***گزارش تمرین هشتم***

***نام: محمد جمشیدی***

***شماره دانش‌جویی: 98100718***

شبیه‌سازی مدل آیزینگ

در این تمرین قصد داریم مدل آیزینگ دو بعدی را با استفاده از روش متروپولیس شبیه‌سازی کنیم. در واقع یک شبیه‌سازی مونت کارلو از مدل آیزینگ را می‌توان با تکرار مراحل زیر انجام داد (نمادهای لاتین بکار رفته در این گزارش کار نمادهای متعارف مکانیک آماری هستند).

1. ***از یک آرایش رندوم ابتدایی شروع می‌کنیم. در این مرحله یک آرایه‌ی دو بعدی از اعداد 1+ و 1- داریم که یکی از این اسپین‌ها را به صورت رندوم انتخاب می‌کنیم و جهت آن را برعکس می‌کنیم، یعنی مقدار خانه را در یک منفی ضرب می‌کنیم.***
2. ***اگر تغییر انرژی در شبکه منفی باشد تغییر جهت اسپین قبول می‌شود.***
3. ***اگر تغییر انرژی در شبکه نامنفی باشد در این صورت تغییر جهت اسپین با احتمال exp(-βΔE) پذیرفته می‌شود.***

بنابراین الگوریتم کلی که برای شبیه‌سازی استفاده شده است الگوریتم آشنا و مشخصی است. در ادامه باید این الگوریتم را پیاده‌سازی کنیم و تاحد امکان کد آن را بهینه کنیم.

برای پیاده سازی مدل، کلاس Ising\_2d\_model را تعریف می‌کنم. در تابع \_\_init\_\_ آبجیکتی که از این کلاس می‌سازیم را با دادن طول لتیس و مقدار بتا و همچنین تعداد نمونه‌برداری، مقداردهی اولیه می‌کنیم. توجه شود که به طور کلی در پیاده‌سازی مدل آیزینگ از واحدهای کاهیده استفاده کرده‌ایم. همچنین در این شبیه سازی از آنزامبل مکانیک آماری NVT (آنزامبل کانونیک) استفاده کرده‌ایم. در ادامه توابعی تعریف کرده‌ام که سیستم را تحول دهند و داده‌های مورد نیاز (انرژی، مغناطش و طول همبستگی مکانی اسپین‌ها) را جمع آوری کنند. توجه می‌کنیم که برای جمع‌آوری داده نیاز داریم سیستم را تا رسیدن به حالت تعادل آن تحول دهیم؛ پس باید برای به تعادل رسیدن سیستم هم عملیاتی را پیاده‌سازی کنیم. برای این منظور یک متد relax\_lattice تعریف کرده‌ام. این متد از توابعی که سیستم را به روش متروپولیس تحول می‌دهند استفاده می‌کند و این کار را برای تعداد num\_of\_samples=100 انجام می‌دهد. دقت شود که عدد 100 چیزی است که به عنوان پیش‌فرض برای این آرگومان تابع \_\_init\_\_ در نظر گرفته شده است. در این بین، با هر قدم متروپولیس مقدار انرژی سیستم را در یک لیست ذخیره می‌کند و سپس اگر خودهمبستگی این انرژی‌ها (j=0.1\* num\_of\_samples) کم‌تر از exp(-2) باشد، آنگاه سیستم به حالت تعادل رسیده است. در غیر این صورت سیستم به تعادل نرسیده است. این معنایش این است که تا رسیدن به تعادل فاصله‌ی بیش‌تری داریم و در نتیجه باید تعداد قدم‌های متروپولیس بیش‌تری برداشته شود. برای اینکار در صورتی که سیستم به تعادل نرسیده باشد، این بار num\_of\_samples دیگر تحول سیستم را تکرار می‌کند و دوباره وضعیت تعادل را بررسی می‌کند و اگر باز هم به تعادل نرسیده باشیم باز دوباره به همان تعداد، تکرار را ادامه می‌دهیم و این‌کار را تا وقتی که به تعادل برسیم تکرار می‌کنیم. در ادامه که سیستم سرانجام به تعادل رسید باید لیست انرژی‌هایی که داریم را بررسی کنیم تا ببینیم با چه‌گامی انرژی‌ها را انتخاب کنیم که تا حد امکان به هم همبستگی اندکی داشته باشند. البته این‌کار را می‌توان به جای بررسی کردن برای انر‌ژی با بررسی مغناطش هم انجام داد اما من در کد خود برای انرژی بررسی را انجام داده‌ام. پیاده سازی این عملیات توسط تابع find\_corr\_length انجام شده است. خروجی این تابع همان طول گام مناسب است که بعداً از آن برای جمع‌آوری داده‌های نهایی استفاده می‌کنیم.

