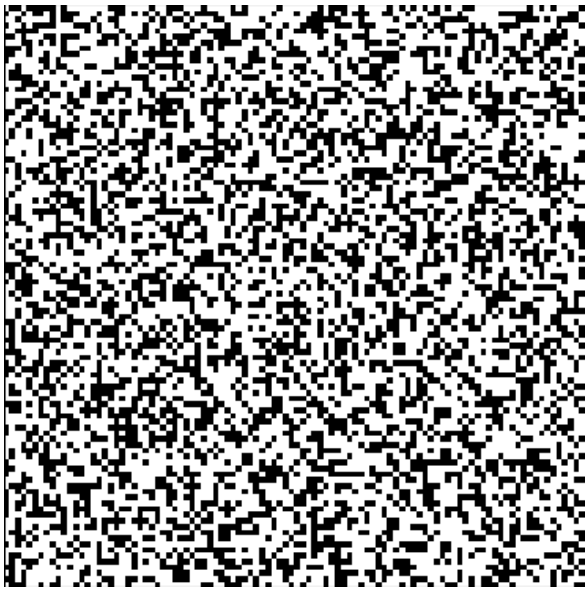
شبیه‌سازی رایانه‌ای در فیزیک
تمرین سوم: تراوش

سینا معمر ۹۵۱۰۲۳۱۶

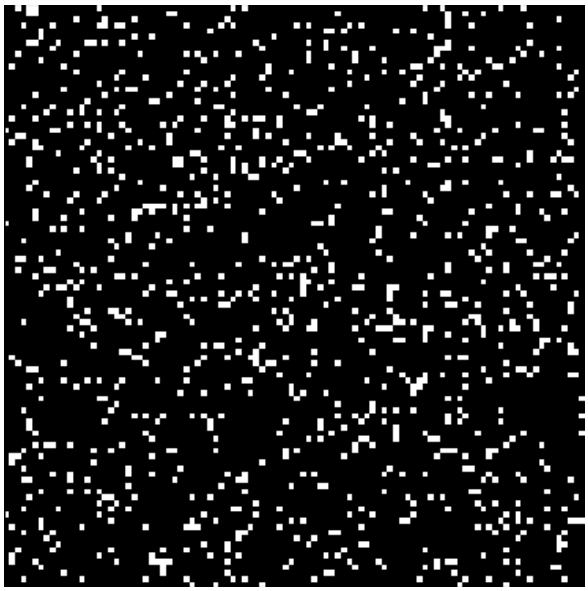
۱۸ شهریور ۱۴۰۰

۱ تراوش

کد این بخش از تمرین را در فایل q1.py می‌توان مشاهده نمود. در ابتدا لازم است که یک object از کلاس Percolation به طول دل‌خواه بسازیم. سپس تابع render را با مقدار احتمال موردنظر صدا می‌کنیم. روش کار این تابع به این صورت است که یک آرایه دو بعدی از اعداد تصادفی به طول داده شده می‌سازد. سپس این آرایه را با احتمال داده شده مقایسه می‌کند و خانه‌هایی که احتمال کم‌تر داشته باشند را روشن می‌کند. در آخر نیز برای نمایش، تابع show را فراخوانی می‌کنیم و با استفاده از تابع pcolormesh آرایه دو بعدی‌مان را نمایش می‌دهیم.
شکل به دست آمده برای شبکه‌ای به طول 100 و احتمال‌های 0.4 و 0.9 را در شکل‌های ۱ و ۲ می‌توان مشاهده نمود. همان‌طور که دیده می‌شود برای احتمال 0.9 تراوش رخ داده است و خوشه‌ی بی نهایت داریم.



شکل ۱: شبکه‌ی تراوش به طول 100 و احتمال 0.4

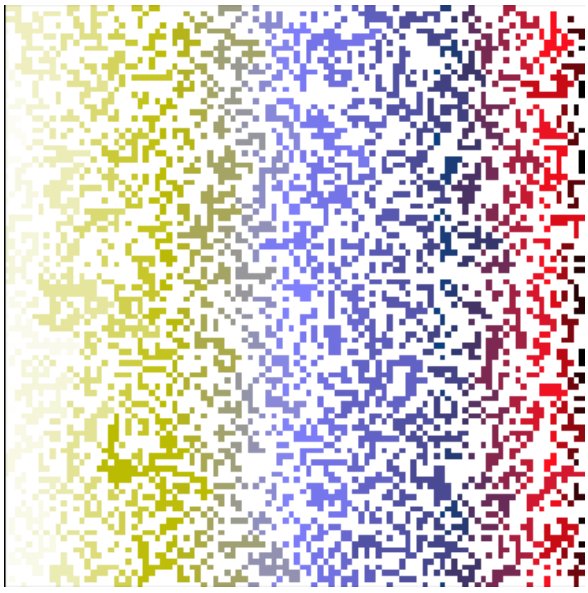


شکل ۲: شبکه‌ی تراوش به طول 100 و احتمال 0.9

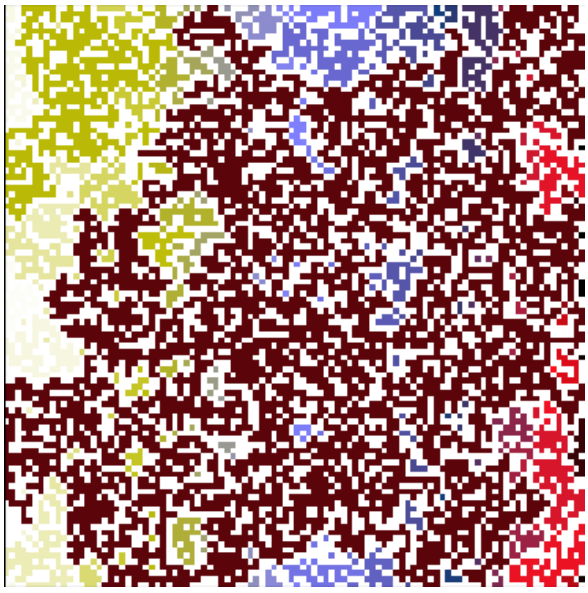
۲ الگوریتم هشن-کپلمن

کد این بخش از تمرین را در فایل q2.py می‌توان مشاهده نمود. در ابتدا باید یک object از کلاس Percolation بسازیم. سپس تابع render را با مقدار احتمال دل‌خواه صدا می‌کنیم. روش کار این تابع به این صورت است که یک

آرایه دو بعدی به طول داده شده از اعداد تصادفی می‌سازد. سپس آرایه به دست آمده را با احتمال داده شده مقایسه می‌کند و خانه‌هایی که احتمال کم‌تر داشته باشند را روشن می‌کند. بعد تعداد خانه‌های روشن را می‌شمارد و دو آرایه labels و sizes را به آن طول می‌سازد. سپس روی تمام خانه‌های آرایه‌ی دو بعدی‌مان پیمایش می‌کند و شماره‌ی خانه‌های بالا و چپ را می‌خواند. حال اگر تنها خانه‌ی بالا و یا چپ خالی باشد، لیبل آن خانه را به آن می‌دهد. در صورتی هم که هر دو پر باشند، لیبل خانه‌ی چپ را به هر دو می‌دهد. ولی باید لیبل نهایی آن خانه را پیدا کند و این کار را با استفاده از تابع find_label__ انجام می دهد. روش کار این تابع به این شکل است که مقدار لیبل را از آرایه‌ی labels به دست می‌آورد. اگر این مقدار برابر با شماره خود لیبل باشد، که همان مقدار نهایی خواهد بود، در غیر این صورت همین تابع را با مقدار به دست آمده صدا می‌کند، تا در نهایت به جایی برسد که مقدار لیبل با شماره‌ی خانه یکی شود.
برای پیدا کردن خوشه‌ی بی‌نهایت، لیبل خانه‌های ستون اول و آخر را برای اشتراک چک می‌کنیم. همین کار را برای خانه‌های ردیف اول و آخر انجام می‌دهیم. در صورت وجود اشتراک، یعنی تراوش رخ داده است. برای این کار تابع infinity_clusters را صدا باید بکنیم. خروجی این تابع لیبل خوشه‌ی بی‌نهایت و اندازه‌ی آن است.
برای پیدا طول هم‌بستگی نیز باید تابع correlation_length را صدا بزنیم. برای به دست آوردن طول هم‌بستگی، روی خانه‌های آرایه‌مان پیمایش می‌کنیم و مختصات خانه‌هایی که لیبل یکسان دارند را در یک آرایه‌ی جدید ذخیره می‌کنیم. سپس لیبل خوشه‌ی بی‌نهایت را از تابع infinity_clusters به دست می‌آوریم و آن را از محاسبات‌مان خارج می‌کنیم. سپس اندیس بزرگ‌ترین خوشه‌ی باقی مانده را پیدا می‌کنیم و انحراف معیار مختصات خانه‌های آن لیبل را به دست می‌آوریم و به عنوان خروجی بر می‌گردانیم. برای نمایش شبکه‌ی تراوش نیز باید تابع show را فراخوانی کنیم.
نتیجه‌ی به دست آمده برای شبکه‌ای به طول 100 و احتمال‌های 0.4 و 0.6 را در شکل‌های ۳ و ۴ می‌توان مشاهده نمود. همان‌طور که دیده می‌شود، برای احتمال 0.6 تراوش رخ داده است.



شکل ۳: شبکه‌ی تراوش به طول 100 و احتمال 0.4

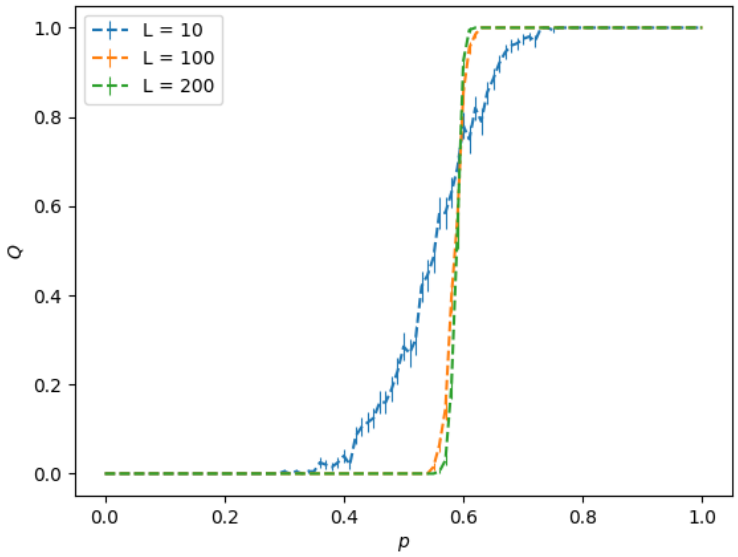


شکل ۴: شبکه‌ی تراوش به طول 100 و احتمال 0.6

۳ احتمال ایجاد خوشه‌ی بی‌نهایت برای شبکه‌ی محدود

کد این بخش از تمرین را در فایل q3.py می‌توان مشاهده نمود. برای به دست آوردن نتایج خواسته شده تابع find_q را باید صدا بزنیم. روش کار این تابع به این صورت است که روی طول‌های داده شده و سپس روی احتمال‌های داده شده، پیمایش می‌کند و object ساخته شده از کلاس Percolation را با آن طول و احتمال به اندازه تعداد نمونه‌های خواسته شده، صدا می‌کند. سپس تابع is_percolated را فراخوانی می‌کند و در صورتی که تراوش رخ داده باشد، خانه‌ی متناظر با آن در آرایه‌ی داده‌ها را 1 می‌کند. نتایج به دست آمده در یک آرایه ذخیره می‌شوند و آن را در نهایت

در یک فایل به فرمت npy. برای استفاده‌های بعدی ذخیره می‌کند. سپس میانگین و انحراف معیار داده‌های به دست آمده را محاسبه و رسم می‌کنیم. نتیجه‌ی به دست آمده را در شکل ۵ می‌توان مشاهده نمود.



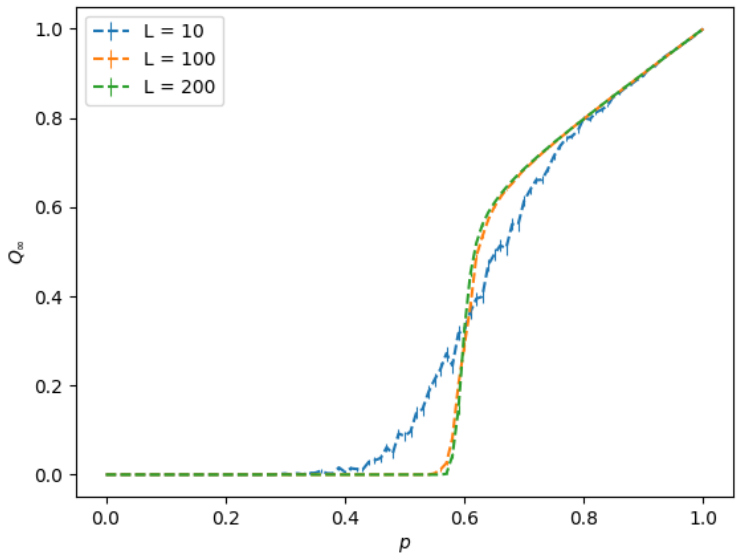
شکل ۵: احتمال تراوش بر حسب p برای شبکه‌هایی به طول 10، 100 و 200 و با 200 بار تکرار

۴ احتمال اتصال به خوشه‌ی بی‌نهایت

کد این بخش از تمرین را در فایل q4.py می‌توان مشاهده نمود. برای به دست آوردن نتایج خواسته شده تابع find_q_inf را باید صدا بزنیم. روش کار این تابع به این صورت است که روی طول‌ها و سپس روی احتمال‌های داده شده، پیمایش می‌کند و object ساخته شده از کلاس Percolation را با آن طول و احتمال به اندازه‌ی تعداد نمونه‌های خواسته شده، صدا می‌کند. سپس تابع infinity_clusters را فراخوانی می‌کند و در صورتی که تراوش رخ داده باشد، اندازه خوشه‌ی بینهایت را در خانه‌ی متناظر با آن در آرایه‌ی داده‌ها را ذخیره می‌کند. نتایج به دست آمده در یک آرایه ذخیره می‌شوند و آن را در نهایت در یک فایل به فرمت npy. برای استفاده‌های بعدی ذخیره می‌کنیم. سپس میانگین و انحراف معیار داده‌های به دست آمده را محاسبه و رسم می‌کنیم. نتیجه‌ی به دست آمده را در شکل ۶ می‌توان مشاهده نمود.

۵ طول هم‌بستگی

کد این بخش از تمرین را در فایل q5.py می‌توان مشاهده نمود. برای به دست آوردن نتایج خواسته شده تابع find_correlation_radius را باید صدا بزنیم. روش کار این تابع به این صورت است که روی طول‌ها و سپس روی احتمال‌های داده شده، پیمایش می‌کند و object ساخته شده از کلاس Percolation را با آن طول و احتمال به تعداد نمونه‌های خواسته شده، صدا می‌کند. سپس تابع correlation_length را فراخوانی می‌کند و مقدار به دست آمده را در خانه‌ی متناظر با آن در آرایه‌ی داده‌ها ذخیره می‌کند. نتایج به دست آمده در یک آرایه ذخیره می‌شوند و آن را در نهایت در یک فایل به فرمت npy. برای استفاده‌های بعدی ذخیره می‌کنیم. سپس میانگین و انحراف معیار داده‌های به دست آمده را محاسبه و رسم می‌کنیم. نتیجه‌ی به دست آمده را در شکل ۷ می‌توان مشاهده نمود.
همان‌طور که در شکل دیده می‌شود، قله‌ی منحنی‌ها در حدود بازه‌ی $(0.5, 0.6)$ قرار دارند. برای اینکه بتوانیم مقدار p_c را برای هر کدام با دقت تعیین کنیم، برای هر طول در یک بازه‌ی محدود و با گام‌های کوچک و تعداد نمونه‌های



شکل ۶: احتمال اتصال به خوشه‌ی بی‌نهایت بر حسب p برای شبکه‌هایی به طول 10، 100 و 200 و با 200 بار تکرار

زیاد، محاسبات قبلی را تکرار می‌کنیم. نتایج به دست آمده را در شکل‌های ۸ تا ۱۲ می‌توان مشاهده نمود. مقدار p_c های به دست آمده از روی این نمودارها در جدول ۱ ثبت شده است.

طول شبکه	10	20	40	80	160
$p_c \pm 0.001$	0.497	0.539	0.563	0.573	0.581

جدول ۱: p_c بر حسب طول شبکه

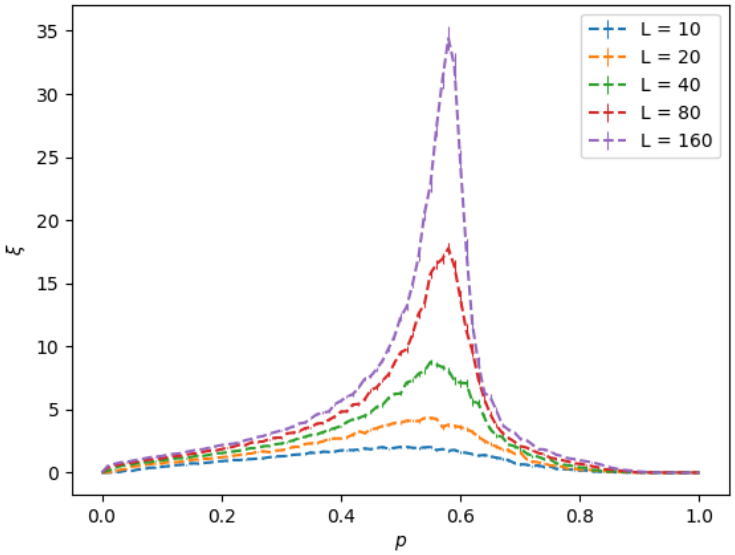
برای به دست آوردن $p_c(\infty)$ از طریق برون‌یابی مقادیر به دست آمده برای طول‌های محدود، می توان از رابطه‌ی (۲) استفاده نمود. در این رابطه 3 پارامتر A، ν و $p_c(\infty)$ مجهول هستند. پس تابع مدل آن‌ها را می‌سازیم و با استفاده از تابع curve_fit از کتاب‌خانه scipy بهترین مقدار آن‌ها را حساب می‌کنیم. مقدار به دست آمده برای $p_c(\infty)$ برابر است با:

$$p_c(\infty) = 0.5932 \pm 10^{-4} \tag{۱}$$

$$\begin{aligned} &|p_c(L) - p_c(\infty)| \sim L^{-\frac{1}{\nu}} \\ &\Rightarrow L^{-\frac{1}{\nu}} = A|p_c(L) - p_c(\infty)| \\ &\Rightarrow L = (A|p_c(L) - p_c(\infty)|)^{-\nu} \end{aligned} \tag{۲}$$

۶ نمای بحرانی ν

کد این بخش از تمرین را در فایل q6.py می‌توان مشاهده نمود. همان‌طور که از رابطه‌ی (۴) دیده می‌شود، با رسم منحنی $|p_c(L) - p_c(\infty)|$ بر حسب L به صورت تمام لگاریتم، می‌توانیم از روی شیب خط فیت شده، مقدار نمای ν



شکل ۷: طول هم‌بستگی بر حسب p برای طول‌های 10، 20، 40، 80 و 160 و با 200 بار تکرار

را به دست بیاوریم. نمودار به دست آمده را در شکل ۱۳ می‌توان مشاهده نمود. مقدار حاصل برابر است با:

$$\nu = 1.331 \pm 0.001 \tag{۳}$$

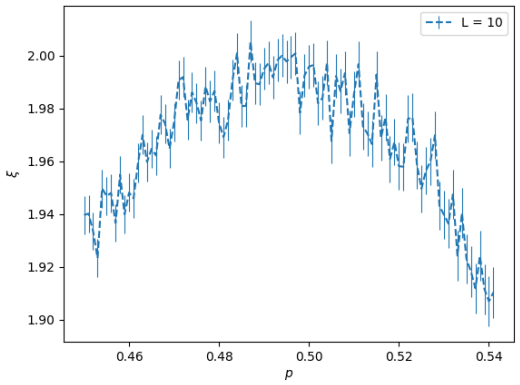
$$\begin{aligned} &|p_c(L) - p_c(\infty)| \sim L^{-\frac{1}{\nu}} \\ &\Rightarrow \ln(|p_c(L) - p_c(\infty)|) = A - \frac{1}{\nu}\ln(L) \end{aligned} \tag{۴}$$

۷ بعد جرمی (فراکتالی) خوشه‌های بی‌نهایت

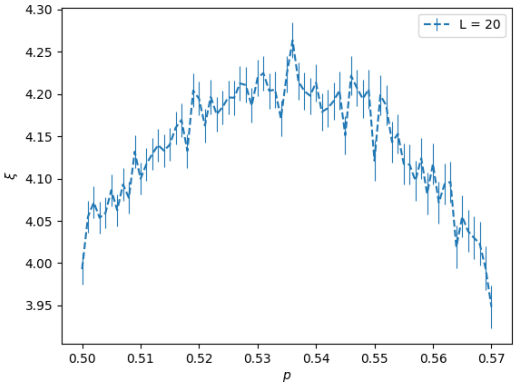
کد این بخش از تمرین را در فایل q7.py می‌توان مشاهده نمود. در ابتدا باید یک object از کلاس Percolation با طول دل‌خواه بسازیم. سپس تابع render را با احتمال موردنظر فراخوانی می‌کنیم. شیوه‌ی کار این تابع به این صورت است که ابتدا یک آرایه‌ی دو بعدی به طول داده شده می‌سازد. سپس آرایه‌ی active_cells را ایجاد می‌کند و مختصات خانه‌ی مرکزی شبکه را به آن اضافه می‌کند. وظیفه‌ی این آرایه آن است که مختصات خانه‌های جدیدی که در هر مرحله روشن می‌شوند را در خود نگه دارد. پس تا زمانی که طول این آرایه غیر صفر باشد، باید به رشد دادن خوشه ادامه دهد. در هر مرحله‌ی رشد، روی تمام خانه‌های آرایه‌ی active_cells پیمایش می‌کند و همسایه‌های آن‌ها را در یک آرایه‌ی جدید ذخیره می‌کند. از آن جایی که این همسایه‌ها می‌توانند اشتراک داشته باشند، در انتها آن‌ها را یکتا می‌کند. سپس با توجه به احتمال داده شده، این همسایه‌ها را روشن می‌کند و باقی را غیرفعال می‌کند. در آخر نیز آرایه‌ی active_cells را برابر با لیست همسایه‌های جدید روشن قرار می‌دهد.
برای انجام محاسبات خواسته شده در سوال، باید تابع calc_size_radius را صدا بزنیم. در ابتدا برای شبکه‌ای به طول 100 و با احتمال‌های 0.5، 0.55 و 0.59 خوشه‌ها را تولید می‌کنیم و نمایش می‌دهیم. شبکه‌های به دست آمده را در شکل‌های ۱۴، ۱۵ و ۱۶ می‌توان مشاهده نمود.
سپس برای احتمال‌های 0.5 تا 0.59 و با گام‌های 0.1 طول هم‌بستگی و اندازه‌ی خوشه را به دست می‌آوریم و آن‌ها را در خانه‌ی متناظر با آرایه‌ی داده‌مان ذخیره می‌کنیم. در نهایت نیز این آرایه را در یک فایل با فرمت npy. برای استفاده‌های بعدی ذخیره می‌کنیم. سپس میانگین و انحراف معیار داده‌ها را محاسبه کرده و منحنی‌های ۱۷، ۱۸ و

۱۹ را رسم می‌کنیم.
همان‌طور که در شکل ۱۹ دیده می‌شود، می‌توان یک خط بر داده‌های به دست آمده، فیت کرد. شیب این خط برابر است با بعد جرمی خوشه‌ی تراوش:

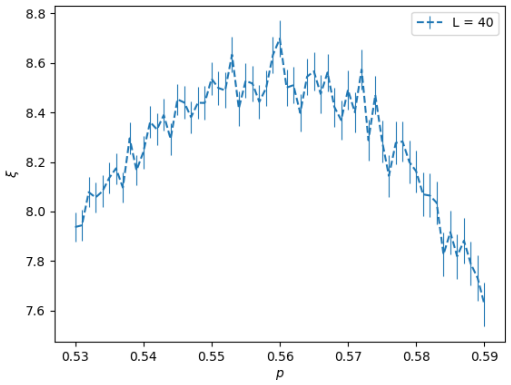
$$D = 1.9801 \pm 10^{-4} \quad (۵)$$



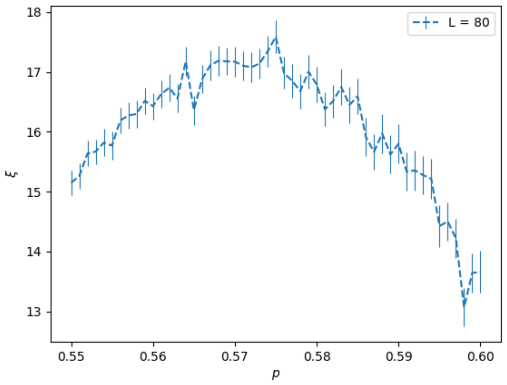
شکل ۸: طول هم‌بستگی بر حسب p برای طول 10 و گام 0.001 و با 6000 بار تکرار



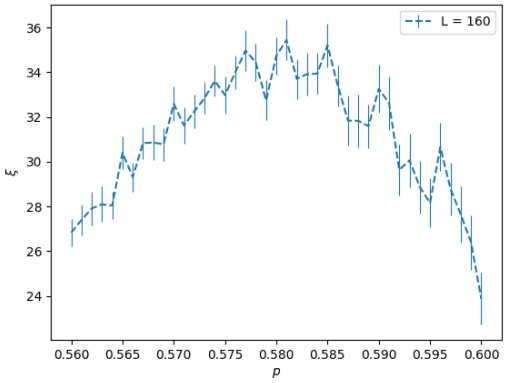
شکل ۹: طول هم‌بستگی بر حسب p برای طول 20 و گام 0.001 و با 3000 بار تکرار



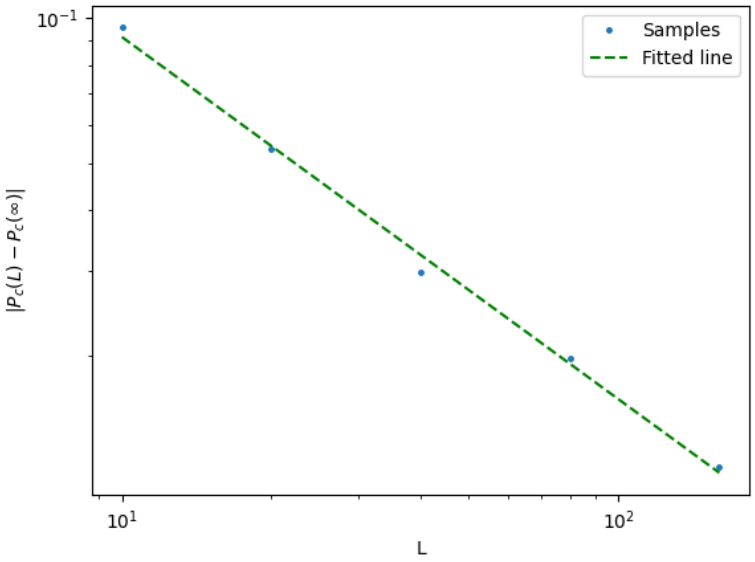
شکل ۱۰: طول هم‌بستگی بر حسب p برای طول 40 و گام 0.001 و با 1000 بار تکرار



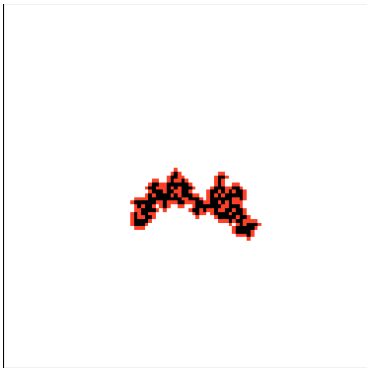
شکل ۱۱: طول هم‌بستگی بر حسب p برای طول 80 و گام 0.001 و با 300 بار تکرار



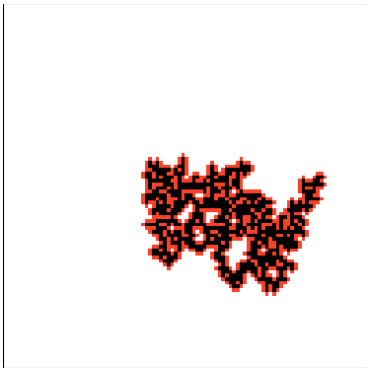
شکل ۱۲: طول هم‌بستگی بر حسب p برای طول 160 و گام 0.001 و با 100 بار تکرار



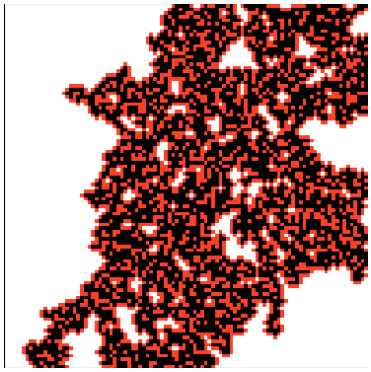
شکل ۱۳: $|p_c(L) - p_c(\infty)|$ بر حسب طول شبکه برای طول‌های 10، 20، 40، 80 و 160



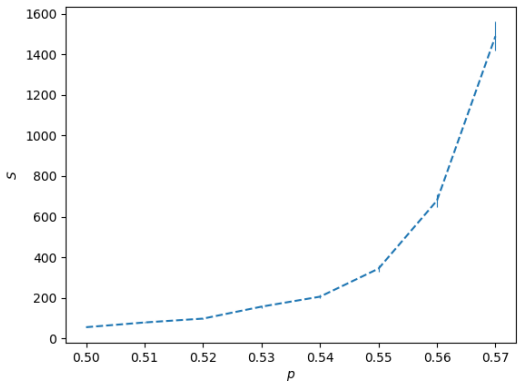
شکل ۱۴: خوشه‌ی تراوش برای شبکه‌ای به طول 100 و احتمال 0.50



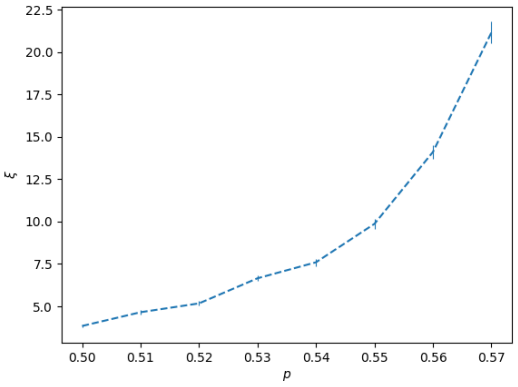
شکل ۱۵: خوشه‌ی تراوش برای شبکه‌ای به طول 100 و احتمال 0.55



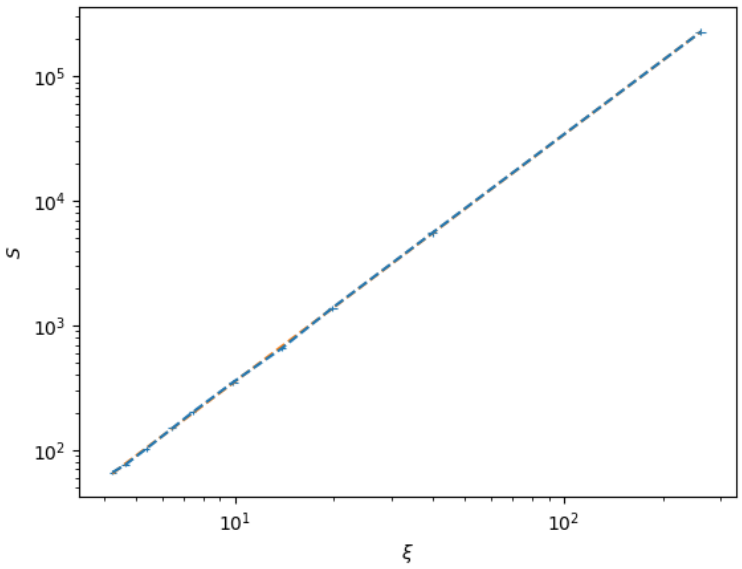
شکل ۱۶: خوشه‌ی تراوش برای شبکه‌ای به طول 100 و احتمال 0.59



شکل ۱۷: منحنی اندازه‌ی خوشه بر حسب p برای شبکه‌ای به طول 10 000 و با 1000 بار تکرار



شکل ۱۸: منحنی طول هم‌بستگی خوشه بر حسب p برای شبکه‌ای به طول 10 000 و با 1000 بار تکرار



شکل ۱۹: منحنی اندازه‌ی خوشه بر حسب طول هم‌بستگی برای شبکه‌ای به طول 10 000 و و با 1000 بار تکرار