Simulation of Traffic Flow and Congestion on a Circular Road

Xinyu Li

xl3665@nyu.edu

December 22, 2022

1 Introduction

In this project, we study the traffic jam on a circular road using the model we discussed in class. In particular, we are interested in the formation and evolution of traffic congestion on a circle. What makes this question interesting is that the road is circular, which means the end of the car queue will be affected by the head. Intuitively, this may leads to some interesting phenomena such as "a looping snake of traffic jams".

The project will be divided into three parts. First, I will theoretically construct the system of differential equations that govern the traffic. I will study the behavior of the system with respect to change in parameter and time. Next, I will use backward-Euler method to numerically solve the differential equations. Finally, I will conduct experiments on numerical stability and traffic jams under different parameters and initial conditions.

2 Equations

2.1 Assumption

To simplify the theoretical and numerical analysis, we will make the following assumptions:

Assumption A1. *All vehicles are driving on a straight line segment whose start and end are glued together.*

Justification. Theoretically, A1 is reasonable because unit circle $S^1 = \{x \in R^2 \mid \|x\| = 1\}$ is isomorphic to the topological space $\mathbb{R}/\mathbb{Z}$. Intuitively, for a vehicle driving on a circle, we can always take its velocity along the tangent line as its velocity on the line segment. This is illustrated by figure 1.

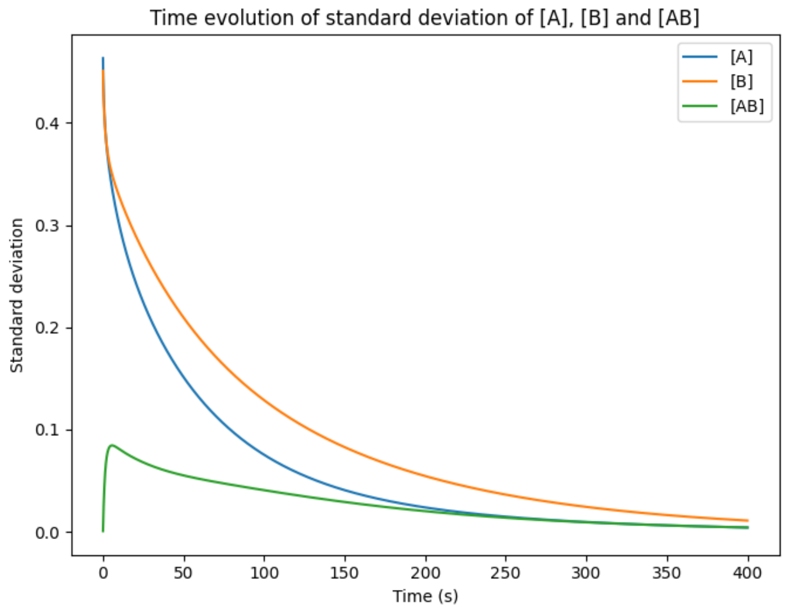


Figure 1: a mapping from a line segment to a circle

$\square$

Assumption A2. *A vehicle is seen as a small, moving line segment. It can be graphically represented by the middle point x of the line.*

Assumption A3. *All vehicles have the same parameters (length, maximal velocity, etc). Each vehicle has*

Justification. This unrealistic assumption can greatly simplify our theoretical and numerical analysis. And it won't affect the formation of traffic jams in an essential way. □

2.2 Equation

Let the road be a line segment of length D meters. Initially, there are N vehicles with zero initial velocity. Each vehicle is represented by $c_n, n = 0, ..., N-1$. We denote the location of vehicle c_n as $x_n = x_n(t)$, its velocity as $v_n = v_n(t)$, its distance to the vehicle in front as $d_n = d_n(t) = x_n - x_{(n+1)\%N}$. The subscript $(n+1)\%N$ is used so that $d_{N-1} = x_{N-1} - x_0$, as is should be on a circular road. Using the model we discussed in the lecture, we have the following system of differential equations

$$\begin{cases} \frac{d}{dt}x_n(t) = v_n(t) \\ \\ \frac{d}{dt}v_n(t) = \frac{1}{\tau}(ov(d_n) - v_n(t)) \\ \\ x_n(0) = x_n^{(0)} \\ \\ v_n(0) = 0 & n = 0, ..., N-1 \end{cases} \tag{1}$$

where $\tau \in [0, \infty)$.

The function $ov(d)$ is the optimal velocity with respect to distance. It is defined by three non-negative parameters: $v_{max}, d_{min}, d_{max}$

$$ov(d) = \begin{cases} 0 & \text{if } d \leq d_{min} \\ v_{max} * \frac{log(d/d_{min})}{log(d_{max}/d_{min}}) & \text{if } d_{min} < d < d_{max} \\ v_{max} & \text{if } d \geq d_{max} \end{cases}$$

$ov(d)$, as the name suggest, is the optimal velocity. The velocity is adjusted based on the distance from a vehicle to the vehicle in front. If the velocity of all vehicles match that given by $ov(d)$, traffic jams will be alleviated within the minimal amount of time. (We will check this statement numerically in Section 5. For now, let's take it as a fact).

