Dynamics of Traffic Flow
Analysis of Two Lane Nagel-Schreckenberg Model

Ali Dalbah*

[1] *Department of Physics — Birzeit University*

(Dated: June 10, 2023)

The Nagel-Schreckenberg model is a well-known cellular automaton used to simulate traffic flow and study traffic congestion phenomena. This paper propose an extension of the Nagel-Schreckenberg model to a two-lane scenario. The model considers a two-lane road system with periodic boundary conditions, where each lane contains a certain number of vehicles. Incorporating lane-changing rules to allow vehicles to switch lanes based on certain conditions, such as speed,available space, and another variable called sw which is the probability a particle will switch lane if it will allow it not to deaccelerate. Through computer simulations, analyzing the impact of lane-changing behavior on traffic flow and congestion patterns.

Keywords: Nagel-Schreckenberg Model, two-lane

CONTENTS

I. INTRODUCTION

Analysis of the complex dynamics encountered in two-lane traffic environments. By incorporating lane-changing rules and parameters that reflect real-world scenarios, this model offers insights into congestion, lane-changing behaviors, and vehicle interactions. This report explores the key features of the two-Lane NaSch Model, investigates its impact on traffic flow dynamics; by exploring the effects of another lane on its flux.

II. MODEL SPECIFICATIONS

The model introduces another lane for the Nasch Model, if there was no interaction between the two (meaning no particle can switch lane), the lanes can be considered as two independent Nasch Models.

A new parameter was introduced (sw) which is the probability that a particle would switch lane if it has to slow down due to the gap and switching can give it a higher velocity.

A particle would switch its lane if the gap is smaller than the "changing gap" and the "backward gap" is larger than 0 as shown in the figure below; which coincides with the real-world of a car won't change lane if its going to slowdown or it thinks it may collide with the a car (checks if it can switch safely).

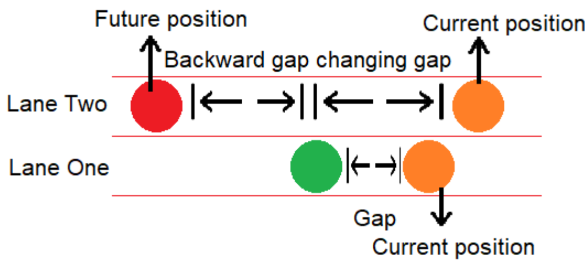

Figure 1: Road changing exaplenation figure

Note: A Particle won't switch its lane randomly only if its going to slowdown.

