

计算物理 作业报告4

PB14203209 张静宁 2017.10.14

第四题

在一正方形盒子中心，分别取一个圆形区域、正方形区域和六边形区域，布满相同数目的粒子。按 HPP 模型的规则，模拟系统随时间的演化过程，作图显示经过相同演化时间三个图形的差异。

算法思路

HPP模型的建立

选取一个三维数组 $S[2][M+1][N+1]$ 来表示格点的状态，画布的大小是 $(M+1) \times (N+1)$ （本程序中选择 1001×1001 ）， $S[0][i][j]$ 是上一时刻格点 (i, j) 的状态， $S[1][i][j]$ 是下一时刻格点 (i, j) 的状态。

格点的状态一共有16种，分别用0~15的整数值来表示，如 $S[1][i][j] = 8$ ，即 (1000) 表示有一个粒子从上方进入该格子， $S[1][i][j] = 3$ (0011) 表示有两个粒子分别从下方和右方进入该格子， $S[1][i][j] = 0$ (0000)，表示没有粒子进入格子， $S[1][i][j] = 15$ (1111) 表示有四个粒子分别从上下左右四个方向进入该格子。

也就是说，分别用0~15的二进制来表示格点状态，二进制从左到右四个数位分别表示上左下右（逆时针）四个方向是否有粒子进入格点，如果数位上是1就有粒子，是0就没有。

格点状态的更新

格点下一时刻($t+1$)的状态是由相邻四个格子上一时刻(t)的状态决定的，故上一时刻的所有格子状态，决定了下一时刻的所有格子状态。在更新完下一时刻所有格子的状态后，才能将上一时刻的所有格子状态 $S[0][i][j]$ 更新成下一时刻 $S[1][i][j]$ 的状态。

需要设定hpp系统一共运行多少步，程序中设定为STEP=2000步。

格点状态的更新由函数 **hpp_step()** 来实现，故这个函数将被调用2000次。

粒子的碰撞

只对两个粒子头对头的碰撞做特殊的设定，两个粒子头对头在一个格点上碰撞后将垂直于原来运动方向弹开。其他的情况，无论是单独一个粒子的前进，还是三个、四个粒子的碰撞，都沿着原来的方向继续前进。

如果粒子碰到的边界，粒子将做反弹运动。

边界格点的设定

设定 $i \in [1, M - 1], j \in [1, N - 1]$ 为画布内的格子（都可以有粒子），在格子外面还包裹着一圈用于放镜像粒子的格子。也就是当粒子和边界墙壁碰撞时，等效于边界外有一个镜像粒子的运动产生的影响。

初始粒子的摆放

初始粒子区域为正方形时，选择中心 100×100 一共 10000 个格点分别放一个粒子，即一共 10000 个粒子。这些粒子的初始运动方向采用随机设定，也就是说它们可以往上下左右任何一个方向。故这些格点的初始取值为 {8, 4, 2, 1} 中的任意一个。

初始粒子区域为圆形时，需要计算对应的圆形半径，假定圆形和正方形面积相等时，圆形区域内的整数格子和正方形的相同，即近似 10000 个。因此需要定义一个距离函数，来判断格点是否在圆内。之后的操作和正方形相同。

初始粒子区域为正六边形，是最麻烦的情况。需要先计算出区域面积相等时，正六边形的边长。从而用几何条件判断格点是否在正六边形内，如果在就放上一个粒子。我采取的几何判断条件是先把六边形分成四个部分，逐一判断四个区域上面的点和这个区域中的一个单位矢量的内积是否小于一个值，如果是就在六边形内。之后的操作和正方形相同。

实现上，分别定义了函数 **square()**, **circle()**, **hexagon()** 来实现初始条件的设置，在运行程序时应该根据需要只调用其中一个函数，注释掉另外两个函数调用的代码，再编译运行。

位运算

具体如何根据相邻格点此刻的状态来决定该格点下一时刻的状态，采用位运算来实现，先用 s_value 来存储该格点下一时刻的状态，每次初始化 $s_value = 0$

- 对于该格点上方相邻的格子此刻的状态，只要有朝下的箭头（除了头对头的情况 $s = 10$ 即 1010 ），下一时刻都会运动到该格点，即从上方进入该格点。故四位二进制从左到右的第一位取1，即该格点的值 $s_value += 8$
- 对于该格点左方相邻的格子此刻的状态，只要有朝右的箭头（除了头对头的情况 $s = 5$ 即 0101 ），下一时刻都会运动到该格点，即从左方进入该格点。故四位二进制的第二位取1，即该格点的值 $s_value += 4$
- 对于该格点下方相邻的格子此刻的状态，只要有朝上的箭头（除了头对头的情况 $s = 10$ 即 1010 ），下一时刻都会运动到该格点，即从下方进入该格点。故四位二进制的第三位取1，即该格点的值 $s_value += 2$
- 对于该格点右方相邻的格子此刻的状态，只要有朝左的箭头（除了头对头的情况 $s = 5$ 即 0101 ），下一时刻都会运动到该格点，即从右方进入该格点。故四位二进制的第四位取1，即该格点的值 $s_value += 1$

以上每一条规则对应一行位运算表达式。

中心粒子 $s[i, j]$ 的状态取决于上下左右四个格子状态

$\begin{cases} s[i-1, j] & \text{上} \\ s[i, j-1] & \text{左} \\ s[i+1, j] & \text{下} \\ s[i, j+1] & \text{右} \end{cases}$

① 上 $s[i-1, j]$ 若处于 $(a == 5) \parallel ((a >> 3) \& \& a \neq 10)$

\downarrow	$\rightarrow \leftarrow$	$\rightarrow \downarrow \leftarrow$	$\downarrow \leftarrow$	$\rightarrow \uparrow$	$\rightarrow \uparrow \leftarrow$	$\uparrow \leftarrow$	\downarrow	
1000	0101	1101 ↓ 10	1011 ↑ 13	1110 ↑ 11	1111 ↑ 14	1111 ↑ 15	1001 ↑ 9	1100 ↑ 12
8	5							