2.3 Meaning of the Equations and Parameter τ

The system of differential equation is essentially saying that all drivers are trying to drive their vehicles according to the optimal velocity. But they can't because of factors like inertia and reaction time. These factors are aggregated into the parameter τ. τ represents the level of deviation from optimal velocity. The smaller τ is, the less deviated the actual velocity is from optimal velocity. And when $\tau = 0$, the velocity will be the same as optimal velocity.

To justify this, we can assume that optimal velocity $ov(d) = ov(d(t)) = ov(t)$ is a known function of time[1]. This allow us to solve the system directly. Under this assumption, we have

$$\begin{cases} \frac{d}{dt}v(t) = \frac{1}{\tau}(ov(t) - v(t)) \\ \\ v(0) = 0 \end{cases} \tag{2}$$

Using the method of integral factor, we have the following solution

$$v(t) = e^{-\frac{1}{\tau}t} \int_0^t \frac{1}{\tau} e^{\frac{1}{\tau}s} ov(s)\ ds$$

Let $K_\tau(s) = \frac{1}{\tau}e^{-\frac{1}{\tau}s}$ be a family of kernel parametrized by τ, then we can rewrite the solution as a convolution[2] between K and ov.

[1] Note that this assumption is unrealistic for simulation because it ignores d_n, i.e. x_n and $x_{(n+1)\%N}$. The complexity of this system lies precisely in the fact that dv_n/dt also depends on the vehicle in front.

[2] Strictly speaking, the convolution is defined on $L^2([0,t])$ (Lebesgue space) for fixed t.

$$\begin{aligned} v(t) &= e^{-\frac{1}{\tau}t} \int_0^t \frac{1}{\tau} e^{\frac{1}{\tau}s} ov(s)\ ds \\ &= \int_0^t \frac{1}{\tau} e^{-\frac{1}{\tau}(t-s)} ov(s)\ ds \\ &= \int_0^t K_\tau(t-s) ov(s)\ ds \\ &= (K_\tau * ov)(t) \end{aligned}$$

Loosely speaking, as $\tau \to 0$, the kernel $K_\tau \to \delta_t$ where $\delta_t = \delta(x-t)$ is the Dirac delta function $\delta(x)$ translated by t. Using some black magic from Analysis[3], we can move the limit inside the convolution. Since the convolution of a function with Dirac delta function evaluated at t is exactly the value of the function at t, we have

$$\lim_{\tau\to 0} v(t) = \lim_{\tau \to 0}(K_\tau * ov)(t) = ((\lim_{\tau\to 0} K_\tau) * ov)(t) = (\delta_t * ov)(t) = ov(t)$$

Moreover, by first moving τ to the left hand side of the differential equation 2, then taking $\tau = 0$, we can also derive that $v(t) = ov(t)$ when $\tau = 0$.

3 Numerical Method

To solve the system 1 numerically, we will use forward-Euler method to update x_n, and backward-Euler method to update v_n. Compared to forward-Euler method, backward-Euler method allows us to take τ arbitrarily close to 0 without the need to adjust time step to a very small number. This comes in handy to simulate the limiting behavior stated in Section 2.3.

Let Δt be time step, T be the duration of simulation. Then, the system 1 can be discretized into

$$\begin{cases} x_n(t+\Delta t) = x_n(t) + v_n(t)\cdot \Delta t \\ v_n(t+\Delta t) = \frac{(\Delta t \cdot ov(d_n) + \tau_n(t))}{\Delta t + \tau} \\ x_n(0) = x_n^{(0)} \\ v_n(0) = 0 \qquad\qquad n = 0, ..., N-1 \end{cases} \tag{3}$$

where the the update of v_n is derived from the discretization

$$\frac{v_n(t+\Delta t) - v_n(t)}{\Delta t} = \frac{ov(d_n) - v(t+\Delta t)}{\tau}$$

For consistence, we will specify a set of default parameters. Unless stated otherwise, simulations conducted in the sections below will use these default parameters.

We conduct our simulation on a circular road of length $D = 1000m$ for $T = 1000s$ with $N = 30$ cars on it. We assume all vehicles have length $l = 4.5m$. We update the velocity using $d_{min} = 0.2 + 3l\ m$ and $d_{max} = 100 + 3l\ m$. The maximal velocity is $v_{max} = 120\ km/h$, and $\tau = 0.5$. The simulation uses a default time step of $\Delta t = 0.1$.

4 Validation

To show that our simulation is independent of time step, we can run the simulation using different Δt and compare the difference between results. One way to characterize the difference is to compare the velocity in the end of the simulation.

[3]One rigorous way to is to define K_τ and δ as distributions, and talk about the convolution between a distribution and a function, but this rigorousness won't add much insight to our analysis, though it would be an interesting topic in itself.

