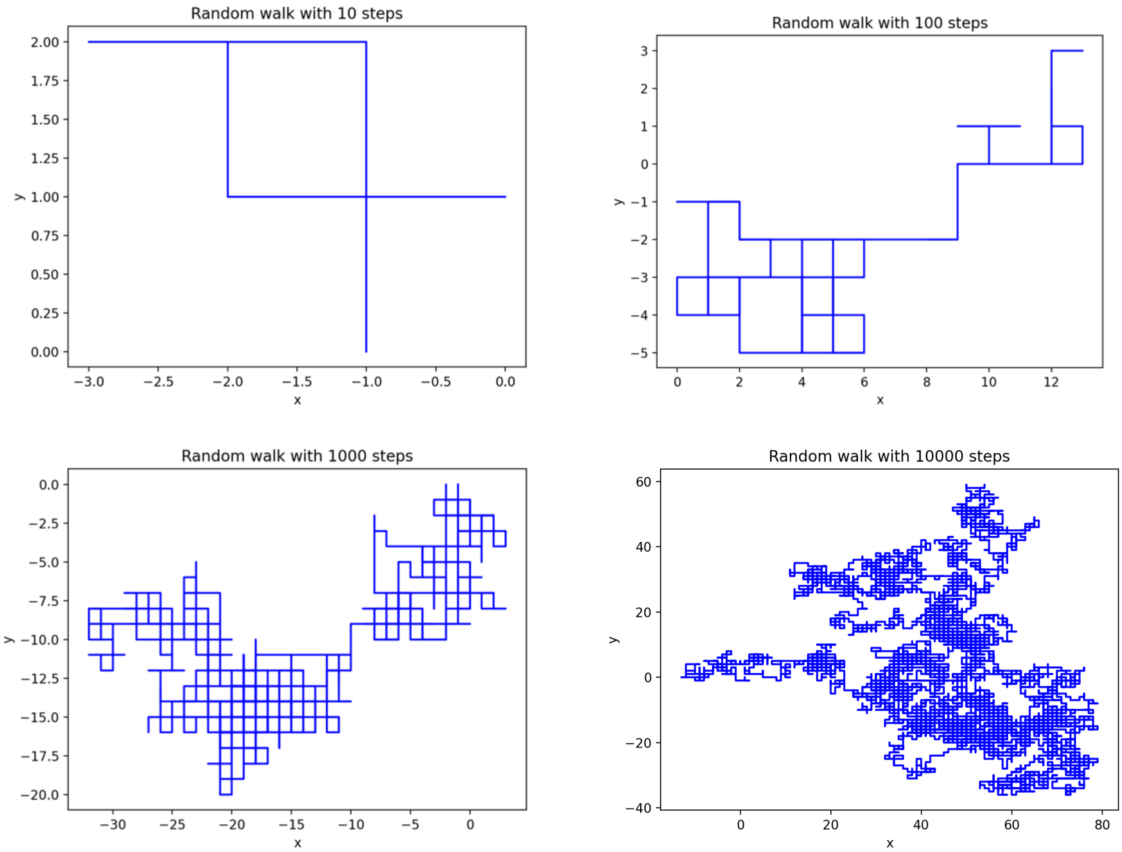
Project 2 – SI1336 Simulation and modeling

