شبیه‌سازی رایانه‌ای در فیزیک
تمرین دوم: فراکتال، لایه‌نشانی و رشد فراکتالی

سینا معمر ۹۵۱۰۲۳۱۶

۱۸ شهریور ۱۴۰۰

۱ مجموعه کوخ

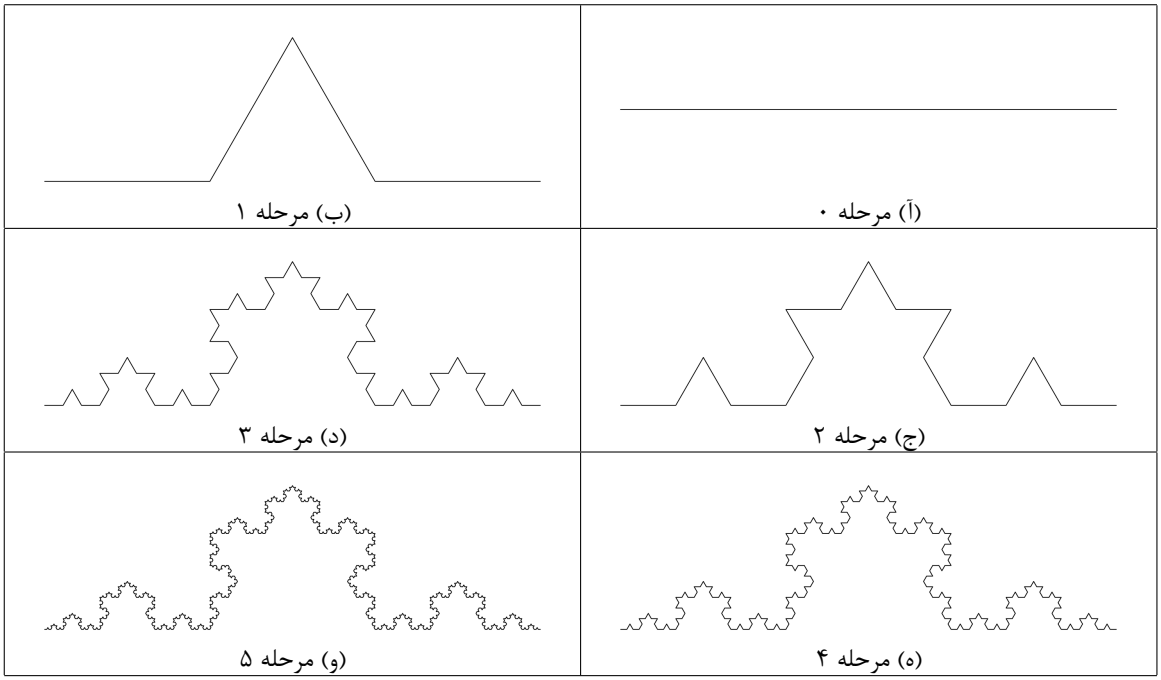
کد این بخش از تمرین در فایل q1.py قابل مشاهده است. روش کار به این صورت است که ابتدا یک object از کلاس KochSnowflake با طول دلخواه می‌سازیم. سپس تابع render را با زمان مورد نظر فراخوانی می‌کنیم. این تابع در یک حلقه به اندازه‌ی زمان داده شده، نقاط راس مرحله‌ی قبلی را به تابع transform__ پاس می‌دهد و نقاط جدید را در خانه‌ی بعدی متغیر stages_points ذخیره می‌کند. تابع transform__ به این صورت عمل می‌کند که ابتدا مختصات تمام نقاط را $\frac{1}{3}$ کرده و بعد آن‌ها را در یک لیست جدید ذخیره می‌کند. سپس همان نقاط را به اندازه ۶۰ درجه به صورت پادساعت‌گرد می‌چرخاند و به اندازه $\frac{1}{3}$ طول اولیه به سمت راست منتقل می‌کند. این نقاط جدید را هم در ادامه‌ی نقاط قبلی در لیست ذخیره می‌کند. سپس نقاط اسکیل شده‌ی اولیه را این‌بار ۶۰ درجه به صورت ساعت‌گرد می‌چرخاند و به اندازه $\frac{2}{3}$ طول اولیه به راست منتقل کرده و سپس در لیست ذخیره می‌کند. در نهایت نیز نقاط اولیه را به اندازه‌ی $\frac{2}{3}$ طول به راست منتقل و آن‌ها را نیز به لیست نهایی اضافه می‌کند. لیست به دست آمده مختصات رئوس مرحله‌ی جدید مجموعه‌ی کوخ است و در جواب تابع برگردانده می‌شود. لازم به ذکر است که تمامی مختصات به صورت اعداد مختلط ذخیره شده‌اند و عمل انتقال از طریق جمع و عمل دوران نیز از طریق ضرب اعداد مختلط انجام شده است. نتیجه به دست آمده تا مرحله‌ی ۵ ام را در شکل ۱ می‌توان مشاهده نمود.

