- C = -1,0
- C = 0,-1 – the Dendrite
- C = 0.5,0
- C = -0.1 0.8 – the Rabbit
- C = 0.36,0.1 – the Dragon

3.4 Relevant course sections

Studying the following course sections will help you complete this checkpoint:

- NumPy arrays
- Plotting using Matplotlib

3.5 Marking Scheme

- Mandelbrot Set Checkpoint Marking Scheme

4 Traffic

4.1 Aim

This checkpoint requires you to code a simple cellular automaton which attempts to model traffic flow. This is an example where the effective theory is not known, so we invent a theory that has the basic properties that we believe are important. By running simulations we can look for emergent phenomena, in this case the onset of congestion (traffic jams).

4.2 The Model

The simulation box is a straight line of N cells (the road) which can each only have two values: 1 if a car is present on that section of road, 0 otherwise. The update rules for each iteration are very simple:

- If the space in front of a car is empty then it moves forward one cell;
- Otherwise it stays where it is.

If we use $c(j)$ to indicate the state of the jth cell, and use a subscript n to represent the iteration, we can write down rules that determine $c_{n+1}(j)$ from values at the previous iteration $c_n(j-1)$, $c_n(j)$ and $c_n(j+1)$.

School of Physics and Astronomy, The University of Edinburgh. All Rights Reserved.

- if $c_n(j) = 1$
  - if $c_n(j+1) = 1$
    - then $c_{n+1}(j) = 1$
  - else $c_{n+1}(j) = 0$
- else if $c_n(j) = 0$
  - if $c_n(j-1) = 1$
    - then $c_{n+1}(j) = 1$
  - else $c_{n+1}(j) = 0$

4.3 Checkpoint task

Write a PYTHON program to simulate the movement of cars on a one-dimensional road according to the rules defined above. The road has periodic boundary conditions, i.e. moving forward from the last cell of the road takes you back to the first cell.

Starting with a small number of cars placed explicitly on the road, check that your update rules and periodic boundary conditions are working correctly. This can be done using a short road, and printing the positions of the cars to the screen.

When you are confident that the model is working, you should initialise the roadway with a certain (user-defined) density of cars in random positions and then update the road for a certain number of (again, user-defined) iterations.

You should also compute the average speed of the cars at regular intervals. ← WHAT DOES 'AVERAGE

You should find that the average speed changes over the first few timesteps, but once the movement of the cars reaches steady state the average speed remains constant, i.e. is the same at each timestep. The number of timesteps needed to reach equilibrium will depend on the density of cars and their initial positions on the road. You should determine this 'steady state' average speed.

You should now use the results of your code to plot or sketch a graph of the steady state average speed against the density of cars (with density from zero for a completely empty road to one for a completely jammed road). What does this show? ← PLOTTING YOUR RESU

Finally, you should graphically represent the positions of the cars on the road as a function of time. The representation does not need to be animated, though you may use animation if you wish.

4.4 Optional extras

1. Is the model sensitive to the initial configuration of the roadway? You might try to implement very different starting conditions, such as random 'clumps' of cars, rather than simple random cars. Does this affect the onset of congestion on the road?

School of Physics and Astronomy, The University of Edinburgh. All Rights Reserved.

2. The update rules are very simple in this model. Extend your code so that cars move forward:

   - with a probability of 0.75 if they moved last iteration
   - with a probability of 0.5 if they did not

   Does this affect the flow of traffic at a given density ?

3. Bad driving. If you initiate an 'event' on the road, for example an obstacle that moves forward with a very low probability or stays fixed for a few iterations before it disappears, you should be able to see the resulting queue of traffic ripple backwards along the road like a wave and get longer. ← NOTE

   In real life, this 'event' may correspond to someone changing lanes without indicating causing the driver behind to brake suddenly. Quite often, slow-moving or even stationary traffic can form from one of these events.

4.5 Relevant course sections

Studying the following course sections will help you complete this checkpoint:

- Plotting using Matplotlib
- Custom Plotting

Additional material that you may find useful:

- Animation using Matplotlib

4.6 Marking Scheme

- Traffic Checkpoint Marking Scheme

5 Orbital Motion

5.1 Aim

The aim of this checkpoint is to simulate the path of the moon Phobos orbiting the planet Mars using numerical integration.

School of Physics and Astronomy, The University of Edinburgh. All Rights Reserved.