Jennifer Ly

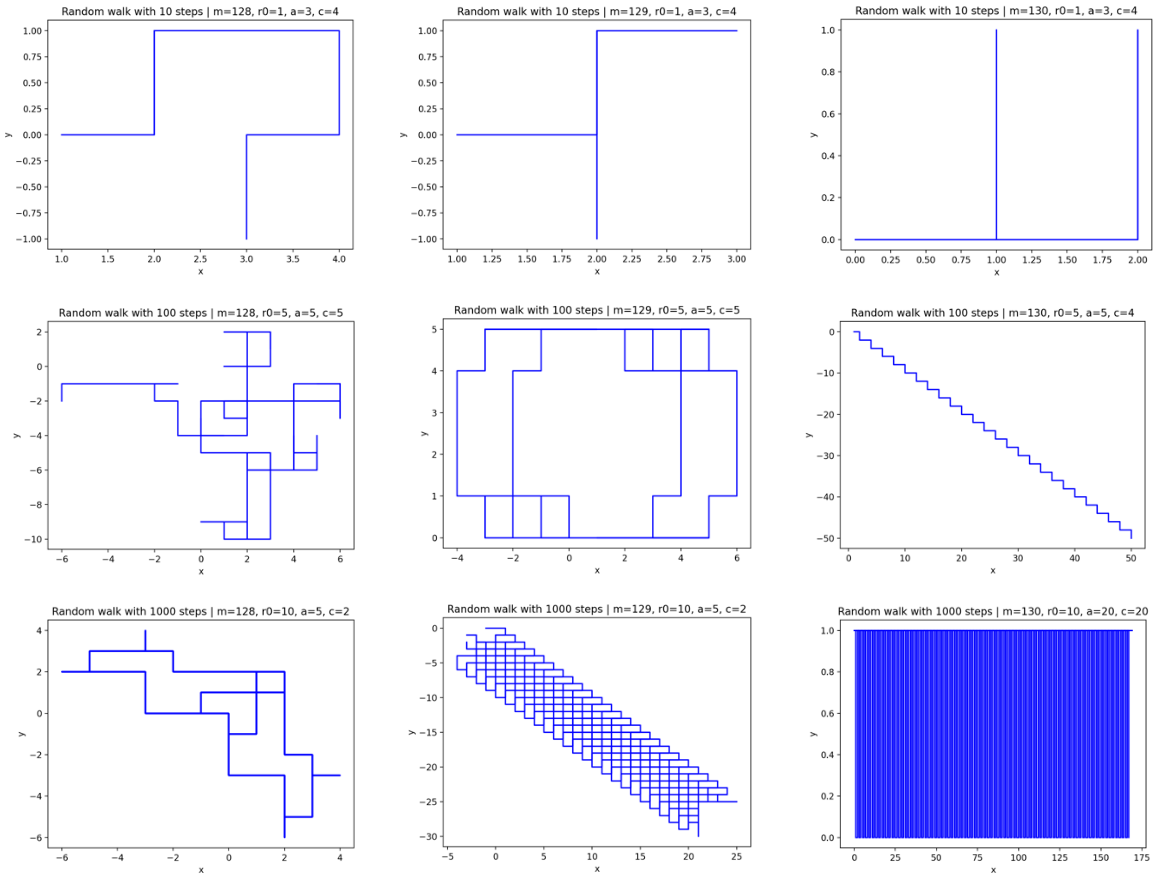
November 18, 2020

2.1a. Generate a 2D random walk with single steps along x and y with random generator rnd.random(). Plot for 10, 100, 1000 steps.



An additional plot with 10000 steps was added to illustrate the difference between the different methods used in 2.1a. and 2.2b.

2.1b. Generate a 2D random walk with single steps along x and y with random generator $r_n = (a * r_{n-1} + c)\%m$. How does the walk look like for $r_0 = 1, a = 3, c = 4$, $m = 128, 129, 130$?



Above plots illustrates the random walk for different values on the constants r0, a, c, m and step N.

Comparing the results in 2.1b with the results in 2.1a, it can be observed that the random walk looks similar for small values on the constants. However, as observed this random generator can generate regular patterns which is not desired. It can fill up the entire diagram for some values on the constants, whereas for 2.1a the walk does not even give such a pattern for a step number = 10000.

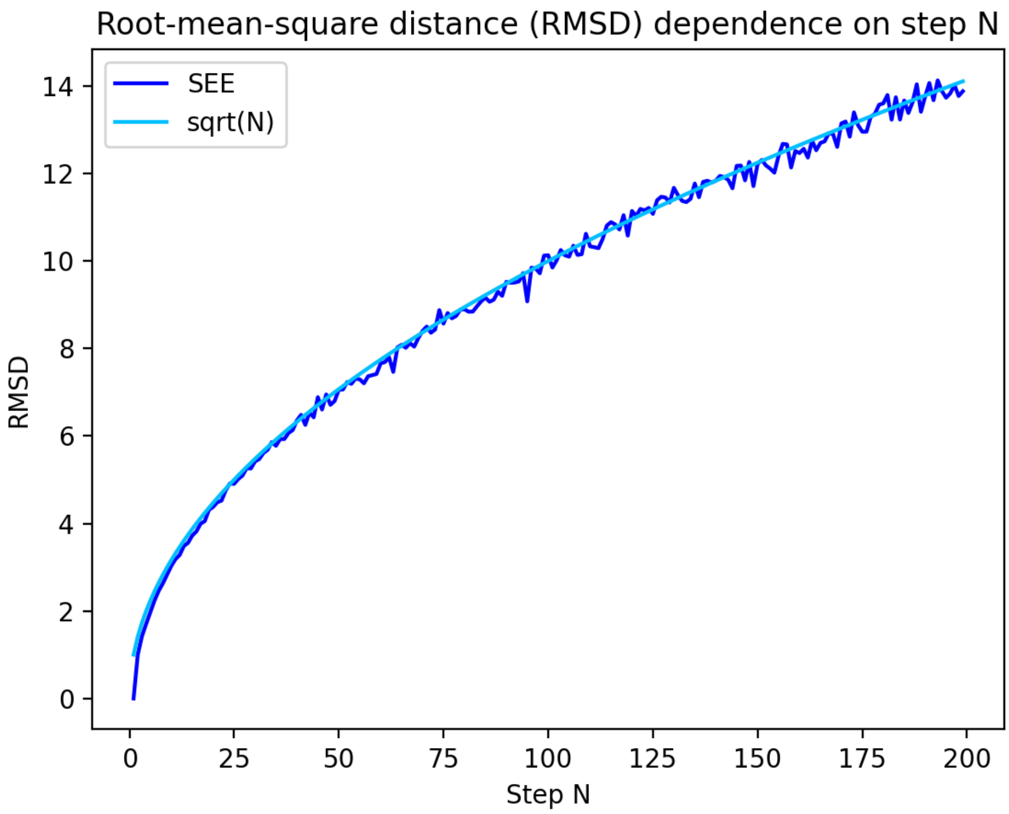
2.1c. Determine the root-mean-squared end-to-end distance and root-mean-square fluctuation. How does the length depend on step N?

