Traffic Dynamics Part II

Brian Ma., mab333@uw.edu
BPhys 450, Computational and Theoretical Modeling in Physics
Winter 2021, Dr. Eric P. Salathe
University of Washington, Bothell

Febuary 17, 2021

Abstract

Utilizing the Intelligent Driver Model(IDM) as a base model, traffic dynamics of a linear line of vehicles in varying circumstances and behavior are both mathematically and computationally modeled. Various dynamics of traffic behavior such as varying starting positions, speed, accelerations and more are computationally modeled to simulate various behaviors.

1 Introduction

In a society where vehicle use has become the choice in transportation by a large majority of people, traffic has become a large part of the daily commute. Based on the behavior of drivers within a controlled environment, traffic dynamics can be mathematically modeled in around a leading vehicle and the following cars behind it. Realistically, we are not capable of accurately develop a model of behavior in a driver because of various factors such as experience, comprehension, vision, rage, texting and more, however in a smaller and more controlled system, we are able to a degree, model the behavior of an intelligent driver.

This program is a developed model of the traffic dynamics of a group of cars spaced apart in a straight line and examining the effects of various parameters and is created based on the Intelligent Driver Model (IDM) developed by Martin Treiber, Ansgar Hennecke and Dirk Mhelbing with data from several German freeways with various types of congested traffic. By experimenting with various parameters, we can use the traffic model to solve and create analytical graphs on various types of traffic.

1.1 Mathematical Model

The acceleration of a car in the IDM model is given by:

$$\frac{dv}{dt} = a\left[1 - \left(\frac{v}{v_0}\right)^{\delta} - \left(\frac{s^*(v,\Delta v)}{s}\right)^2\right] \tag{1}$$

where:

$$s^*(v,\Delta v) = s_0 + vT + \frac{v\Delta v}{2\sqrt{ab}} \tag{2}$$

Which compares the "desired" acceleration of a car on a free road ($a[1-(v/v_0)^{\delta}]$) to the braking deceleration induced by the braking term which is based on the distance to the vehicle ahead $(-s^*/s)^2$ to calculate the overall acceleration of the car.

The acceleration term is designed go down to zero when the car approaches the "desired" speed of v_0, and is affected by the constant δ which determines the rate of deceleration as the car approaches the ideal speed. Whereas, the breaking term on the is calculated through the comparison between the "desired dynamical distance" (Equation 2) and the distance s to the preceding car. As the car ahead gets closer, the braking deceleration will compensate for the free acceleration so the total acceleration would be close to zero. Also as s changes throughout the model, s^* will be altered dynamically to correspond to the gap when following other vehicles in flowing traffic. Therefore, constants a, b, T, δ and v_0 represent driver behavior.

Translating the IDM into a Traffic model, Equation 1 is modified into a rate function which then is used to solve for the motions of each individual car.

$$\frac{dx_i}{dt} = v_i \tag{3}$$

For calculating the future position of each car, the calculated position will be based on the current velocity of each individual car as shown in Equation 3. The dv is then modeled by Equation 1 where spacing between the cars in the traffic model will then be described as

$$s = (\Delta x + \Delta v * t) - L \tag{4}$$
$$\Delta v = v_i - v_{i_{-1}} \tag{5}$$
$$\Delta x = x_i - x_{i_{-1}} \tag{6}$$

where L is the Length of the car and $x_{i_{-1}}$ and $v_{i_{-1}}$ is the position and the velocity of the preceding car respectively.

