Submit your solutions by 18.4.25.

IMPORTANT: To submit your solution material for this exercise sheet, use the assigned sciebo folder, and create a new subfolder named "S1" for uploading the solution material of this exercise sheet. For exercises marked by **V** you must upload a video in addition to uploading your code. The video should show your computer screen, and your voice should explain your code, and in addition show and explain the results that you obtain from running the code. For those exercises that are not marked by **V**, just upload a pdf-file that contains your solution.

1.1. Nagel-Schreckenberg traffic model (V)

The phenomenon of "traffic jams out of nowhere" is something that probably everyone has already experienced in freeway traffic: Without any obvious reason, such as congestion or road blockages due to construction or accidents, traffic jams form and then dissipate just as unexpectedly.

The first formulation of a model that predicts such a phantom traffic jam as a consequence of dawdling and overreactions when braking was achieved in 1992 at the University of Cologne, with the so-called Nagel-Schreckenberg model. This model is based on a one-dimensional cellular automaton, which you will implement in this exercise. The cells represent road sections that are either occupied by a vehicle or not. Each vehicle has an additional parameter besides its position, which is its speed. For simplicity, the possible values for speed are limited to the numbers 0, 1, 2, 3, 4, 5. We may relate this model to reality using the following mapping:

- The length of a cell corresponds to 7.5 m.
- The speed i corresponds to $i \cdot 27$ km/h.
- One iteration step corresponds to 1 s.

The iteration step of this cellular automaton consists of 4 parts:

1. Acceleration: The speed of each vehicle is increased by 1 as long as the maximum speed 5 has not been reached.
2. Braking: For all vehicles whose distance d to the vehicle in front is less than or equal to their speed (i.e., they would crash in the next step if the vehicle in front had a speed of 0), the speed is reduced to $d - 1$ (to avoid a crash).
3. Dawdling: Each vehicle (that is not stationary) reduces its speed by one unit with probability p.
4. Movement: Each vehicle finally moves forward by its speed. Here, we interpret the speed as the number of cells advanced per time step.

All vehicles go through this cycle simultaneously, meaning the values from the previous cycle set are used for each vehicle and not the partially updated ones.

For this exercise, we consider a circular (racetrack-like) road, which has to be taken into account in the implementation (we are thus using periodic boundary conditions). You may set the number of cells along this circular road to $N_c = 1000$.

(a) Write a code to simulate the Nagel-Schreckenberg model, starting from a random placement of cars, where for each cell the probability to be occupied by a car is equal to $\rho = 0.2$ (ρ thus specifies the car density along the road). We fix the dawdling probability to $p = 0.15$. Follow the evolution of the system for $N_t = 1000$ time steps. Illustrate the resulting evolution of the traffic by plotting a space-time picture (like in special relativity) of the cells's occupation (e.g. using a GLMakie heatmap in Julia). What do you observe?

(b) Now take the same setup, but with different parameters: Consider the „Sunday drivers" scenario, upon setting $\rho = 0.1$, and $p = 0.5$. What do you observe now?

(c) Consider again $p = 0.15$, keeping $\rho = 0.1$, but now arrange all the cars to initially be next to each other, like in front of a traffic light. What happens now?

1.2. Derivative formula

Prove the following 5-point formula for the third derivative:

$$f'''(x) = \frac{f(x+2h) - 2f(x+h) + 2f(x-h) - f(x-2h)}{2h^3} + O(h^2)\,.$$

1.3. Simpson's rule

Implement Simpson's rule for numerical integration of a function $f(x)$ on a finite interval $x \in [a, b]$ using N points of function evaluation. Analyze the error in the integral upon increasing N, in case $f(x) = \sin(x)$ is integrated on $[0, \pi/2]$, using an appropriate plot based on your code.