---

* 1192085@student.birzeit.edu; Instructor: Franz-Josef Ulm

III. GRAPHS

Graphs key:
Green is for sw = 1
red is for sw = 0

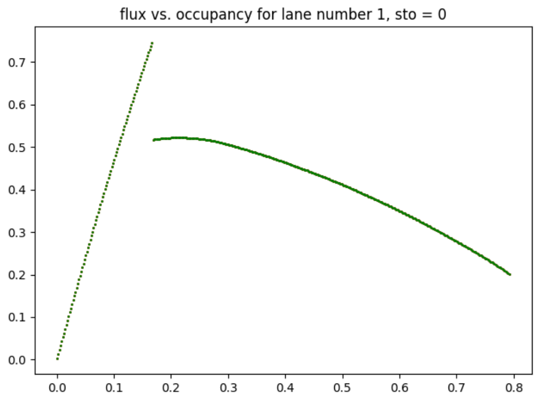

Figure 2: $flux\,(\rho \cdot \bar{v})$ vs. ρ for lane one when $p = 0$

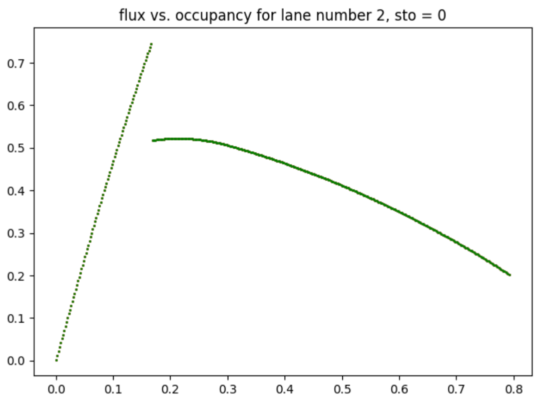

Figure 3: $flux\,(\rho \cdot \bar{v})$ vs. ρ for lane two when $p = 0$

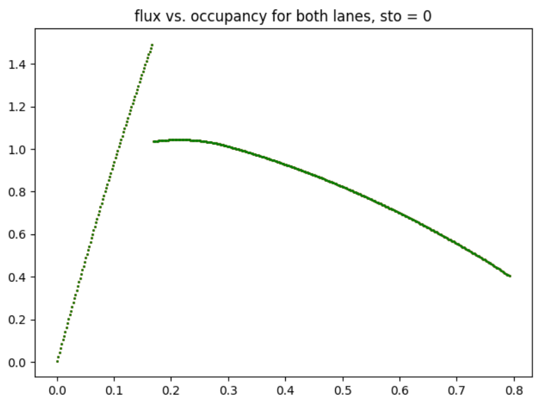

Figure 4: $flux\,(\rho \cdot \bar{v})$ vs. ρ for both lanes when $p = 0$

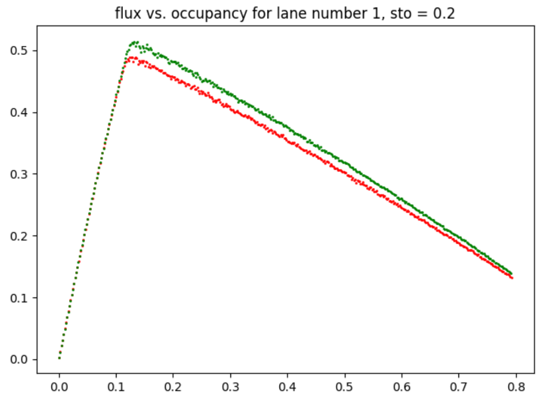

Figure 5: $flux\,(\rho \cdot \bar{v})$ vs. ρ for lane one when $p = 0.2$

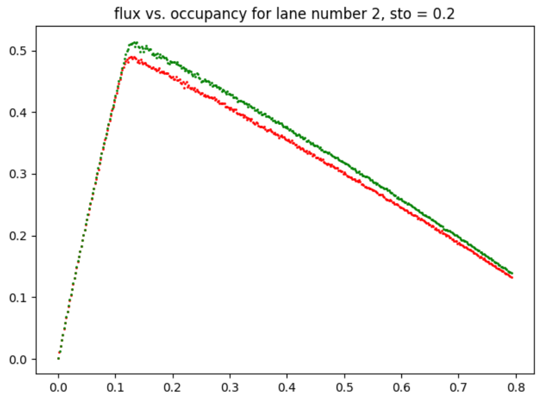

Figure 6: $flux\,(\rho \cdot \bar{v})$ vs. ρ for lane two when $p = 0.2$

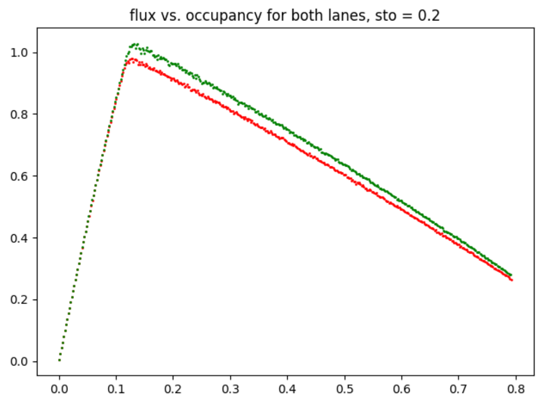

Figure 7: $flux\,(\rho \cdot \bar{v})$ vs. ρ for both lanes when $p = 0.2$

IV. DISCUSSION

As the figures shows, lane changing doesn't effect the road with $p = 0$, but for values $p > 0$, its noticeable that the lane switching can in fact increase the flux of the system (as it increases the mean velocity).