$s[i, j]$ 第1位是 1，即 $(+1000)_{(2)} = +8$

② 在 $s[i, j-1]$ 若处于 $((a == 10) \parallel (a >> 2) \& \& a \neq 5)$

$\rightarrow \circ$	\downarrow	$\rightarrow \leftarrow$	$\rightarrow \downarrow$	$\rightarrow \leftarrow$	$\rightarrow \downarrow \leftarrow$	$\rightarrow \uparrow$	\uparrow
0100	1010 ↑ 5	1110 ↑ 14	1101 ↑ 13	0111 ↑ 7	1111 ↑ 15	1100 ↑ 12	0110 ↑ 6
4	10	14	13	7	15	12	6

$S[i, j]$ 第2位是1, $(+0100)_{(2)} = +4$

③ 下 $S[i+1, j]$ 若处于 $(a==5) \parallel [(a>>1)&1]$ && $a \neq 10$

$S[i, j]$ 第3位是1, $(+0010)_{(2)} = +2$

④ 右 $S[i, j+1]$ 是 $((a==10) \parallel [a&1]) \&& a \neq 5$)
 $(4+2+4+)$

$S[i, j]$ 第4位是1, $(+0001)_{(2)} = +1$

对于粒子碰壁的情况，将边界分为上下边界、左右边界两种情况，每种情况镜像粒子的状态处理不同：

1. 对于上下边界，边界外的镜像格子的状态与最靠近边界的粒子的状态上下是相反的，也就是二进制从左到右第一位和第三位是互换的
2. 对于左右边界，边界外的镜像格子的状态与最靠近边界的粒子的状态左右是相反的，也就是二进制从左到右第二位和第四位是互换的。

以上两条规则也对应于两行位运算表达式。

程序使用说明

编程环境：Ubuntu，要预装 C 和 Python3，当前目录下有以下文件和result文件夹

- hpp.c C 语言编写的计算程序，负责计算并输出数据点
- hpp 编译后在Linux系统可执行的文件
- dynamic_plot.py Python 编写，实时查看的动态绘图程序
- image_plot.py Python 编写，输出绘制图片
- video_plot.py Python 编写，输出视频的程序（这三个程序需要在linux下运行...因为文件路径不通用）

分别执行以下五条命令：

```
$ gcc hpp.c -o hpp          # 编译 hpp.c 程序
$ ./hpp > data              # 在当前目录下执行 hpp 程序，并将结果输出到 data 文件
$ python3 dynamic_plot.py    # 查看数据效果
$ python3 image_plot.py      # 生成图片
$ python3 video_plot.py      # 生成视频
```

计算结果分析

注：生成的视频、动图放下作业目录里，没插入PDF，助教可以单独看一下。

初始盒子与画布的相对大小会影响模型演化的效果，故选取3种不同边界进行对比。

模型测试

压缩包里有一个叫做**modeltest.mp4**的视频，是用来测试粒子数较少时实现的效果是否满足HPP模型的要求。

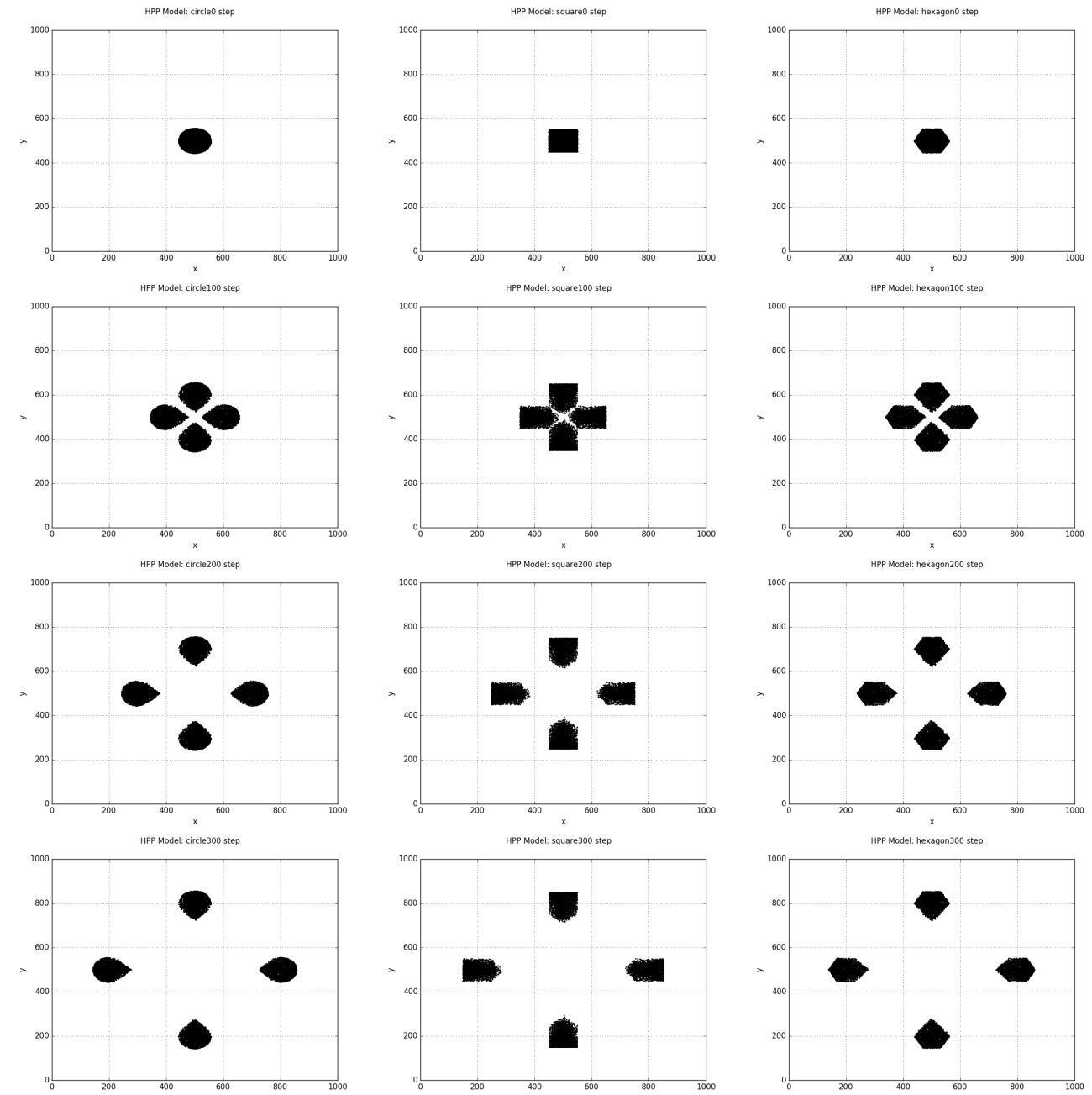
大部分碰撞都符合条件，发现左下方有一次头对头碰撞似乎没有垂直转向，经过分析查看了数据，发现这并不是代码的Bug导致的。这两个粒子并没有真的碰撞，而是分别在相邻的左右格子相向运动，之后交换了格子，没有在格点碰撞。