همه‌ی فرایند توضیح داده شده را برای تعدادی طول شبکه و بتای مختلف تکرار می‌کنم. طول‌های شبکه را به صورت زیر گرفته‌ام.

length\_s = np.array([250, 184, 136, 100])

دلیل این‌که طول‌ها را به این صورت انتخاب کرده‌ام این است که به صورت نمایی از 100 تا 250 رشد می‌کنند(با ضریب 1.36). فایده‌ی این کار این است که در نهایت که قصد داریم بعضی نمودارهای لگاریتمی برحسب طول داشته باشیم، پراکندگی نقاط یکنواخت خواهد بود.

بتاهای مورد نظر را هم به صورت زیر انتخاب کرده‌ام.

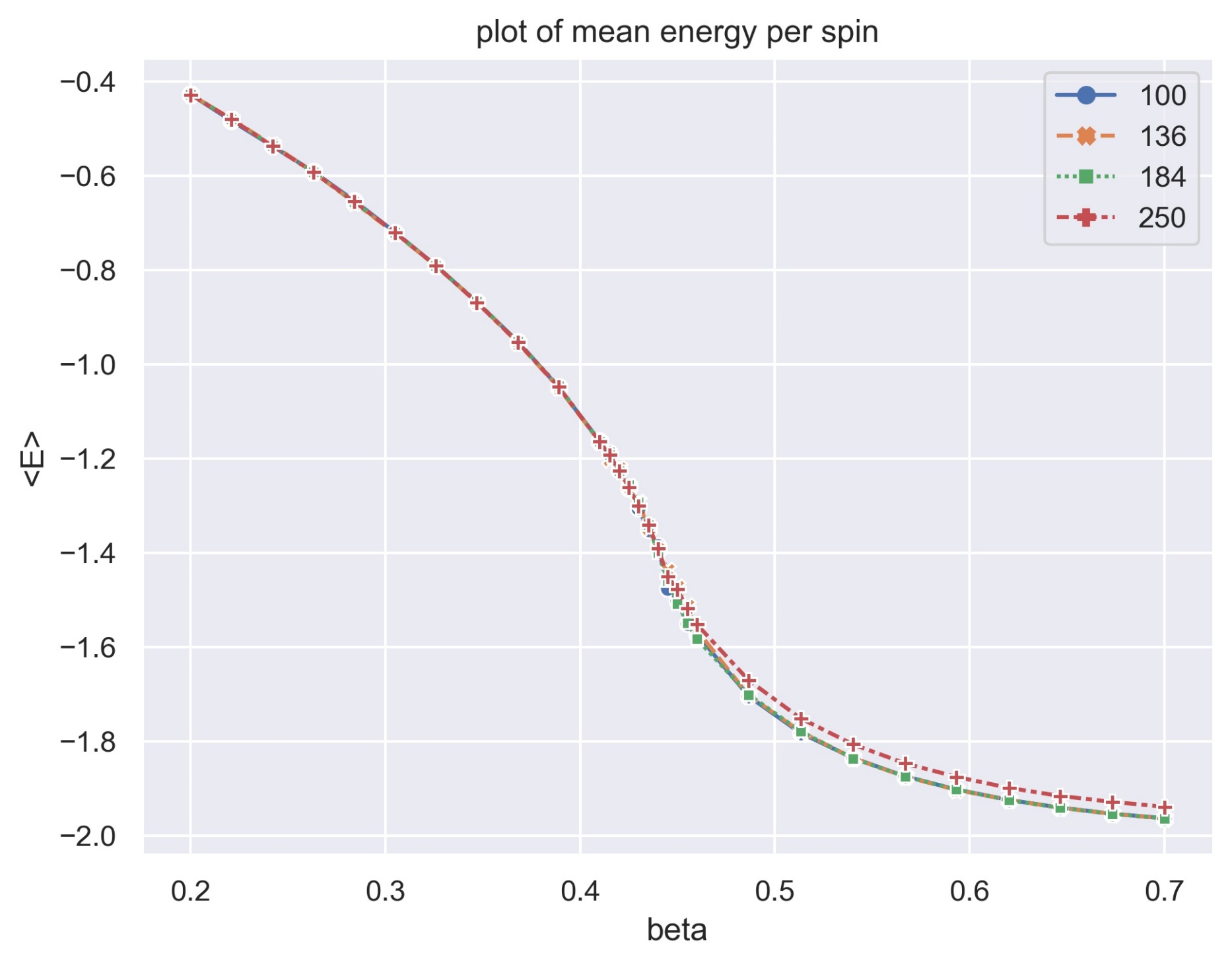
beta\_s = np.concatenate((np.linspace(0.2, 0.41, 10, endpoint=False, dtype=np.float16),