1.2 Computational Model

```python
from numpy import sqrt, array, linspace, zeros, concatenate
from random import randint
from matplotlib.pyplot import figure, subplot
from scipy.integrate import odeint

#Traffic dynamics for a set of cars in a linear line.

def rate_func( t, V ):
    # RATE_FUNC: IDM Car model
    # Model a car approaching a solid wall

    # unpack
    x = V[:ncars] # position
    v = V[ncars:] # velocity

    dv = zeros(ncars)
    for i in range(ncars):

        # Compute acceleration from IDM
        # Lead Car
        if i == 0:

            Vblock = v_a
            Xblock = x_a0 - x[i]

            s = (Xblock + Vblock*t) - L# distance to car ahead
            delta_v = abs(v[i] - v_a) # approach speed to car ahead
            sStar = s0 + v[i]*T + (v[i]*delta_v)/(2*sqrt(a_accel*b_decel))
            a_idm = a_accel*(1-(v[i]/v0)**delta_exp-(sStar/s)**2)

        # Following Cars
        else:
            delta_v = abs(v[i] - v[i-1])
            sStar = s0 + v[i]*T + (v[i]*delta_v)/(2*sqrt(a_accel*b_decel))
            Vblock = v[i] - v[i-1] #diff of velocity
            Xblock = x[i] - x[i-1] #diff of position
            s = (Xblock + Vblock*t) - L
            a_idm = a_accel*(1-(v[i]/v0)**delta_exp-(sStar/s)**2)
         dv[i] = a_idm

     # compute derivatives
     dx = v

     # pack rate array
     rate = concatenate([dx, dv])
     return rate

 # set parameters
 T = 1.8 #time headway
 delta_exp = 4
 L = 5
 a_accel = 0.3
 b_decel = 3
 v0 = 28 #desired speed in m/s
 s0 = 2.0 #desired gap m

 #Lead Car
 x_a0 = 2000
 v_a = v0/2

 #number of cars
 ncars = 10

 # set initial conditions
 xinit = array([0, -10, -20, -30, -40, -50, -60, -70, -80, -90]) *10
 vinit = zeros(ncars)

 #Main
 # pack i.c.
 X0 = concatenate([xinit, vinit])

 # set the time interval for solving
 Tstart = 0
 Tend = 600

 # Form Time array
 time = linspace(Tstart,Tend,400) # 400 steps for nice plot

 X = odeint(rate_func, X0, time, tfirst=True)

 # unpack the results. In the output array, variables are columns, times are rows
 pos = X[:,:ncars]
 vel = X[:,ncars:]

 #graphing -------------
 fig=figure()
 ax1 = subplot()
 ax1.plot(time,pos)
 ax1.set_xlabel('time (s)')
 ax1.set_ylabel('distance (m)', color='b')
 ax1.tick_params('y', colors='b')

 ax2=ax1.twinx()
 ax2.plot(time,vel, 'r')
 ax2.set_ylabel('velocity (m/s)', color='r')
 ax2.tick_params('y', colors='r')

 fig.tight_layout()
```

1.3 Program Description

The program utilizes a rate function and the odeint python method to calculate the next positions and the cars for each car and prints out the progression over time. The rate function is a method of which uses Equation 1, 2 and 4 in order to calculate the dx and dv for each car and odeint uses the rate function to find it at every time-step.

Universal Variables are set outside of the methods for easy access and visibility, which allows for simple manipulations to review the results. The data is then drawn onto the same plot with the graphs *time vs position* and *time vs velocity* on the same plot for comparison.

2 Experiments

2.1 Establishing Reference

The first step was to establish a reference point using a standard set of parameters and initial conditions as shown in Table 1 and 2 and the graphical results can be found in Figure 1. As expected, all vehicles follow a relatively similar trend where the cars accelerate to 28 m/s and then rapidly decelerate when they reach the leading car. The position of the cars stay relatively equidistant to each other all throughout the trial which is to be expected from the fact that they all have the same parameters aside from the starting position.

Table 1: Standard Behavior Parameters

Parameter	Value
Desired Speed v_0	28 m/s
Time Headway T	1.8 s
Minimum gap s_0	2.0 m
Acceleration a	0.3 m/s^2
Deceleration b	3 m/s^2
Vehicle Length L	5 m
Acceleration Exponent δ	4

Table 2: Initial Conditions

Initials	Values
Number of Cars	10
Starting distance between each Car	10 m
Initial velocity of all cars	0 m/s
Lead Car Position	2000 m
Lead Car Velocity	$v_0/2$ m/s

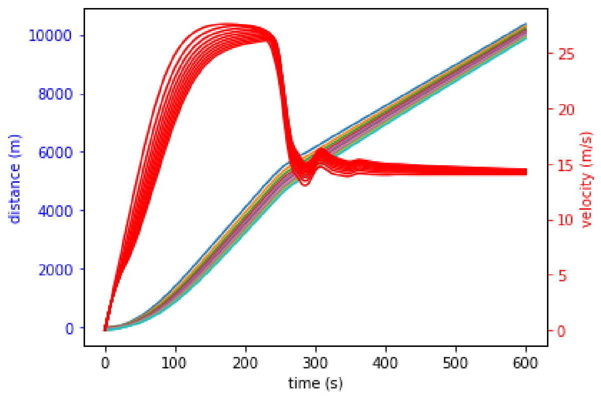
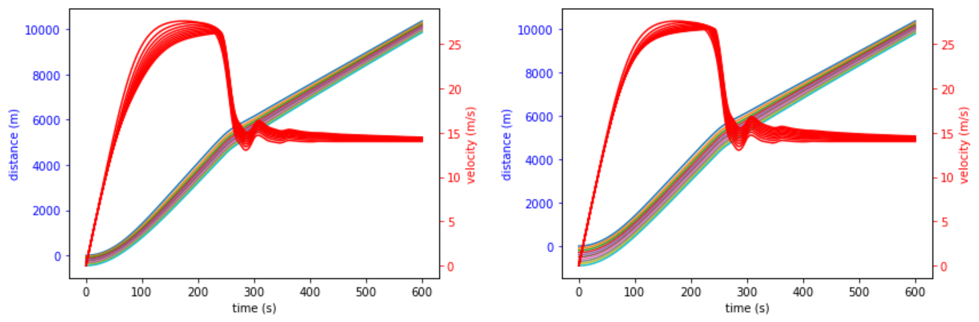


Figure 1: Reference Case.

2.2 Experiment 1 - varying starting positions

In the reference case, the cars are set 10 m apart from each other in the beginning, which leads to the question of what happens when the starting gap is larger. By multiplying the Starting distance between each car by a factor of 5 and 10, we are able to model what the traffic would look like after the starting distances are expanded to 50 and 100 m as shown in figure 2a and 2b respectively. As a result it shows that the cars are all able to reach the "desired" speed of 28 m/s at a much a much more faster rate when the starting gaps are higher. As the leading car's parameters have not been changed, the time where deceleration happens remains the same in comparison to the reference case.



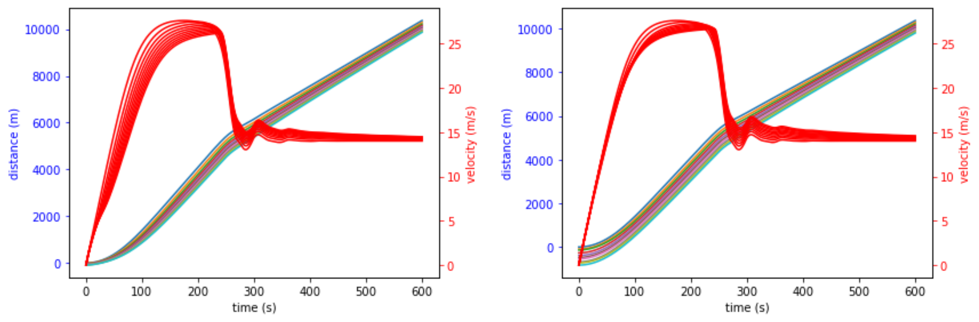
(a) Starting Gap of 50m (b) Starting Gap of 100m

Figure 2: Traffic Dynamics with larger starting gaps

So what happens when the starting positions are not equidistant from each other? In order to model this, a randint() method was added into the code as shown below:

```
xinit = array([0, -10, -20, -30, -40, -50, -60, -70, -80, -90])
for i in range(len(xinit)):
    xinit[i] = xinit[i] + randint(0,5)
```

By doing so we were able to create a case where the starting positions were in a much more random state. However as shown in figure 3a. the difference compared to the base case is almost unidentifiable so we expanded the initial gap to be 100 m and then allow the random integer added to have a range of 0 - 75 with the results as shown in figure 3b.



(a) 10 m Gap + a Random Int from 0 to 5 (b) 100 m Gap + a Random Int from 0 to 75

Figure 3: Traffic Dynamics with random starting gaps

As a result there is no discernible difference in the results of a varying gap size to a equidistant gap size in both the base case and the 100 m case. This shows that the starting positions of the car has little to no effect on the positions and velocities of the cars before and after the following cars catch up to the leading car.

2.3 Experiment 2 - Sudden drop in speed in the lead car

As shown in the reference case, the velocity of the lead car is a constant and all of the following cars adjust their speed according to the parameters of the lead car. But what happens if the lead car suddenly drops speed? To model this, a conditional statement that is designed to drop the velocity of the lead car to 1 m/s after 500 seconds have passed and is placed into the code. The reason why the speed does not drop to 0 m/s is because it would then cause a mathematical error of divide by zero. The result, as shown in figure 4, shows that at time 500, all cars velocities are dropped to 1 m/s and remains as such to the end.

