

# Workreport<sub>(16th)</sub>

## 1. 进一步完成黑洞判据代码的改进和测试:

上周进行测试时粒子数较少，当粒子数多起来后总是跳出 segmentation fault(core dumped)的情况，所以上周只是初步令代码运行了起来。频繁碰壁后，终于在 YYH 的启发下，发现问题所在：在函数中使用了一个结构体指针 `struct *r`,而未对它赋予足够的空间，即数组长度，所以导致当输出数组过长时由于空间不够而崩溃

```
particle_data *r;
```

```
.....
```

```
return r;
```

解决:

```
particle_data *r;
```

```
.....
```

```
r=(particle_data*)malloc(n*sizeof(struct particle_data));
```

```
for(i=0;i<n;i++) r[i]=q[i];
```

```
return r;
```

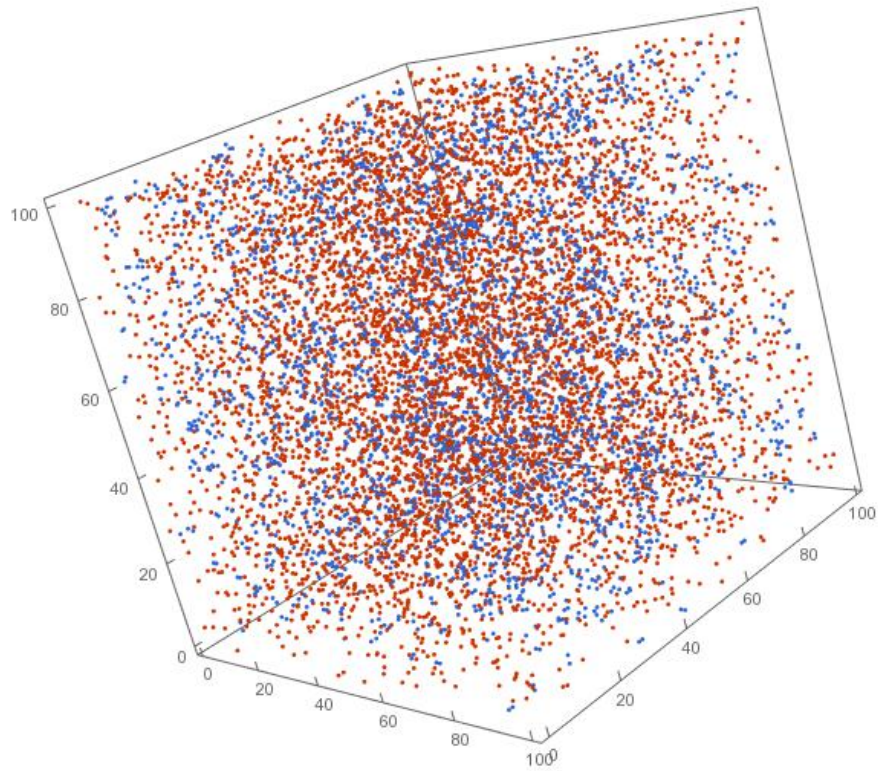
完成了两次测试，结果大致如下：

A. 均匀分布的 10000 个粒子的代码运行以及 Mathematica 可视化：


红色部分为筛选出的粒子（即局部密度较高），蓝色加红色为所有粒子

```
ListPointPlot3D[{data1, data2}, BoxRatios -> {1, 1, 1}]
```

```
ListPointPlot3D[{data1, data2}, PlotTheme -> "Web", BoxRatios -> {1, 1, 1}]
```