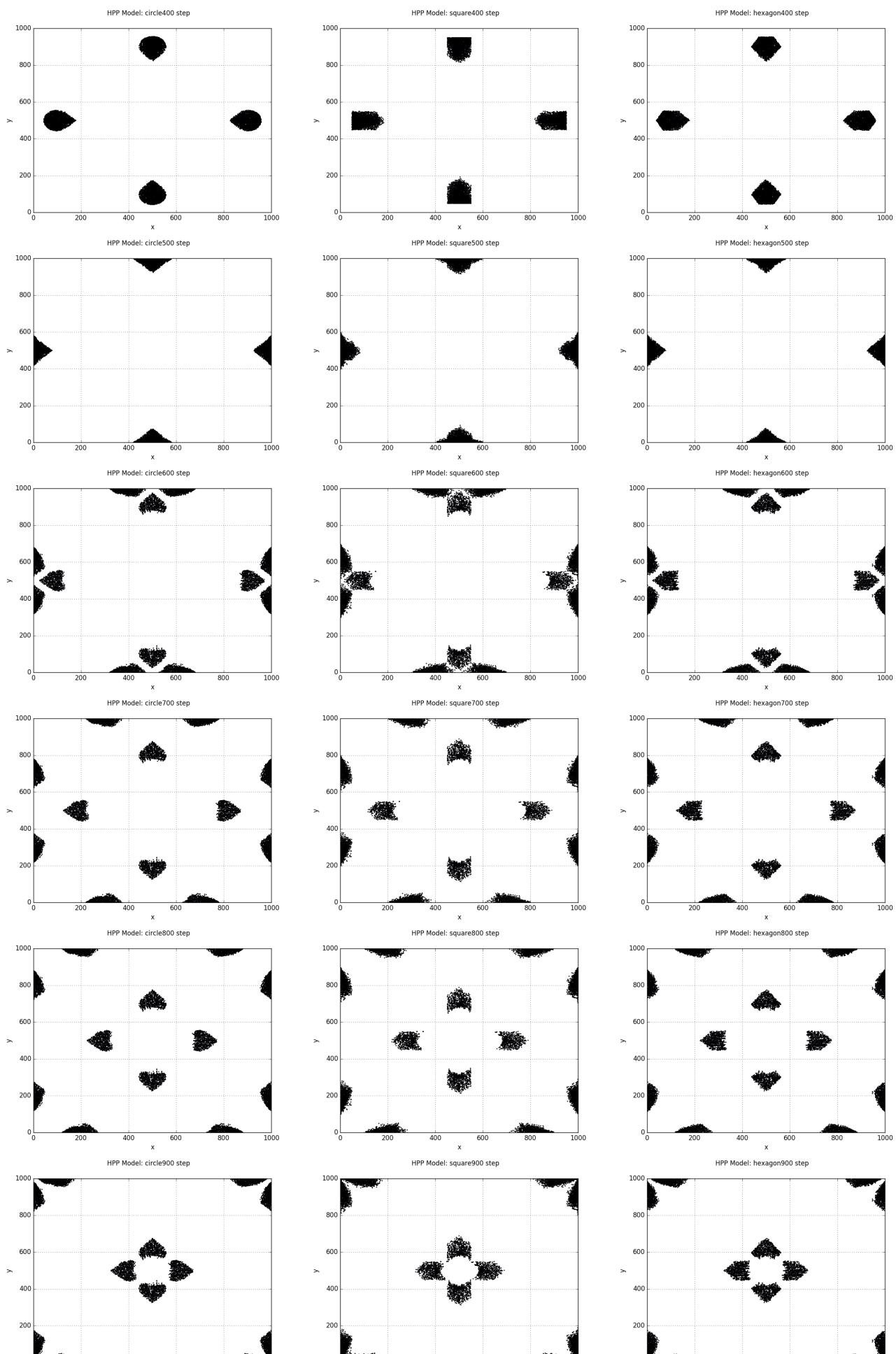
蛮有趣的，助教可以看一下视频。

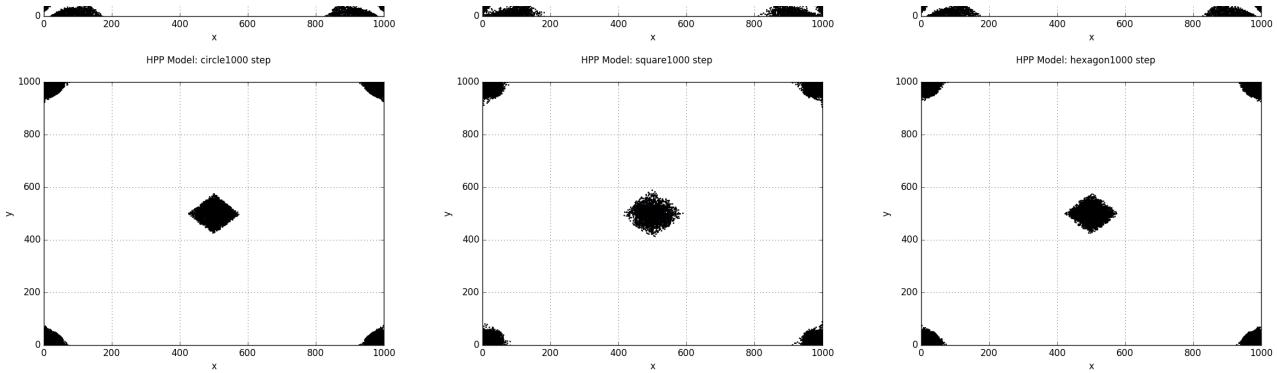
画布是盒子100倍大小

画布大小1000*1000个格点，初始盒子为100*100格点，每个格点一个粒子，共10000个。

下图从左往右分别是圆形、正方形、六边形初始区域的hpp模型演化图示，从上到下一共11幅图，每隔100步画一张图，即显示了三种初始情况下0步、100步、200步、...、1000步的系统粒子状态。