If you collect some values of the root-mean-squared end-to-end distance for some values on the step N, you get:

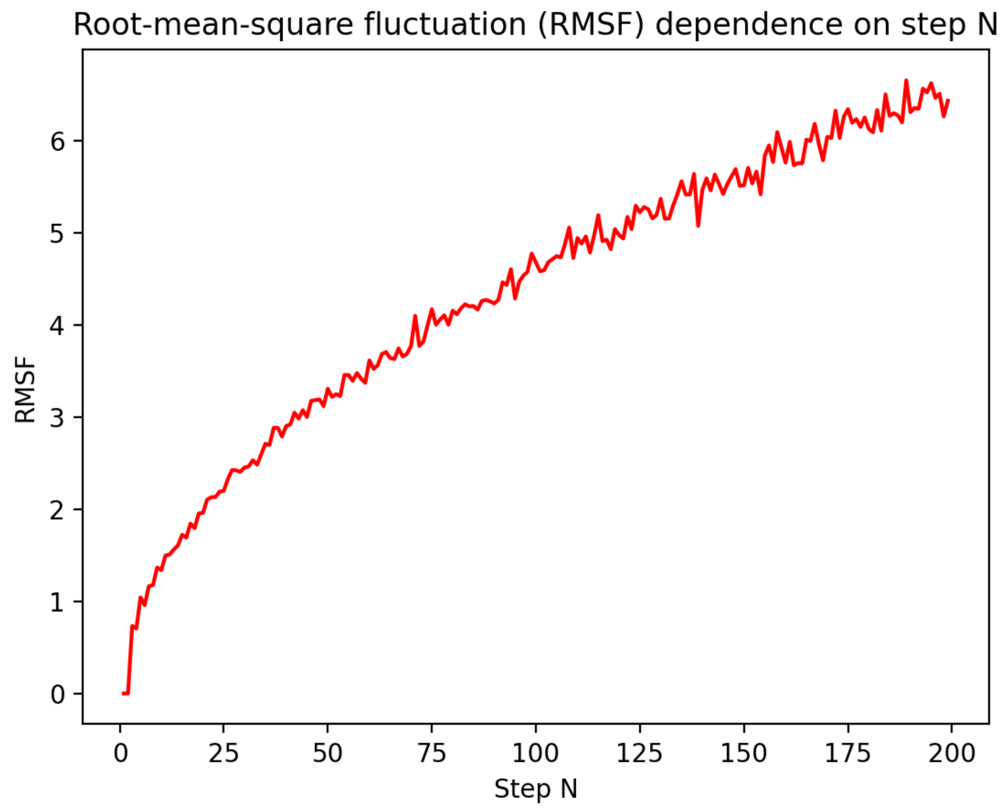
$N = 10 \rightarrow RMSD = 3.111269837220809$
$N = 100 \rightarrow RMSD = 10.04987562112089$
$N = 1000 \rightarrow RMSD = 30.54766766874355$

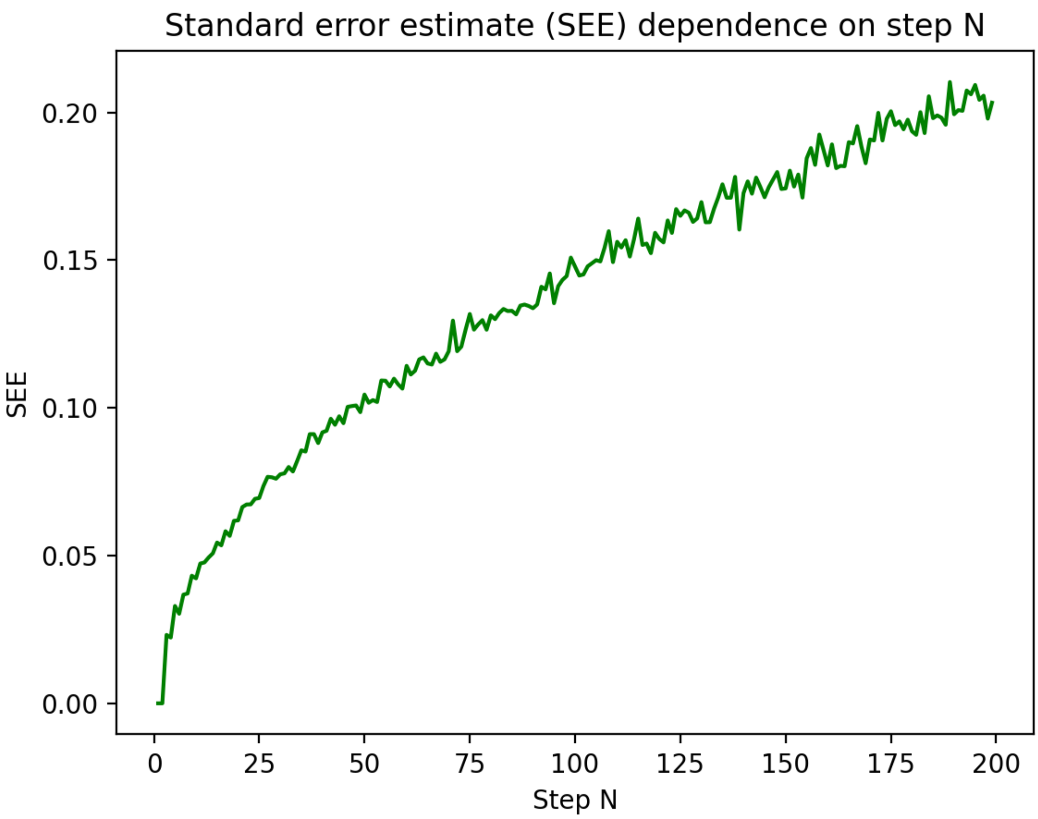
Conclusion: $RMSD \approx \sqrt{N}$

This can be verified by plotting the RMSD dependence on N and $\sqrt{N}$ in the same plot.

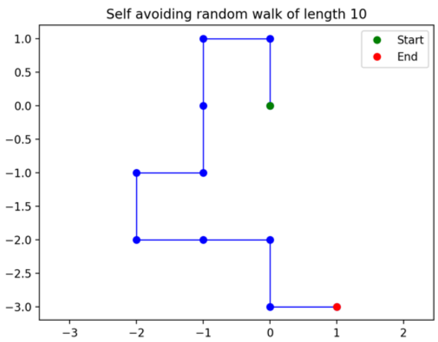


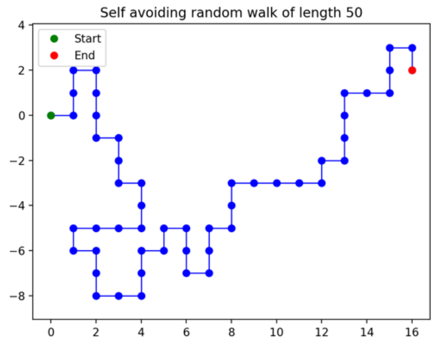
Plot the root-mean-square fluctuation and the standard error estimate.

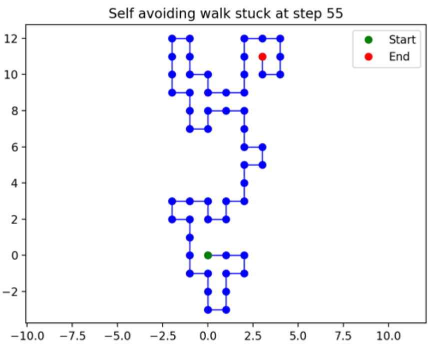




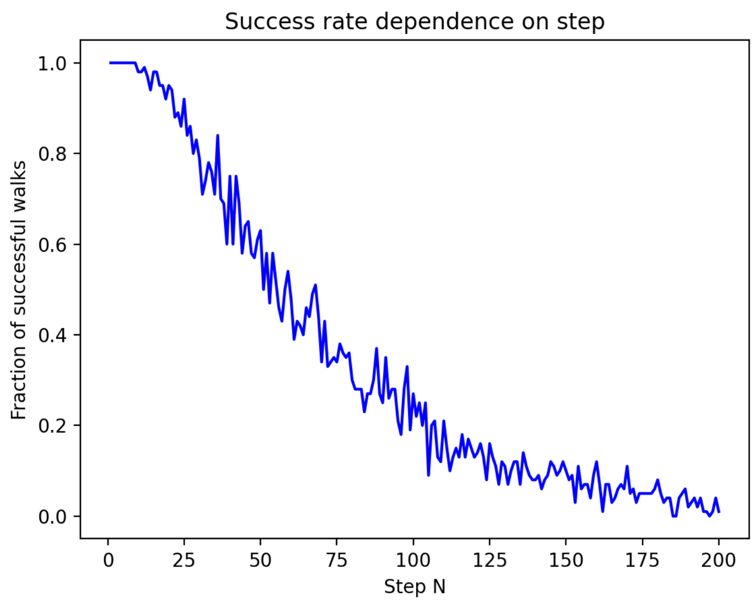
2.1d. How does the fraction of successful walks depend on N? What is the maximum N?