۲ مثلث سرپینسکی

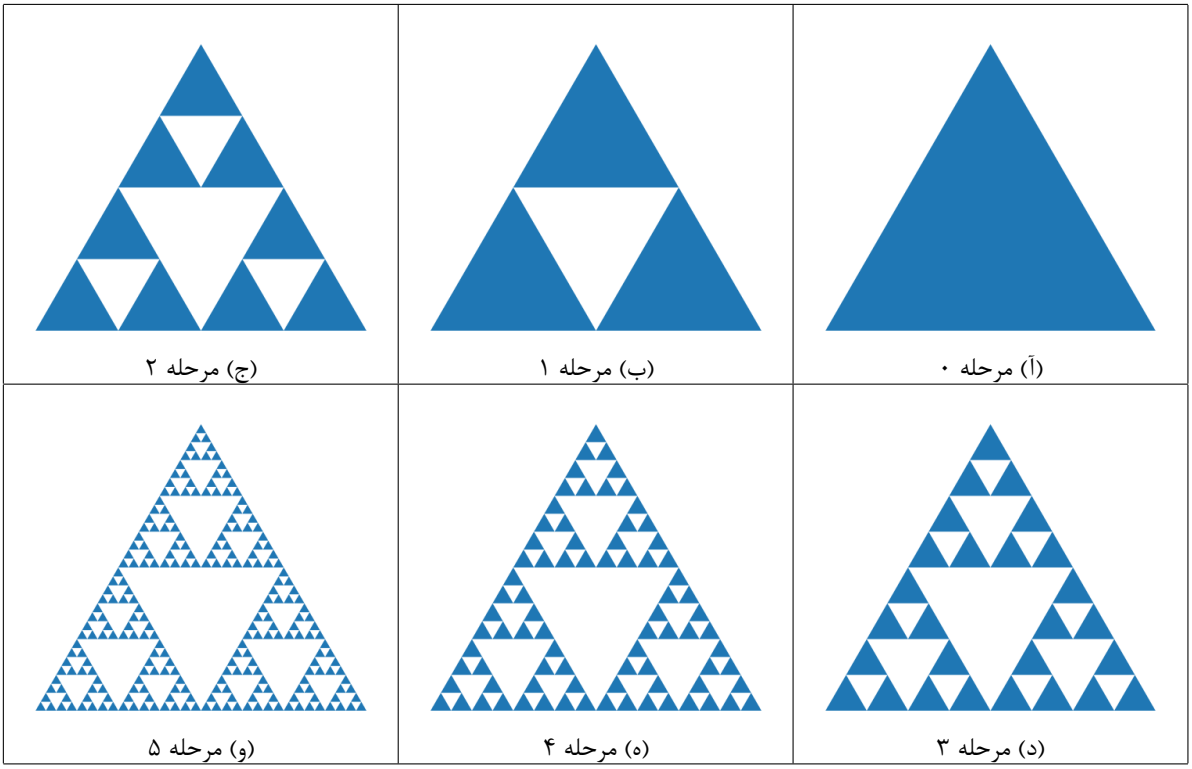
کد این بخش از تمرین در فایل q2.py قابل مشاهده است. روش کار به این صورت است که ابتدا یک object از کلاس SierpinskiTriangle با طول دلخواه می‌سازیم. سپس تابع render را با زمان مورد نظر فراخوانی می‌کنیم. این تابع در یک حلقه به اندازه‌ی زمان داده شده، نقاط راس مرحله‌ی قبلی را به تابع transform__ پاس می‌دهد و نقاط جدید را در خانه‌ی بعدی متغیر stages_points ذخیره می‌کند. تابع transform__ به این صورت عمل می‌کند که ابتدا مختصات تمام نقاط را $\frac{1}{2}$ کرده و بعد آن‌ها را در یک لیست جدید ذخیره می‌کند. سپس همان نقاط را به اندازه $\frac{1}{2}$ طول اولیه به سمت راست منتقل می‌کند. این نقاط جدید را هم در ادامه‌ی نقاط قبلی در لیست ذخیره می‌کند. سپس نقاط اسکیل شده‌ی اولیه را این‌بار به اندازه $\frac{1}{4}$ طول اولیه به راست و به اندازه‌ی $\frac{\sqrt{3}}{4}$ طول به بالا منتقل کرده و آن‌ها را نیز به لیست نهایی اضافه می‌کند. لیست به دست آمده مختصات رئوس مرحله‌ی جدید مثلث سرپینسکی است و در جواب تابع برگردانده می‌شود. لازم به ذکر است که تمامی مختصات به صورت اعداد مختلط ذخیره شده‌اند و عمل انتقال از طریق جمع و عمل دوران نیز از طریق ضرب اعداد مختلط انجام شده است. نتیجه به دست آمده تا مرحله‌ی ۵ ام را در شکل ۲ می‌توان مشاهده نمود.

