displayHalos

May 18, 2022

[1]:
```
import sys
sys.path.append('/usr/local/lib/python3/dist-packages/crpropa/')
sys.path.append('/home/eiche/Dokumente/Dokumente_Uni/Projects/RGdeflCons/')
import crpropa as crp
import numpy as np
import matplotlib.pylab as plt
from matplotlib.markers import MarkerStyle
plt.rcParams.update({'font.size': 15})
import seaborn as sns
PATH_EGMF = "/media/eiche/Samsung_T3/Bfields_Hackstein2018/"
PATH_EGMF_new = "/media/eiche/Samsung_T3/Bfields_Hackstein2018/newFiles/"
PATH_plots = "./Plots/"
```

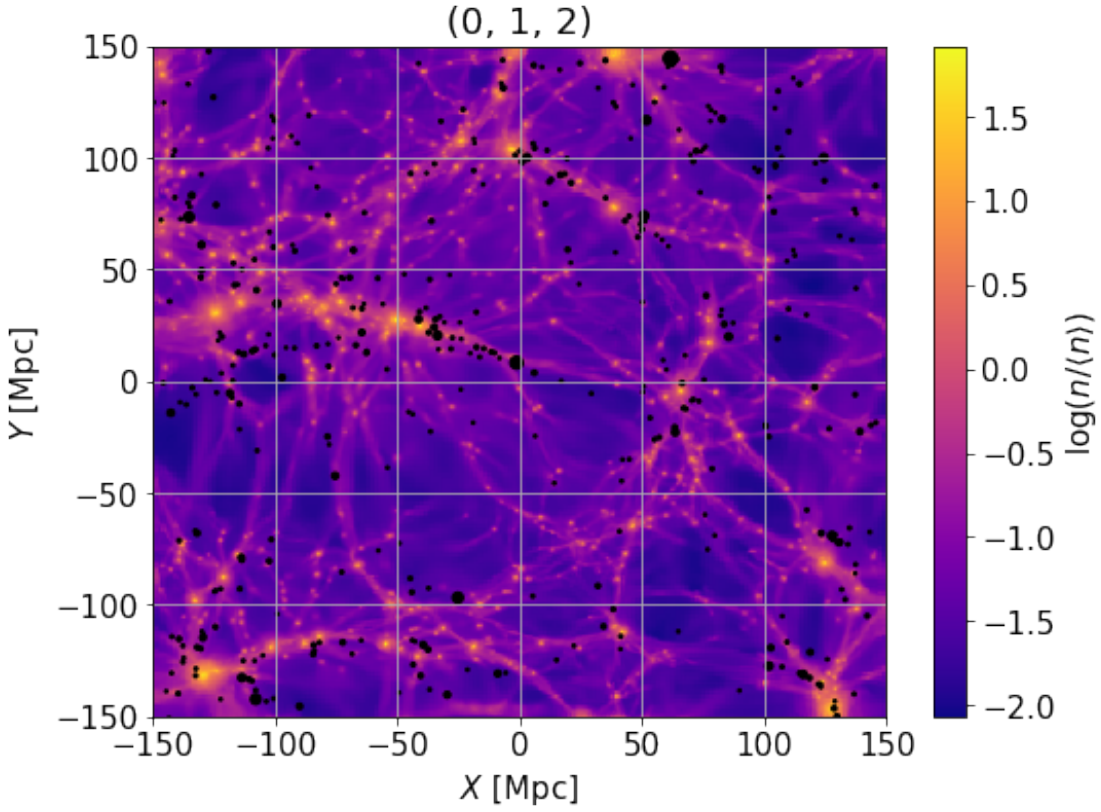
[2]:
```
filename_density = PATH_EGMF_new+"mass-density.dat"
gridOrigin = crp.Vector3d(0,0,0)              ## origin of the 3D data,␣
 ↪preferably at boxOrigin
gridSize = 1024                              ## size of uniform grid in data points
h = 0.677                                    ## dimensionless Hubble parameter
size = 249.827*crp.Mpc / h                       ## physical edgelength of␣
 ↪volume in Mpc
b_factor = 1.                                ## global renormalizatino factor for␣
 ↪the field
obsPosition = crp.Vector3d(0.5*size,0.5*size,0.5*size)/crp.Mpc # position of␣
 ↪observer, MW is in center of constrained simulations

## settings of simulation
boxSize = crp.Vector3d( size, size, size )   ## end of the full box of the␣
 ↪simulation

## settings for computation
minStep = 10.*crp.kpc                        ## minimum length of single step␣
 ↪of calculation
maxStep = 4.*crp.Mpc                         ## maximum length of single step␣
 ↪of calculation
tolerance = 1e-2                             ## tolerance for error in iterative␣
 ↪calculation of propagation step
```