B. 对 YYH 跑的模拟进行了数据提取（修改 `snapshot_reader.c`, 将位置信息单独输出到一个 `.txt` 文件中）

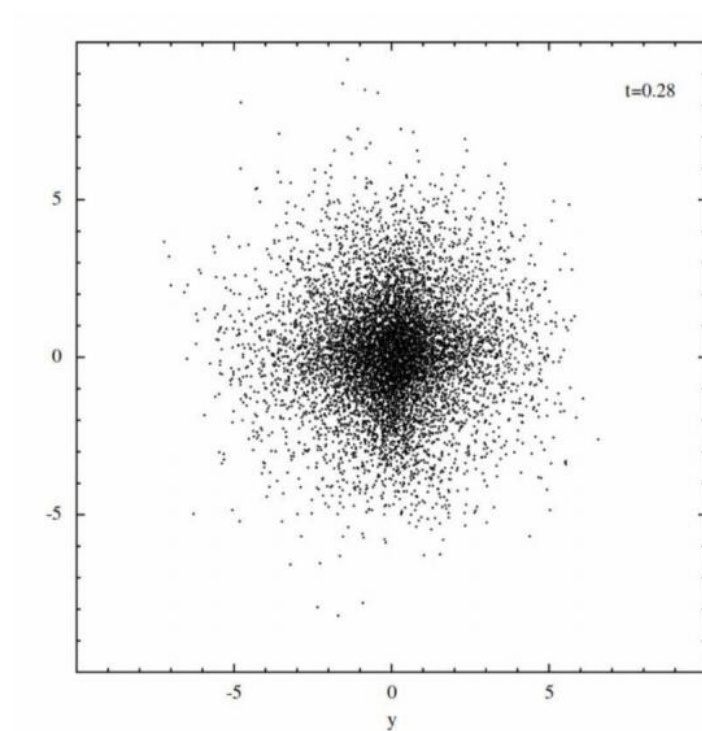


random1\_028\_posi.txt

```
random1_028_posi.txt
-4.97 -6.29 -5.52
-3.07 -5.32 -1.16
-3.37 -5.39 -1.31
-0.87 -5.10 -4.06
0.55 -5.50 -4.35
-0.38 -4.85 -2.63
0.25 -5.39 -2.47
-0.25 -5.45 -1.64
-0.25 -5.45 -1.64
-0.46 -5.02 -1.39
-0.78 -4.80 -2.01
-1.33 -4.79 -2.94
-0.54 -5.38 -2.63
-0.51 -5.45 -2.95
0.33 -0.50 -9.51
-0.95 -0.43 -5.55
-0.28 -0.82 -5.99
-0.14 -0.94 -4.68
-0.14 -0.94 -4.68
-1.48 -0.35 -4.74
-0.73 -0.63 -4.44
-1.46 -1.53 -4.48
-1.05 -1.44 -4.58
-1.45 -2.52 -4.95
0.46 -1.73 -5.34
0.07 -2.18 -4.43
0.12 -2.48 -3.95
-0.26 -1.94 -4.04
-0.26 -1.94 -4.03
-0.26 -1.94 -4.04
-0.04 -1.47 -3.68
-0.45 -1.93 -3.43
-0.26 -1.58 -3.43
0.07 -1.48 -3.51
0.08 -2.02 -3.51
0.46 -2.04 -3.57
0.46 -2.04 -3.57
-0.79 -2.47 -3.59
-0.79 -2.47 -3.59
-0.50 -2.50 -4.40
-0.99 -2.14 -4.02
-1.14 -1.51 -3.93
-1.14 -1.51 -3.93
-0.74 -1.89 -3.66
-0.62 -0.89 -3.47
-0.61 -1.25 -3.72
-0.61 -1.25 -3.72
-0.99 -1.15 -3.49
```

Line 21, Column 18

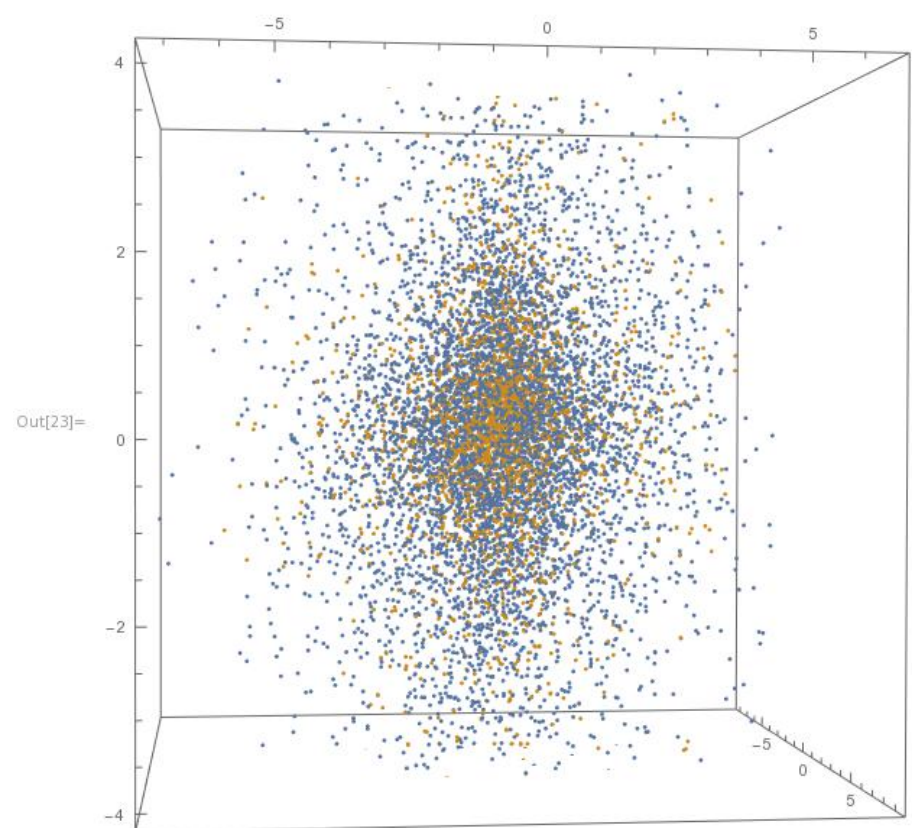
代码运行和 Mathematica 可视化:



(d)  $t = 0.28$  Gyr

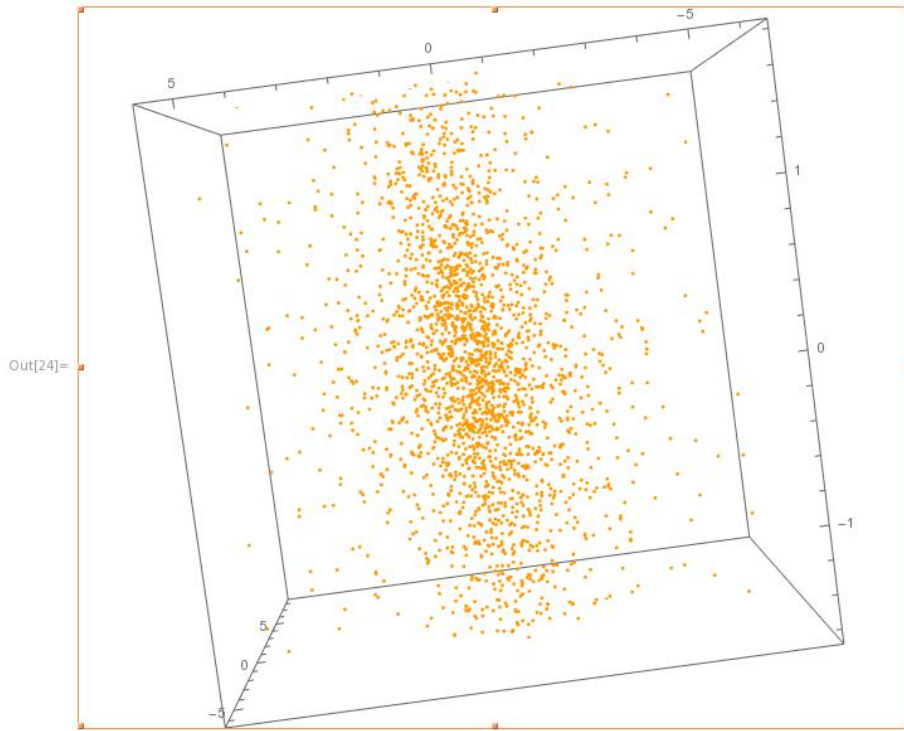
Gsplash 对 snapshot 的直接可视化

In[23]:= `ListPointPlot3D[{data1, data2}, BoxRatios -> {1, 1, 1}]`



画在同一坐标系（橙色为筛选出的粒子，蓝色+橙色为所有粒子）

```
In[24]:= ListPointPlot3D[data2, PlotStyle -> RGBColor[1., 0.6, 0.], BoxRatios -> {1, 1, 1}]
```

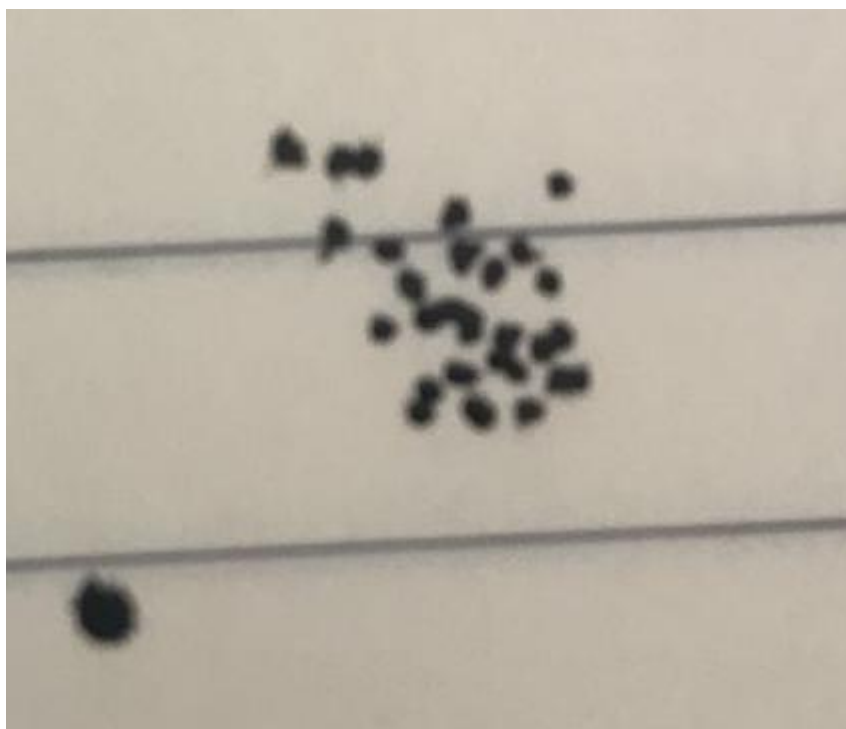


分开看这些粒子（此时橙色为筛选出粒子，因第一张图中位置相同的粒子全部为橙色）

结果分析：

1.起到了一定的筛选效果，可以看到，橙色粒子较所有粒子而言，更多的集中在中心部分，而更少的分布在周围  
但是结果不是很理想，理想的效果应该是，橙色几乎全部集中在中间部分，而周边部分几乎没有，而且中间部分几乎不存在蓝色.原因可能在算法内部：