۳ مثلث سرپینسکی (الگوریتم تصادفی)

کد این بخش از تمرین در فایل q3.py قابل مشاهده است. روش کار به این صورت است که ابتدا یک object از کلاس RandomSierpinskiTriangle با تعداد نقاط دلخواه می‌سازیم. سپس تابع render را با زمان مورد نظر فراخوانی

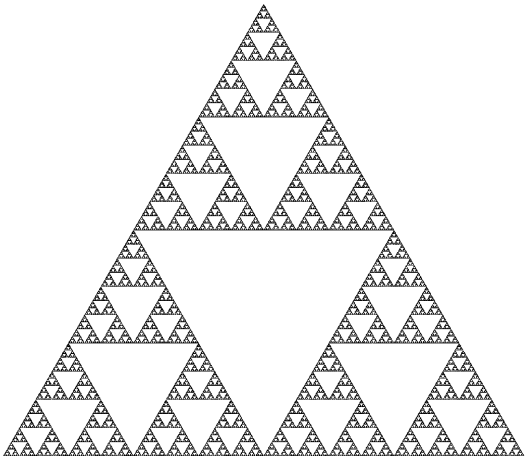


شکل ۱: مجموعه کوخ تا مرحله ۵ ام



شکل ۲: مثلث سرپینسکی تا مرحله ۵ ام

می‌کنیم. این تابع در ابتدا با استفاده از تابع __random_points__ به اندازه‌ی تعداد داده شده، نقاط با مختصات تصادفی در مثلث دلخواه تولید می‌کند. برای این کار نقاط را از یک مستطیل به عرض $\frac{1}{2}$ و ارتفاع $\frac{\sqrt{3}}{2}$ ضلع مثلث انتخاب می‌کند و نقاط بالاتر از قطر را معادل با نقاط نیمه دوم در نظر می‌گیرد. پس از تولید این نقاط، تابع render در یک حلقه به اندازه‌ی زمان داده شده، نقاط راس مرحله‌ی قبلی را به تابع __random_transform__ پاس می‌دهد و نقاط جدید را در خانه‌ی بعدی متغیر stages_points ذخیره می‌کند. تابع __random_transform__ به این صورت عمل می‌کند که ابتدا از بین ۳ نوع تبدیل خودمتشابه‌ای که برای مثلث سرپینسکی وجود دارد، به طور تصادفی و به اندازه‌ی تعداد نقاط داده شده، تبدیل انتخاب می‌کند و آن‌ها را بر روی نقاط اولیه اثر می‌دهد. این تبدیل‌ها همان تبدیل‌هایی هستند که در سوال قبلی توضیح داده شده‌اند. لازم به ذکر است که تمامی مختصات به صورت اعداد مختلط ذخیره شده‌اند و عمل انتقال از طریق جمع و عمل دوران نیز از طریق ضرب اعداد مختلط انجام شده است. نتیجه به دست آمده برای مرحله‌ی ۹ ام و با 1 000 000 نقطه را در شکل ۳ می‌توان مشاهده نمود.



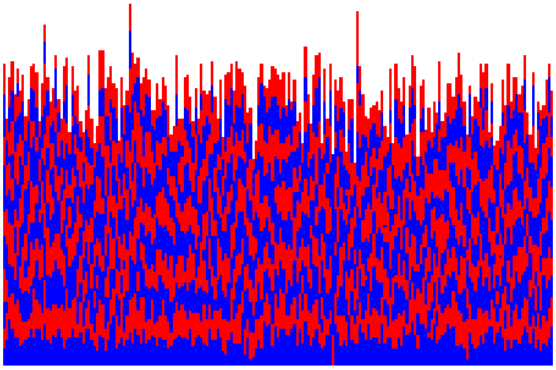
شکل ۳: مثلث سرپینسکی با الگوریتم تصادفی در مرحله‌ی ۹ با 1 000 000 نقطه

۴ ول‌نشست

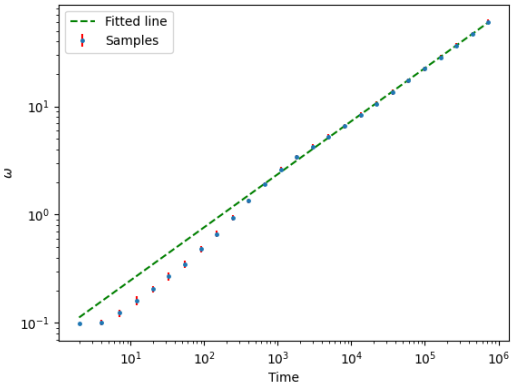
کد این بخش از تمرین در فایل q4.py قابل مشاهده است. روش کار به این صورت است که ابتدا یک object از کلاس RandomBallisticDeposition با طول دلخواه می‌سازیم. سپس تابع render را با زمان مورد نظر فراخوانی می‌کنیم. روش کار این تابع به این صورت است که روی بازه‌های زمانی داده شده پیمایش می‌کند و در هر بازه، به اندازه‌ی طول آن، اعداد تصادفی از مجموعه‌ی اندیس‌ها تولید می‌کند. سپس تعداد تکرار هر یک از اندیس‌ها را می‌شمارد و به ارتفاع آن خانه اضافه می‌کند. در نهایت داده‌های به دست آمده را در یک فایل به فرمت npy. ذخیره می‌کند. برای رسم تصویر داده‌ها باید تابع show را صدا بزنیم. تصویر به دست آمده برای طول 200 و زمان 20 000 را در شکل ۴ می‌توان مشاهده نمود.
برای ایجاد یک آنسامبل آماری، می توانیم تابع make_ensemble را با بازه‌ی زمانی و تعداد دلخواه فراخوانی کنیم. این تابع به تعداد داده شده، تابع render را صدا می‌زند و در نهایت میانگین ارتفاع و ω را برای این آنسامبل محاسبه کرده و در یک فایل ذخیره می‌کند. برای انجام تحلیل بر روی این داده‌های به دست آمده و محاسبه شیب رشد، تابع analyse را صدا می‌زنیم. منحنی تغییرات ناهمواری و میانگین ارتفاع بر حسب زمان را برای ول‌نشستی به طول 200 و با 200 تکرار را در شکل‌های ۵ و ۶ می‌توان مشاهده نمود.
شیب خط فیت شده به این دو منحنی را در جدول ۱ همراه با خطای محاسبه‌شان آورده شده است.

میانگین ارتفاع	1.001406 ± 10^{-6}
ناهمواری (ω)	$\beta = 0.4904 \pm 10^{-4}$

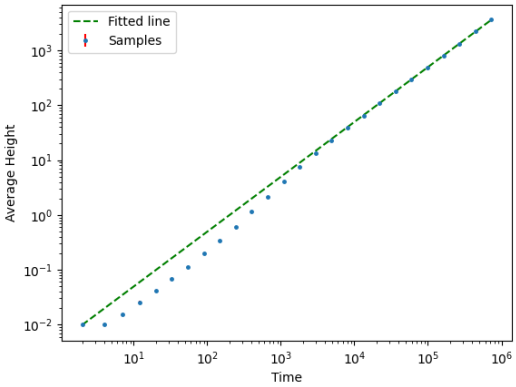
جدول ۱: شیب خطوط فیت شده به منحنی تغییرات میانگین ارتفاع و ناهمواری ول‌نشستی به طول ۲۰۰



شکل ۴: ول‌نشست به طول 200 و زمان 20 000



شکل ۵: منحنی تغییرات ناهمواری بر حسب زمان برای ول‌نشستی به طول 200 و با 200 بار تکرار



شکل ۶: منحنی تغییرات میانگین ارتفاع بر حسب زمان برای ول‌نشستی به طول 200 و با 200 بار تکرار

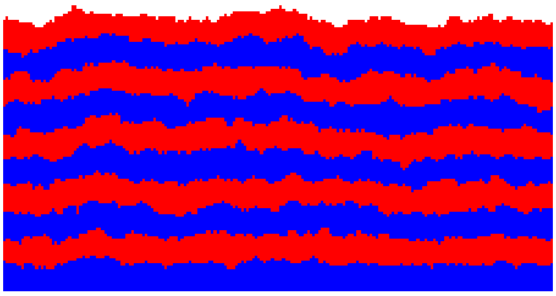
۵ پایین‌نشست

کد این بخش از تمرین در فایل q5.py قابل مشاهده است. روش کار به این صورت است که ابتدا یک object از کلاس RandomBallisticWithRelaxation با طول دلخواه می‌سازیم. سپس تابع render را با زمان مورد نظر فراخوانی می‌کنیم. روش کار این تابع به این صورت است که روی بازه‌های زمانی داده شده پیمایش می‌کند و در هر بازه، به اندازه‌ی طول آن، اعداد تصادفی از مجموعه اندیس‌ها تولید می‌کند. سپس روی این اندیس‌های تصادفی پیمایش می‌کند و کمترین ارتفاع را در همسایگی آن پیدا کرده و ارتفاع آن خانه را یکی افزایش می‌دهد. در نهایت داده‌های به دست آمده را در یک فایل به فرمت npy. ذخیره می‌کند. برای رسم تصویر داده‌ها باید تابع show را صدا بزنیم. تصویر به دست آمده برای طول 200 و زمان 20 000 را در شکل ۷ می‌توان مشاهده نمود.
برای ایجاد یک آنسامبل آماری، می توانیم تابع make_ensemble را با بازه‌ی زمانی و تعداد دلخواه فراخوانی کنیم. این تابع به تعداد داده شده، تابع render را صدا می‌زند و در نهایت میانگین ارتفاع و ω را برای این آنسامبل محاسبه کرده و در یک فایل ذخیره می‌کند. برای انجام تحلیل بر روی این داده‌های به دست آمده و محاسبه شیب رشد، تابع analyse را صدا می‌زنیم. منحنی تغییرات ناهمواری و میانگین ارتفاع بر حسب زمان را برای ول‌نشستی به طول 200 و با 50 تکرار را در شکل‌های ۸ و ۹ می‌توان مشاهده نمود. همان‌طور که دیده می‌شود، برای مشاهده‌ی رفتار اشباع، به بیش از 100 000 ذره نیاز است.
شیب خط فیت شده به این دو منحنی و مقدار اشباع برای طول‌های مختلف را در جدول ۲ همراه با خطای محاسبه‌شان

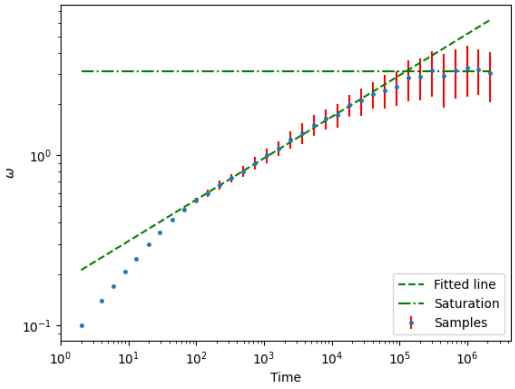
آورده شده است. با توجه مقادیر اشباع به دست آمده، منحنی تغییرات اشباع بر حسب طول را در شکل ۱۰ رسم می‌کنیم. شیب خط فیت شده به آن برابر است با:

$$\alpha = 0.4986 \pm 10^{-4} \tag{۱}$$

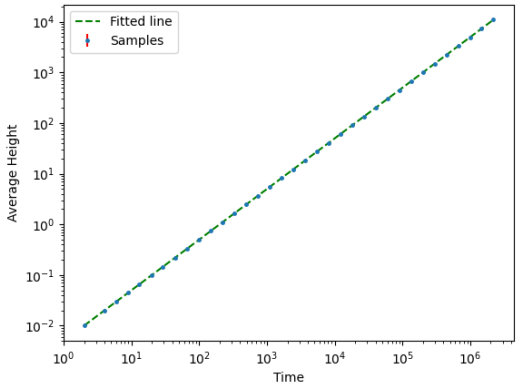
$$\alpha = z\beta \overset{(۱)}{\Longrightarrow} z = 2.046 \pm 10^{-3} \tag{۲}$$



شکل ۷: پایین‌نشست به طول 200 و زمان 20 000



شکل ۸: منحنی تغییرات ناهمواری بر حسب زمان برای پایین‌نشستی به طول 200 و با 50 بار تکرار



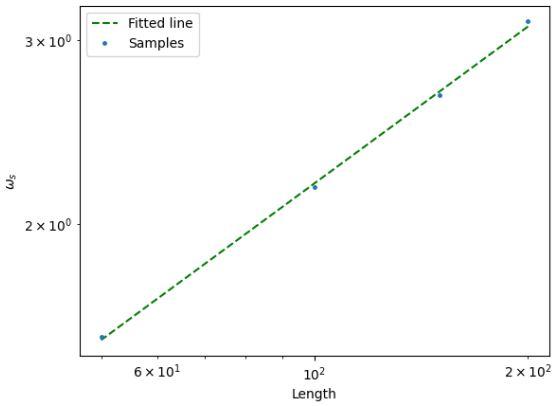
شکل ۹: منحنی تغییرات میانگین ارتفاع بر حسب زمان برای پایین‌نشستی به طول 200 و با 50 بار تکرار