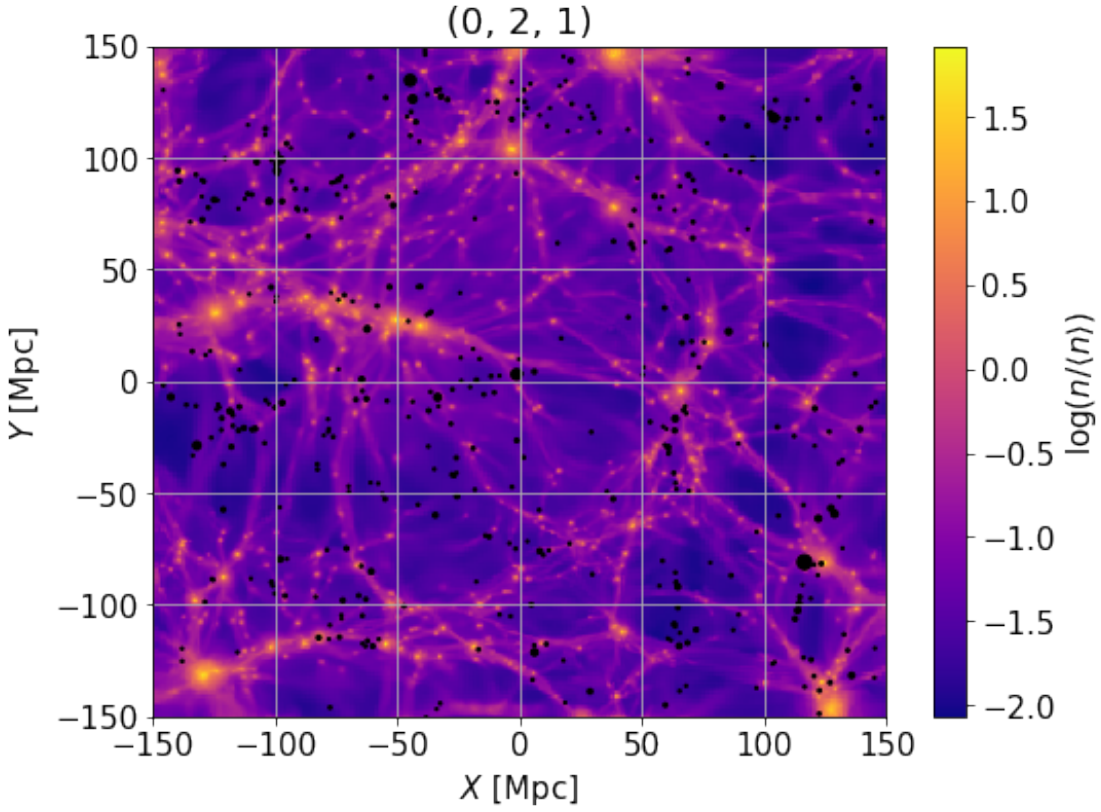
```
spacing = size/(gridSize)                ## resolution, physical size of single␣
↪cell

####Load the mass grid:
mgrid = crp.Grid1f( gridOrigin, gridSize, spacing )
crp.loadGrid( mgrid, filename_density )

#######################################################################
filename_halos = PATH_EGMF+'clues_halos.dat'

# read data from file
data = np.loadtxt(filename_halos, unpack=True, skiprows=39)
sX = data[0]
sY = data[1]
sZ = data[2]
mass_halo = data[5]

## find only those mass halos inside the provided volume (see Hackstein et al.␣
↪2018 for more details)
Xdown= sX >= 0.25
Xup= sX <= 0.75
Ydown= sY >= 0.25
Yup= sY <= 0.75
Zdown= sZ >= 0.25
Zup= sZ <= 0.75
insider= Xdown*Xup*Ydown*Yup*Zdown*Zup

## transform relative positions to physical positions within given grid
sX = (sX[insider]-0.25)*2*size
sY = (sY[insider]-0.25)*2*size
sZ = (sZ[insider]-0.25)*2*size
mass_halo = mass_halo[insider]
```