Let $v^{\Delta t_k}(T)$ be the simulated velocity at time T using time step Δt_k (wlog, we assume Δt_k is increasing with respect to k). Then $v^{\Delta t_0}$ is the most accurate result because Δt_k is the smallest time step, and $e_k = \|v^{\Delta t_k} - v^{\Delta t_0}\|$, the deviation from $v^{\Delta t_k}$, can represent the accuracy of the simulation using time step Δt_k.

Below are the results for $\Delta t \in \{0.0025, 0.005, 0.01, 0.025, 0.05\}$.

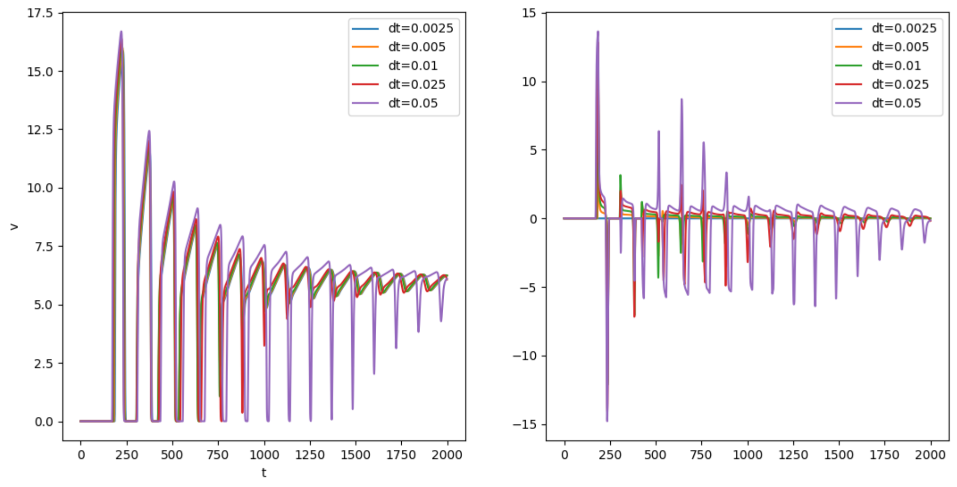


Figure 2: The left figure is the v-t relation under different time step. The right figure is the relation between $v^{\Delta t_k}(t) - v^{\Delta t_0}(t)$ under different time step.

The relation between Δt_k and e_k is illustrated in Table 1

Δt_k	e_k
0.0025	0.0000
0.005	0.0134
0.01	0.0235
0.025	0.0322
0.05	0.0510

Table 1: Relation between Δt_k and e_k

Judging from the figures and table, we can see that $\Delta t \leq 0.025$ shows no significance differnece from $\Delta t = 0.0001$, which means our default choice of time step is valid.

5 Results and Discussion

Before the experiment, let's discuss one important diagram: x-t plot.

x-t plot visualize the $x_n(t)$ function for each n[4]. The intersection between the graph of $x_n(t)$ and a vertical line $t = t_1$ shows the location of vehicle c_n at $t = t_1$. Figure 3 gives an example. When the distance between several graph are very close and their slopes are close to 0, it indicates these vehicles are in a congestion, as the vehicles in the red rectangle in figure 3.

[4]For visualization purpose, what we plot is not the absolute location but the mileage

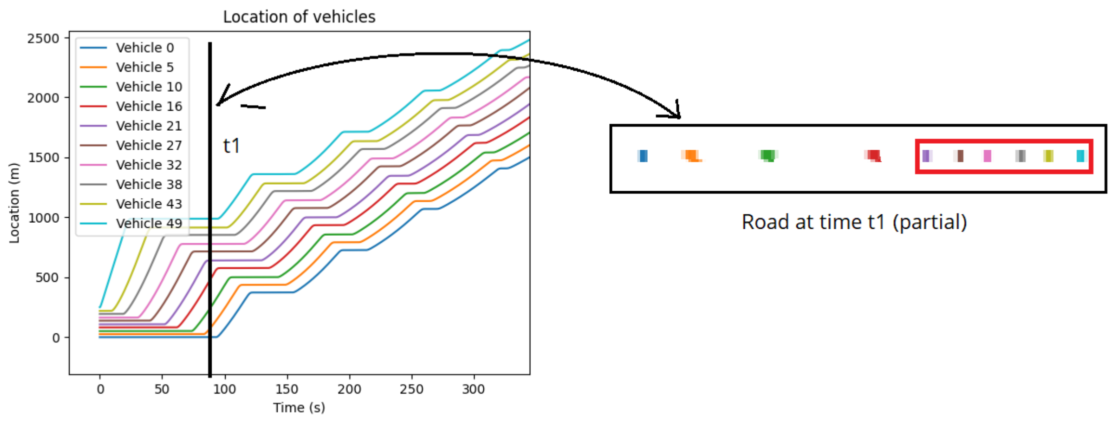


Figure 3: Correspondence between the x-t plot of a vehicle and it's actual location. We learnt this method of visualization from [Boyles(2018)]

