گزارش تمرین چهارم

نام: محمد جمشیدی

شماره دانش‌جویی: ۹۸۱۰۰۷۱۸

۱. طول همبستگی (۵–۴) و نمای بحرانی (۶–۴)

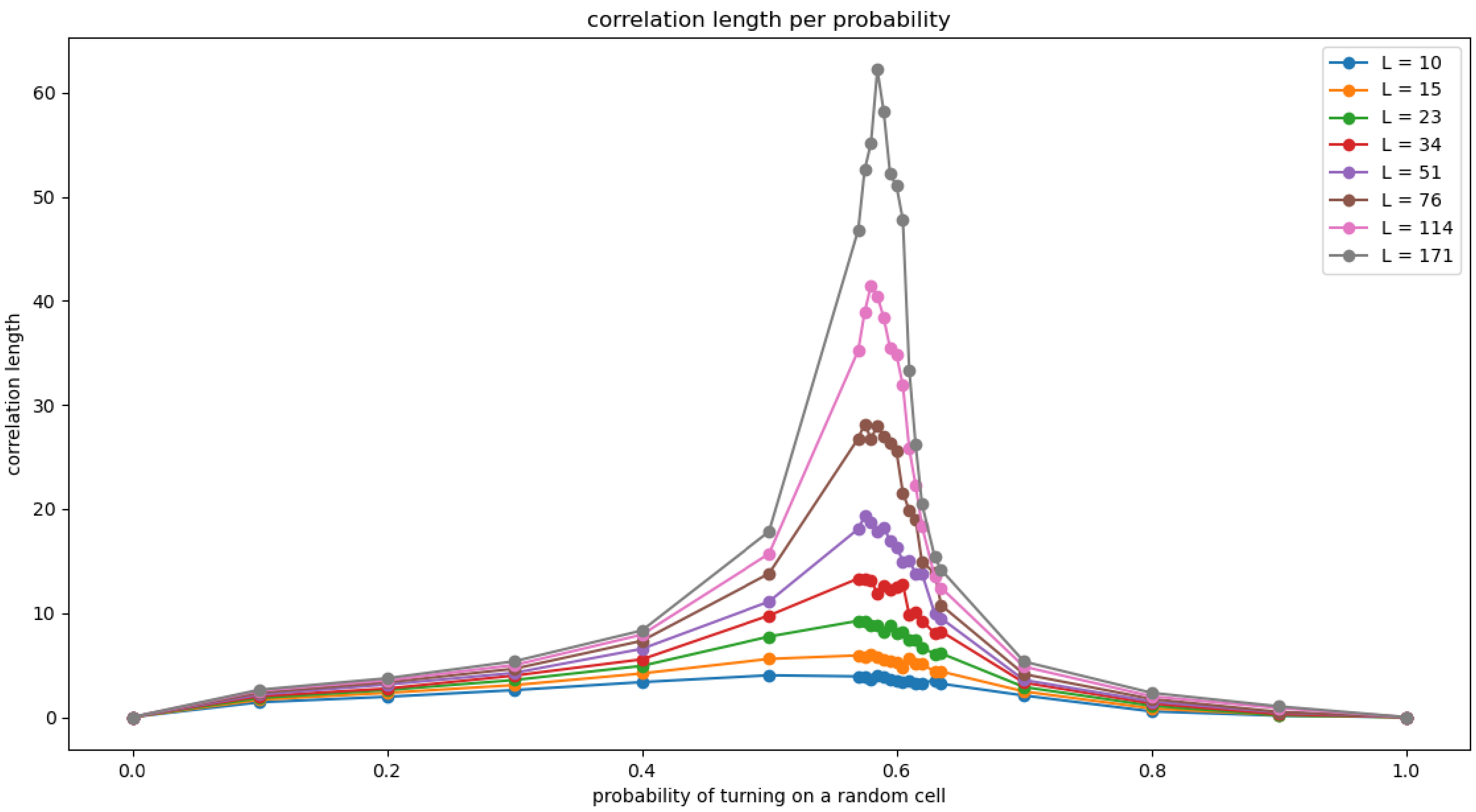
در این سوال می‌خواهیم طول همبستگی را در احتمال‌های متفاوت بررسی کنیم. معیاری که برای طول همبستگی در نظر گرفته‌ام، جذر مساحت بزرگ‌ترین خوشه‌ی غیر بی‌نهایت است. مساحت خوشه هم که در واقع همان تعداد خانه‌های خوشه است. انتظار داریم که با نزدیک شدن احتمال روشن کردن خانه‌ها به احتمال بحرانی، طول هم‌بستگی به بی‌نهایت میل کند. دلیل این موضوع هم این است که در این احتمال‌ها اندازه‌ی خوشه‌ها بزرگ‌تر می‌شود. به ازای احتمال‌های خیلی کوچک‌تر اندازه‌ی خوشه‌ها کوچک‌تر است و به ازای احتمال‌های خیلی بزرگ‌تر خوشه‌ها به خوشه‌ی بی‌نهایت متصل می‌شوند.

الگوریتم محاسبه‌ای که استفاده کرده‌ام این است که در ابتدا بزرگ‌ترین خوشه‌ای که غیربی‌نهایت است را شناسایی و جذر مساحت آن را به عنوان طول همبستگی در نظر می‌گیرد. سپس برای طول شبکه‌های [10, 15, 23, 34, 51, 76, 114, 171] و احتمال‌های زیر، ۱۰۰ بار محاسبات لازم برای طول هم‌بستگی را انجام می‌دهد و میانگین را به عنوان طول همبستگی در آن طول شبکه و احتمال در نظر می‌گیرد.

P Values = [0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.57, 0.575, 0.58, 0.585, 0.59, 0.595, 0.6, 0.605, 0.610
0.615, 0.62, 0.63, 0.635, 0.7, 0.8, 0.9, 1]

دلیل اینکه طول‌های شبکه را به صورتی که در بالا گفته شد گرفته‌ام، این است که این طول‌ها تقریباً به صورت لگاریتمی و با ضریب ۱٫۵ رشد می‌کنند. این باعث می‌شود که در انتها که نمودار احتمال‌های قله‌ها برحسب طول شبکه را رسم می‌کنیم، نمودار تر و تمیزتری داشته باشیم.

در حد ترمودینامیکی که طول سیستم به بی‌نهایت میل می‌کند، در احتمال بحرانی طول همبستگی دچار واگرایی می‌شود. اما در شبیه‌سازی ما که طول سیستم محدود است به جای واگرایی شاهد قله‌هایی نسبتاً نوک‌تیز در احتمال بحرانی هستیم. نمودار زیر این موضوع را برای طول‌های بزرگ‌تری مانند ١١٤ و ١٧١ به خوبی نشان می‌دهد.



همچنین لیست احتمال‌های مربوط به این قله‌ها برای هر طول به ترتیب به صورت زیر به دست آمده است.

Pc = [0.5, 0.58, 0.57, 0.57, 0.575, 0.575, 0.58, 0.585]

L = [10, 15, 23, 34, 51, 76, 114, 171]

همان‌طور که دیده می‌شود بیش‌ترین احتمال ۰,۵۸۵ بدست آمده است. این را می‌توان به عنوان احتمال بحرانی تراوش جایگاهی دوبعدی در نظر گرفت. البته از مقدار تئوری آن که حدوداً ۰,۵۹۳ است به اندازه‌ی ۰,۰۸ انحراف دارد. همان‌طور که در نمودار بالا هم کاملاً مشهود است، مقدار ۰,۵۸ برای طول ۱۵ یک داده‌ی پرت محسوب می‌شود. چون قله‌ی این نمودار بسیار پهن است و قاعدتاً احتمال قله‌ای این نمودار نباید اینقدر به احتمال بحرانی نزدیک باشد، چون که طول شبکه در این حالت بسیار کوچک است! این هم از اثرات طول محدود است که اگر ملاحظات لازم را مد نظر نداشته باشیم، در محاسبات بعدی دچار خطا می‌شویم.

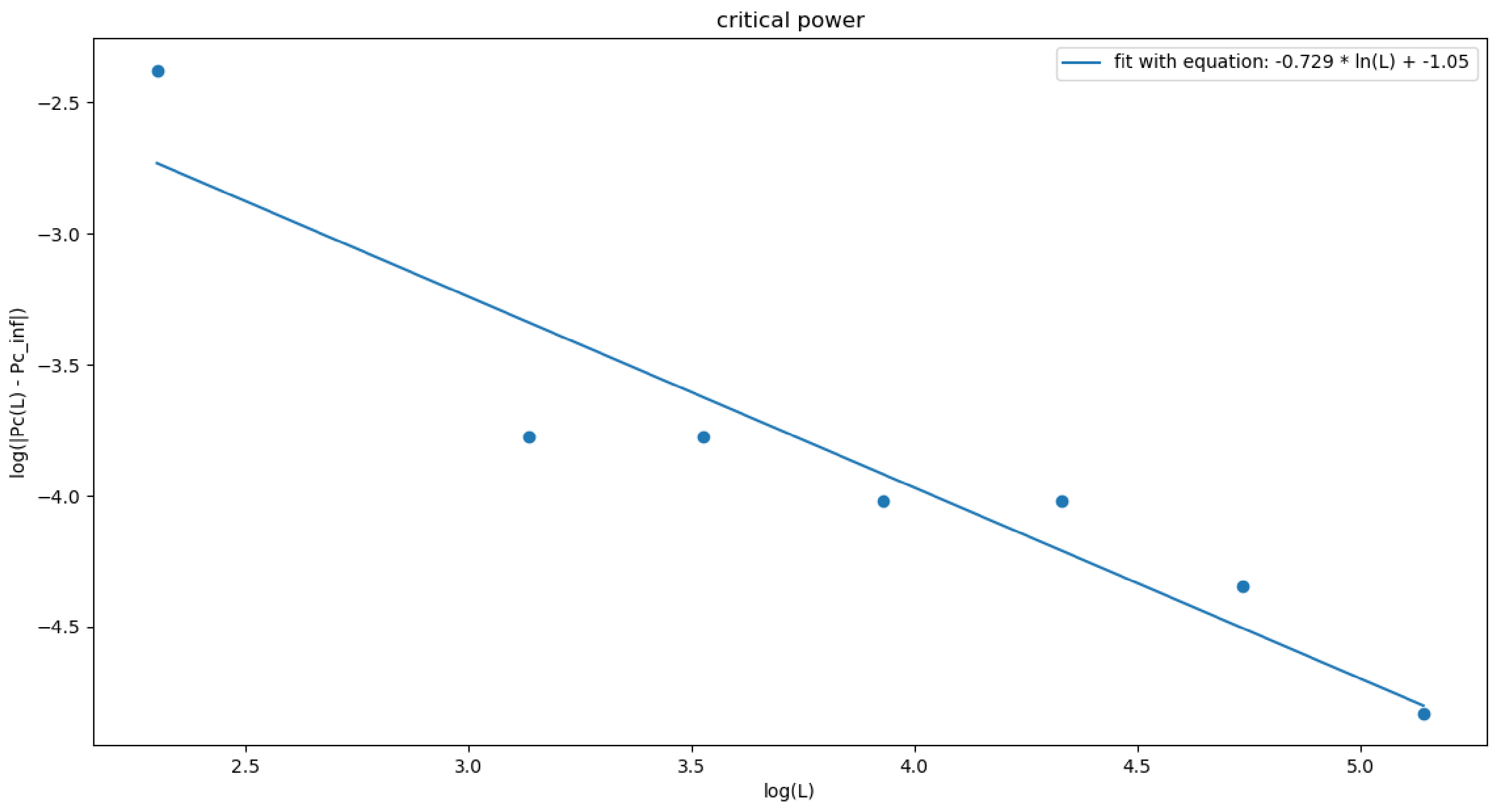
در ادامه کاری که می‌خواهیم انجام دهیم این است که نمودار لگاریتم قدرمطلق تفاضل این احتمال‌های بحرانی از ۰,۵۹۳ را برحسب لگاریتم طول شبکه رسم کنیم. دلیل این موضوع این است که می‌خواهیم نمای بحرانی ν را بدست آوریم. این قله‌ها در رابطه‌ی زیر صدق می‌کنند.

$$|P_c(L) - P_c(\infty)| \sim L^{\frac{-1}{\vartheta}}$$

با لگاریتم گرفتن از دو طرف تساوی به تساوی زیر می‌رسیم.

$$\log|P_c - 0.593| = -\frac{1}{\vartheta}\log L$$

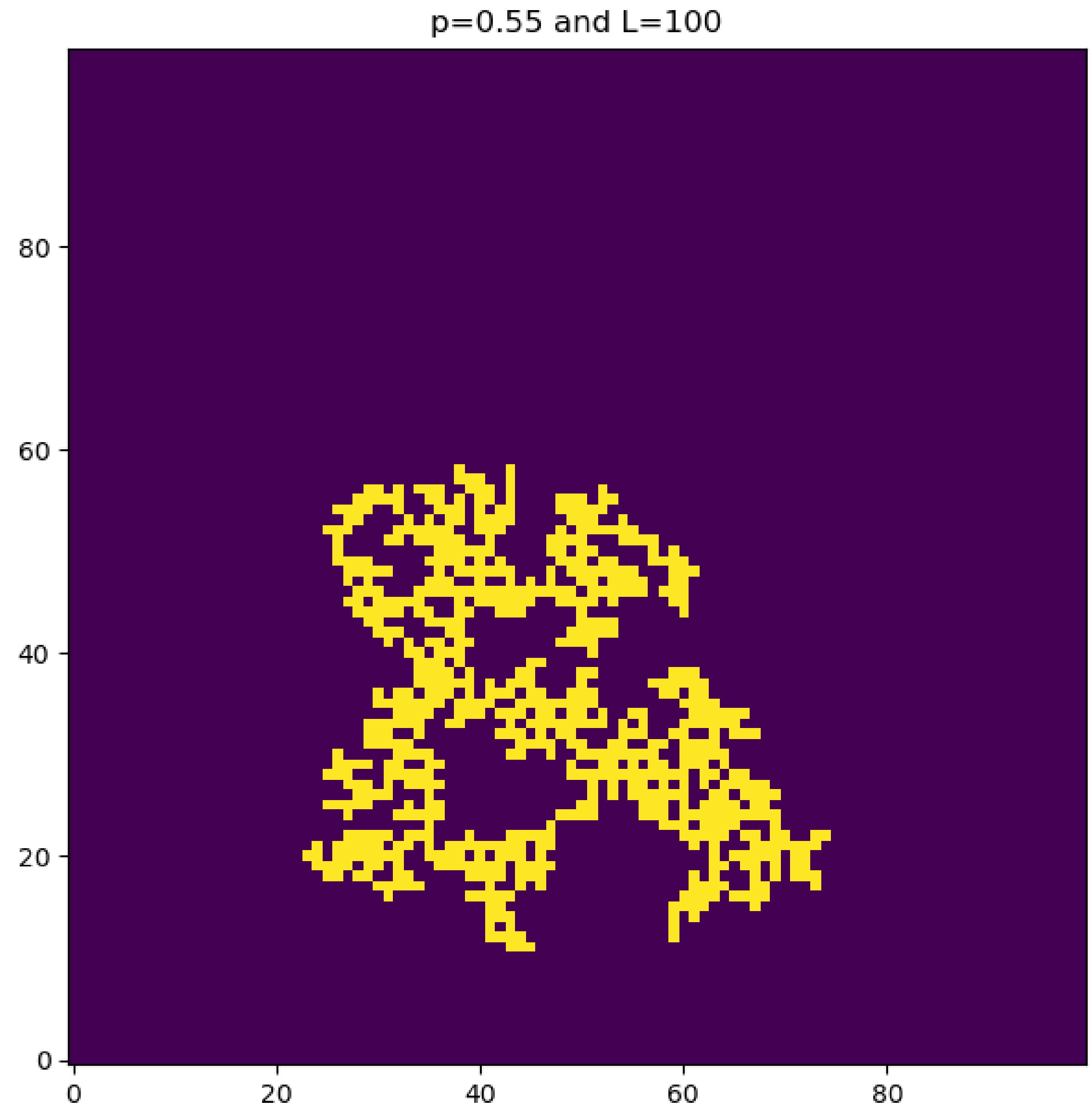
در ادامه نمودار معادله‌ی بالا را برای احتمال‌ها و طول‌هایی که در بالا آوردم، رسم می‌کنم. فقط به این نکته که احتمال ۰,۵۸ را برای طول ۱۵ از لیست‌ها حذف می‌کنم. چون این داده‌ی پرت است و محاسبه‌ی نمای بحرانی را دچار خطا می‌کند. این داده در نمودار زیر هم نشان داده نشده است!

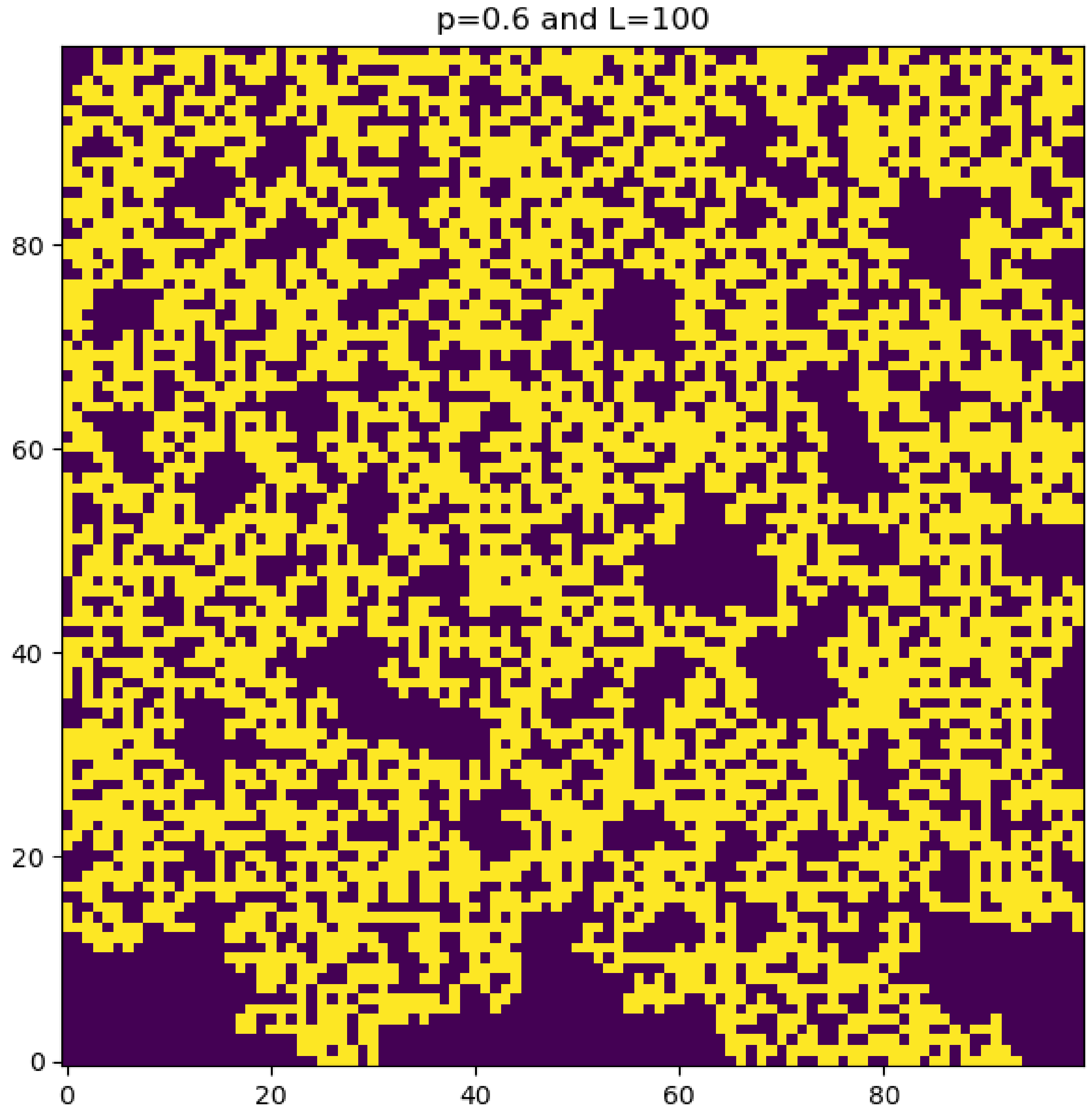


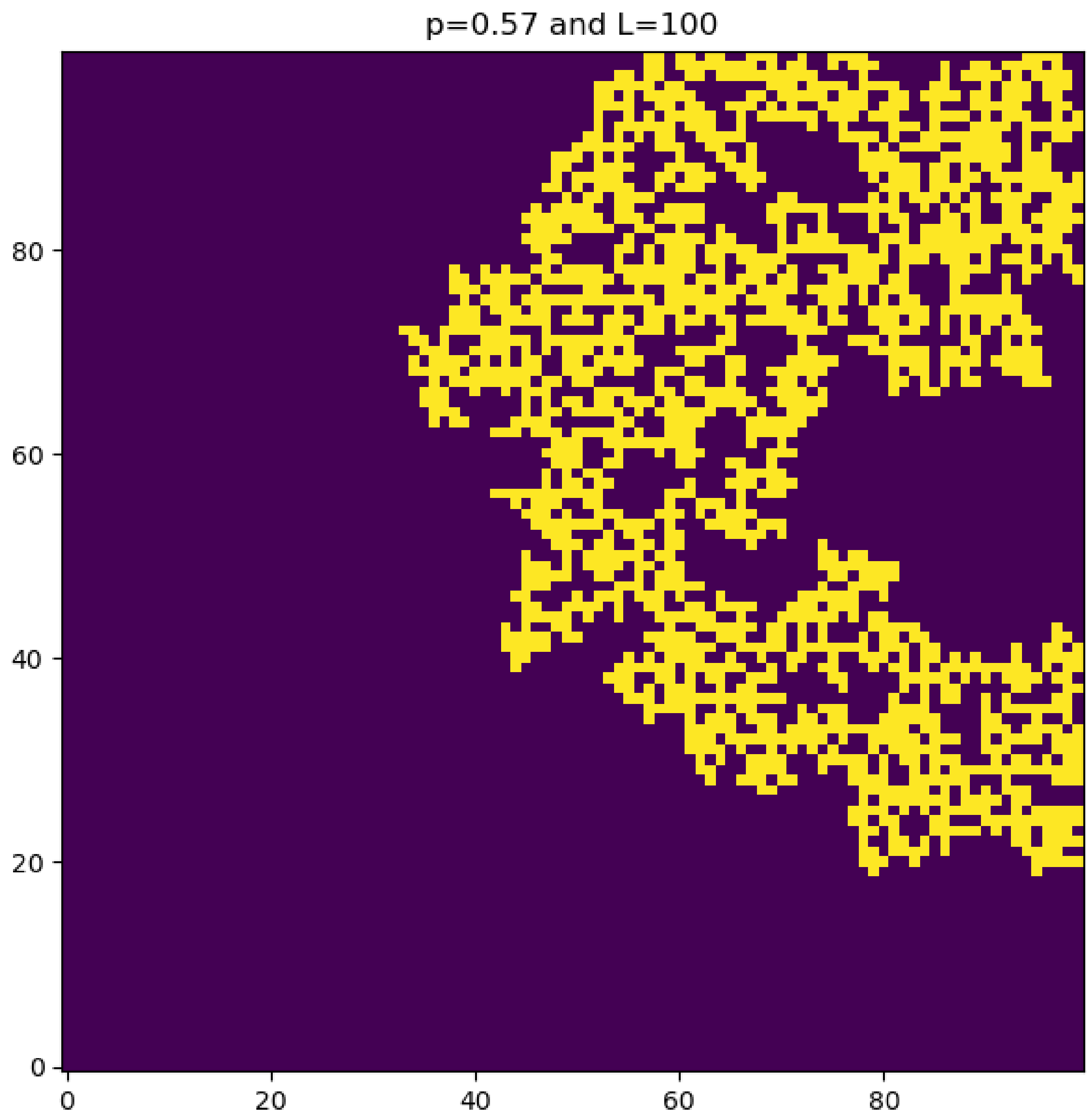
شیب نمودار بالا در واقع $-\frac{1}{\vartheta}$ است. بنابراین مقدار نمای بحرانی تا سه رقم اعشار به صورت ۱٫۳۷۲ محاسبه می‌شود. از مقدار تئوری آن که ۱٫۳۳۳ است حدود ۰٫۰۳۹ اختلاف دارد. نکته‌ای که وجود دارد این است که اگر طول‌های بزرگ‌تری برای شبکه در نظر بگیریم این اختلاف کم‌تر هم می‌شود. اما چون زمان اجرا برای طول‌های بزرگ‌تر خیلی طولانی می‌شد به همان بیشینه طول ۱۷۱ قناعت کردم!

۲. بعد جرمی خوشه‌های تراوش (۷-۴)

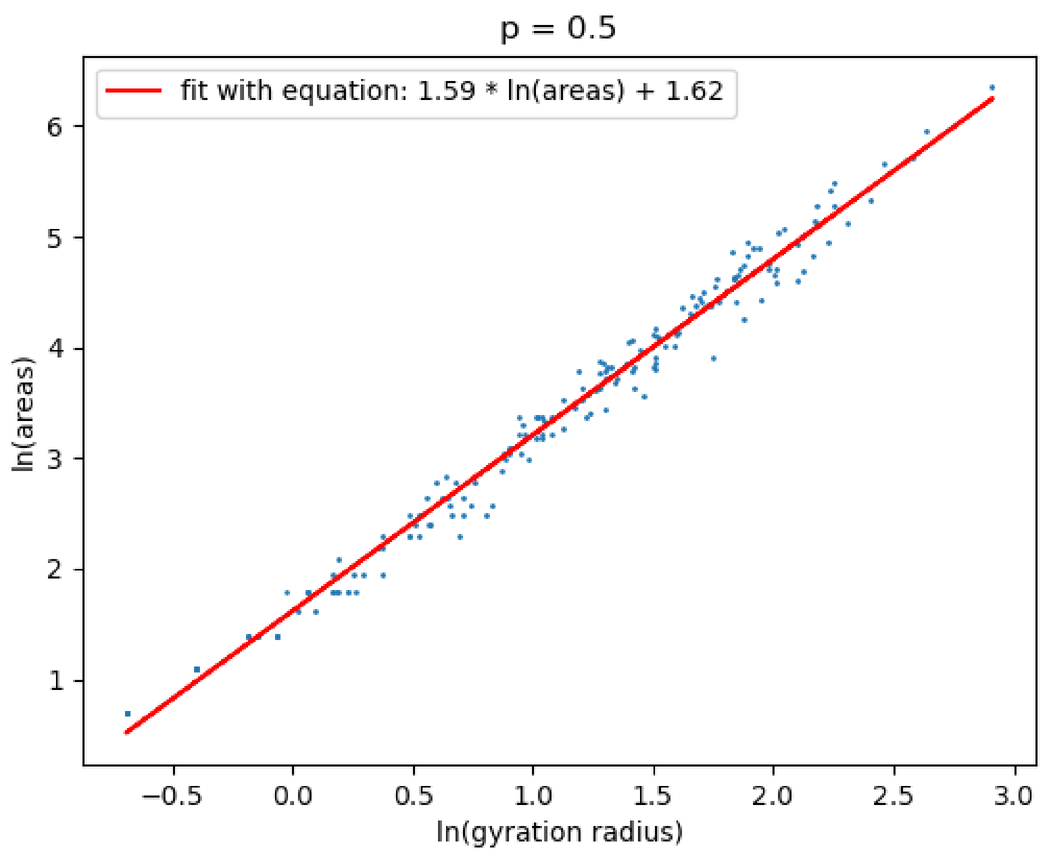
در این مسئله قصد داریم بعد جرمی فراکتال خوشه‌های تراوش را بدست آوریم. برای این منظور اولین کاری که انجام می‌دهیم این است که به کمک الگوریتم رشد خوشه، خوشه‌ای را تولید کنیم. برای این کار یک آرایه‌ی دوبعدی می‌سازیم. برای آنکه بعدا در مرزها دچار مشکل نشویم از هر طرف یک ستون یا ردیف اضافه در نظر می‌گیریم. در ادامه یک خانه از این آرایه را به عنوان مرکز رشد خوشه روشن می‌کنیم. سپس با کمک الگوریتم رشد خوشه، خوشه را توسعه می‌دهیم. اینگونه که هر خانه‌ای که روشن است، تمام همسایه‌های آن هم به صورت رندوم مقداردهی می‌شود. خانه‌هایی که روشن نمی‌شوند باید مسدود شوند. این فرایند تاجایی ادامه پیدا می‌کند تا مرزهای خوشه مسدود شود. همه‌ی این کارها به کمک تابع makeCluster() انجام می‌شود. در زیر نمونه‌هایی از خوشه‌های تولید شده را آورده‌ام. دقت شود که آن سطر و ستون‌های اضافه‌ای که در ابتدا اضافه کرده بودیم را حذف کرده‌ام و سپس نمایش داده‌ام.

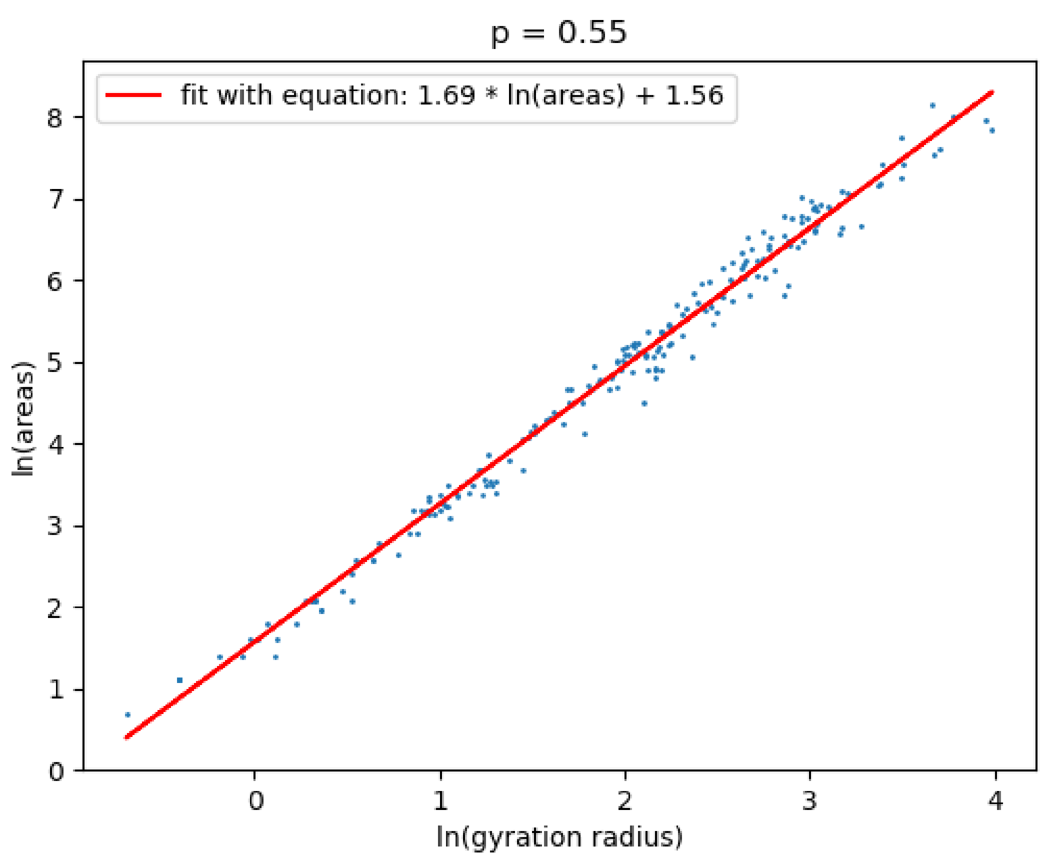


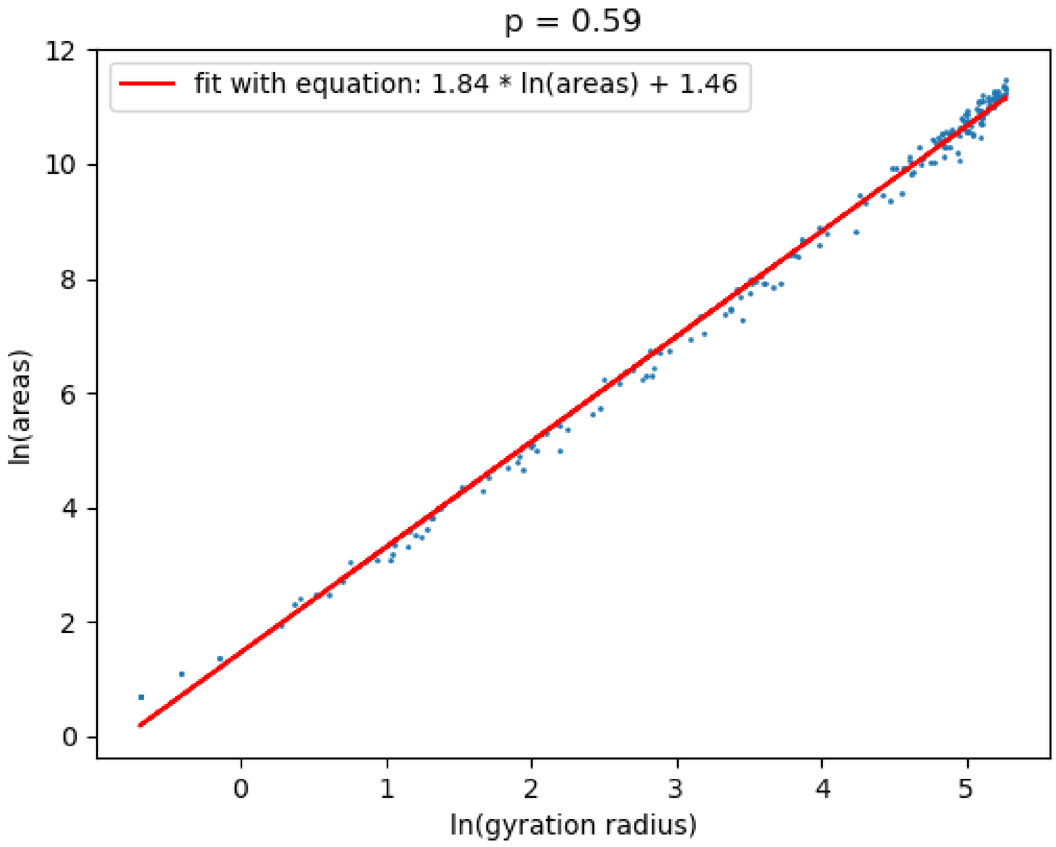

p=0.6 and L=100
0
20
40
60
80


p=0.57 and L=100
0
20
40
60
80

در ادامه کاری که انجام می‌دهیم این است که لگاریتم مساحت خوشه را برحسب شعاع ژیراسیون آن رسم کنیم. این کار را برای احتمال‌های ۰٫۵و ۰٫۵۵و ۰٫۵۹ و هر کدام با ۲۰۰ بار اجرا انجام می‌دهیم. توجه شود که مساحت خوشه تعداد خانه‌های روشن است و شعاع ژیراسیون از جذر میانگین مجذور فاصله‌ی خانه‌های روشن از مختصات میانگین بدست می آید. نمودارهای زیر برای هرکدام از احتمال‌های گفته شده بدست می‌آید. توجه شود که در درون این تصویرها، برای معادله‌ی بهترین خط، اشتباهاً ln(areas) نوشته شده است. منظورم ln(gyration radius) است! در کد این موضوع را تصحیح کرده‌ام!







در نمودارهای بالا، شیب خط بعد جرمی فراکتال خوشه‌ای را می‌دهد. خط رسم شده، بهترین خط است که به کمک scipy.status.linregress رسم شده است. همانطور که می‌بینیم و البته قابل انتظار بود این عدد بین ۱ و ۲ قرار دارد. در واقع وقتی به تعریفی که از بعد فراکتال‌ها داشتیم مراجعه می‌کنیم این موضوع کاملاً قابل انتظار است. با افزایش احتمال بعد خوشه‌ها هم بزرگ‌تر می‌شود. دلیلش آن است که خوشه پرتر و به هم چسبیده‌تر می‌شود و به یک توپولوژی دو بعدی نزدیک‌تر می‌شود. توجه شود که داده‌های مربوط به نمودارها به صورت فایل .npy ذخیره شده است. البته چون حجم این فایل‌ها زیاد شد (برای همه‌ی مسئله‌های این سری روی هم رفته ۱۵۲ مگابایت شد!) نتوانستم آن‌ها را بفرستم. در صورت لزوم امکان ارسال و یا به اشتراک گذاری آن‌ها را از طریق‌های دیگر دارم!

۳.اثبات رابطه‌ی ۵ فصل ول‌گشت کتاب (۱–۵)

برای اثبات رابطه‌ی گفته شده، ابتدا می‌دانیم که مکان ولگرد در زمان t به صورت زیر به مکان آن در زمان t-τ بستگی دارد.

$$x(t) = x(t-\tau) + al$$

که a یک متغیر تصادفی است که با احتمال p، مقدار ۱+ و با احتمال q، مقدار ۱– به خود می‌گیرد. در نتیجه می‌توانیم رابطه‌ی زیر را بنویسیم.

$$< x(t) > = < x(t-\tau) > + (p-q)l$$

اکنون سعی می‌کنیم رابطه‌ای مشابه برای مجذور مکان به‌دست آوریم.

$$< x(t)^2 > = < x(t-\tau)^2 > + < a^2 > l^2 + 2 < a >< x(t-\tau) > l$$

در نتیجه رابطه‌ی زیر را برای واریانس می‌توانیم بدست آوریم.

$$\sigma^2(t) = < x(t)^2 > - < x(t) >^2 = < x(t-\tau)^2 > + < a^2 > l^2 + 2 < a >< x(t-\tau) > l$$
$$- < x(t-\tau) >^2 - (p-q)^2 l^2 - 2\,(p-q) l < x(t-\tau) >$$

حال با توجه به اینکه

$$< a^2 > = p + q$$

به رابطه‌ی زیر می‌رسیم.

$$\sigma(t)^2 = \sigma(t-\tau)^2 + ((p+q) - (p-q)^2) l^2$$

حال کافی‌ست که توجه کنیم که عبارت داخل پرانتز را می‌توان آن‌گونه که در زیر آمده است ساده‌تر کرد.

$$p + q - (p-q)^2 = p - p^2 + q - q^2 + 2pq = p(1-p) + q(1-q) + 2pq = 4pq$$

که در تساوی آخر از این نکته‌ی بدیهی که 1-q=p است و 1-p=q است استفاده کردم. به این ترتیب به رابطه‌ی زیر می‌رسیم.

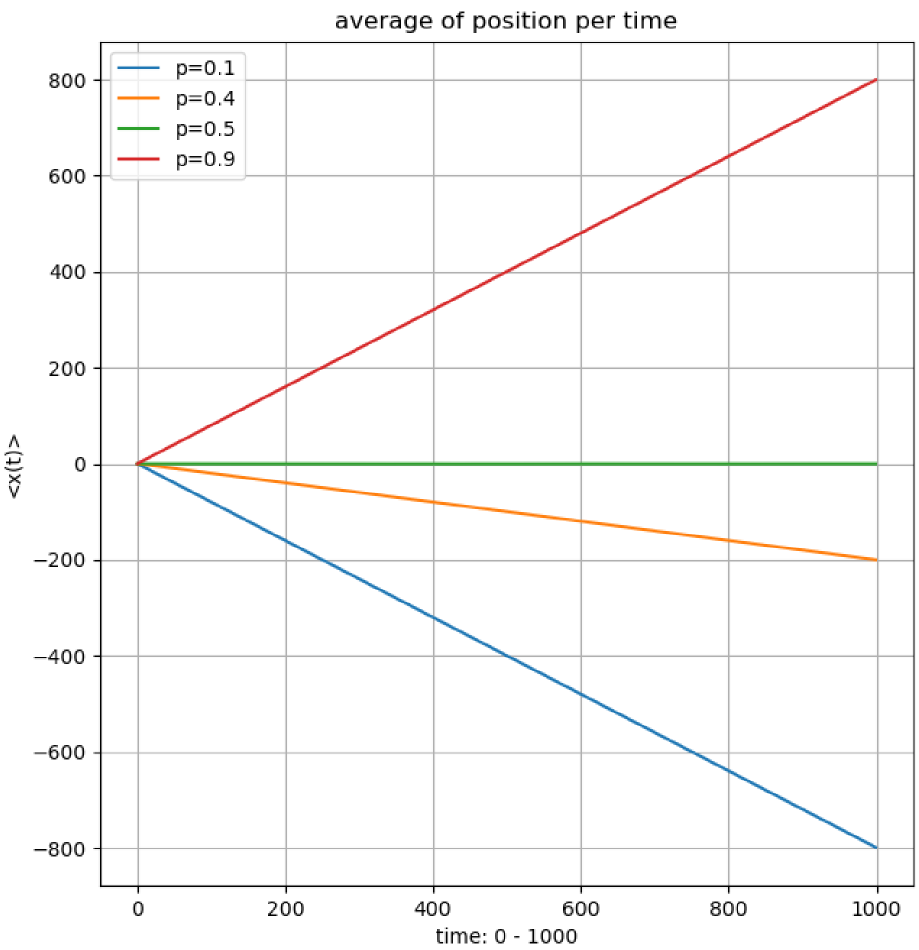
$$\sigma(t)^2 - \sigma(t-\tau)^2 = 4pql^2$$

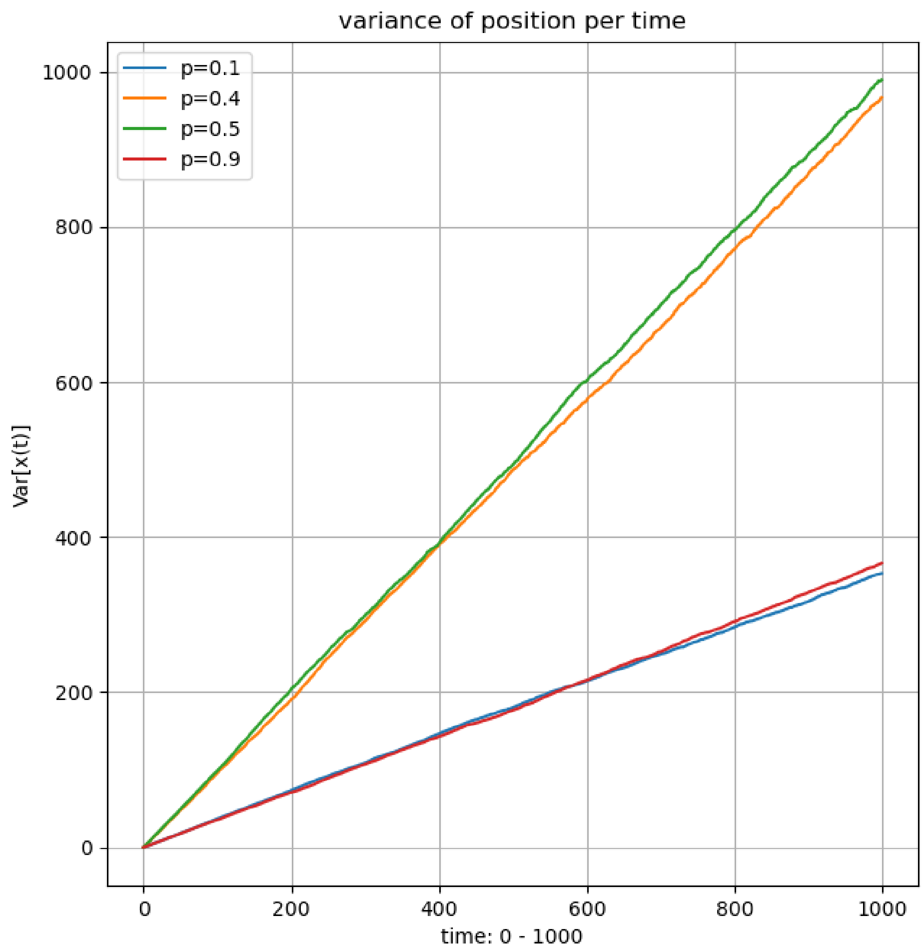
پس روشن است که در زمان t که ولگرد به تعداد $\frac{t}{\tau}$ قدم برداشت، پاسخ عبارت بازگشتی بالا به صورت زیر خواهد بود و نتیجه‌ی داده شده در جزوه اثبات می‌شود.

$$\sigma(t)^2 = 4pq\frac{t}{\tau} l^2$$

۴. برنامه‌ای برای ولگشت یک بعدی و بررسی صحت روابط کتاب (۲–۵)

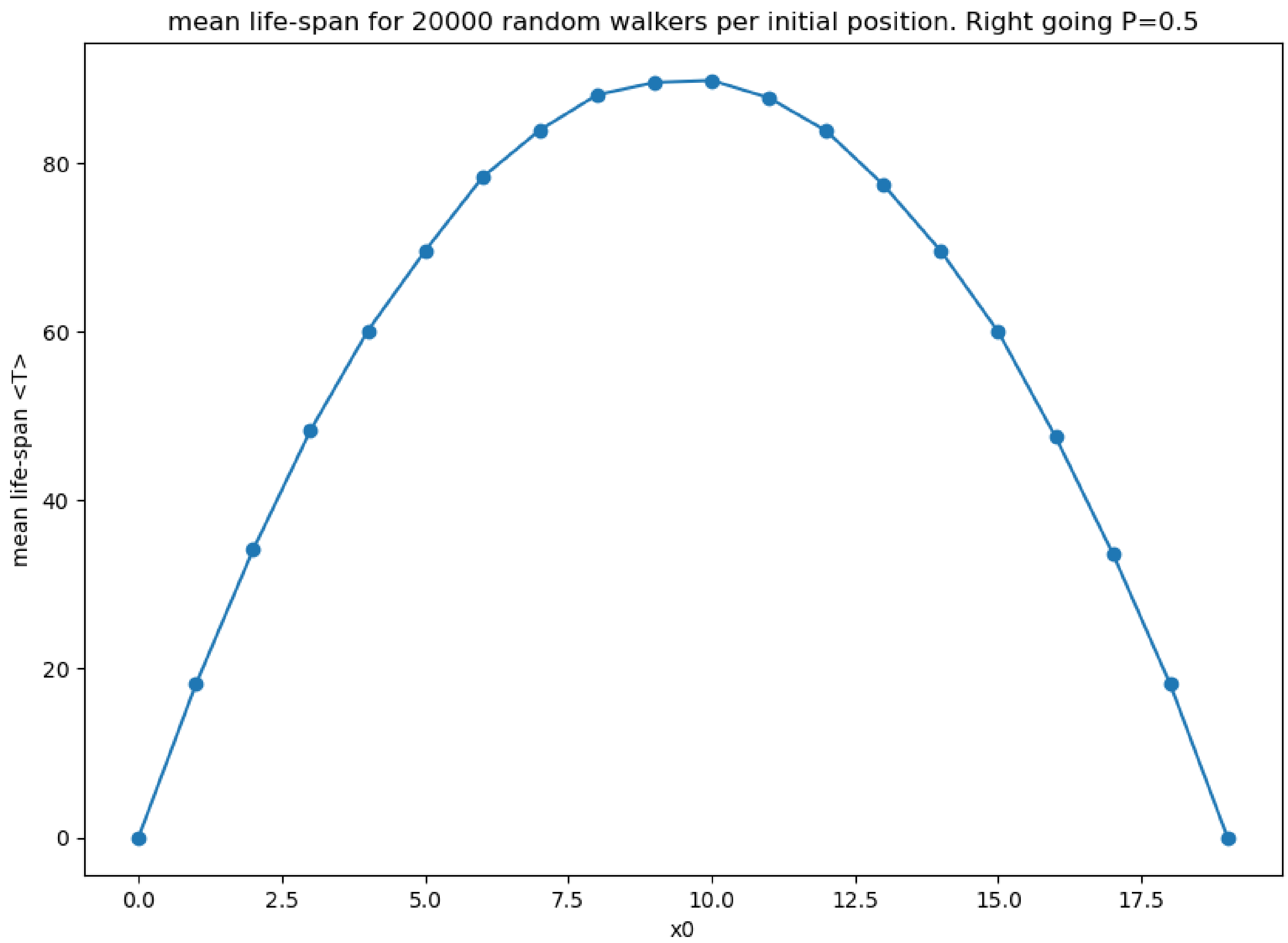
در این برنامه قصد داریم تا ولگرد یک بعدی را شبیه‌سازی کنیم. برای این کار، کافی‌ست که تعدادی گام زمانی (در داخل برنامه ۱۰۰۰ گام یک واحدی به صورت پیش‌فرض در نظر گرفته شده است) در نظر بگیریم. در ادامه برای بدست آوردن کمیت‌های <x(t)> و<Var[x(t)]> نیاز است تا تعدادی ولگرد در نظر بگیریم و بعد بر روی این ولگردها این کمیت‌ها را حساب کنیم. در داخل برنامه به صورت پیش‌فرض تعداد ولگردها را ۱۰,۰۰۰ گرفته‌ام. همچنین این کار را به صورت پیش‌فرض برای ٤ احتمال ۰,۱، ۰,٤، ۰,۵ و ۰,۹ انجام داده‌ام و برای هر کدام یک فایل با فرمت npy. بدست آورده‌ام. سطرهای این فایل همان گام‌های زمانی است و ستون‌های آن متناظر با تعداد ولگردهاست. در داخل هر خانه هم مکان ولگرد ثبت شده است. نتیجه به صورت نموداری در زیر آمده است. برای نمونه برای حالتی که p=0.5 باشد، انتظار داریم که میانگین مکان ولگرد در هر زمان، صفر باشد. همچنین انتظار داریم که شیب خط نمودار واریانس برحسب زمان برابر با یک باشد. این دو در نمودارهای زیر کاملاً مشهود است.





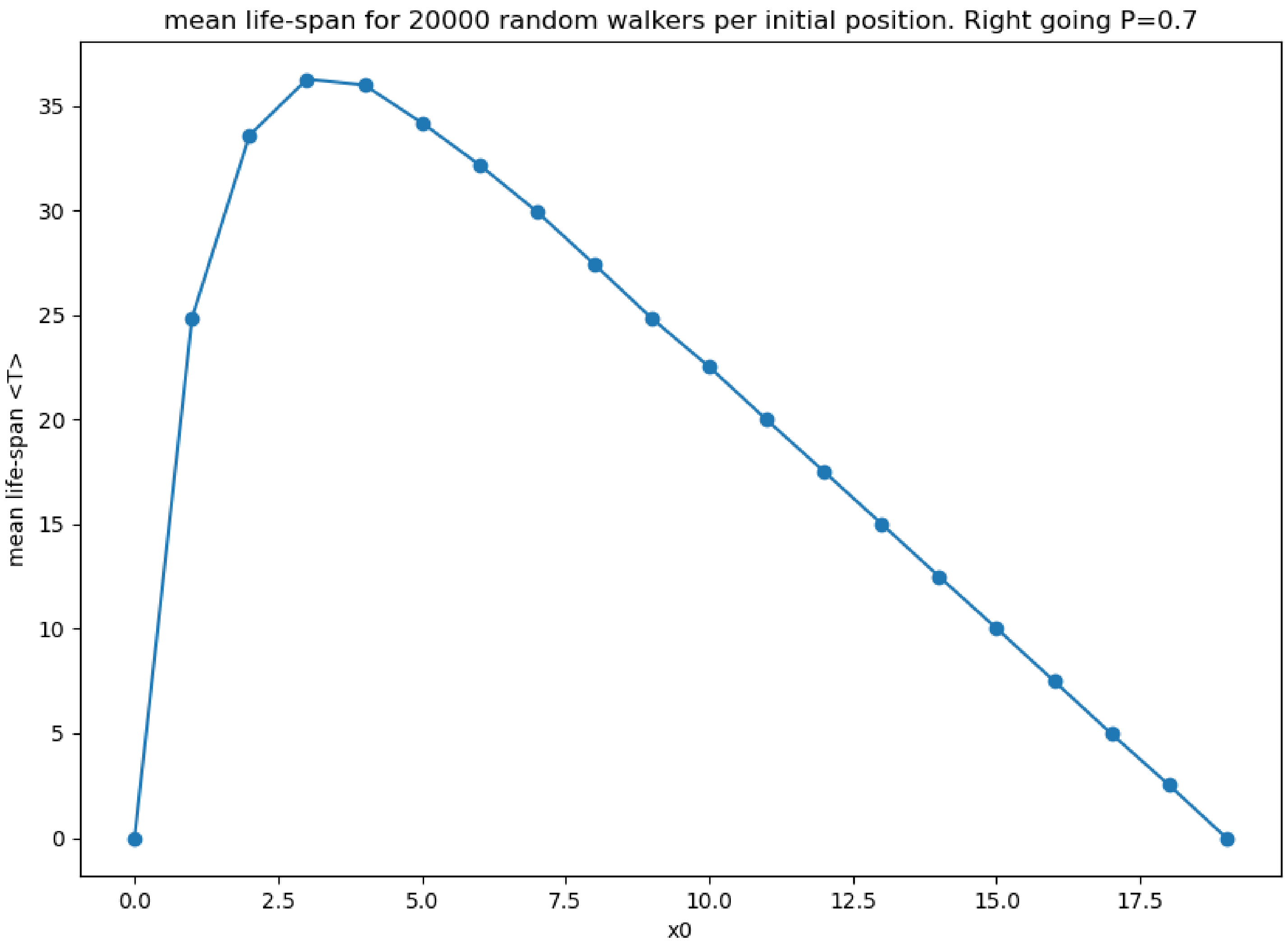
۵. شرط مرزی جاذب (۳–۵)

در اینجا همان مسئله‌ی ولگرد را با یک شرط مرزی متناوب پیاده سازی می‌کنیم. کلیت پیاده سازی در درون تابع randomWalkerWithTraps() انجام شده است. تابع makeLifeTimePerInitialPosition() را تعریف می‌کنم. کاری که این تابع انجام می‌دهد این است که یک آرایه به طول بیست (مقدار پیش‌فرض برای تعداد سایت‌ها) را می‌سازد و سپس در هر خانه‌ی این آرایه، مدت زمانی که طول می‌کشد تا ولگرد به یکی از مرزها برسد را قرار می‌دهد. البته بدیهی است که برای نقاط مرزی این زمان صفر است و خروجی‌های برنامه با همین فرض است. البته این فرض عملاً هیچ تاثیری در کیفیت نمودار و تابعیت زمان میانگین از مکان اولیه ندارد، ولی در عمر میانگین نهایی ولگرد اثر دارد. سرانجام پس از آنکه این تابع محاسبات لازم را انجام داد و آرایه‌ی یاد شده را ساخت آن را برمی‌گرداند. در ادامه در تابع analyser() با فراخوانی تابع makeLifeTimePerInitialPosition() درون یک حلقه‌ی for بر روی تعداد زیادی ولگرد میانگین‌گیری را انجام می‌دهم. به صورت پیش‌فرض تعداد ولگردها را ۲۰,۰۰۰ گرفته‌ام. این تعداد کفایت می‌کند تا یک نمودار تمیز و نرم به عنوان خروجی داشته باشیم. همچنین تابع analyser() مقدار عمر میانگین ولگرد را با استفاده از دستور print() برای کاربر نمایش می‌دهد. سرانجام خروجی به صورت زیر خواهد بود.



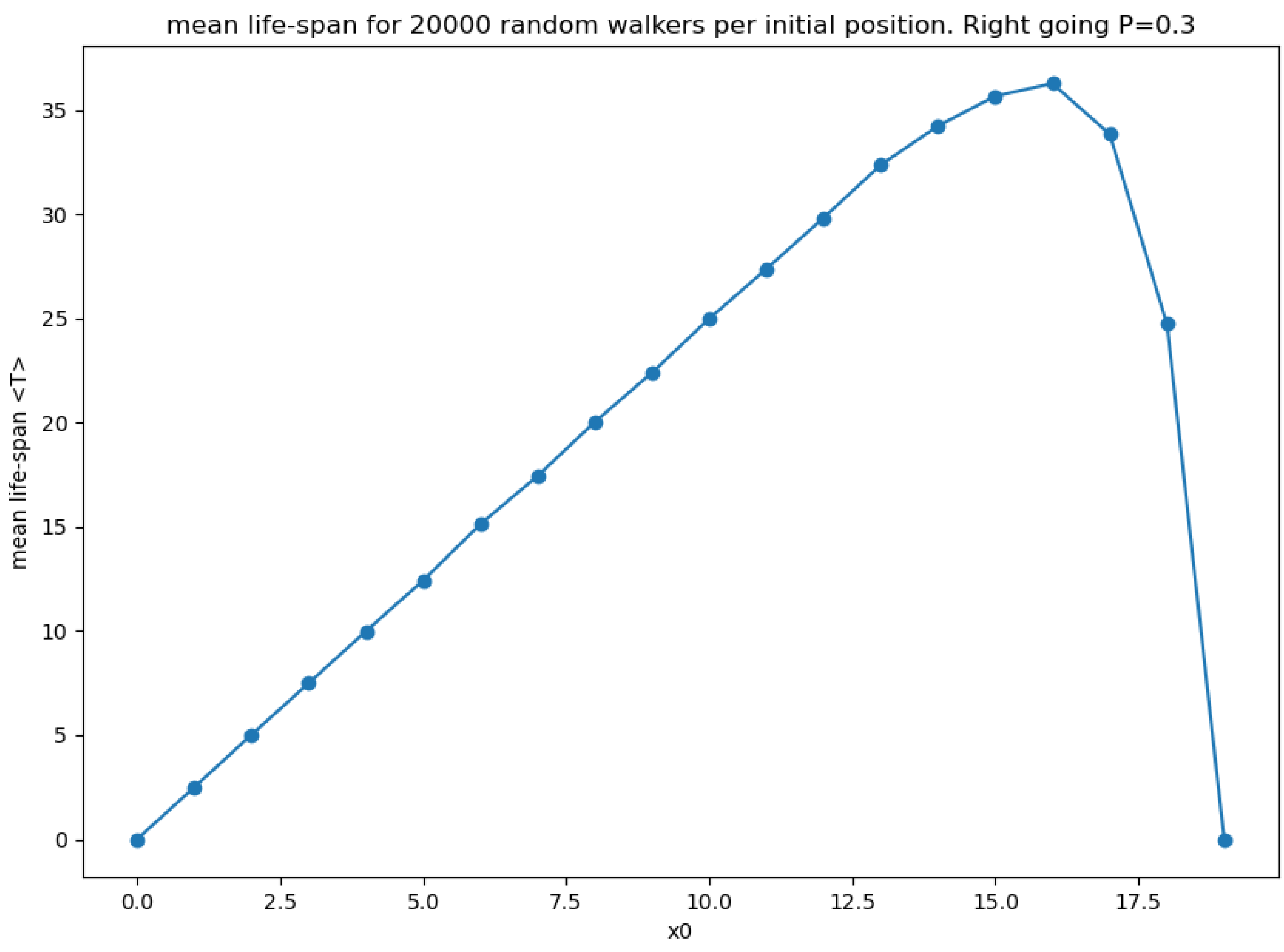
برای نتایج بالا، عمر میانگین ولگرد ۵۶,۹۳ بدست آمده است. همان‌طور که می‌بینید شکل نمودار نسبت به مکان میانی (۱۰,۵ در واقع کاملاً متقارن است. دلیل این موضوع این است که احتمال قدم گذاشتن ولگرد به راست و چپ با هم برابر و برابر با ۰,۵ است.

به عنوان نمونه‌ای دیگر اگر احتمال قدم گذاشتن ولگرد به سمت راست را ۰,۷ بگیریم، نمودار زیر را خواهیم داشت.



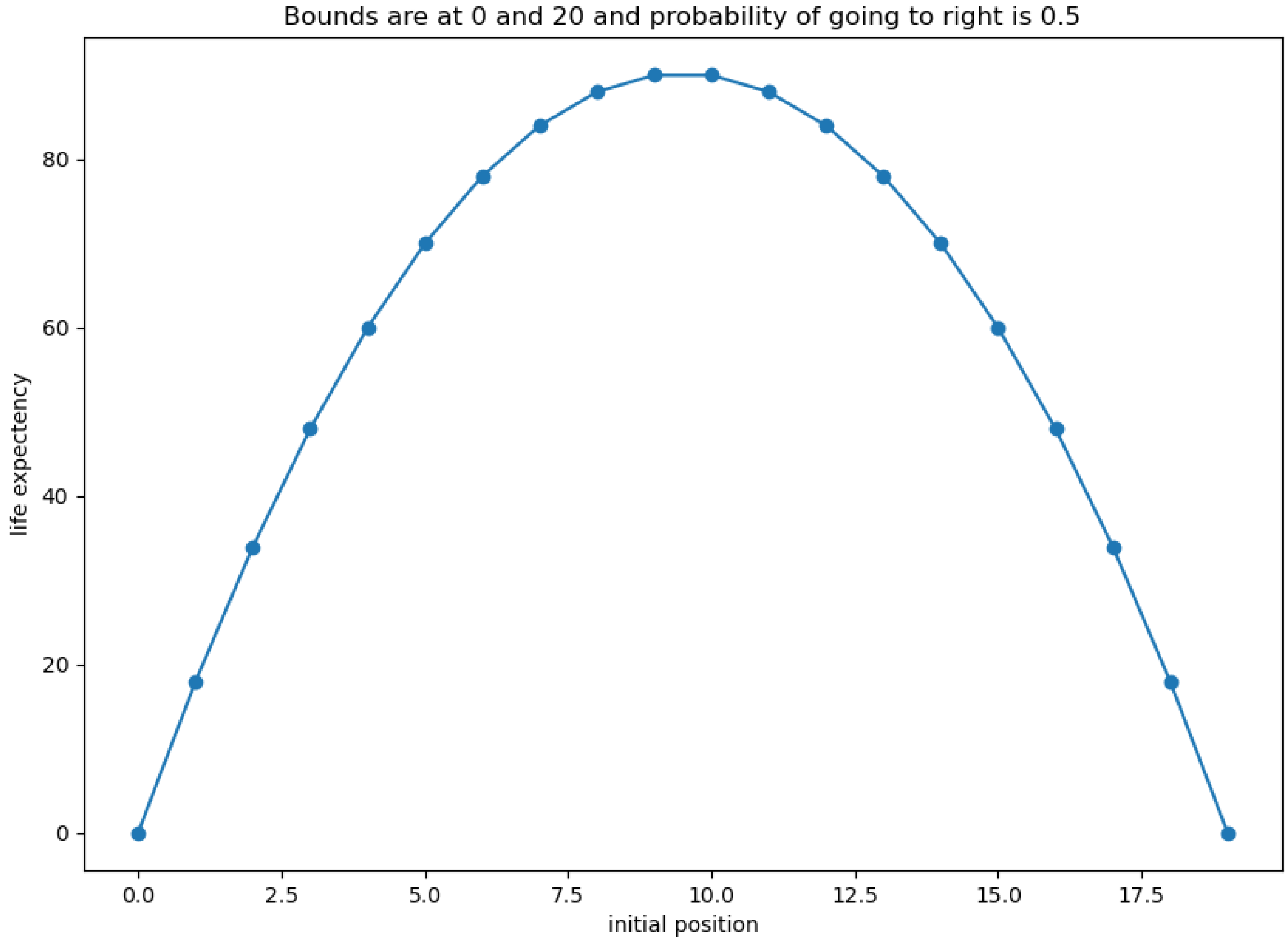
و میانگین عمر ولگرد برابر با ۱۹,٦۰ بدست آمد. همان‌طور که می‌بینیم برای مکان‌های اولیه‌ی سمت راست‌تر عمر ولگرد کم‌تر است. این موضوع کاملاً قابل انتظار است. چون ولگرد به سمت راست رفتن را با احتمال بیش‌تری انتخاب می‌کند و در نتیجه خیلی زودتر به مرز سمت راست می‌رسد.

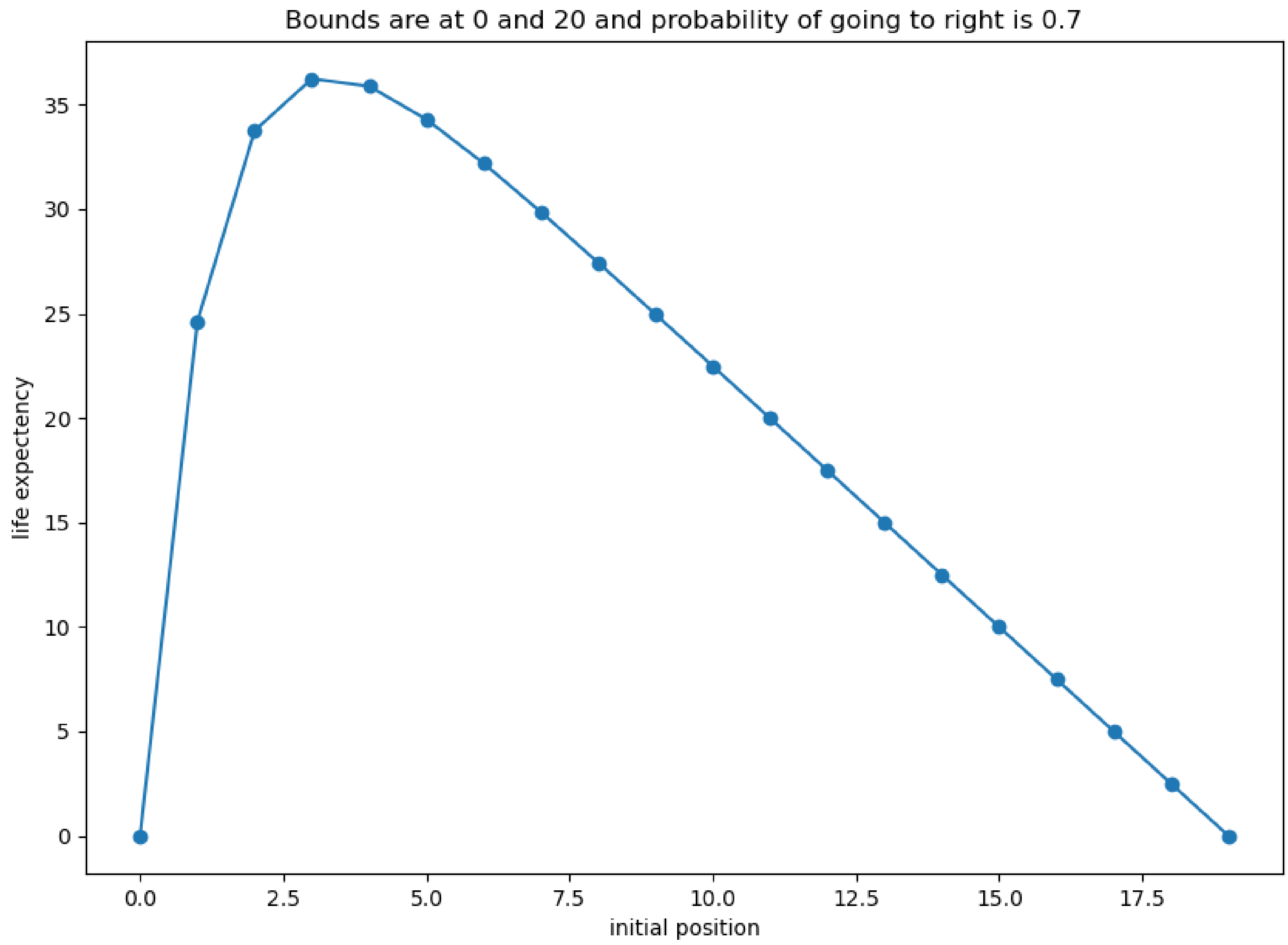
به عنوان یک نمونه‌ی دیگر احتمال گام برداشتن به سمت راست را این‌بار ۰,۳ گرفته‌ام. نمودار زیر بدست می‌آید. در این حالت هم عمر ولگرد برابر با ۱۹,٦۰ بدست آمد.

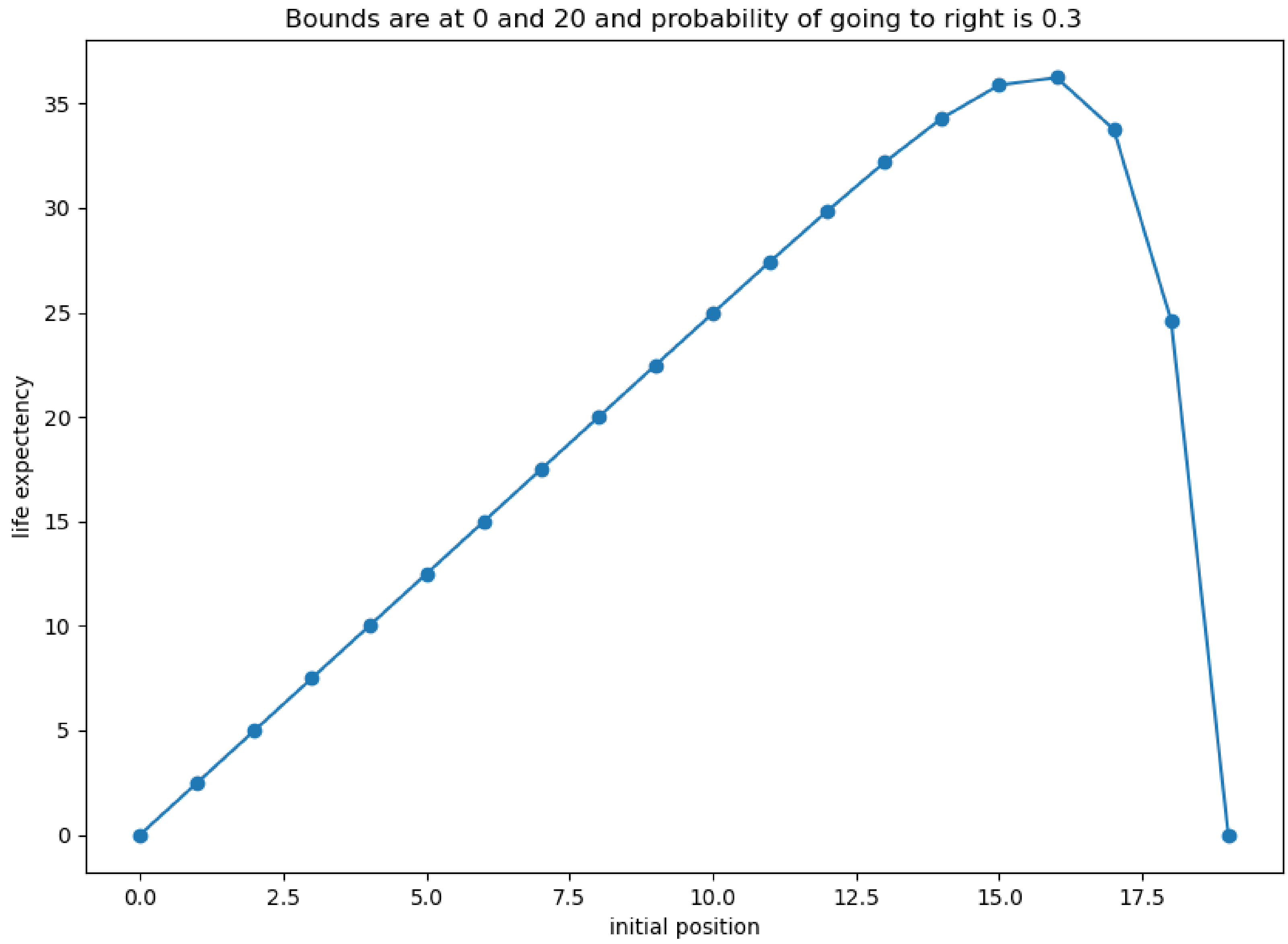


۶. الگوریتم سرشماری (۴-۵)

در این مسئله قصد داریم با استفاده از یک الگوریتم تعینی، عمر متوسط ولگرد را به صورت تابعی از مکان اولیه‌ی آن بدست آوریم. الگوریتمی که از آن استفاده می‌کنیم، الگوریتم سرشماری است. این الگوریتم این گونه است که احتمال حضور ولگرد در هر خانه را در هر زمان (یا گام زمانی) محاسبه می‌کند. در مورد شرایط مرزی (که شرایط مرزی را خانه‌های ۰ و آخر گرفته‌ام) این گونه عمل می‌کند که با حضور ولگرد در این خانه‌ها، عمر او به پایان می‌رسد. بنابراین احتمال‌های مربوط به این خانه‌ها نباید در سایرخانه‌ها انتشار پیدا کند. در هر مرحله، مجموع احتمال‌های خانه‌های اول و آخر به عنوان احتمال مرگ در آن مرحله در نظر گرفته می‌شود. توجه شود که عمر متوسط ولگرد را می‌توان در هر مرحله از حاصلضرب احتمال مرگ در آن مرحله در زمان آن مرحله به دست آورد. چیزی که در نهایت به عنوان عمر متوسط ولگرد در نظر گرفته می‌شود حاصل جمع همه‌ی این مقادیر است. در ضمن نکته‌ای که وجود دارد این است که به لحاظ تئوری، ولگرد بعد از گذشت زمان بی‌نهایت حتماً می‌میردا (ولگردی عاقبت خوبی ندارد!!) اما از آن‌جا که در شبیه‌سازی نمی‌توانیم زمان بی‌نهایت را مدل کنیم، ناچاریم با استفاده از محدودیت‌های معقول این کار را انجام دهیم. کاری که من انجام داده‌ام این است که دو محدودیت در نظر گرفته‌ام. یکی محدودیت روی جمع احتمال‌های خانه‌های مرگ است. کدی که نوشته‌ام تا وقتی محاسبات را پیش‌می‌برد که احتمال مرگ کم‌تر از یک حد معقول (مثلاً ۰٫۹۹۹۹) باشد. محدودیت دوم بر روی گام‌های زمانی است. چون به هرحال برای مقداردهی‌های اولیه لازم است از بعد آرایه‌ای که احتمال‌ها را در آن ذخیره می‌کنیم با خبر باشیم (از آرایه‌های نامپای استفاده کرده‌ام). در مجموع محاسبات تا زمانی که این دو شرط برقرار باشند انجام می‌شود. پیاده‌سازی برنامه در درون یک تابع مادر به نام randomWalkerByNoseCounting() انجام شده است. با استفاده از آرگومان‌های این تابع می‌توان همه‌ی پارامترهای مسئله را مقداردهی اولیه کرد (البته برای این آرگومان‌ها مقدار پیش‌فرض در نظر گرفته شده است!). با استفاده از تابع makeProbabilities() همه‌ی توزیع احتمال برای مکان اولیه‌ی مورد نظر ساخته می‌شود. سپس این آرایه‌ی دوبعدی را برمی‌گرداند. تابع calculateExpectancy() آرایه‌ی دوبعدی احتمالات را دریافت می‌کند و با استفاده از آن، عمر متوسط ولگرد را محاسبه می‌کند و مقدار آن را برمی‌گرداند. سرانجام تابع plotLifeExpectancyPerInitialPosition() برای همه‌ی مکان‌های اولیه تابع calculateExpectancy() را فراخوانی می‌کند و همه‌ی عمرمتوسط‌ها را در یک آرایه ذخیره می‌کند. سرانجام نمودار عمر متوسط ولگرد برحسب مکان اولیه را رسم می‌کند. نمودارهای زیر برای بعضی احتمال‌ها بدست آمده است.


Bounds are at 0 and 20 and probability of going to right is 0.5
life expectency
initial position
0
20
40
60
80
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5





همان‌طور که دیده می‌شود نتایج کاملاً با نتایج مسئله‌ی قبل سازگار است. البته این الگوریتم قاعدتاً خیلی سریع‌تر نتایج را به ما می‌دهد.