۶ کنارنشست

کد این بخش از تمرین در فایل q6.py قابل مشاهده است. روش کار به این صورت است که ابتدا یک object از کلاس NearestNeighborBallisticDeposition با طول دلخواه می‌سازیم. سپس تابع render را با زمان مورد نظر فراخوانی می‌کنیم. روش کار این تابع به این صورت است که روی بازه‌های زمانی داده شده پیمایش می‌کند و در

طول (l)	میانگین ارتفاع	ناهمواری (ω)	اشباع (ω_s)
200	1.0 ± 10^{-32}	$\beta = 0.24366 \pm 10^{-5}$	3.124 ± 10^{-3}
150	1.0 ± 10^{-32}	$\beta = 0.2428 \pm 10^{-4}$	2.655 ± 10^{-3}
100	1.0 ± 10^{-32}	$\beta = 0.2363 \pm 10^{-4}$	2.168 ± 10^{-3}
50	1.0 ± 10^{-31}	$\beta = 0.2320 \pm 10^{-4}$	1.5575 ± 10^{-4}

جدول ۲: شیب خطوط فیت شده به منحنی تغییرات میانگین ارتفاع و ناهمواری و مقدار اشباع برای طول‌های مختلف پایین‌نشست



شکل ۱۰: منحنی تغییرات اشباع بر حسب طول پایین‌نشست

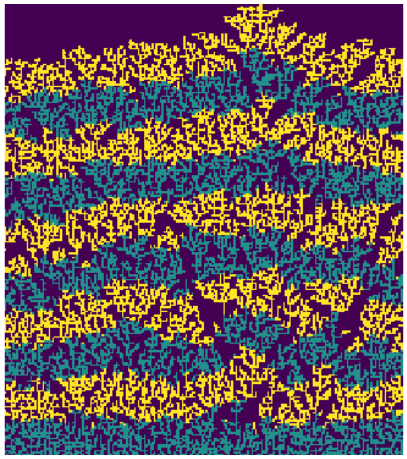
هر بازه، به اندازه‌ی طول آن، اعداد تصادفی از مجموعه اندیس‌ها تولید می‌کند. سپس روی این اندیس‌های تصادفی پیمایش می‌کند و بیش‌ترین ارتفاع را در همسایگی آن پیدا کرده و ارتفاع آن خانه را برابر با آن قرار می‌دهد. در نهایت داده‌های به دست آمده را در یک فایل به فرمت npy. ذخیره می‌کند. برای رسم تصویر داده‌ها باید تابع show را صدا بزنیم. تصویر به دست آمده برای طول 200 و زمان 20 000 را در شکل ۱۱ می‌توان مشاهده نمود.
برای ایجاد یک آنسامبل آماری، می توانیم تابع make_ensemble را با بازه‌ی زمانی و تعداد دلخواه فراخوانی کنیم. این تابع به تعداد داده شده، تابع render را صدا می‌زند و در نهایت میانگین ارتفاع و ω را برای این آنسامبل محاسبه کرده و در یک فایل ذخیره می‌کند. برای انجام تحلیل بر روی این داده‌های به دست آمده و محاسبه شیب رشد، تابع analyse را صدا می‌زنیم. منحنی تغییرات ناهمواری و میانگین ارتفاع بر حسب زمان را برای کنارنشستی به طول 200 و با 50 تکرار را در شکل‌های ۱۲ و ۱۳ می‌توان مشاهده نمود.
شیب خط فیت شده به این دو منحنی و مقدار اشباع برای طول‌های مختلف را در جدول ۳ همراه با خطای محاسبه‌شان آورده شده است. با توجه به مقادیر اشباع به دست آمده، منحنی تغییرات اشباع بر حسب طول را در شکل ۱۴ رسم می‌کنیم. شیب خط فیت شده به آن برابر است با:

$$\alpha = 0.460 \pm 10^{-3} \tag{۳}$$

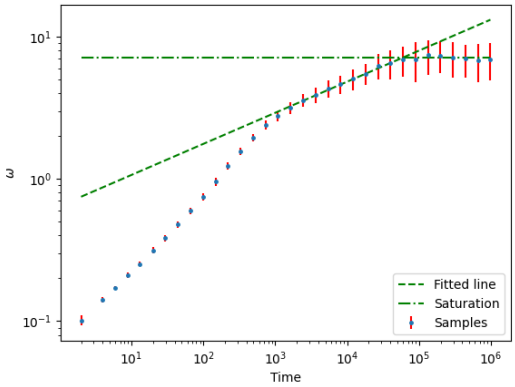
$$\alpha = z\beta \overset{(۳)}{\Longrightarrow} z = 2.10 \pm 10^{-2} \tag{۴}$$

