Python Project 3: Fourier Propagation

David Schmidt

April 23, 2018

1 Background

The project focuses on using Fourier analysis to propagate laser beams through space and lenses. The general theory is that we can take an input laser electric field and transform to spatial frequency space. Once there we can multiply by a propagation factor and then transform pack. It is a pretty simple concept but can take some legwork to get working code.

2 Imports

```python
# -*- coding: utf-8 -*-
"""
Created on Wed Apr 11 08:28:10 2018
@author: ddsch (Python 3.6)
David Schmidt

Description:
"""
import numpy as np
import matplotlib.pyplot as plt
import numpy.fft as nf
import time

```

Typical for python code.

3 Useful Functions

```python
#%%%%%%%%%%%%%%% Useful Functions

def circ_aperture(x,y,A_dia,In):
    boolarray=(x**2+y**2)<A_dia**2
    return boolarray*In

def wGaus(z,zR,w0):
    return(w0*np.sqrt(1+z/zR)**2)

def plot2DInt(axin,Iin,Lin,title):
    axin.set_title(title)
    Inorm=Iin/Iin.max()
    plt.imshow(Inorm,origin='lower', interpolation='none', extent=[-Lin/2,Lin/2,-Lin/2,Lin/2],cmap='gray')
    axin.set_aspect('equal')

def plot1DInt(axin,xin,Iin,Min,title):
    axin.set_title(title)
    plt.plot(xin,Iin[int(Min/2),:]/max(Iin[int(Min/2),:]))
    axin.grid(color='black', linestyle='-', linewidth=.1)


def ScanTF2D(Ein,L,lam,zf,dz,M):
    """
    Uses TF propagation to get a scan from z=0 to z=zf in steps of dz and returns the typical 2D plot of x and z
    """
    z=np.arange(0,zf+dz,dz)
    array=np.zeros((len(z),M))
    for ii in range(len(z)):
        ufull=propTF(Ein,L,lam,z[ii])
        Ifull=abs(ufull)**2
        Icut=Ifull[int(M/2),:]
        array[ii]=Icut
    f=plt.figure()
    axf = f.add_subplot(111)
    axf.set_title("Cross-section propagation along z. (m=2 with clipping)")
    Inorm=array/array.max()
    plt.imshow(Inorm,origin='lower')# interpolation='none', extent=[-Lin/2,Lin/2,-Lin/2,Lin/2],cmap='gray')
    axf.set_aspect('equal')


```

I make functions that I can call to do things that we do a lot in Fourier propagation. A circular aperture is used a lot to test the propagation code. It has a well known analytical result with Airy disks. The function simple returns an array with 0's if outside a radius or the input value. The plot functions are pretty self explanatory as they take in variables that allow it to plot a normalized line out or 2D density plot. The ScanTF2D is a pretty beefy function. It takes and propagates an initial beam through a z of 0 to zf. Then it takes a line-out and makes a 2D density plot over z. This might not make sense now but images later will hopefully help.

4 Propagation Functions

```
#%%%%%%%%%%%%%%%% Propagation Functions

def propTF(Ein,L,lam,z):
    """
    propagation - transfer function approach
    assumes same x and y side lengths and uniform sampling
    Ein - source plane field
    L - source and observation plane side length
    lam - wavelength
    z - propagation distance
    u2 - observation plane
    """
    (M,N)=np.shape(Ein)
    dx=L/M

    fx=np.arange(-1/(2*dx),1/(2*dx),1/L)
    FX, FY = np.meshgrid(fx, fx, sparse=True)

    H=np.exp(-1j*np.pi*lam*z*(FX**2+FY**2))
    H=nf.fftshift(H)
    U1=nf.fft2(nf.fftshift(Ein))
    U2=H*U1
    Eout=nf.ifftshift(nf.ifft2(U2))
    return(Eout)


def propFF(Ein,L,lam,z):
    """
    propagation - transfer function approach
    assumes same x and y side lengths and uniform sampling
    Ein - source plane field
    L - source plane side length
    lam - wavelength
    z - propagation distance
    Eout - observation plane
    L2 - observation plane side length
    """
    (M,N)=np.shape(Ein)
    dx=L/M
    k=2*np.pi/lam

    L2=lam*z/dx
    dx2=lam*z/L

    x2=np.arange(-L2/2,L2/2,dx2)
    X2, Y2 = np.meshgrid(x2, x2, sparse=True)

    c=1/(1j*lam*z)*np.exp(1j*k/(2*z)*(X2**2+Y2**2))
    Eout=c*nf.ifftshift(nf.fft2(nf.fftshift(Ein)))*dx**2
    return([Eout,L2,x2])

```

These are the main functions. One that propagates the input beam through a transfer function method. This is the method where we multiply our field in frequency space to propagate a set z distance. The other propagator does the Fraunhofer propagation which is what you use to find what your laser beam looks like at the focus of a lens.

5 Parameters

```
#%%%%%%%%%%%%%%%% Parameters

plt.close('all')
L1=60e-3
M=2**10
dx1=L1/M
x1=np.arange(-L1/2,L1/2,dx1)
y1=x1
lam=0.8e-6
k=2*np.pi/lam
w0=4e-3
zR=np.pi*w0**2/lam
E0=1
phase=0
m=12
print("Rayleigh range: {} mm".format(np.round(zR*10**3,2)));
```

Setup of initial beam parameters. We also calculate the Rayleigh range as it is a useful measurement to know the scale over which your beam changes in z. A large Rayleigh range means your beam size shouldn't deviate much over that range. Beam structure might change, but size is essentially the same.

6 Propagation

```
#%%%%%%%%%%%%%%% Propagation
z0=0
z=10
zf=1

[X1,Y1]=np.meshgrid(x1,y1,sparse=True);
u1=E0*(w0/wGaus(z0,zR,w0))*np.exp(-((X1**2+Y1**2)**m)/wGaus(z0,zR,w0)**(2*m));
u1=circ_aperture(X1,Y1,10,u1)
I1=abs(u1)**2;

x2=x1
dx2=dx1
y2=y1
u2=propTF(u1,L1,lam,z)
I2=abs(u2)**2

[u3,L3,x3]=propFF(u2,L1,lam,zf)
I3=abs(u3)**2
```

I actually propagate the beam forward and then focus it.

7 Plotting

```
#%%%%%%%%%%%%%%% Plotting
fig = plt.figure(figsize=(16*2, 9*2))

ax1 = fig.add_subplot(231)
plot2DInt(ax1,I1,L1,'Input Beam')

ax2 = fig.add_subplot(234)
plot1DInt(ax2,x1,I1,M,'Center Cross Section')

ax3 = fig.add_subplot(232)
plot2DInt(ax3,I2,L1,'Beam after z=10m')

ax4 = fig.add_subplot(235)
plot1DInt(ax4,x2,I2,M,'Center Cross Section')

ax5 = fig.add_subplot(233)
plot2DInt(ax5,I3,L3,'Focal Plane')

ax6 = fig.add_subplot(236)
plot1DInt(ax6,x3,I3,M,'Center Cross Section')

#zf=100
#dz=.1
#t1=time.time()
#ScanTF2D(u1,L1,lam,zf,dz,M)
#t2=time.time()
#print(t2-t1)

plt.show()
```

I use my functions to make a nice subplot array to show the beam propagation. In comments is also the code to generate the zscan plot

8 Example Outputs

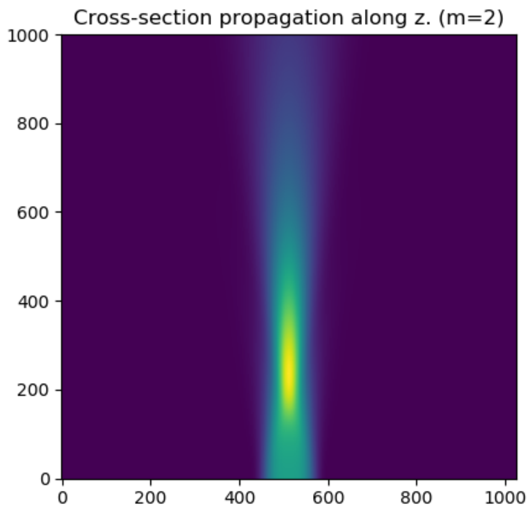

Figure 1: Gaussian Input zscan

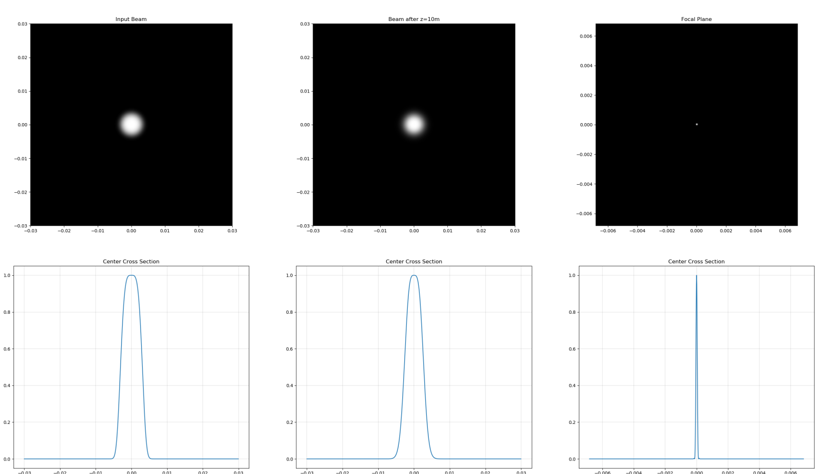

Figure 2: Gaussian input propagation then focus

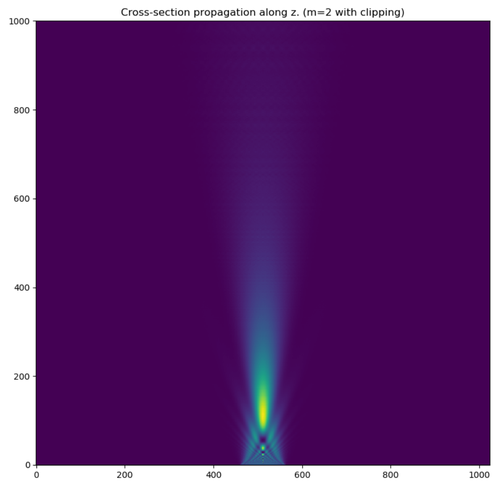

Figure 3: Clipped Gaussian Input zscan

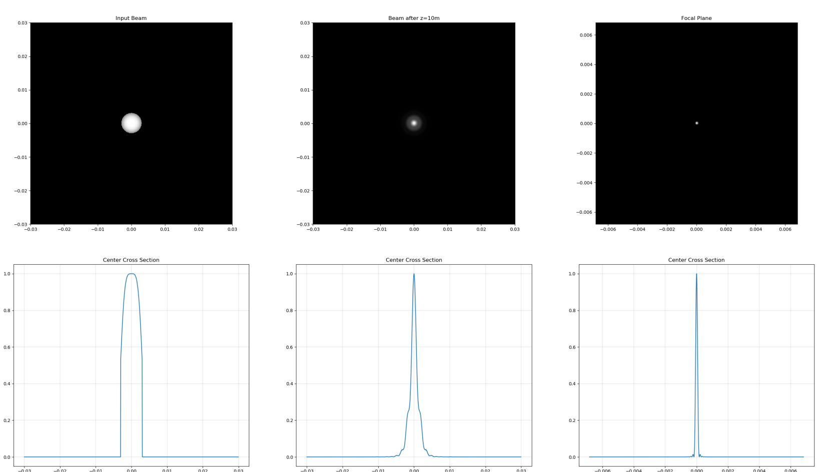

Figure 4: Clipped Gaussian input propagation then focus

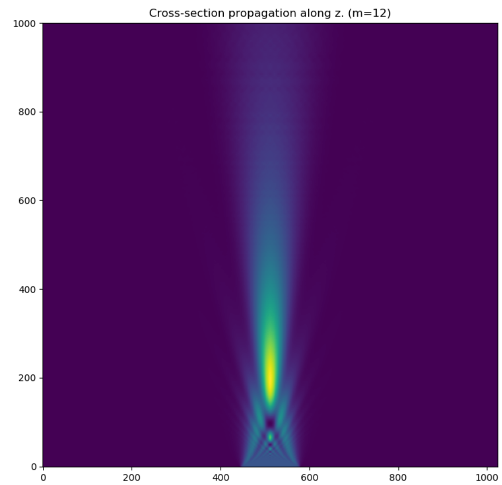

Figure 5: m=12 super-Gaussian Input zscan

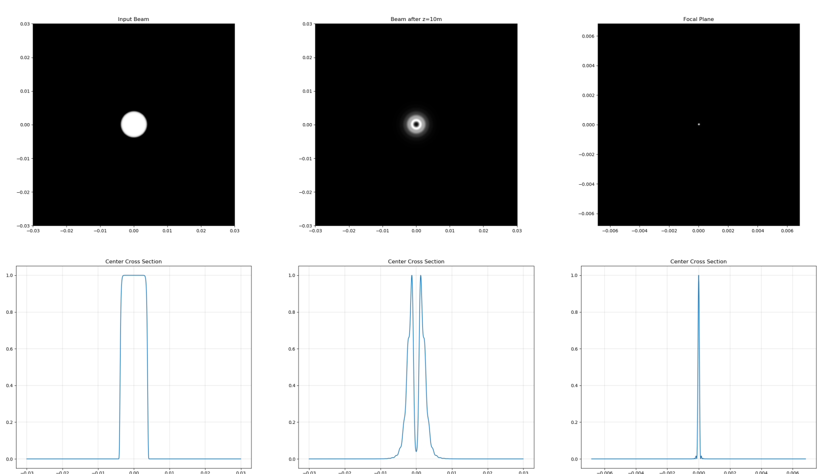

Figure 6: m=12 super-Gaussian Input propagation then focus