V. LANE SWITCHING RATE

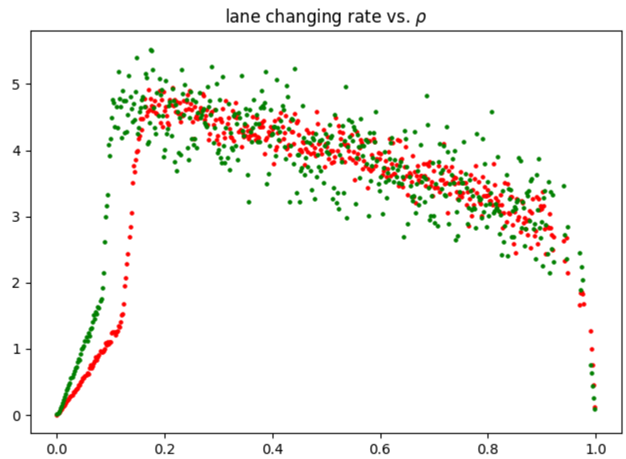

Figure 8: lane switching rate (particle / time) vs. ρ
$p = 0.2$ $p = 0.5$

As it appears from the graph above, switching between lanes goes to maximum at around 20%, then starts to decline.

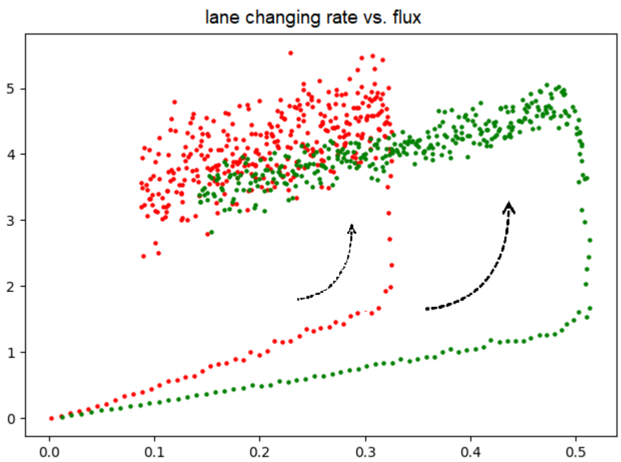

Figure 9: lane switching rate (particle / time) vs. flux
$p = 0.2$ $p = 0.5$
the arrows point to the direction of increasing ρ

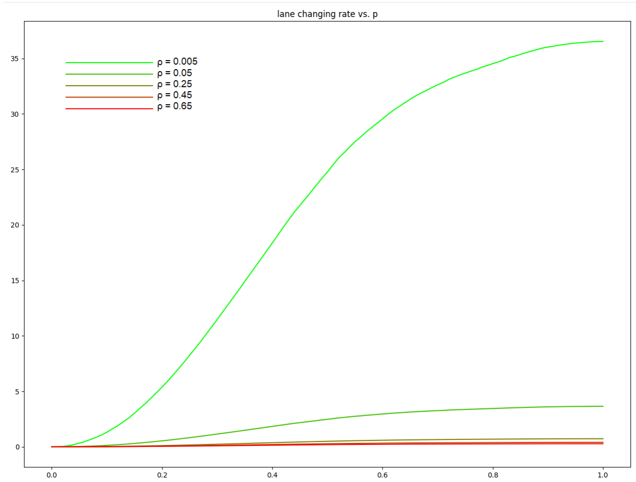

Figure 10: lane changing rate vs. p

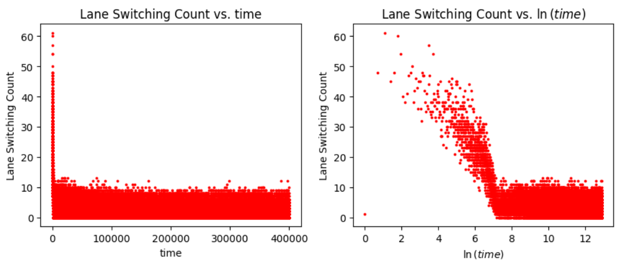

Figure 11: lane changing count vs time, vs $ln\,(time)$
$p = 0.5,\ \rho = 0.05$

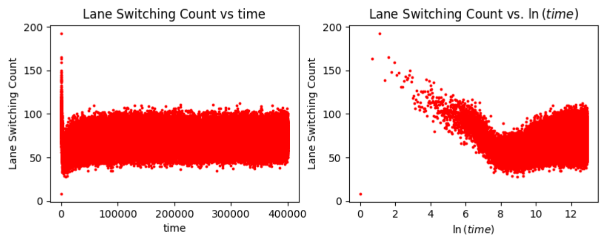

Figure 12: lane changing count vs time, vs $ln\,(time)$
$p = 0.5,\ \rho = 0.15$

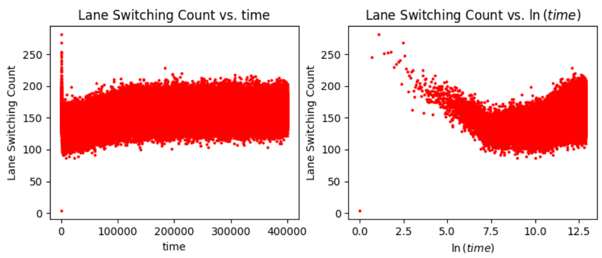

Figure 13: lane changing count vs time, vs $ln\,(time)$
$p = 0.5,\ \rho = 0.25$

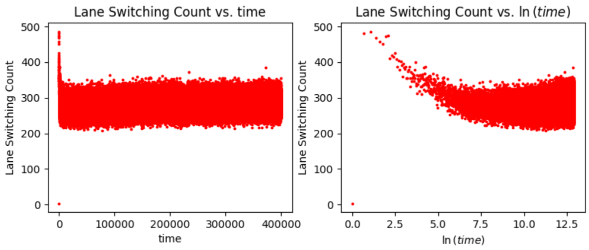

Figure 14: lane changing count vs time, vs $ln\,(time)$ $p = 0.5,\ \rho = 0.45$

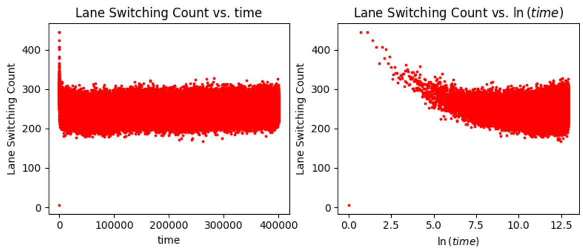

Figure 15: lane changing count vs time, vs $ln\,(time)$ $p = 0.5,\ \rho = 0.6$

VI. DENSITY VS TIME

The following figures are ρ vs. t, both lanes start with random number of particle, the 2 lanes have roughly the same number, simulation time $4 \cdot 10^4$.
Where each color represents a different lane.

$$\rho_{total} = \frac{1}{2}\left(\rho_1 + \rho_2\right) \tag{1}$$

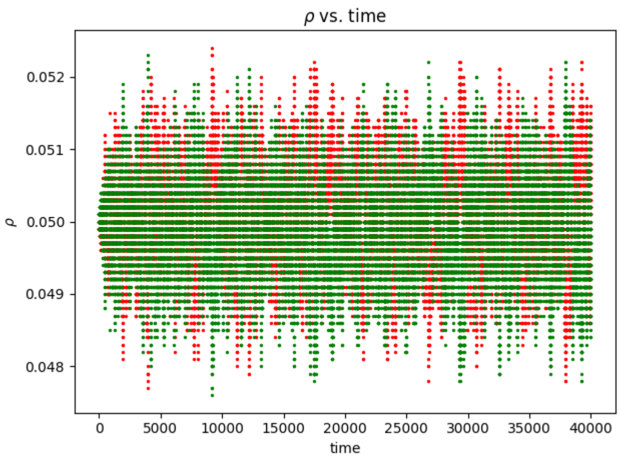

Figure 16: $\rho_{total} = 0.05$

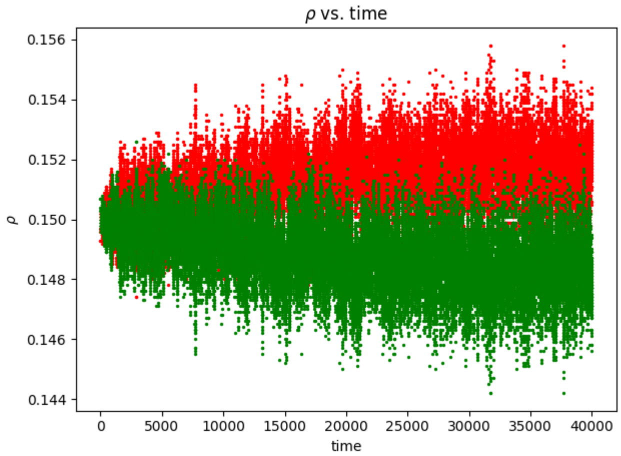

Figure 17: $\rho_{total} = 0.15$

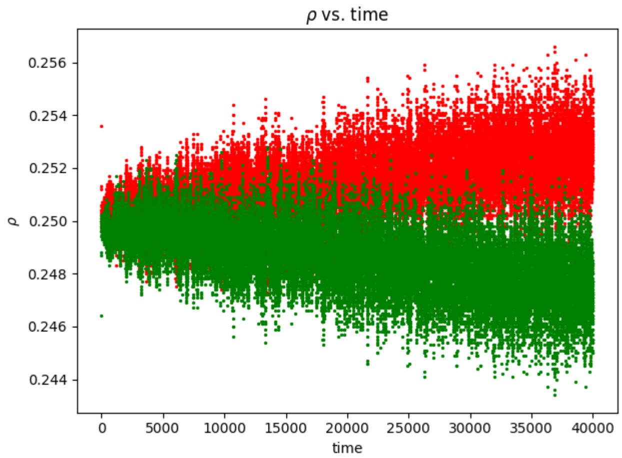

Figure 18: $\rho_{total} = 0.25$

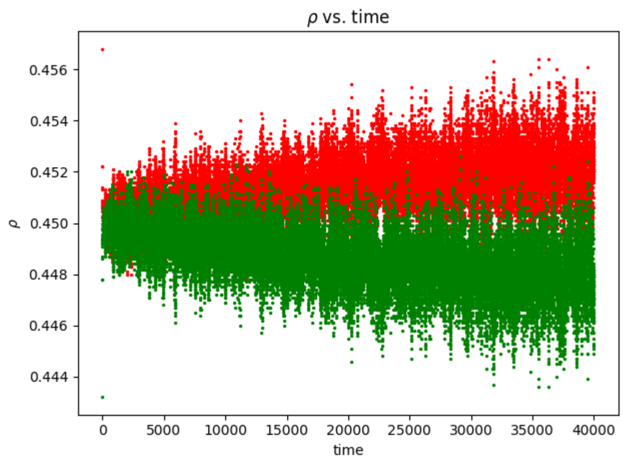

Figure 19: $\rho_{total} = 0.45$

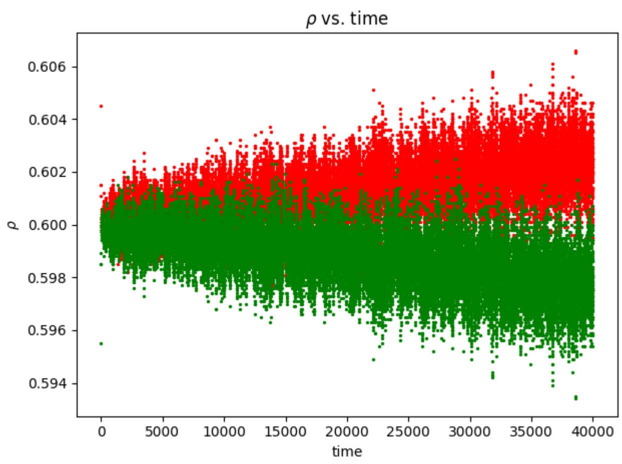

Figure 20: $\rho_{total} = 0.6$

Even if one lane was loaded with $\rho = 2 \cdot \rho_{total}$, and second lane was loaded with $\rho = 0$, The same graphs appear.