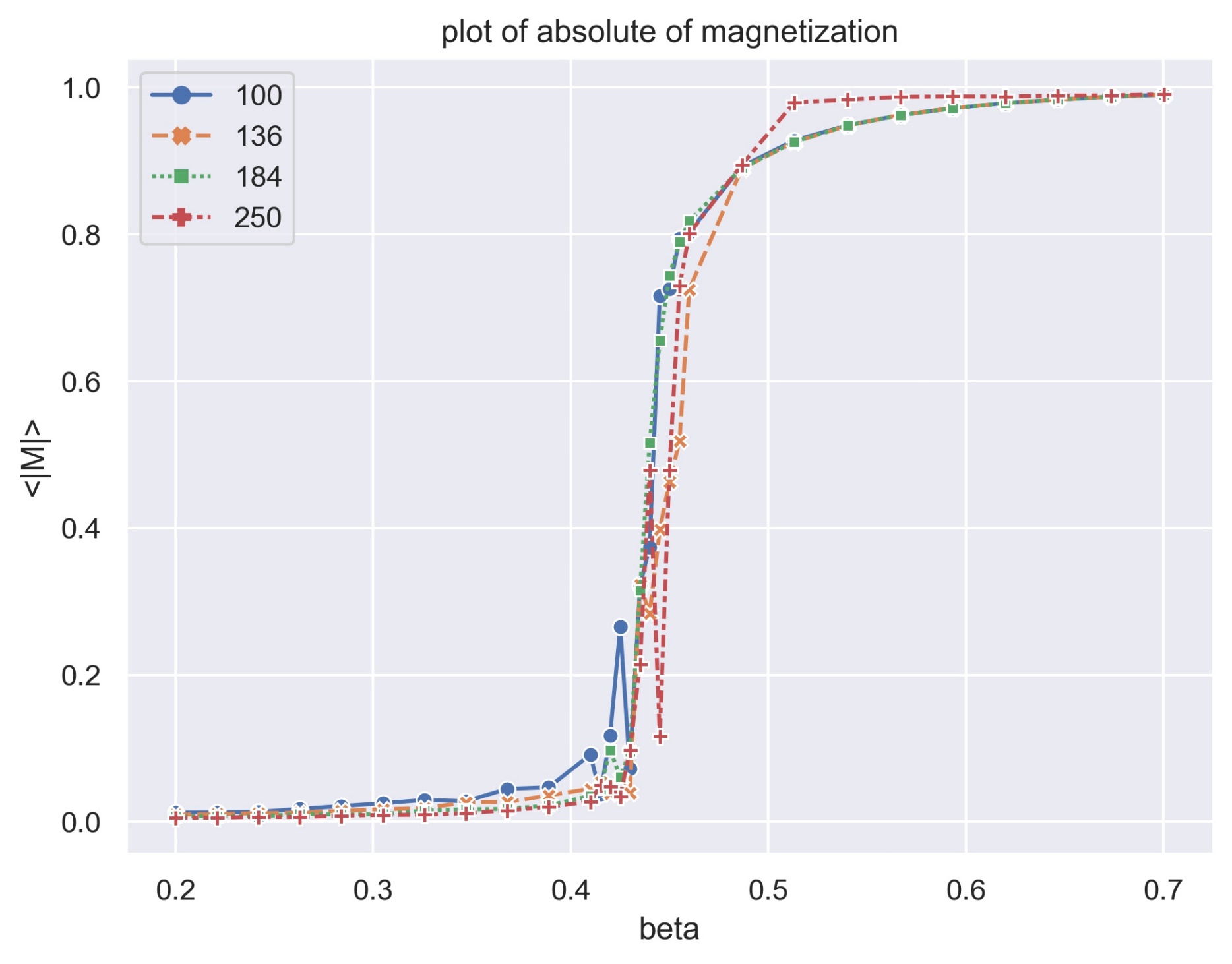
np.linspace(0.41, 0.46, 10, endpoint=False, dtype=np.float16),