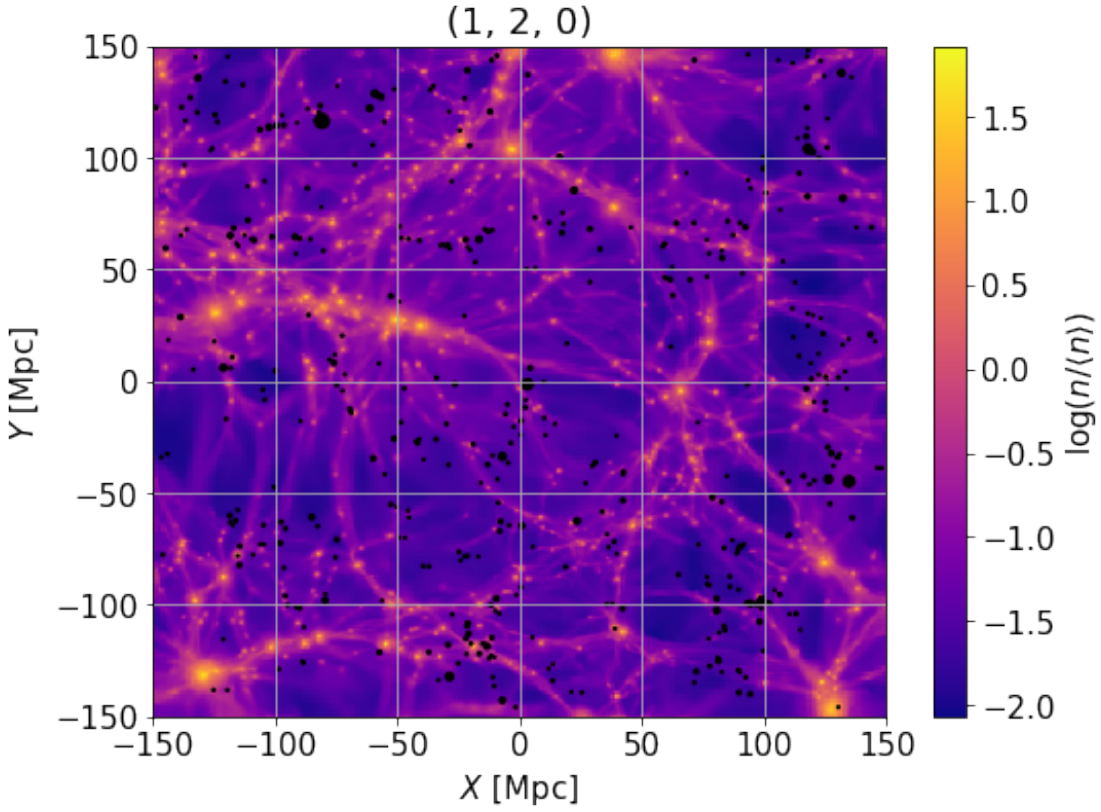
[3]:
```
s = 128
off = (1024-s)//2
R = s * spacing/crp.Mpc
Xs, Ys = np.arange(0,size/crp.Mpc,size/crp.Mpc/1024), np.arange(0,size/crp.
↪Mpc,size/crp.Mpc/1024)   # Mpc
Zs = np.arange(off*size/crp.Mpc/1024,(1024-off)*size/crp.Mpc/1024,size/crp.Mpc/
↪1024)
#
dens_crp = np.zeros((len(Xs),len(Ys)))

for i,x in enumerate(Xs):
    for j,y in enumerate(Ys):
        _dens = []
        for k,z in enumerate(Zs):
```