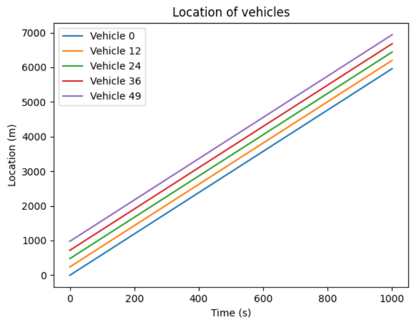
5.1 Experiment 1: Equidistant Equilibrium and its Stability (theoretically and numerically)

Consider using an equidistant initialization where $d_i(0) = d_j(0) = d_e$ for any $0 \leq i, j \leq N-1$. Theoretically, After a short period of time (when all vehicles speed up), the system should reach an equilibrium where all their velocities are equal to $v_e = ov(d_e)$, and constant with respect to time. The reasons are twofold.

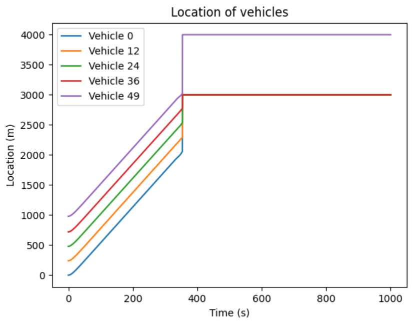
1. First, the velocity $v_n(t)$ are equal to one another at any time t. This is because, initially, all vehicles have the same $d_n(0)$ and $v_n(0)$. Since the acceleration (update to velocity v_n) is determined by $v_n(t)$ and $d_n(0)$, all vehicles should update their velocities in the same way.
2. Moreover, by definition of dv/dt, we know that the velocity v_n will tend to $v_e = ov(d_e)$ at t increases ($v_n(t)$ is an increasing function with a least upper bound v_e), and whenever it reaches v_e, it will stay at v_e because $dv/dt = 0$ when $v(t) = v_e$.

Note that this property should hold for arbitrary τ in theory. But can our simulation reproduce this equilibrium for arbitrary τ?

To see this, we conduct simulations for $\tau \in \{0.1, 0.2, 0.5, 1, 5, 10\}$. We find out that car crash happens for $tau = 5, 10$. Figure 4 shows two examples.



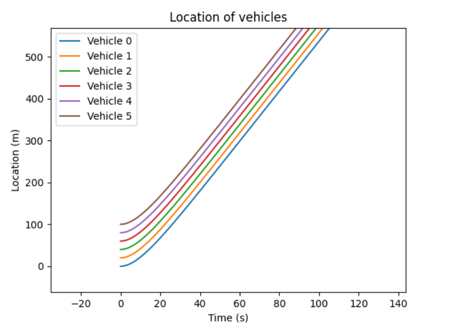
(a) $\tau = 0.5$



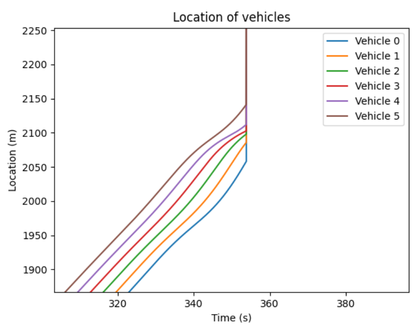
(b) $\tau = 10.0$

Figure 4: Simulation on the left has no car crash; The one on the write has a car crash

Our first claim that velocity of all vehicles are equal is justified by all simulations whether a car crash happens or not. From figure 5(a), we can see that at any time t, the velocity of all vehicles



(a) $\tau = 10.0$, speed up to v_e at the beginning



(b) $\tau = 10.0$, car crash

Figure 5: The left figure shows the start of the simulation, where all vehicles have the same velocity and speeds up to an equilibrium velocity v_e; The right figure shows the car crash between vehicles c_2 and c_3.

(the slope of the tangent line to the curve at t) are equal. Moreover, we can clearly observe a process of "speeding up to a fixed velocity v_e" from figure 5(a).

However, the result in figure 5(b) isn't aligned with our second claim.

One possible reason is that the equilibrium gets more unstable as τ increases. For a large τ, the numerical errors accumulate as time proceeds, and break the equilibrium in the end. We also observe that $\tau = 1.25$ apprears to be a critical value: $\tau \leq 1.25$ has no car crash; $\tau > 1.25$ has car crash, and the farther away τ is to 1, the sooner a car crash will happen. This is demonstrated in table 2.

τ	T_{crash}(s)
0.1	$>$20000
0.5	$>$20000
0.9	$>$20000
1.25	$>$20000
1.2505	2424.25
1.255	1240.32
1.26	930.89
1.28	703.04
1.3	615.36
1.4	512.58
1.5	519.26
2.0	361.22
5.0	318.31
10.0	353.88

Table 2: This table shows the exact time when a car crash apprears. As

Moreover, for $\tau < 1.25$, the closer τ is to 1.25, the sooner the equi-distant queue of vehicles will degenerate into congestion. Figure 6 gives an example

We observe the same phenomenon with the same critical value 1.25 for different d_{min}, d_{max} and v_{max}.