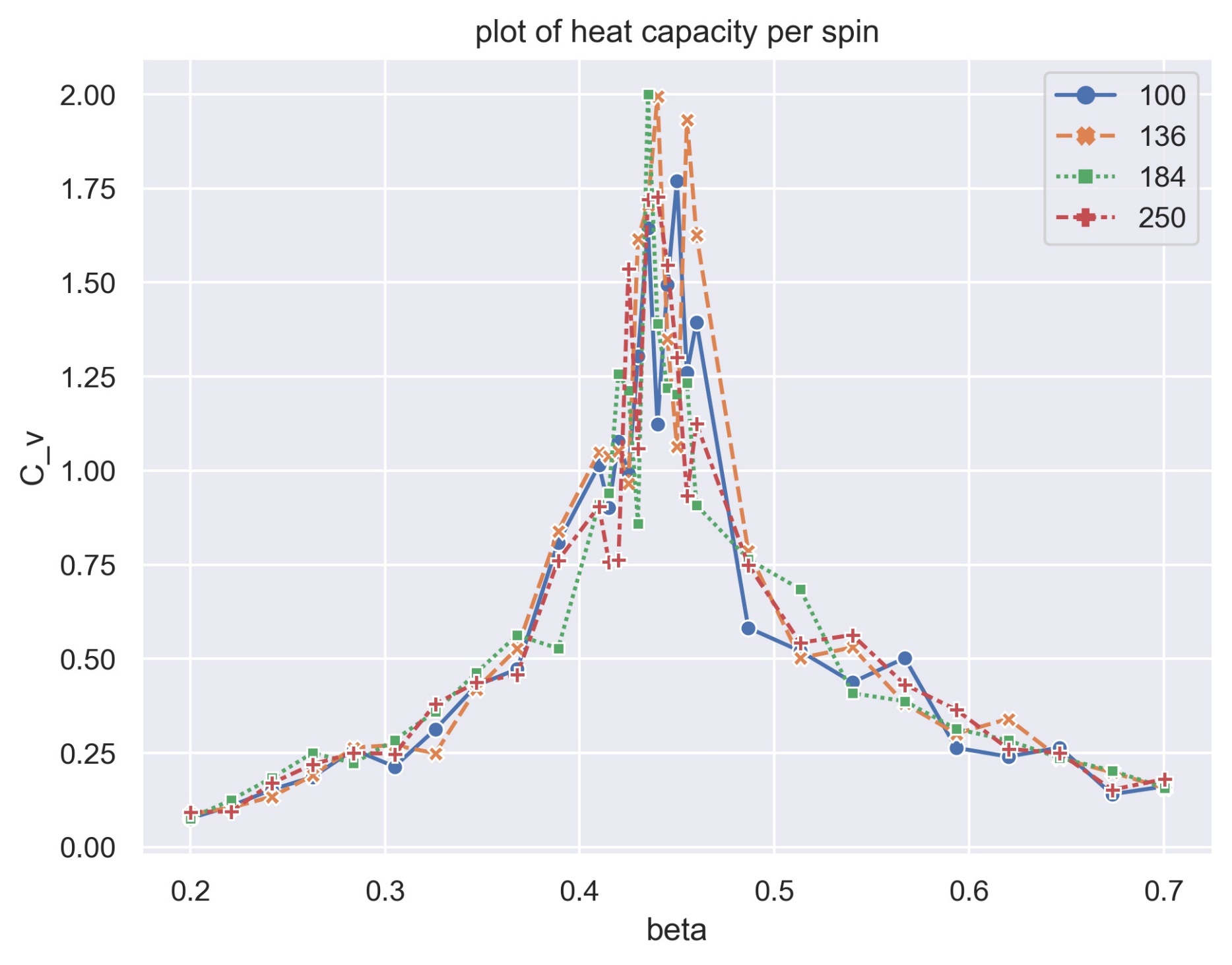
np.linspace(0.46, 0.7, 10, endpoint=True, dtype=np.float16))