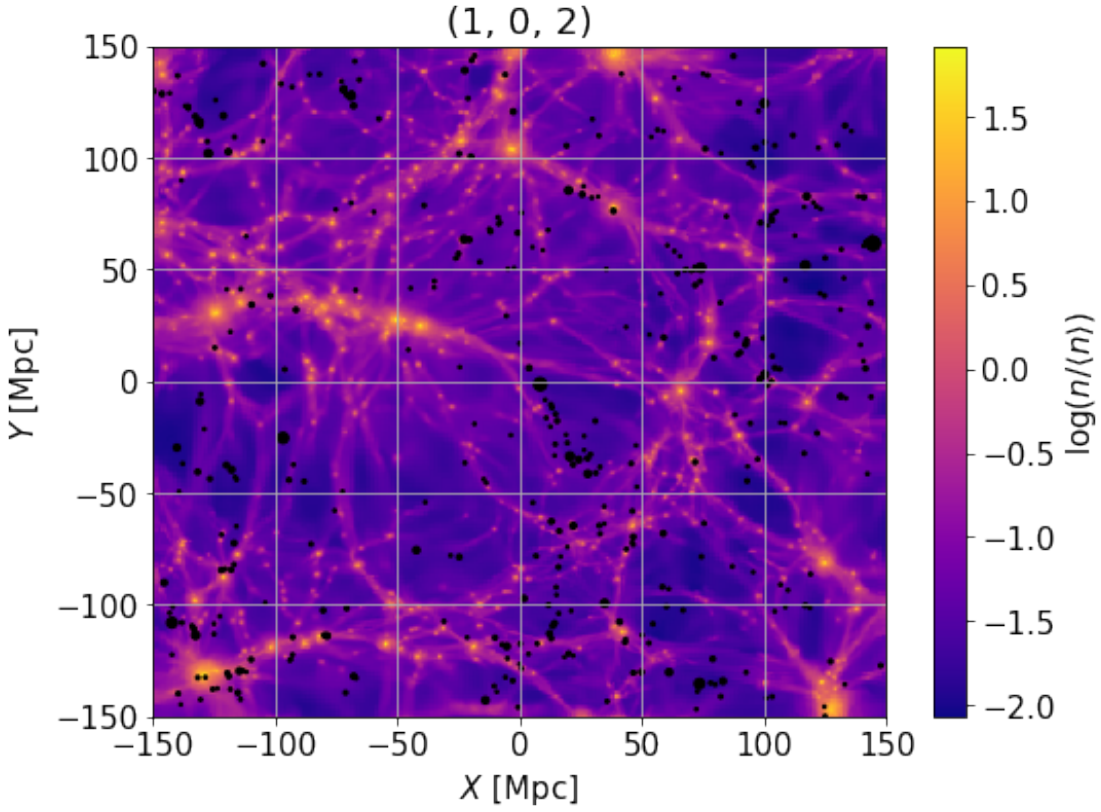
```
            r = crp.Vector3d(x, y, z)*crp.Mpc
            _dens.append(mgrid.closestValue(r))
        dens_crp[i,j] = np.mean(_dens)
```

