Gravitational Lensing

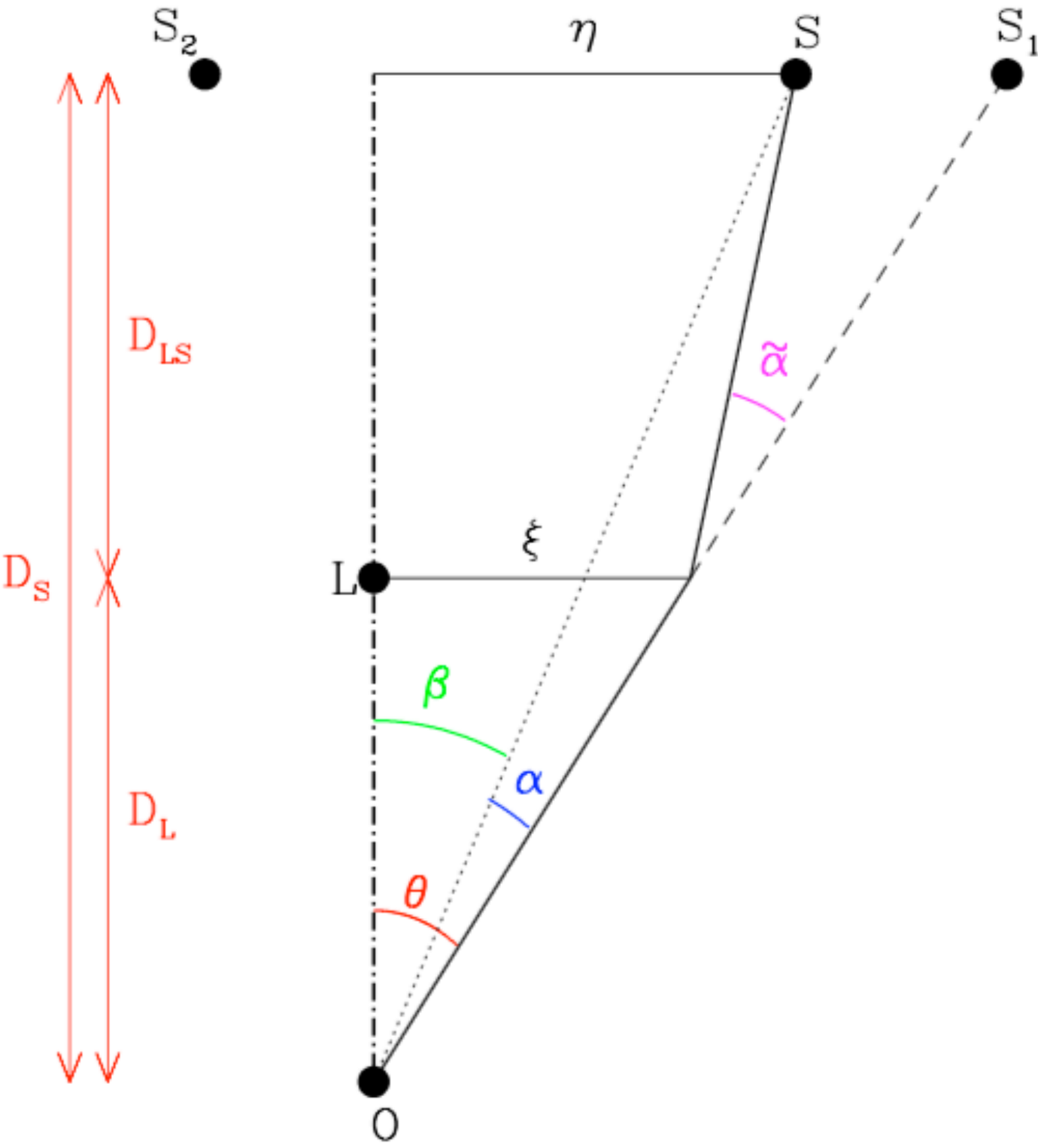
Yvan Quinn

Basic geometry

approximations:

- small angle
- weak gravity
- thin lens

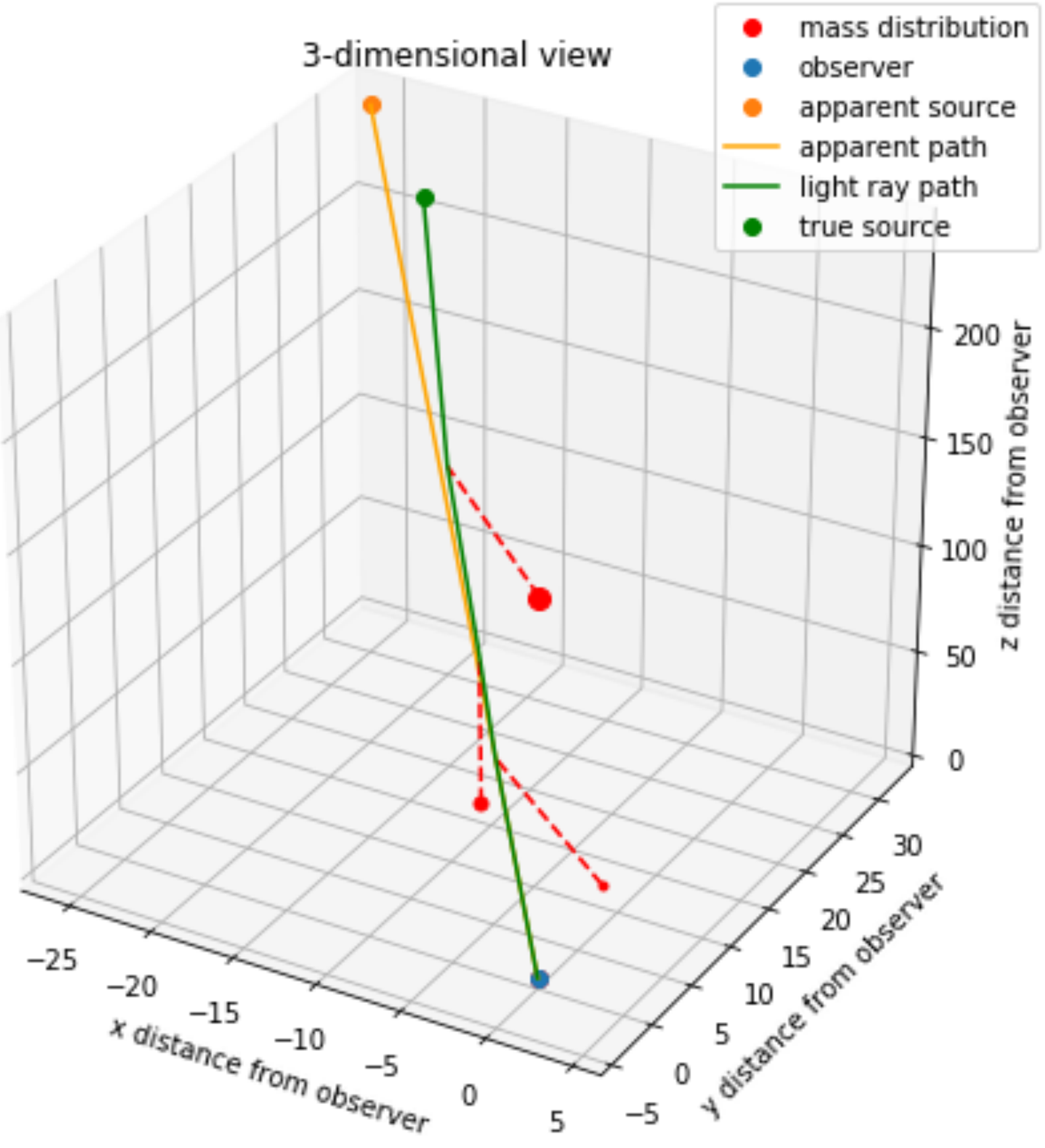
result in linearity of $\tilde{\alpha}$ within lens

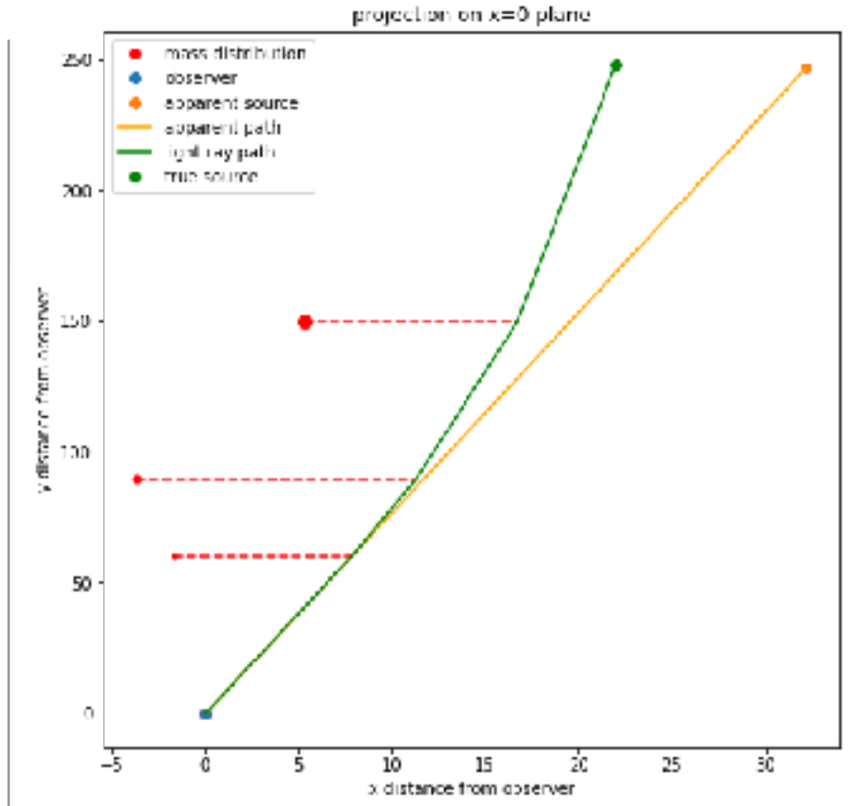


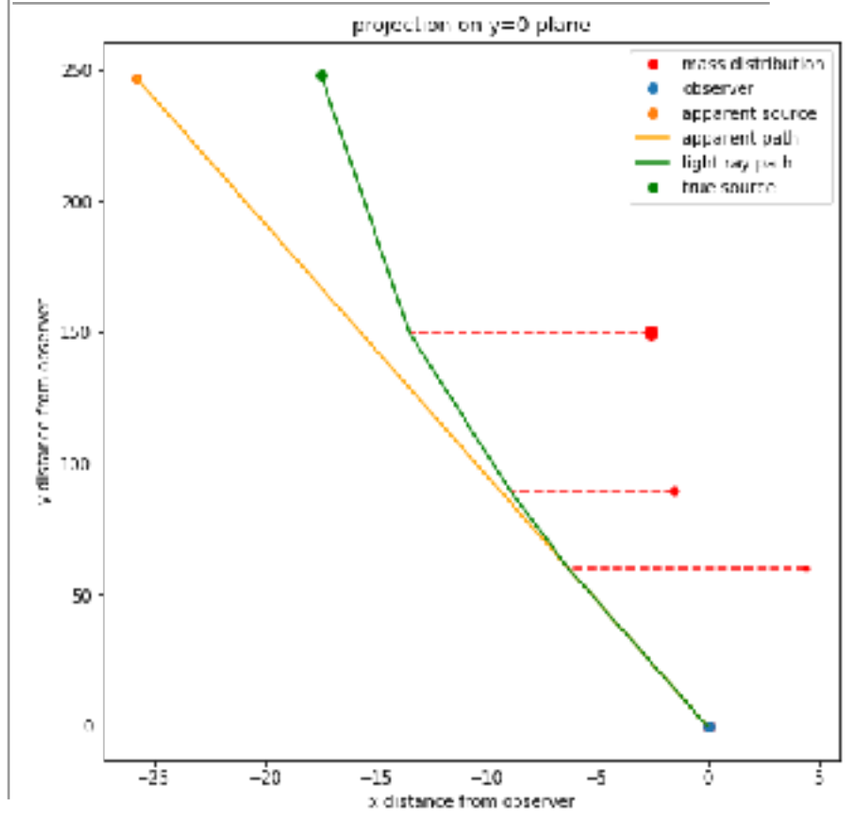
```
    for m in massdistr:
        dz = m.loc.z - loc.z
        dx = dz*np.sin(xangle)
        dy = dz*np.sin(yangle)
        redshift -= np.sqrt(loc.dl2(dx,dy,dz)) #update ray location and pathlength
        if(plot):
            [illegible]
        xdist = loc.x-m.loc.x #calculate bend angle
        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
```

iteration over lenses

deflection, magnification







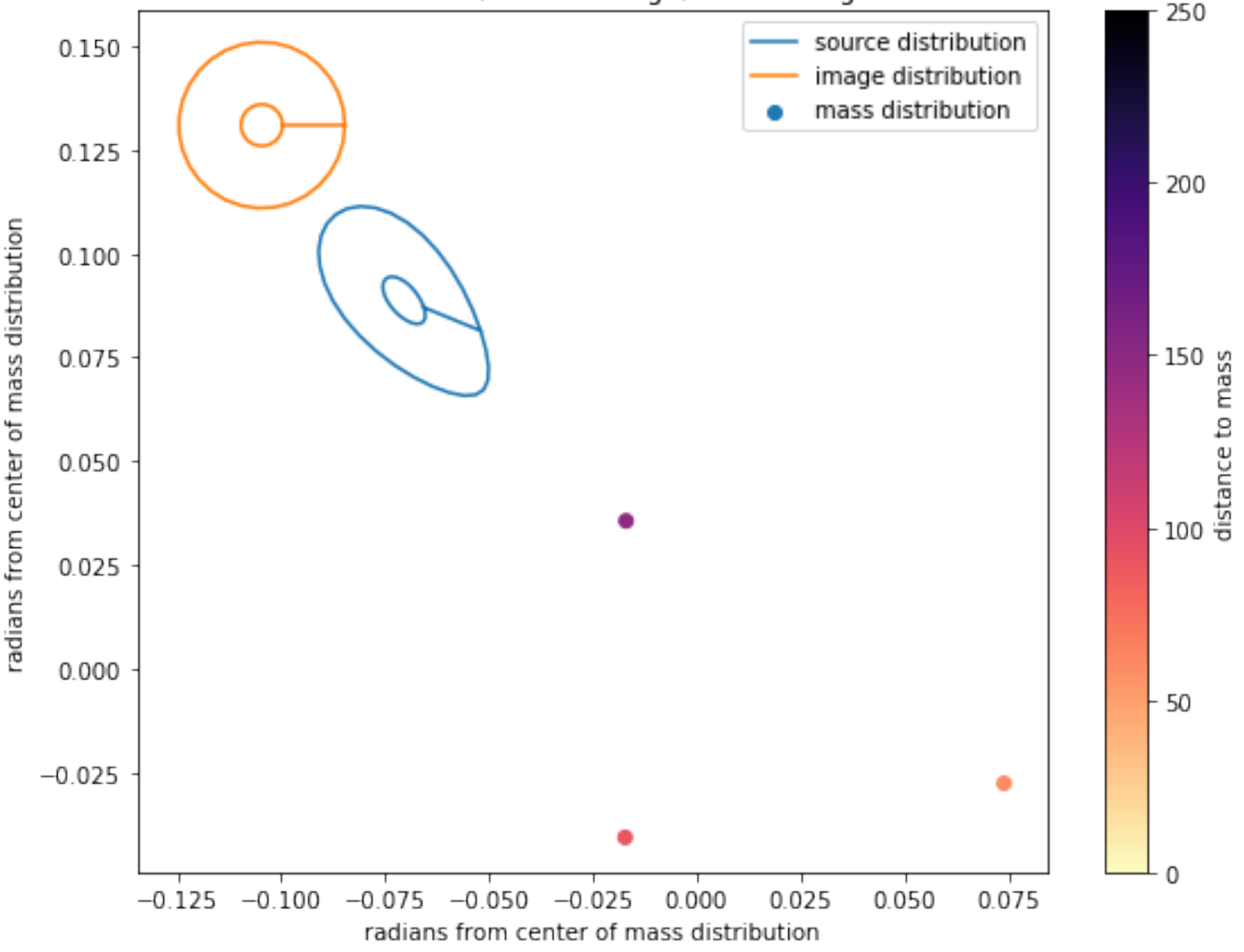
a fairly pedestrian situation with three lensing masses

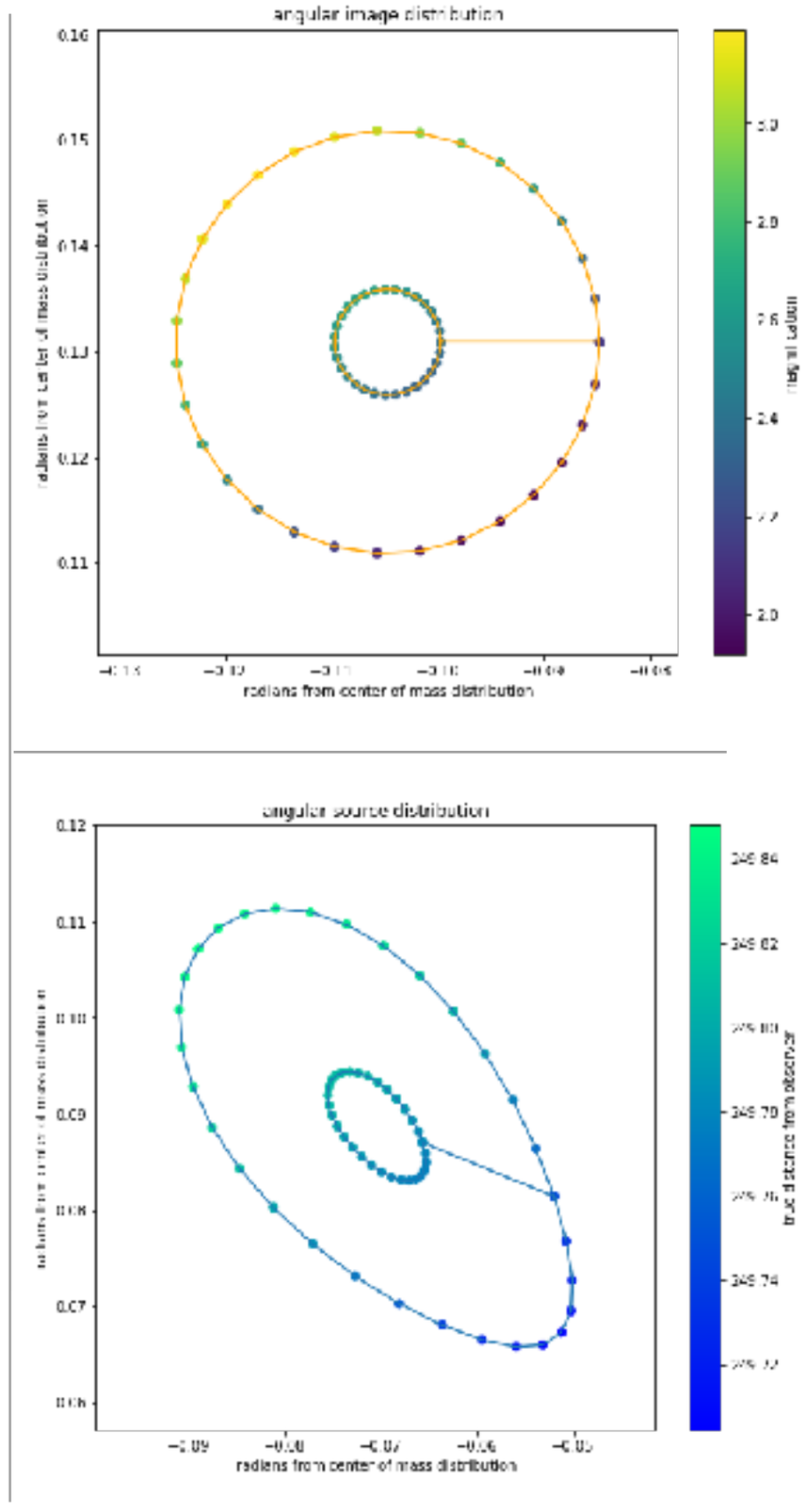
```
def imagesource(mags, massdistr, imagedistr, redshift, sourcezs):
    sourcedistr = []
    for xy in imagedistr:
        loc = beam(False,mags,redshift,sorted(massdistr, key=lambda m: m.loc.z),xy[0],xy[1])
        truexangle = np.arcsin(loc.x/loc.z)
        trueyangle = np.arcsin(loc.y/loc.z)
  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

def 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
```

```
redshift =250
massdistr = [mass(0.3,3,-2,60),mass(0.5,-3,-4,90),mass(0.8,-4,5,150)]
center(massdistr)
apparentxangle = -np.pi/30
apparentyangle = np.pi/24
loc = beam(True, [], redshift,sorted(massdistr, key=lambda m: m.loc.z),apparentxangle,apparentyangle)
truexangle = np.arcsin(loc.x/loc.z)
trueyangle = np.arcsin(loc.y/loc.z)
zs = []
mags = []
thetas = np.linspace(0,2*np.pi,32)
image = [(apparentxangle + 0.02*np.cos(theta), apparentyangle + 0.02*np.sin(theta)) for theta in thetas]
image += [(apparentxangle + 0.005*np.cos(theta), apparentyangle + 0.005*np.sin(theta)) for theta in thetas]
source = imagesource(mags,massdistr,image,redshift,zs)
```

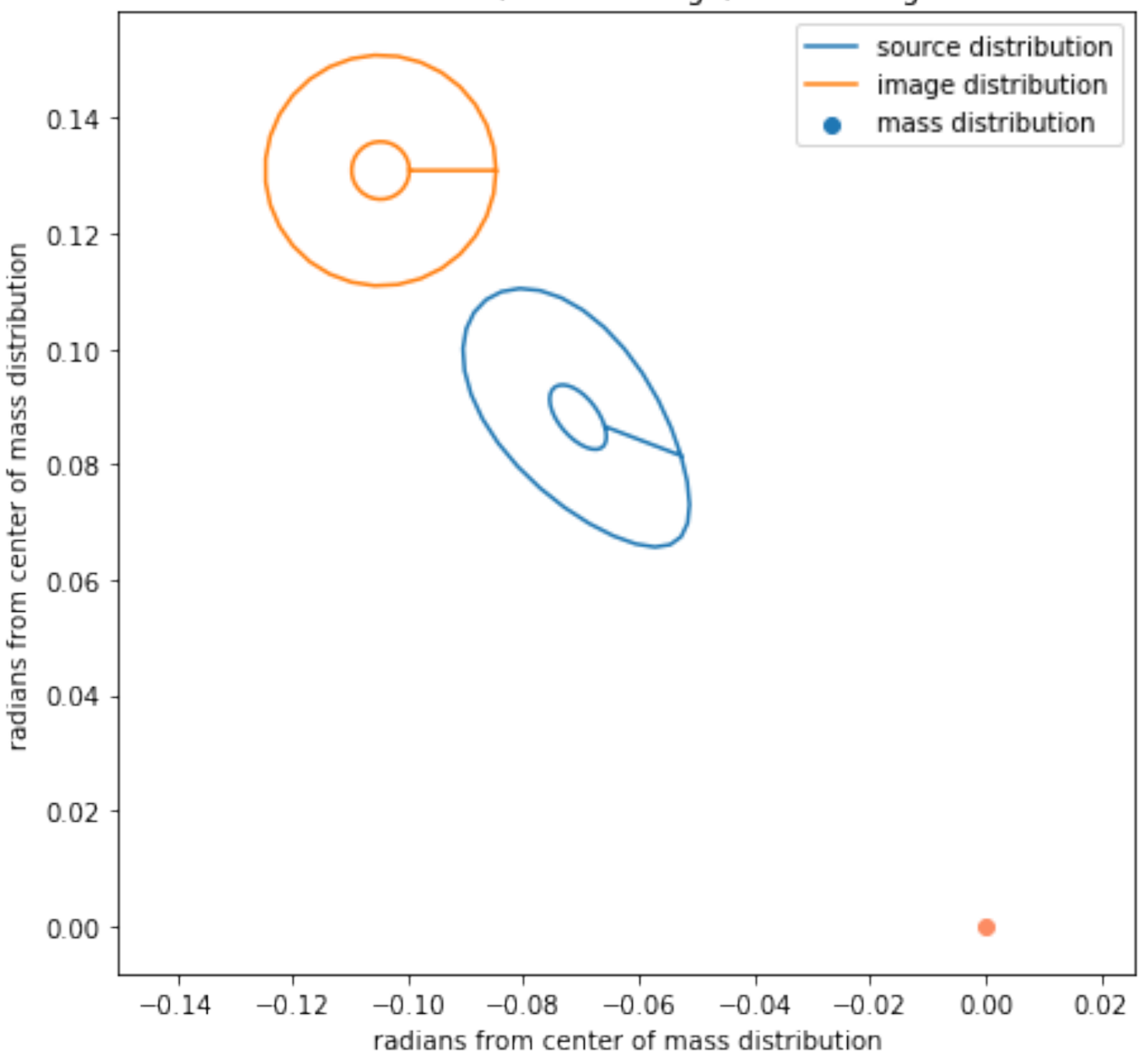
basic setup, execution

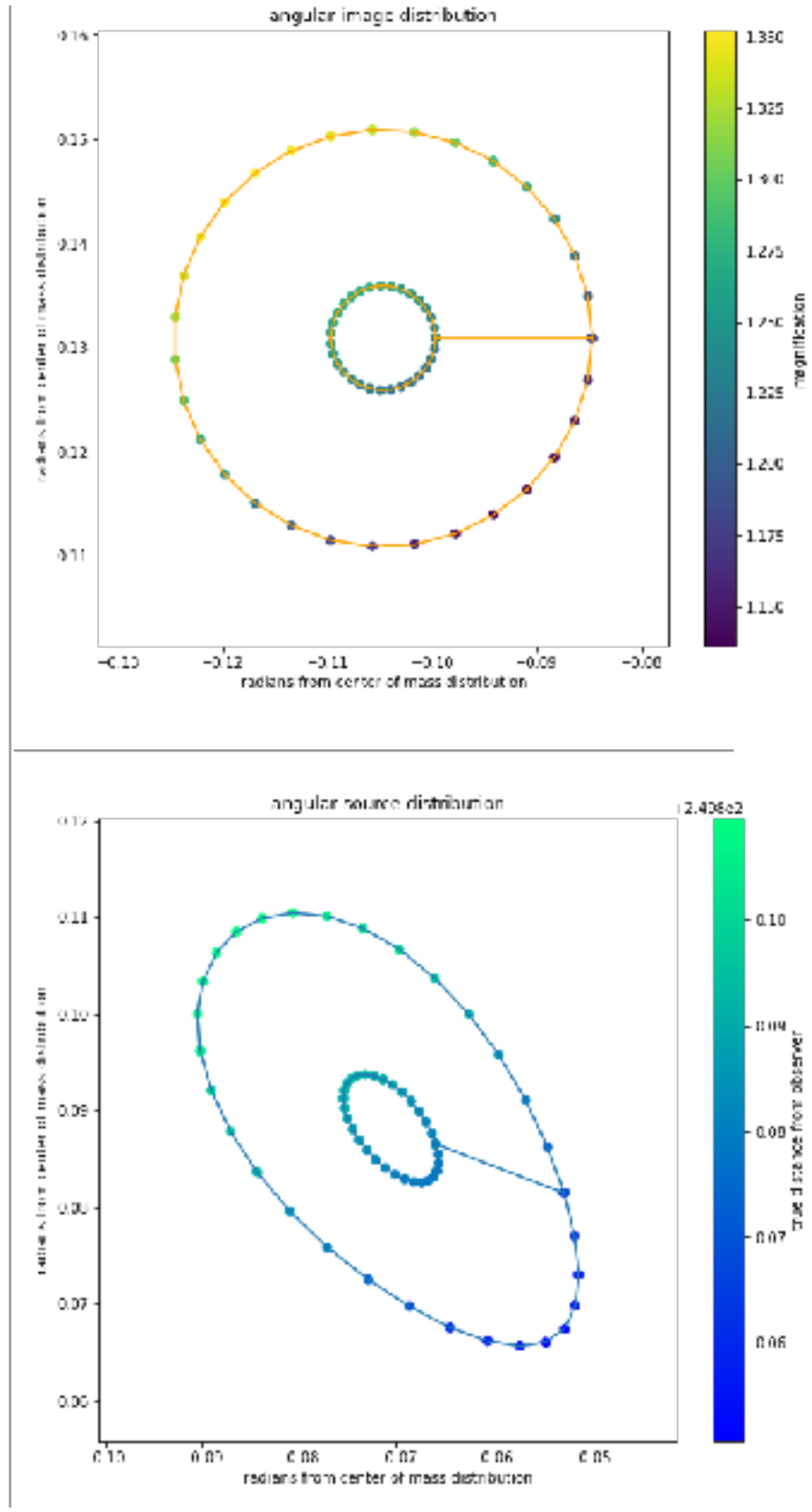
mass and source distribution



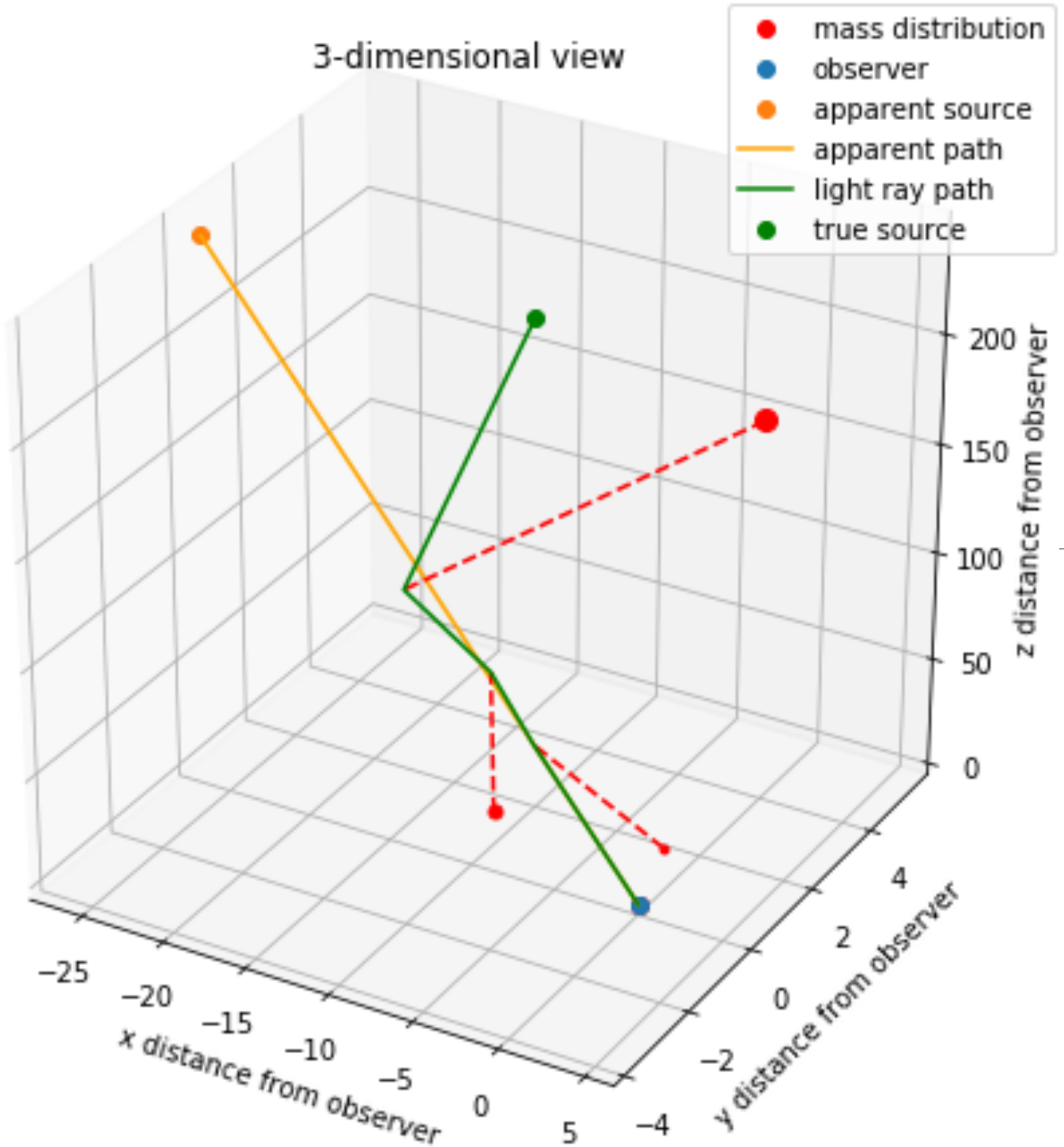


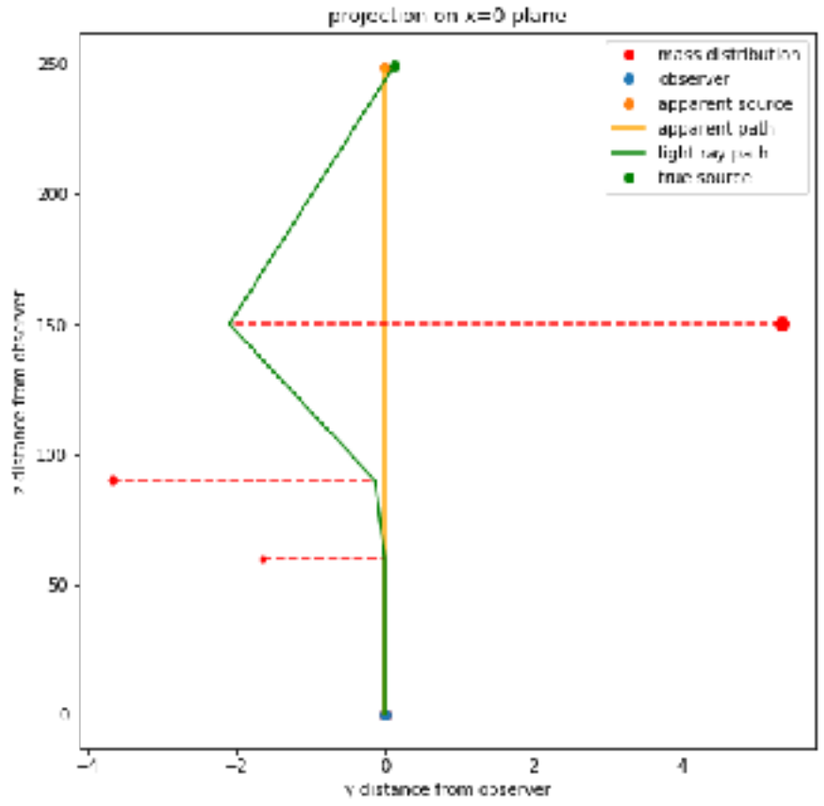
acts much the same as with a single lens

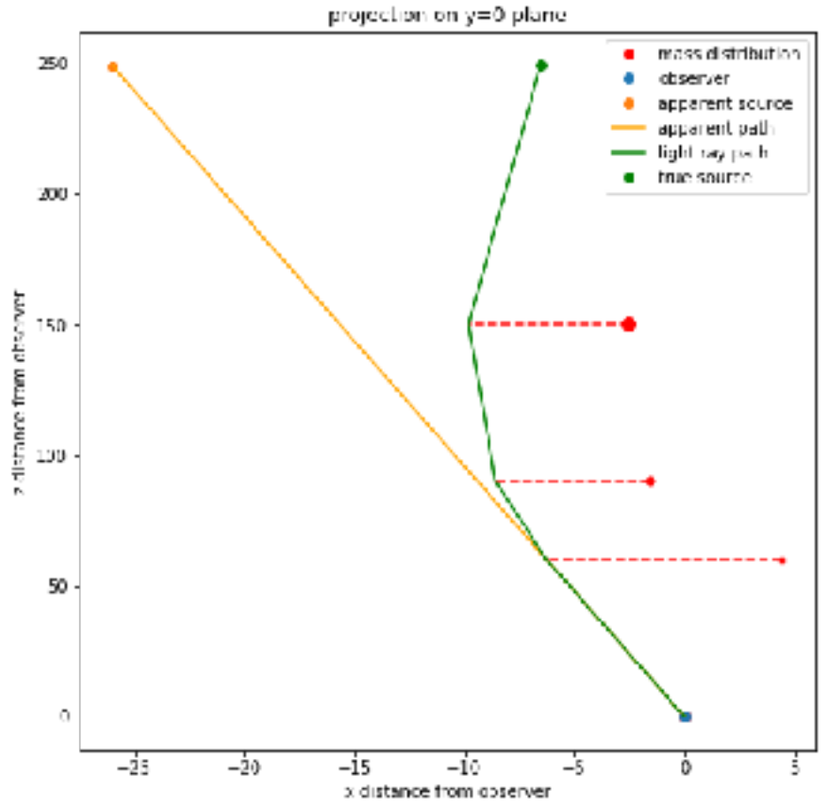




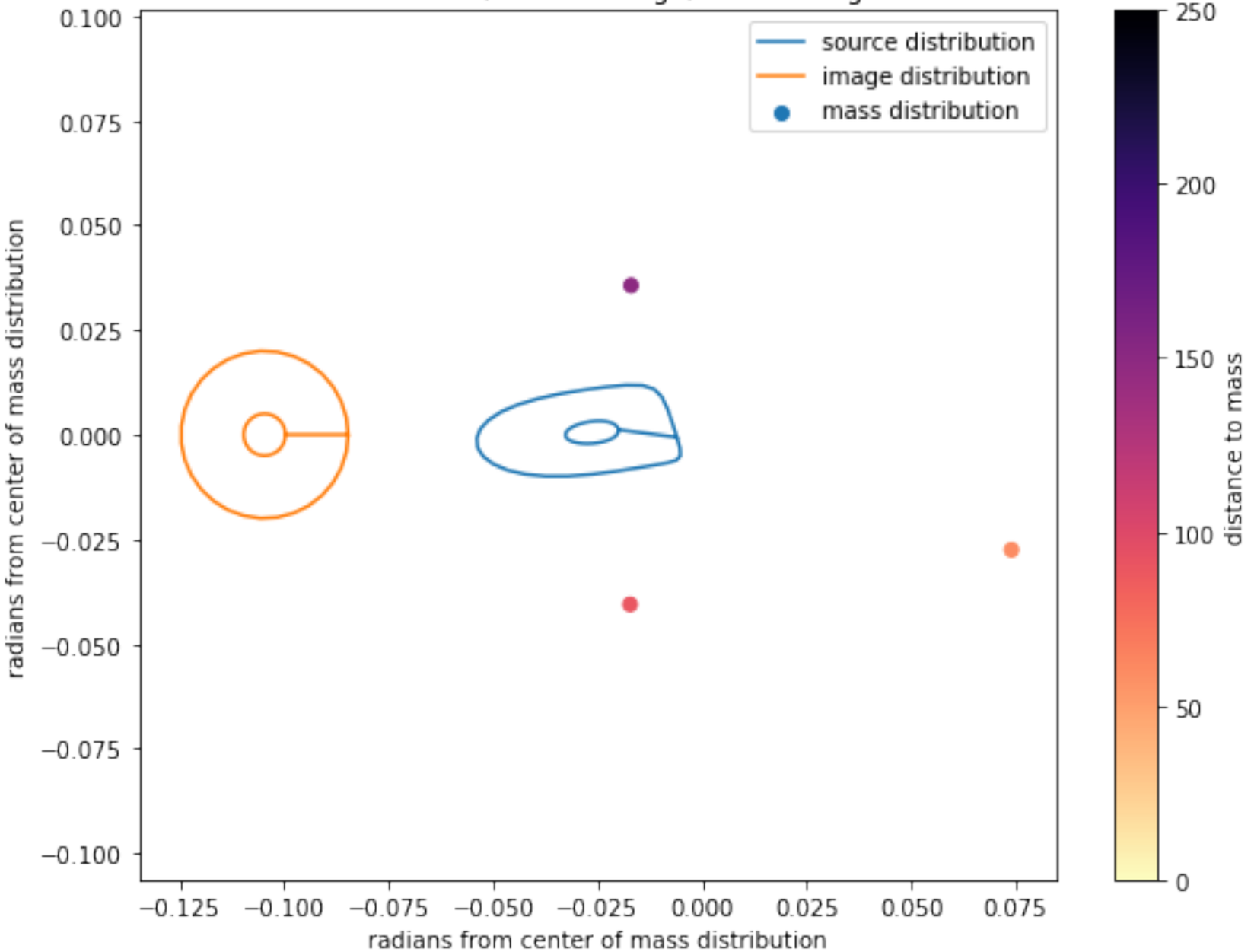
note smaller magnification and greater distance spread

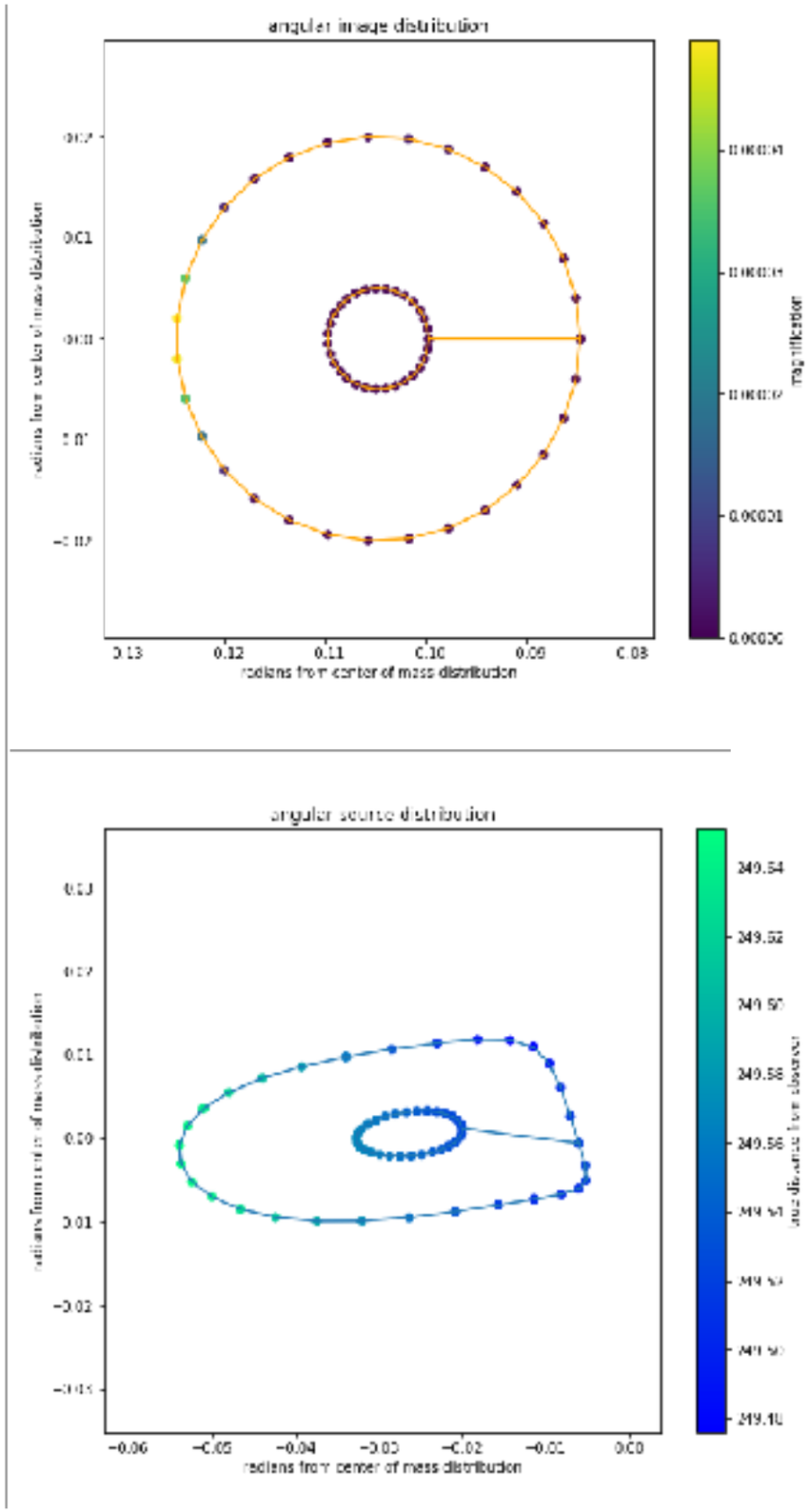






a more exotic situation from same mass distribution





this distortion is clearly not possible from one mass