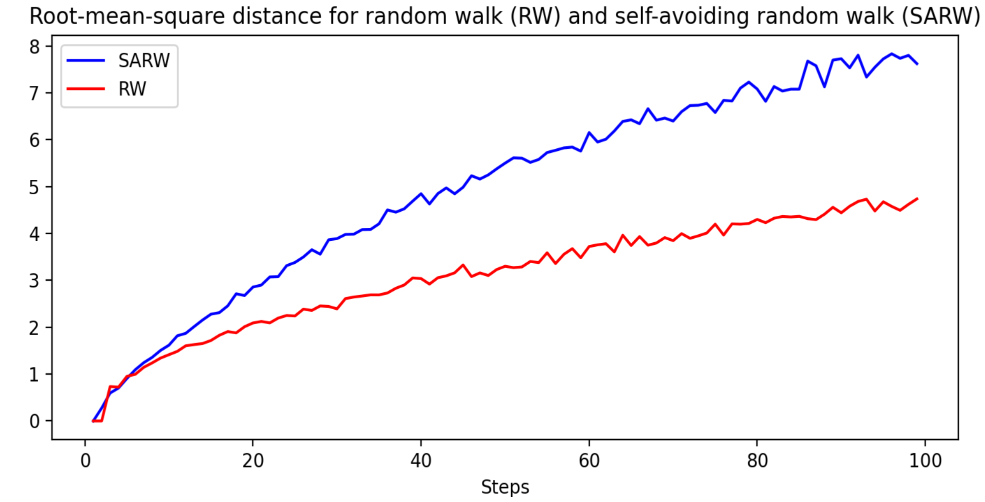


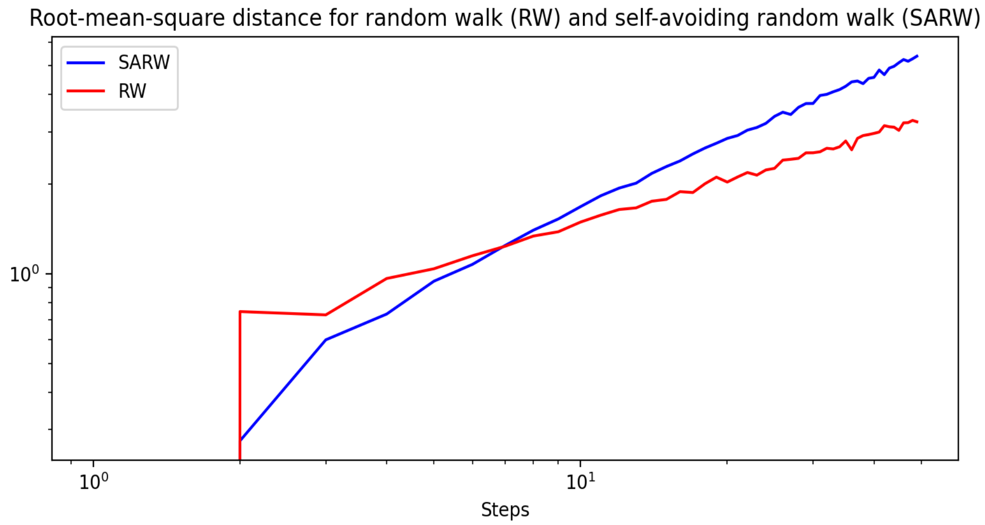
An increase in N also increases the chance of the walk getting stuck. When running the program for different values of N, the majority of the walks got stuck around step N=50-60. However, because it is a randomized process there is a chance that some walks survive beyond that number, which can be illustrated in the last picture to the right with a step of N=163. The success ratio dependence on step N:



As observed from the plot, the ratio of successful walks decreases as the step number increases. For 100 self-avoiding random walks with step N=100, the mean value of the step where the walk got stuck is approximately N=63.

2.1e. Compute the root-mean-square end-to-end distance. What is the difference with the normal random walk vs. self-avoiding random walk?





The main difference between the normal random walk vs. the self-avoiding random walk is that the root-mean-square end-to-end distance (RMSD) increases much faster for the self-avoiding random walk compared to the normal random walk as the step number N increases.

Due to the condition of a self-avoiding random walk to only move in the direction where it has not been before, it takes up more space than a normal random walk (that can move in any direction) and thus has a larger RMSD for step N.

2.2a. Explain the fundamental diagram's qualitive shape. At what density do traffic jams begin to occur?

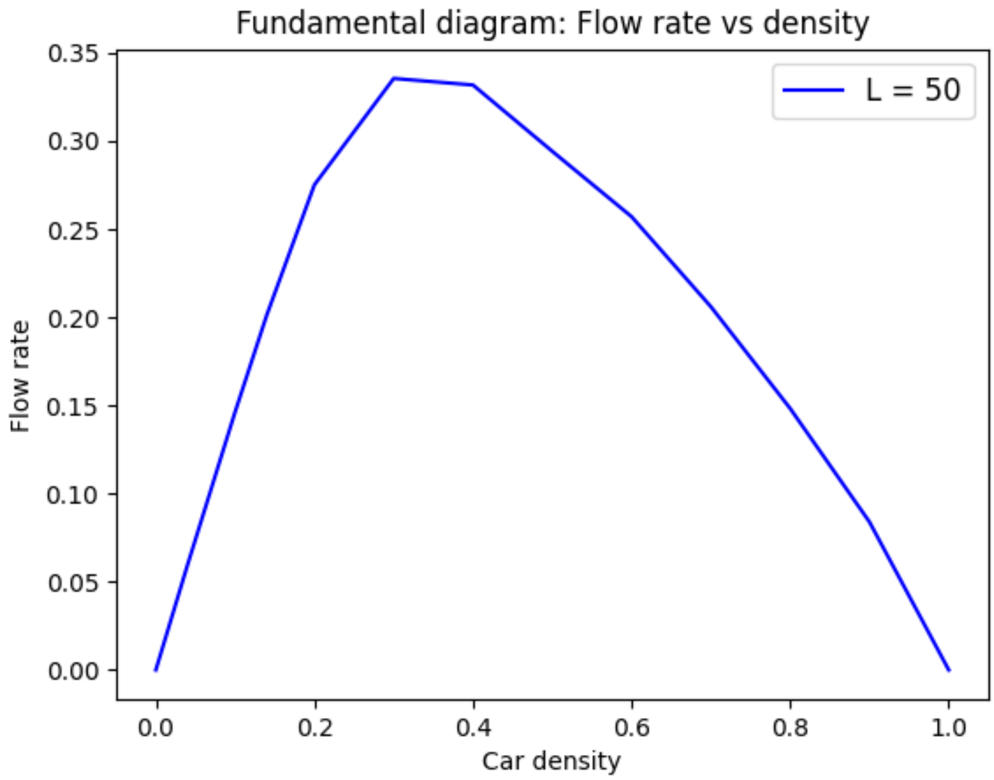


Figure 1: vmax=2, p=0.5, L=50

The flow rate is the velocity at which cars pass a given point on the road length L. As observed from the fundamental diagram above, the flow rate varies depending on the car density, i.e. the number of cars per length unit. The flow rate increases to a certain value of the density and afterwards start to decrease, which in reality represents traffic jams beginning to occur.

At density≈0.3, cars start to jam up as the slope of the curve start to drop.

2.2b. How many simulations do you need to get a standard error of 0.001? Does the equilibration time depend on how you choose the initial conditions?

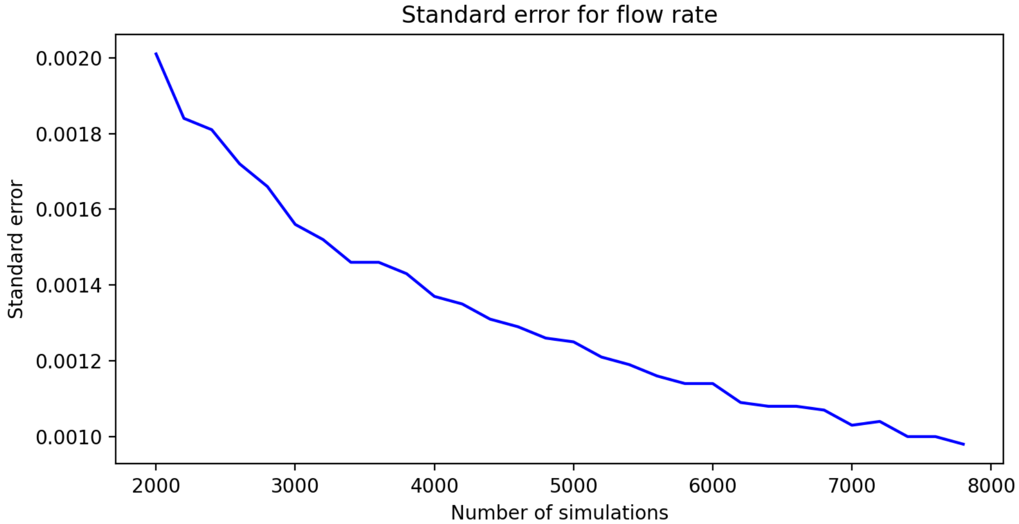


Figure 2: vmax=2, p=0.5, L=50

The number of simulations needed to be done before getting a standard error of 0.001 is approximately 7800-8000 simulations.

The equilibration time do depend on the initial conditions as they are randomized and can result in inaccurate results. To avoid artifacts due to the initial conditions, it is important to let the system equilibrate and “forget” its initial state. That can be achieved by increasing the equilibration time (and only collect the values of the cars after this time). That is also why many simulations are needed to get a small standard error.

2.2c. As the road length shortens, when do the fundamental diagram start to deviate from those of long road lengths?

