08：元胞自动机实现Ising Model

许传奇 PB16021546

1 题目

在512×512的二维正方形网格上，设置几个初始状态的自旋比例值，按照Q2规则Ising自旋动力学模型，模拟体系自旋状态随时间的演化，作图比较体系初始状态和平衡状态自旋的分布。

2 原理与算法

2.1 原理

2.1.1 元胞自动机

1. 定义：

   元胞自动机是定义在一个由具有**离散、有限状态**的元胞组成的元胞空间上，并按照一定**局部规则**，在**离散的时间维**上演化的动力学系统。

2. 原理：

   当元胞自动机中的某个元胞的邻居满足某种特定的条件时，该元胞状态按特定的条件发生改变。因此，(T+1)时刻整个元胞空间由T时刻的元胞空间与特定的条件和变化规则决定。

3. 元胞自动机中的边界条件：

   在理论上，元胞空间通常是在各维方向上是无限延展的，但是在实际应用过程中，无法在计算机上实现这一理想条件，因此，需要定义不同的边界条件，边界条件主要有三种类型：

   (a) 周期型：相对边界连接起来的元胞空间。

   (b) 反射型：边界外邻居的元胞状态是以边界为轴的镜面反射。

   (c) 定值型：所有边界外的元胞均取某一固定常量，如0，1等。

4. 二维元胞自动机常用的邻居定义：

   如下图所示：

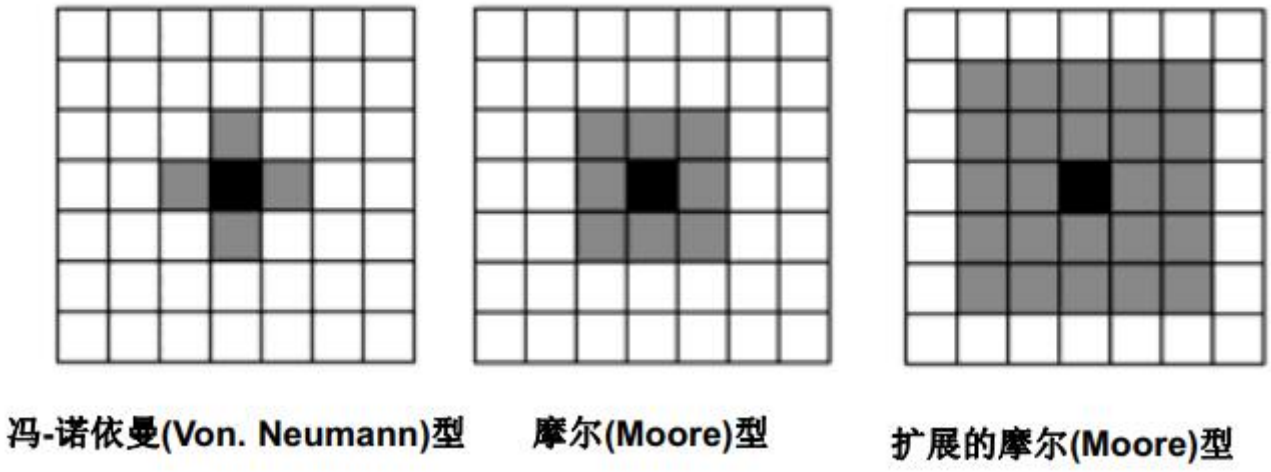

图 1: 二维元胞自动机常用的邻居类型

2.1.2 Ising自旋动力学模型的Q2规则

对于Ising自选动力学模型的模拟，我们可以采用元胞自动机。元胞自动机的相关设置和原理如下：

1. 网格：

   设为二维正方网格，每个格点有自旋$S = 0, 1$。

2. 邻居：

   Von Neumann型，如上图1中的第一个图所示。

3. 物理依据：

   自旋对为相同自旋时排列能量较低(设为-J)，相反自旋时排列能量较高(设为J)，自旋翻转时要求局部没有任何能量交换。

4. 演化规则：

   要求保持局部能量守恒，即只有当邻居中自旋向上和自旋向下的邻居数相同时，自旋才会翻转。

2.2 算法

2.2.1 算法简述

由于Ising Model中自旋状态改变是建立在邻居不变化的假定上的，如果一个接一个对自旋进行翻转的判定，那么会违背这个假设，也就会违背物理上要求的局部无任何能量交换的前提，即违背了能量守恒的条件。