۷ ول‌نشست رقابتی

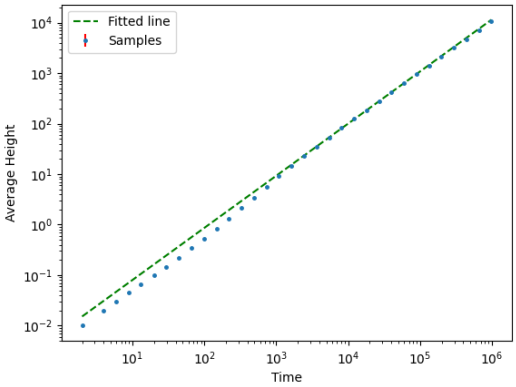
کد این بخش از تمرین در فایل q7.py قابل مشاهده است. روش کار به این صورت است که ابتدا یک object از کلاس NearestNeighborBallisticDepositionWithInitialCondition با طول دلخواه می‌سازیم. سپس تابع render را



شکل ۱۱: کنارنشست به طول 200 و زمان 20 000



شکل ۱۲: منحنی تغییرات ناهمواری بر حسب زمان برای کنارنشستی به طول 200 و با 50 بار تکرار



شکل ۱۳: منحنی تغییرات میانگین ارتفاع بر حسب زمان برای کنارنشستی به طول 200 و با 50 بار تکرار

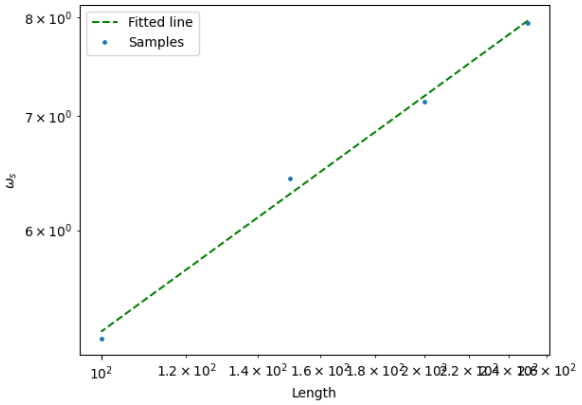
با زمان مورد نظر فراخوانی می‌کنیم. روش کار این تابع به این صورت است که روی بازه‌های زمانی داده شده پیمایش می‌کند و در هر بازه، به اندازه‌ی طول آن، اعداد تصادفی از مجموعه اندیس‌ها تولید می‌کند. سپس روی این اندیس‌های تصادفی پیمایش می‌کند و بیش‌ترین ارتفاع را در همسایگی آن، در صورتی که غیر صفر باشد، پیدا کرده و ارتفاع آن خانه را برابر با آن قرار می‌دهد. در نهایت داده‌های به دست آمده را در یک فایل به فرمت npy. ذخیره می‌کند. برای رسم تصویر داده‌ها باید تابع show را صدا بزنیم. تصویر به دست آمده برای طول 200 و زمان 18 000 را در شکل ۱۵ می‌توان مشاهده نمود.

برای ایجاد یک آنسامبل آماری، می توانیم تابع make_ensemble را با بازه‌ی زمانی و تعداد دلخواه فراخوانی کنیم. این تابع به تعداد داده شده، تابع render را صدا می‌زند و در نهایت میانگین ارتفاع و ω را برای این آنسامبل محاسبه کرده و در یک فایل ذخیره می‌کند. برای انجام تحلیل بر روی این داده‌های به دست آمده و محاسبه شیب رشد، تابع analyse را صدا می‌زنیم. منحنی تغییرات عرض، میانگین ارتفاع و ناهواری بر حسب زمان را برای کنارنشستی رقابتی به طول 200 و با 100 تکرار را در شکل‌های ۱۶ و ۱۷ و ۱۸ می‌توان مشاهده نمود.

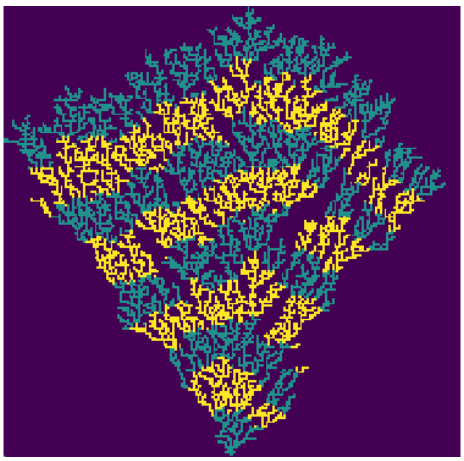
شیب خط فیت شده به این سه منحنی را در جدول ۴ همراه با خطای محاسبه‌شان آورده شده است. همان‌طور که دیده می‌شود، نماهای بحرانی کاملا متفاوتی نسبت به باقی مدل‌های بررسی شده دارد.

اشباع (ω_s)	ناهمواری (ω)	میانگین ارتفاع	طول (l)
7.94 ± 10^{-2}	$\beta = 0.2346 \pm 10^{-4}$	1.0261 ± 10^{-4}	250
7.14 ± 10^{-2}	$\beta = 0.219 \pm 10^{-3}$	1.0322 ± 10^{-4}	200
6.436 ± 10^{-3}	$\beta = 0.2113 \pm 10^{-4}$	1.0398 ± 10^{-4}	150
5.177 ± 10^{-3}	$\beta = 0.242 \pm 10^{-3}$	1.0703 ± 10^{-4}	100

جدول ۳: شیب خطوط فیت شده به منحنی تغییرات میانگین ارتفاع و ناهمواری و مقدار اشباع برای طول‌های مختلف کنارنشست



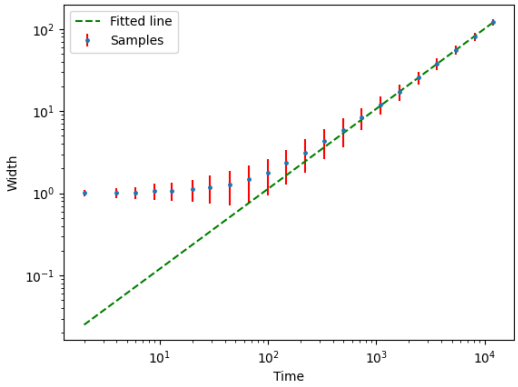
شکل ۱۴: منحنی تغییرات اشباع بر حسب طول کنارنشست



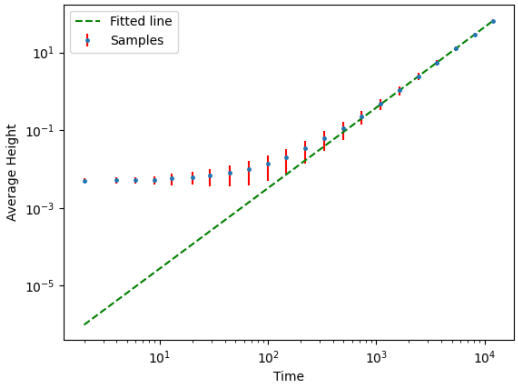
شکل ۱۵: ولنشست رقابتی به طول 200 و زمان 18 000

میانگین ارتفاع	2.0697 ± 10^{-4}
عرض	0.97357 ± 10^{-5}
ناهمواری (ω)	$\beta = 1.4663 \pm 10^{-4}$

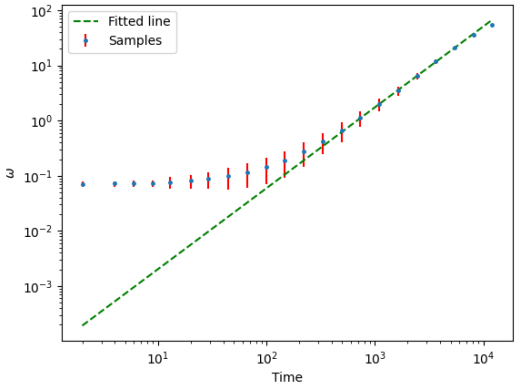
جدول ۴: شیب خطوط فیت شده به منحنی تغییرات میانگین ارتفاع و عرض ولنشستی رقابتی به طول ۲۰۰



شکل ۱۶: منحنی تغییرات عرض بر حسب زمان برای کنارنشستی رقابتی به طول 200 و با 100 بار تکرار



شکل ۱۷: منحنی تغییرات میانگین ارتفاع بر حسب زمان برای کنارنشستی رقابتی به طول 200 و با 100 بار تکرار



شکل ۱۸: منحنی تغییرات ناهمواری بر حسب زمان برای کنارنشستی رقابتی به طول 200 و با 100 بار تکرار