In-Class Work: Traffic Flow

4. Distance

If you had at the end of last class finished step 3, that is the initialization of the road, then you may continue with your program from last class. If you had not finished step 3, please copy the program
`~kvollmay/share.dir/inclass2019.dir/traffic3.py` into your working directory, and use this program as starting program for today's inclass work.

4a. Next we work on finding the distance between a car and the car in front of it. To get this distance we will avoid in the following to have to check each site in front of a car if it is empty or not. Instead we define an additional array: `carpos` which stores for each car on the road its position on the road. So if the first car (index 0) on the road is on site 3 then `carpos[0]=3`, if the second car is on site 5 then `carpos[1]=5`, etc.. Add to your program of 3. the array `carpos`, initialize `carpos` in the same loop when you initialize the `road` array, and check your program by printing out both the complete `road` and the `carpos` array.

4b. Now add to your program (after the initialization of `road` and `carpos`) a loop over cars on the street to determine (and print out) for each car the distance to the car in front of it. Take into accout the periodic boundary conditions both to determine the car in front and to determine the distance.
Comment: This task is more difficult than it might seem at first glance. Please get me after having thought about this task.

5. Update Velocities

5a. For the update of the velocities we will need a function which determines the smallest number of three integers. Python has a function which does this, for example `min(3,1,9)` gives `1` as result. Test this function with a few print commands

5b. Add to the program that it determines all new velocities (and updates them in the `road` array), and prints out the road with the new velocities. Check your program with print commands.

6. Update Positions

Now you are ready to add to your program the update of the positions. To do so use a new loop over all cars and update both the `carpos` array and the `road` array. For the update of `road` make sure to first copy the new velocity into a variable (e.g. `vnew`), then empty the old site and then put the car in road on its new site `xnew = xold + vnew` with the new velocity (the order of these commands matters). After the update of all positions print the complete road and check if it is what you expected. Using the same parameters as used in `traffic3.py` compare your result with
`~kvollmay/share.dir/inclass2019.dir/traffic6_PCAR03.data`
Rerun the simulation for `PCAR=0.5` and compare your result with

~kvollmay/share.dir/inclass2019.dir/traffic6_PCAR05.data

7. Finish Program (if time)

Now you are set to finish the program for our traffic flow model. Add to your program from 6. the time loop. Compare your result for the case of 100 timesteps (MAXTIMESTEPS = 100), PCAR=0.3, and otherwise the same parameters as before. Compare your output with
~kvollmay/share.dir/inclass2019.dir/traffic7_PCAR03.data
And again change the probability of a car on a road site to PCAR=0.5 and compare your output with
~kvollmay/share.dir/inclass2019.dir/traffic7_PCAR05.data

8. Space-Time Diagrams (IF TIME)

8a. Now we are ready to have a look at the flow of the cars. Switch back to PCAR=0.3. Next we will make a space-time diagram. This is an image of the road for successive time timesteps, i.e. on the x-axis is the road position and the y-axis is downwards and equal to the number of time steps. You find an example in Fig. 3 of the Chowdhury et al. traffic flow paper. This means that we want a picture similar to the DLA-fractal growth picture we made in class when we worked on the DLA model. You will need to first make a two-dimensional array which stores the space-time diagram data, for example named spacetimearray

Add to your program such an array and set the values of the spacetimearray during your time-loop. To print the array at the end into a file, you may use the following lines:

```
#print spacetimearray into file
fileoutdat=open("spacetime8.data",mode='w')
for i in range(MAXTIMESTEPS):
  for j in range(ROADLENGTH):
     print(spacetimearray[i,j], end=" ",file=fileoutdat)
  print("\n", end=" ",file=fileoutdat)
fileoutdat.close()
```

To test your program compare the resulting spacetime8.data with your results in step 7.

8b. To be able to see the main patterns (and to get nice pictures) use a larger road ROADLENGTH= 200. Now you are ready to make nice picture of this space-time diagram. You find an example for the python syntax at the end of
~kvollmay/share.dir/inclass.dir/classfractal3.py
To get also a colorbar, use before the command plt.savefig the following command:
plt.colorbar(orientation='horizontal')
Look at the resulting space-time diagram and interpret it.

8b. To distinguish stopped cars easily from driving cars, let's indicate every empty site instead of with -1 now with -VMAX. Look at the space-time diagram.

8c. Vary PCAR. Interpret the resulting space-time diagrams. Once you have several space-time diagrams for different PCAR, get me so that we can discuss as a class your results.