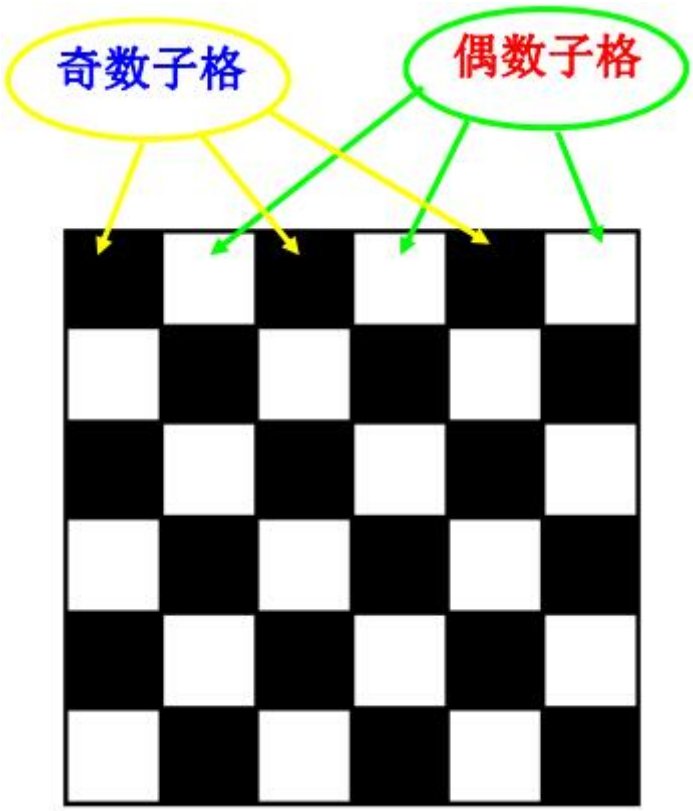

图 2: 奇数子格与偶数子格的划分

由于邻居类型是Von Neumann型，我们可以通过选择奇数子格与偶数子格并分别对其翻转来实现，如上图2所示。

因为Von Neumann型的邻居类型中，奇数子格中的格点的邻居都是偶数子格中的格点，同样的，偶数子格中的格点的邻居也都是奇数子格中的格点，所以对奇数子格和偶数子格分别进行翻转时，翻转过程中的邻居状态不会发生改变，也就满足了我们的假定和物理前提。

因此，本次作业中的翻转算法就是此算法，步骤如下：

1. 把正方形网格划分为奇数子格和偶数子格。
2. 基于偶数子格作为邻居，对奇数子格上的格点进行翻转；再基于奇数子格作为邻居，对偶数子格上的格点进行翻转。

2.2.2 源程序函数简述

源程序中边界条件设置为周期条件。

1. void InitSpins(char Spins[LENGTH][LENGTH], int start, double ratio)：

   由16807随机数产生器产生的随机数初始化正方形网格。

   Spins为保存自旋状态的二维数组，start为16807种子值，ratio为起始时自旋向上的比例。

   遍历二维数组，每次遍历都根据16807随机数产生器产生一个在[0,1]内的随机数，根据比例ratio(ratio∈[0,1])，当随机数小于ratio时，初始化该位置的自旋状态为0，否则为1。

2. int WhetherFlip(char Spins[LENGTH][LENGTH], int x, int y)：

   判断(x,y)处自旋是否要翻转，如果要反转，则返回1，否则返回0。边界条件为周期条件。

   函数内部根据周期条件设置(x,y)的邻居坐标，即若(x,y)处于正方形四条边中的一条边上，则上、下、左、右邻居中的一个或两个根据周期条件设成另一边的元素。

   再进行判断：当邻居中为0的个数与为1的个数相等时，表面(x,y)处的自旋状态可以翻转，函数返回1；否则返回0。

3. void Flip(char Spins[LENGTH][LENGTH])：

   对网格按Q2规则进行一次自旋翻转。

   按照之前所说的原理，将网格分为奇数子格和偶数子格，然后分别遍历两个子格，用WhetherFlip判断是否需要翻转，需要翻转则将二维数组元素改变为1-S(S为当前自旋状态，为0或1)；否则不进行操作。

   遍历的过程中设置flag变量，遍历二维数组中的两个循环中的第一个循环设置操作flag=flag-1，第二个循环的起始值设为flag，通过设置起始的flag来对奇数子格或者偶数子格进行选择，这种方法同样也会保证遍历的是奇数子格或偶数子格（遍历的元素中没有两个元素相邻）。