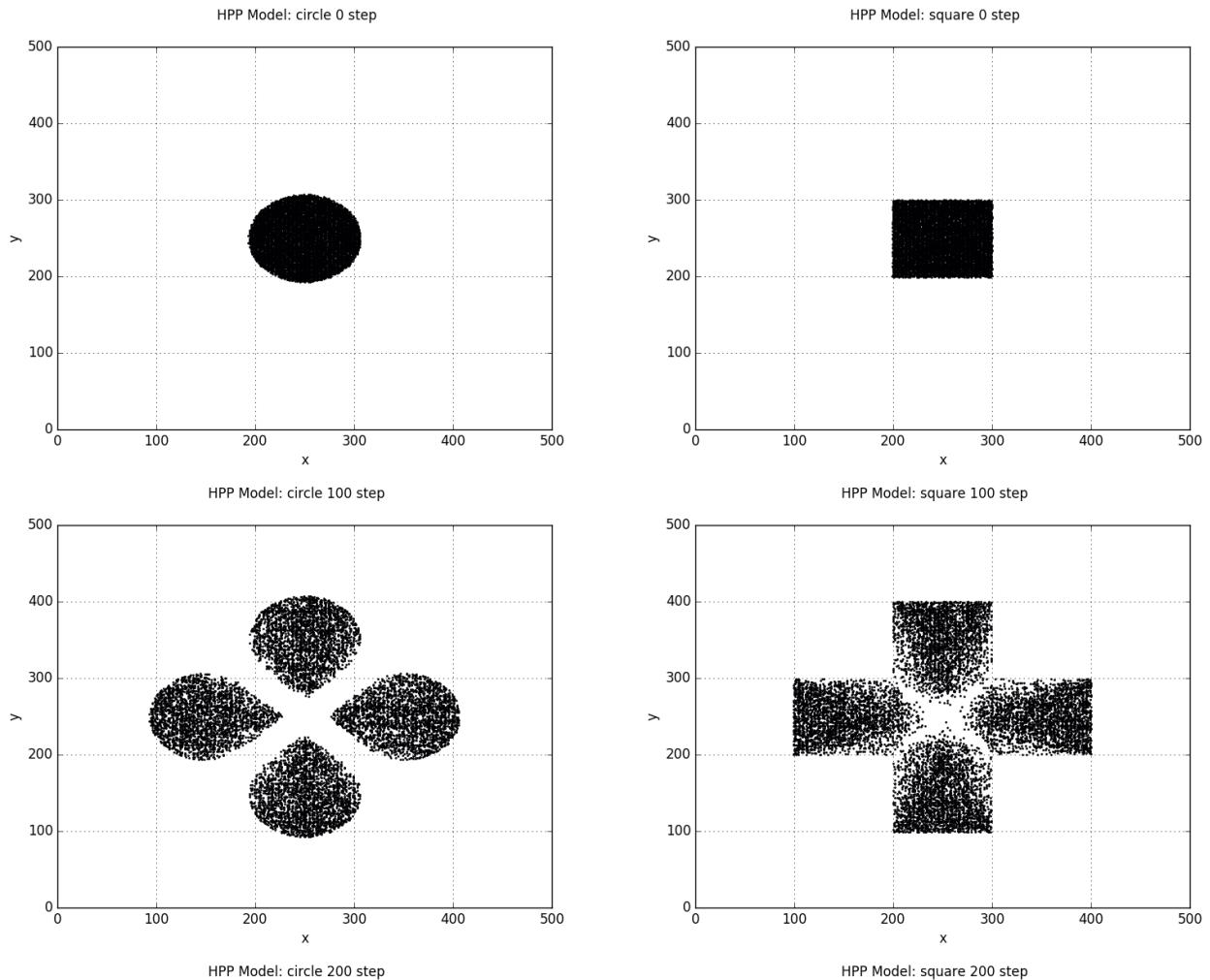


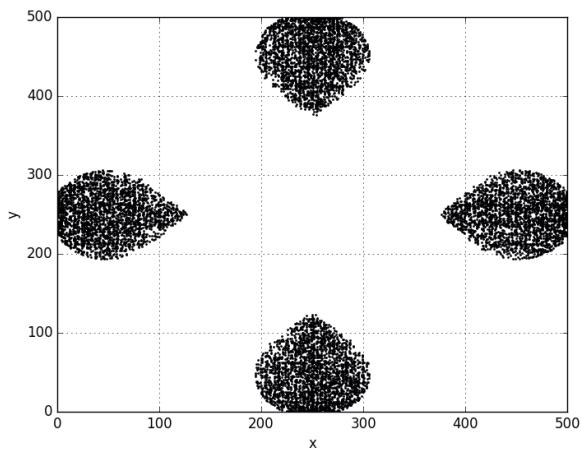


- 粒子群、形状：总的来说，粒子都是一群一群的运动，一开始群体的形状较能反应初始条件，如圆形边界初始条件的粒子群体形状边界较为圆润。
- 各向同性：由于设定粒子只能沿上下左右四个方向运动，系统演变时粒子群体也是沿着上下左右运动。并没有体现出空间的各向同性。
- 对称性：在设定的四个方向上粒子群运动是中心对称的。
- 周期性：系统的演变整体来看像是周期过程，如1000~2000步和1~1000步的过程基本一致，一直画到10000步都是这样平凡的重复。并没有出现粒子越来越混乱，充满整个空间的过程。
- 三种中心区域粒子的演化，在一开始区别比较明显，越到后面区别越小，1000步以上初始条件的区别肉眼不可区分。

