# Self Organizing Motorway Simulation

Ayoola Bashir Idris-Animashaun
*Modelling, Simulation and Optimization*
*M.Sc. Data Analytics*
*National College of Ireland*
Dublin, Ireland
x20103689@student.ncirl.ie

***Abstract*—In this paper, we will simulate a traffic environment proposed for Self Organizing Motorways. Modelling and simulation techniques will be deployed to arrive at the traffic flow rate, the traffic density, the mean speed of vehicles, and the total time travelled. The scenario consists of a two kilometer three lane segment that merges into two lanes for one kilometer. At the end of the paper, a recommendation will be given for the speed limit on the highway that will result in full capacity with free flowing movements and as little bottlenecks as possible.**

***Index Terms*—modelling, traffic, simulation, discrete event systems**

## I. Introduction

Modelling and Simulation refers to a discipline within computer science that assists industries to design, create and evaluate simple and complex working environments. The discipline is able to model real or proposed systems with the help of programming tools and object oriented base techniques [2]. The goal of most simulations is to test 'what-if' scenarios to better prepare or adjust for an on-going or unknown situation. One advantage of simulations is they are often useful to test changes that are difficult to implement, carry high costs, will take a long time to complete or are nearly impractical. Examples of simulation use-cases include genome engineering in healthcare, remote drone applications used in space, weather forecast, highway traffic optimization and more.

### A. Simulating traffic environments

In simulating highway traffic, the goal is often to determine how the given players within the traffic environment operate within themselves and between themselves to have free flow of traffic and no bottle-necks. The research by [3] in using a decomposed queuing theory for multi lane multiple intersections is particularly interesting. The research is carried out using discrete-event simulation for building a traffic model that can predict traffic performance measures.

In this paper, we analyze the "Self organizing Motorway" proposed by Trinity College Dublin (TCD) and simulate a traffic environment to answer questions like

1. What is the number of vehicles per kilometer available on all lanes, k, and the mean speed of the vehicles, u. 2. What is the volume of vehicles passing all the lanes per hour and the average total travelling time. 3. What is the highest traffic volume the motorway can accommodate and give recommendations on speed limits for the motorway segment

## II. Methodology

The sequence in Figure 1 shows the sequence of steps in our method. We will be using the, Multi-lane Mixed Simulation by [1] and expanding on some parts of it.

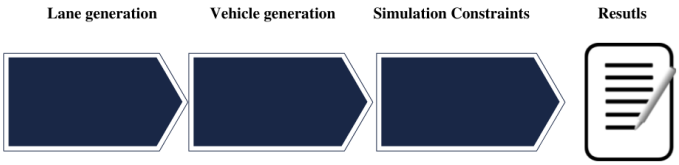


Fig. 1: Methodology sequence

To begin with, we will need a method to generate the distributions for our lanes, our vehicles, speed, length and more. To achieve this, we will implement a function to generate uniform random distribution when we call it. The function will take a mean, standard deviation and a delta value that restricts the random values we want to generate in a positive and negative direction.

Next, we generate lanes with a python class and give it attributes like length, merge on (left/right), directions (left lane or right lane) like in real life. The lanes are created as left or right lanes, with right lanes termed the 'fast' lane and is where overtaking happens, while the left lane, is the 'slow lane' and goes at the current speed limit. The lanes are programmatically linked such that vehicles know they can switch to this lane if allowed, and vehicles also know not to switch if not allowed. Lastly, the lane class has event states that track when vehicles enter the lane, when they exit the lane and what lane they exited to.

Next, a **vehicle class** is responsible for generating different kinds of vehicles with differing attributes like length, position of vehicle on the lane, acceleration, deceleration, emergency braking, thinking time of driver, distance to other vehicles, etc. A vehicle has to know its own position and its position in relation to other vehicles and the lane itself. A surround class handles this for this vehicles by checking whether there is a vehicle or obstruction in 8 locations around each vehicle. A vehicle uses the surround class when it wants to overtake and to avoid collisions with other vehicles or crashed vehicles.

## III. The Simulation Model

Adjustments were made to the existing classes within code base in [1]to simulate the environment based on new attributes. A summary table is available in Table 1

### A. Model for Vehicle

The vehicle class from the code base is modified to introduce new attributes. The attributes will help us generate different types of vehicles to simulate real world scenario. We introduce heavy goods vehicles (HGV), electronic vehicles (EV) to our vehicle types. Each vehicle comes with its own set of acceleration, deceleration, free flow velocity and length as shown in Fig vehicle types

Fig vehicle types

```
#Create different vehicle types
if vehicleType == 'FamilyCar':
    self.a_brake = -4.0 # [m/s²]
    self.a_coast = -0.6 # [m/s²]
    self.a_max =    2.5 # [m/s²]
    self.length = rv(4.2, delta=2) # [m]
    self.vtype = 'Family Car'
elif vehicleType == 'Tesla':
    self.a_brake = -8.0 # [m/s²]
    self.a_coast = -1.8 # [m/s²]
    self.a_max =    4.6 # [m/s²]
    self.length = rv(4.6, delta=2) # [m]
    self.vtype = 'Tesla'
elif vehicleType == 'HGV':
    self.a_brake = -2.5 # [m/s²]
    self.a_coast = -1.3 # [m/s²]
    self.a_max =    1.8 # [m/s²]
    self.length = rv(15, delta=2) # [m]
    self.vtype = 'HGV'
else:
    raise ValueError('Unknown vehicle type: '+vehicleType)
```

Fig. 2: Vehicle types

From a review of [4], we find that the test lengths of the SUVs used in the research ranged from 4m to 5m, as a result, for each of our vehicle types, we modify the existing code to generate vehicles with varying lengths using some set average parameters. The *rv()* function in the code base helps us generate the random lengths at vehicle generation. See fig 2 for sample lengths with vehicle types.

The constructor is modified to receive a vehicle type and return the set attributes for that vehicle. The *'FamilyCar'* vehicle type is left as the default if no vehicle type is given.

Other modifications on the **vehicle class** include adding the *'vtype'* attribute as a vehicle type to allow easy analysis when

```
t=    3.6s x=    0.0m v5, a Family Car, with length 4.5
t=    4.0s x=   13.1m v5, a Family Car, with length 4.5
t=    4.0s x=   13.1m v5, a Family Car, with length 4.5
t=    4.5s x=   25.6m v5, a Family Car, with length 4.5
t=    4.5s x=   25.6m v5, a Family Car, with length 4.5
t=    4.7s x=    0.0m v0, a Tesla, with length 4.4 Adjus
t=    5.0s x=   10.0m v0, a Tesla, with length 4.4 Adjus
t=    5.0s x=   10.0m v0, a Tesla, with length 4.4 Adjus
t=    5.5s x=   22.5m v0, a Tesla, with length 4.4 Adjus
t=    5.5s x=   22.5m v0, a Tesla, with length 4.4 Adjus
t=    8.0s x=    0.0m v6, a Family Car, with length 3.7
t
t=    8.0s x=    0.7m v6, a Family Car, with length 3.7
t=    8.0s x=    0.7m v6, a Family Car, with length 3.7
t
```

Fig. 3: Vehicle lengths

looking at the simulation data. Visit Table 1 to see the list of modifications. The *vtype* attribute will help put context to the scenarios we see, for instance, is it normal for a HGV to be faster than a Tesla?, a situation like this needs further attention.

The logic that controls the vehicle behaviour when in motion is left alone except for a modification in the distance from front, which is the distance between a vehicle and the vehicle in front of it. This value was set at 200m. We found that reducing this distance to 150m reduced the number of crashes. There is a **surround class** that monitors what vehicles are present to the *left, right, front, back, right front, left front, right back, left back* positions of a vehicle. This class ensures all positions, especially, the positions in the direction of the vehicle are free in the scenario where a vehicle wants to switch lanes.

In this paper, we generate 500-2000 vehicles for our simulations in steps of 500. We believe this numbers reflect real life scenarios.

### B. Model for Motorway

**Lane generation analysis:** We generate three lane segments according to the TCD proposal of having a 3km stretch of lane with a 2km 3 lane highway merging into a 1km 2 lane highway segment.

We have a *left lane*, a *right lane* and an *extra lane*. Since we have 2km of 3 lanes, we build the *extra lane* segment with length of 1.7km as seen in Figure 4, and add a *merge lane* segment of 300m to be merged with the left lane. The *left lane* is built with a length of 1.7km and a 300m segment that is merged with the *extra lane* and lastly, a 1km stretch as part of the 2 lane in our simulation. The *right lane* is simply copied from the *left lane* with the *widenRight()* function. With this, our lane generation is complete. The link from the 300m left segment is set to *'None'* on the left so that cars on the *left lane* do not drive to the *merged lane*. Fig 4 shows our lane. The lane class does not need to be modified for our purpose.

### C. Traffic Generation - simul()

When our vehicle and lane objects are ready, we run a simulate function, written as *simul()* to pass in variables for simulation. The function takes in number of cars (N), the inter-arrival time of the cars on the freeway, the speed limit, and the percentage of vehicle types (get ratio of HGV to EV to urban vehicles). In result, we get a data frame, Figure 5, that

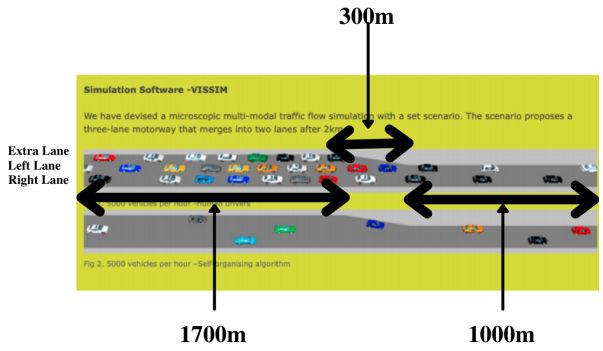


Fig. 4: Lane generation

shows the *density* in *'veh/km'*, *time taken* in *'hours'*, average speed for vehicles in *'km/h'* and *traffic volume* in *'veh/hr'*, crash summary and more.

For this research, we take a small, focused but broad range of parameters to test within the simulation environment. Some assumptions we make are;

- Vehicles are at full or near full manufacturer operating specifications. We assume the vehicles are functioning at top form and are not in need of services or repairs
- At least 15% of the vehicles on the high way are HGVs and 10% are EVs

This 'N' is the total number of vehicles to generate and run during a simulation. We divided this N into two, 75% of N goes to vehicles starting on the *'l'* and *'r'* lanes, while the rest goes to the *extra lane*. The *r lane* is set as the entry lane for all HGVs. This is done to reduce collision with vehicles coming from the extra lane.

### *D. Data Collection*

The simulation keeps track of everything. From tracking crashes, emergency braking, overtaking to changing lanes and more.

- **Recorder class:** The recorder class in our code base keeps track of all events happening within the simulation. From this, we are able to see a vehicle's position on a lane, the velocity, its current activity at a particular time, when it overtakes, when it applies emergency braking and/or when it crashes. This data is used to tweak the parameters of our simulation model in other to get answers. Increasing or decreasing the distance between vehicles could affect the rate of crashes and setting certain vehicles to always begin on a particular lane could also reduce the rate of crashing.
- **Density, average speed, flow:** Our simulation model can tell us the rate of vehicles per kilometer of lane, the average speed of the vehicles and the volume of the vehicles. This data combined with the number of cars and the active speed limit can be used to determine the optimal flow of traffic based on parameters we input into the simulation. For this project, we have created a table to be returned which has values for traffic density, traffic volume, total travelling time, and we introduced, 'no_of_crashes', 'no_of_emergency_braking' to the mix to get a feel of what is actually happening on the highway. Too many emergency braking tells us there is an inevitable crash and that this driving experience will not be smooth for the drivers. An example of this is shown in fig xyz where we have simulation 0 with a high traffic flow at 4498 veh/hr but also high emergency braking at 42 for the tracked duration.
To calculate the number of crashes and emergency braking events in our simulation, we simply take the length of events labelled as 'crash' and 'brake' respectively. (Kindly refer to segment 'Simulation' in accompanying code documentation). To get the total_volume, we run a for loop on each lane in our simulation and call the *rec.flow()* function available in our code base. To get the average total travel time, the *rec.avgTravelTime()* function is called on all the lanes and their values summed up. The traffic density (or traffic flow) based on the fundamental law is derived by multiplying the density of each lane, gotten from the function, *rec.density()* by the average speed of the vehicles, gotten from, *rec.avgSpeed()* in the lane. Lastly, we derive our average speed for each vehicle by summing the average speeds, *rec.avgSpeed()*, for each lane and dividing by the number of lanes.

## CODE FORMAT/STYLE

We use a random seed of 16 to obtain the same result over the course of our simulations. Our simulation function, *simul()* is called with varying parameters. A summary of the tested parameters is itemized below. Our simulation function takes in the number of vehicles, an inter-arrival time, the speed limit, and a percentage of HGV to Electronic to Passenger vehicles, i.e. 15% HGV, 10% EV, and the rest as passenger vehicles. The function returns a table of values showing the total density, the total volume and the total time travelled by the vehicles. We also take note of events like crashes, emergency braking, and overtaking to get an idea of what the driving experience will be.

We take a wide simulation scope and test our simulation for the following inputs;

- Speed limit - 60km/h, 80km/h, 100km/h, and 120km/h
- Inter arrival time - 5, 10, 15 seconds
- Number of vehicles - 500, 1000, 1500, 2000
- Vehicle type rate - HGV, EV, Urban vehicles

## EVALUATION

To make a recommendation on speed limit, this report will include details on the optimal number of vehicles, the IAT and what the traffic volume will be. If we look at the emergency braking column, we see it is one of the highest on the simulation table. This volume is not ideal as it is slow and involves a high crash probability.

We see from Table 2 that the simulation with the highest vehicle flow, derived from our *rec.flow()* function, is simulation-29, it has 2000 vehicles at a speed limit of 80km/h in the lanes, and an average total travelling time of 261s. When we look at the q value however, the hourly flow rate, derived from the fundamental law of *traffic density * lane mean speed*, we see that the flow rate of this simulation is low. We also record 2 crashes in this scenario. We definitely want to recommend a scenario with no crashes and as low causes for emergency braking as possible so we will look for another favorable situation. A graph of simulation-29 is presented in Figure 5. You will notice the emergency braking events in round red ink and the 'X' marks crash events. The Plot attribute in the simul() function we have added to the code can take a 'True' value to display the plot relating to the scenario.

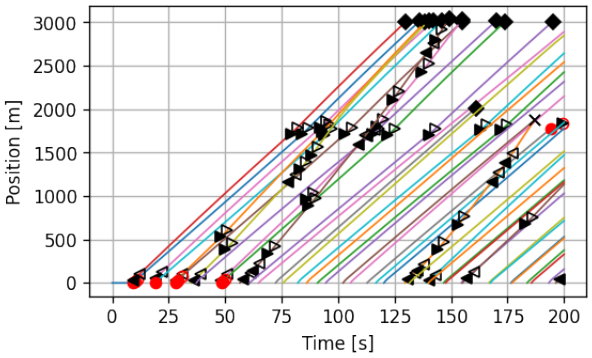


Fig. 5: Simulation 29

We take a closer look at simulation-3. There are 500 vehicles in this simulation with a q value of 3148 veh/hr running at an average speed of 122km/h, slightly over the speed limit. The average total travelling time for vehicles is 235 seconds, and we have no crashes, only 4 emergency braking events. We take a further look at this scenarios graph. Simulation 3 looks like an ideal scenario with speed limits to recommend, but when you take the number of vehicles in this scenario into account, it might not be able to reflect real life scenarios which has thousands of vehicles passing a highway each day. Due to this, we will drop simulation-3. Figure 6 contains the plot of simulation-3.

The next simulation we will observe closely is simulation-28. It has 2000 vehicles, a q value of 2837.32 veh/hr, an average total travelling time of 282.43 seconds, no crash event and 3 emergency brake events. Note that if were to use the traffic flow, derived by the *rec.flow()* function as a deciding factor, this would still be the simulation that will be recommended as other simulations before it either have crash events or the number of vehicles is lesser. Figure 7 contains the plot for simulation-28.

## IV. Conclusion

In this paper, we have carried out a simulation based on the self organizing network environment proposed by TCD. We used the modified code base in [1] to achieve our objective

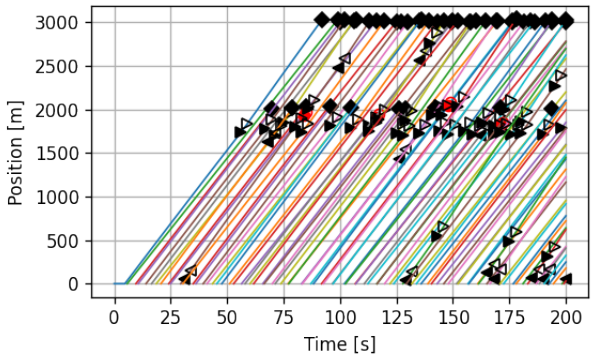


Fig. 6: Simulation 3

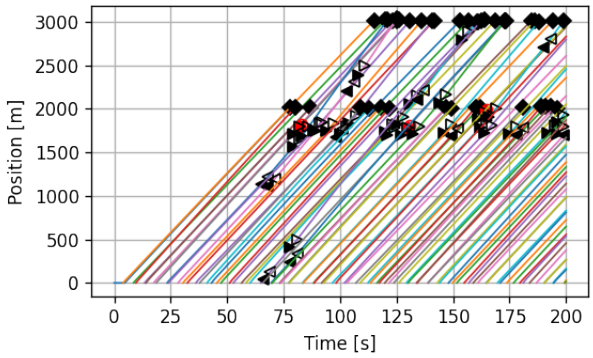


Fig. 7: Simulation 28

of deriving traffic density, average mean speed, the total travelling time and the flow rate of the lanes we generated for the simulation. We found that at a speed limit of 100km/h, the 3km stretch of lane will take over 2500 veh/hr moving with an average of 122km/hr for a total travelling time less than 300 seconds. According to our simulation, we expect up to 3 emergency braking scenarios and no crashes. Our recommendation is to set the speed limit at 100km/hr.

## V. Future work

In the future, we will like to conduct further analysis into our simulation result to see the mix of vehicles that will give the highway a better free flow. If we find that a large amount of electronic vehicles have a correlating effect with free flow, we could recommend to the city to implement adoption of electronic vehicles as a means to aid traffic congestion.

## VI. TABLES

| Class | Method | Comment | Purpose |
|---|---|---|---|
| Vehicle | self | Added static values. | Simulate real life vehicles with differing attributes |
| Vehicle | self | Changed self.far_away_in_front from 200 to 150 | This was found to reduce the crash events |
| Vehicle | trace | Added vehicle type | Aids understanding of the vehicle events |
| Vehicle | recordCrash | Added vehicle type | Aids understanding of the vehicle events |
| Vehicle | emergencyBraking | Added vehicle type | Aids understanding of the vehicle events |
| Recorder | avgTravelTime | Changed mean function to use harmonic mean | Harmonic mean is used when values have different units |
| Recorder | avgSpeed | Added if-else clause | To manage floating error derived when avgTravelTime is zero |

TABLE I: Code base modification table

| Vehicle count | IAT(s) | Speed limit (km/h) | flow rate q (veh/hr) | speed u (km/h) | flow rate (veh/hr) | Total time (sec) | Crashes | Emergency braking |
|---|---|---|---|---|---|---|---|---|
| 500 | 5 | 60 | 2020.37 | 61.28 | 4498.76 | 469.6 | 0 | 42 |
| 500 | 5 | 80 | 2408 | 78.71 | 3957.78 | 360.3 | 18 | 27 |
| 500 | 5 | 100 | 2758.12 | 101.25 | 4073.66 | 284.9 | 7 | 8 |
| 500 | 5 | 120 | 3148.82 | 122.84 | 4170.65 | 235.81 | 0 | 4 |
| 500 | 10 | 60 | 1021.25 | 61.59 | 2452.25 | 468.26 | 0 | 4 |
| 500 | 10 | 80 | 1229.62 | 77.42 | 2230.77 | 261.47 | 5 | 10 |
| 500 | 10 | 100 | 1421.92 | 80.2 | 2193.17 | 205.07 | 0 | 2 |
| 500 | 10 | 120 | 1584 | 94.91 | 2574.8 | 173.41 | 0 | 1 |
| 1000 | 5 | 60 | 1984.06 | 60.19 | 3960.78 | 479.46 | 0 | 49 |
| 1000 | 5 | 80 | 2494.46 | 81.09 | 4109.34 | 355.26 | 0 | 0 |
| 1000 | 5 | 100 | 2781.01 | 104.73 | 4043.35 | 275.89 | 3 | 5 |
| 1000 | 5 | 120 | 2681.72 | 124.61 | 4380.86 | 231.26 | 23 | 12 |
| 1000 | 10 | 60 | 991.39 | 63.07 | 1825.89 | 456.67 | 0 | 0 |
| 1000 | 10 | 80 | 1261.84 | 92.26 | 2091.38 | 320.69 | 0 | 1 |
| 1000 | 10 | 100 | 1431.73 | 80.8 | 2149.66 | 203.04 | 0 | 0 |
| 1000 | 10 | 120 | 1596.28 | 116.14 | 3455.58 | 175.33 | 0 | 1 |
| 1500 | 5 | 60 | 2006.44 | 61.74 | 3726.1 | 465.39 | 0 | 39 |
| 1500 | 5 | 80 | 2513.19 | 82.02 | 3642.88 | 353.85 | 7 | 11 |
| 1500 | 5 | 100 | 1391.18 | 49.36 | 1725.45 | 317.17 | 49 | 54 |
| 1500 | 5 | 120 | 1995.1 | 93.49 | 2136.81 | 262.9 | 47 | 57 |
| 1500 | 10 | 60 | 921.43 | 37.99 | 1438.41 | 294.77 | 8 | 21 |
| 1500 | 10 | 80 | 1174.23 | 67.33 | 1486.15 | 311.2 | 3 | 11 |
| 1500 | 10 | 100 | 1155.37 | 56.42 | 1282.9 | 190.95 | 15 | 16 |
| 1500 | 10 | 120 | 1503.56 | 94.93 | 2236.39 | 156.51 | 0 | 0 |
| 2000 | 5 | 60 | 1971.11 | 61.73 | 4092.02 | 468.92 | 2 | 50 |
| 2000 | 5 | 80 | 2513.59 | 81.32 | 4241.42 | 355.67 | 0 | 2 |
| 2000 | 5 | 100 | 2837.31 | 102.75 | 4183.74 | 282.43 | 0 | 3 |
| 2000 | 5 | 120 | 3065.32 | 108.72 | 3742.85 | 223.49 | 16 | 19 |
| 2000 | 10 | 60 | 977.51 | 48.34 | 1755.2 | 338.89 | 0 | 4 |
| 2000 | 10 | 80 | 1289.89 | 80.91 | 8724.84 | 261.85 | 2 | 8 |
| 2000 | 10 | 100 | 927.44 | 68.47 | 1031.73 | 259.83 | 16 | 21 |
| 2000 | 10 | 120 | 1601.92 | 116.73 | 2082.96 | 176.87 | 0 | 1 |

TABLE II: Simulation results

## References


[1] Chris Horton. *Multilane Mixed Simulation 6, Traffic*. https://mymoodle.ncirl.ie/mod/resource/view.php?id=95318, Last accessed on 2021-08-13.

[2] National Institutes of Health, USA. *Computer Modelling and Simulation*. https://ors.od.nih.gov/OD/OQM/cms/Pages/default.aspx, Last accessed on 2021-08-13.

[3] Azura Che Soh et al. "A discrete-event traffic simulation model for multilane-multiple intersection". In: *2013 9th Asian Control Conference (ASCC)*. 2013, pp. 1–7. DOI: 10.1109/ASCC.2013.6606228.

[4] Jin Xu, Xiao Luo, and Yi-Ming Shao. "Vehicle trajectory at curved sections of two-lane mountain roads: a field study under natural driving conditions". In: *European Transport Research Review* 10.1 (Jan. 2018), p. 12. ISSN: 1866-8887. DOI: 10.1007/s12544-018-0284-x. URL: https://doi.org/10.1007/s12544-018-0284-x.