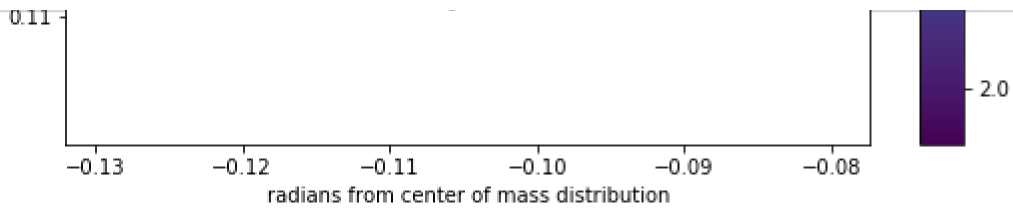
In [2]:
```
import numpy as np
from matplotlib import pyplot as plt
from matplotlib import cm as cmap
from mpl_toolkits.mplot3d import Axes3D


#working in weak lensing domain, 3d, with mass distributed across distance

class location:
    def __init__(self,x=0,y=0,z=0):
        self.x = x
        self.y = y
        self.z = z

    def dl2(self,dx,dy,dz):
        self.x += dx
        self.y += dy
        self.z += dz
        return dx**2+dy**2+dz**2

class mass:
    def __init__(self,mass,x,y,z):
        self.mass = mass
        self.loc = location(x,y,z)


def bend(dist, mass):
    angle = mass/dist #units chosen such that 4G/c^2 = 1
    if(angle>0.2): #so weak lensing dist >> 2GM/c^2 = mass/2
        print(str(mass)+" mass is within strong lensing domain")
    return angle

#redshift is distance light has traversed (pathlength)
#massdistr sorted by proximity to observer (z)
#make sure massdistr doesn't extend past z of source
def beam(plot, magnifications, redshift, massdistr, xangle, yangle):
    magnification =1
    loc = location()
    totaldz = redshift/np.sqrt(1 + np.sin(xangle)**2 + np.sin(yangle)**2)
    if(plot):
        apparentloc = location(totaldz*np.sin(xangle), totaldz*np.sin(yangle), totaldz)
        fig = plt.figure(1, figsize=(8,8))
        az = fig.add_subplot(111, projection='3d')
        az.plot([loc.x], [loc.y], 'ro',zs=loc.z, label="mass distribution")
        az.plot([loc.x],[loc.y], 'o',zs=loc.z,label="observer")
        az.plot([apparentloc.x],[apparentloc.y],'o',zs=apparentloc.z,label="apparent source")
        az.plot([loc.x,apparentloc.x],[loc.y,apparentloc.y],'orange',zs=[loc.z,apparentloc.z],label="apparent pat
        az.set_xlabel("x distance from observer")
        az.set_ylabel("y distance from observer")
        az.set_zlabel("z distance from observer")

        fig2 = plt.figure(2, figsize=(8,8))
        ay = plt.subplot(111)
        plt.plot(loc.x,loc.z,'ro', label="mass distribution")
        plt.plot(loc.x,loc.z,'o',label="observer")
        plt.plot(apparentloc.x,apparentloc.z,'o',label="apparent source")
        plt.plot([loc.x,apparentloc.x],[loc.z,apparentloc.z],'orange',label="apparent path")
        plt.xlabel("x distance from observer")
        plt.ylabel("y distance from observer")

        fig3 = plt.figure(3, figsize=(8,8))
        ax = plt.subplot(111)
        plt.plot(loc.y,loc.z,'ro', label="mass distribution")
        plt.plot(loc.y,loc.z,'o',label="observer")
        plt.plot(apparentloc.y,apparentloc.z,'o',label="apparent source")
        plt.plot([loc.y,apparentloc.y],[loc.z,apparentloc.z],'orange',label="apparent path")
        plt.xlabel("x distance from observer")
        plt.ylabel("y distance from observer")

        locrecord = [(loc.x,loc.y,loc.z)]

    for m in massdistr:
        dz = m.loc.z - loc.z
        dx = dz*np.sin(xangle)
        dy = dz*np.sin(yangle)
        redshift -= np.sqrt(loc.dl2(dx,dy,dz)) #update ray location and pathlength
        if(plot):
            locrecord.append((loc.x,loc.y,loc.z))
            plt.figure(1)
