In-Depth MATLAB Example

Modelling Traffic Flow

In this example, you will construct a MATLAB program for modelling traffic flow using the NagelâĂŞSchreckenberg (NaSch) model. When traffic flow along a stretch of road reaches a critical density, due to various human errors in driving, discontinuities in the flow of the traffic occur (analogous in nature to shock waves in fluid flow) and traffic-jams form. The NaSch model is a simple, yet effective model at demonstrating these occurrences, and this is the model you will code in MATLAB.

Coding this is slightly more involved than previous MATLAB exercises.

1 NaSch Model

The model splits the road down into a series of B discrete points/'boxes' and at each of these points, there can either be a car or not. Each box is generally given a discrete number to represent its horizontal location, x. Figure 1 shows this schematically where a car is represented by a red box. Each car also has a 'velocity', v, associated with it which represents how many boxes that car will move during this turn, which is also a discrete number.

Figure 1: Problem schematic

1.1 Algorithm

At each turn, each car advances a number of boxes (i.e. a distance along the road) by considering four stages: 1) acceleration; 2) safety distance; 3) randomisation; 4) moving. Stage 1 represents the desire for a car to drive as fast as possible. Stage 2 simply states that a car cannot overtake another car (we are dealing with a one-dimensional problem i.e. a single carriageway road). Stage 3 models the imperfections in people's driving and the tendency to brake randomly. Finally, stage 4 is the movement process. The equations for each of the stages are:

1. $v_n = \min(v_n + 1, v_{max})$
2. $v_n = \min(v_n, d_n)$
3. $v_n = \max(v_n - 1, 0)$if rand $\leq p$
4. $x_n = x_n + v_n$

where v_n is the velocity of the n-th car, x_n is the location of the n-th car, v_{max} is the maximum velocity (speed limit) any car can have, d is the number of blank boxes between the n-th car and the next car, rand is a random number between 0 and 1, and p is a probability constant.

1.2 Initial Values

Of course, the cars must have an initial location for the simulation and an initial value of velocity. It is recommended to start the cars at uniformly spaced intervals in the boxes (e.g. at $x = 0, 2, 4, 6, \dots$), so if there are N cars that are spaced at c intervals, then the initial location of the n-th car would be:

$$x_n = 1 + (n-1)c \tag{1}$$

For the initial velocity of each car, it is recommended that each car have the same initial velocity that is slightly lower than v_{max} i.e.

$$v_1 = v_2 = \cdots = v_N < v_{max} \tag{2}$$

1.3 Exit Boundary

A number of boundary conditions can be used in the model, however, it is recommended that if a car exits the road (i.e. if $x_n > B$) then this car is removed from the simulation. Hence the code should stop when there are no longer any cars on the road.

1.4 Recommended Values

Table 1 gives recommended values for the constants to try to cause traffic jams.

Table 1: Recommended Values

Variable	Value
B	100
N	30
v_{max}	5
p	0.5
c	3

1.5 Random numbers

In MATLAB, to generate a random number, the `rand` command can be used. However, it is also recommended that you use the `rng('shuffle')` command at the top of your

code to reset the random number generator seed every time.

2 Exercises

1. (Optional) Construct a flow-chart for the algorithm of the NaSch model. This will help consolidate your understanding of the model and how you will need to go about programming it

2. Program the NaSch model and produce a graph of x vs. t (i.e. the location of each car at each iteration). An example is shown in figure 3.

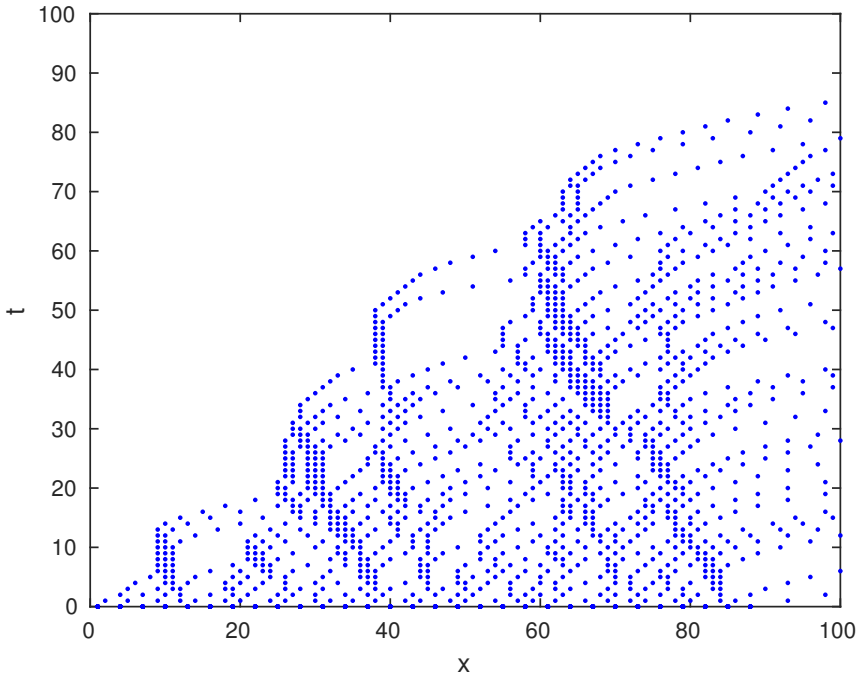

Figure 2: Example graph of x vs. t

3. Expand your code to also produce a graph of x for all the cars that overwrites at each iteration. This will have the effect of updating the location when running and produce a moving image. See the appendix for a script which updates an $x - y$ plot at run-time. Figure **??** shows two stills of the type of graph that results.

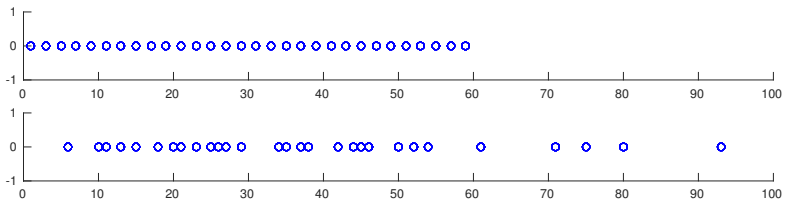

Figure 3: Example graphs of x at different t

Appendix: Updating Plots During Runtime

The following MATLAB script generates an initial set of data points that lie on a straight line and then moves each of them to a random location, where each movement is overwritten on the plot.

```
clear all
close all
rng('shuffle');

%SET-UP FIGURE
figure;
hold on
axis([0 1 0 1]);

%CREATE INITIAL STRAIGHT LINE AND PLOT
for i=1:10
    x(i)=0+(i-1)*0.1; y(i)=0+(i-1)*0.1;
end
h=plot(x,y,'*')
hold on

%MOVE POINTS TO RANDOM LOCATION AND RE-PLOT
for i=1:10
    x(i)=rand; y(i)=rand;
    set(h,'XData',x)
    set(h,'YData',y)
    pause(0.3)
end
```