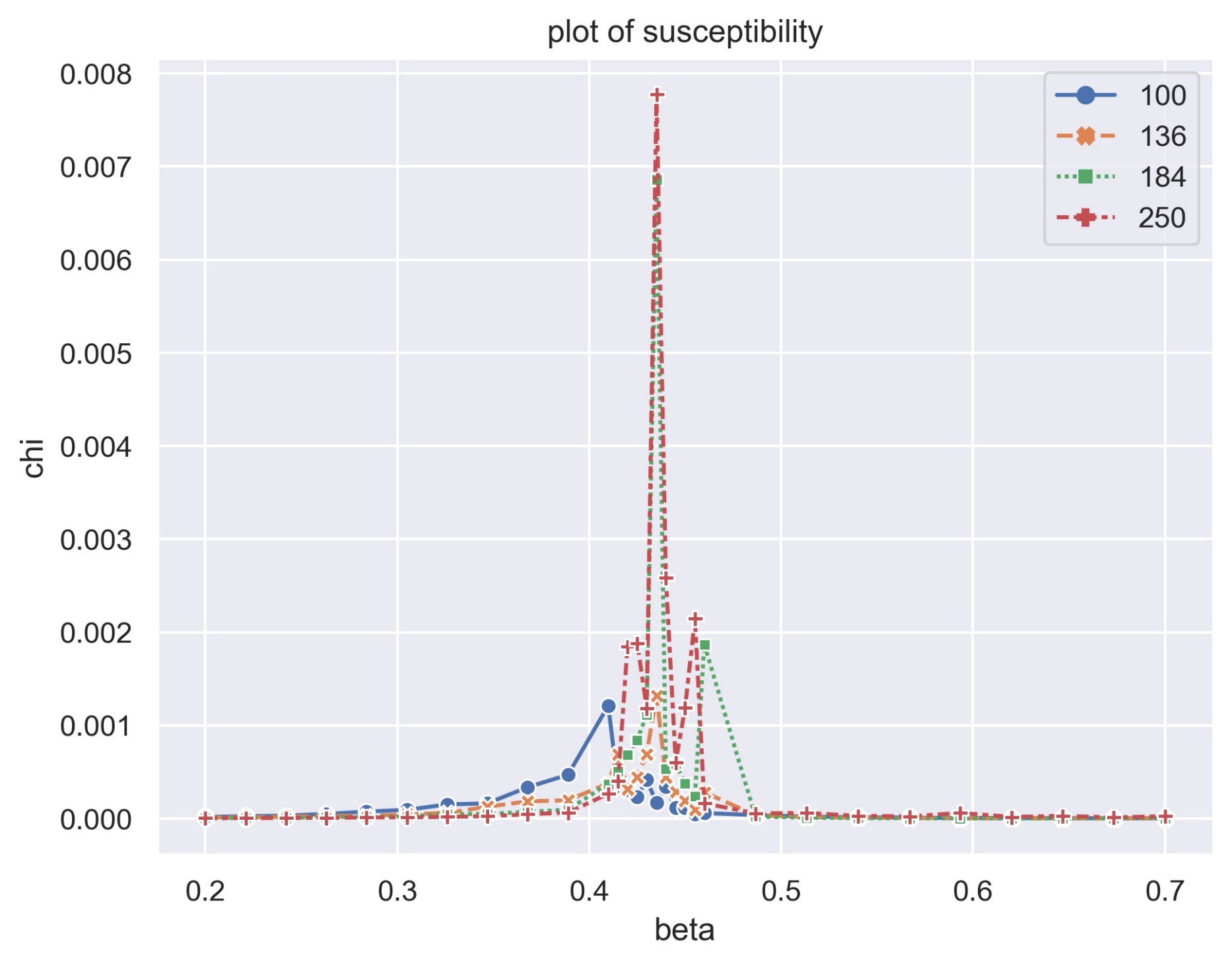
همان‌طور که دیده می‌شود طول بازه‌ها را حول و حوش مقدار بتای بحرانی (که از تئوری می‌دانیم حدوداً 0.44 است) کوچک‌تر گرفته‌ام تا در این محدوده‌ی بحرانی داده‌ی بیش‌تری داشته باشیم. یک نکته‌ی دیگر که ذکر آن خالی از لطف نیست این است که در راستای کم‌تر شدن زمان اجراها، هربار که سیستم به ازای بتای مثلاً n ام به تعادل رسید، این وضیعت سیستم را به عنوان شرایط اولیه برای بتای n+1 ام در نظر می‌گیریم به جای آن‌که دوباره از یک شرایط اولیه‌ی کاتوره‌ای شروع کنیم. این موضوع باعث کم‌تر شدن زمان رسیدن به تعادل می‌شود. همه‌ی داده‌های نهایی به دست آمده به صورت فایل‌های .csv ذخیره شده است. در نهایت با استفاده از تابع load\_and\_plot نمودار کمیت‌های ترمودینامیکی خواسته شده برحسب بتا را می‌توانیم رسم کنیم.

