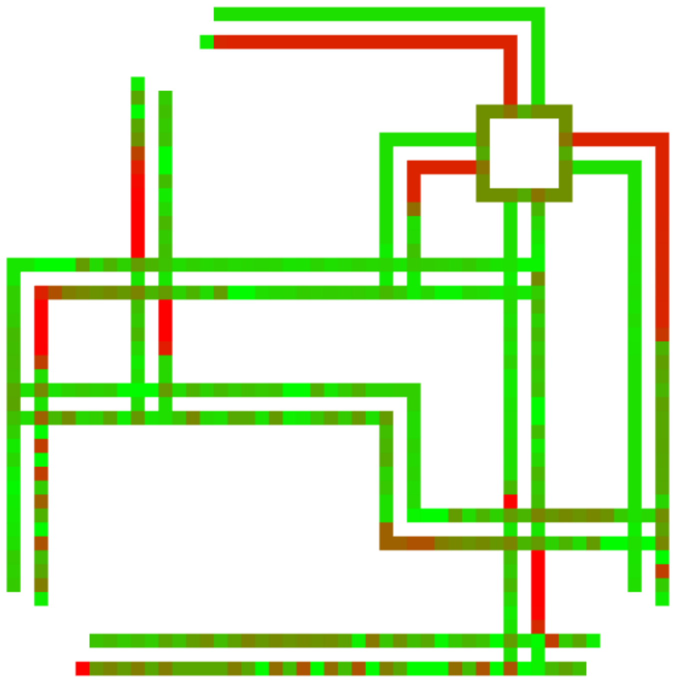
Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Mathematical Sciences

Ludwig Lahmeyer

A Modular Implementation of the Traffic Equation using the Cell Transmission Model

Master's thesis in MSMNFMA
Supervisor: Helge Holden
May 2019



NTNU
Norwegian University of
Science and Technology

Ludwig Lahmeyer

A Modular Implementation of the Traffic Equation using the Cell Transmission Model

Master's thesis in MSMNFMA
Supervisor: Helge Holden
May 2019

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Mathematical Sciences

Abstract

The ability to simulate and modify a road network in a virtual simulation is an invaluable tool to save resources and time in order to identify the best solution to a given problem. The goal of this project was to create a modular implementation of the traffic equation that simulated the traffic flow in a user defined road network. While the project did not produce an implementation with all of the desired functionality, it presents a starting point that can be improved upon in future work. The scripts for the implementation can be found and downloaded from:
`https://github.com/Ardchaius/A-Modular-Implementation-of-the-Traffic-Equation-using-the-Cell-Transmission-Method`.

Contents

Abstract **1**

1 The Cell Transmission Model **4**

2 Choice of Method **4**

3 Brief Overview of the Lighthill-Whitham-Richards Model (LWR) **4**

4 Equivalence to a Hydrodynamic Model **5**

5 Cell Transmission Model and Shocks **6**

6 Implementation **7**
6.1 Two Lane Highway with Overtaking . . . 7
6.2 T-Intersection . . . 8
6.2.1 Intersection with 1 Lane in Both Directions . . . 8
6.2.2 Implementation Challenges . . . 8
6.3 Roundabout . . . 10
6.4 4-Way Traffic Light Crossing . . . 10

7 Improvements to the Initial CTM **13**
7.1 Simplification of the Flow Equation . . . 13
7.2 Backward Waves and Free Flow . . . 13

8 Implementation of Graphical User Interface (GUI) for Modular Network **14**
8.1 General idea . . . 14
8.2 Choice of System Size . . . 15
8.3 Construction Window . . . 15
8.4 Simulation Window . . . 16
8.5 Storage of Data . . . 17
8.6 Saving and Loading of Data . . . 17

9 Instructions on the Use of the Implementation **18**

10 Practical Examples **20**
10.1 Demonstration of Validity Checking Tool . . . 20
10.2 Three-way Intersection . . . 21
10.3 Roundabout . . . 23
10.4 Traffic Light . . . 26
10.5 Editing and, Open and Closed System . . . 29
10.6 Demonstration of Large System, Open and Closed . . . 30

11 Limitations of the CTM **34**
11.1 Velocity . . . 34
11.1.1 Potential Solution . . . 34
11.2 Intersections . . . 34
11.2.1 Potential Solution . . . 35

12 Additional Improvements and Functionality **35**
12.1 The Lagged Cell Transmission Model . . . 35
12.1.1 Advantages . . . 35
12.2 Models with Known Routes . . . 36
12.3 Further Customization of Road Segments . . . 37
12.4 Custom Road Segments . . . 38

12.5 Improved Traffic Light Functionality . . . 39

13 Concluding Remarks **39**

List of Figures

1 1D Highway . . . 4
2 Density-flow (k-q) relationship [2, p. 272] . . . 6
3 2D Highway . . . 7
4 T-Intersection 1 . . . 8
5 T-Intersection 2 . . . 9
6 Intersection with Dual Lanes . . . 9
7 Roundabout . . . 10
8 Generic 4 Way Traffic Crossing . . . 11
9 4 Way Traffic Crossing: Green light for top to bottom . . . 12
10 4 Way Traffic Crossing: Green light for left to right . . . 12
11 Density-flow (k-q) relationship with $w \leq v$ [2, p. 278] . . . 14
12 An example of the Construction Window and how the data is stored for size 3. . . . 16
13 Examples of full matrices that would need to be stored if the whole data matrix for each road segment were to be stored. 0's represent points with no data, 1's represent points with data. . . . 17
14 The start up window as seen by the user upon first running the Start_Up_Window.m script . . . 19
15 System creation window along with the choice of road segments the user is presented with . . . 19
16 The simulation window that the user is presented with upon having created a valid system . . . 19
17 An example of validity checking . . . 21
18 Intersection simulation under light flow . . . 22
19 Simulation of intersection with heavy flow, Stable state after multiple time steps . . . 23
20 Roundabout under light flow . . . 24
21 Stable state under light traffic flow . . . 25
22 Stable state under heavy traffic flow . . . 26
23 Traffic light intersection under light flow part 1 . . . 27
24 Traffic light intersection under light flow part 2 . . . 28
25 Demonstration of system editing, part 1 . . . 29
26 Demonstration of system editing, part 2 . . . 30
27 6x6 road network demonstration, initial conditions . . . 31
28 Large system demonstration, part 1 . . . 32
29 Large system demonstration, part 2 . . . 32
30 Large system demonstration, part 3 . . . 33
31 Demonstration system with all roundabouts after stabilization . . . 33

1 The Cell Transmission Model

The cell transmission model was devised and first described by Carlos F. Daganzo in his 1993 paper "The Cell Transmission Model: A Dynamic Representation of Highway Traffic Consistent With the Hydrodynamic Theory". The model, as described in this paper, takes a single one lane section of highway, and breaks it down into several cells. Then, every clock tick, each cell is updated depending on what the inflow of traffic from the previous cell is and the outflow into the next cell. So for a cell i at time t, the number of vehicles is denoted as $n_i(t)$. If we assume that each cell has a relatively large capacity and the amount of traffic is fairly small, this model would then move all of the vehicles from the previous cell into the next cell upon a clock tick. In general however, the amount of vehicles that move onward to the next cell would depend on the available capacity, the number of vehicles in the previous cell, and the maximum rate at which vehicles can move forward. Generally speaking, the update for time $t + 1$ for cell i can then be denoted as:

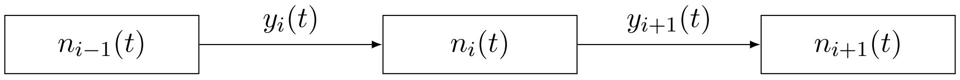
$$n_i(t + 1) = F(n_i(t), n_{i-1}(t), n_{i+1}(t), Q_i, N_i), \tag{1}$$

where Q is the maximum inflow rate of the cell (the maximum number of cars that can in ideal conditions move from one cell to the next), and N is the capacity of the cell. We can rewrite this as:

$$n_i(t + 1) = n_i(t) + y_i(t) - y_{i+1}(t), \tag{2}$$

with y_i defined as the inflow of cars into cell $n_i(t)$, which of course implies that it is also the outflow of cars from cell $n_{i-1}(t)$. To determine the inflow of cars into cell i, $y_i(t)$, we find the minimum over the amount of available space in the cell, the maximal inflow rate and the number of cars present in the previous cell:

$$y_i(t) = \min\{n_{i-1}(t), Q_i, N_i - n_i(t)\}. \tag{3}$$



$$\begin{tikzcd} \boxed{n_{i-1}(t)} \arrow[r, "y_i(t)"] & \boxed{n_i(t)} \arrow[r, "y_{i+1}(t)"] & \boxed{n_{i+1}(t)} \end{tikzcd}$$

Figure 1: 1D Highway

This concept can then be extrapolated to produce models for different types of highways or roads and intersections. Each of the road types that are extrapolated from this basic idea will be detailed in a later section.

2 Choice of Method

The goal of this project is to create a tool set that can be used to simulate large and complex road networks in a modular manner. As such, given that we create a solid basis of initial road segments, a method that allows us to chain together existing road segments is advantageous. Due to this, the method that will be implemented is the Cell Transmission Method (CTM).

Instead of having to solve a set of differential equations for each time step, the CTM is a car-following model. This means that cars will be moved from one cell to the next depending on whether or not there is space available in the next cell. This makes the chaining of several road segments relatively simple, as you simply take the output of the last cell to be the input of the first cell in the next road segment.

3 Brief Overview of the Lighthill-Whitham-Richards Model (LWR)

As it will be necessary to show an equivalence between the CTM and a hydrodynamic model to establish the validity of the model, it would be prudent to give a brief introduction to the LWR with which we will in the next section establish an equivalence.

The governing relation of the model was proposed simultaneously by Lightill-Whitham and Richards in 1955/56, namely that the traffic flow $Q(x,t) = \rho(x,t)V(x,t)$ or speed $V(x,t)$ is always in a local equilibrium with the actual current density. This means that the traffic flow and local velocity immediately change in accordance with changes in density in all situations, not only steady state traffic flows.

$$\nabla \cdot j = -\frac{\partial p}{\partial t}. \tag{4}$$

Plugging the static LWR relation: $Q(x,t) = Q_e(\rho(x,t))$ into the flow conservation equation (4) yields the simplest form of the LWR model:

$$\frac{\partial \rho}{\partial t} + \frac{\partial Q_e(\rho(x,t))}{\partial \rho}\frac{\partial \rho}{\partial x} = 0, \tag{5}$$

which can be rewritten into the more common form:

$$\frac{\partial \rho}{\partial t} + \left(V_e + \rho\frac{\partial V_e}{\partial \rho}\right)\frac{\partial \rho}{\partial x} = 0. \tag{6}$$

The LWR method does not necessarily refer to one specific model, rather a set of models, with the continuity equation being the only dynamic equation in use. Since only one dynamic equation is used in LWR models, they are known as first order models. More complex models, second order models, do exist with the additional dynamic parameter being the local speed considered to be independent quantity.

4 Equivalence to a Hydrodynamic Model

In order to establish the validity of the implemented model, it is necessary to show an equivalence to a hydrodynamic model, such as the LWR in order to prove its validity as a traffic simulation method. We consider a one lane, straight section of road, a homogeneous highway, modeled by a density flow (k-q) relationship in the shape of a trapezoid 2. In terms of an equation, the relationship can be expressed as a function for q:

$$q = \min\{vk, q_{\max}, v(k_j - k)\}. \tag{7}$$

Here, v is the free flow speed, k_j the maximum density, and q_{max} is the maximum flow, bounded by $q_{\max} \leq \frac{k_j v}{2}$.

To now show the equivalence to the LWR model, we place equation (7) into the flow conservation equation (4), giving us:

$$\frac{\partial q(x,t)}{\partial x} = -\frac{\partial k(x,t)}{\partial t}, \tag{8}$$

the differential equation, that would determine the propagation of the system under a hydrodynamic model like the LWR.

Now all the remains is to show that (8) is equivalent to the Cell Transmission Model introduced earlier. As the highway is homogeneous, it allows us to make a few simplifications to (3). Namely, the homogeneity of the highway implies that the characteristics N and Q are independent of both i and t, meaning we simplify as follows: $N_i(t) \to N, Q_i(t) \to Q$. As we are attempting to show an equivalence between a discrete and continuous method, we choose to define one clock tick as being equal to dt, and a unit of distance that gives $vdt = 1$. This allows for a few more simplifications: the length of a cell is 1, $v = 1$, $x = i$, $k_j = N$, $q_{\max} = Q$, and finally $k(x,t) = n_i(t)$. With the appropriate substitutions, (8) can then be rewritten as:

$$\frac{\partial \min\{n_i(t), Q_i, N_i - n_i(t)\}}{\partial x} = -\frac{\partial n_i(t)}{\partial t}. \tag{9}$$

Which is identical to our definition of $y_{i+1}(t)$, apart from the fact that the second n_i should be n_{i+1}. This however is of no import, as the hydrodynamic differential equations are only meaningful when the density is differentiable in x. This means that we can once again re-write our equation above into the following form:

$$y_{i+1}(t) - y_i(t) = -(n_i(t+1) - n_i(t)) \tag{10}$$
$$\rightarrow n_i(t+1) = n_i(t) - y_{i+1}(t) + y_i(t), \tag{11}$$

proving the equivalence.

5 Cell Transmission Model and Shocks

The propagation of shocks or disturbances is an important aspect of any traffic simulation. In the event of a car crash on a highway one can easily see in real life that the ensuing car jam does not resolve itself as soon as the crash is cleared. The blockade of traffic that builds up until the road is reopened propagates backwards meaning the traffic jam still exists , but has moved further upstream even after the roads have been cleared. Ensuring that a chosen method simulates such a situation correctly is vital to ensure an accurate simulation.

Let us assume a simple case, a section of highway where the density varies within a narrow range as we move downstream, meaning $k(x, 0)$ varies with x, the distance downstream. We assume though that the density falls within 3 different possible categories:

1. $0 \leq k(x, 0) \leq k_A$
2. $k_A \leq k(x, 0) \leq k_B$
3. $k_B \leq k(x, 0) \leq k_j$

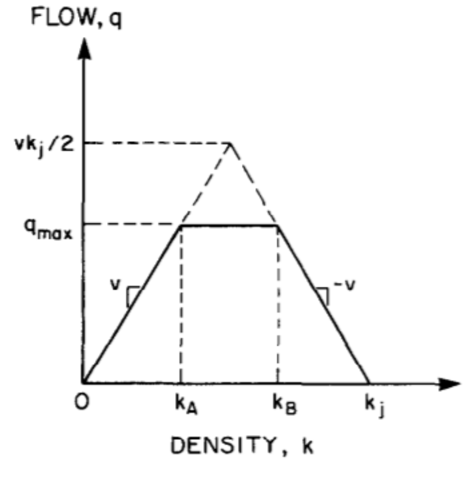


Figure 2: Density-flow (k-q) relationship [2, p. 272]

In such a situation, the hydrodynamic model tells us that the initial density curve we have will be preserved throughout time, the only difference being its position in space, i.e. it is shifted as time progresses, and no shocks are produced. In other words: $k(x, t) = k(x - w_k t, 0)$, with w_k being the wave speed of the section of our model containing our initial densities. With our chosen velocity of $v = 1$, this gives $w_k = 1$, $w_k = 0$, $w_k = -1$ for each of our three scenarios respectively.

It is fairly simple to see that the CTM also satisfies the propagation $k(x, t) = k(x - w_k t, 0)$. The number of vehicles that is transmitted from cell to cell is determined by equation (3), which tells us that the number of vehicles moving from cell $i - 1$ to i is either n_{i-1}, Q_i or $N_i - n_i$ according to the density k. Combining this with (2), we can quickly see that the $n_i(t + 1)$ is either $n_{i-1}(t)$, $n_i(t)$ or

$n_it + 1$, depending on which of the three categories the density is currently in. This indicates that the density profile propagates either upstream or downstream at a speed consistent with the wave speed.

This is however the best case scenario, where the densities lie within a narrow range. The situation becomes much more complicated if the density profile is not completely in this range, resulting in shocks being created. An in-depth explanation of the possible situations, increasing and decreasing density in the direction of travel can be found in [2, p. 274-276].

6 Implementation

The CTM method was implemented in Matlab, and the homogeneous highway section with a single lane was just a matter of implementing the initial equations 2 & 3. However, modifications had to be made for more complex road types, for example a two lane highway with the option of overtaking in case of bottlenecks. These situations, and modifications made to the method are described bellow. It should be noted very shortly that turns, purely left or right, will not have a dedicated section, as they can be viewed as simple straight road sections, and do not require any changes to the original formulas.

6.1 Two Lane Highway with Overtaking

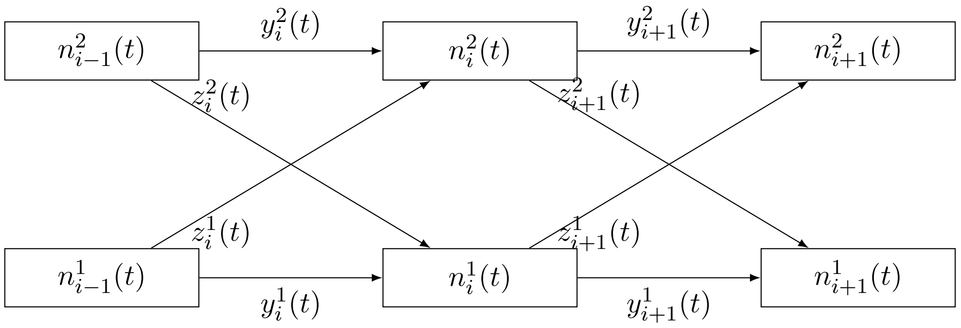


Figure 3: 2D Highway

On this highway the cars have to stay in the bottom lane, only moving into the top lane in order to overtake, and having to re-merge with the bottom one as soon as space allows. The first step is to identify the order in which flows should be calculated. The first step would be to calculate the forward flow in the bottom lane, as this is the primary lane, and cars will stay in it unless prohibited due to a road block for example. Next the flow from the bottom to the top lane should be calculated, this would be defined as any remaining cars that were not able to move ahead into the next cell in the bottom lane:

$$z^1_i(t) = \min\{n^1_{i-1}(t) - y^1_i(t), Q, N^2_i - n^2_i(t)\}. \tag{12}$$

Step three is calculating how many of the cars move from the top lane back into the bottom lane, ensuring that the 2 lanes merge again. This would mean calculating how many cars can re-merge, after the cars in the bottom lane have moved on to the next cell.

$$z^2_i(t) = \min\{n^2_{i-1}(t), Q, N^1_i - n^1_i(t) - y^1_i(t)\}. \tag{13}$$

And last but not least, the final step will be to calculate how many cars that could not re-merge with the bottom lane will continue moving forwards in the top lane:

$$y^2_i(t) = \min\{n^2_{i-1}(t) - z^2_i(t), Q, N^2_i - n^2_i(t) - z^1_i(t)\}. \tag{14}$$

$$\begin{tikzcd}
 & m_{i-1}(t) \arrow[d, "z_i(t)"] & \\
 & m_i(t) \arrow[d, "z_{i+1}(t)"] & \\
n_{i-1}(t) \arrow[r, "y_i(t)"] & n_i(t) \arrow[r, "y_{i+1}(t)"] & n_{i+1}(t)
\end{tikzcd}$$

Figure 4: T-Intersection 1

6.2 T-Intersection

Another important case to implement is the case of right-of-way. In certain situations vehicles will have to wait until there is a break in traffic before making a turn for example. This is the first and simplest of such cases that we will tackle. The easiest way to implement this would be to treat each of the two road sections independently, the road going left to right and the road going top to bottom, but have them share the cell where the intersection occurs.

For this scenario let's assume that the road going from left to right has right-of-way, meaning the vehicles coming from the top have to wait for a break in traffic before being able to join the flow going left to right. Firstly we calculate the flows $y_i(t)$ & $z_{i+1}(t)$, and if $y_i(t) \neq 0$, we know that vehicles are moving from $n_{i-1}(t)$ to $n_i(t)$ and there will not be a break in traffic for the vehicles in $m_i(t)$, so we set $z_{i+1}(t) = 0$. If $y_i(t) = 0$ however, we know there will be a break in the traffic for the vehicles to use, and leave $z_{i+1}(t)$ as calculated. The only other modification that needs to be made is to the calculation of $n_i(t+1)$:

$$n_i(t+1) = n_i(t) + y_i(t) - y_{i+1}(t) + z_{i+1}(t) \tag{15}$$

This same implementation can then be extended one step further to cars crossing over multiple lanes of traffic going in different directions, as seen in Figure 5.

The only change that needs to be made is that we now check that either one or both $y_i^1(t), y_i^2(t) \neq 0$ before setting $z_{i+1}(t) = 0$.

6.2.1 Intersection with 1 Lane in Both Directions

Once more, the change to a situation with a single lane in both directions, is rather simple to achieve in theory, but requires a few more steps in the implementation. It is once more assumed that the lanes going to the left and right in Figure 6 are the main road, and as such have priority, while the lane coming from the top and the one going to the top, have to wait for a suitable break in traffic before continuing.

6.2.2 Implementation Challenges

While there is not a large increase in difficulty from Figure 5 to Figure 6, it did however prove to introduce some challenges. Not in terms of the idea of the implementation itself, but in keeping track of the indexing, and ensuring vehicles are moved forwards to the correct cell in the next time step. If this is implemented incorrectly, it can lead to phantom vehicles being introduced to a closed system. This can occur if vehicles are added to two different cells if the indexing is carefully implemented. The

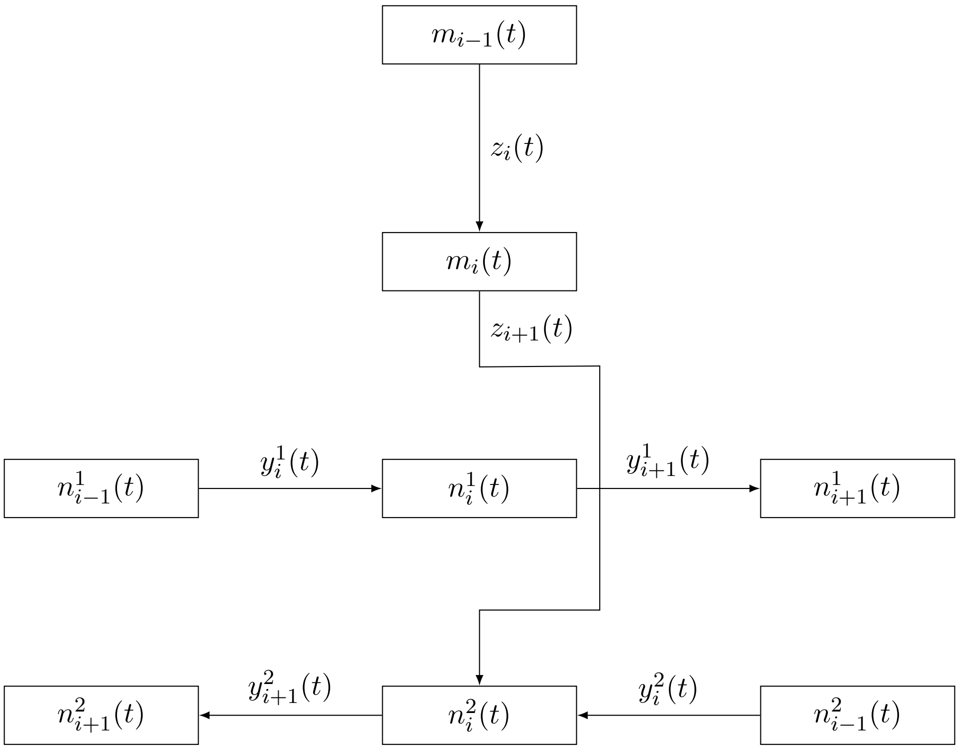


Figure 5: T-Intersection 2

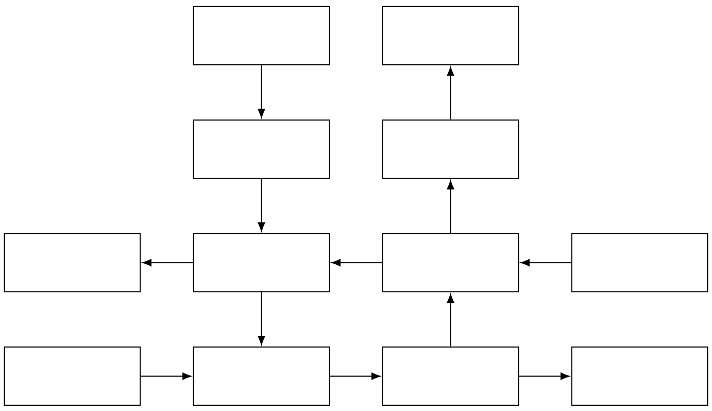


Figure 6: Intersection with Dual Lanes

reason that this issue was not present in the straight road sections is due to the fact that each cell only has one possible destination.

However with intersections, and with the upcoming roundabouts and traffic lights, some cells split the vehicle stream, providing 2 (or for other segments potentially more) possible destinations to some cells. This means the implementation has to keep very careful track of the comings and goings of each cell, as one simple error can produce a system not in equilibrium, producing and deleting vehicles for no intended reason.

The way the method was implemented, the cell where the split occurs is divided into to separate cells, the movement of vehicles is performed, and finally, the two halves are merged together again. If the re-merging of the cells is performed incorrectly, this is already enough to introduce such a phantom vehicle error, as was the case in the first implementation of an intersection of the form in Figure 6.

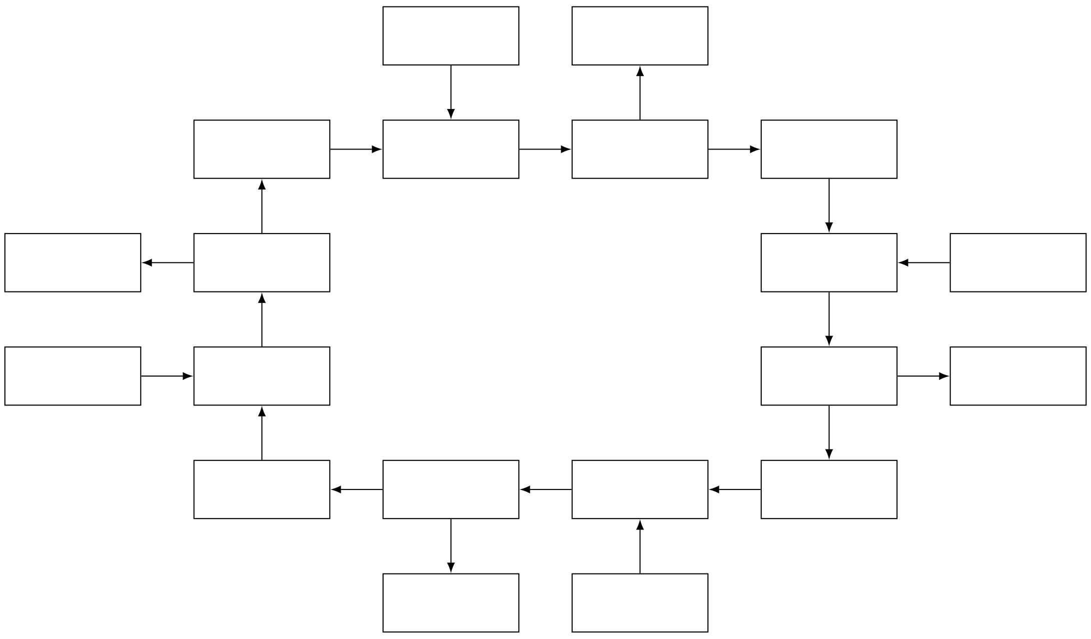


Figure 7: Roundabout

6.3 Roundabout

Rather than directly implementing the roundabout, which could be done by possibly unraveling it, and treating it as a long section of road where the last and first cells are the same, we treat this as 2 smaller problems. Firstly we implement just one side of the roundabout, the top section. This is a relatively simple extension of the T intersection with an additional exit. Once this has been implemented, it is a simple matter of placing four of these in succession with the last cell looping back around to the first one.

This is were the modularity of the CTM comes in particularly handy, as this placing of 4 sections after each other is almost trivial, and even the loop is easily implemented. This is not something to take for granted, as a cursory glance of other literature seems to indicate that loops with differential methods can be somewhat tricky.

For the off ramps to this first roundabout implementation, it is assumed that the routes of individual vehicles is not known, and we are working with turning percentage instead. This means that the vehicles in the cell by the off ramp have to possible choices, they can either continue in the roundabout or take the off ramp out of the roundabout. This choice is represented as a turning percentage, namely the percentage of cars that will take the off ramp instead of continuing to drive in the roundabout. A simple example in this case would be that each off ramp has a turning percentage of 25% for example, meaning that the exiting cars are evenly split between all the off ramps.

While this method does provide a decent model, it is of course much more desirable to obtain a model that is able to assign or track individual vehicle destinations, assigning turns based upon their desired route. Such an implementation/ variation of the CTM is introduced and discussed in a latter section.

6.4 4-Way Traffic Light Crossing

As a counterpoint to the roundabout implementation, one other road segment that merits implementation is the 4-way traffic light crossing. This is due to the fact that varying which of these two options is used in certain situations, different and possibly more satisfying results can be achieved. From a cursory glance, this would appear to be a very similar implementation to the intersection implementation with an extra road coming in from a fourth side, however the addition of traffic lights, makes this implementation slightly more complex than the intersection case.

Firstly a decision has to be made in regards to switching the traffic lights on whether to keep the

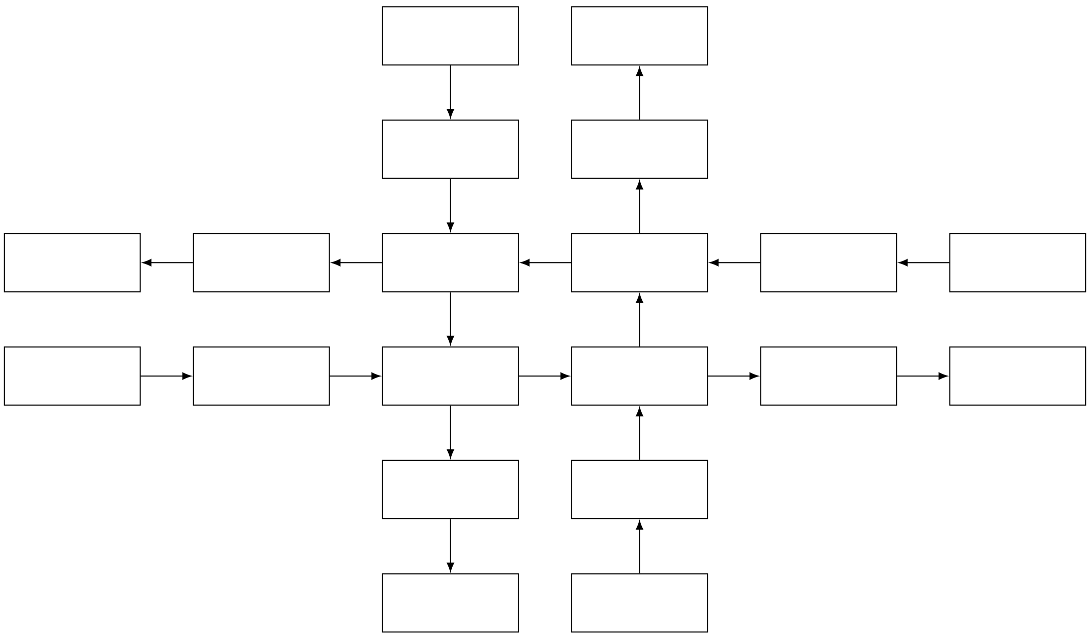


Figure 8: Generic 4 Way Traffic Crossing

yellow light or just use red and green. For this implementation it was decided to make use of just the red and green lights, with the option of adding a time interval between them to give the crossing time to clear. Secondly a choice needs to be made as to how to implement the two different situations, when the light is green for the top to bottom vehicles and when the light is green for the left to right vehicles.

There are two choices here, either implement each method separately, or implement just one of them and use that implementation with modified input for the second option. This implementation uses the latter approach, only coding one of the two situations, but with a switch to modify the input for the second situation, the pseudo code and diagrams of the two situations can be seen below.

```
if (0 < timer) && (timer < light_Switch + light_Pause + 1) % Top—Bottom light is green
    % Set values for Top—Bottom case: x
    if (light_Switch < timer) % Both lights are red, intersection clearing
        % Cut off input from top and bottom
    end
    % Calculate_next_time_step(x)
    timer = timer + 1;
elseif (light_Switch + light_Pause < timer) &&...
        (timer < 2 * (light_Switch + light_Pause) + 1) % Left—Right light is green
    % Set values for Left—Right case: x
    if (2 * light_Switch + light_Pause < timer) % Both lights are red, intersection clearing
        % Cut off input from left and right
    end
    % Calculate_next_time_step(x)
    timer = timer + 1;
    if timer > 2 * (light_Switch + light_Pause)
        timer = 1;
    end
end
```

Here, the value timer is used to switch the lights from red to green and vice versa at certain intervals set by light_Switch and light_Pause, that respectively indicate the time a light stays green and the clearing time between green lights. It is readily apparent from comparing the 2 possible situations in Figures 10 and 9, that we can achieve the situation in Figure 10 by rotating Figure 9 clockwise 90 degrees. Hence it is only necessary to implement a function to calculate the next time step for figure 9, and then modify the input to reflect the situation in 10 when it arises. This can be

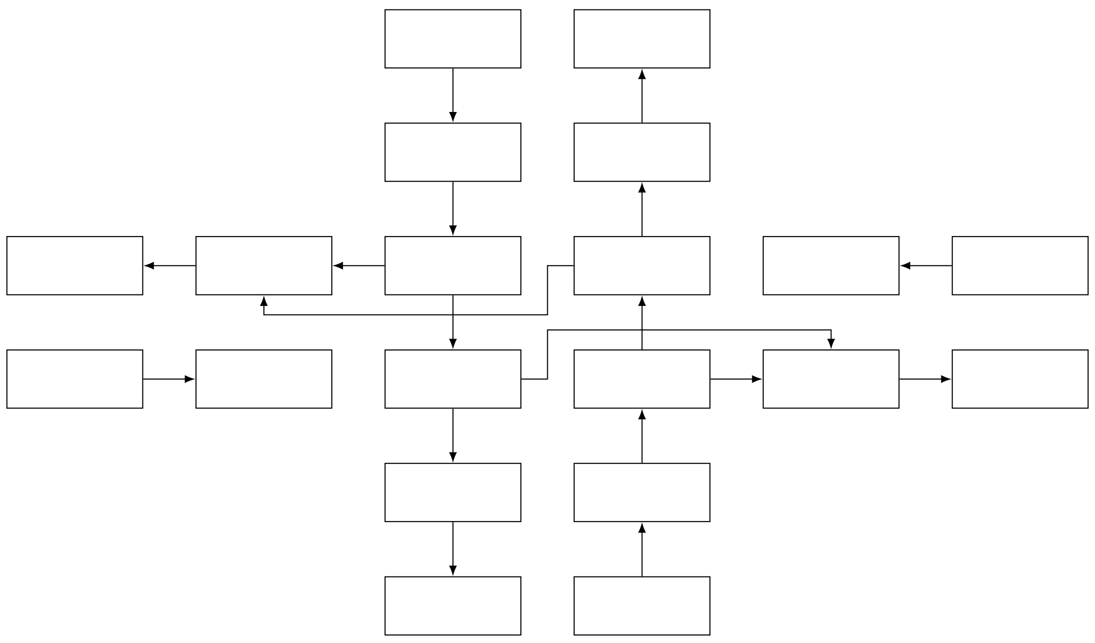


Figure 9: 4 Way Traffic Crossing: Green light for top to bottom

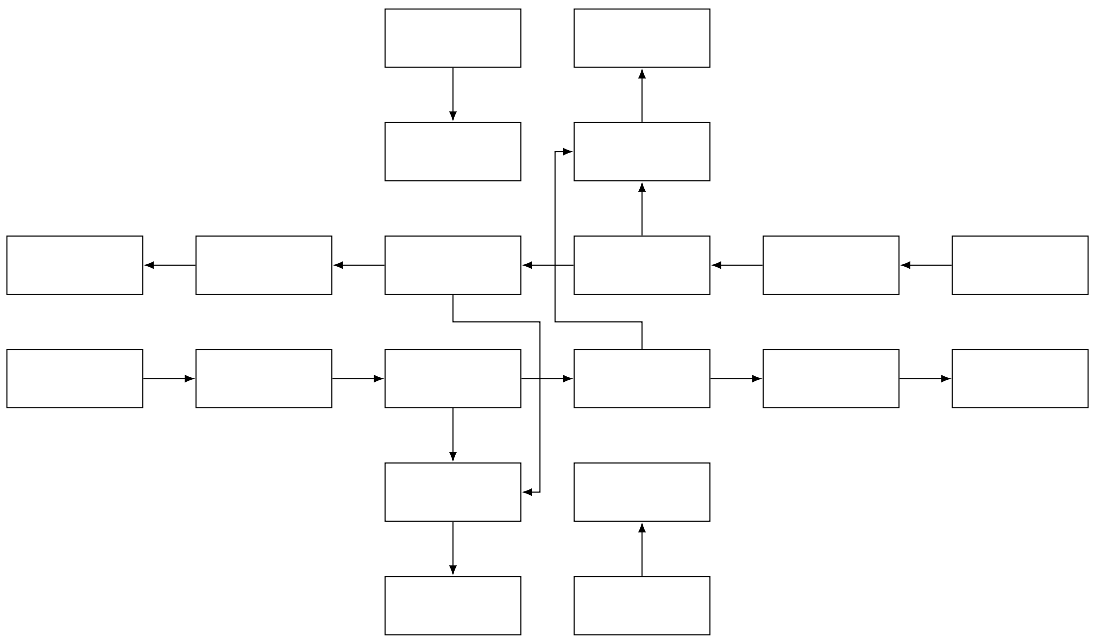


Figure 10: 4 Way Traffic Crossing: Green light for left to right

seen in the pseudo code in lines 2 and 10. Line 2 sets the variables for the case in 9, while line 10 sets the variables for the case in 10, before using the same function, in lines 6 and 14, to calculate the next time step for both cases.

Concerning the case where vehicles cross lanes, it is assumed that they cross the lane and enter the next cell within one time frame. However in order to ensure that vehicles do not cross through a lane that is at capacity, the formula in the implementation takes into account the flow into the next cell, as well as the flow into the cell that is being crossed. This ensures that if vehicles are aiming to drive across lanes to continue on their journey, they will be unable to do so if the lane they are crossing is completely full.

7 Improvements to the Initial CTM

While the initial CTM that has been described so far, seems to provide a generally good estimate for traffic simulation, it can be improved upon of course. The next section delves into known improvements of the method and their effects upon the simulation.

7.1 Simplification of the Flow Equation

The equation that has so far been introduced and used for the calculation of the flow from cell $i-1$ to i is somewhat messy and can be simplified and cleaned somewhat, as demonstrated by Daganzo in his follow up paper in 1995[3]. A few changes are introduced here in order to make the function (3) somewhat easier to work with.

So far the value Q_i has been used to denote the maximum number of vehicles that can enter into cell i each clock tick given adequate room. At this point we will give a slightly adjusted definition, Q_i will be the maximum flow possible *through* (not into) cell i, which can be used to indicate cell width for example, and the maximum flow from cell $i-1$ to i will be defined as the minimum of Q_{i-1} and Q_i. So from this point forward each cell i will have two characteristics: maximum cell occupancy N_i and maximum flow through the cell Q_i.

Due to the change in notation definition, we will very quickly restate the figure of the simplest possible situation in order to avoid confusion:

$$\begin{tikzcd}\boxed{\begin{array}{c} n_{i-1}(t) : \\ \{Q_{i-1}; N_{i-1}\} \end{array}} \arrow[r, "y_i(t)"] & \boxed{\begin{array}{c} n_i(t) : \\ \{Q_i; N_i\} \end{array}}\end{tikzcd}$$

Given our re-definition of Q_i, our the flow function can now be written as:

$$y_i(t) = \min\{n_{i-1}(t), \min\{Q_{i-1}, Q_i\}, N_i - n_i(t)\}, \tag{16}$$

and further simplified by defining:

$$S_i(t) = \min\{Q_i, n_i\} \tag{17}$$
$$R_i(t) = \min\{Q_i, N_i - n_i\}, \tag{18}$$

giving us our much compacted version:

$$y_i(t) = \min\{S_{i-1}(t), R_i(t)\}. \tag{19}$$

S and R were chosen as the function $S_i(t)$ indicates how many vehicles are sent by cell i at time t, and $R_i(t)$ indicates the number of vehicles cell i is able to receive at time t.

7.2 Backward Waves and Free Flow

One of the most obvious restrictions that the initial CMT imposes the condition on the backward waves, by forcing the backward wave speed to be equal to the speed of free flow. This however is far from realistic, as the backward propagation speed of the waves is multiple times slower than the speed of free flow, which means we have unrealistic behaviour simulation when it comes to for example bottlenecks, traffic jam's etc., basically any situation that creates a backward wave. For example, the traffic jam caused by the bottleneck at a roundabout would, with slower backward waves, be dissipated far further upstream than with the initial CTM described thus far.

So we would have to introduce changes that would allow the CTM to approximate the hydrodynamic model that permits waves speed that satisfy $w \leq v$. The hydrodynamic model that satisfies this condition is the following relationship between density and flow $(k-q)$, depicted in pictorial form in Figure 11:

$$q = \min\{vk, q_{\max}, w(k_j - k)\}. \tag{20}$$

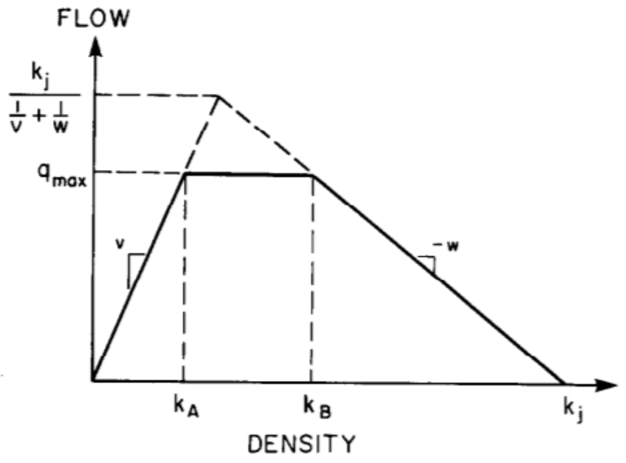


Figure 11: Density-flow (k-q) relationship with $w \leq v$ [2, p. 278]

To ensure that the CTM models this hydrodynamic model correctly, only one small change has to be made to the flow calculation function (3):

$$y_i(t) = \min\left\{n_{i-1}(t), \min\{Q_{i-1}, Q_i\}, \frac{w}{v}[N_i - n_i(t)]\right\}. \tag{21}$$

Or in its simplified form:

$$y_i(t) = \min\{S_i(t), R_i(t)\} \tag{22}$$
$$S_i(t) = \min\{Q_i, n_i\} \tag{23}$$
$$R_i(t) = \min\left\{Q_i, \frac{w}{v}[N_i - n_i]\right\}. \tag{24}$$

However, due to the change in the flow calculation function it is necessary to shortly re-check the statement made earlier regarding the equivalency between the hydrodynamic model and the CTM. This is however trivial, as we simply replace the function for q in equation (9), and repeat the same arguments that were previously made.

8 Implementation of Graphical User Interface (GUI) for Modular Network

As previously stated, one of the goals of this project is to create a GUI that a user will be able to employ to create their own road networks and simulate traffic within these networks. The use of the CTM method simplifies this immensely, and the following sections describe how such a program was implemented.

8.1 General idea

There are several aspects that a program as described above should contain:

- An option for the user to choose the size of the system
- Ability for the user to choose the construction of the network freely
- Allow the user to then advance the simulation one singular step at a time, or to play out automatically
- The ability to modify their network, and to reset their network to the initial conditions

- Being able to choose between a steady stream of vehicles, or a random inflow (Ideally also allow them to choose exact input, but much more difficult to do so)
- Give users as much control as possible over their system to enable them to freely simulate whichever situation they wish.

These points outline the general concept of the GUI, and what was aimed at. The following sections will give a description and a brief outline of how this was achieved.

8.2 Choice of System Size

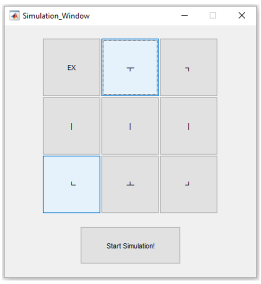
The most obvious starting point would be the starting point of the user, the choice of the size of the system. This was a fairly simple step and as such not much time will be devoted to its explanation. Simply said, the user is greeted with a text field where they may enter a number between 1 and 10 to indicate the square size of the system they wish to simulate. Once this has been entered they click the button below which checks that their input is a valid choice before opening up the next window, the construction window where users can construct the system to their specifications.

8.3 Construction Window

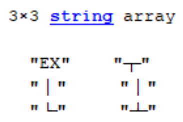
The next step is to create a window that enables the users construction of their system, that adapts to the chosen system size. The construction window will consist of a square of buttons, the same size as the chosen system size, and will enable the to user to choose what road piece they would like to place in each of the individual segments. This was slightly more complicated than initially thought, as the buttons would of course be in different locations and different sizes for each system size. This was eventually resolved with the following code-snippet:

```
        simSize = varargin{1}.Size; % Collect the size
        dist = 0.7/(simSize); % Variables for the arrangment of the boxes
        boxSize = 0.7/(simSize);

        road_Network = strings(simSize); % Create an empty array to store network for calculation
        road_Network(:) = '0';

        handles.SimSize = simSize; % Store size globally
        handles.Network = road_Network; % Store network globally
        handles.Connections = cell(simSize);
        connections = handles.Connections;

        for i = 1:simSize
            for j = 1:simSize
                tag = sprintf('pushbutton%i',i+(simSize-j)*simSize); % Generate tag for button
                uicontrol('Parent',hObject,'Style','pushbutton','String','0','Units','normalized',..
                'Position',[0.15+dist*(i-1) 0.25+dist*(j-1) boxSize boxSize],'Visible','on',...
                'Callback',{@selection_screen,hObject},'Tag', tag); % Create pushbutton
                connections(i,j) = {[0,0,0,0]};
            end
        end

        handles.Connections = connections;

        uicontrol('Parent',hObject,'Style','pushbutton','String','Start Simulation!',...
        'Units','normalized','Position',[0.3 0.05 0.4 0.15],'Visible','on',...
        'Callback',{@start_Simulation,hObject},'Tag', 'Start'); % Create start simulation button
```

This generalized piece of code takes the available space of the construction window and systematically places the required buttons depending on the size of the system chosen. It has been written in such a way that if necessary, the size of the window may be changed in the GUI editor in Matlab without requiring any re-writing of the code. Essentially this means that we could eventually create systems larger than the current size 10 restriction that we have placed upon it. Additionally this code



(a) State of the Construction Window



```
3×3 string array

  "EX"    "┬"    "┐ "
  " | "   " | "   " | "
  " └"    "┴"    "┘ "
```

(b) State of the corresponding matrix to be passed to the simulation window, Road_Network

Figure 12: An example of the Construction Window and how the data is stored for size 3.

also assigns each button a function that presents the user with a choice of road pieces when the button is clicked. Depending on the users choice the program then visualizes their choice, and constructs a matrix with the given information that is later passed on to the simulation program. An example of what this construction window and accompanying matrix might look like can be found below in Figure 12:

Once the user is satisfied with their layout, they may click the ”Start Simulation!” button. Before launching the simulation window however, the program will check for inconsistencies, ensuring that all pieces have a valid connection to their neighbouring pieces before starting the simulation.

8.4 Simulation Window

The simulation window is the brains of the program, taking the created road network and simulating traffic flow within it. It does this by updating each road segment at a time, before updating the results in the plot on the left hand side of the screen. This process is simplified and demonstrated in the following pseudo code for the case seen in Figure 12.

```
for i = 1:3
    for j = 1:3
        Identify what type of road segment we are working with;
        Identify the input and output capacities of neighbouring cells;
        Update Road_Network(i,j);
    end
end
Visualize Road_Network;
```

As already mentioned, this is a very simplified version of the pseudo code. The actual implementation has a few additional steps relating to the structure of how the data is saved. Specifically this relates to saving storage space for larger systems. Each segment of the road network is of a predetermined size, with the default size being 9, that is adjustable to an odd value greater than or equal to 7. This means that each segment of the road network is represented by a size 9 square matrix, examples of which are pictured in Figure 13. It is however a waste of storage space to store the entirety of the matrix when a majority of it does not contain any data.

$$\begin{bmatrix} 0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0 \\ 1&1&1&1&1&1&1&1&1 \\ 0&0&0&0&0&0&0&0&0 \\ 1&1&1&1&1&1&1&1&1 \\ 0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0 \end{bmatrix}$$

(a) Straight road from left to right

$$\begin{bmatrix} 0&0&0&1&0&1&0&0&0 \\ 0&0&0&1&0&1&0&0&0 \\ 0&0&0&1&0&1&0&0&0 \\ 1&1&1&1&1&1&1&1&1 \\ 0&0&0&1&0&1&0&0&0 \\ 1&1&1&1&1&1&1&1&1 \\ 0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0 \end{bmatrix}$$

(b) Intersection with road from the top

$$\begin{bmatrix} 0&0&0&1&0&1&0&0&0 \\ 0&1&1&1&1&1&1&1&0 \\ 0&1&0&0&0&0&0&1&0 \\ 1&1&0&0&0&0&0&1&1 \\ 0&1&0&0&0&0&0&1&0 \\ 1&1&0&0&0&0&0&1&1 \\ 0&1&0&0&0&0&0&1&0 \\ 0&1&1&1&1&1&1&1&0 \\ 0&0&0&1&0&1&0&0&0 \end{bmatrix}$$

(c) Roundabout

Figure 13: Examples of full matrices that would need to be stored if the whole data matrix for each road segment were to be stored. 0's represent points with no data, 1's represent points with data.

The first considered solution would of course be to use sparse matrices, however during early testing of the GUI interface, the sparse matrix implementation produced some errors, preventing us from using this solution. Instead, only the data points that contain useful data will be stored, however some additional steps are needed to format the data for processing in the function for the calculation of the next time step, and the visualization. For the case of the straight segment, this is straightforward, but when working with the 4 way intersection with traffic light or the roundabout, it becomes slightly more complicated. Adjusting our pseudo code we are then left with the following.

```
for i = 1:3
    for j = 1:3
        Identify what type of road segment we are working with;
        Identify the input and output capacities of neighbouring cells;
        Format data in Road_Network(i,j);
        Update Road_Network(i,j);
    end
end
Format data for visualization;
Visualize Road_Network;
```

This is an extremely bare-bones pseudo code of the actual implementation, but in essence these are the steps the simulation window performs every time the user presses the Single Time Step button. The other buttons are fairly self explanatory: the Start Auto-Play with continuously execute these steps until the button has been pressed a second time; the Edit Network button allows the user to make changes to their network, without loss of information in any unchanged segments; and lastly the Reset Network button will roll the network back to its initial conditions.

8.5 Storage of Data

An important part of the implementation of the method is to find an efficient storage mechanism for the relevant data, and generalizing as much as possible to simplify the code. As mentioned earlier, the greatest save in storage space comes from the fact the segments are not stored in their entirety, but only the data points that contain any actual data. So for each segment we end up storing at most $\mathcal{O}(4n)$ instead of $\mathcal{O}(n^2)$ data points, a fairly significant storage save for larger networks.

One section of the code were the structure of this saved data is important, is the formatting of the data into a usable format for the functions that calculate the next time step.

8.6 Saving and Loading of Data

With the data having now been formatted, one extremely useful attribute would be the ability to save and load previously saved networks in order to have the opportunity or larger networks to pause the simulation if the computing power is needed elsewhere temporarily for example. As such, a rudimentary save and load function was implemented, the implementation of which proved somewhat more challenging than expected due to some of the symbols used in the implementation and an ordinary .txt document not supporting all of the symbols.

For this save and load function, the order in which the data is saved plays an important role. Depending on the size of the system and the individual cells, the matrices storing the cell data for the segments will be of differing sizes. As such, the first thing to be saved and hence loaded, will have to be the size of the system and the size of the segments. After that the information of the segments is then stored in a line by line basis, meanign that the matrices are vectorized and each occupy 1 line of the .txt. document that acts as the save file.

Specifically the process works thusly:

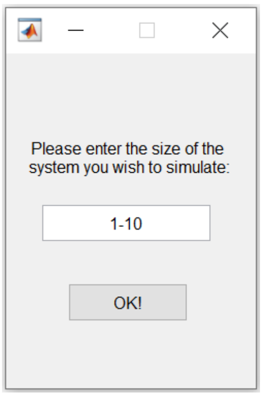
```
Save size of simulation
Save size of Segments
Save input volume/rate
for i = 1:size of simulation
    for j = 1:size of simulation

        for segment in position (i,j) do:
            Save type of segment i.e. left—right, right—top, roundabout, etc.
            Save data from cells as vector
            Save in out data matrices
            Save connections matrix

    end
end
```

The loading process works in the exact same order, and overwrites the information of the previous simulation unless saved beforehand. Initially, the save and load function was not intended to be a part of the project, however, due to this being a practical project, example files are provided. Instead of instructing the reader on how to construct each of the intended examples, now it is a simple matter of starting the program and loading the examples at the appropriate time. Obviously this functionality can be improved through the addition of an autosave feature that would automatically backup the simulation at regular intervals. This however is a trivial extension of this function as the save functionality is already in place, and would require minimal additional coding if desired.

9 Instructions on the Use of the Implementation

Due to the nature of this project, most of the time was spent on the writing, improvement and error checking of the implementation, and all of the code can be found online in a Github code repository: https://github.com/Ardchaius/A-Modular-Implementation-of-the-Traffic-Equation-using-the-Cell-Transmission-Method. The implementation was written in Matlab, so a Matlab license or access to Matlab is required in order to be able to play around with or modify the implementation, and in order to view the examples detailed in a latter section.

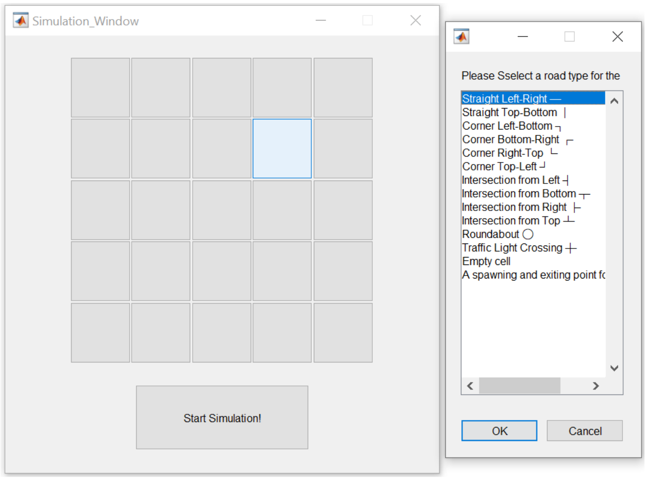
In order to use the implementation, the files need to be downloaded and placed into a single folder. The only script that needs to be run in Matlab is the script Start_Up_Window.m, this script will initiate the program and present the user with their first choice, the size of the system as seen in Figure 14.


Please enter the size of the system you wish to simulate:

1-10

Figure 14: The start up window as seen by the user upon first running the Start_Up_Window.m script

Once the size of the system has been chosen and the user has clicked "OK!", the new window presents the user with a grid of buttons that will enable them to create their system. Upon clicking any of these buttons they are presented with a choice of road segments that they can place in the clicked position, Figure 15.



Please Sselect a road type for the

Straight Left-Right —
Straight Top-Bottom |
Corner Left-Bottom ┐
Corner Bottom-Right ┌
Corner Right-Top └
Corner Top-Left ┘
Intersection from Left ┤
Intersection from Bottom ┬
Intersection from Right ├
Intersection from Top ┴
Roundabout ○
Traffic Light Crossing ┼
Empty cell
A spawning and exiting point fc

Figure 15: System creation window along with the choice of road segments the user is presented with

Upon completing their system and pushing the "Start Simulation!" button, they are either asked to make changes to their system if it is invalid, or will be taken to the simulation window itself that will provide a visual of their system during the simulation, Figure 16.

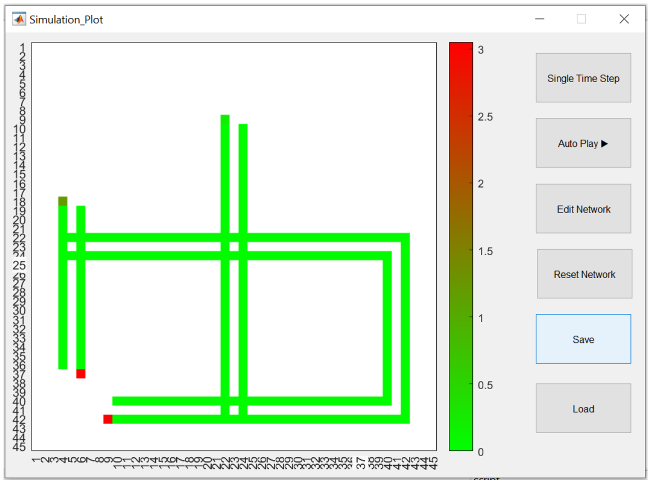


Figure 16: The simulation window that the user is presented with upon having created a valid system

This window also presents the user with a few different options during their simulation, in the form of the labelled buttons on the right hand side, all of which are relatively self explanatory:

- "Single Time Step": Performs a single step of the cell transmission method.
- "Auto Play": Will continuously perform steps of the cell transmission method until it is pressed again. During this time, all other buttons will be disabled.
- "Edit Network": Provides the user with an interface identical to that in 15 for editing their system.
- "Reset Network": Allows the user to completely reset their network to an empty state.
- "Save": Opens a window to let the user save the document as a .txt file with a name of their choosing.
- "Load": Allows the loading of previously saved systems.

In addition to these functionalities, there are several other options open to the user if they are willing to modify the code itself. When downloaded, the program comes with some values pre-set, for example the input flow for entrance and exit pieces is set to 'random', meaning that a random amount of vehicles between 0 and 3 will attempt to enter the system from each entrance piece every time step. If a constant flow of vehicles is desired, this can be achieved by modifying the InputVolume value in the Simulation_Plot.m script. Additionally, there are several values associated with the different road segments that can be adjusted according to the users wishes, for example turn rates for intersections, and green light durations for traffic lights. All of these can be adjusted freely within the scripts in the repository.

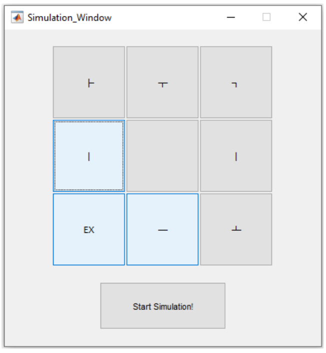
However, these values are all global, meaning that once changed, these new values will apply to all segments of that type. A more user friendly gui option for editing these valuers was planned, as well as options for individually editing these values for different pieces, however due to time restrictions this was left for future improvement.

10 Practical Examples

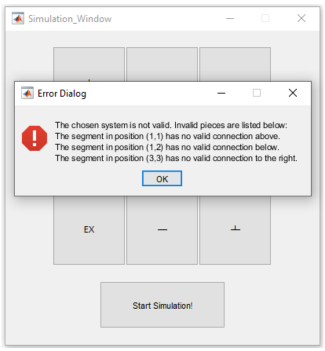
This next section provides a few practical examples of how the program performs during simulations, and to what uses it may be put. This will also feature some of the "fail-safes" that have been implemented to ensure a valid system. All of the examples that are demonstrated below, with the exception of the validity checking tool example, can be found in the Github repository mentioned earlier. In order to run these examples, simply start the program, as detailed above, choose a system size of 1, and click "Start Simulation" without specifying any road segment. Once the simulation window is open, it is a simple matter of clicking the "Load" button to load one of the available examples.

10.1 Demonstration of Validity Checking Tool

When designing larger road networks with a library of given pieces, it is easy to lose track of changes one has made in the system that could cause a road segment to lead into nowhere, causing potential problems for the program. Due to this, a validity checker has been implemented, that does a quick sweep of the proposed road network to ensure valid connections. This short piece of code has been briefly touched upon before in a previous section. Let us say that we wish to implement a fairly simple road network, just a circular segment as can be seen in Figure 17a. However, as can be easily seen, some small mistakes have been made where road segments lead to nowhere that would cause the program, in its form at the time of writing, to run into some problems. Hence, when pushing the Start Simulation button, the message as seen in Figure 17b, will appear informing the user of the potential issues the program would encounter. This message also prevents the program from sending the network to the simulation program, and as such the simulation will not initiate until these problems have been addressed and solved.



(a) Road Network as entered into the program



The chosen system is not valid. Invalid pieces are listed below:
The segment in position (1,1) has no valid connection above.
The segment in position (1,2) has no valid connection below.
The segment in position (3,3) has no valid connection to the right.

(b) Error message informing user of conflicts with their design

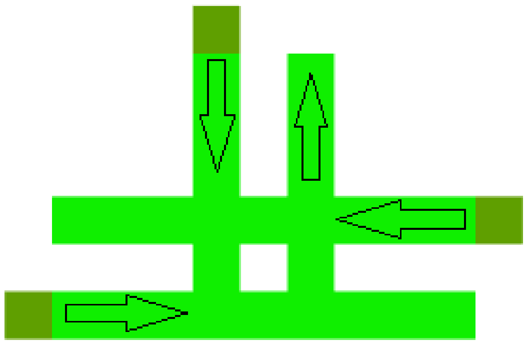
Figure 17: An example of validity checking

The method that the program uses to check the validity of the road network is relatively simple. Each road segment in the library the user has to work with, has an associated $1x4$ binary matrix. The binary value indicates whether the segment expects or has a connection in a given direction, column 1 is in the northerly direction, 2 in the easterly, 3 in the southerly, and finally 4 in the westerly direction. When checking a piece it checks whether the neighbouring pieces have an equal binary value in the relevant position. This process has been made somewhat more efficient due to the program only performing this comparison if the segment has a connection in a given direction. This means it will skip empty segments, and only perform as many comparisons as the segment has connections, which is usually not the full 4, but 2 or 3. This is not a massive time save, but means it performs quicker, especially on larger systems, where the potentiality of empty segments is much higher.

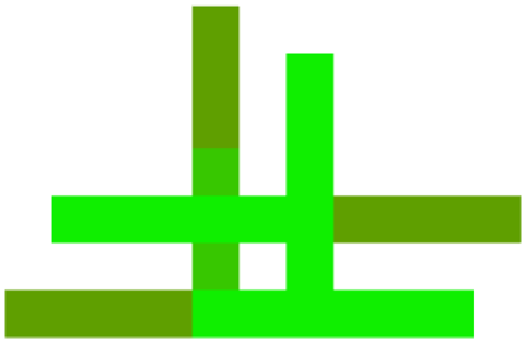
10.2 Three-way Intersection

As the simple straight pieces and turns are relatively self explanatory, we will forego an in-depth analysis of them, and instead examine the more intricate segments starting with the three way intersection. As previously introduced, the tree-way intersection consists of one main road, with traffic in both directions, and an intersecting road that allows vehicles to merge with the main lanes, but also provides a new alternate path for the vehicles on the main road. A graphical illustration can be seen in Figure 6.

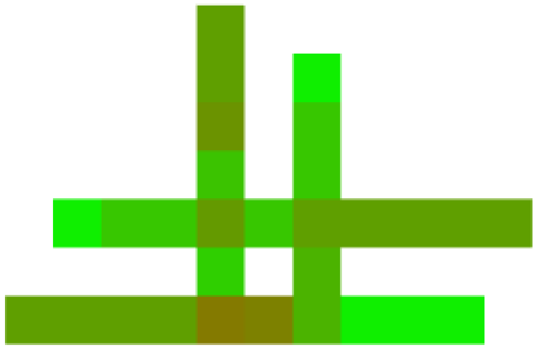
We will start simply by observing, during constant inflow of vehicles, how the flow stabilizes after several time-steps and analyze how the model behaves compared to real life situations. In Figure 18a we can see the initial conditions of the system, with the slightly darker cells being the "input" cells that introduce new vehicles into the system.



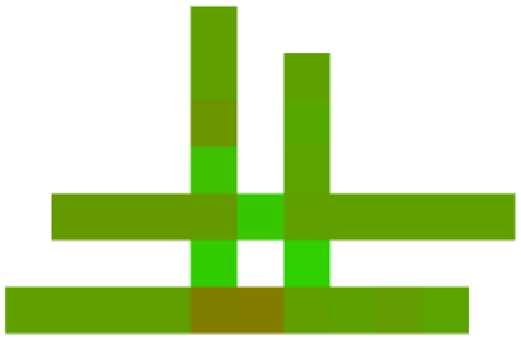
(a) Initial conditions of system, constant input of 1



(b) Simulation after 3 time steps



(c) Simulation after 6 time steps



(d) Simulation after 15 time steps, having reached a stable state

Figure 18: Intersection simulation under light flow

In Figure 18b we can see the first vehicles are approaching the intersection, with the incoming lane from the top splitting into 2 separate cells right at the intersection, those vehicles heading to the left, and those heading to the right. The vehicles that are heading to the left are located in the cell above the intersection, while the vehicles heading to the right are stored in the cell in between the lanes going left and right.

If left by itself for a short while we can already see what form the stable state of this segment will take. In Figure 18c the lane heading to the right is starting to see some increased vehicle density as compared to the other lanes. This can very easily be explained by the fact that it at this point has vehicles joining it from the lane coming from above. We can in this figure also observe the lane heading to the right splitting into two cells as it approaches the intersection allowing vehicles to head towards the top.

As the simulation stabilizes in Figure 18d, we can see that in this particular situation, with constant input of 1 and a maximum capacity of 3 for the cells, the system reaches a stable equilibrium that still permits constant flow in all directions with causing any blockages. This however is with minimal traffic, and so it would be prudent to also observe and analyse the segment under heavy flow situations.

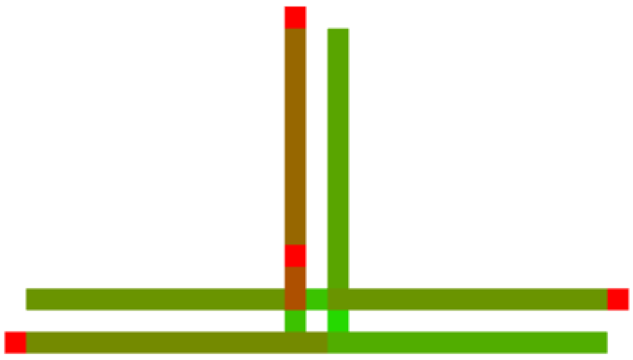


Figure 19: Simulation of intersection with heavy flow, Stable state after multiple time steps

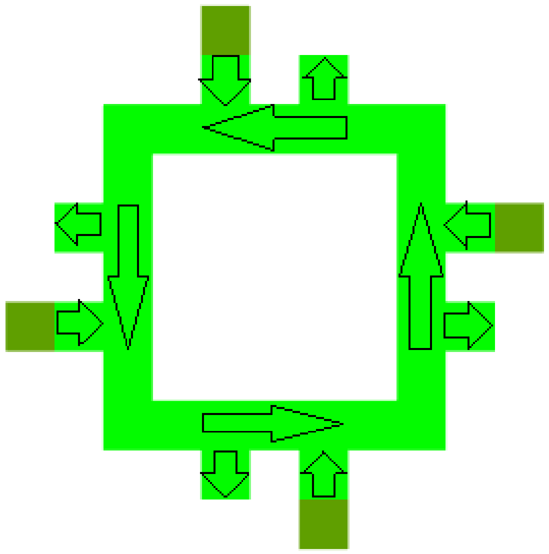
In Figure 19 we can see that the stable state for heavy flow is fairly similar to that of light flow. In this case heavy flow means that we are attempting to introduce 3 new vehicles to the system in each of the "input" lanes. As is evident from the figure, not all of these vehicles are able to enter the system, however this ensures that the system is completely saturated with as many vehicles as it is able to handle.

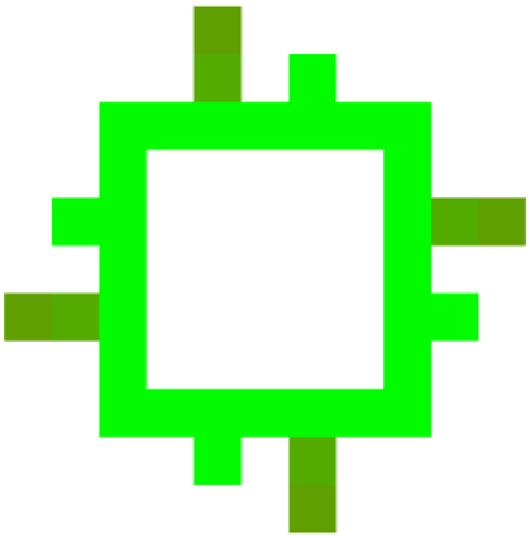
The stable state of the intersection under heavy flow does seem to replicate the real life situation of the intersection, as the intersecting lane coming from the top has on average a higher density than the two main lanes. This results from the main lane having right of way and the intersecting lane having to make use of small gaps in traffic to merge with their desired lane.

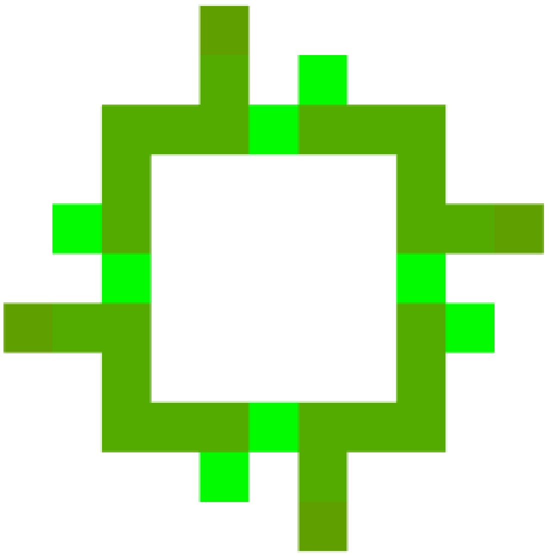
What needs to be kept in mind however is that this the segment under ideal circumstances where it is not influenced or influencing any neighbouring segments. If it is placed inside of a larger system, it is possible that the stable state of this segment could have further reaching consequences for other segments in the system.

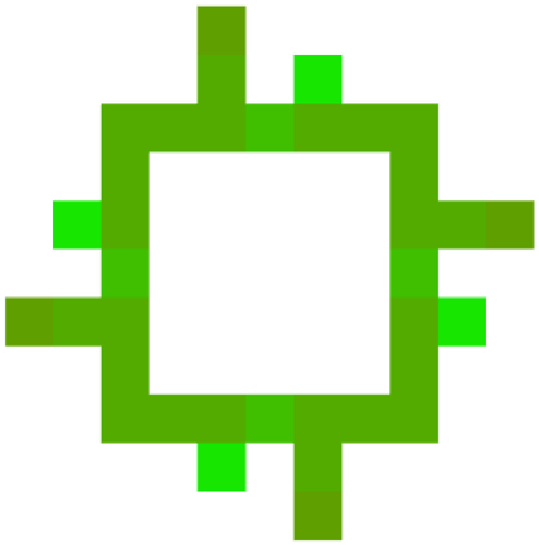
10.3 Roundabout

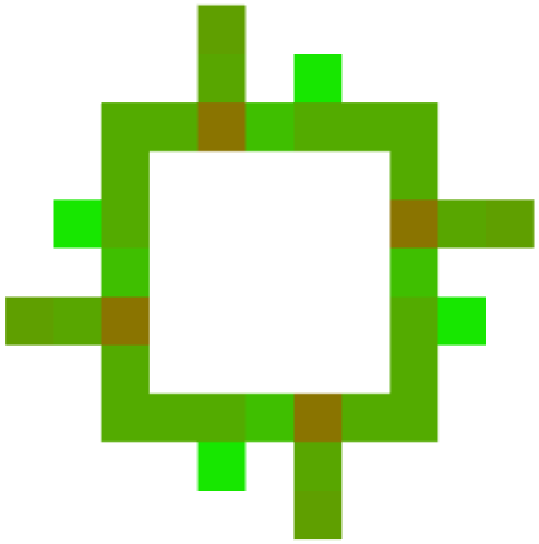
After having touched upon the three way intersection in the previous section, a natural progression would be the roundabout, as it is behaves similarly to four intersection placed in circular succession. As with our previous example, we will start by demonstrating the roundabout during relatively light traffic to glimpse how the traffic behaves.


(a) Initial conditions of the segment under light traffic, input of 1


(b) Simulation after 1 time step


(c) Simulation after 6 time steps


(d) Simulation after 7 time steps


(e) Simulation after 8 time steps

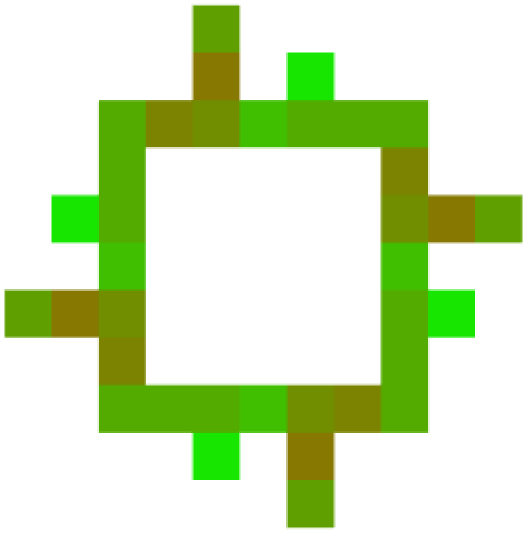

(f) Simulation after 9 time steps

Figure 20: Roundabout under light flow

The initial conditions are the same as with the first intersection example, a constant input of 1 vehicle from all directions. Entering the roundabout is a relatively simple endeavour, so we skip ahead a few time steps to the point in time where the vehicles have entered the roundabout and are approaching their first exiting opportunity, Figure 20c. In the next step, Figure 20d, we can see that a small portion of the vehicles exits the roundabout at this opportunity, while the majority of the vehicles continue onwards, approaching the location where vehicles are entering the roundabout.

Clearly the vehicles that are in the roundabout already have priority, and as such we can see in Figure 20e and 20f that a small blockade starts to become apparent at the entrances to the roundabout, as vehicles are forced to wait for a break in traffic before continuing into the roundabout. This becomes even more pronounced as we reach a stable state in Figure 21, where it is plain to see that the cells in the roundabout itself are not at capacity, but due to their priority are causing blockages in the cells at the entrances to the roundabout, that are almost at full capacity.

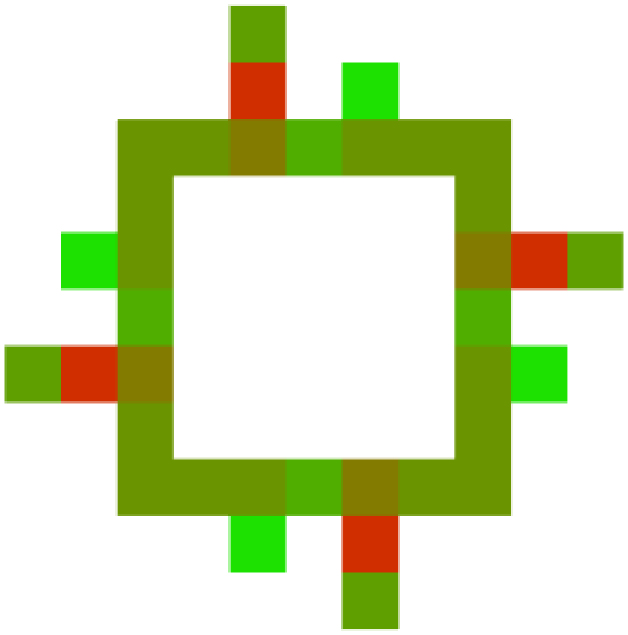


Figure 21: Stable state under light traffic flow

While the far reaching consequences of are not visible in this small single segment visualization, it is easy to imagine that in a larger system, this blockage would have much farther reaching consequences that could under the right circumstances completely block up the system it is situated in. What would be of interest in this case would be to observe whether or not a heavier flow of traffic would result in a more densely packed roundabout, and as such we will observe the system under a constant inflow of 3 vehicles from each direction.

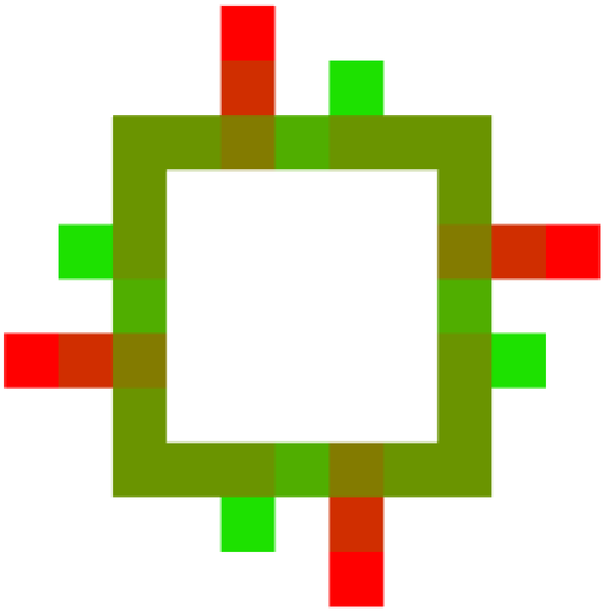


Figure 22: Stable state under heavy traffic flow

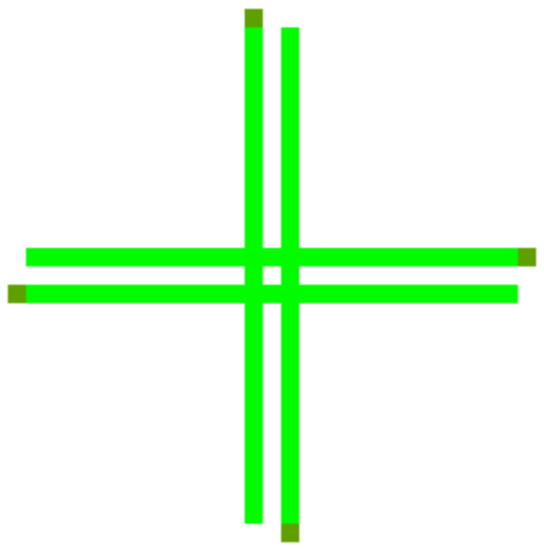
What we see in Figure 22 is not at all surprising, it is identical to the stable state under light flow, Figure 21. This is due to the stable state under light flow already having blockages at the entrances to the roundabout that could potentially enter the roundabout, but due to the priority of the interior cells of the roundabout are unable to do so. This enables the interior cells to have a constant manageable flow of traffic without becoming congested. This mirrors the function and effects of roundabouts in reality, as they rarely if ever become completely congested and have a constant flow of traffic through them, even under heavy conditions. But they can cause build ups of queues, especially under heavy conditions, as they prioritize the vehicles already in the roundabout itself.

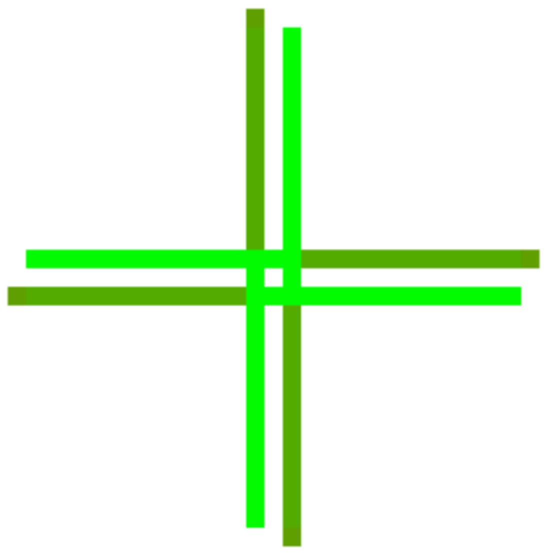
The only real conditions that would cause the roundabout start to become congested are segments further downstream that cause blockages at the exits of a roundabout, such as a roundabout with a smaller capacity than the first one, or potential traffic accidents, etc..

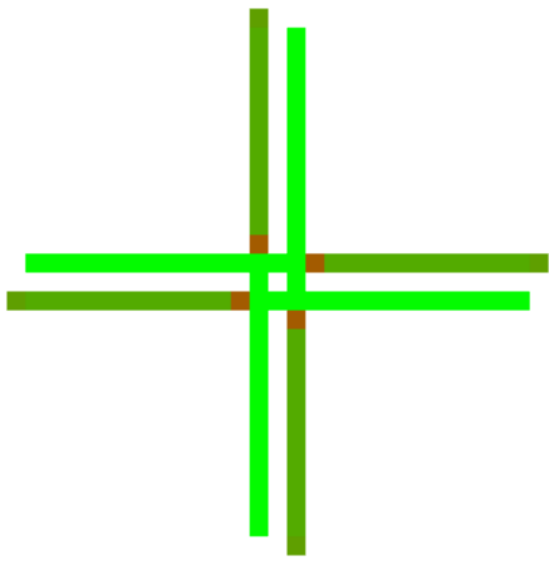
10.4 Traffic Light

The last segments that will be discussed and demonstrated in detail is the traffic light intersection, as it is, at least of the implemented segments, the most complex one. It can be thought of and treated in a very similar way to the three way intersection, but due to the fact that traffic flow changes depending on the light, the implementation becomes a fair bit more difficult, as detailed in an earlier section. This difficulty does not so much come from the actual calculation, but from the desire to reduce duplication of code by generalizing in the right places so code can be reused.

As with previous example, we shall first examine the segment under low traffic flow, however, in this case we shall not be simulating just the one segment itself. Due to the effects of the traffic light, we will be adding a straight segment of road in each direction to get a better idea of how far the blockage stretches under low conditions. Additionally, there are a few more parameters that can be toggled with the traffic light, specifically the amount of time the light is green for, and the time the vehicles have to clear the traffic light intersection before the light changes again. For this first example we shall be conducting a simulation under light flow, 1 vehicle from each direction, and 10 time steps of green light with 3 time steps between red and green light.


(a) Initial conditions of system under light flow, 1 vehicle entering the system from each direction


(b) System after 12 time steps, vehicles approaching the traffic light


(c) System after 13 time steps, traffic light is in the middle of changing

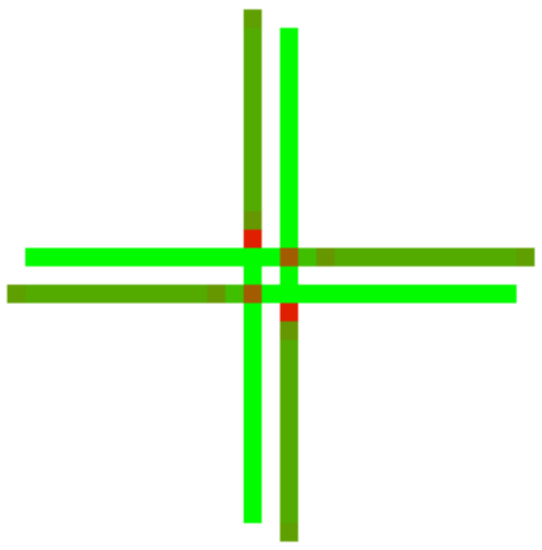
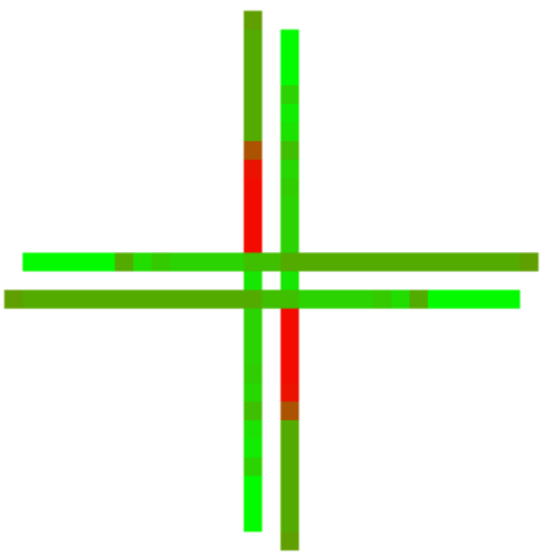
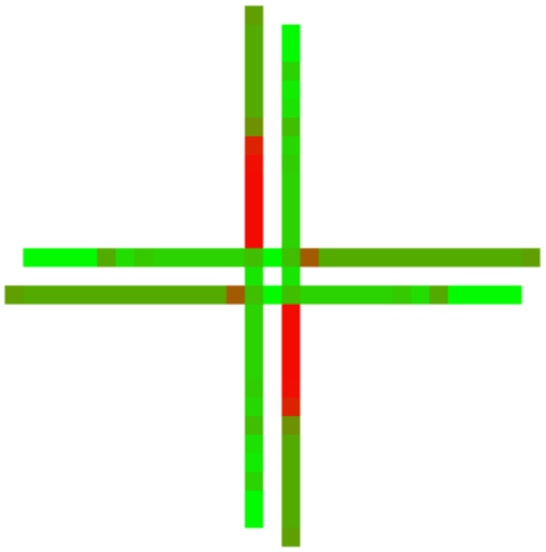
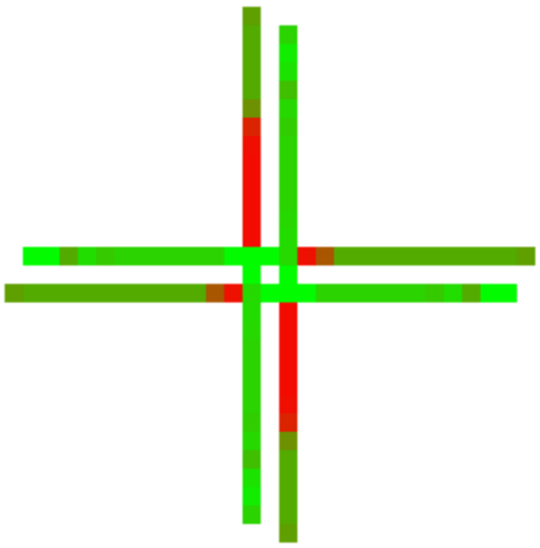

(d) System after 14 time steps, light becomes green for lanes coming from the left and right

Figure 23: Traffic light intersection under light flow part 1


(a) System after 23 time steps, last time step of green light


(b) System after 24 time steps, light turns red for all directions


(c) System after 26 time steps, last time step of red light for all directions

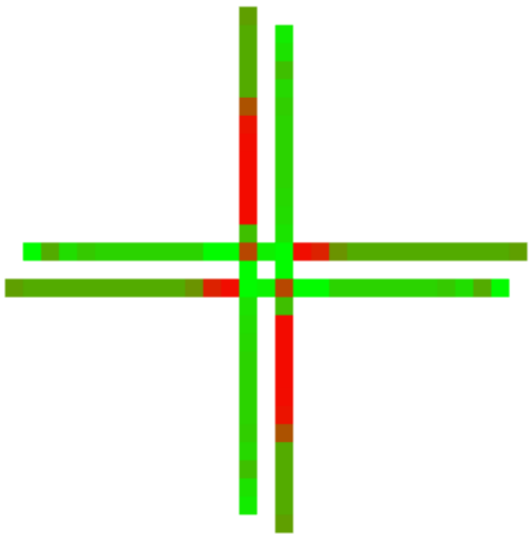

(d) System after 27 time steps, light becomes green for lanes coming from the top and bottom

Figure 24: Traffic light intersection under light flow part 2

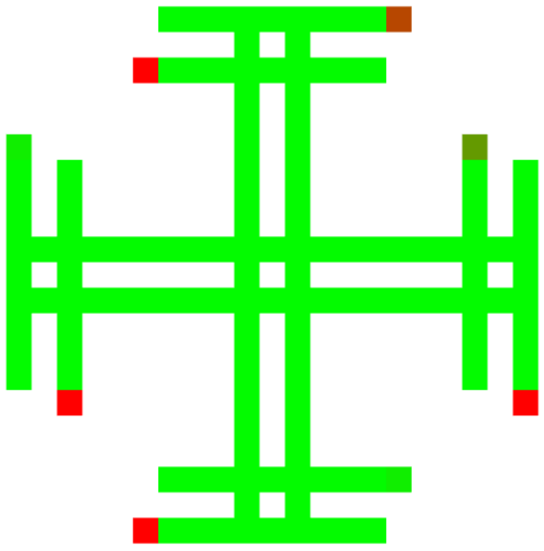
Everything progresses rather simply for the 12 first time steps, with the vehicles approaching the intersection at a point in time where all lights are red, as seen in Figures 23b and 23c. At time step 14, the light for the lanes coming from the left and right changes to green, permitting those lanes to continue on their journey, Figure 23d. This continues for the next 9 time steps as well, with vehicles branching of upwards and downwards from both the incoming left and right lanes, Figure 24a. At time step 24, the light once again turns red for all directions, stopping all traffic from entering the intersection, but still permitting vehicles in the intersection to continue on their way. Once again this changes in time step 27, when the light now becomes green for the incoming lanes from the top and bottom. This segment does not have a stable state as compared with the previous segments, so a figure of this is not provided. Instead it has a repeating pattern, the building of a blockage during red light, and the release of this blockage of vehicles once the light turns green. This pattern repeats itself, in this case, every 13 steps, alternating between the directions of travel.

As with the roundabout, this segment is able to accurately recreate real life situations, where queues of vehicles build up at a red light and are released once it turns green. However, as opposed to the last 2 pieces, there is no need to really observe it under heavy load in detail. The only difference between light and heavy flow in this case would be the fact that the queues during red lights would build up much faster, and the effects of the red light would spread further and faster through the system.

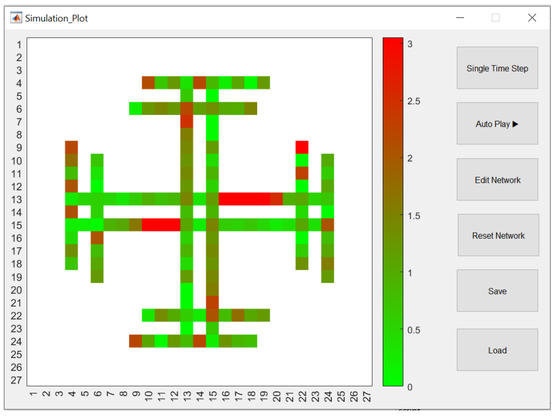
10.5 Editing and, Open and Closed System

So far, examples have been presented of open systems, where vehicles may freely enter and exit the system from predetermined points of entry and exit. However, this is not always desirable. Occasionally it is desirable to observe a system under closed conditions with a finite number of vehicles to see if the bottleneck that was observed in the open system remains the same, or whether the flow of traffic will stabilize in a different fashion over time. This is where the edit functionality is of some use, as it allows us to first populate the system with vehicles, and then close it so we can observe it with a finite, constant number of vehicles.

We will present a small example of this functionality in this section, to demonstrate and explain how it works. We will be using a relatively small example on a 3x3 system, with a traffic light in the centre, as can be seen in Figure 25a. We let this system run for several time steps in order to populate it with some vehicles, which results in what we see in Figure 25b. In this same figure we can also see the location of the edit button on the GUI, when pressed, it takes us to the edit window, visible in Figure 26a. The use of the edit window is identical to that of the initial creation window. If a change is desired, clicking on a segment brings up a list of possible pieces for the user to choose from, and all changes are finalized with the ”Resume Simulation!” button.

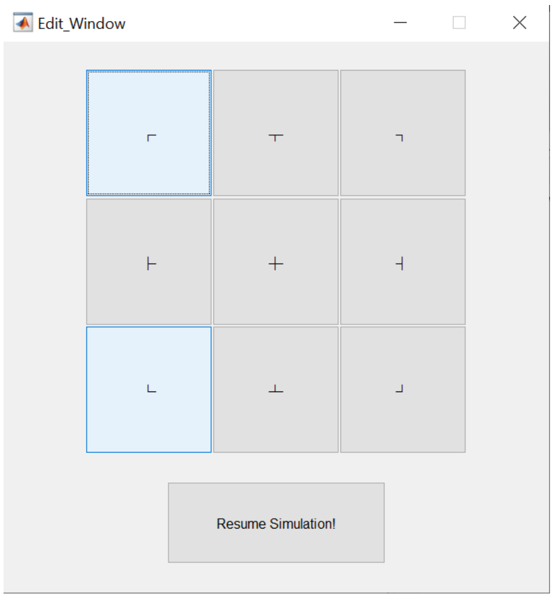


(a) Editing system example, initial conditions with random input

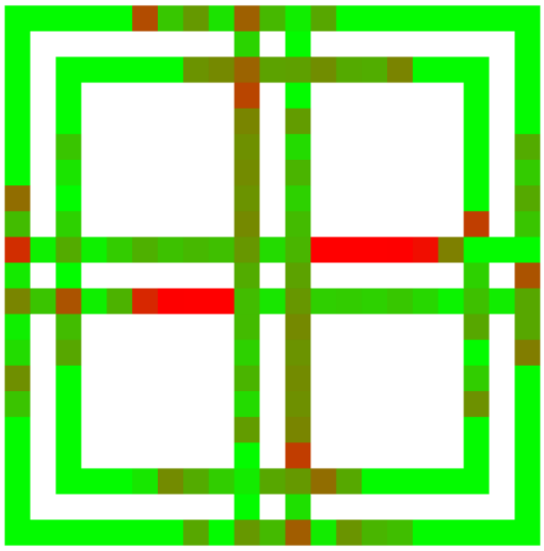


(b) System after introducing some vehicles, with the edit button visible on the right

Figure 25: Demonstration of system editing, part 1



(a) Editing window, identical to the initial creation window



(b) System after being edited

Figure 26: Demonstration of system editing, part 2

Upon resuming the simulation, there is one thing that should be noted. The way the edit function is currently implemented, it does not map the existing cars of the previous segment to the new segment, but rather initializes the new segment to be all zeros. It would not be a great problem however to modify this behaviour if desired to for example reflect the average value of the cells of the previous segment in the new segment, such a modification should be trivial.

While simple in concept, this functionality relies heavily upon the fact that the system is conservative, meaning the total number of vehicles in a closed system does not change. This was not the case with the first implementations, as mentioned before, phantom cars were introduced in intersection road segments, leading to the total number of vehicles in a closed system increasing. This fact was only discovered thanks to the edit functionality.

Spotting an error in conservation in an open system is difficult, as cars are introduced and lost every time step. However, a closed system allowed for a more in depth check of the implementations, providing useful feedback that enabled the tracking and eventual correcting of mistakes in the code that introduced the phantom vehicles.

10.6 Demonstration of Large System, Open and Closed

As previously stated, currently the size of the system is limited to a $10x10$ size, however this is not a limitation of the method, but rather a hardwired limit, that is directly coded into the program due to a limitation of the hardware that was worked with, in order to provide a smooth simulation. It is a simple matter to remove such a limit entirely by editing the file Start_Up_Window.m, depending upon the hardware that is used. Additionally, due to the limitations of a paper copy, to ensure legibility, we will be demonstrating an example on the scale of $6x6$.

The initial system we will be working with can be seen in the Figure 27, consisting of 2 traffic lights, a roundabout, 3 entrances and exits points, and a network of roads connecting them. The entrances and exits will be removed at a later time, but we need them initially to introduce vehicles into the system.

As we can see in Figure 28a, after 60 time steps queues are forming both at the traffic light and roundabout, with the traffic light currently having the denser traffic. However, we can see in Figures 28b, 29a, and 29b, that as the system moves on, the queue at the roundabout is consistently getting longer and longer, while the queue at the traffic light is regularly released as the light changes. This situation is not completely unexpected. While the roundabout does permit a constant flow of vehicles to enter, it needs to ensure that the interior cells of the roundabout do not become too densely packed,

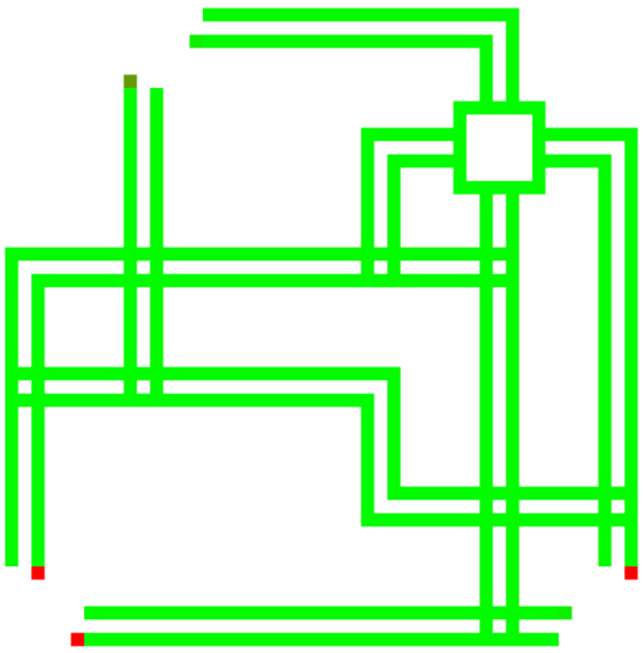
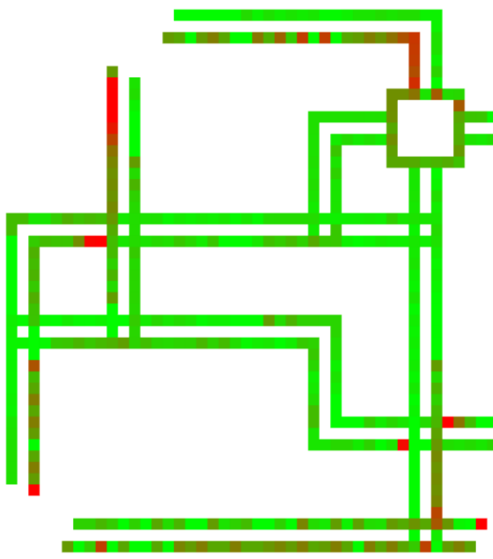


Figure 27: 6x6 road network demonstration, initial conditions

as that would lead to a complete blockage of the roundabout, which means it can accept less vehicles than are available. This leads to the effects as seen in the figures, with the queues at the roundabout spreading further and further throughout the system. The traffic light on the other hand, while completely stopping traffic in its tracks during the red light phase, enables more or less complete flow of vehicles during the green light phase. To see how this system develops with a finite number, or unchanging number, of vehicles, the edit function of the program was used to remove the entrance and exit segments, and replace them with appropriate pieces to create a closed system.

In Figure 30a it can be seen that the trend described earlier continues to develop, with the roundabouts effects spreading even further across the network. The traffic lights effects, have not spread nearly as much as the roundabout and are still permitting a relatively steady flow of vehicles to move throughout the system.

To see the effects on the system if the roundabout is replaced with a third traffic light, we did exactly this, and let the system run for a short period of time to stabilize it, resulting in what we see in Figure 30b. As can be clearly seen, the resulting traffic flow is much smoother than with the roundabout, permitting a steadier flow through out the entire system.


(a) Simulation after 60 time steps

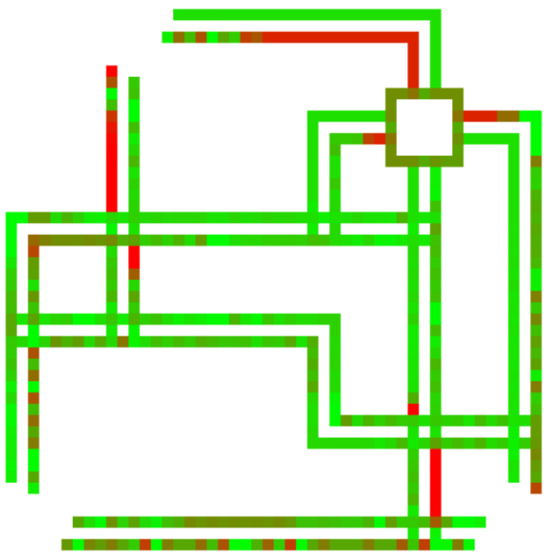
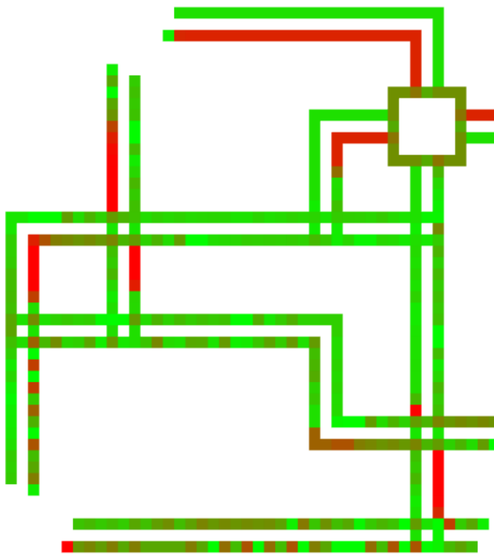

(b) Simulation after 100 time steps

Figure 28: Large system demonstration, part 1


(a) Simulation after 150 time steps

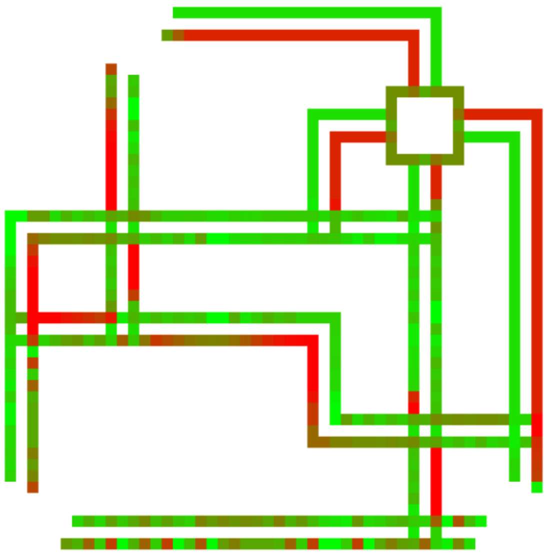
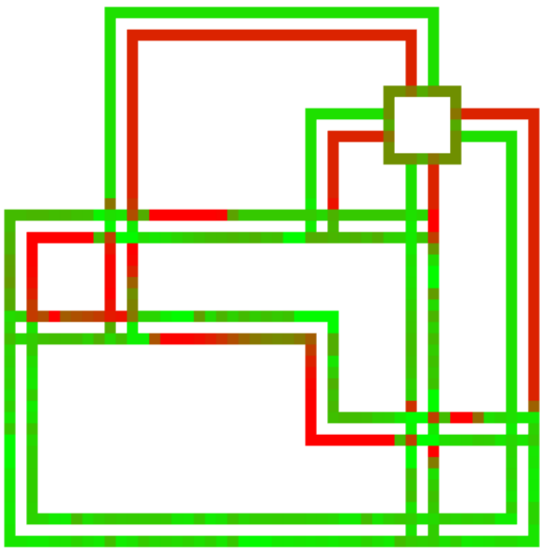

(b) Simulation after auto-playing to stabilize the system

Figure 29: Large system demonstration, part 2


(a) Simulation after closing the system and letting it run to stabilize

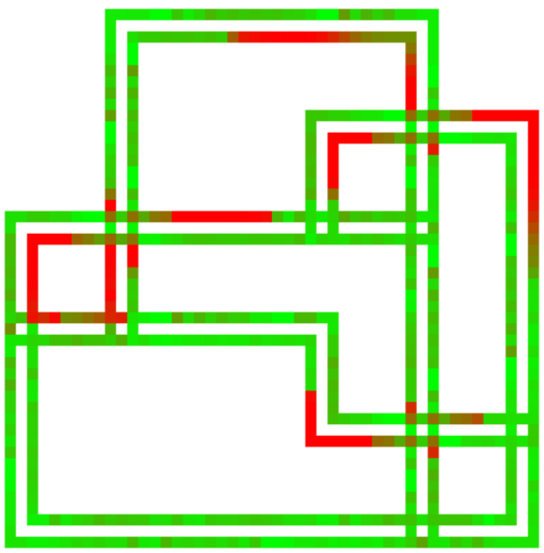

(b) Simulation after exchanging the roundabout for a traffic light and stabilizing

Figure 30: Large system demonstration, part 3

On the contrary, if we were to exchange all traffic lights with roundabouts, the consequences are much more drastic and cause the entirety of the vehicles in the system to become stuck in queues at the roundabout. This result is demonstrated in Figure 31.

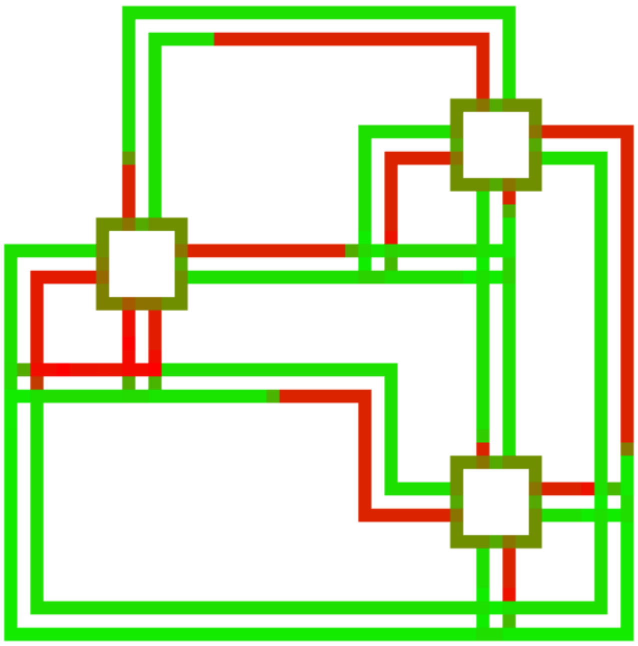


Figure 31: Demonstration system with all roundabouts after stabilization

This simulation does run somewhat counter to what research suggests happens in real lie situations. Research indicates that roundabouts tend to improve traffic flow as opposed to traffic lights [1]. The reason why our simulation would suggest otherwise could be due to a number of different factors, among them the most likely culprits being the fact that this is a very simplistic roundabout, with only a single lane in the interior of it. In actual roundabouts, there tend to be 2 or more lanes to differentiate between vehicles intended to head to the right, straight or the left, which could greatly improve the flow of traffic and also allow more vehicles to enter the roundabout. Additionally, a further effect of the roundabouts not witnessed in the simulation (at least not in the current version as of writing), is the number of accidents. As drivers are not attempting to beat the traffic lights, and just make it across before the light turns red, and have to judge the traffic in the roundabout, the number of accidents in roundabouts is greatly decreased versus the number of accidents in traffic light intersections.

Additionally, it should be noted that the images in this document are not able to show all of the fine variations in density, so it is highly recommended to download the accompanying files from the Github repository mentioned earlier to get a better feeling for the actual flow of the different systems.

11 Limitations of the CTM

While the CT method is simple to implement and equivalent to the hydrodynamic model, it does suffer from some limitations that can lead to it being overlooked or dismissed for certain implementations. In this section some of these flaws are outlined, and potential solutions or fixes are outlined for some of the flaws.

11.1 Velocity

The first and most obvious limitation of the so far described implementation of the CT method, is the simple fact that it does not take into account the velocity of the vehicles. All vehicles simply move from one cell to the next as time steps forward. This is however not in keeping with reality. The model thus far indicates that all vehicles have two states, dead stop, or full speed ahead, whereas in real life situations vehicles would accelerate towards their top speeds and gradually come to a stop.

This limitation however does not just extend to the non existence of a velocity element, but also introduces a limitation of the simulation itself. As we are unable to control the speed of the vehicles directly, all roads of equal length, will be travelled at the same speed, meaning it is not possible to introduce speed limits. This, once again, does not conform to reality, as a 1 km section of highway enables a much higher speed than a 1km section of road in the city centre. Unfortunately, in the current model, both of these road sections would be travelled at the same speed, meaning that if they were parallel, the model would predict that both would be equally fast, whereas in reality the highway would be much quicker than taking a detour through the city centre.

11.1.1 Potential Solution

Due to the nature of the method itself, constructing a solution to this problem is not a simple matter. However, one idea does spring to mind, that would at the very least enable the user to choose different maximum speeds for each road segment. The idea is fairly simple, depending on the chosen maximum velocity the simulation consists of fewer and fewer cells, and then for the final step of visualization would be mapped onto a standard format that makes all segments equal in length. It would then appear as if some road have a higher maximum speed than others, also widening the possibilities of the simulation.

However, this is far from a perfect solution. This idea would potentially provide a user with more possibilities for their model, however it does not solve the problem of the two states, the fact that vehicles are either moving at top speed or at a stand still. For the scope of this project, a solution to this particular aspect was not thoroughly investigated.

11.2 Intersections

The way intersections are implemented currently there is one small flaw in the case of for example roundabouts. The cell in which the flow of vehicles splits, takes a certain fraction of the vehicles and splits them from the "main" flow of traffic. While this does work quite nicely for a single intersection, for multiple consecutive intersections in a circular formation, this produces a small error. Due to the nature of how the vehicles are split, if we were to want a roundabout where $\frac{1}{4}$ of the vehicles take each of the 4 exits, the current method would not work. The first exit would take a quarter of the vehicles, but the next exit in line would only take $\frac{1}{4}\frac{3}{4} = \frac{3}{16}$. This trend continues, for the next 2 exits as well, each taking progressively fewer and fewer vehicles, with the last one only taking $\frac{27}{256}$ of the original amount of vehicles, decisively less than the intended quarter.

This also introduces another issue with the roundabout, the fact that it will never "completely" empty. As we are continuously taking only a fraction of the vehicles in the roundabout and having them exit it, there will always be some small, eventually infinitesimally so, amount of vehicles left in the cell unable to exit. Intrinsically, these two problems are linked, and so if a solution could be found for the initial issue as addressed in the previous paragraph, both of them would be solved.

11.2.1 Potential Solution

There is one potential solution that springs to mind, and would involve providing each vehicle with a destination. With a known destination, splitting the cells at intersections would no longer be necessary, and would resolve both issues mentioned above. This solutions however, at a first glance, would be vastly more complicated to implement as each vehicle now has several more values associated with it, and as seen will be seen shortly requires several additional calculation and comparison steps.

Unfortunately for the scope of this project, this potential solution was not implemented, but would be within the realm of possibilities given further time.

12 Additional Improvements and Functionality

During the course of the this project, the scope of the goal changed somewhat, and several additional useful features for simulation were considered, but not implemented due to time restraints. However, these features could provide a vastly improved implementation, and as such are documented here for use for future projects or improvements.

12.1 The Lagged Cell Transmission Model

In 1999 Daganzo published a new article, introducing a new modification of the CTM, intended to once more improve the accuracy of the method, namely the Lagged Cell Transmission Method (LCTM). It was discovered that the initial method of 1993, while accurate for the limit of vanishingly small cells and clock ticks, it generated numerical errors when using more realistic/practical values. Namely, when using more practical (i.e. larger) cells sizes and clock intervals, the method became increasingly inaccurate.

The solution that was devised was once again a modification to the method of flow calculation. Instead of using the current downstream density to calculate the flow into the next cell, a lag $l > 0$ is implemented. In effect, instead of using the density of the current clock tick when calculating the flow into the next cell, the density from l clock ticks earlier would be used. This new and improved method is closely related to Godunov's first order method for general flow-density relations and Newell's exact method for concave flow-density relations. The advantage of the LCTM over these two methods is that it is easier to implement than Newell's, and more accurate than the Godunov first order method, going so far as to be a second order method given the right flow-density relation and choice of l.

Applying this modification to our model (21), we obtain the following LCTM:

$$y_i(t) = \min\{S_i(t), R_i(t-l)\} \tag{25}$$

$$S_i(t) = \min\{Q_i, n_i\} \tag{26}$$

$$R_i(t) = \min\left\{Q_i, \frac{w}{v}[N_i - n_i]\right\}, \tag{27}$$

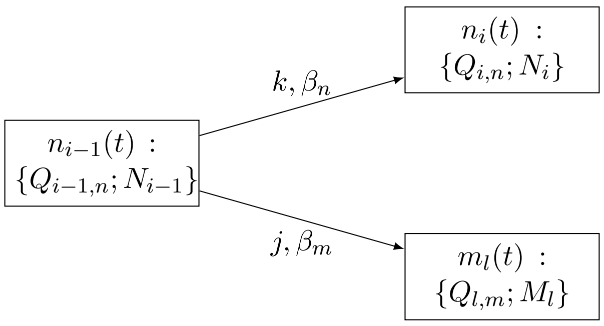
12.1.1 Advantages

As briefly touched upon previously, the LCTM changes the definitions of some variables as compared to the CTM. These changes in definition provide us with further freedom when creating traffic simulations. Specifically, due to our redefinition of Q_i we are now able to apply the LCTM to road networks which have been discretized with cells of different lengths. This is advantage for example in the case of long stretches of straight highway, instead of having section only 100m long, it would be possible to have cells representing for example 10's of kilometres.

12.2 Models with Known Routes

As mentioned earlier, the method of having predefined turning percentages does indeed work, but in this section we will present a method that does not need pre existing turning percentages, but rather calculates these turning percentages based upon the vehicles destination and chosen route.

The simplest form of a situation with diverging path's is depicted below, along with the variables associated with the different cells:



$n_i(t)$: $\{Q_{i,n}; N_i\}$
k, β_n
$n_{i-1}(t)$: $\{Q_{i-1,n}; N_{i-1}\}$
j, β_m
$m_l(t)$: $\{Q_{l,m}; M_l\}$

The variables here are as described in earlier sections with 2 exceptions. The variables β_m & β_n are the turning percentages of the different paths, they are not given, but included in the diagram nonetheless as they will be calculated from the destination and path information from the different vehicles. Secondly the variables k & j represent the links between the cells.

It is therefore necessary to further separate or disaggregate the data in the cells according to their destination, d and time waited, τ. This new set of disaggregated data will be denoted by: $n_{Id\tau}(t)$, the number of vehicles in cell I heading to destination d that entered cell I $(t - \tau)$ clock ticks ago. This however does not simply hold true for the cells themselves, but also the links flowing from one cell to the next, meaning the data on the links will be disaggregated as follows: $y_{kd\tau}(t)$, the number of vehicles flowing on link k at time t, headed for destination d after having waited τ clock ticks in the previous cell.

As new have separated the data in new categories, we will of course have to adjust our update scheme accordingly. Instead of all traffic flowing into cell, the new traffic coming to a cell will be separated according to their destination, with $\tau = 1$ as they have just now entered their new cell.

$$n_{id1}(t+1) = \sum_{\tau} y_{kd\tau}, \ \forall k, d. \tag{28}$$

This takes the sum of all vehicles flowing on link k on the time interval $(t, t+1)$ heading to destination d, over the time they have waited τ and saves it in a value representing the number of vehicles in cell i headed for destination d that have just entered the cell, hence $\tau = 1$.

Additionally we will also have to each clock tick update the wait times for the vehicles that were unable to move onwards towards their destination:

$$n_{(i-1)d(\tau+1)}(t+1) = n_{(i-1)d\tau}(t+1) - y_{kd\tau}, \ \forall k, d, \tau. \tag{29}$$

The next step will be to find new expressions for the disaggregated flow functions $y_{kd\tau}$. The approach that will be employed to determine which vehicles move on to the next cell will be a FIFO (First In First Out) approach, meaning that the vehicles that have waited the longest are the first to move on to the next cell. This step is reduced by Daganzo et al in 1995 to the task of finding a real variable $a_I(t)$ to denote the minimum wait time of the vehicles leaving cell I during the time interval $(t, t+1)$. This means that any vehicles that have a $\tau < a_I$ do not get to move on as others have waited longer; any vehicles with $\tau > a_I$ get to move on as they have waited the longest; and finally a portion of the vehicles with $\tau = a_I$ get to move on. Expressed mathematically, this gives us the expression:

$$y_{kd\tau}(t, a_{i-1}) = \begin{cases} n_{(i-1)d\tau}(t)\beta_n, & \tau > \lceil a_{i-1} \rceil \\ (\tau - a_{i-1})n_{(i-1)d\tau}(t)\beta_n, & \tau = \lceil a_{i-1} \rceil \\ 0, & \tau < \lceil a_{i-1} \rceil \end{cases}, \tag{30}$$

where $\lceil a_{i-1} \rceil$ denotes the smallest integer equal to or greater than a_{i-1}. This particular method can also be applied to ordinary road sections, all that has to be done is set $\beta_{i-1} = 1$, indicating that all vehicles take the same route, i.e. the only one available to them. Using this method we ensure that the vehicles that have been in the cell the longest are the first to leave it. Next comes the calculation of the values a_I. As an intermediate step, we will define to functions that relate aggregate link flows to a_I. One where we take the sum of all link flows over the wait times and another were we take the sum over the wait times and destinations.

$$y_{kd}(t, a_{i-1}) = \sum_{\tau} y_{kd\tau}(t, a_{i-1}), \tag{31}$$

$$y_k(t, a_{i-1}) = \sum_{\tau} \sum_{d} y_{kd\tau}(t, a_{i-1}). \tag{32}$$

Using the inverse relationship between a_{i-1} and y_k we can devise a method for the calculation of a_I. Both of the functions (32) & (30) are non-increasing, piecewise-linear with changes in slop at the integers. Due to the non-increasing property, it is possible to define a non-increasing inverse function to (32), A_k, which calculates the minimum wait time at the link's source cell, $i-1$, given the aggregate flow y_k.

$$a_{i-1} = A_k(t, y_k). \tag{33}$$

In the case of a straight road, he defined equations (33) & (32) are only meaningful as long as the flow ranges between 0 and the maximum number of vehicles that the cell $i - 14$ is able to send, $0 \leq y_k \leq S_{i-1}(t)$. In the case of a intersection or other situations were the road diverges into several paths, the flow needs to satisfy $0 \leq y_k(t, a_{i-1}) + y_j(t, a_{i-1}) \leq S_{i-1}(t)$, i.e. the sum of the flows out of $i-1$ needs to between 0 and the maximum that cell $i-1$ is able to send. This means that the variable a_{i-1} varies from the smallest value that does not break the upper limit (the smaller the value of a_{i-1} the more cars have the chance to move on to the next cell), we will denote this value as $A_{i-1}(t) \geq 0$; to the largest wait time that is present in the previous upstream cell, denote by $\tau_{i-1}(t)$.

Using all of this, we can now move on to identifying $a_I(t)$ for our situation of diverging paths. The sending cell at the diverge will send as many vehicles as it possibly can down both paths, k & l providing that these two flows satisfy the constraint conditions outlined earlier. From this we can conclude that a_{i-1} would be the smallest value satisfying the following conditions:

$$y_k(t, a_{i-1}) \leq R_i \tag{34}$$

$$y_j(t, a_{i-1}) \leq R_l \tag{35}$$

$$0 \leq y_k(t, a_{i-1}) + y_j(t, a_{i-1}) \leq S_{i-1}(t). \tag{36}$$

Due to the properties of y_k and y_j, the left sides of these inequalities is non-increasing, and additionally the last inequality is only satisfied if $a_{i-1} \geq A_{i-1}$. As a consequence of these two properties, our solution is then:

$$a_{i-1} = \max\{A_k(t, R_i), A_j(t, R_l), A_{i-1}\}. \tag{37}$$

12.3 Further Customization of Road Segments

While the current state of the program does provide the user with a fair degree of customization, it is still fairly bare bones. The user may have control over the shape, size and exact dimensions of the system, however they can not change specific values for specific cells, i.e. allow them to specify how many vehicles a certain cell can contain, and what each individual cells flow rate is. The way the program works currently, it will continuously simulate the road network in ideal conditions, not allowing the user to add in values for potential accidents or traffic stops etc..

What one could implement is a window that displays a desired road segment much like the representation shown in Figure 13, but with the values for flow rate or maximum occupancy for example, and allow the user to make direct changes. This would allow the user to pause the simulation, decide that an accident occurred in a certain cell, and modify that cell to have a maximum occupancy of 0, creating a traffic jam, and forcing other vehicles to re-route and take different paths.

There would however be an increase in storage space associated with this implementation. Currently, there is one global value that dictates the maximum occupancy and maximum flow rate respectively. If this idea were to be implemented however, one would have to store one such value for each individual cell. This means that for each cell, instead of just storing the current occupancy, we would also store the maximum occupancy and flow rate, effectively tripling the required storage space for each segment. However, as we are still talking orders of n, this increase in storage space should be bearable, and would provide the user with further tools to accurately model their desired network.

This idea can also be extended not just to the maximum occupancy and such, but also the timers for traffic lights. Currently, all traffic light intersections have the same timers, for both the main and side roads. But it should not be a difficult task to implement a small window, enabling the user to freely define the timers for each intersection, ensuring that more frequently travelled roads are green for longer at the intersection.

12.4 Custom Road Segments

One other functionality that was debated and somewhat experimented with was a custom road segment creator, allowing the user to create their own unique road segments to suit their tastes. This idea was split into 2 sub sections, creating from previously defined segments, and creating from scratch.

Creating a new road segment from already existing ones would basically be a matter of letting the user create a road network, and saving it as an actual road segment that could be inserted into other networks. While this idea in and of itself is very intriguing, and should be achievable, it does however run into a small problem. If we were to say create a new road segment out of a 3x3 grid of already existing road segments, it would become difficult to use this newly defined road segment as it has a different length compared to the predefined segments, meaning that visualization becomes somewhat more difficult.

While potentially a daunting issue, it is resolvable to some degree if a predetermined size of the new road segments is agreed upon. If this is the case, then it is simply a matter of having segments check their neighbouring segments and adjusting their length accordingly to ensure that connections are still made in the correct location. But as stated, this is only achievable if the user defined road segments are ALL of a predetermined size. If this is not the case, then scaling becomes immensely difficult as there are multiple adjustments that have to be made, especially if user defined segments have to be scaled.

The creating from scratch idea would to some degree solve this issue. The idea is to allow the in a separate window create a new segment of the same size as the already existing ones. If the size of the predefined segments is 9 for example, the new window would open up a 9x9 grid, that would allow the user to ”draw” their own segment.

While this method of creating new segments would solve the issue of size, it does however create a new issue. Due to giving the user complete control of the segment there are a myriad of things that the program will firstly have to check in order to assure itself and the user that it is a valid segment. Such a debugging tool would take time and trial and error to create, but not impossible. And secondly of course there is the matter of the actual code to move vehicles to the next cell during a time step for such a user defined segment.

This is by far the biggest of the two challenges, but also not completely out of reach. The general idea that was toyed with is having the program check the connectivity of the user defined segment cell by cell. It would check this connectivity against a predefined database of all possibly connectivities a cell can have and calculate the next time step accordingly. Duplication of effort could be avoided if it were possible for the program to automatically generate an individual function for the new road segment, avoiding having to recheck the connectivities each time such a road segment is placed in the network. However, this possibility of having a function write and create functions has not been explored, but at the same time does not seem completely unrealistic.

12.5 Improved Traffic Light Functionality

The way traffic lights are implemented at the moment is relatively simple, with all of the lights being on a global timer. However, this is not always desirable, specifically in the case where no traffic is incoming from the left and right lanes, but heavy traffic is flowing from the top and bottom lanes. Switching the top and bottom lanes' light to red would be foolish as it would accomplish nothing and only hinder traffic flow. As such some further functionality could be implemented to make the traffic lights "intelligent".

This would involve an extra condition that would regulate whether the traffic light changes colour. It would be a condition that checks the density of the cells in the directions in which the light is currently red, and if they are occupied, change the lights given that a minimum amount of time has elapsed for the lanes that currently have the green light. This would enable a much smoother flow of traffic during differing flows from differing directions.

13 Concluding Remarks

The Cell Transmission Method provides several advantages, it is easy to implement, can be implemented in such a way as to provide a great deal of modularity and flexibility, and can realistically simulate real life situations. However in its current state, it has drawbacks, most notably it does not take into account the speed at which vehicles are travelling, only providing them with a binary state for velocity.

The current state of the implementation, while providing a great deal of flexibility and user choice, lays the groundwork for further functionality. While it would be wise not to rely on this implementation for planing a real road network in its current state, it would be able to at least indicate the effects certain segments will have on the overall system.

The most crucial improvements that would have to be made for this implementation to achieve such a role would mostly be user freedom options, such has being able to customize the flow rate and cell capacities, and constructing user defined segments. Such options while potentially time consuming to implement, should be trivial. The more difficult task being the one of attempting to solve the issue of the binary velocity state.

References

[1] Roundabouts vs. traffic lights. `https://www.acsengineers.com.au/2016/08/22/roundabouts-vs-traffic-lights/`. Accessed: 16-05-2019.

[2] Carlos F Daganzo. The cell transmission model: A dynamic representation of highway traffic consistent with the hydrodynamic theory. *Transportation Research Part B: Methodological*, 28(4):269–287, 1994.

[3] Carlos F Daganzo. The cell transmission model, part ii: network traffic. *Transportation Research Part B: Methodological*, 29(2):79–93, 1995.

[4] Carlos F. Daganzo. The lagged cell-transmission model. *Part II: Network Traffic, Transpn. Res. -B*, 29, 01 1995.

[5] Martin Treiber and Arne Kesting. Traffic flow dynamics. *Traffic Flow Dynamics: Data, Models and Simulation, Springer-Verlag Berlin Heidelberg*, 2013.