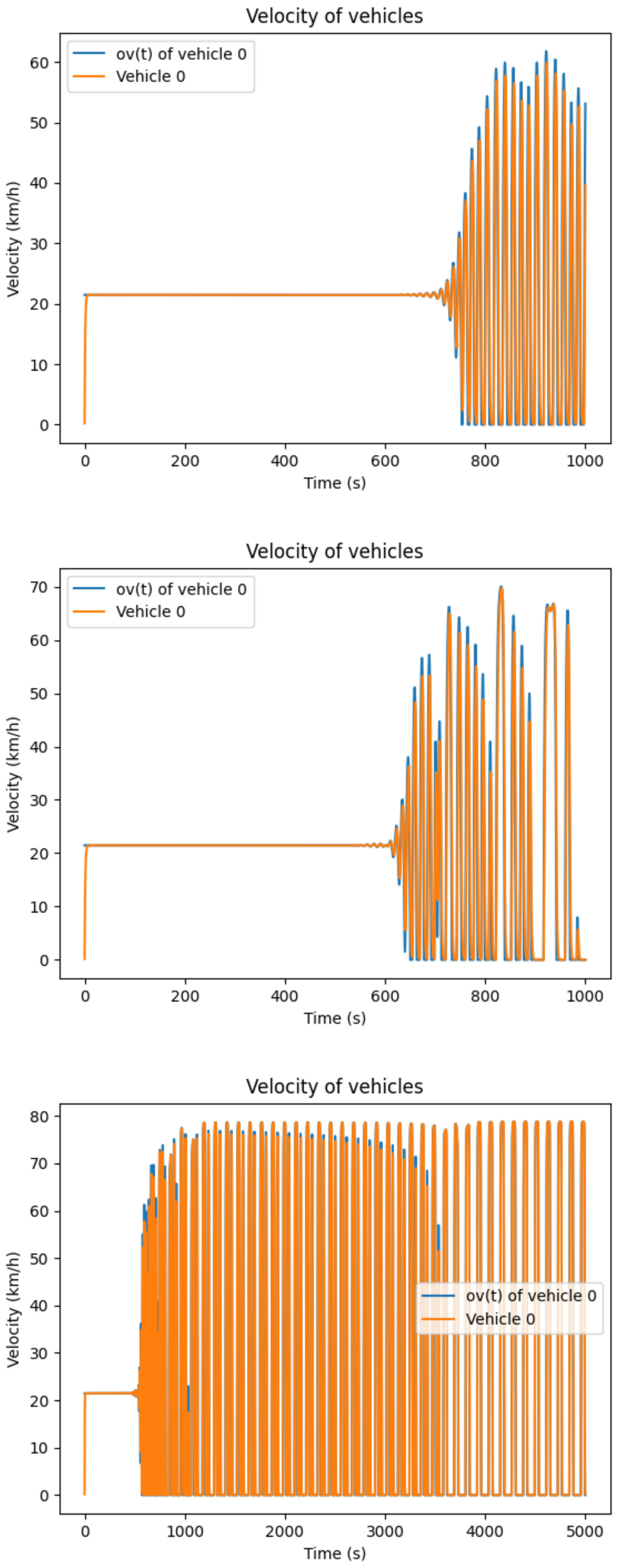


Figure 6: A comparison between $\tau = 0.11$(above), 0.12(middle) and 0.125(below). Severe oscillation of velocity indicates the degeneration to congestion

5.2 Experiment 2: Traffic Snake

Traffic congestion has a funny nickname: **traffic snake**. This is because whenever there is traffic congestion, there will always be a consecutive queue of vehicles that are stuck; and, depending on the density and velocity of the car flow, this queue will either grow or shrink.

To put it in a more rigorous way, for certain initial condition, the system of differential equations that models a traffic flow will have a shock-wave solution. The snake is the shock wave: the snake moves as the shock wave propagates; the snake grows/shrink as the shock wave grows/shrink.

We observe this phenomenon in our simulation as well. The eaisest way to generate a traffic snake is to initialize vehicles really colse to one another and set the road length to an appropriate value. This way, the vehicle that leaves first will finish one loop so fast that, when it meets the last vehicle after finishing one loop, it will meet a queue of vehicles that haven't even started yet. This is how a typical traffic congestion starts on a circular road. For a demonstration, see this video: `xinyu-li-123.github.io/videos/congestion.mp4`

In specific, we initialize the vehicle such that $d_n = l + 0.5$ for $n < N - 1$ where l is the car length. This is to say each vehicle has a gap of $0.5m$ initially (ignoring the distance from c_{N-1} to c_0).

Using this initialization, we can also observe the shock wave pattern from the x-t relation of the vehicles. Figure 7 shows the result of the simulation using default parameters.

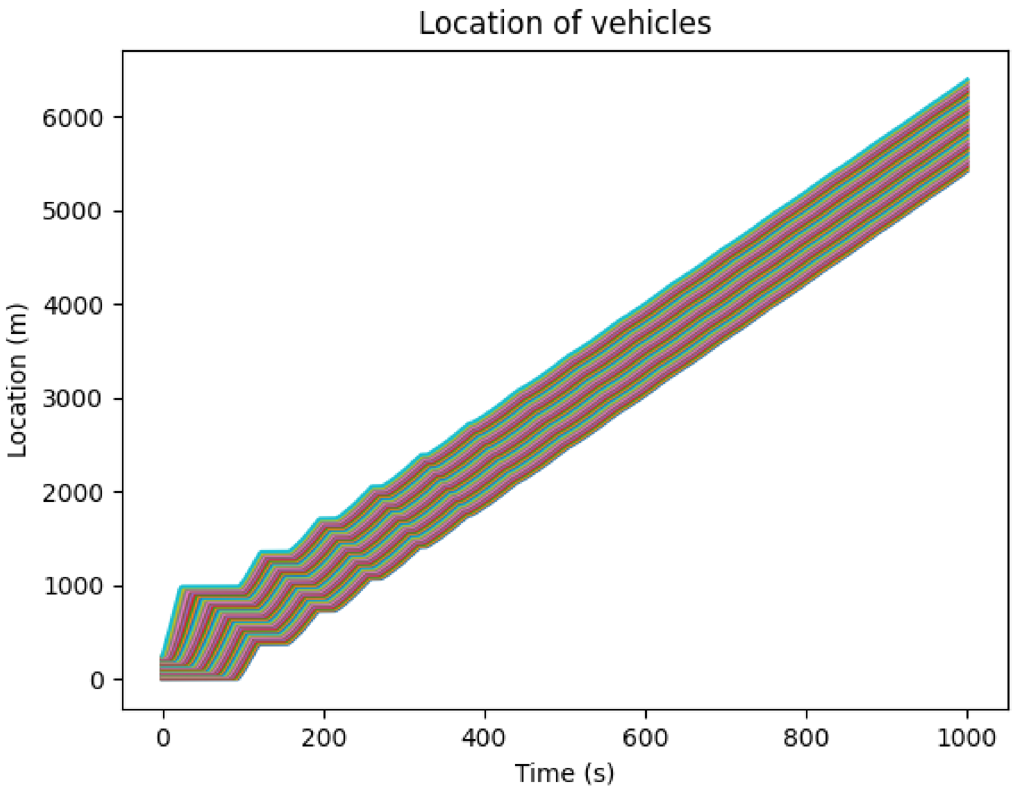


Figure 7: The x-t relation of all vehicles

Figure 9 demonstrate a zoomed-in version of Figure 7. From these figures, we have three observations:

1. First, there is traffic congestion. Examples are the region in the red quadrilaterals.
2. Second, the traffic congestion is alleviated as time proceeds, thanks to our model. In figure 7 (a), we can see that the duration of a vehicle in the congestion region 2 (the red quadrilateral

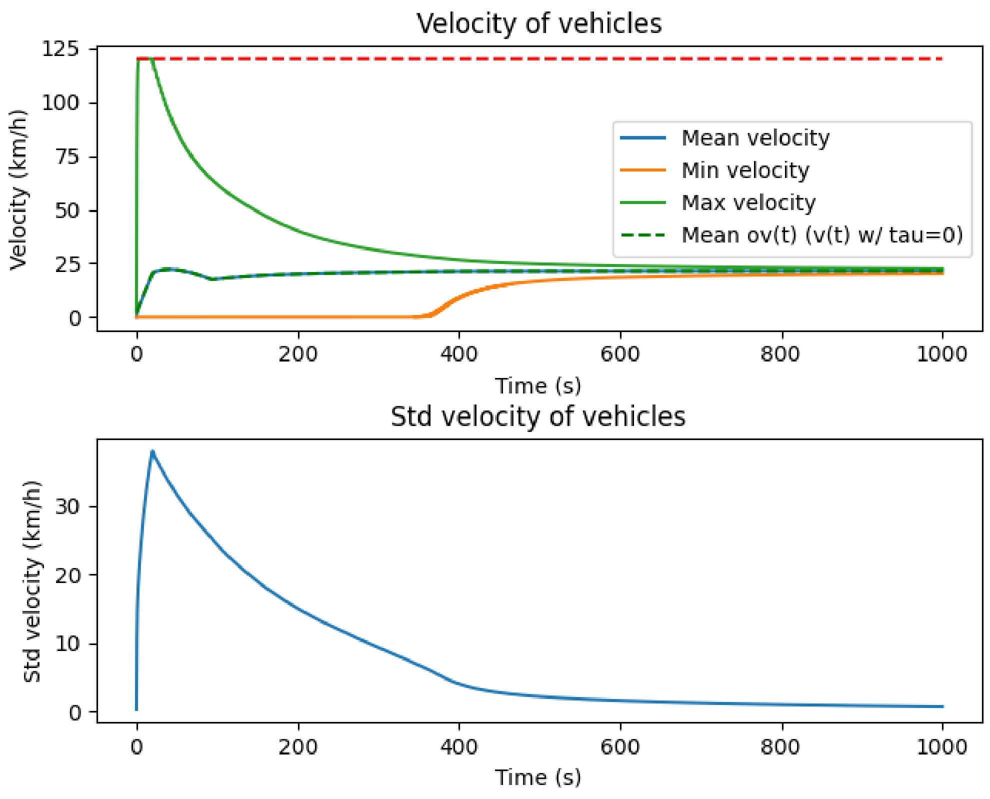


Figure 8: The evolution of mean/min/max velocity

#2) is shorter than in the congestion region 1. Moreover, the traffic congetion is entirely eliminated in figure 7 (b), which is in accord to our theory in Section 2.3

3. Lastly, the velocity in the equilibrium is smaller than the maximal velocity, which is an aftermath of congestion on a circle. This drop of velocity is demonstrated in figure 8. This might goes against our intuition since normally cars should be able to speed up when the traffic congestion is gone. This is because the vehicles are evenly distributed on the road when the traffic congestion is eliminated. And from the first experiment, we know that this is an equilibrium where the velocity will be constant with respect to time.

6 Summary and Conclusions

In this report, we studied the behavior of traffic flow on a circular road. First, we define the problem. Next, we stated the necessary assumptions and discussed the system of differential equations that governs the traffic flow. We also stated and proved the limiting behavior of the system as $\tau \to 0$. Then, we discretized the system using backward Euler method. After that, we validated that the result of the simulation is independent of time step Δt so long as Δt is small enough. Finally, we conducted two experiments. The first experiment demonstrated the relation between the stability of equilibrium and the parameter τ. Large τ results in unstable equilibrium, and the larger τ is, the more sensitive the equilibrium is to small perturbation; the second experiment showed the behavior of traffic congestion on a circular road: the traffic congestion will be gradually eliminated, and the system will tends to an equi-distant equilibrium. However there will be a permanent drop of velocity.

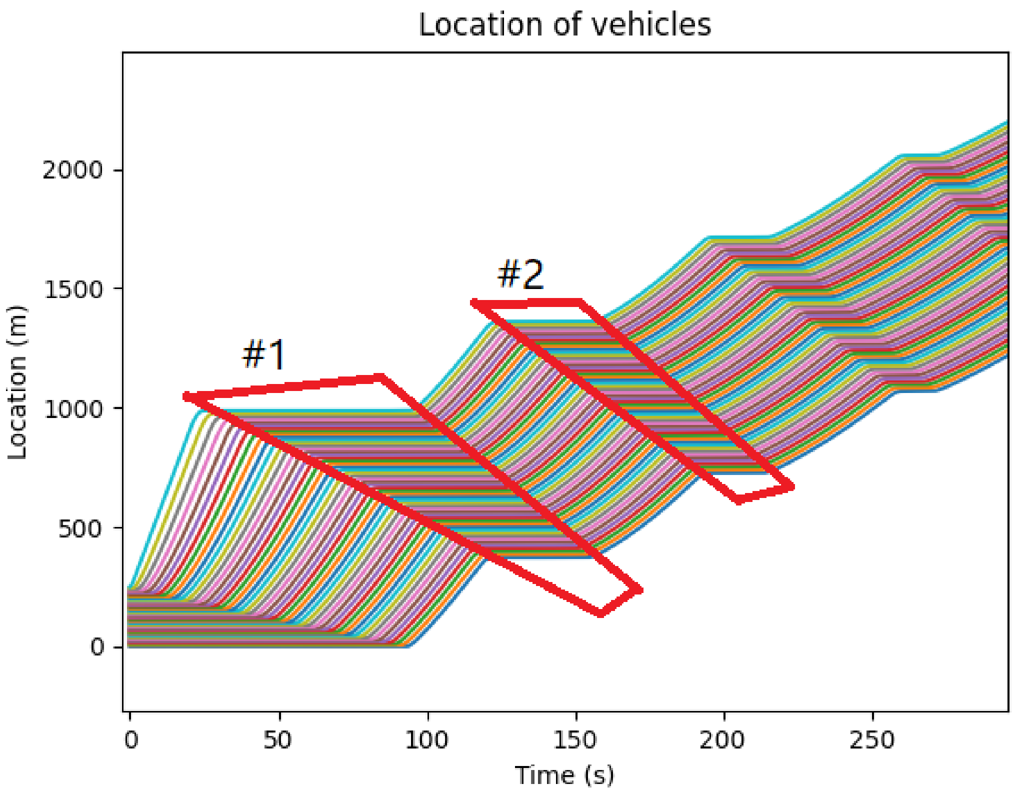
References

[Boyles(2018)] Stephen Boyles. 2018. Lecture slides on LWR model. `https://sboyles.github.io/teaching/ce392d/5-lwrmodel.pdf`

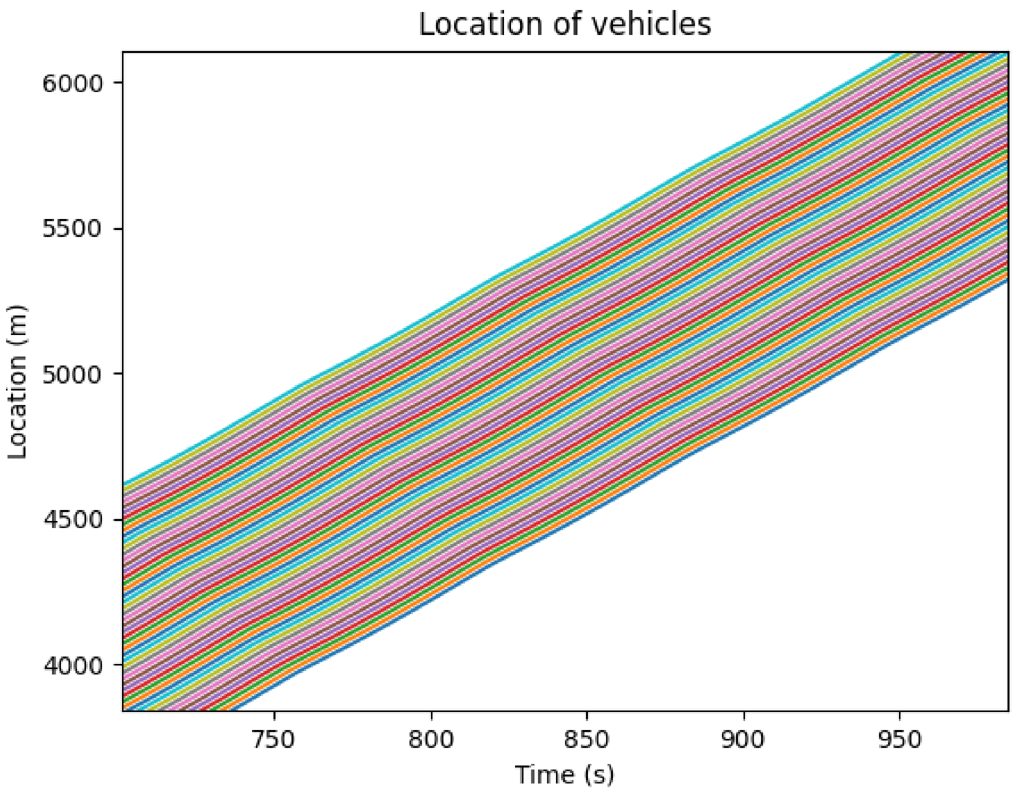
A Code

This appendix presents the code for Euler method that solves the differential equation mentioned in section 2. The code is written in Python using the python packages `numpy` and `scipy` for computation, and `matplotlib` for visualization. For a full version of the code, you may refer to this github repo `https://github.com/Xinyu-Li-123/TrafficSimulation`.

```python
"""
distance-based optimal velocity
"""

from parameter.road import *
from parameter.vehicle import *
from parameter.simulation import *
from parameter.model import dov_param
from utils.utils import ONE_TO_ZERO, log_inf, log_inf_approx

import numpy as np

ov = np.zeros(N)

def _compute_optimal_velocity(d, dov_update_type):
    min_mask = d < dov_param.dmin
    max_mask = d > dov_param.dmax
    mid_mask = np.logical_not(np.logical_or(min_mask, max_mask))
    ov[min_mask] = 0
    if dov_update_type == dov_param.dov_update_types[0]:
        ov[mid_mask] = \
            vmax * log_inf(d[mid_mask] / dov_param.dmin) / \
            log_inf(dov_param.dmax/dov_param.dmin)
    elif dov_update_type == dov_param.dov_update_types[1]:
        # second order approximation of log_inf
        # seems to fail
        ov[mid_mask] = \
            vmax * log_inf_approx(
                d[mid_mask]/dov_param.dmin,
        order=dov_param.dov_update_approx_order) / \
            log_inf(dov_param.dmax / dov_param.dmin)
    elif dov_update_type == dov_param.dov_update_types[2]:
        ov[mid_mask] = \
            vmax * (d[mid_mask] - dov_param.dmin) / \
                (dov_param.dmax - dov_param.dmin)
    else:
        raise ValueError('dov_update_type should be one of the following: {}'.format(
            dov_param.dov_update_types))
    ov[max_mask] = vmax


def dov_update(loc, d, v, a, i, dov_update_type="smoothing"):
    global count
    _compute_optimal_velocity(d, dov_update_type=dov_param.dov_update_type)
# update ov
    loc[:] = (loc + v*dt) % D                  # update loc
    d[:] = (loc[ONE_TO_ZERO] - loc) % D        # update d
    v[:] = (dt * ov + dov_param.tau*v) / (dt + dov_param.tau)
```



(a) Traffic congestion appears the beginning of the simulation.



(b) Traffic congestion is alleviated as time proceeds

Figure 9