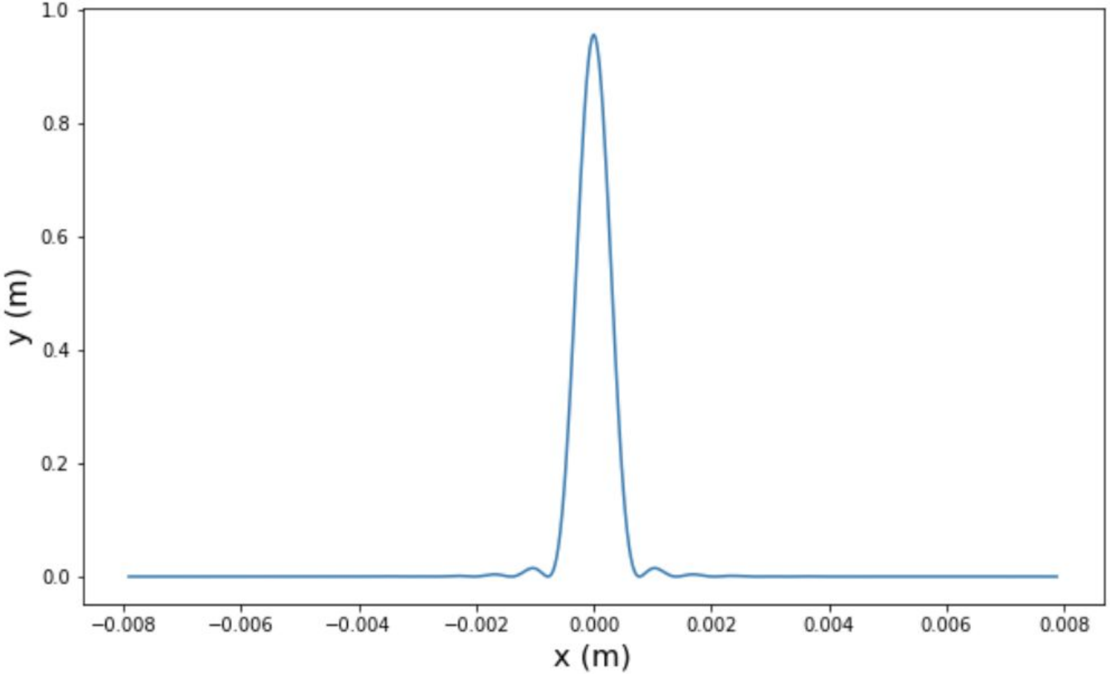
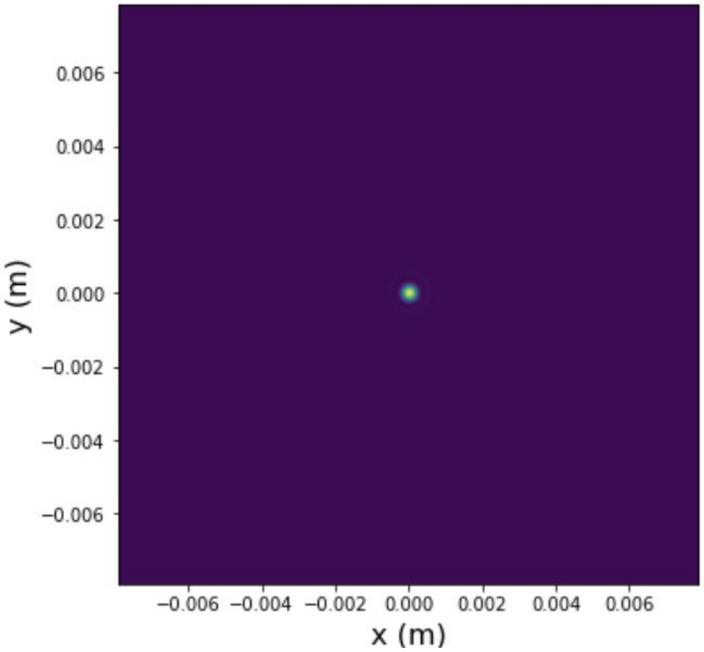


Fig: (a) Intensity (b) Slice of Intensity at a distance f from lens for w = L/40

Result: Intensity Pattern for w = L/20

- With larger w, the focus point gets smaller



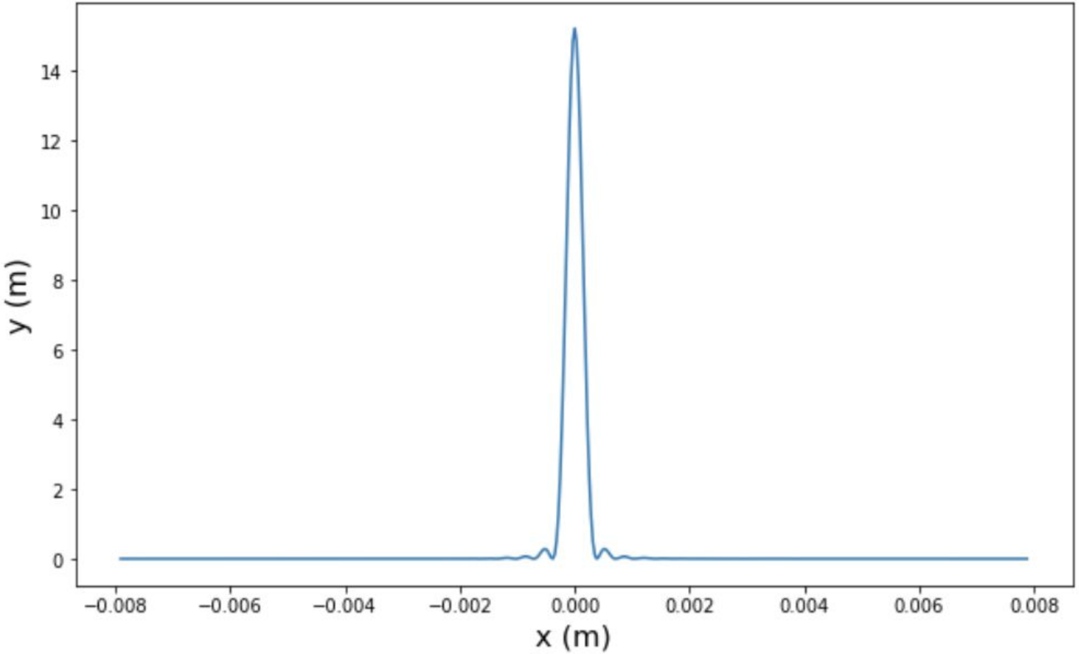
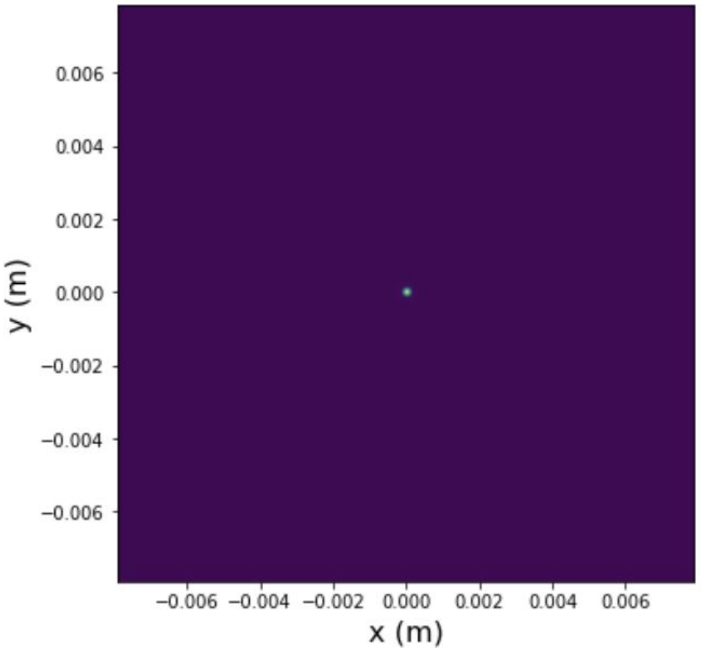


Fig: (a) Intensity (b) Slice of Intensity at a distance f from lens for w = L/20

Result: Intensity Pattern for w = L/10

- With larger w, the focus point gets smaller



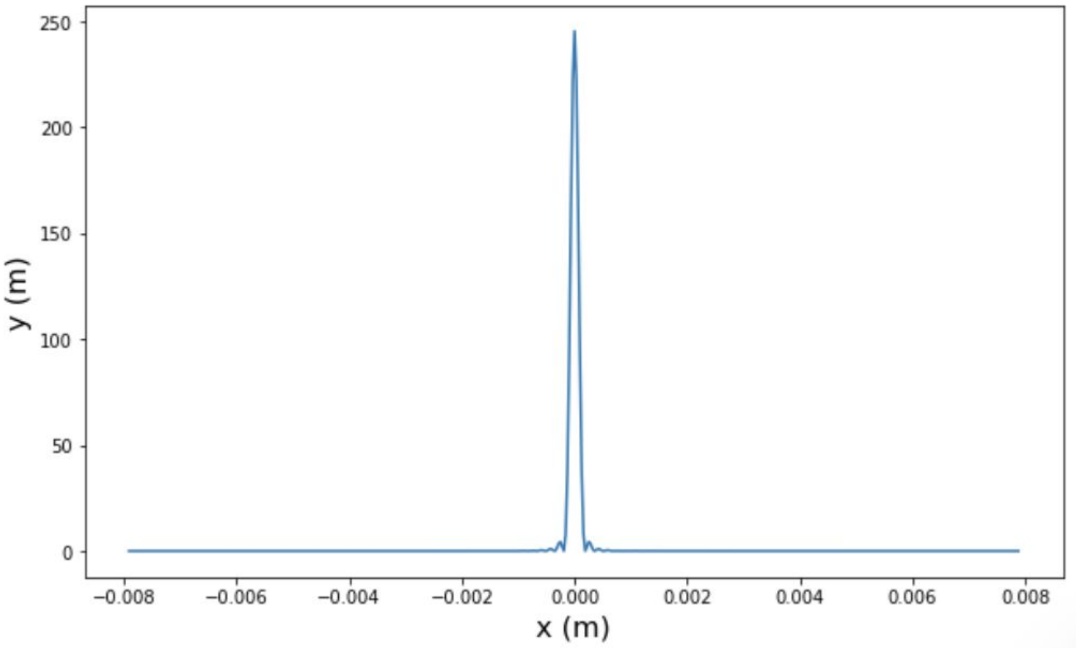


Fig: (a) Intensity (b) Slice of Intensity at a distance f from lens for w = L/10

Comparison between Theoretical & Simulation Result

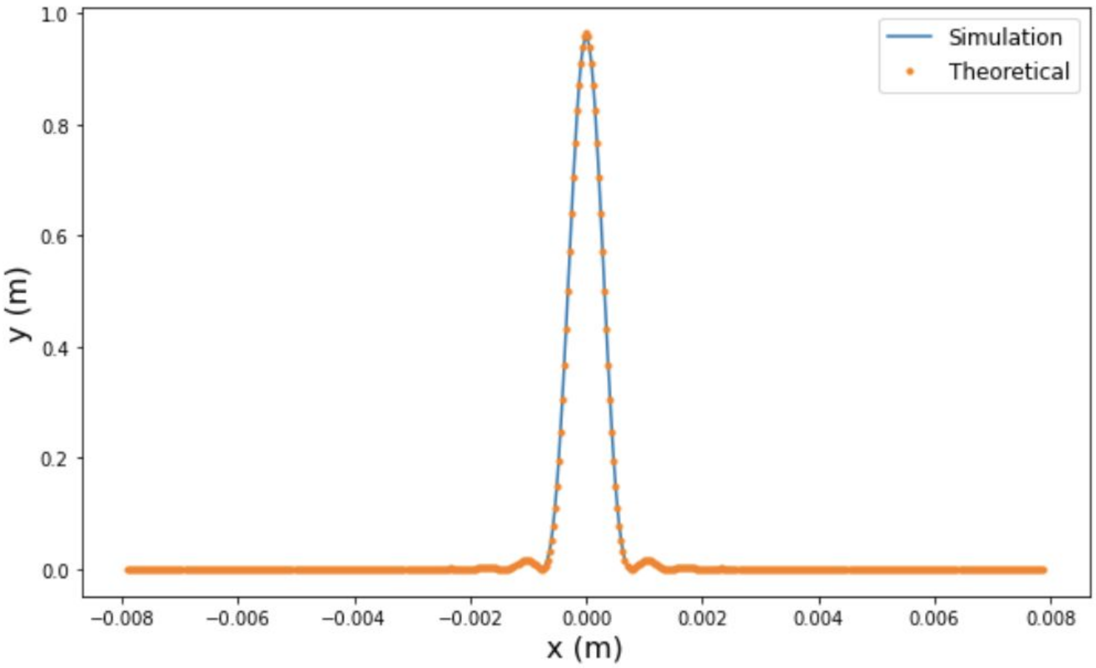


Fig: Comparison of intensity slices of Theoretical & Simulation Result

Conclusion

- We study wave optics simulation & apply it for focusing with a lens.
- We investigate Fresnel Transfer Function (TF) Approach for propagating a field.
- From critical sampling expression we find that side length $L = \sqrt{M \lambda z}$
- With the increment of lens radius, the focus point on the observation plane gets smaller
- Comparison between theoretical & simulation result shows similar intensity pattern