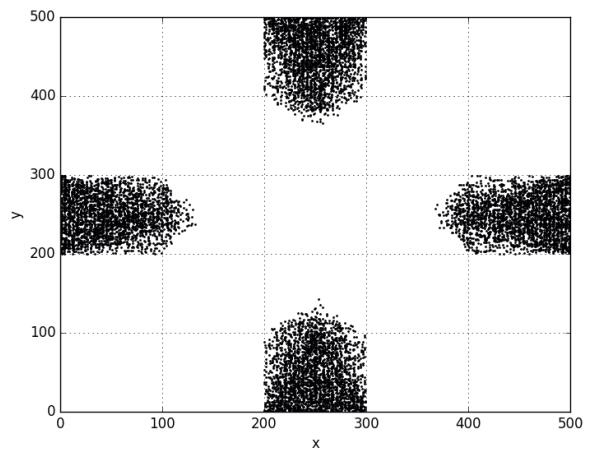
画布是盒子25倍大小

画布大小500*500个格点，初始盒子为100*100格点，每个格点一个粒子，共10000个。

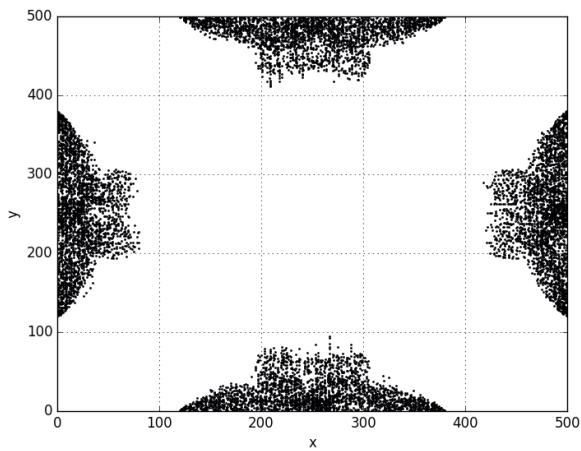




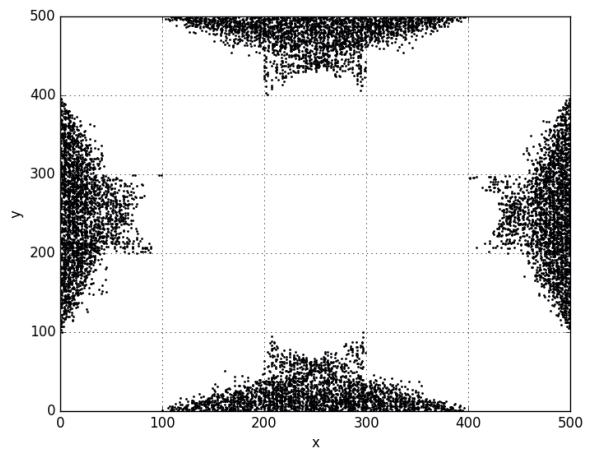
HPP Model: circle 300 step



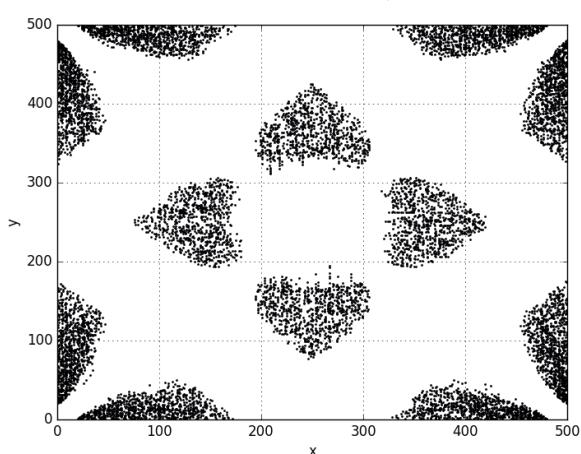
HPP Model: square 300 step



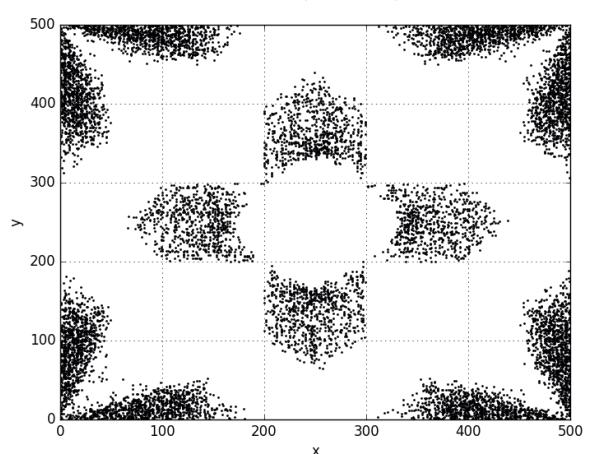
HPP Model: circle 400 step



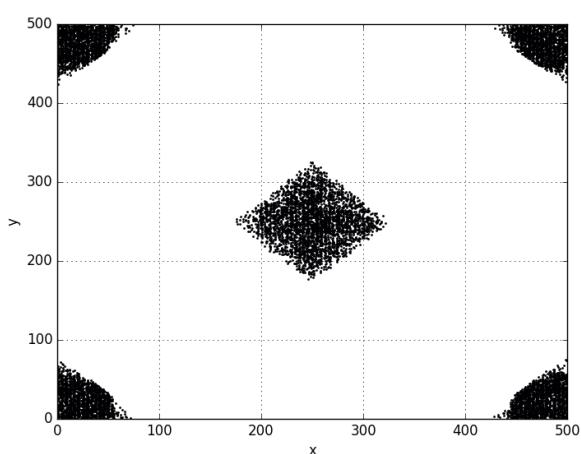
HPP Model: square 400 step