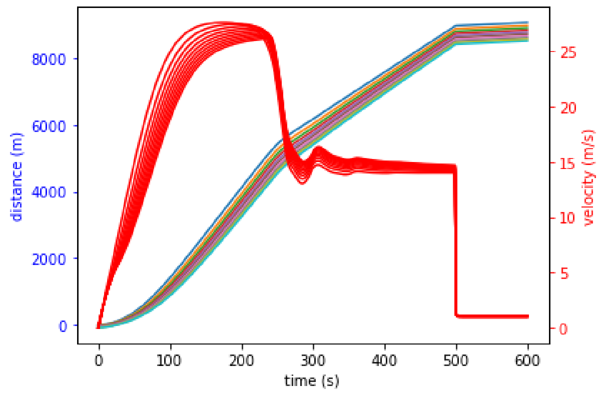


Figure 4: Traffic Dynamics when the Lead Car Velocity Drops to 1 m/s at 500s

2.4 Experiment 3 - Varying acceleration and deceleration

The last variable we wanted to observe were the acceleration and deceleration of the cars. By giving each car a different acceleration and deceleration, we can create a much more randomized model that represents the various types of cars and drivers as they drive in a straight line. The drivers are given randomized values of acceleration and deceleration as shown in table 3. The results of the model shown in Figure 5 display a much more erratic behavior in all cars, some in more extreme cases, but as the same as the reference case, when the time reaches 250s, all of the following cars experience a drop in velocity and converges to the velocity that the lead car is at. Although the resulting gaps between each car does vary a wide amount compared to each other.

Table 3: Randomized car Acceleration (a) and Deceleration (b)

	Car 1	Car 2	Car 3	Car 4	Car 5	Car 6	Car 7	Car 8	Car 9	Car 10
Acceleration (a) (m/s^2)	0.3	0.5	0.2	0.7	0.6	0.3	0.9	0.4	0.1	1.2
Deceleration (b) (m/s^2)	3	2	5	4	1	7	5	3	4	6

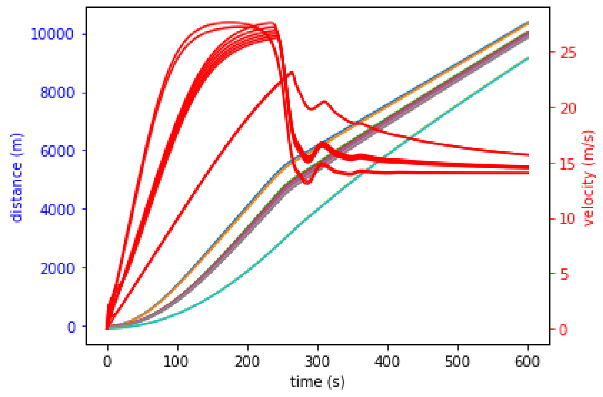


Figure 5: Model where each Car has a Randomized Acceleration and Deceleration

3 Analysis and Discussion

As a result of each experiment run, the ending behavior of the cars in a traffic model will always end up at the same velocity of the lead car and will retain a gap which is only affected by the acceleration and deceleration of the individual car. When you analyze the results in figures 1, 2, 3 and 5, where in all cases the lead cars speed is at 14 m/s, all cars regardless of their current parameters have converged at the lead cars speed. As the leading car is at half the "desired" speed of the IDM, it then becomes the limiting factor which affects the final speed each car will be at. Even when the leading car changes it's speed, as shown in figure 4, the following cars will adjust their speed to match the lead cars final speed.

The gap between the cars is affected by the acceleration and deceleration of the individual cars and nothing else. In Figure 2, there was no discernible difference in the the gap size in both graphs despite having a different initial positions, nor were there any differences that could be observed in gaps between the cars final state when initially the cars were at a random distance apart. On the other hand, when the acceleration and deceleration of all the cars were randomized, significant differences can be found in both the acceleration of all the cars and the positions of the cars relative to each other. The reason for this can be attributed to how the Mathematical Model is designed. in equation 1, the term is multiplied by the acceleration constant of each car which in turn affects the rate of change in the velocity. the deceleration appears in equation 2 alongside the acceleration again. the position these constants occupy in equation 2 doesn't make too large of a change, but it does cause an effect in the change in velocity. Therefore we can conclude that the acceleration will affect the ideal distance between the moving vehicles.

4 conclusion

In conclusion we can model the behavior of drivers in a traffic model with various parameters such as starting positions, changes in the leading car, acceleration and deceleration variations which can help develop a more accurate model in the end.

References

Treiber, Martin et al. "Congested traffic states in empirical observations and microscopic simulations". Phys. Rev. E 62. (2000): 1805–1824.