            az.plot([m.loc.x], [m.loc.y], 'ro',zs=m.loc.z, markersize = 10*m.mass)
            az.plot([m.loc.x, loc.x], [m.loc.y, loc.y], 'r--',zs=[m.loc.z, loc.z])
            plt.figure(2)
            plt.plot(m.loc.x, m.loc.z, 'ro', markersize = 10*m.mass)
            plt.plot([m.loc.x, loc.x], [m.loc.z, loc.z], 'r--')
            plt.figure(3)
            plt.plot(m.loc.y, m.loc.z, 'ro', markersize = 10*m.mass)
            plt.plot([m.loc.y, loc.y], [m.loc.z, loc.z], 'r--')
        xdist = loc.x-m.loc.x #calculate bend angle
```

```
            ydist = loc.y-m.loc.y
            mdist = np.sqrt(xdist**2 + ydist**2)
            angle = bend(mdist, m.mass)
            dz_LS = totaldz-loc.z #calculate magnification
            dz_S = dz+dz_LS
            thetaE2 = m.mass*dz_LS/(dz*dz_S)
            theta = np.arcsin(np.sqrt(np.sin(xangle)**2 + np.sin(yangle)**2))
            beta = theta-angle*dz_LS/dz_S
            u2 = thetaE2 / beta**2
            stepmag = (u2 +2)/(2*np.sqrt(u2)*np.sqrt(u2 +4)) + 1/2
            if(theta**2 < thetaE2):
                stepmag -= 1
            magnification *= stepmag
            if(plot):
                print(str(m.mass)+" mass magnifies by "+str(stepmag))
            xangle -= angle*xdist/mdist #adjust angles
            yangle -= angle*ydist/mdist #do these two lines require small angle approximation?
        dz = redshift/np.sqrt(1 + np.sin(xangle)**2 + np.sin(yangle)**2)
        loc.x += dz*np.sin(xangle)
        loc.y += dz*np.sin(yangle)
        loc.z += dz
        if(plot):
            print("for a total magnification of "+str(magnification))
            locrecord.append((loc.x,loc.y,loc.z))
            xs = [l[0] for l in locrecord]
            ys = [l[1] for l in locrecord]
            zs = [l[2] for l in locrecord]
            plt.figure(1)
            plt.title("3-dimensional view")
            az.plot(xs ,[l[1] for l in locrecord],'g',zs=zs,label="light ray path")
            az.plot([loc.x],[loc.y], 'go', zs=loc.z, label="true source")
            az.legend()
            plt.figure(2)
            plt.title("projection on y=0 plane")
            plt.plot(xs, zs,'g',label="light ray path")
            plt.plot(loc.x,loc.z,'go',label="true source")
            plt.legend()
            plt.figure(3)
            plt.title("projection on x=0 plane")
            plt.plot(ys, zs,'g',label="light ray path")
            plt.plot(loc.y,loc.z,'go',label="true source")
            plt.legend()
            plt.show()
        magnifications.append(magnification)
        return loc #final x,y,z
```

In [3]:
```
ef imagesource(mags, massdistr, imagedistr, redshift, sourcezs):
   sourcedistr = []
   for xy in imagedistr:
       loc = beam(False,mags,redshift,sorted(massdistr, key=lambda m: m.loc.z),xy[0],xy[1])
       truexangle = np.arcsin(loc.x/loc.z)
       trueyangle = np.arcsin(loc.y/loc.z)
       sourcezs.append(loc.z)
       trueangle = (truexangle,trueyangle)
       sourcedistr.append(trueangle)
   return sourcedistr

edshift =250
assdistr = [mass(0.3,3,-2,60),mass(0.5,-3,-4,90),mass(0.8,-4,5,150)]