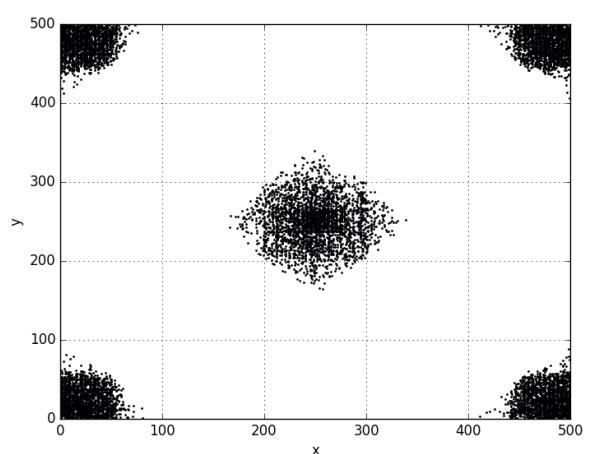
HPP Model: circle 500 step



HPP Model: square 500 step

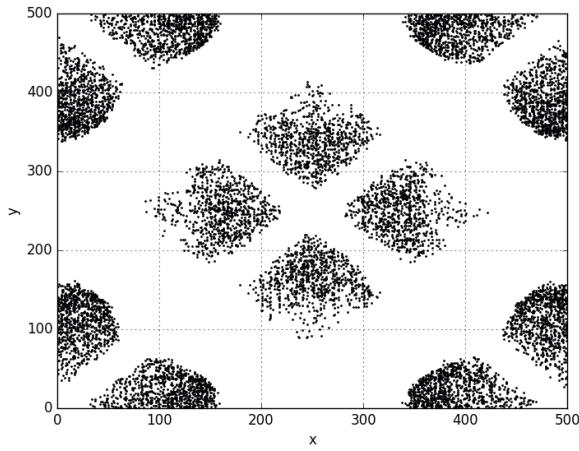


HPP Model: circle 300 step

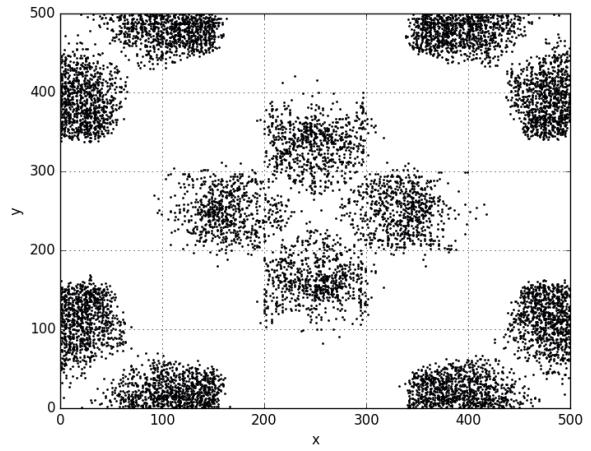


HPP Model: square 300 step

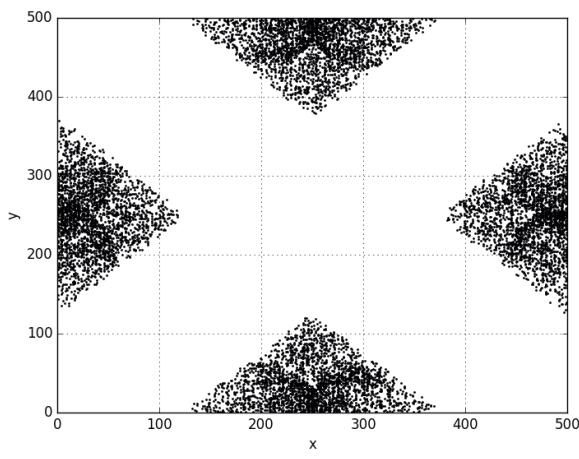
HPP Model: circle 600 step



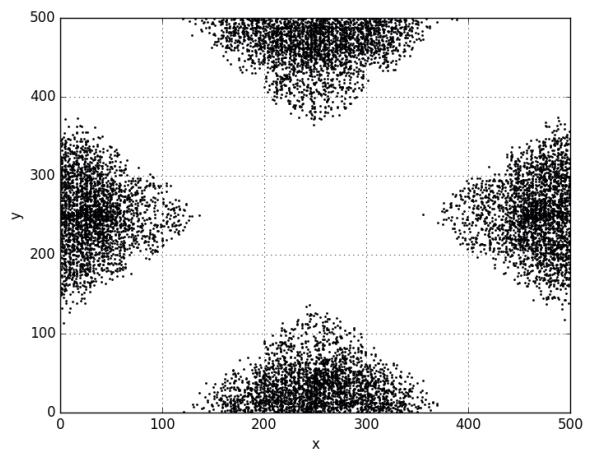
HPP Model: square 600 step



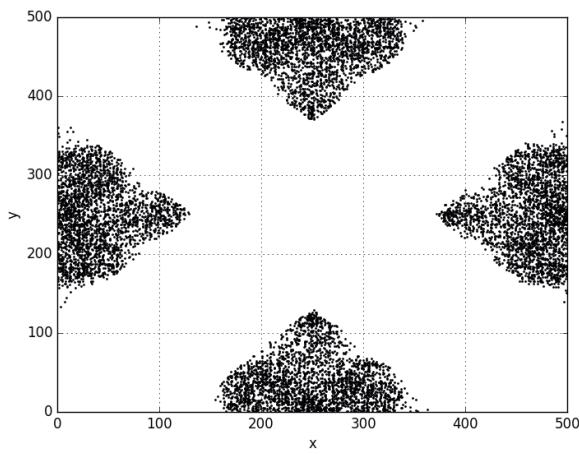
HPP Model: circle 700 step