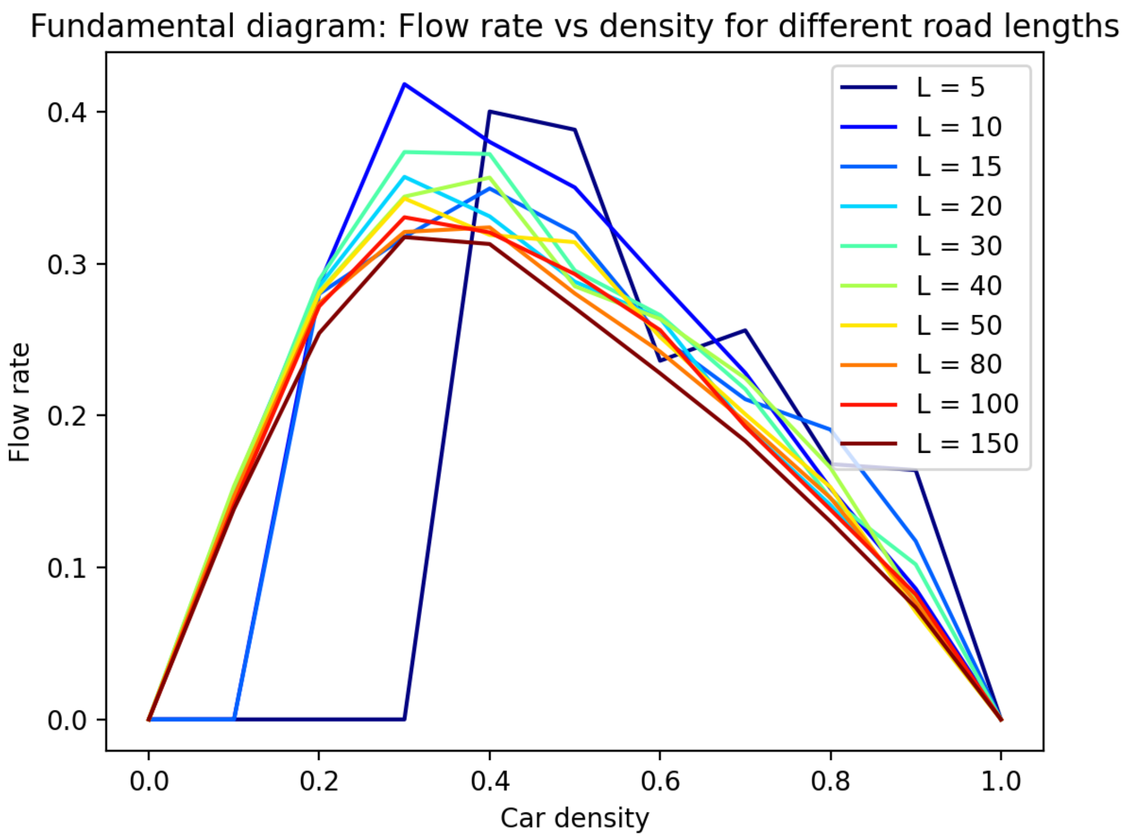


Figure 3: vmax=2, p=0.5, L=see label

As observed from the fundamental diagram, the curves start to deviate from the "normal" pattern at length $L \leq 10$. Even though the flow rate decreases as the density increases as expected, the curve for $L \leq 10$ has zero flow rate up until a certain density and have a very irregular behavior as the density increases.

This can be interpreted as the road being so short that the cars cannot move (hence zero flowrate) in a continuous way without getting jammed up (hence the jagged slope).

2.2d. Are there any quantitative differences in the behavior of the cars?