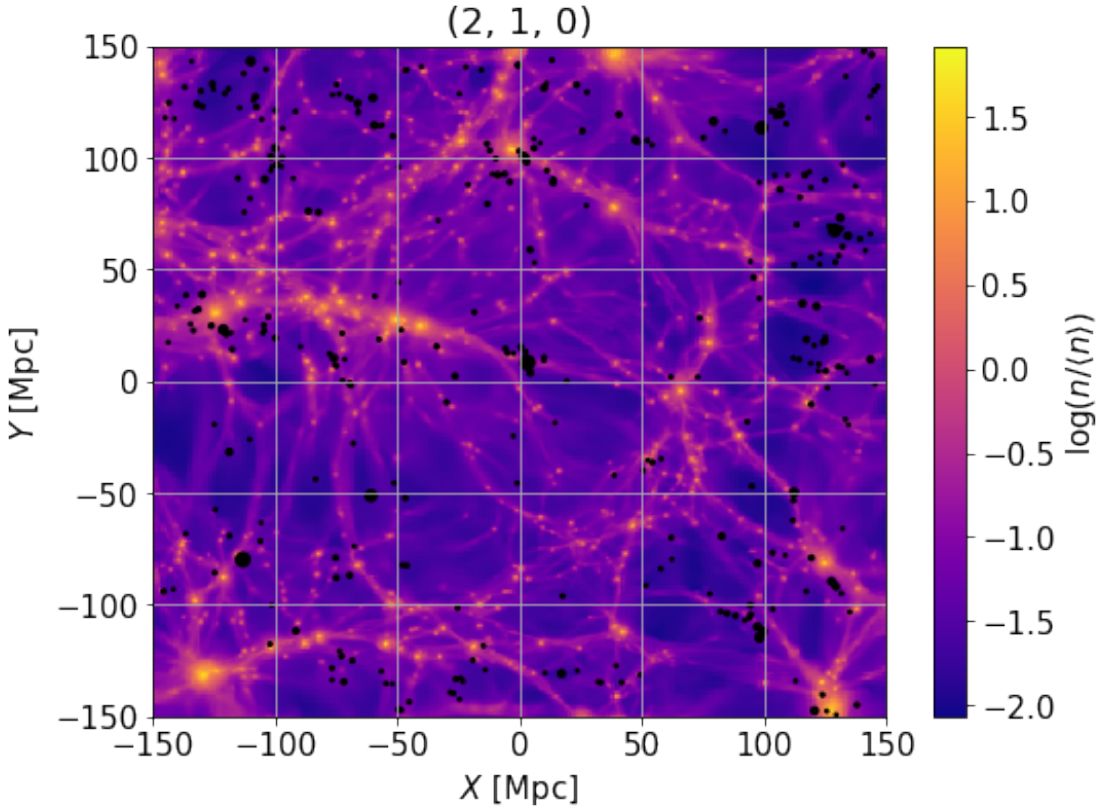
[11]:
```
#########
Xsgal, Ysgal = Xs - obsPosition[0], Ys - obsPosition[1]

####
#order = (1,2,0)
orders = [(0,1,2), (0,2,1), (1,2,0), (1,0,2), (2,1,0), (2,0,1)]
for order in orders:
    if order == (0,1,2):
        _x,_y,_z = sX, sY, sZ
    elif order == (2,1,0):
        _z,_y,_x = sX, sY, sZ
    elif order == (2,0,1):
        _z,_x,_y = sX, sY, sZ
    elif order == (0,2,1):
        _x,_z,_y = sX, sY, sZ
    elif order == (1,2,0):
        _y,_z,_x = sX, sY, sZ
    elif order == (1,0,2):
        _y,_x,_z = sX, sY, sZ
    ####
    plt.figure(figsize=(8, 6))
    cont = plt.pcolormesh(Xsgal, Ysgal, np.log10(dens_crp.T), cmap='plasma',␣
↪shading='auto')
    ####
    idx = (_z/crp.Mpc>off*size/crp.Mpc/1024) & (_z/crp.Mpc<(1024-off)*size/crp.
↪Mpc/1024)
    plt.scatter(_x[idx]/crp.Mpc- obsPosition[order[0]], _y[idx]/crp.Mpc-␣
↪obsPosition[order[1]], s=mass_halo[idx]/np.max(mass_halo[idx])*200,␣
↪color='black', marker='.')

    ####
    ####
    clb = plt.colorbar(cont, orientation='vertical')
    clb.set_label(r'log($n/\langle n \rangle$)')
    plt.title(order)
    plt.xlabel('$X$ [Mpc]')
    plt.ylabel('$Y$ [Mpc]')
    plt.xlim(-150,150)
    plt.ylim(-150,150)
    plt.grid()
    #plt.savefig(PATH_plots+'halosSwX-Z_massDens_s'+str(s)+'.pdf',␣
↪bbox_inches='tight')
    plt.show()
```