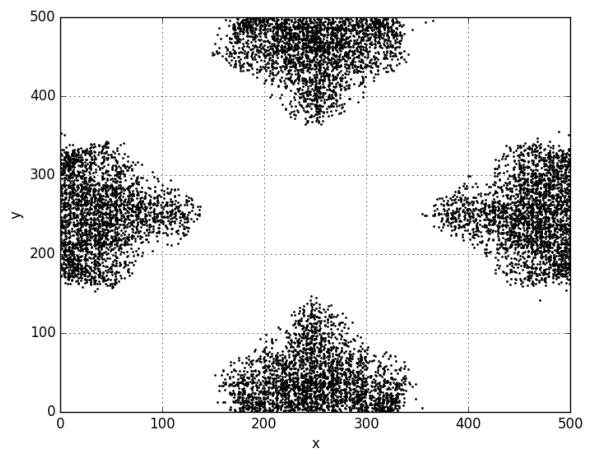
HPP Model: square 700 step



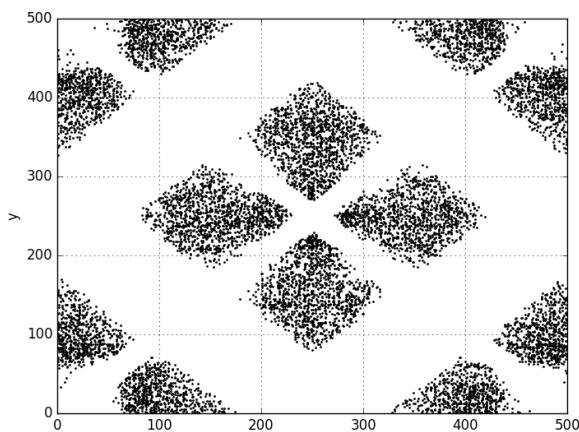
HPP Model: circle 800 step



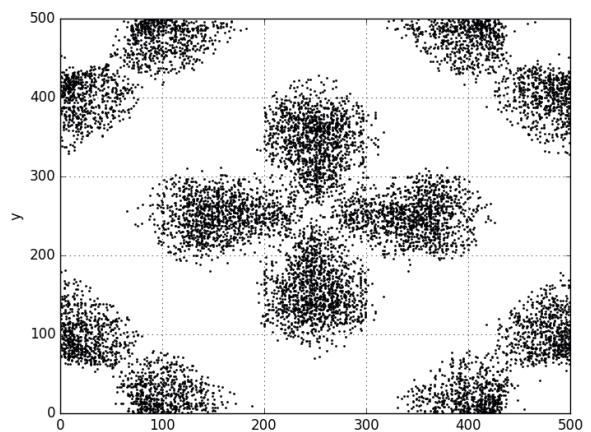
HPP Model: square 800 step

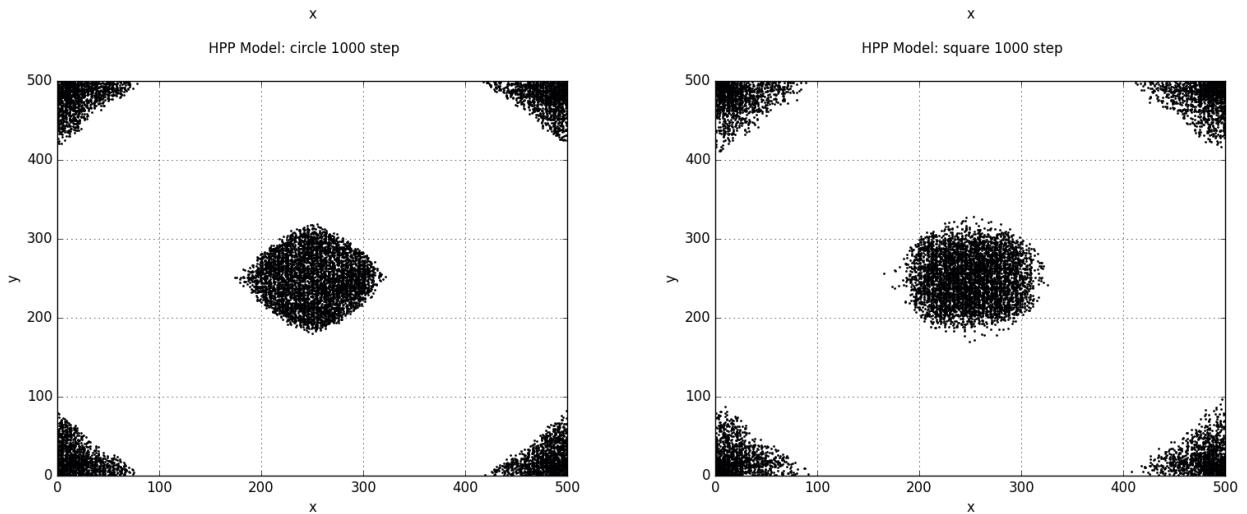


HPP Model: circle 900 step



HPP Model: square 900 step



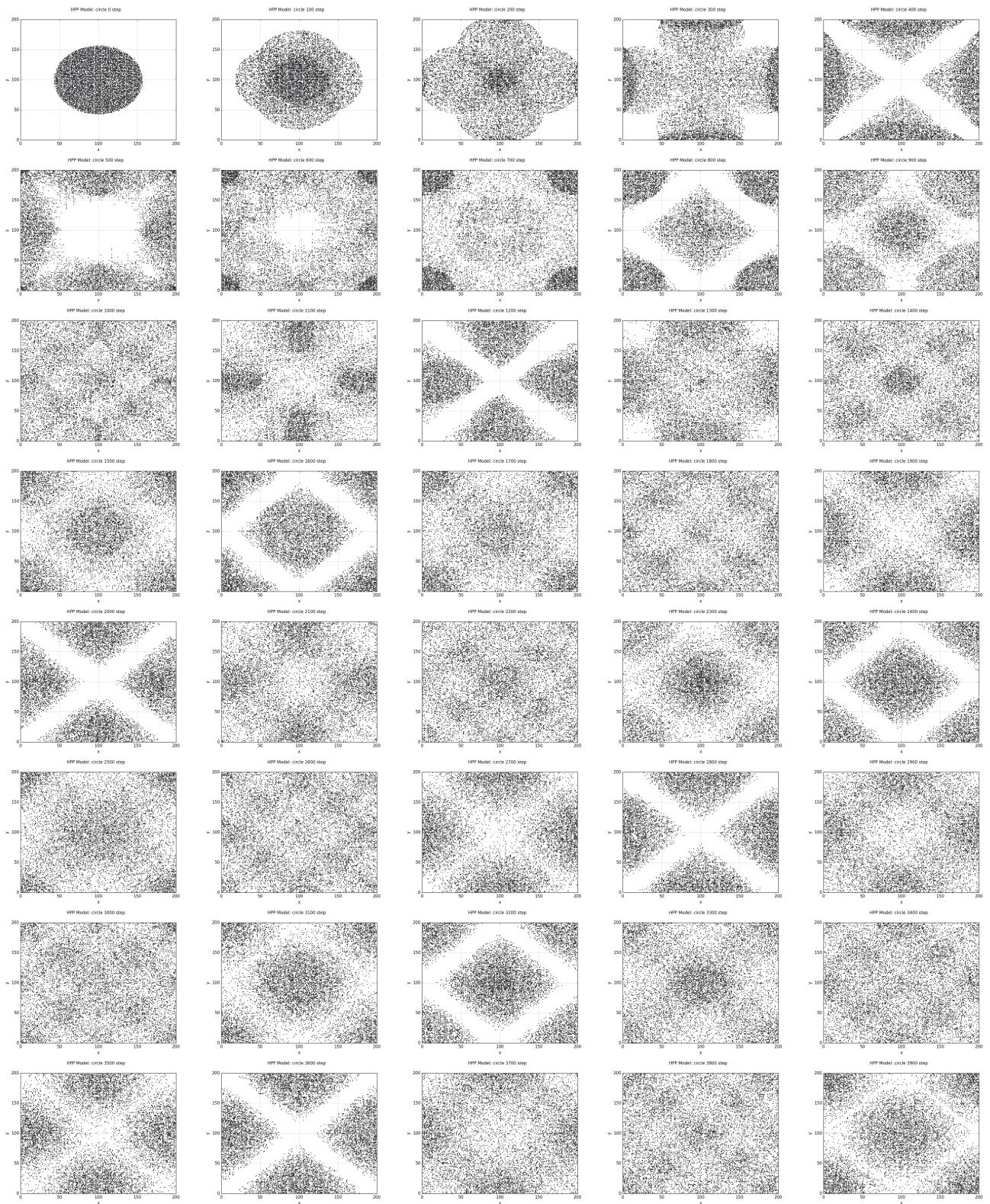


分析：依旧是能够看出周期性，而且经过两个周期，正方形的居然比圆形的粒子还混乱一些。

画布是盒子4倍大小

画布大小 200×200 个格点，初始盒子为 100×100 格点，每个格点一个粒子，共10000个。