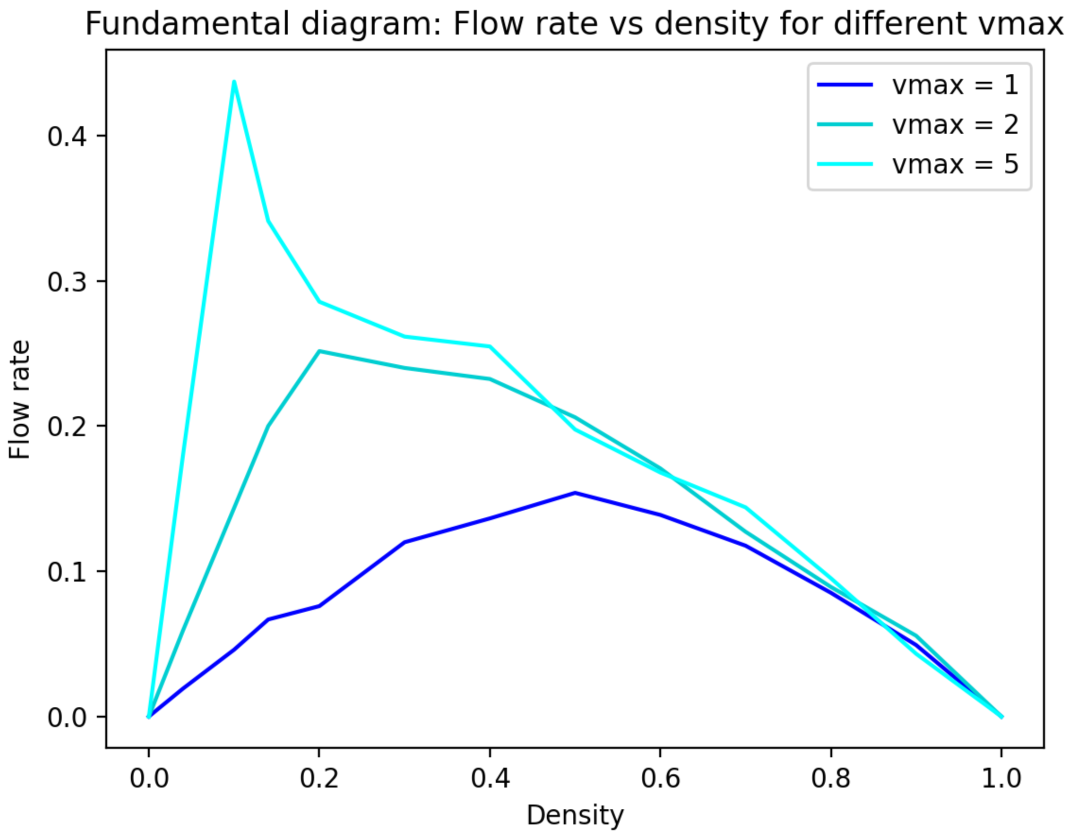


Figure 4: vmax=see label, p=0.5, L=50

As vmax increases, the density of which traffic jams begin to occur decreases while the slope of the flow rate to reach its maximum value increases.

The flow rate also reaches a higher number as vmax increases, which is the result of the total speed of the set of cars increasing due to the condition:

If $v_i < v_{max}$, increase the velocity vi of car i by one unit, that is, $v_i \rightarrow v_i + 1$

As vmax increases, the acceleration to the maximum velocity also increases.

2.2e. Explore the effect of the speed reduction probability by considering p=0.2, 0.8.

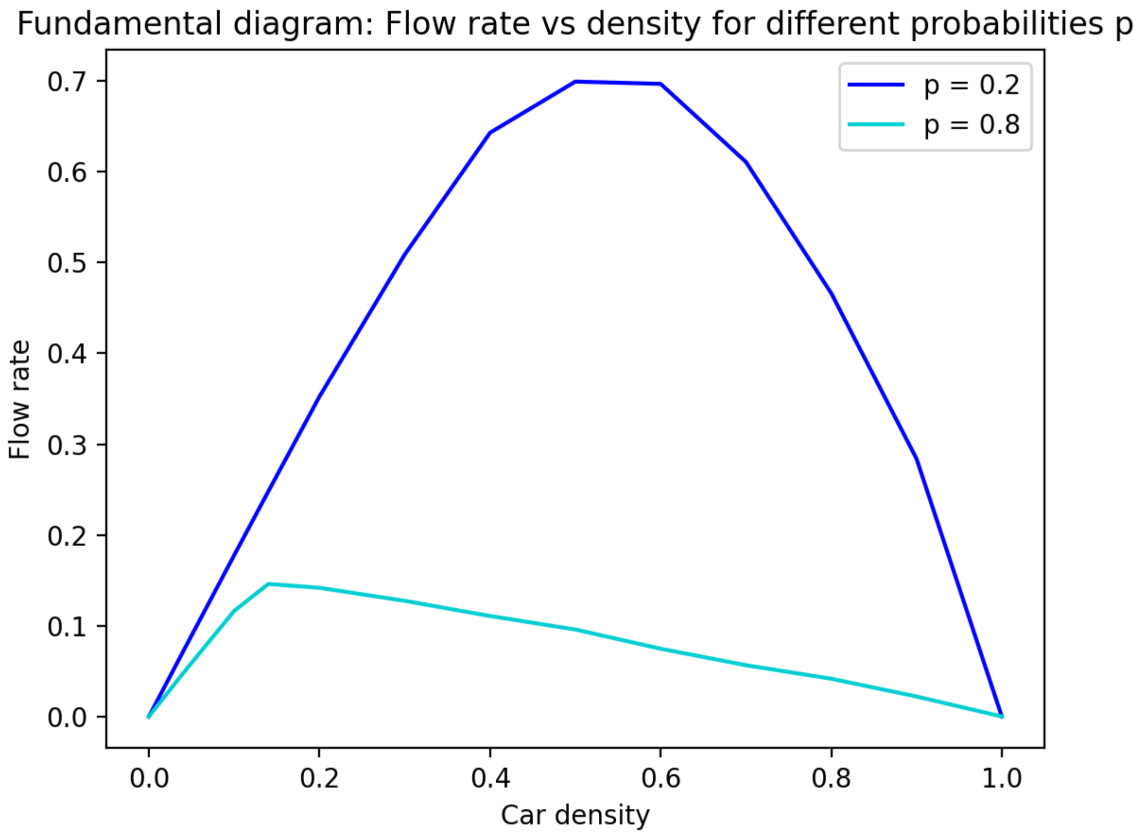


Figure 5: vmax=2, p=see label, L=50

Condition depending on probability p:

> With probability p, reduce the velocity of a moving car by one unit: $v_i \rightarrow v_i - 1$, only do this when $v > 0$ to avoid negative velocities

As I defined the condition according to:

```
comparisonnumber = np.random.rand()                    # Generates random float in the range [0.0, 1.0)
if p > comparisonnumber and self.vel > 0:              # Condition: vi > 0 => vi -> vi-1 with probability p (randomizer)
    self.vel = max(self.vel-1, 0)
```

Where an object is a car, a higher value on p results in a higher chance of the condition being fulfilled. Consequently, the speed and flowrate for every set of cars get reduced much faster for p=0.8 compared to p=0.2, which is what can be observed in the fundamental diagram above.