There's a small difference for $\rho > \rho_c$ at most 0.008, its not significant and can be ignored as the two lanes keeps switching ρ as one might say.
For a more detailed graph, the following figure had a simulation time of $1 \cdot 10^6$, 100 times more than the last graphs.

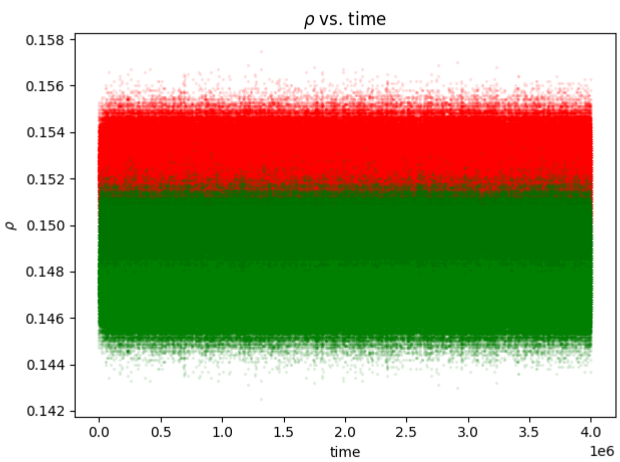

Figure 21: $\rho_{total} = 0.15$

VII. PROBABILITY DISTRIBUTION FUNCTION

The simulation time for the following figures was $4 \cdot 10^5$ to reach statistical stationarity then the velocities for every single particle was measured. V_{MAX} sat to 10 to give more distribution.
green for switching enabled.
red for switching disabled.

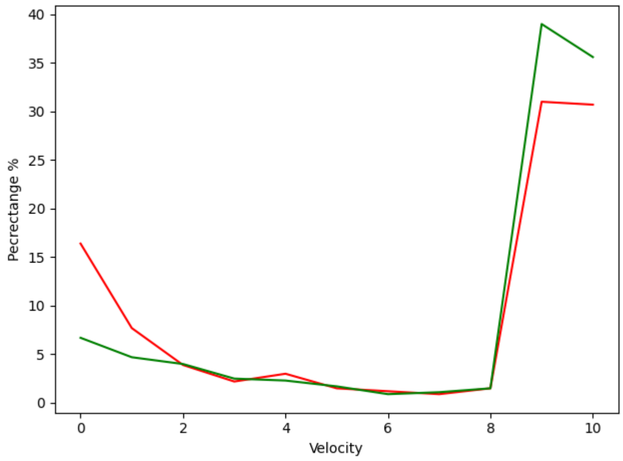

Figure 22: $\rho_{total} = 0.05$

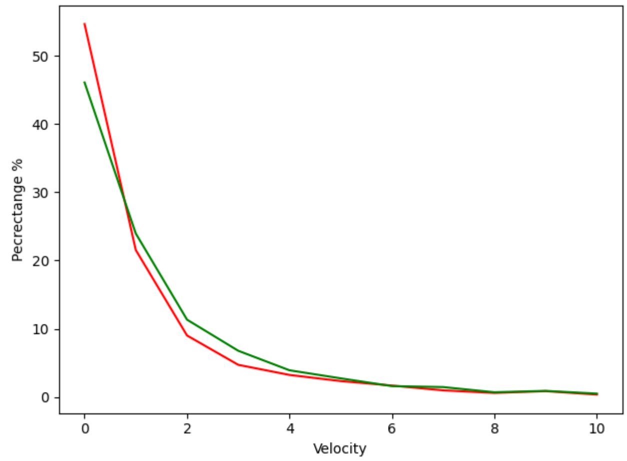

Figure 23: $\rho_{total} = 0.25$

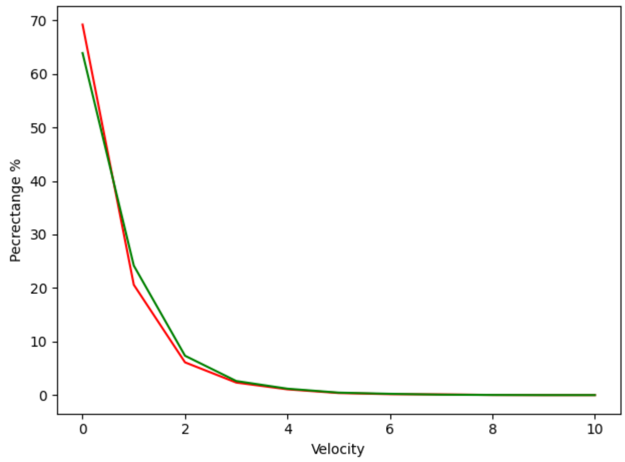

Figure 24: $\rho_{total} = 0.45$

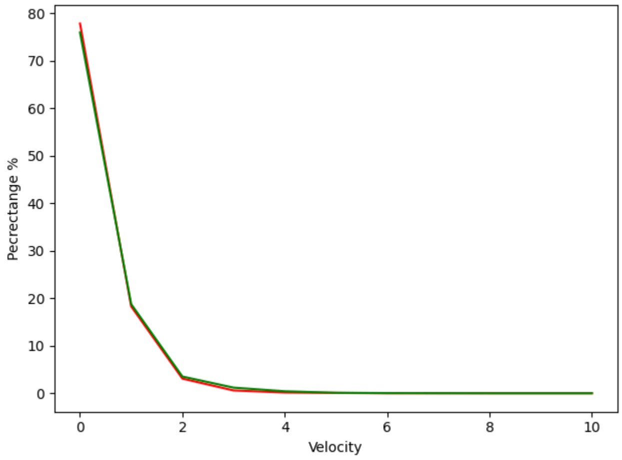

Figure 25: $\rho_{total} = 0.6$

As the figures show, as ρ_{total} increases the two curves tend to get similar.

VIII. VELOCITY-VELOCITY COVARIANCE

To calculate the Velocity-Velocity Covariance of a lane the following equation was used.

$$C_{vv}(r) = \left(\frac{1}{T} \sum_{t=1}^{T} \frac{1}{N_t} \sum_{i=1}^{N_t} v_i(t) v_{i+r}(t) \right) - \langle v \rangle^2 \tag{2}$$

$$\langle v \rangle \text{: is the mean velocity of the lane.} \tag{3}$$

$p = 0.5$ for all the next figures, each color represents a different lane. simulation time was $3 \cdot 10^5$ with $1 \cdot 10^5$ warm up.