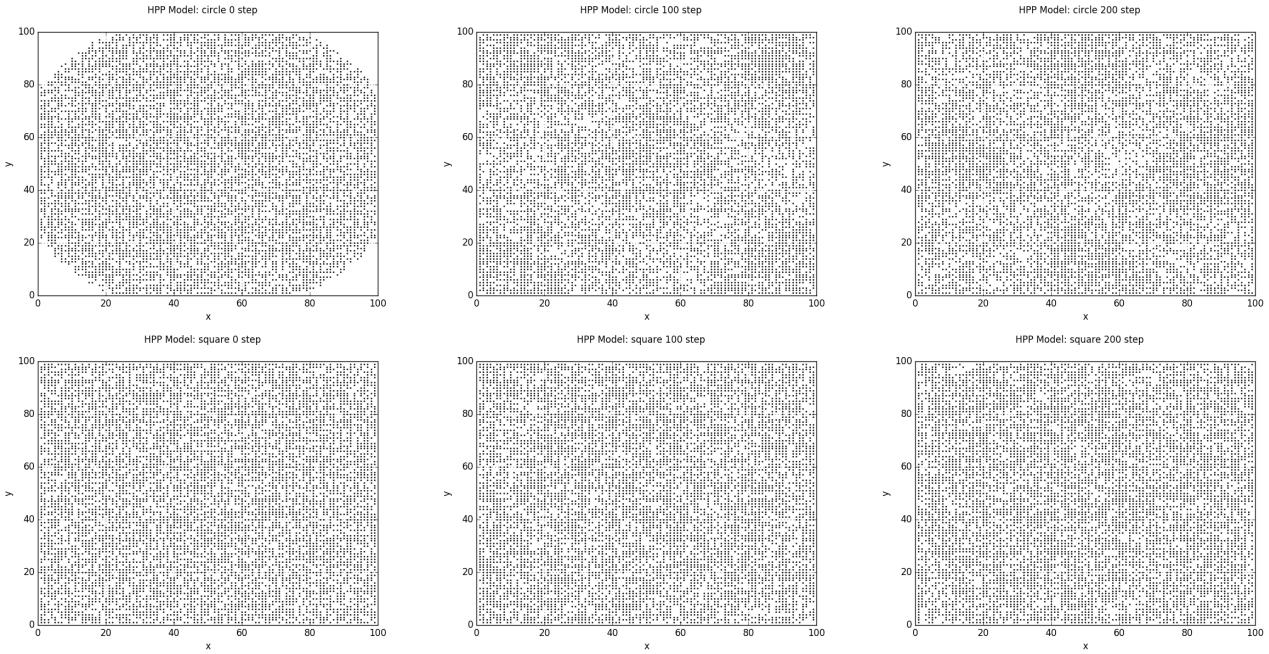
下图之画了圆形盒子的情况，可以看出来经过很长时间，系统还呈现周期性，前后图片有非常相似。



画布和盒子一样大

画布大小100*100个格点，初始盒子为100*100格点，每个格点一个粒子，共10000个。

这种情况图片看不出粒子在运动，通过动图能够感受到粒子在运动，但似乎杂乱无章。



总结

对计算结果的分析总结见上一部分，下面说说完成这次作业的一些收获：

- 在写程序时，圆形和正方形的边界条件都还好处理，正六边形的边界条件处理起来真是比较繁琐。
- 大部分时间花在缕清思路建立模型上面，另外一大部分时间花在写绘图程序上面。当测试粒子只有一个、步数只有10多步时，尚可肉眼看数据判断模拟是否正确，当测试粒子增加、步数增加，则必须要借助绘图结果才能判断程序是否有错误。可见在做科研的过程中，数据可视化的重要性，必须借助数据可视化才能解读出计算结果，看到系统的性质，提取所需的信息。