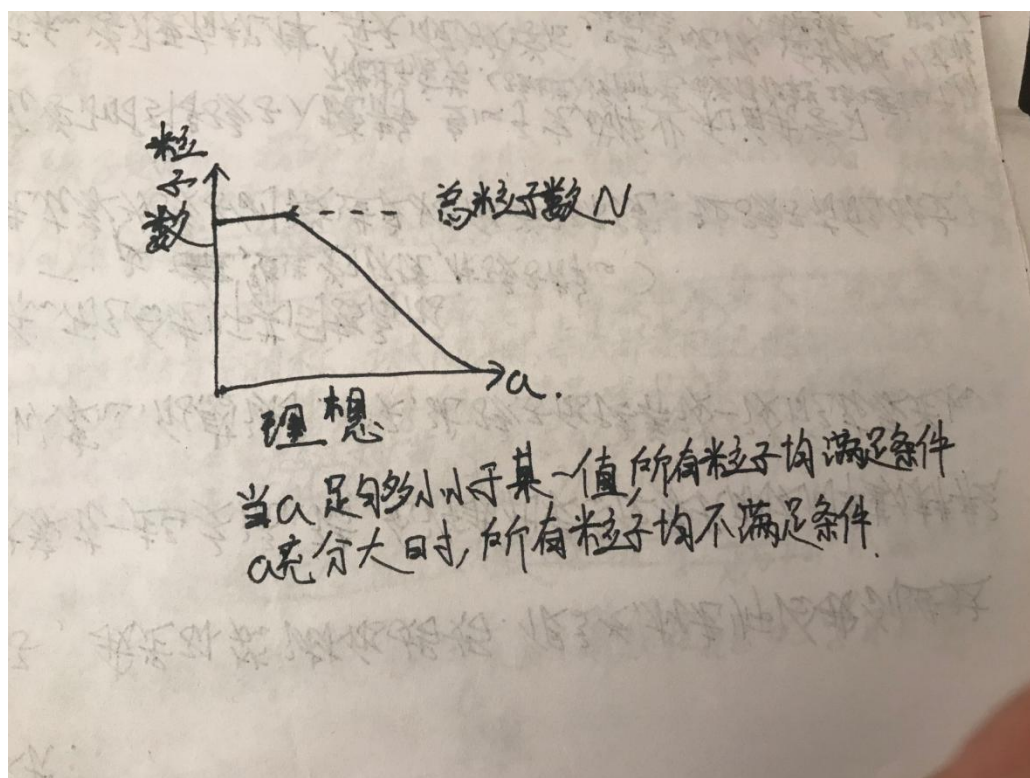
如下图例中情况：



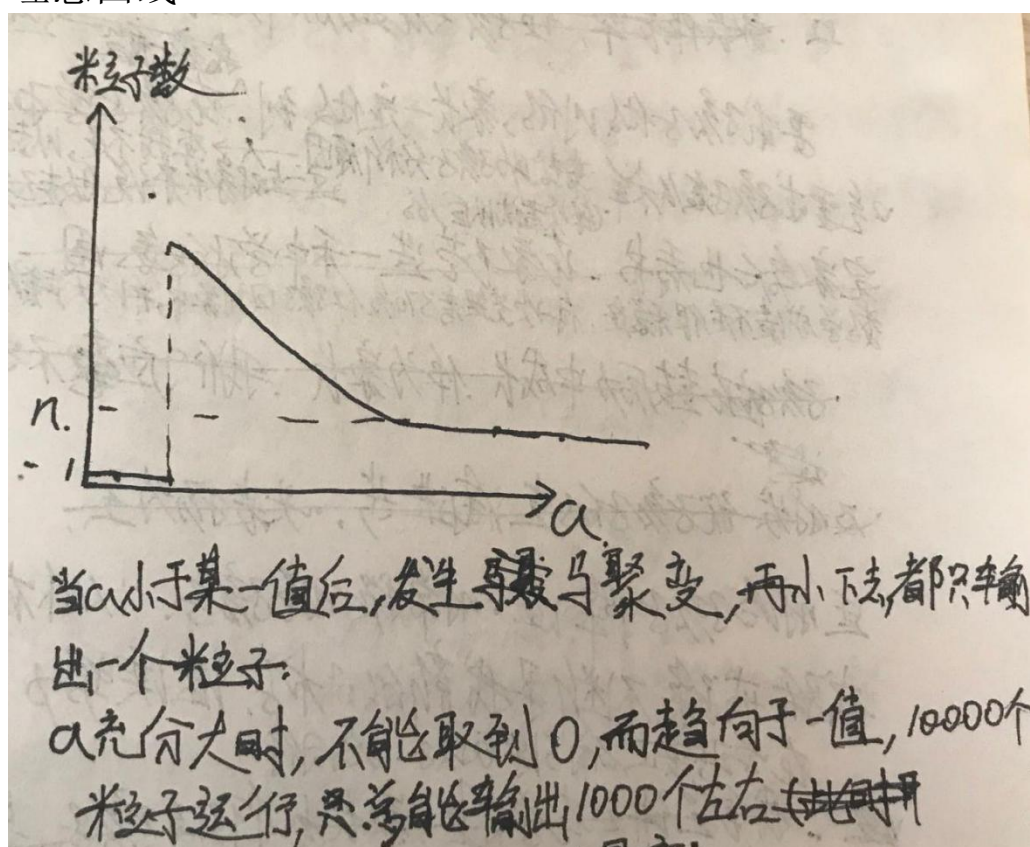
左下角粒子处于稀疏环境，但是，现在版本的代码是对所有粒子进行遍历，如果周围存在密度较大的粒子团，那么这颗粒子经过筛选也会被判定密度较大区域的粒子，所以应加一限制条件：对每一颗粒子，在计算与其他粒子之间距离时，只小于某一个值，如  $d$  之内的粒子参与循环， $d$  可以是位置坐标精度的某一倍数

2. 运行过程中发现对  $a$  值（即判定临界值）进行调整时，出现了如下的函数关系图像（筛选出粒子数与  $a$  值大小关系）（递减关系不一定是线性，随具体情况而定，这里是为了简洁）（ $a$  越大，则对局部密度要求越宽松，反之越严格）





理想曲线



实际曲线

### Plans:

1. 对代码进行进一步完善，即解决上述测试中不太理想的问题（目前引如了一个  $d$  值以后，比如  $d=0.1$ ，那么就只会以某个粒子为中心  $0.1$  为半径的球内遍历循环，可以跑起来，但是只能一部分解决以上问题周边还是会散落很多点
2. 这周和 YYH 进行了多次讨论，最终发现还是需要从 RSL 下载  $z=25$  的数据，目前思路是，需要从下载来的 SNAPSHOT 里面找到气体团，去掉暗物质（`particle_data` 数组里面的 `int Type` 不同值代表不同种类粒子）后孤立地进行 MOND 模拟
3. 目前尚不知道以上 SNAPSHOT 是什么样的，是否需要划分星团的代码，已从网上找到一部分 `kmeans` 的代码，目前的思路是结合 `Kmeans` 代码完成吸积率判据代码，已经开了一点小头