4. double Calculate(char Spins[LENGTH][LENGTH])：

   统计自旋向上的粒子所占的比例。

   遍历二维网格，当当前元素为0时，计数加一。遍历完毕后，返回(计数/256^2)，即为自旋向上的粒子数所占的比例。

3 源文件使用说明

编译并运行“08Cellular_Automata.cpp”，将弹出命令行，要求输入起始时自旋向上的比例ratio与总翻转数num。

输入后，程序运行，实时打印出目前翻转次数时自旋向上所占比例。

达到输入的次数后，将起始的网格元素、起始时自旋向上所占的比例、最终次的网格元素、最终次自旋向上所占比例数据输出到文件“num=输入的num.txt”中。同时打印运行时间。

编译并运行“plot.py”即可绘制起始与最终次的网格图样，其中自旋向上的为白色区域，自旋向下的为黑色或红色区域，或者与这个规律相反。

4 计算结果及具体分析

4.1 翻转次数：1000000，起始自旋向上比例为0.08

设置翻转次数为1000000，得到如下图所示的比例结果：

图 3: 翻转1000000次后自旋向上的比例

由于数量巨大，耗时非常之长，达到两个多小时，经过观察，发现自旋向上的比例在0.35-0.45之间波动，甚至达到0.30，即使是运行了两个多小时的程序，也没有达到一个十分稳定的状态。不过这个均值约为0.4。

对应图片如下：

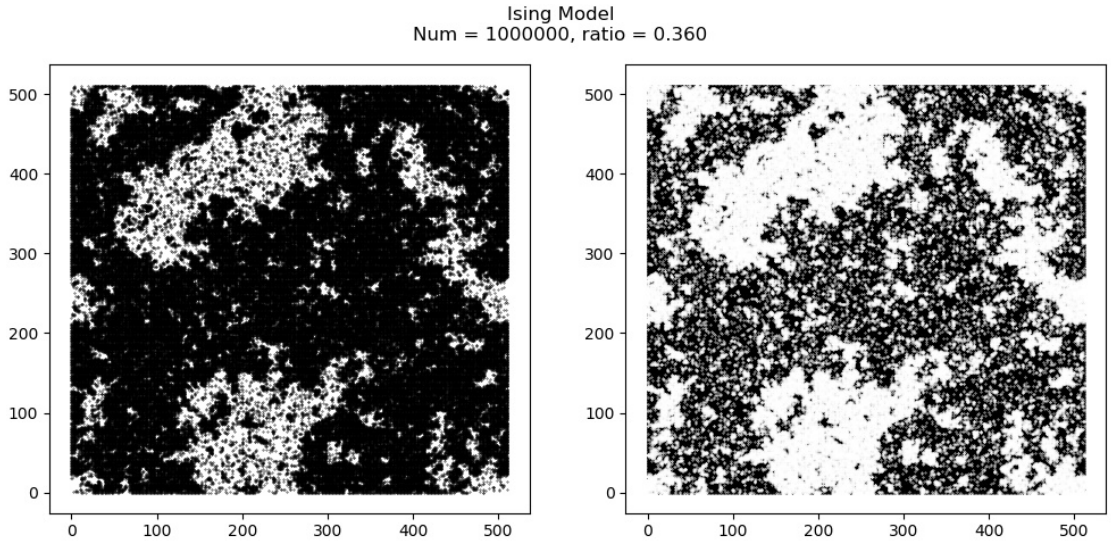

图 4: 翻转1000000次后的结果

上图中，左图为先画白点再画黑点的结果，右图为先画黑点再画白点的图，由于图片显示的分辨率的问题，画点的顺序会对图片的结果有影响，因为后画的点会覆盖住前面的点。

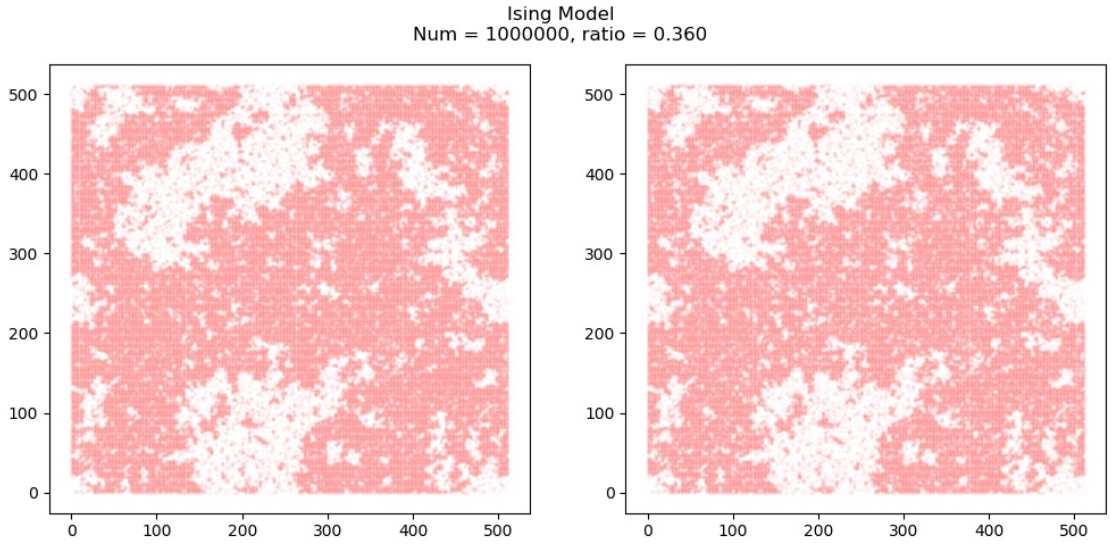

图 5: 翻转1000000次后的结果

如果我们把点设置得非常小，以至于两个像素点之间没有太多的重叠，则画点的顺序影响不大，如上图所示，但对比度会显著降低。因此我们将颜色设置为红色来稍微增强一点对比度。

4.2 其他起始比例

为了防止画点顺序的影响，以下的图片把点的大小设置的较小，牺牲对比度来获得比较准确的图样，并将颜色设置成红色，增强对比度。

为节省时间，以下的翻转次数都设置为10000次，一次程序运行耗时约为90秒。

由于设置的自旋起始值较小，我们把颜色设置成与之前相反的结果，即红色代表自旋向上，白色代表自旋向下。

当起始值为0.05时，最终值几乎不变，仍在0.05左右，如下图所示。如果把翻转次数设置的更大，可能会继续增加。

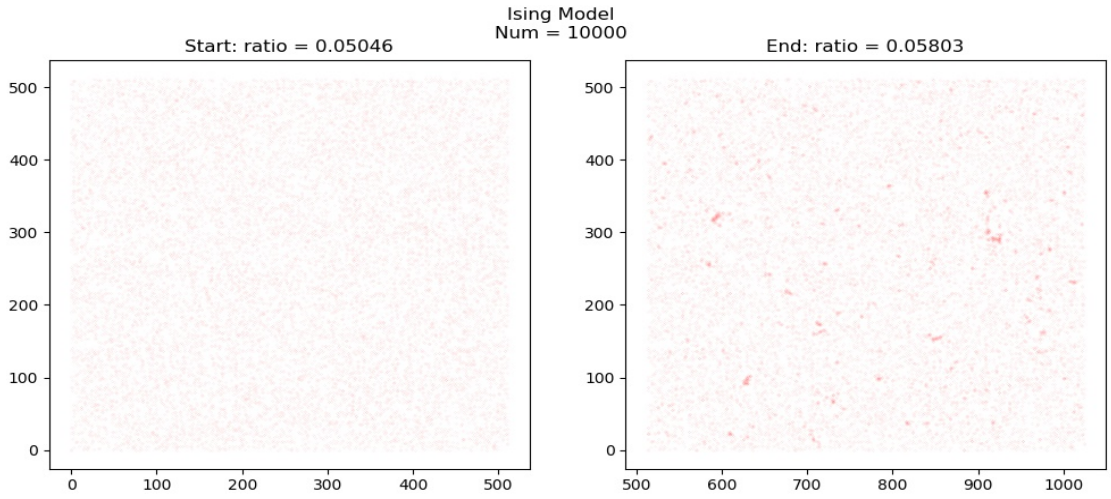

图 6: 起始比例为0.05046

当起始值为0.08时，最终值会变大，即变大的速度会比0.05时快。设置翻转次数为10000次时，得到的最终结果0.23276，还有上升的空间。我们前面讨论的，当起始值为0.08时，运行两个多小时后的结果在0.4附近波动。

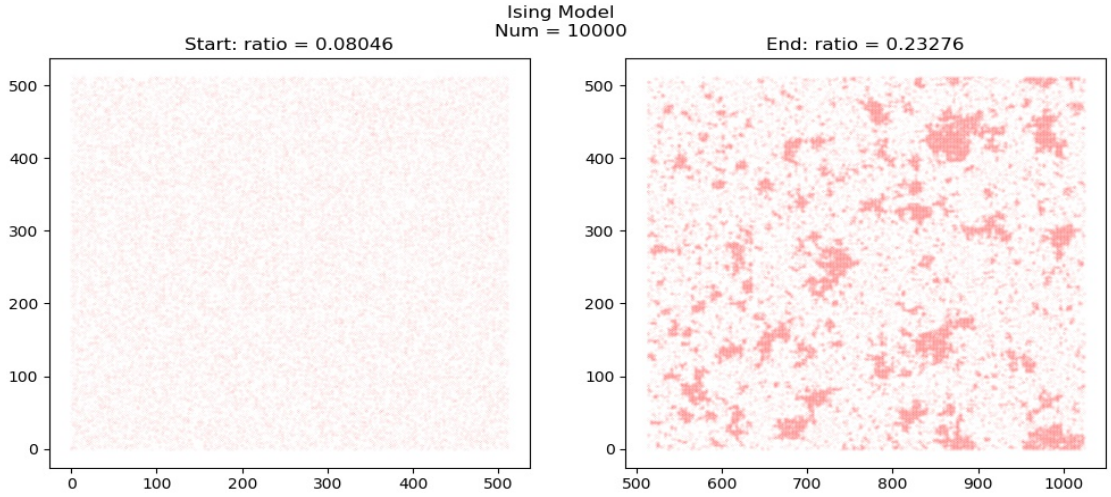

图 7: 起始比例为0.08046

当为起始值0.10时，最终值在0.43到0.48之间，如下图所示：

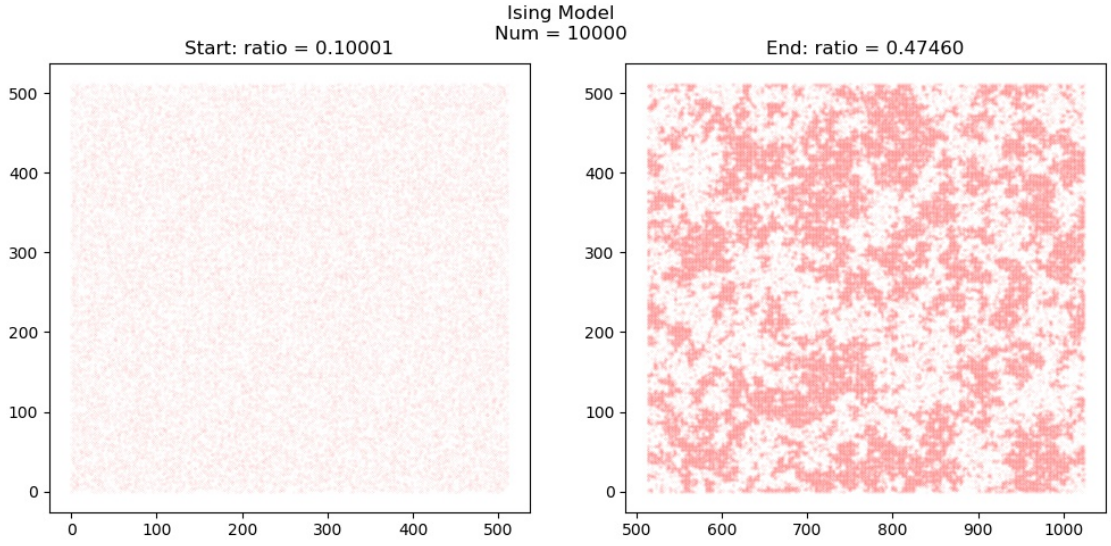

图 8: 起始比例为0.10001

当为起始值0.12时，最终值约为0.5，如下图所示：

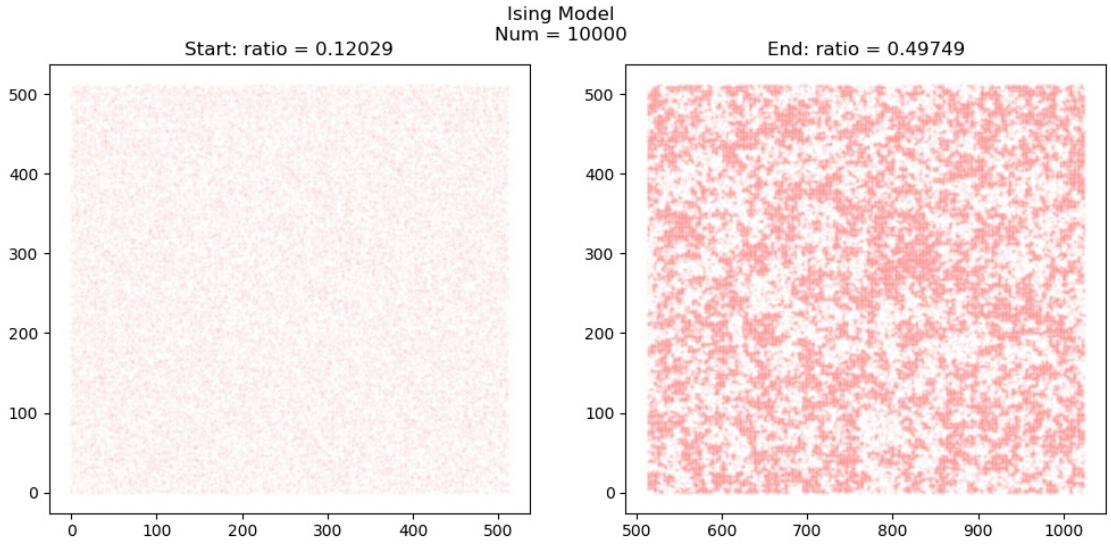

图 9: 起始比例为0.12029

由于自旋状态向上和向下在本题中地位是相等的，因此当起始比例关于0.5对称时，得到的结果应该也是关于0.5对称的。

设置起始比例为0.95，如下图所示，验证了我们的结果：

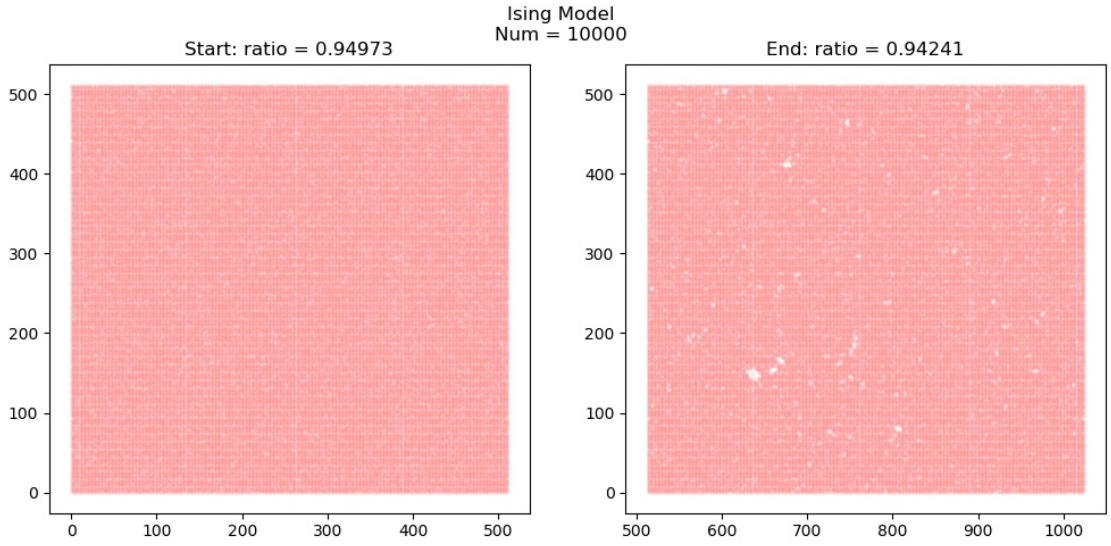

图 10: 起始比例为0.94973

该结果与起始值为0.05对应的结果应该类似。只不过比例应该是(1-0.05对应的比例)。

5 讨论

5.1 原理的讨论

本次作业采用元胞自动机来实验Ising Model。原理部分最重要的是要满足物理前提，因此不能依次对网格元素进行翻转，而要分为奇数子格和偶数子格分别进行翻转。这是本次作业最难的原理部分，其余部分直接按照要求写出代码即可。

5.2 算法的改进

由于本人的计算机知识并不是很丰富，下面的讨论可能会欠妥，源代码的编写也肯定会有很多需要改进的地方，希望以后学习中能够不断完善。

1. 由于本次作业为了能够直观地观察自旋状态，采用实时打印出当前翻转对应的自旋向上的比例的方法。这样会导致计算效率变慢，这部分代码可以注释掉。

2. 在源代码中也设置了把每次结果都输出的代码，可以通过这些数据进行动态图的绘制，不过数据量将很大，绘图效率也将很慢。可通过OpenCV中的函数进行实时观察图样，不需要保存数据然后再读取这个中间过程，可以减小空间和时间复杂度。

3. 本次作业把点设置的非常小来防止点与点之间的遮盖现象。但实际上有一个非常容易的办法，就是把点设置成正方形，然后设置512×512个正方形小格，让每个小格填充不同颜色的正方形即可。但由于像素点的分辨率和正方形大小的原因，我尝试了很长时间，也没有调整python绘图成上述的情况。主要是边长512过大，就会导致每个点的大小变得很小，而点的形状为正方形的散点图的点的大小达不到这么小。但上述的叙述可以适用于边长较小的模型。

4. 本次作业中，物质的磁性用自旋向上与自旋向下之间的相对比例来表示。对于Ising Model，另一种做法是采用类似于小磁针的箭头来表示，这样磁矩将会带有方向，能够更加清楚地展现物质磁性的性质。如下图所示[1]：

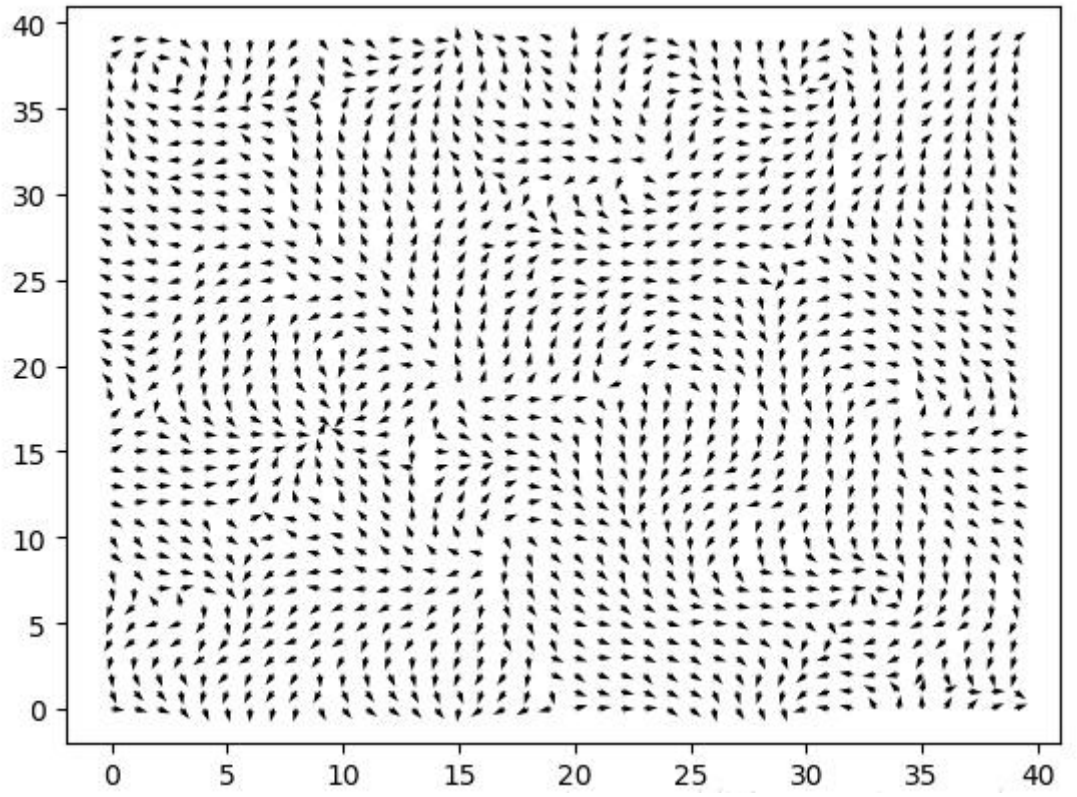

图 11: 小磁针表示的Ising Model

5.3 结果的讨论

1. 结果中可以看出，存在一个临界的起始值η_c，当起始自旋比例$\eta \in [\min(\eta_c, 1-\eta_c), \max(\eta_c, 1-\eta_c)]$时，演化最终结果会在0.5附近，即自旋向上与自旋向下的比例基本相同，可以看做零磁化状态。当起始自旋比不在上述范围内，自旋向上和自旋向下的比例会有明显的差别，可以认为表现出宏观磁化的现象。