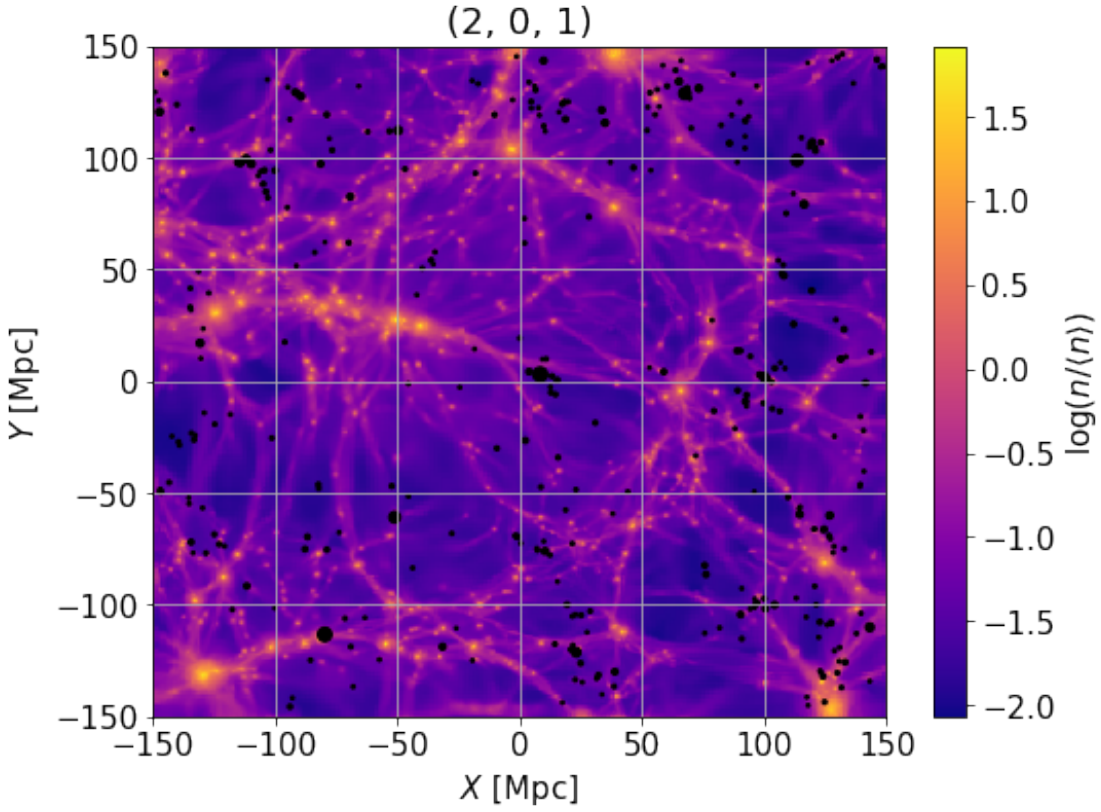

(0, 1, 2)
150
100
50
0
−50
−100
−150
Y [Mpc]
−150
−100
−50
0
50
100
150
X [Mpc]
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
log(n/⟨n⟩)


(0, 2, 1)
150
100
50
0
−50
−100
−150
Y [Mpc]
−150
−100
−50
0
50
100
150
X [Mpc]
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
log(n/⟨n⟩)


(1, 2, 0)
150
100
50
0
−50
−100
−150
Y [Mpc]
−150
−100
−50
0
50
100
150
X [Mpc]
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
log(n/⟨n⟩)


(1, 0, 2)
150
100
50
0
−50
−100
−150
Y [Mpc]
−150
−100
−50
0
50
100
150
X [Mpc]
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
log(n/⟨n⟩)


(2, 1, 0)
150
100
50
0
−50
−100
−150
Y [Mpc]
−150
−100
−50
0
50
100
150
X [Mpc]
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
log(n/⟨n⟩)


(2, 0, 1)
150
100
50
0
−50
−100
−150
Y [Mpc]
−150
−100
−50
0
50
100
150
X [Mpc]
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
log(n/⟨n⟩)

[ ]:

[ ]: