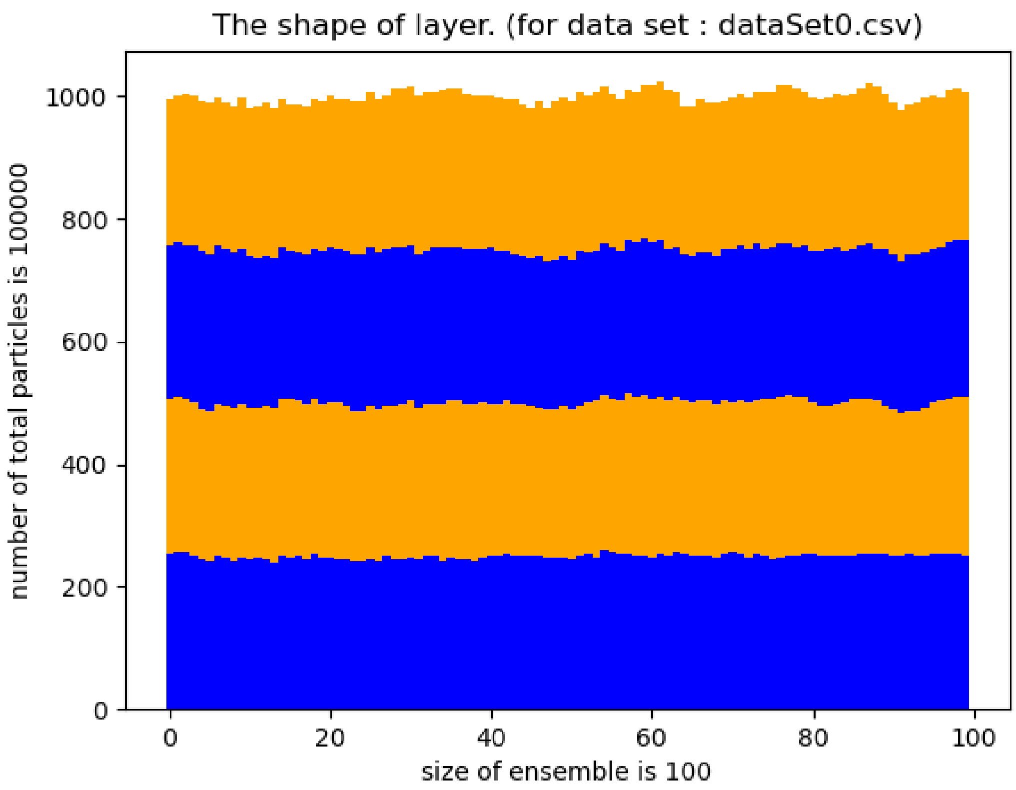
گزارش تمرین سوم

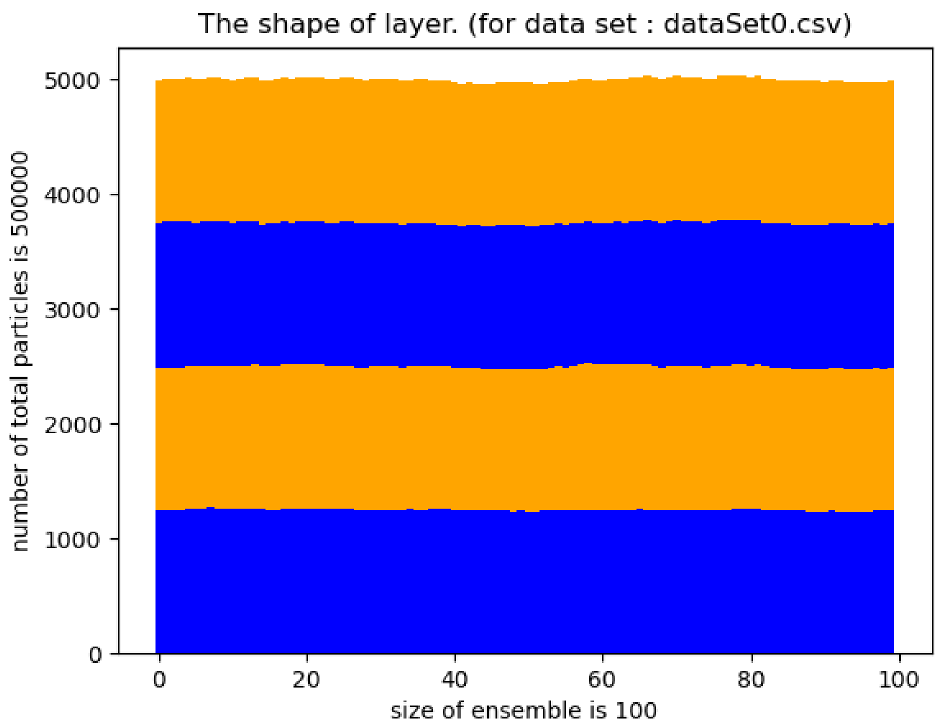
نام: محمد جمشیدی

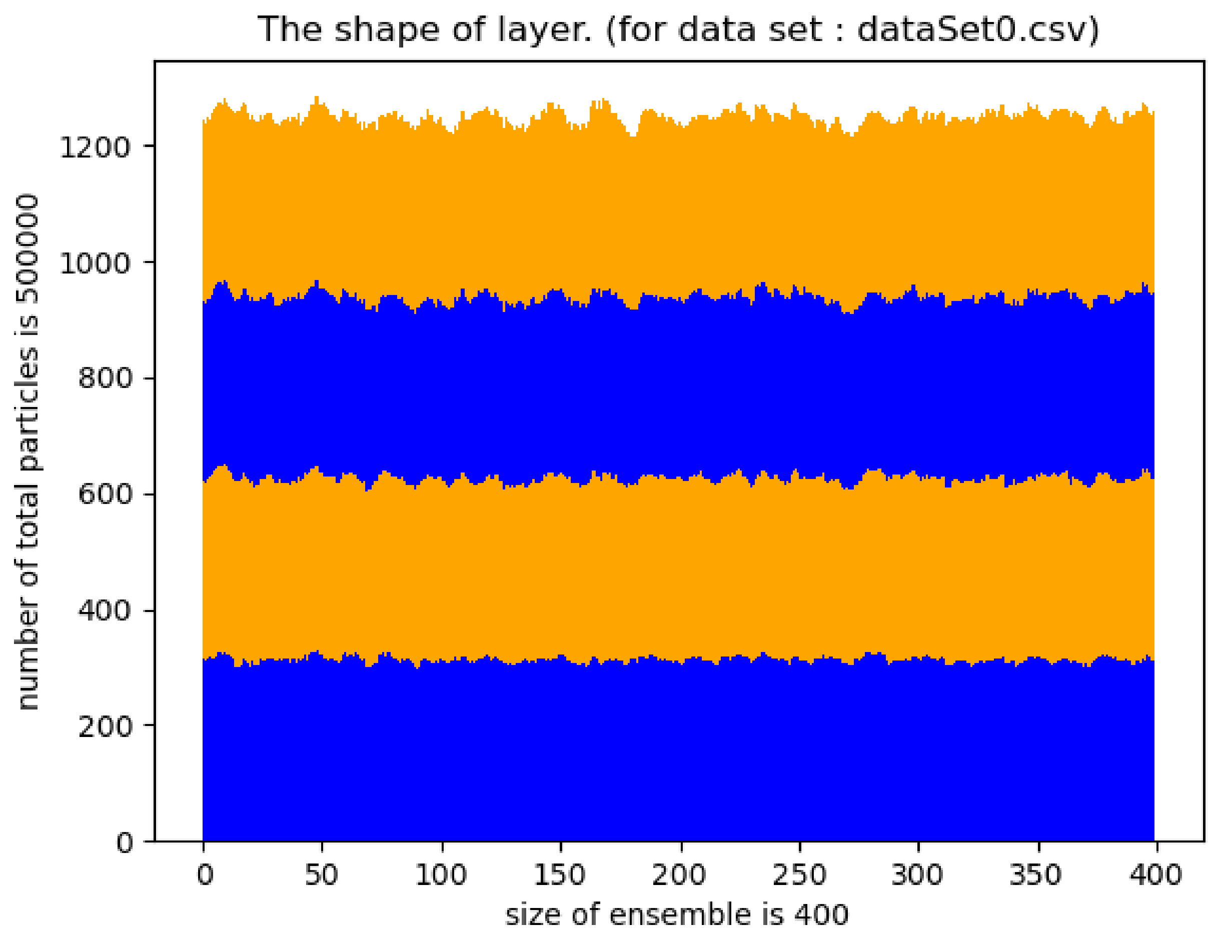
شماره دانشجویی: ۹۸۱۰۰۷۱۸

۱. مدل پایین نشست در یک بعد

در مدل پایین‌نشست در یک بعد و با شرایط مرزی تناوبی، الگوریتم دینامیک به این صورت است که در هر گام یک خانه از شبکه به صورت تصادفی انتخاب می‌شود و سپس ذره‌ی جدیدی که در آن مکان به شبکه اضافه می‌شود، به پایین‌ترین ارتفاع میان همان خانه و خانه‌های چپ و راست می‌رود. چون شرایط مرزی تناوبی است ابهامی در مورد خانه‌های ابتدا و انتهای لتیس نخواهیم داشت. یعنی اگر خانه‌ای که انتخاب شده است ارتفاع کم‌تری در مقایسه با خانه‌های چپ و راست خود داشته باشد که ذره در همان‌جا خواهد نشست. اگر ارتفاع خانه‌های چپ و راست با هم برابر و از ارتفاع خانه‌ی میانی کوچک‌تر باشد در این صورت ذره به طور تصادفی در یکی از آن دو خانه قرار می‌گیرد. در ادامه به کمک تابع showLayer() که درون تابع ballisticDeposition() تعریف شده است تصویر لایه‌ها را تولید می‌کنیم. در ادامه سه نمونه از لایه‌ای که از این مدل نشانده می‌شود را آورده‌ام.







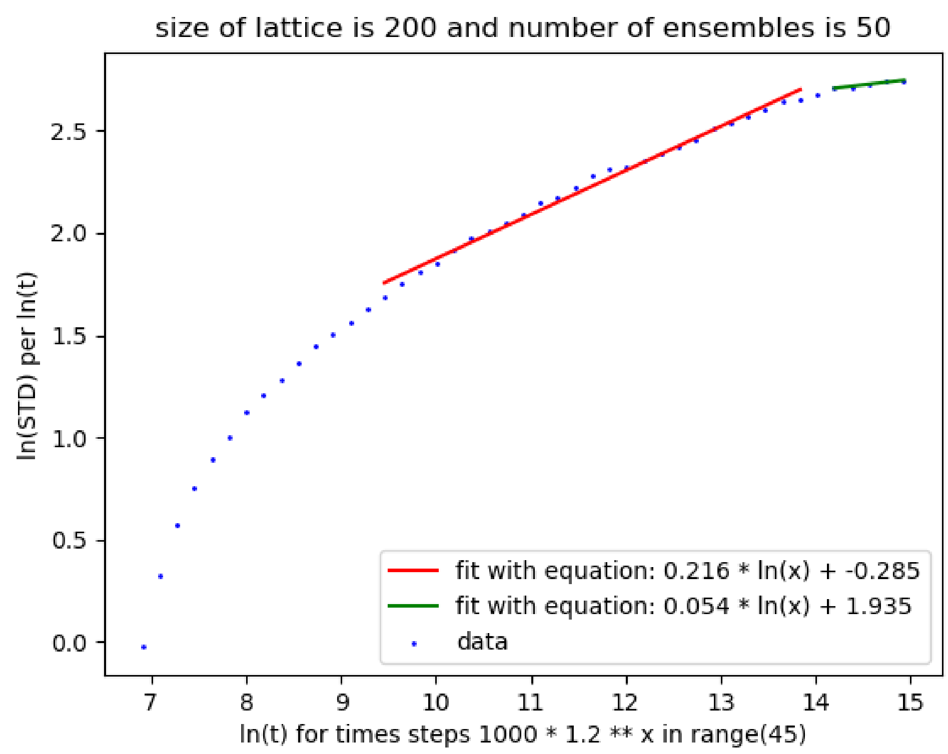
البته در محور افقی شکل‌های بالا نوشته size of ensemble که منظورم همان اندازه‌ی لتیس است (در کد مربوط به سوال این موضوع را تصحیح کرده‌ام و size of lattice نوشته‌ام). همان‌طور که می‌بینیم با افزایش تعداد ذرات ناهمواری رشد می‌کند ولی از یک حدی به بعد دیگر رشد نمی‌کند (این را در نتایج عددی که در ادامه می‌آورم بهتر می‌بینید). همچنین اگر اندازه‌ی لتیس را زیاد کنیم ناهمواری سطح افزایش می‌یابد و دیرتر به حد اشباع خود می‌رسد.

در ادامه نتایج عددی را ارائه می‌کنم. فقط یک نکته‌ای که باید یادآوری شود این است که زمان اجرای کدهای این بخش بسیار بالا بود. به عنوان نمونه برای گام‌های زمانی به صورت

1000 * 1.2 ^ x for x in range(45)

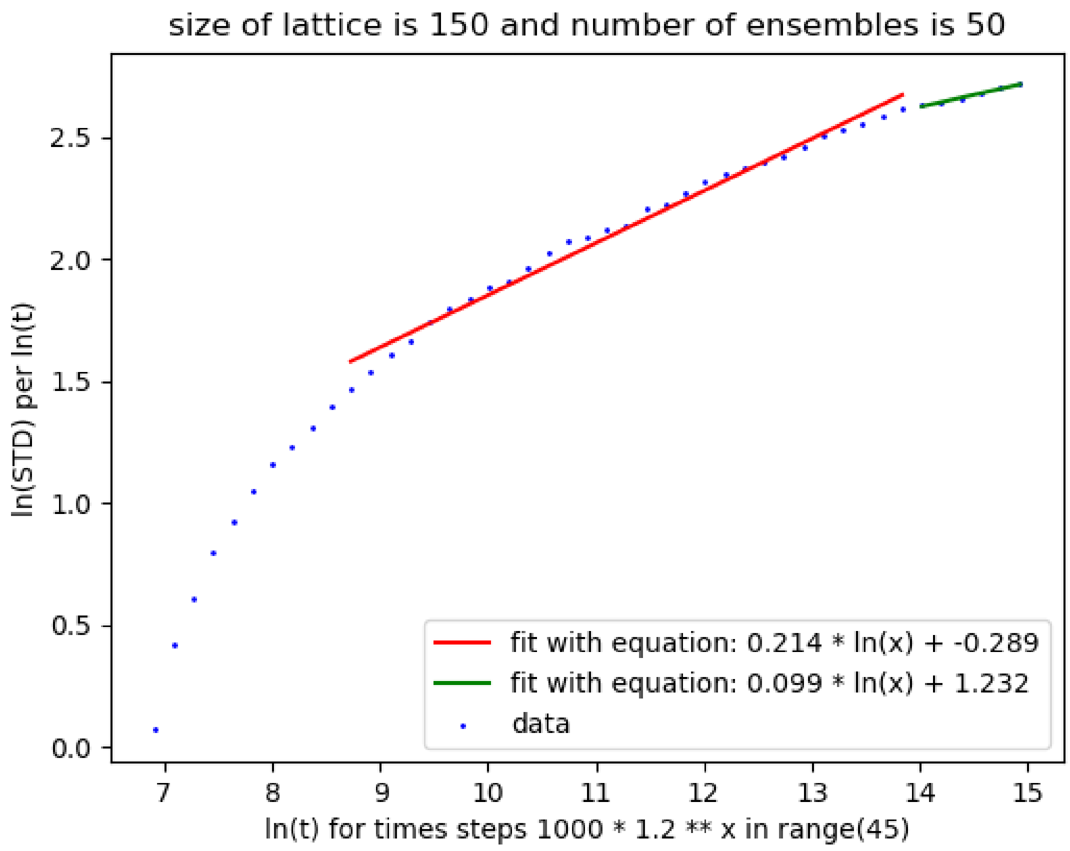
و برای تعداد ۵۰ آنزامبل، زمان اجرا حدود ۲ ساعت بود!

به همین خاطر به ناچار برای داده‌های مربوط به وضعیت اشباع که مربوط به زمان‌های بزرگ‌تر است مقدار خطای بیش‌تری شاهد هستیم. به همین خاطر هم برای فیت کردن بهترین خط مربوط به این داده‌ها، تعداد داده‌ی مناسب کمی داشتم و شیب خطها هم دقیقا صفر نمی‌شد!



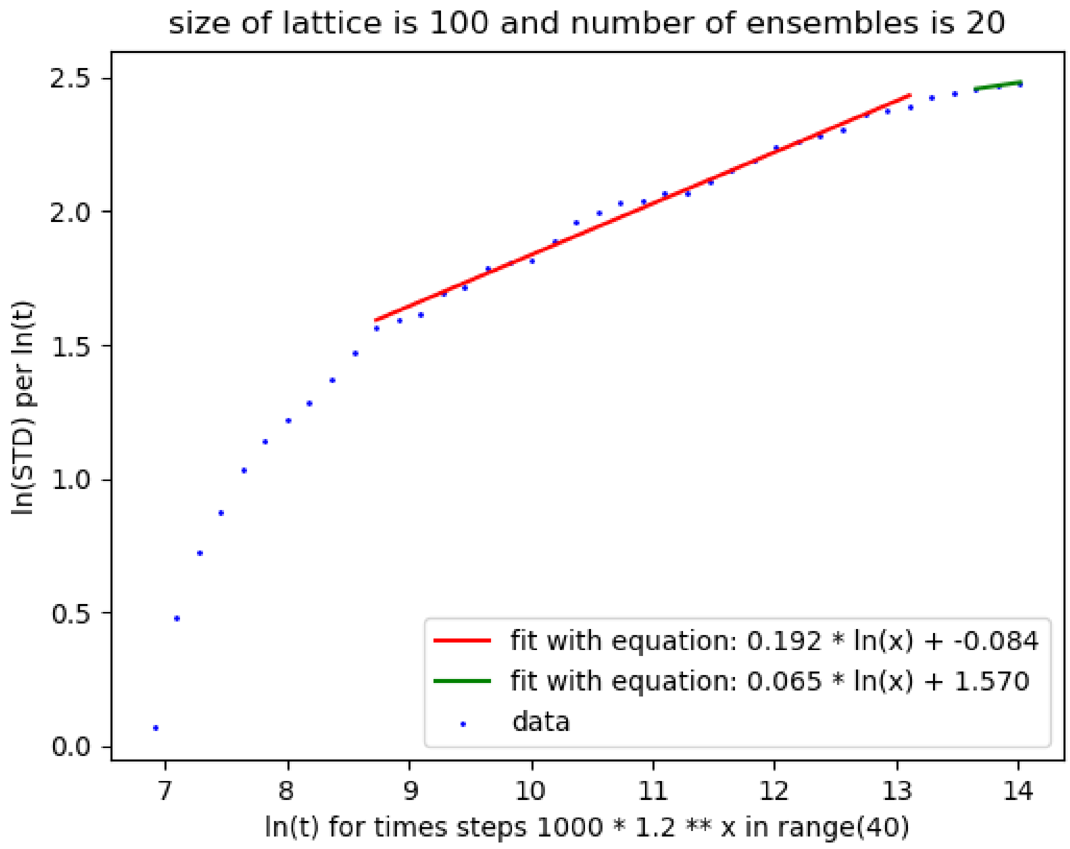
با توجه به نمودار بالا:

$$\beta = 0.216\,, \ln(w_S) = 1.935$$



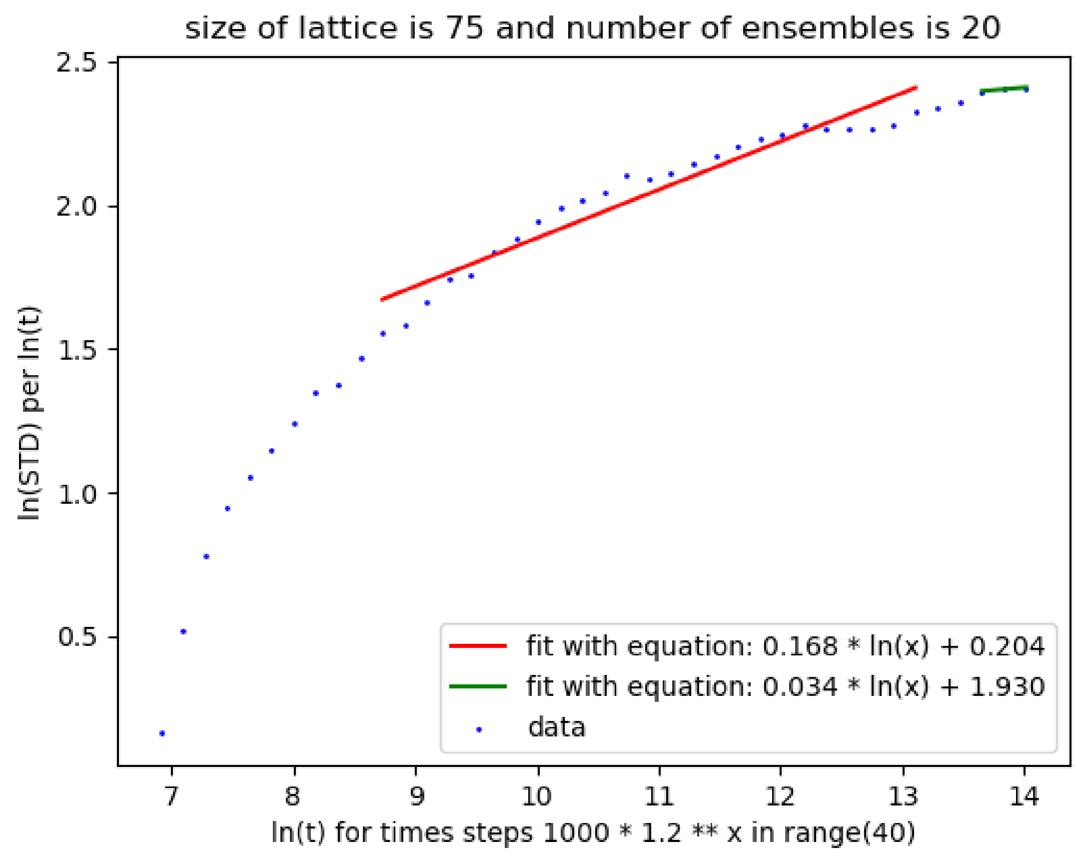
با توجه به نمودار بالا:

$$\beta = 0.214, \ln(w_S) = 1.232$$



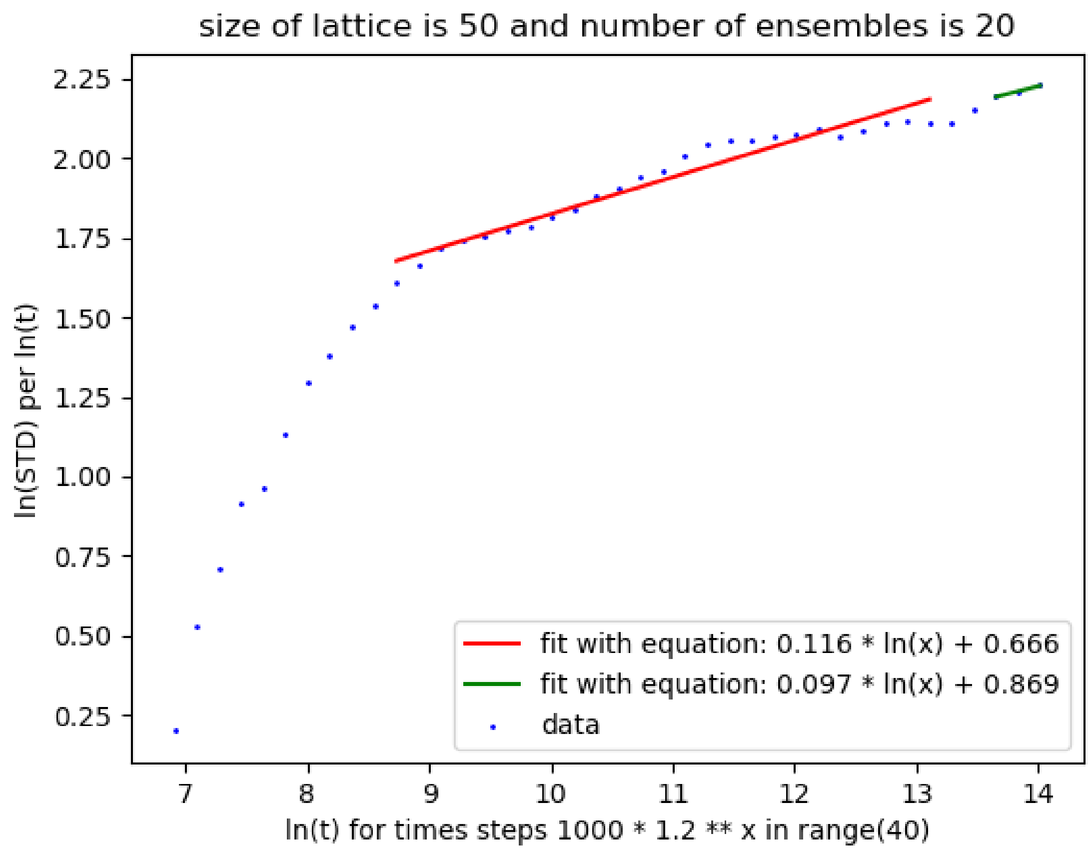
با توجه به نمودار بالا:

$$\beta = 0.192, \ln(w_s) = 1.570$$



با توجه به نمودار بالا:

$$\beta = 0.168, \ln(w_s) = 1.930$$



با توجه به نمودار بالا:

$$\beta = 0.116\,, \ln(w_s) = 0.869$$

پس نتیجه‌ی کلی به صورت زیر است.

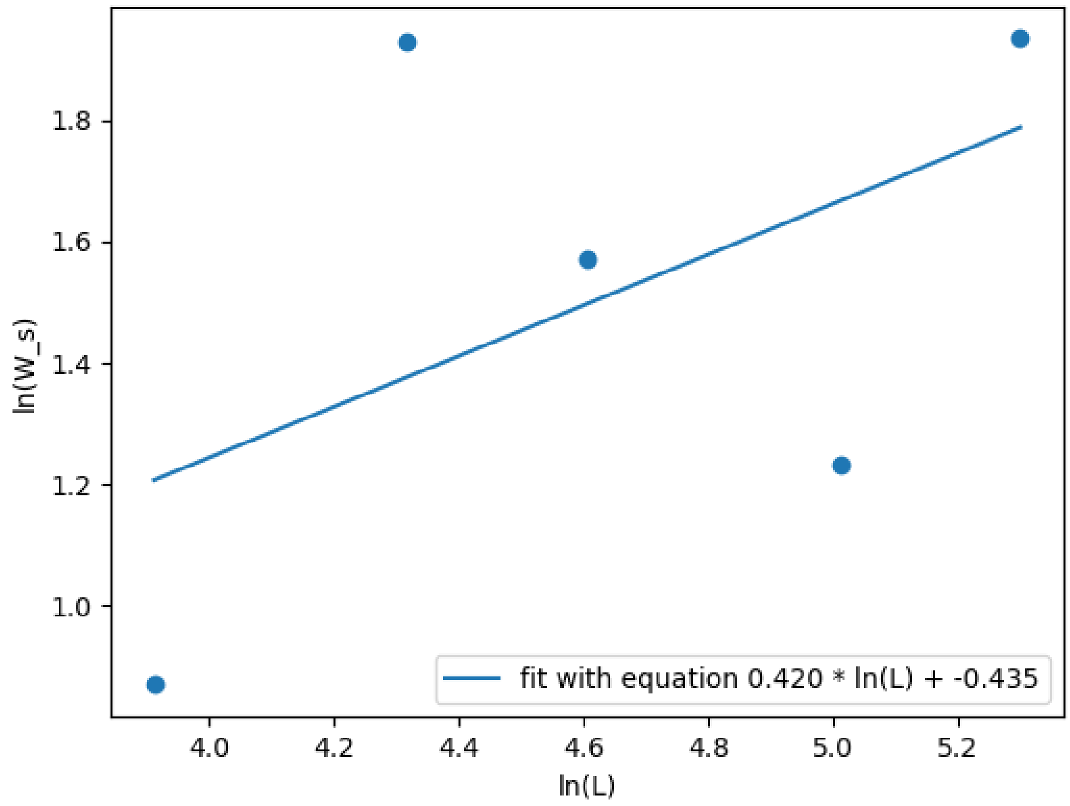
L	$\mathrm{Ln}(w_s)$	β
200	1.935	0.216
150	1.232	0.214
100	1.570	0.192
75	1.930	0.168
50	0.869	0.116
Average:		0.181

پس داریم:

$$\beta_{simulated} = 0.181$$

که با مقدار تئوری آن که ۰٫۲۴ است، حدوداً ۲۵٪ خطای نسبی دارد!

اکنون برای بدست آوردن مقدار آلفا، نمودار زیر را تحلیل می‌کنیم.



شیب بهترین خط در بالا، مقدار آلفا را به ما می‌دهد.

$$\alpha_{simulated} = 0.420$$

این مقدار با مقدار تئوری آن یعنی ۰٫۴۸، حدوداً ۱۳٪ خطای نسبی دارد!

اکنون می‌توان ضریب z را بدست آورد.

$$z_{theory} = 2$$

$$z_{simulated} = \frac{\alpha}{\beta} = 2.320$$

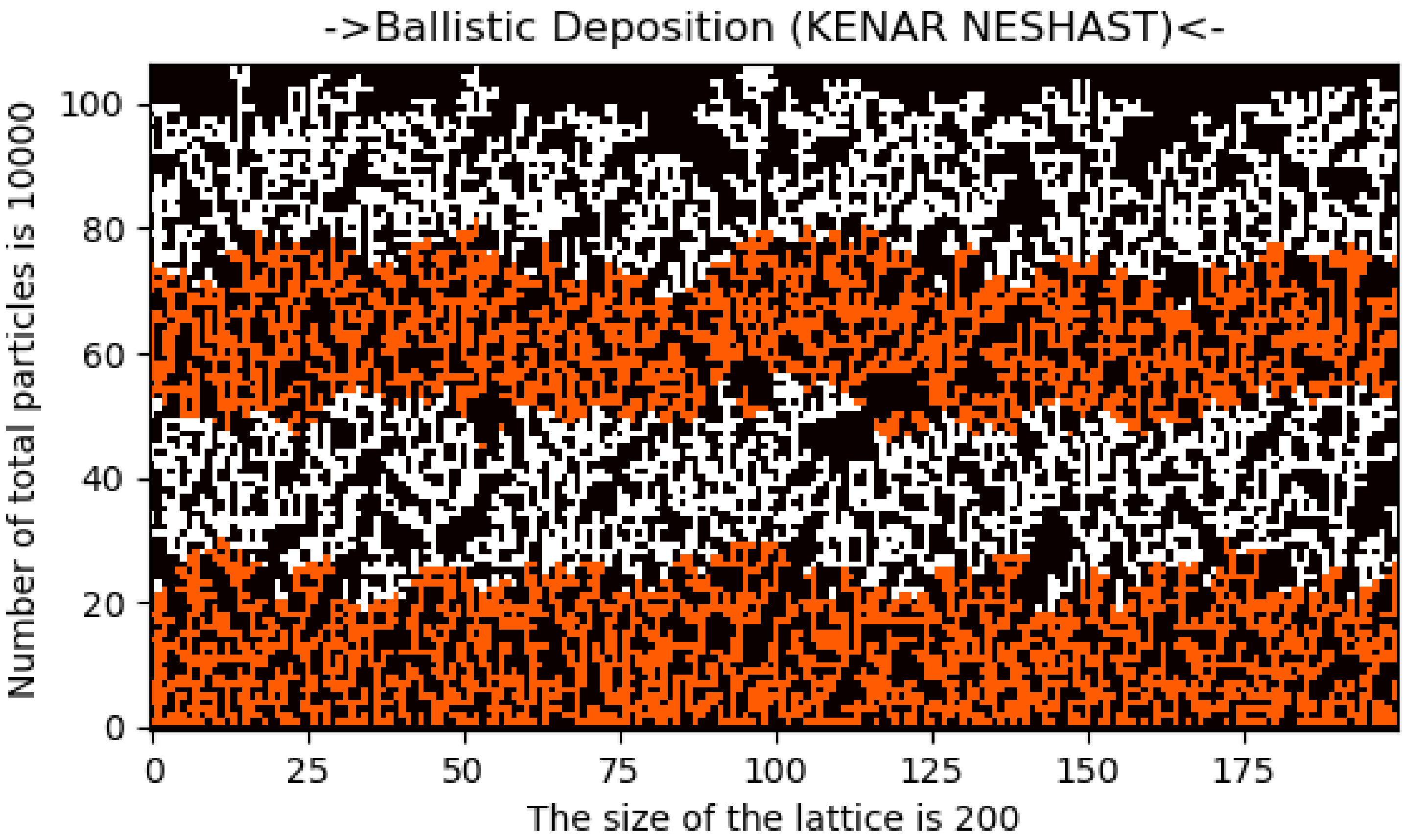
که با مقدار تئوری حدوداً ۱۶٪ خطای نسبی دارد.

۲. مدل کنارنشست در یک بعد

در مدل کنارنشست در یک بعد و با شرایط مرزی تناوبی الگوریتم به این صورت است که در هر گام یک خانه از شبکه به صورت تصادفی انتخاب می‌شود و سپس ذره‌ی جدیدی که در آن مکان به شبکه اضافه می‌شود به بلندترین خانه‌ی مجاور می‌چسبد. به عبارتی دیگر اگر

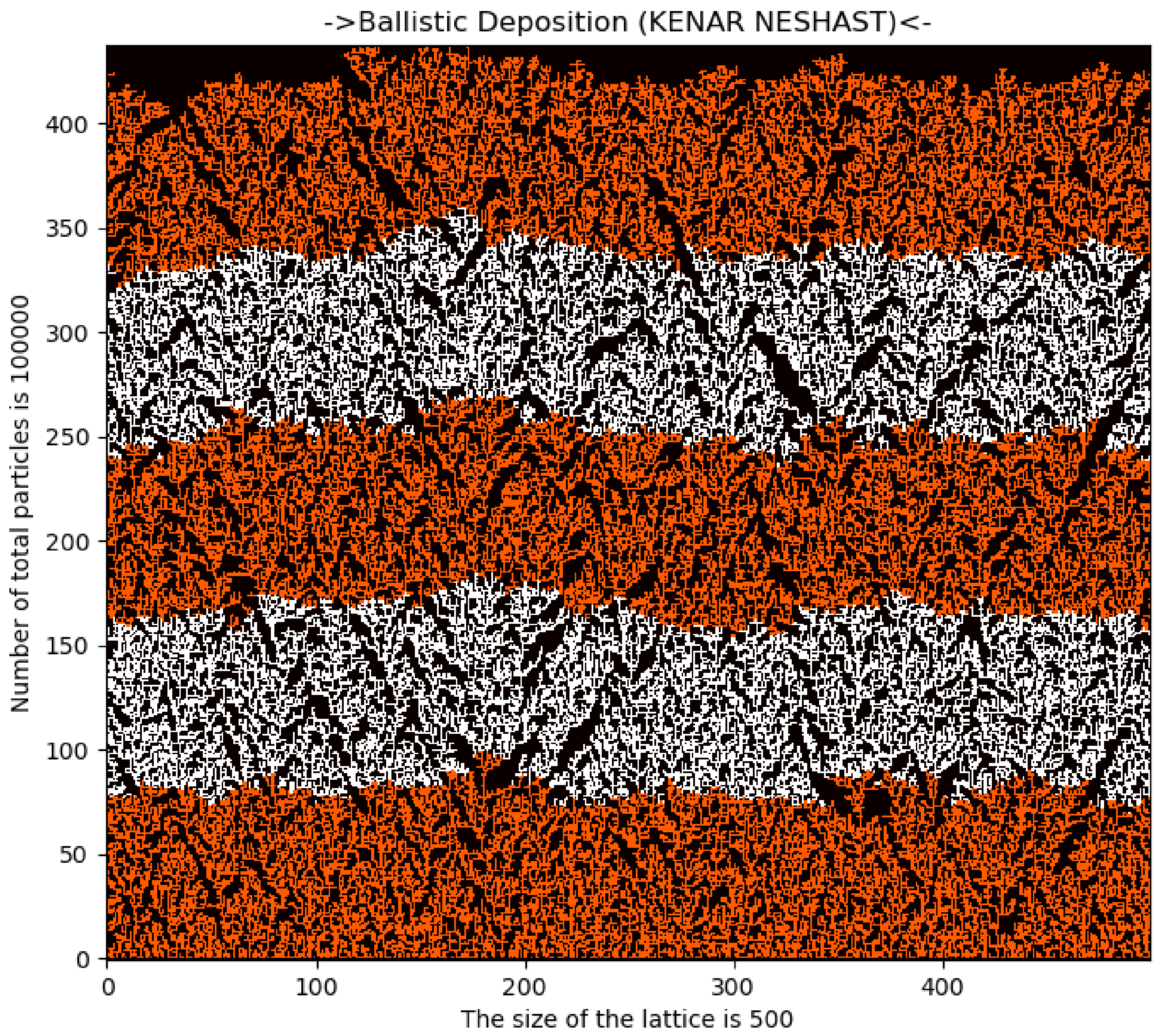
در مجاورت راست و چپ خود همسایه‌ی بلندتری باشد به بالاترین خانه‌ی آن می‌چسبد در غیر این صورت به همان مکان یک واحد اضافه می‌شود.

در ادامه به کمک تابع showLayer() که درون تابع ballisticSideDeposition() تعریف شده است تصویر لایه‌ها را تولید می‌کنیم. در ادامه سه نمونه از لایه‌ای که از این مدل نشانده می‌شود را آورده‌ام.



همان‌طور که انتظار داشتیم با توجه به الگوریتم دینامیک این مسئله، شاهد یک لایه‌ی متخلخل هستیم. شکل بالا برای ۱۰,۰۰۰ ذره و شبکه‌ای به طول ۲۰۰ بدست آمده است. یک نکته‌ی دیگر که جالب توجه است این است که رشد ناهمواری در این لایه (با توجه به مرزهای دولایه‌ی ناهم‌رنگ همسایه) نسبت به ول‌نشست کم‌تر است اما نسبت به پایین نشست بیش‌تر است.

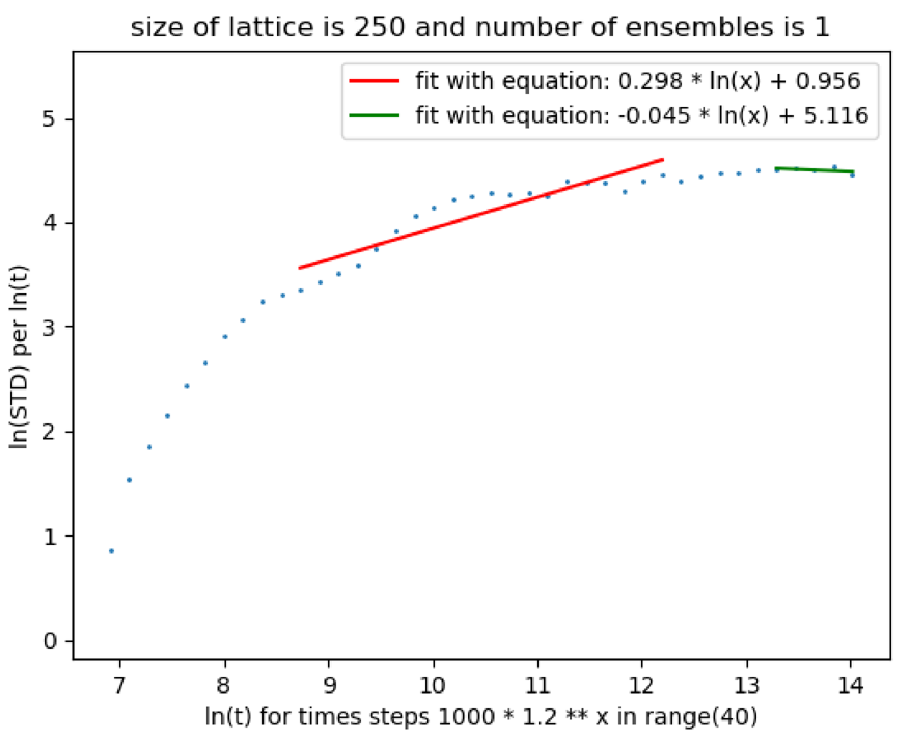
در ادامه یک نمونه‌ی دیگر از لایه‌های کنارنشست را آورده‌ام.



تعداد ذرات این‌بار ۱۰۰,۰۰۰ و طول شبکه ۵۰۰ است.

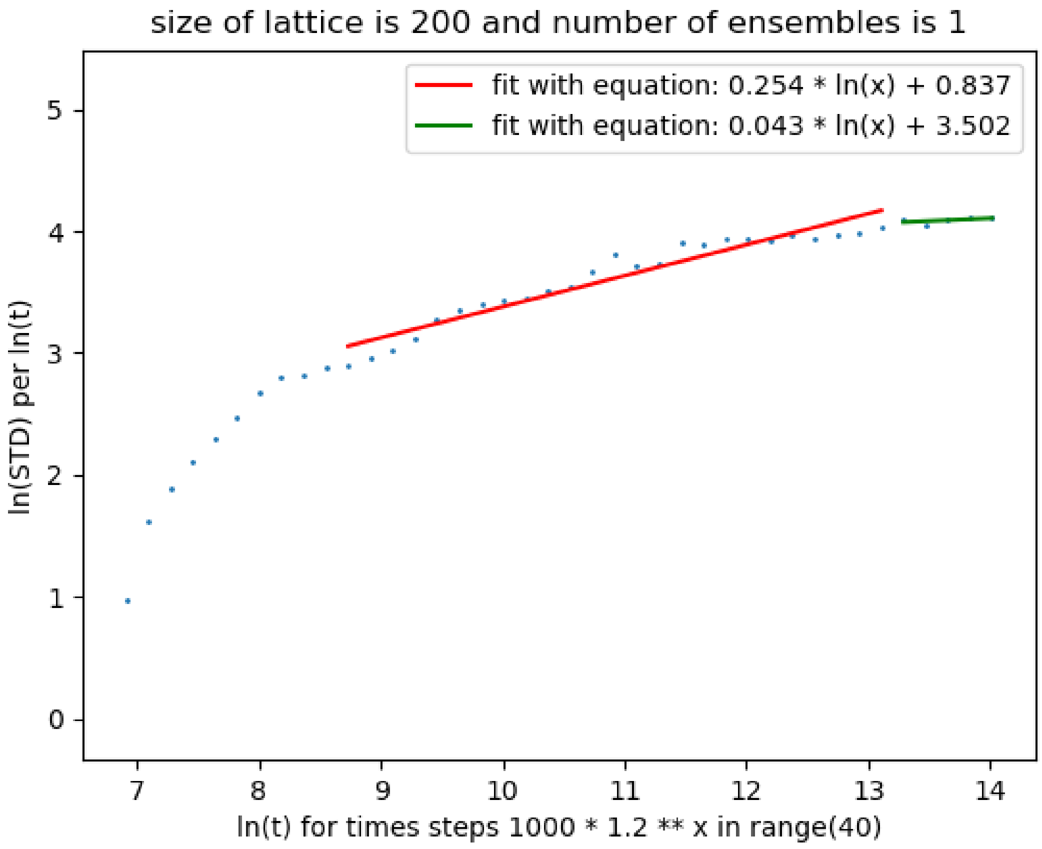
در ادامه نتایج عددی مورد نیاز را بدست آورده‌ام. البته در مورد این نمودارها به علت مضیقه‌ی وقت تنها یک آنزامبل در نظر گرفته‌ام. طبیعتاً با بیش‌تر کردن تعداد آنزامبل‌های می‌توان نتایج بهتری بدست آورد. ابتدا یادآور می‌شوم که به لحاظ تئوری مقادیر عددی آلفا و بتا به صورت زیر است.

$$\alpha_{theory} = 0.47 \,, \beta_{theory} = 0.33$$



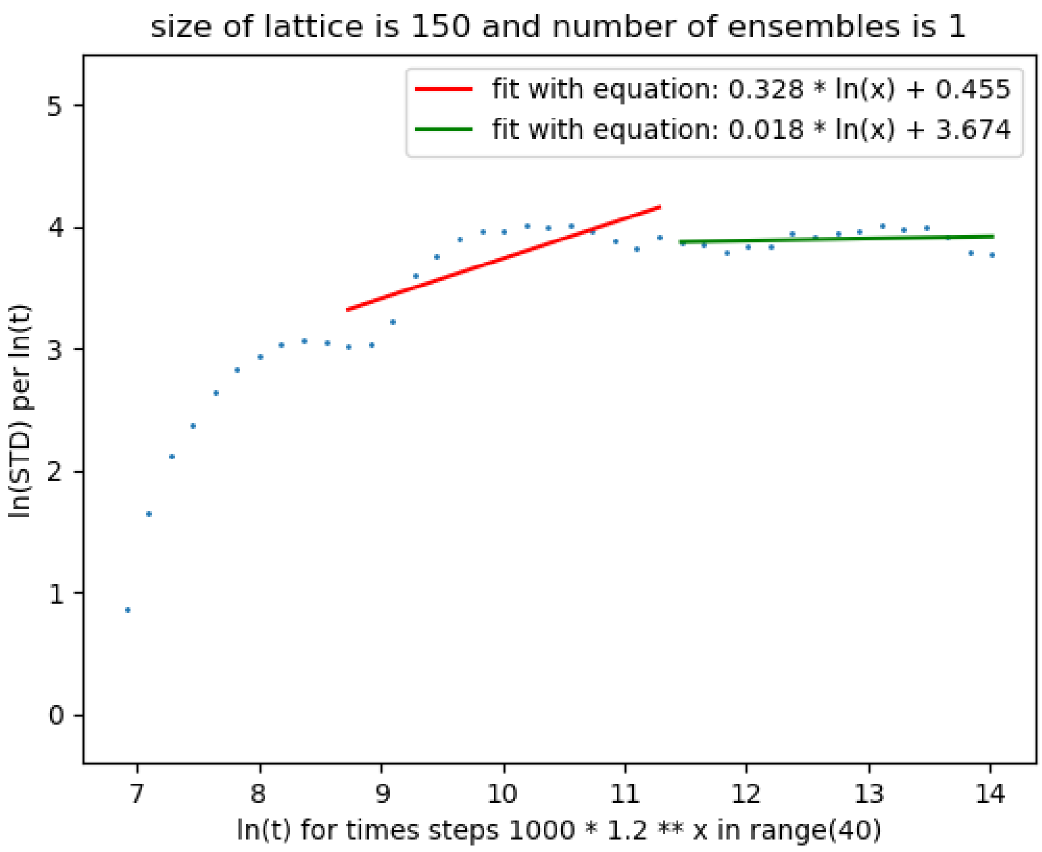
با توجه به نمودار بالا:

$$\beta = 0.298\,, \ln(w_s) = 5.116$$



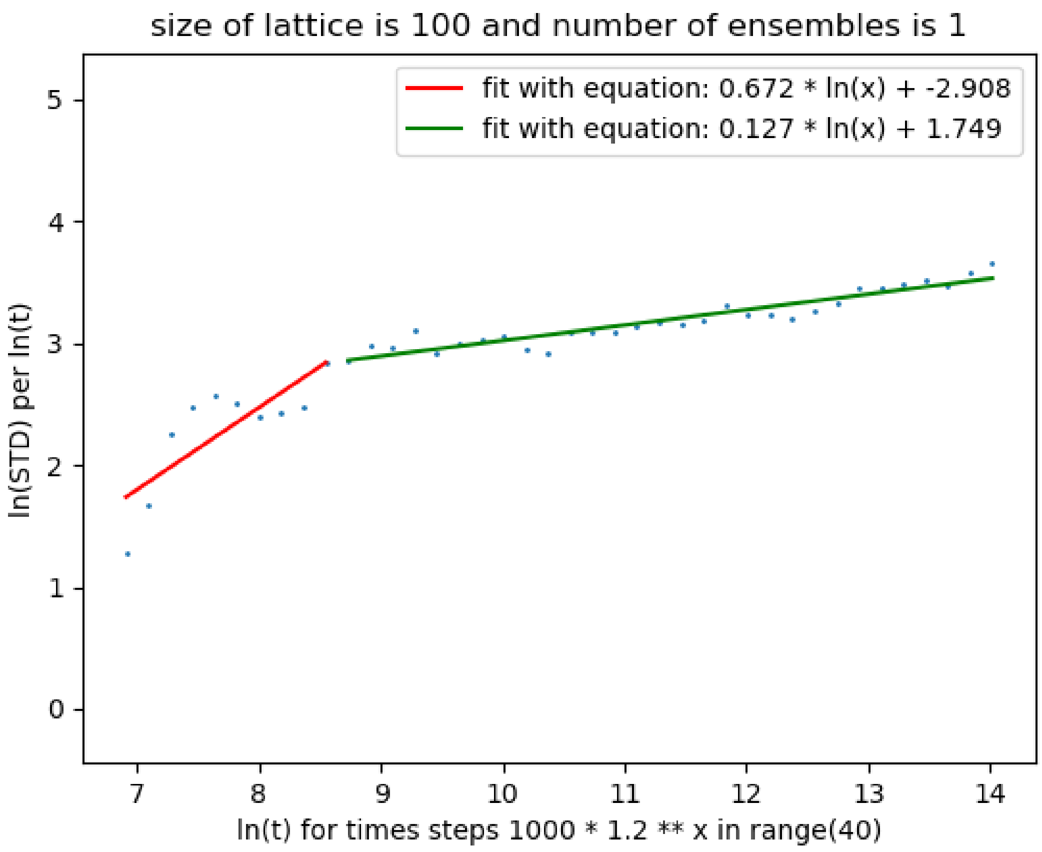
با توجه به نمودار بالا:

$$\beta = 0.254\,, \ln(w_s) = 3.502$$



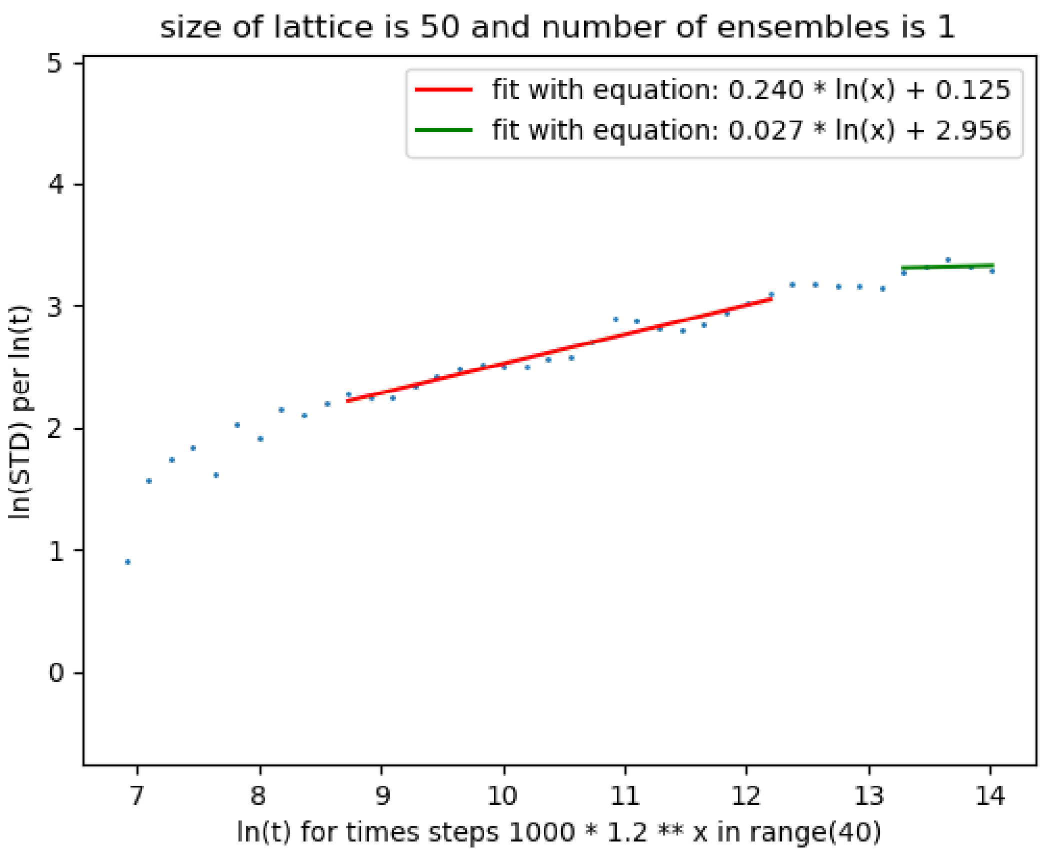
با توجه به نمودار بالا:

$$\beta = 0.328\,, \ln(w_s) = 3.674$$



باتوجه به نمودار بالا:

$$\beta = 0.672, \ln(w_s) = 1.749$$



باتوجه به نمودار بالا:

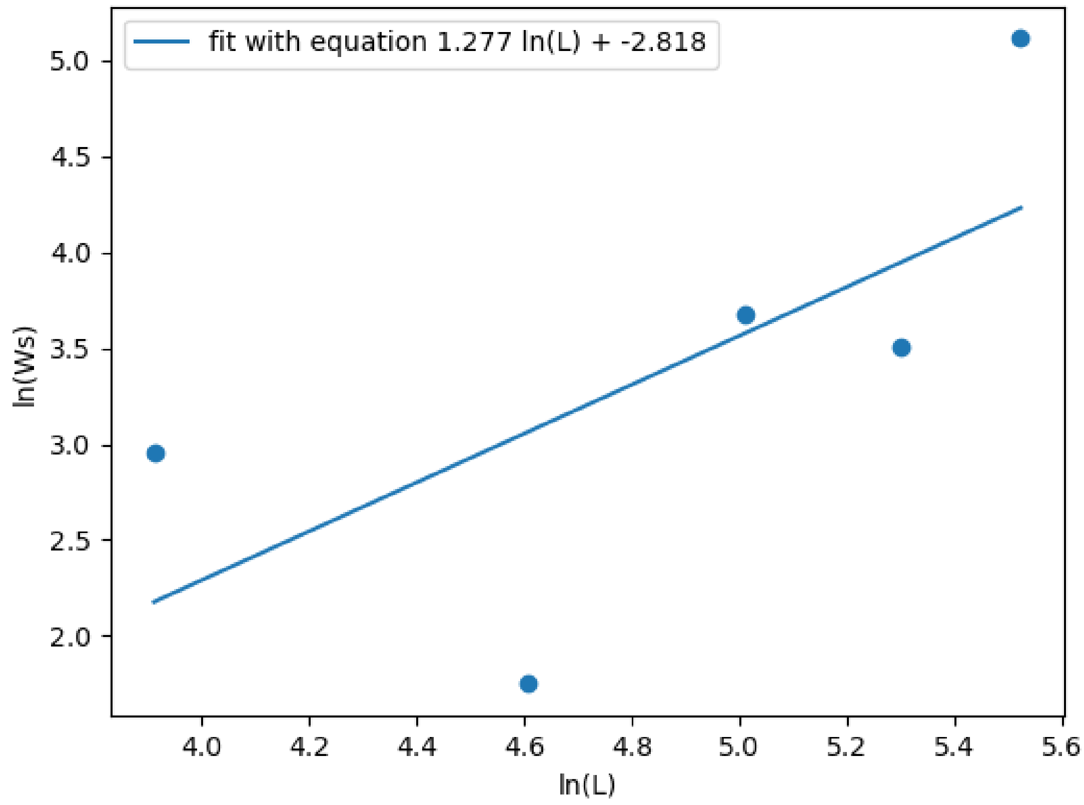
$$\beta = 0.240\,, \ln(w_s) = 2.956$$

این نتایج را می‌توان در جدول زیر گردآوری کرد.

L	Ln(w_s)	β
250	5.116	0.298
200	3.502	0.254
150	3.674	0.328
100	1.749	0.672
50	2.956	0.240
Average:		0.358

از این داده‌ها نتیجه می‌گیریم که مقدار بتا حدوداً ۰,۳۵۸ است که با مقدار واقعی آن۰,۳۳ حدود ٪۸ خطای نسبی دارد.

داده‌های جدول بالا را می‌توان در نمودار زیر هم نمایش داد تا از شیب آن مقدار آلفا را بدست آوریم.



با توجه به شیب خط نمودار بالا مقدار آلفا حدوداً ۱,۲۷۷ است که با مقدار واقعی یعنی ۰,٤٧ مقدار زیادی اختلاف دارد! دلیل این اختلاف زیاد را می‌توان در این دید که باید تعداد آنزامبل‌های زیادی برای انجام شبیه‌سازی انتخاب کنیم.

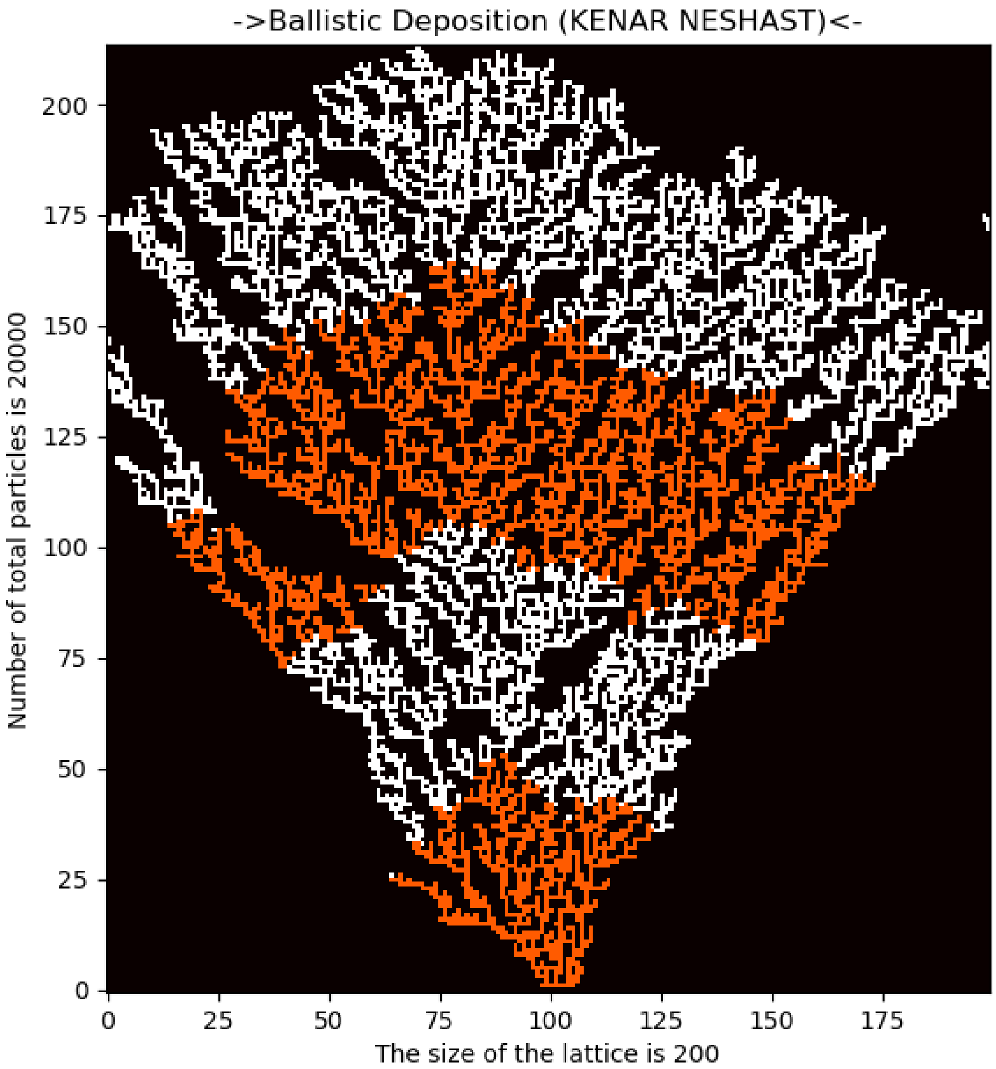
در نهایت مقدار z را می‌توانیم حساب کنیم.

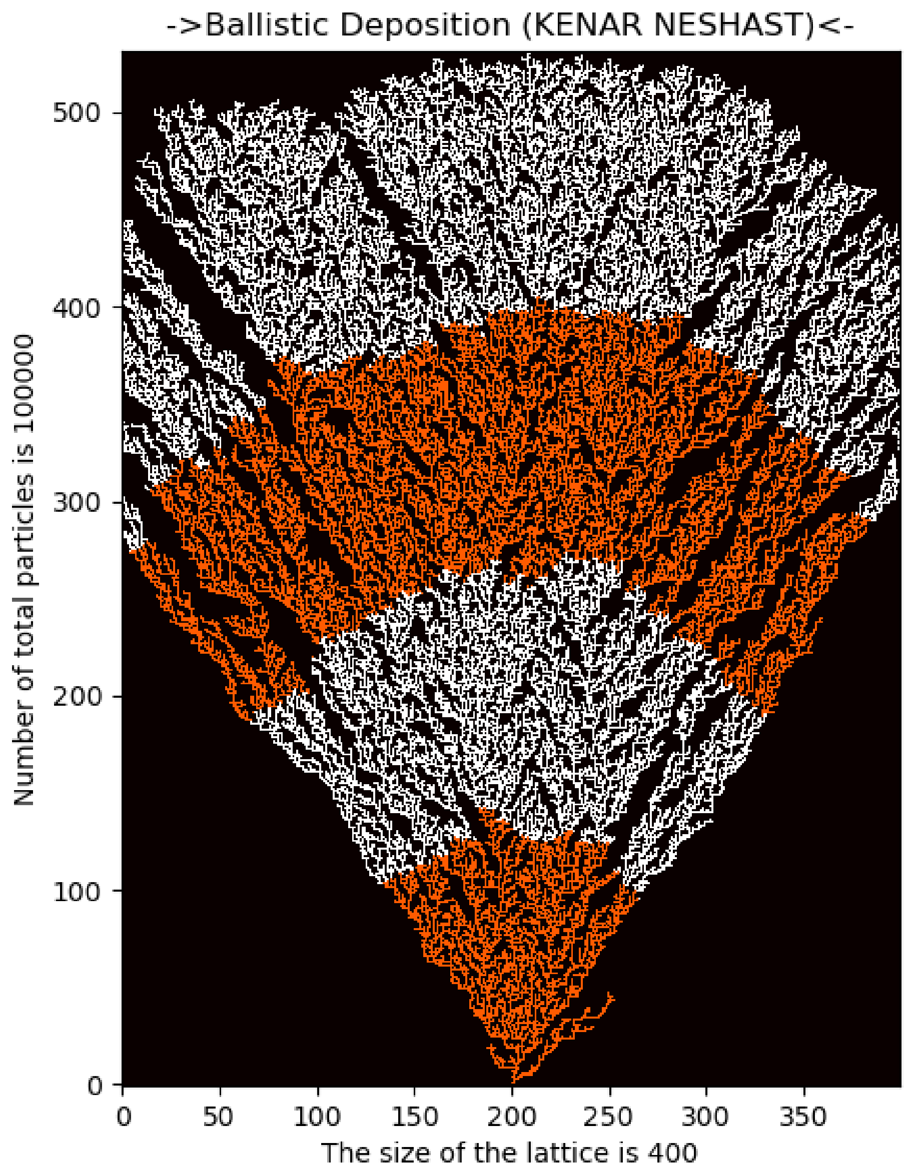
$$z_{simulated} = \frac{\alpha}{\beta} = 3.567$$

که از مقدار تئوری آن یعنی ۱,٤٢ مقدار زیادی اختلاف دارد! این اختلاف زیاد از آلفا به ارث رسیده است. به نظر می‌رسد مقدار آلفا حساسیت بیش‌تری نسبت به تعداد آنزامبل‌ها دارد.

۳.طول همبستگی در کنارنشست

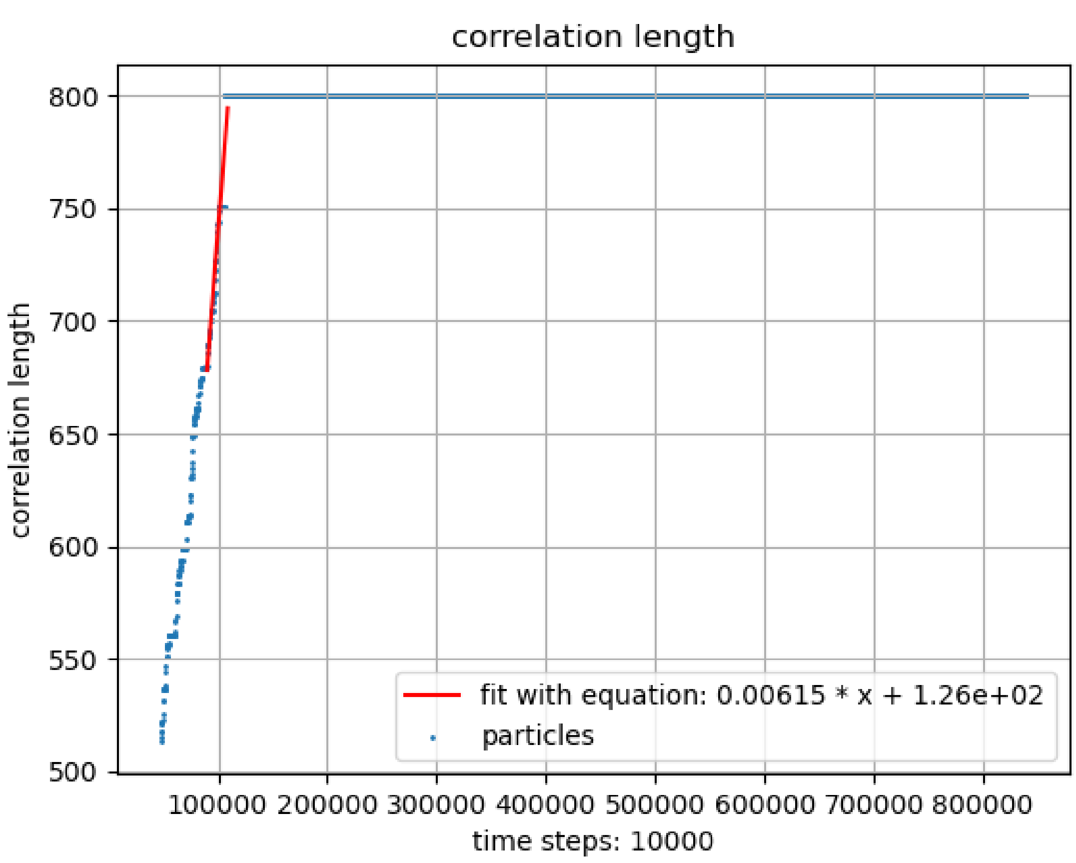
برای آن‌که رشد طول همبستگی در کنارنشست را به خوبی ببینیم، می‌توانیم ابتدا یک خانه‌ی میانی لتیس در پایین‌ترین لایه را روشن کنیم. سپس با توجه به دینامیکی که این مدل رشد لایه دارد، هر ذره‌ی فرودی اگر همسایه‌ای برای خود پیدا کند در جای خود متوقف می‌شود و در غیر این صورت از شبکه خارج می‌شود. در ادامه شکل‌های زیبایی به صورت زیر تولید می‌شود. این شکل‌ها هر کدام برای تعداد متفاوتی از ذرات و طول شبکه رسم شده‌اند.



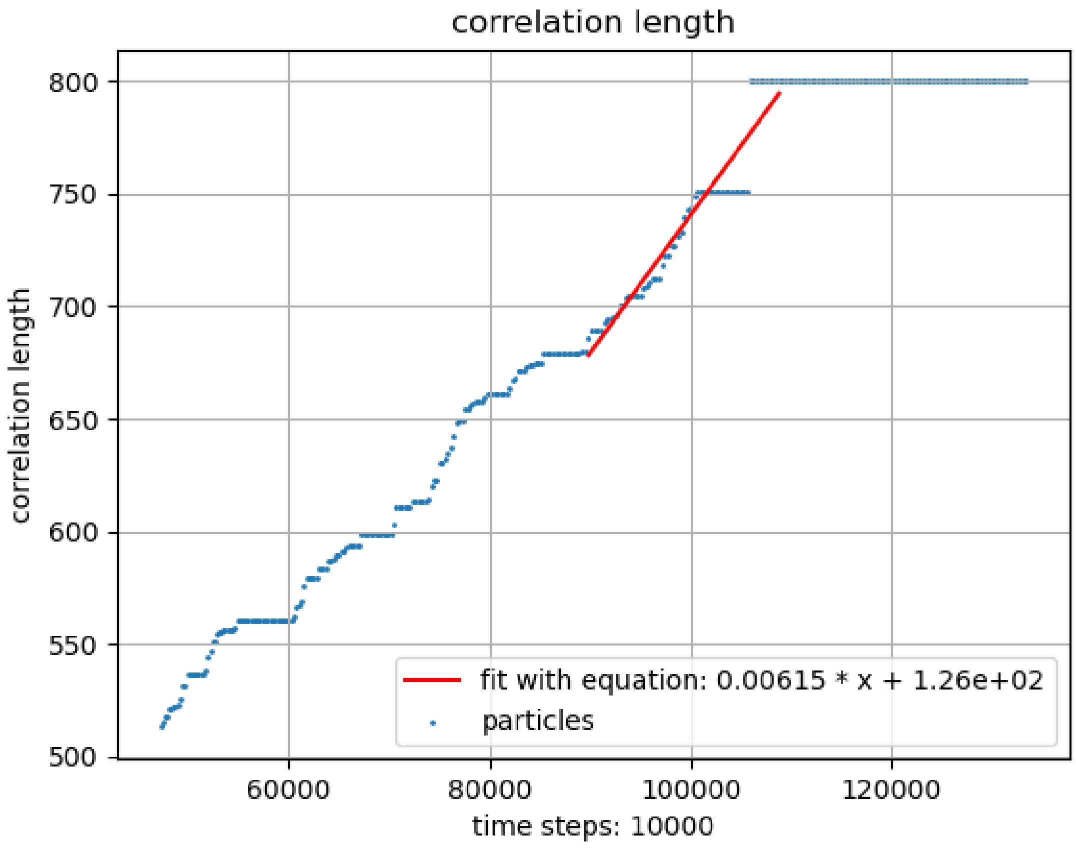


همان‌طور که در شکل‌های بالا قابل مشاهده است، پس از آن‌که تعداد ذرات از یک حدی بیش‌تر شود، عرض درختچه کل طول شبکه را می‌پوشاند. به این ترتیب عرض درختچه بعد از یک مدت زمانی دیگر تغییر نمی‌کند و مقداری ثابت و برابر با طول شبکه به خود می‌گیرد. یک نکته‌ی دیگر که باید به آن توجه کرد این است که ما در اینجا شرایط مرزی را تناوبی گرفته‌ایم. بنابراین ممکن است جزیره‌های کوچکی در سمت راست یا در سمت چپ شکل دیده شود که به درخت اصلی متصل نیستند.

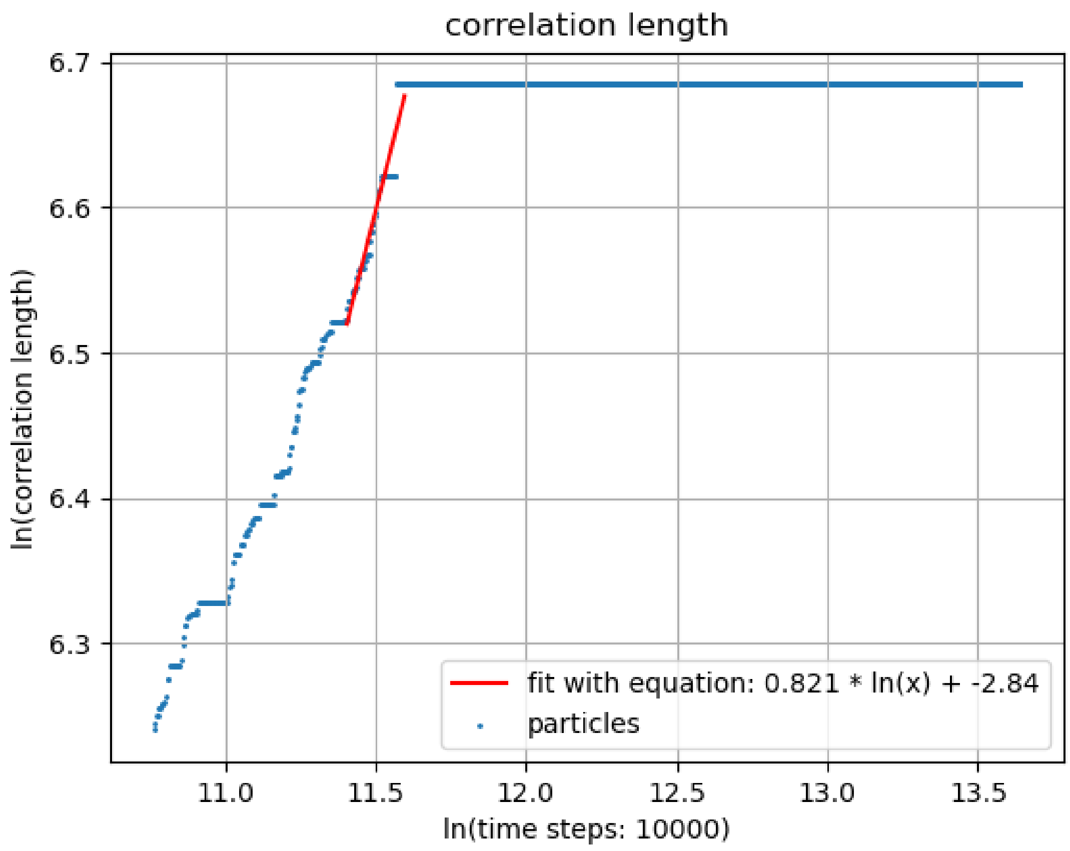
در ادامه برای درک بهتر رابطه‌ی تعداد ذرات و طول همبستگی (یا همان زمان و طول همبستگی) نمودار عرض درخت برحسب تعداد ذرات (یا همان زمان) را رسم می‌کنم. نمودار مطابق شکل زیر است.



همان‌طور که در نمودار بالا مشاهده می‌کنید، میان طول همبستگی و زمان به خوبی یک رابطه‌ی خطی برقرار است و نه یک رابطه‌ی توانی. البته باید به این نکته توجه داشته باشیم که از یک زمانی به بعد، که طول درخت با طول لتیس برابر می‌شود، این طول همبستگی دیگر افزایش نمی‌یابد و مقدار ثابت طول لتیس را دارد. در واقع در این زمان‌ها به بعد، میان تمام ذرات سیستم همبستگی وجود دارد. برای آن‌که رفتار خطی بهتر در معرض دید باشد، بخشی از داده‌هایی که مربوط به فراگیر شدن طول لیتس هستند را حذف می‌کنم و نمودار زیر را خواهیم داشت.



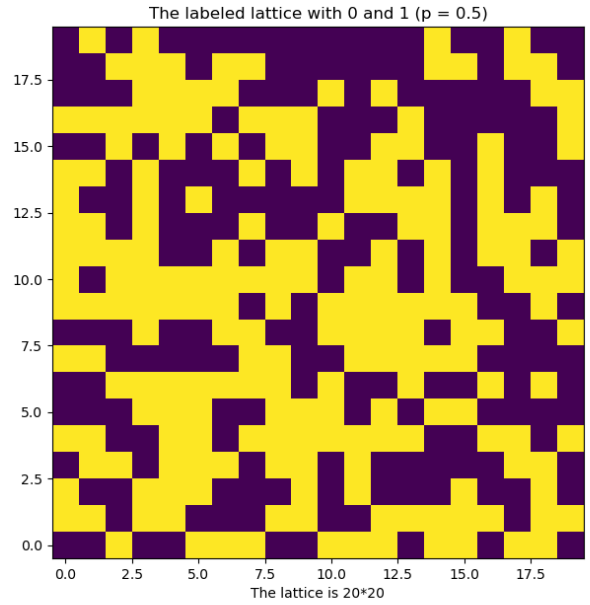
در نمودار بالا مشاهده می‌شود که در بعضی بازه‌های زمانی نسبتاً کوچک طول همبستگی تغییر نمی‌کند. این موضوع را می‌توان با دقت کردن به شکل‌های لایه‌ها که در ابتدا رسم کردم هم مشاهده کرد. به جهت آن‌که بررسی کامل‌تری صورت گیرد نمودار لگ-لگ را هم رسم می‌کنم.

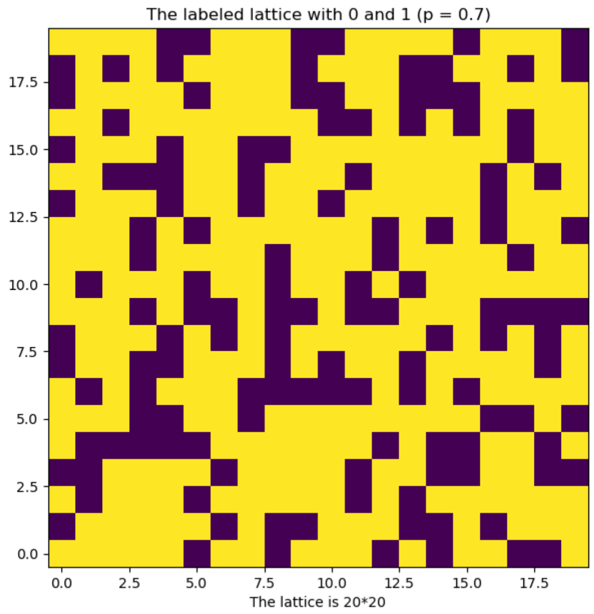


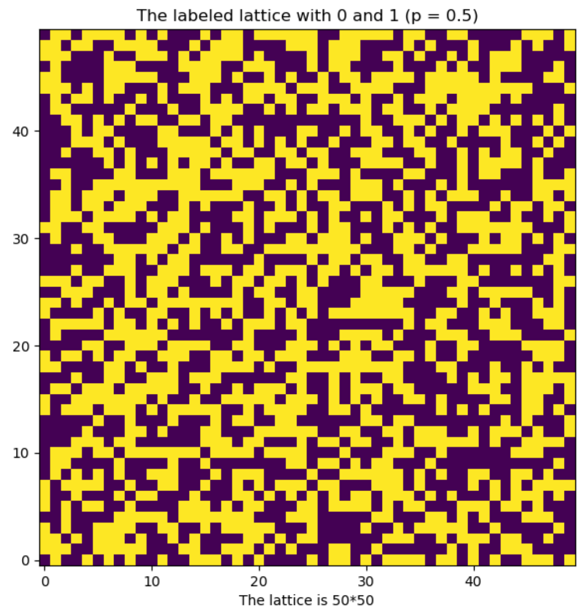
با استفاده از ابزارهای curve fitting پیکج scipy توانستم خط بهینه را که همان خط قرمز رنگ است رسم کنم. شیب این خط حدوداً ۰٫۸۲ است که البته از مقدار دقیق ۱ مقداری انحراف دارد. برای بهتر کردن شیب این خط می‌توان آنزامبل‌گیری کرد و میانگین شیب‌هایی که بدست می‌آید را گزارش کرد. اما خب به هرحال از شکل نمودارهای بالا رفتار خطی کاملاً استنباط می‌شود و به نظرم چندان نیازی به آنزامبل‌گیری نباشد. بنابراین نمای رفتار همبستگی با زمان ۱ است.

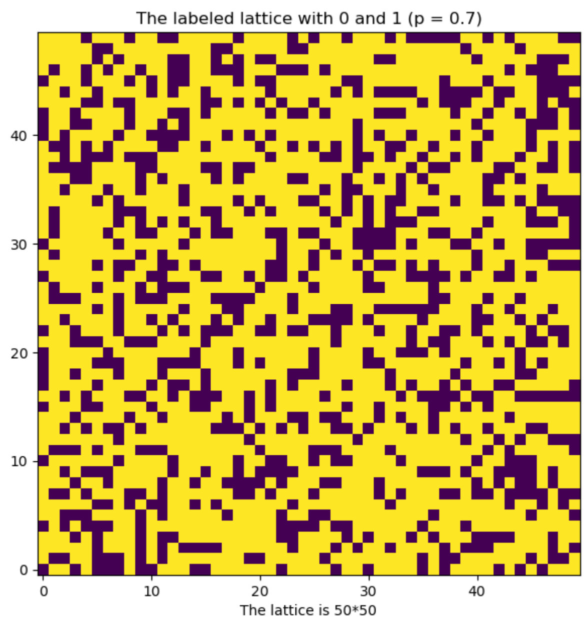
۴.تراوش

برای انجام محاسبات و رسم نمودارهای مورد نیاز مربوط به تراوش، تابع ()percolation را تعریف کرده‌ام. در دل این تابع، توابع دیگری تعریف شده‌اند که هر کدام وظیفه‌ی خاصی برای انجام دادن دارد (که فکر می‌کنم از نامگذاری‌هایی که استفاده کرده‌ام، کار هر یک از توابع به خوبی معلوم باشد!). در ادامه، اولین کاری که انجام می‌دهیم، این است که یک شبکه‌ی مربعی که طول آن را کاربر تعیین می‌کند می‌سازیم. سپس هر خانه از این شبکه را با یک احتمالی که آن را نیز کاربر تعیین می‌کند، روشن می‌کنیم. در ادامه ٤ شکل زیر را برای طول‌ها و احتمال‌های مختلف تولید کرده‌ام. این بخش از کار با استفاده از زیرتابع‌های ()makeBinaryLattice و ()showBinaryLattice انجام می‌شود.









در شکل‌های بالا که با ()imshow تولید شده‌اند، خانه‌های زردرنگ نشان‌دهنده‌ی خانه‌هایی است روشن شده‌اند.

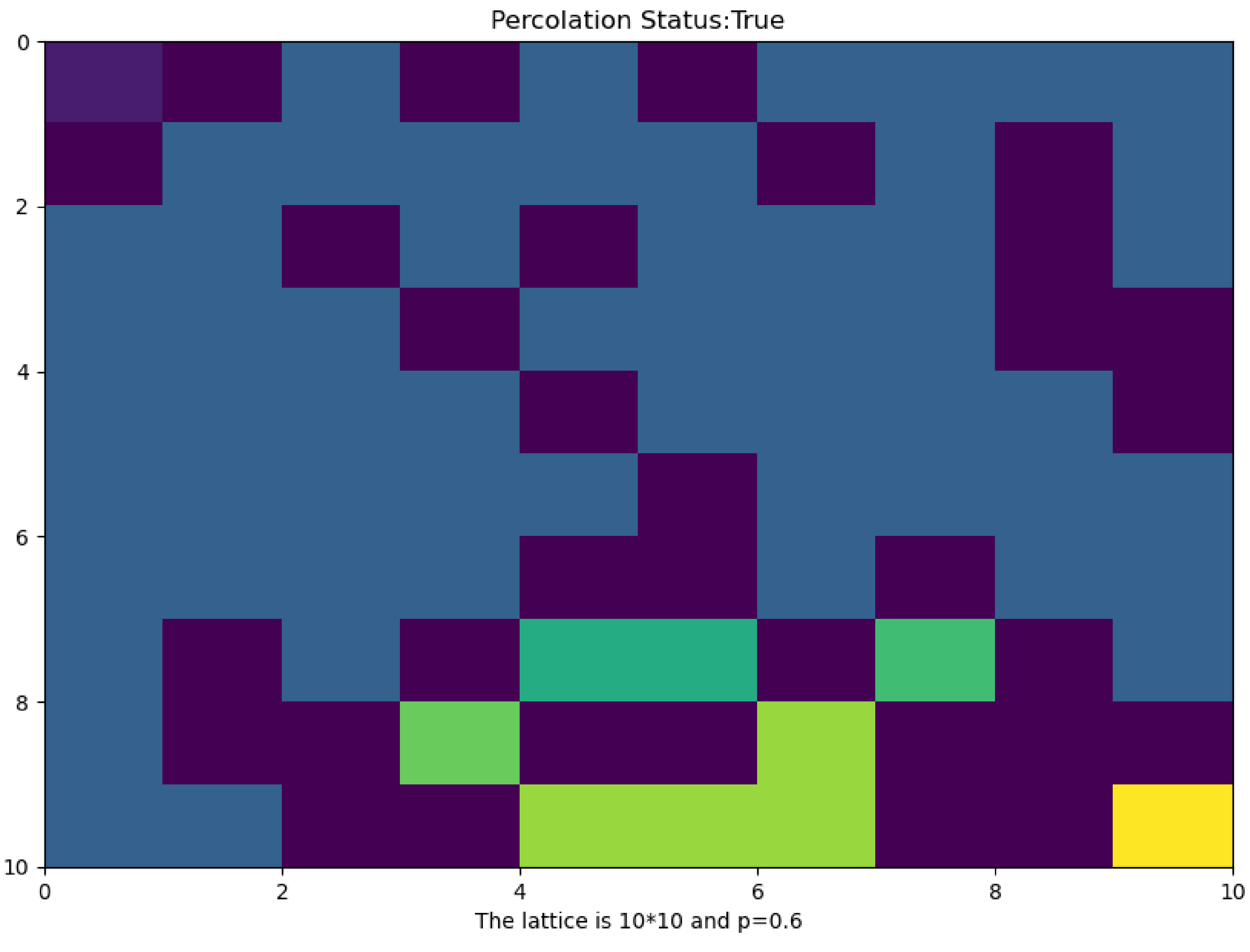
در ادامه قصد داریم برای هر شبکه‌ی مربعی دلخواه و با هر احتمال روشن کردن دلخواه، مشخص کنیم که آیا تراوش رخ داده است یا خیر. برای انجام این کار از الگوریتم هشن-کپلمن استفاده می‌کنم. این الگوریتم با اندازه‌ی شبکه (تعداد خانه‌ها)، به صورت خطی رشد می‌کند و نسبت به الگوریتم رنگ‌آمیزی بهینه‌تر و سریع‌تر است. پیاده‌سازی این الگوریتم در تابع ()makeLabeledLattice انجام شده است. برای این منظور ابتدا دو تابع دیگر ()mergeClusterLabels و ()findClusterLabel را تعریف می‌کنم. کار تابع اول این است که هر گاه دو خوشه به هم رسیدند، برچسب متناظر با آن‌ها را هم‌ارز قرار دهد. کار تابع دوم هم این است که تعیین می‌کند هر خانه‌ی شبکه به کدام خوشه تعلق دارد. برای این منظور متناظر با خوشه‌ای که خانه‌ی مورد نظر به آن تعلق دارد یک برچسب نسبت می‌دهد. برچسبی که نسبت می‌دهد به وضعیت همسایگی

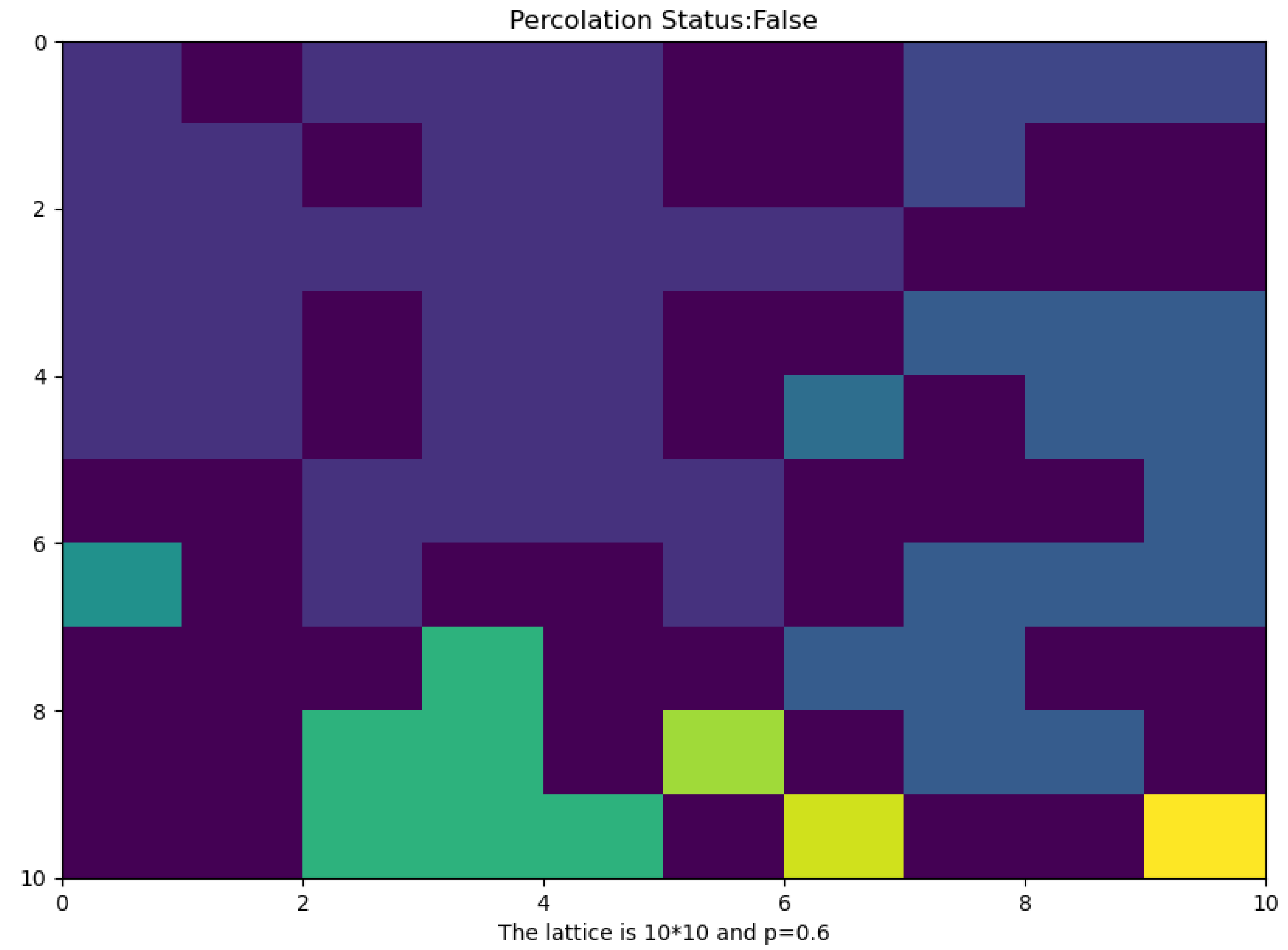
آن خانه بستگی دارد. در ادامه با دو حلقه‌ی for تو در تو، تمام خانه‌های شبکه را پیمایش می‌کنیم. اگر خانه هیچ همسایه‌ی بالا و چپی نداشته باشد، یک برچسب جدید اختیار می‌کند. اگر تنها همسایه‌ی بالا یا تنها همسایه‌ی چپ داشته باشد برچسب آن را به خود می‌گیرد. در نهایت اگر هم همسایه‌ی بالا و هم همسایه‌ی چپ داشته باشد، تابع mergeClusterLabels() را فراخوانی می‌کنیم تا این دو برچسب را هم‌ارز کند و سپس برچسب خانه‌ی چپی را به خانه‌ی جاری نسبت دهد. به این ترتیب تمام خانه‌های شبکه‌ای که در ابتدا با یک و صفر پر شده بود تعیین تکلیف می‌شوند و همه‌ی خوشه‌های موجود در شبکه مشخص و برچسب‌گذاری می‌شوند. توجه شود که در ابتدا، ماتریسی که ساخته می‌شود یک سطر در بالای بالا و یک ستون در سمت چپ شبکه اضافه دارد. دلیل این موضوع این است که از نظر شرایط مرزی محدود، کدمان دچار خطا نشود. پس از آن‌که شبکه‌ی مورد نظرمان با الگوریتم هشن–کپلمن بدست آمد، سطر و ستون اضافه را حذف می‌کنیم.

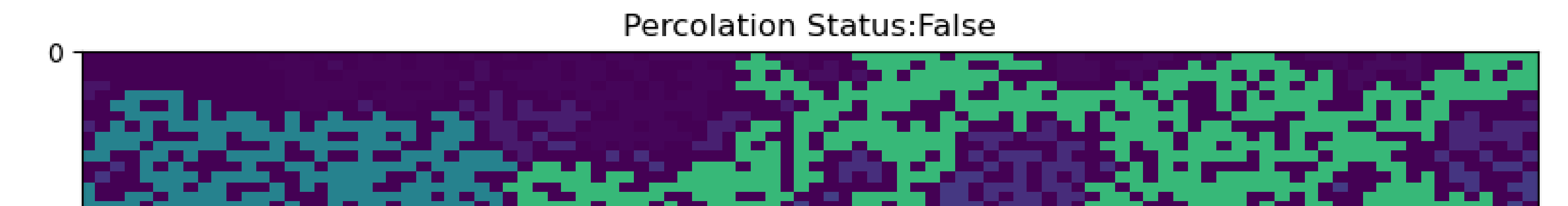
در ادامه چون قصد داریم شبکه را نمایش دهیم، یک بار دیگر بر روی تمام خانه‌های شبکه پیمایش می‌کنیم و برچسب هر خانه را با توجه به خوشه‌ای که به آن تعلق دارد به آن نسبت می‌دهیم. به این ترتیب شبکه‌ی ما از هر نظر کامل و آماده‌ی انجام فعالیت‌های بعدی است.

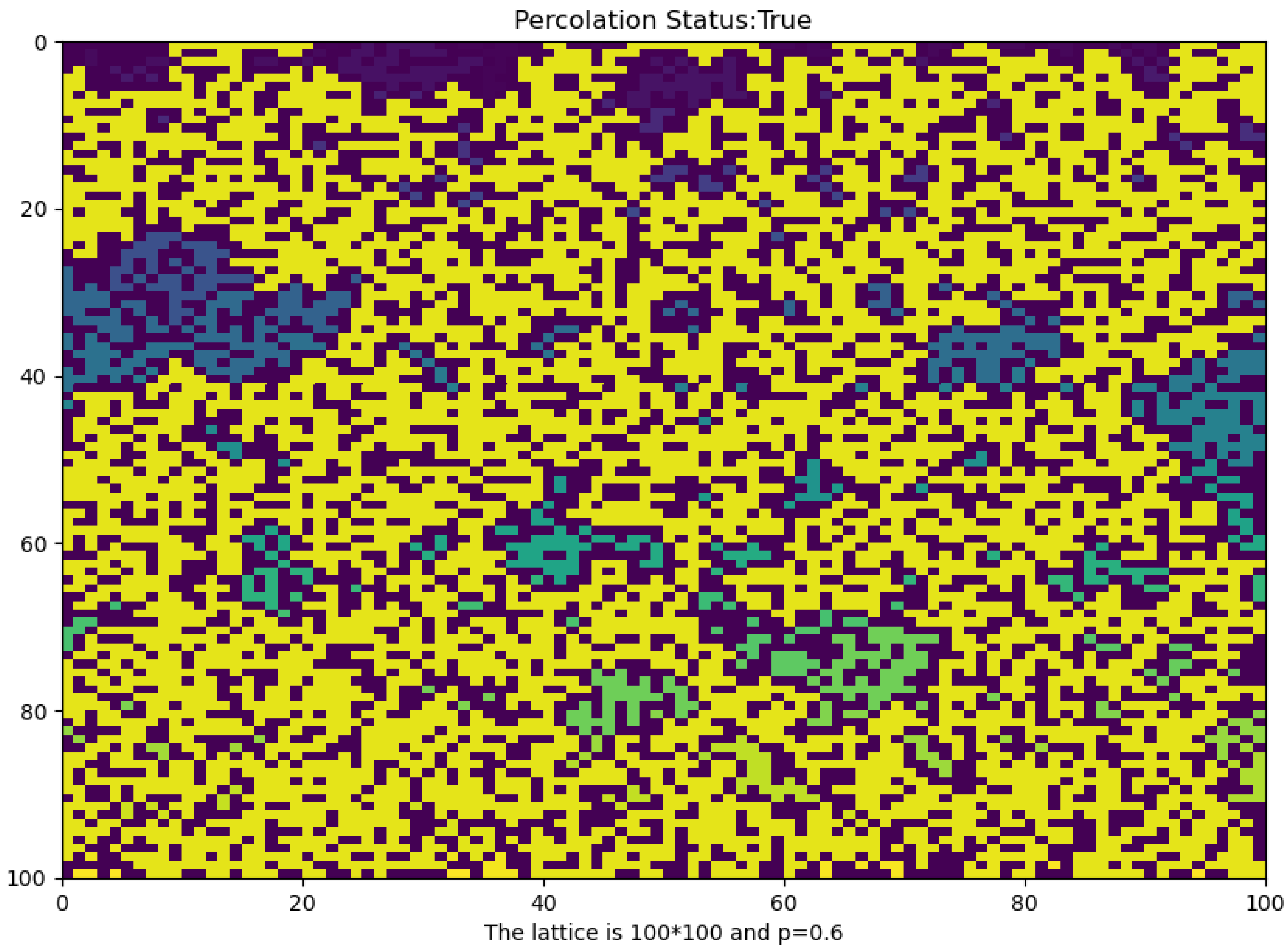
در ادامه یکی از تمرین‌ها از ما خواسته است که شبکه‌ی دوتایی داده شده را که به صورت رندوم تولید شده است تعیین وضعیت کنیم که آیا تراوش (در جهت عمودی از بالا به پایین) رخ داده است یا خیر. نتیجه‌های زیر ارائه می‌شود.