   由本次作业的模拟与上述推断可以看出，电子自旋间的相互作用对系统的磁性起着十分重要的作用。

2. 一般而言，对于η小于0.5时，当起始比例η增大时，系统比例增大的速度会变快。

3. 本次作业中，自旋向上与自旋向下地位相同，因此上述讨论中起始自旋比例会有关于0.5对称的规律。之前结果展示中也直接验证了这个观点。

4. 可以看到，因为起始状态我们使用后16807随机数生成器初始化，因此起始时自旋向上和自旋向下的状态分布十分均匀。经过不停地翻转后，分布将会改变，最直观的描述是自旋相同的粒子会“聚集成团”。总体上分布显得均匀，但局部的分布就十分不均匀了。即便是自旋向上和自旋向下几乎相等的0.5的比例条件时，也只是总体上两者相等而已，考察局部的话，不同的区域会有很大差别。

5. 本次作业使用周期型边界条件。如果使用不同的边界条件，会对结果产生影响，如下图所示：

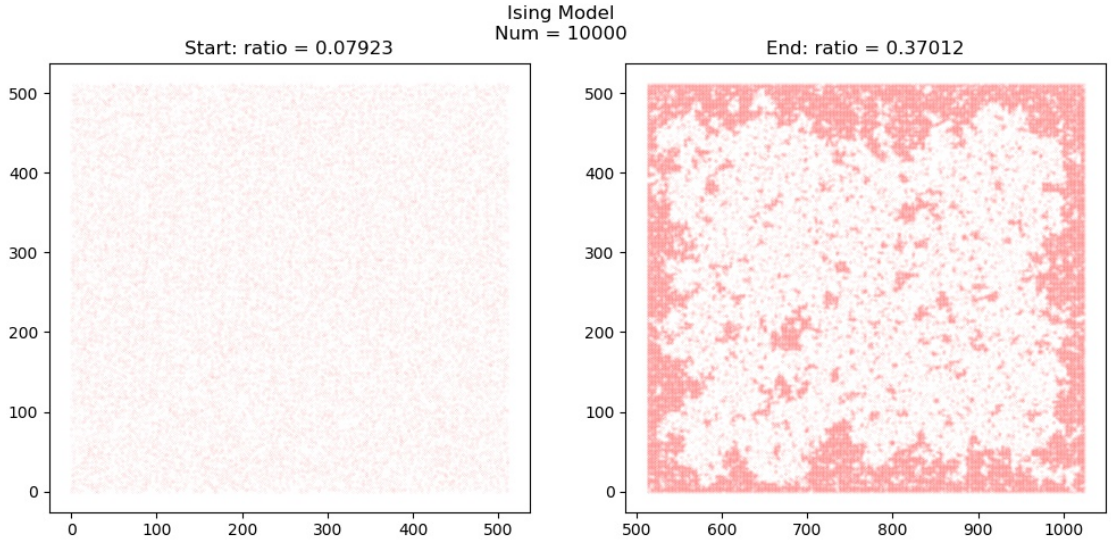

图 12: 定值型边界条件（定值为0），起始比例为0.07923

可以看出，与图7的起始值为0.08046，采用周期型边界条件的结果相比，定值型边界条件（定值为0）的结果的边界上会聚集很多自旋向上（即值为0）的自旋状态，而且同样是翻转10000次，该结果下的比例上升更快，为0.37012，而之前的为0.23276。

6. Ising Model用于物理上则可以解释许多问题，最主要的一点是铁磁物质的相变。当温度超过某个临界值时，随机涨落干扰很强，会导致系统内部状态剧烈改变，导致系统更加无序，从而使物质失去磁性。运用Ising Model的原理和其他原理与手段，例如统计物理中的Maxwell–Boltzmann Distribution、Partition function和计算物理中的Marcov Chain和Metropolis–Hastings algorithm等，可以研究给定温度下系统的能量与磁化状态。这个课题可以作为本题的延伸。

   同时，近年来机器学习的手段也被采用在Ising Model的原理中。已经有相关研究根据机器学习来预测Ising Model中的变分自由能等物理量、推导Gibbs-Bogoliubov-Feynman Inequation等[2]。

   不过对于Ising Model的研究也有局限性，经过与同学的讨论得知，目前3D Ising Model没有一个比较好的解，主要面临的困境是非常巨大的时间和空间复杂度，目前运算水平难以承受这种压力。

参考文献

[1] qq_281617953. 蒙特卡洛模拟ising模型. https://blog.csdn.net/tortelee/article/details/79646049, 3 2018.

[2] Jaan Altosaar. How does physics connect to machine learning? https://jaan.io/how-does-physics-connect-machine-learning/, 8 2017.