Projects: Polymers as 2-dimensional random walks

2.1a) Write a program that generates a two-dimensional random walk with single steps along x or y. The simplest way to do this is by generating a number randomly taken out of the sequence (0,1,2,3) by multiplying random.rnd() by 4 and rounding the result down to an integer using int(). Then you can increase x by one for 0, decrease x by one for 1, and the same for y with 2 and 3. Plot random walks for 10, 100 and 1000 steps.

2.1b) Instead of rnd.random() use the simple random generator
$r_n = (a\ r_{n-1} + c)\ \%\ m$ which generates random number in the range 0 to m-1. In Python you can use r//((m + 3)//4) to convert r to an integer between 0 and 3. How does the walk look like for r_0=1, a=3, c=4, m=128? Also try m=101 and some other values for all four parameters.

2.1c) Switch back to rnd.random(). Generate many random walks for a few, not all 1000, values of the step number in the range from 1 to 1000 and determine the root mean squared end-to-end distance $\mathrm{sqrt}(\langle R^2\rangle)$ for each length. How does the length depend on N? Also plot the estimate of the root-mean-square fluctuation: $\mathrm{sqrt}((\langle R^2\rangle-\langle R\rangle^2)\#\mathrm{walks}/(\#\mathrm{walks}-1))$, as well as the standard error estimate.

polymers (continued)

2.1d) A real polymer can not cross itself, i.e. different atoms can not occupy the same space. Generate self-avoiding random walks by storing all previously visited sites of the same walk and terminate and discard the walk when a previously visited is revisited. How does the fraction of successful walks depend on N? What is the maximum value of N that you can reasonably consider? You can improve the algorithm somewhat by only generating moves in three directions, not back in the direction where you just came from. How does that improve the success?

2.1e) Compute the root mean square end-to-end distances for the self-avoiding random walk for the range of N that you can cover. What is the difference with the normal random walk (suggestion: use log-log scale)?

Projects: Traffic model

2.2 a) Implement the cellular automaton traffic model with periodic boundary conditions. Run it for parameters v_{max}=2 and p=0.5. (Plotting suggestions: plot road on x-axis, time on y-axis and dots for the cars; for a movie plot symbols running along the edge of a square or circle (don't use radians for the simulation though)). Allow the system to equilibrate before recording the flow rate. Average the flow rate over multiple times to get more accurate results. Run with different car densities (number of cars / road length). Plot the flow rate versus the density. This plot is called the fundamental diagram. Explain its qualitative shape. At what density do traffic jams begin to occur?

2.2 b) Now we will estimate the statistical accuracy. Take a road length of 50, use 25 cars, v_{max}=2 and p=0.5. Run multiple simulations, for each collect the flow rate over 100 time steps. Compute the standard error of the flow rate. How many simulations do you need to get a standard error of 0.001? How long does the equilibration need to be to get accurate results? Does the equilibration time depend on how you choose the initial conditions?

Projects: Traffic model (continued)

2.2 c) When do the results, i.e. the fundamental diagram, become independent of road length? Or conversely, when shortening the road length, when do the results start to deviate from those of long road lengths? Be aware that longer roads need longer equilibration times.

2.2 d) For a fixed road length, compare your results for $v_{max}=1$ with your results for $v_{max}=2$. Also consider $v_{max}=5$. Are there any quantitative differences in the behavior of the cars?

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

Tip: you can compute the (periodic) distance d_n of car n to car n+1 as:

$$d_n = (pos_{n+1} - pos_n + road\_length) \, \% \, road\_length$$