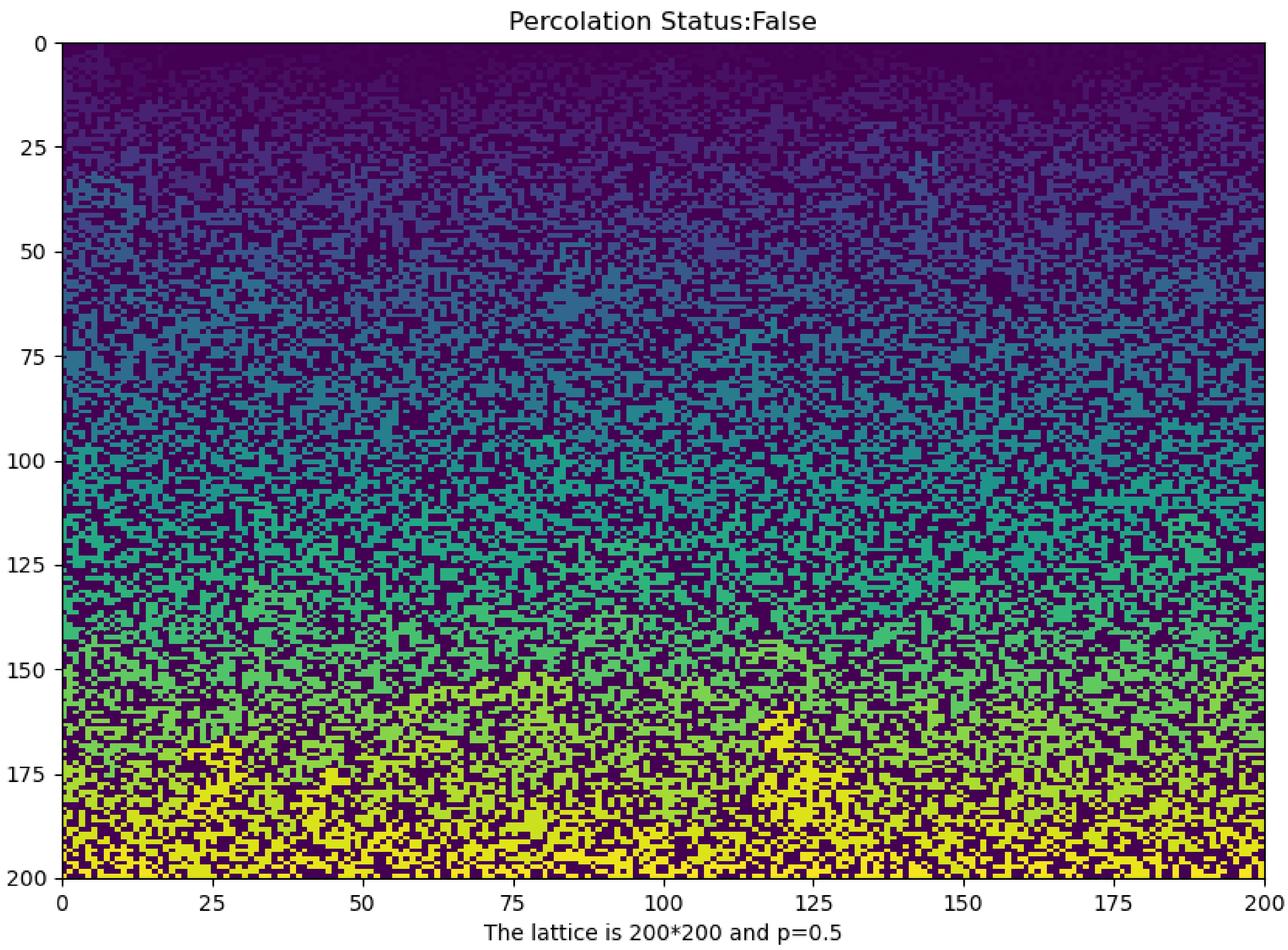
>> توجه: در شکل‌های زیر محور عمودی وارونه نمایش داده شده است. البته این موضوع اهمیتی ندارد اما بعد از آن‌که عکس‌ها را وارد کردم متوجه آن شدم. با این حال در کد اصلی آن را تصحیح کردم. نکته‌ی دیگر این‌که هر خوشه یک برچسب و در نتیجه یک رنگ منحصر به فرد دارد. چون طیف رنگ‌های پیوسته در نظر گرفته شده است ممکن است در بعضی جاها مرز میان خوشه‌های کاملاً واضح نباشد؛ اما اگر بر روی شکل تولید شده زوم کنید حتماً مشخص و واضح خواهد بود.

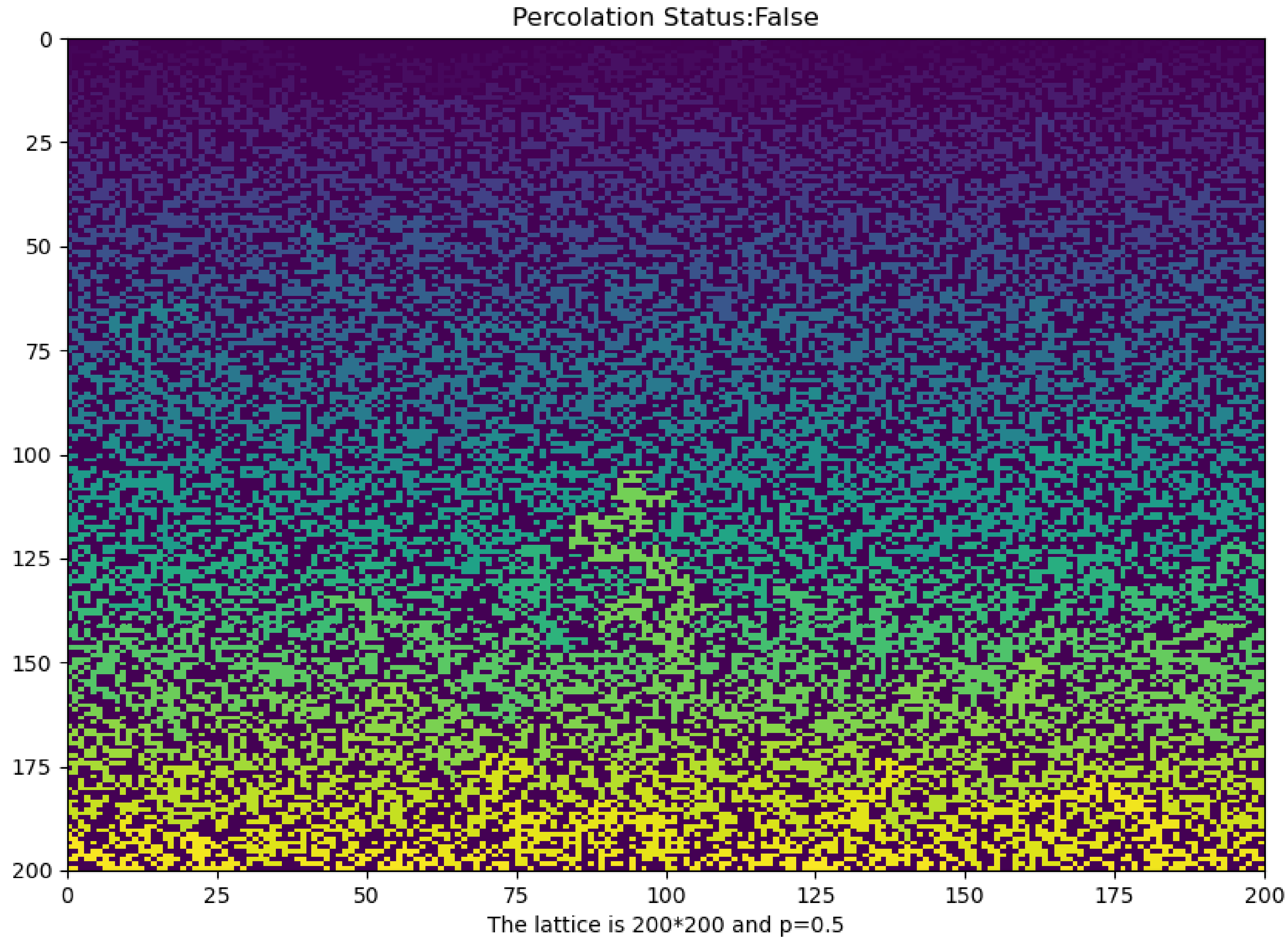



Percolation Status:False
0
2
4
6
8
10
The lattice is 10*10 and p=0.6

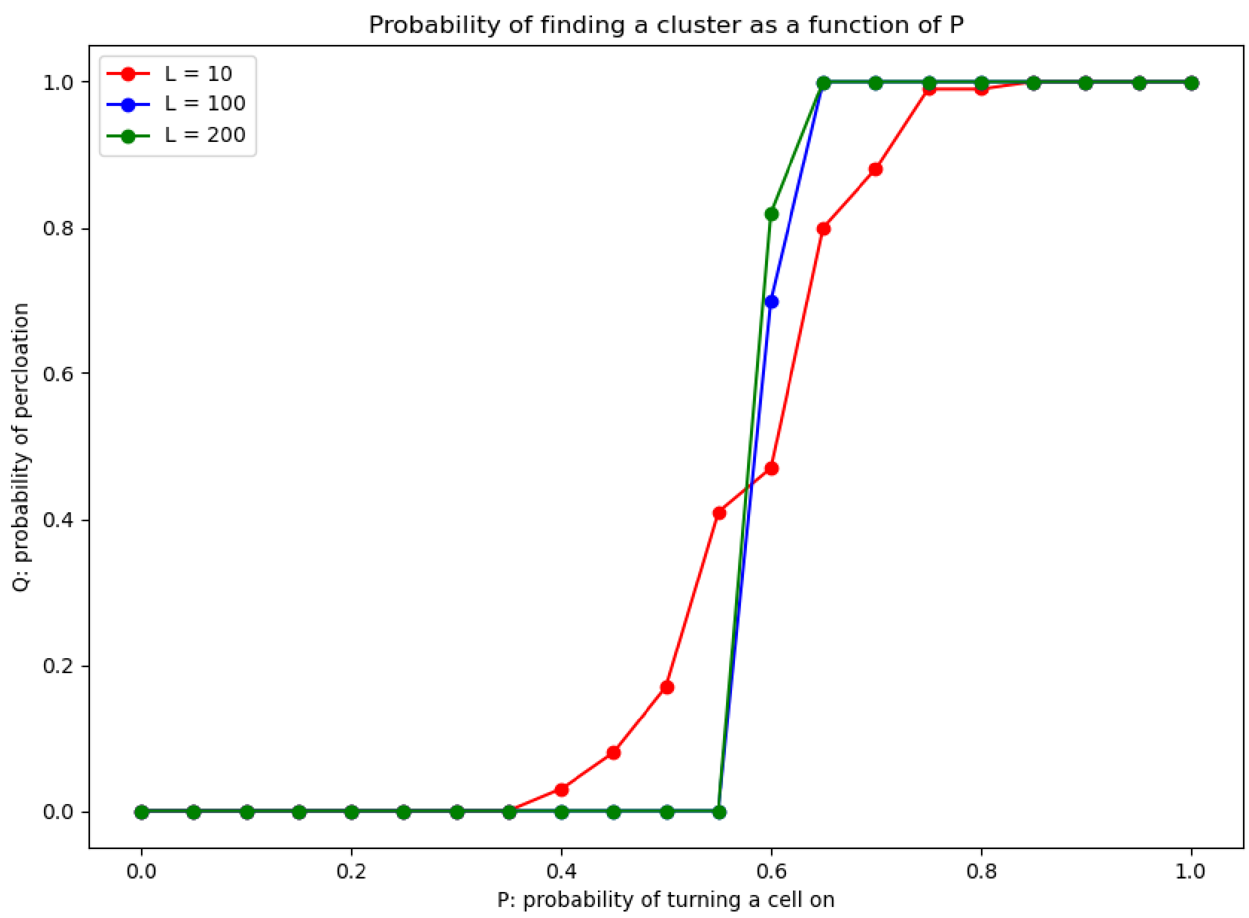

Percolation Status:False
0
20
40
60
80
100
The lattice is 100*100 and p=0.6


Percolation Status:True
0
20
40
60
80
100
0
20
40
60
80
100
The lattice is 100*100 and p=0.6


Percolation Status:False
0
25
50
75
100
125
150
175
200
0
25
50
75
100
125
150
175
200
The lattice is 200*200 and p=0.5



در ادامه می‌خواهیم ببینیم که از چه احتمالی به بعد تراوش رخ می‌دهد. در واقع در حدی که طول شبکه به بی‌نهایت میل کند، تابع توزیع احتمال رخ دادن تراوش دو خروجی بیش‌تر ندارد. به ازای احتمال‌های روشن کردن کم‌تر از یک مقدار حدی، تراوش رخ نمی‌دهد و به ازای احتمال‌های بزرگ‌تر از آن، تراوش حتماً رخ می‌دهد. برای بدست آوردن احتمال تراوش مطابق توضیحات مسئله عمل می‌کنیم. نتیجه در شکل زیر آمده است.



همان‌طور که در نمودار بالا به خوبی دیده می‌شود، با بزرگ‌تر کردن اندازه‌ی شبکه، رفتار پلکانی تابع توزیع احتمال مشخص‌تر می‌شود. هم‌چنین همان‌طور که در نمودار معلوم است، احتمال بحرانی حدوداً ۰٫٦ است که به مقدار تئوری آن (۰٫۵۹) نزدیک است. البته این احتمال بحرانی کمیتی جهان‌شمول نیست و برای یک شبکه‌ی مربعی این مقدار بدست می‌آید. برای نمونه اگر شبکه‌مان را مثلثی بگیریم احتمال بحرانی عدد دیگری خواهد بود. در ضمن برای انجام محاسبات این بخش از تابع infiniteClusterProbabilityDisturb() که درون تابع percolation() تعریف شده است استفاده می‌کنیم.

در آخرین بخش از مسائل تراوش مربوط به این نوبت تمرین، می‌خواهیم احتمال اتصال به خوشه‌ی بی‌نهایت را بدست آوریم. در واقع می‌خواهیم بدانیم که اگر یک خانه را به صورت تصادفی انتخاب کنیم، چقدر احتمال دارد که به یک خوشه‌ی بی‌نهایت متصل باشد. نتیجه در زیر آمده است.