ef center(massdistr):
   centerx = 0
   centery = 0
   totalmassdist = 0
   for m in massdistr:
       centerx += m.mass*m.loc.x/m.loc.z
       centery += m.mass*m.loc.y/m.loc.z
       totalmassdist += m.mass/m.loc.z
   centerx /= totalmassdist
   centery /= totalmassdist
   for m in massdistr:
       m.loc.x -= centerx
       m.loc.y -= centery
   return

enter(massdistr)

pparentxangle = -np.pi/30
pparentyangle = np.pi/24

oc = beam(True, [], redshift,sorted(massdistr, key=lambda m: m.loc.z),apparentxangle,apparentyangle)
cuexangle = np.arcsin(loc.x/loc.z)
cueyangle = np.arcsin(loc.y/loc.z)

s = []
ags = []
netas = np.linspace(0,2*np.pi,32)
nage = [(apparentxangle + 0.02*np.cos(theta), apparentyangle + 0.02*np.sin(theta)) for theta in thetas]
nage += [(apparentxangle + 0.005*np.cos(theta), apparentyangle + 0.005*np.sin(theta)) for theta in thetas]
ource = imagesource(mags,massdistr,image,redshift,zs)

nagefig = plt.figure(4,figsize=(9,8))
 = plt.subplot(111)
omags =np.array(mags)
image = [pt[0] for pt in image]
image = [pt[1] for pt in image]
lt.scatter(ximage,yimage, c = mags, cmap="viridis")
lt.plot(ximage,yimage,"orange")
lt.axis('equal')
lt.title("angular image distribution")
lt.clim(npmags.min(),npmags.max())
oar = plt.colorbar()
oar.set_label("magnification")
lt.xlabel("radians from center of mass distribution")
lt.ylabel("radians from center of mass distribution")

ourcefig = plt.figure(5,figsize=(9,8))
 = plt.subplot(111)
source = [pt[0] for pt in source]
source = [pt[1] for pt in source]
ourcedists = np.array([zs[n]*np.sqrt(1+np.sin(source[n][0])**2+np.sin(source[n][1])**2) for n in range(0,len(zs))
lt.scatter(xsource,ysource, c= sourcedists, cmap="winter")
lt.plot(xsource,ysource)
lt.axis('equal')
lt.clim(sourcedists.min(),sourcedists.max())
oar = plt.colorbar()
oar.set_label("true distance from observer")
lt.title("angular source distribution")
lt.xlabel("radians from center of mass distribution")
lt.ylabel("radians from center of mass distribution")

ombfig = plt.figure(6, figsize=(9,7))
omb = plt.subplot(111)
lt.plot(xsource,ysource,label="source distribution")
lt.plot(ximage,yimage,label="image distribution")
asszs = np.array([m.loc.z for m in massdistr])
lt.scatter([np.arctan(m.loc.x/m.loc.z) for m in massdistr],[np.arctan(m.loc.y/m.loc.z) for m in massdistr],c=mass
lt.clim(0,redshift)
oar = plt.colorbar()
oar.set_label("distance to mass")
lt.title("view of true source, lensed image, and lensing masses")
lt.legend()
lt.xlabel("radians from center of mass distribution")
lt.ylabel("radians from center of mass distribution")
```

```
lt.axis('equal')
distfig = plt.figure(7, figsize=(7,8))
dist = plt.subplot(111)

def cn(array):
    return (array - array.min())/(array.max()-array.min())

cnxsource = cn(xsource)
cnysource = cn(ysource)
cnximage = cn(ximage)
cnyimage = cn(yimage)
plt.scatter(cnxsource,cnysource,label="source distribution")
plt.scatter(cnximage,cnyimage,"orange",label="image distribution")
plt.plot([cnxsource])
plt.title("distortion")
plt.legend()
plt.axis('equal')
lt.show()
```



In [ ]: