[6]:
```python
# Simpson's Rule integration function
# this function uses the formula for Simpson's rule that was
# introduced in the last problem set

def simps_rule(x,y):
    """Approximate the integral of y(x) using Simpson's rule.
    Note it assumes that the x samples are evenly spaced."""

    # if len(x) is even, then there are an odd number of slices
    # Simpson's rule needs an even number of slices
    if len(x) % 2 == 0: # if len(x) is even, complain
        raise Exception("Simpson's Rule requires an even number of slices. "+
                        "This means len(x) must be odd.")
    else:
        Y = np.copy(y) # make a copy of y so that input y isn't changed
        Y[1:-1:2] *= 4 # multiply odd entries by 4, except first/last
        Y[2:-1:2] *= 2 # multiply even entries by 2, except first/last
        h = (x[-1] - x[0])/(len(x)-1) # x spacing
        return 1/3*h*sum(Y)
```

[7]:
```python
# Bessel Function
```

[7]:
```
# Bessel Function
def bessel(m,x):
    """Calculate J_m(x) where J_m is the mth Bessel function."""

    try: # if x is a scalar
        theta = np.linspace(0,np.pi,1001) # domain
        y = np.cos(m*theta-x*np.sin(theta)) # integrand
        return 1/np.pi*simps_rule(theta,y) # calculate

    except: # if x is an array
        result = [] # array of results
        for i in x:
            theta = np.linspace(0,np.pi,1001) # domain
            y = np.cos(m*theta-i*np.sin(theta)) # integrand
            result.append(1/np.pi*simps_rule(theta,y)) # append calculation
        return result
```

[8]:
```
# Plot of the first three Bessel functions
fig2,ax2 = plt.subplots(1,1)
```

Now I will plot the diffraction pattern

[9]:
```
# set parameters
wavelength = 500e-9 # meters
radius = 1e-6 # plot radius, meters
nsamples = 400

k = 2*np.pi/wavelength # wave number

# grid of intensities
grid = np.empty([nsamples,nsamples],float)

# iterate across plane
for i,x in enumerate(np.linspace(-1*radius,radius,nsamples)):
    for j,y in enumerate(np.linspace(-1*radius,radius,nsamples)):
        r = np.sqrt(x**2 + y**2) # calculate distance
        grid[i,j] = (bessel(1,k*r)/(k*r))**2 # calc. intensity
```

10]:
```
# Plot of diffraction pattern
fig3,ax3 = plt.subplots(1,1)

# plot
```

```
    for j,y in enumerate(np.linspace(-1*radius,radius,nsamples)):
        r = np.sqrt(x**2 + y**2) # calculate distance
        grid[i,j] = (bessel(1,k*r)/(k*r))**2 # calc. intensity
```

10]:

```
# Plot of diffraction pattern
fig3,ax3 = plt.subplots(1,1)

# plot
ax3.imshow(grid,vmax=0.01,origin='lower',extent=[-1,1,-1,1],cmap='hot')

# aesthetics
ax3.set_title("Diffraction pattern on focal plane")
ax3.set_xlabel("x ($\mu m$)")
ax3.set_ylabel("y ($\mu m$)")
ax3.set_xticks([-1,-0.5,0,0.5,1])
ax3.set_yticks([-1,-0.5,0,0.5,1])
ax3.text(-0.9,0.85,'$\lambda$ = 500nm',color='w',fontsize=10)

plt.show()
```