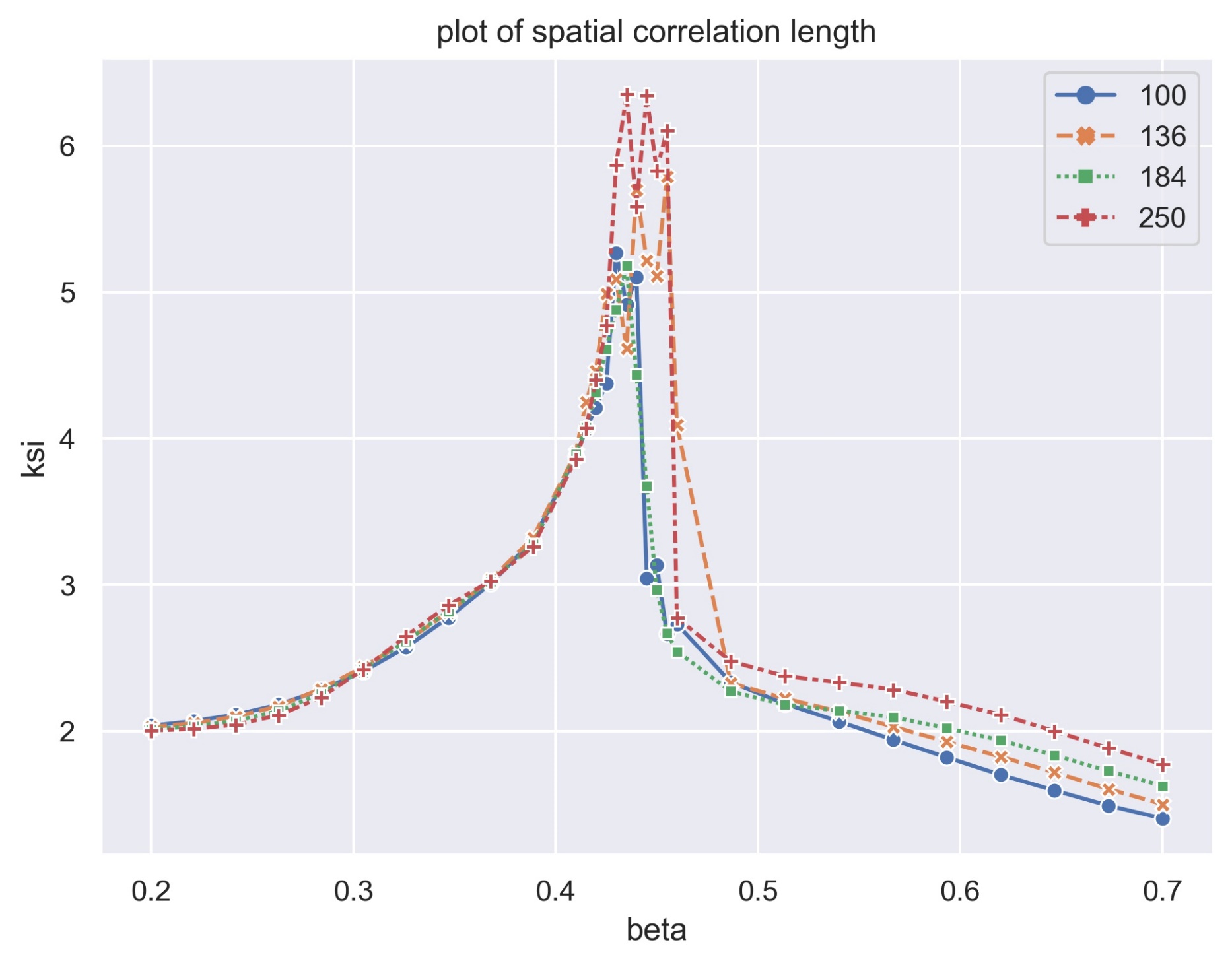
در ادامه نمودارهای کمیت‌های ترمودینامیکی خواسته شده آمده است.







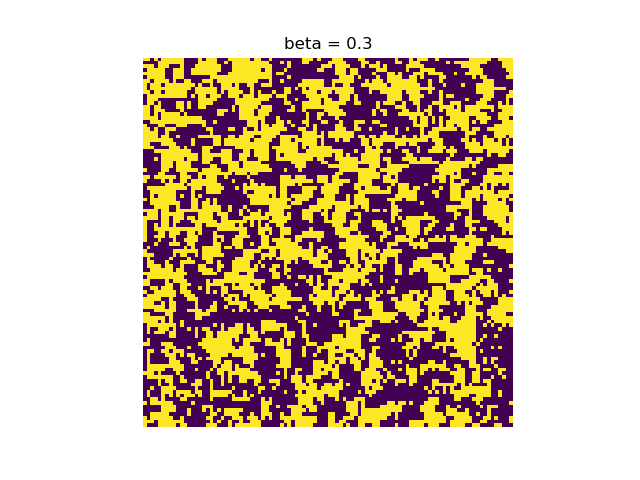


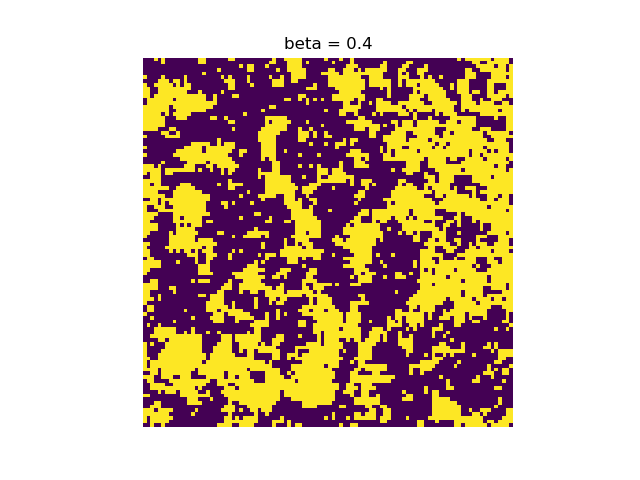


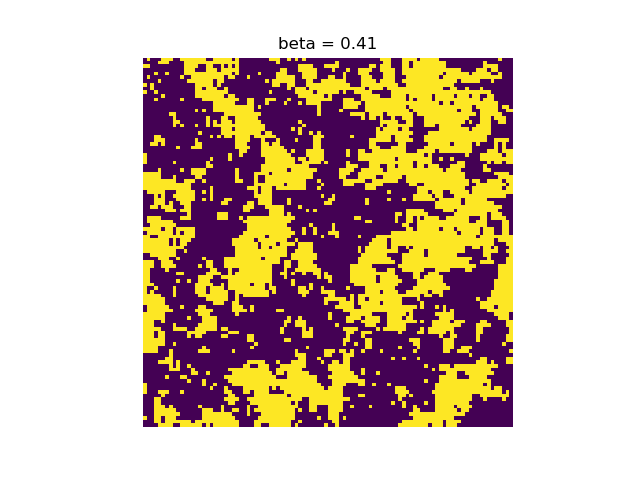
ناگفته نماند که جعبه خطاهای مربوط به نمودارهای بالا را هم در کد حساب کرده‌ و ذخیره کرده‌ام. اما در نمودار نمایش ندادم چون به نظرم خیلی نمودار شلوغ می‌شد! در ادامه برای بدست آوردن نماهای بحرانی، از داده‌هایی که در فایل‌ها گرد آورده‌ام و رسم نمودار لگ-لگ آن‌ها حول نقطه‌ی بحرانی برحسب طول شبکه استفاده می‌کنیم (N تعداد اسپین‌های شبکه است). برای خواندن داده‌ی بیشینه، به صورت دستی و با یک جستجوی ساده در فایل مربوطه داده‌خوانی کرده‌ام. نتیجه به صورت زیر است.

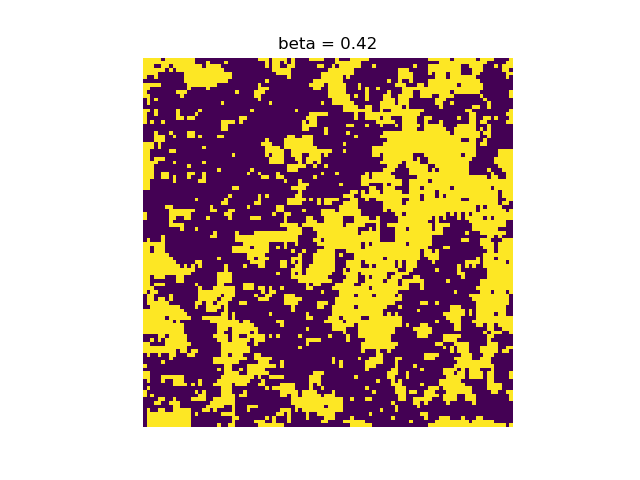
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Critical Exponents | | | Theoretical Relation |
| Simulation value | Theoretical value |  |
| *2.273* | 0.84 | 1 | *ν* |  |
| *2.247* | -0.37 | 0.44 |  |  |
| *2.273* | 1.96 | 1.75 | *γ* |  |
| *2.222* | 0.19 | 0.125 | *β* |  |

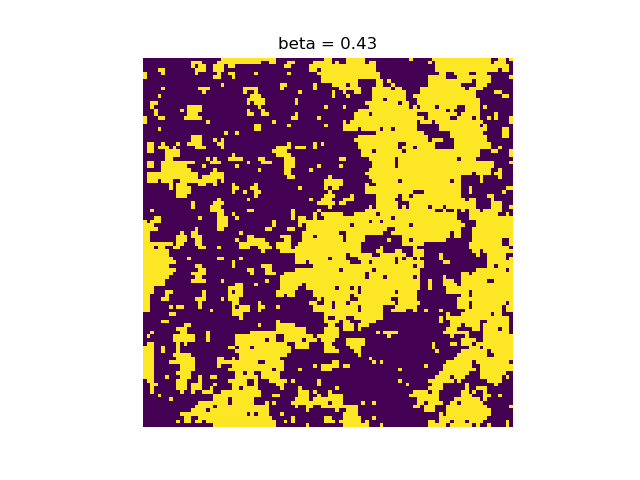
در ادامه نمایش گرافیکی شبکه را برای چند مقدار بتا و طول لتیس 100 آورده‌ام.

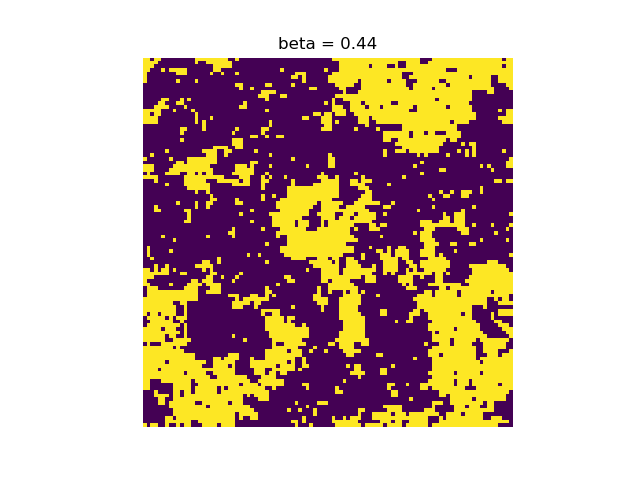


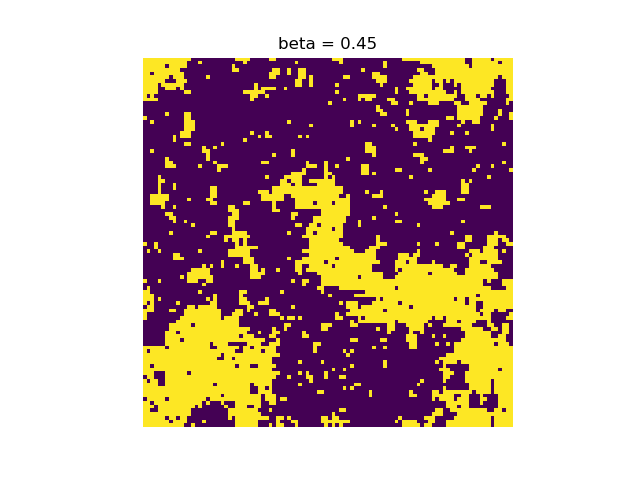


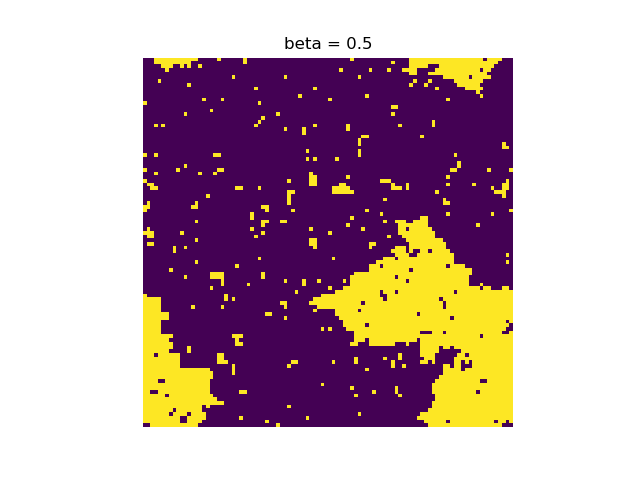


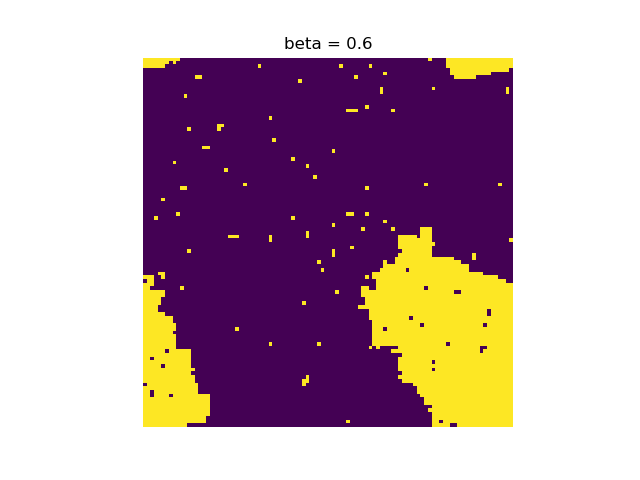












ناگفته نماند که به طور کلی از نمونه‌گزارش‌ها و کدهای مربوطه در گیت‌هاب دستیاران هم بهره گرفتم و نکات آموزنده‌ای دستگیرم شد.