A. Figures for Lane Switching Off

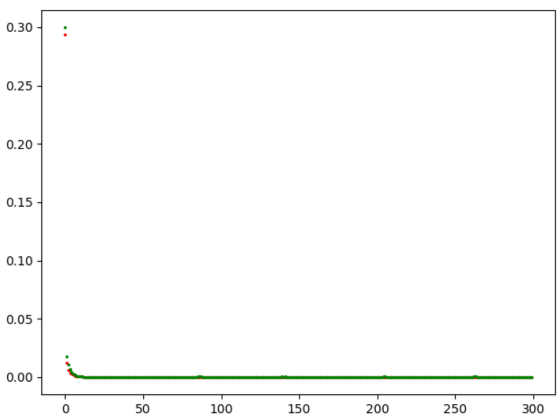

Figure 26: $C_{vv}(r)$ vs r, $\rho_{total} = 0.05$, lane switching off

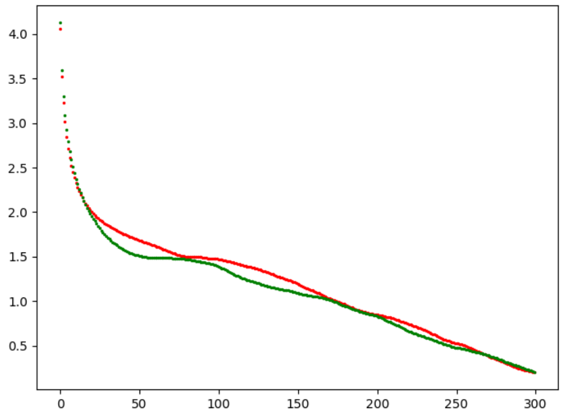

Figure 27: $C_{vv}(r)$ vs r, $\rho_{total} = 0.15$, lane switching off

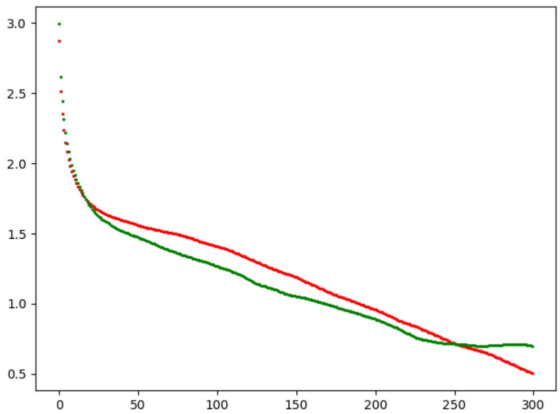

Figure 28: $C_{vv}(r)$ vs r, $\rho_{total} = 0.25$, lane switching off

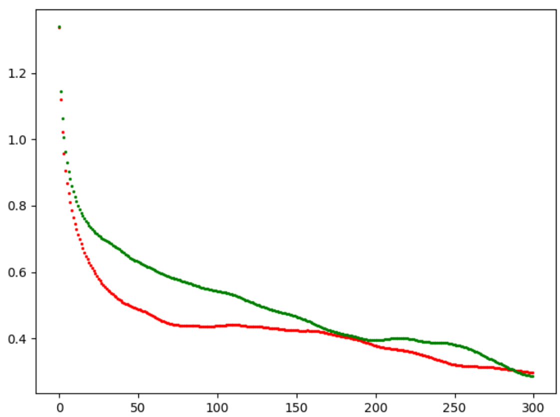

Figure 29: $C_{vv}(r)$ vs r, $\rho_{total} = 0.45$, lane switching off

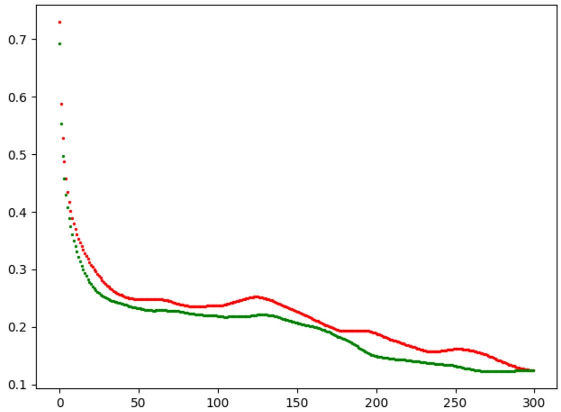

Figure 30: $C_{vv}(r)$ vs r, $\rho_{total} = 0.6$, lane switching off

B. Figures for Lane Switching On

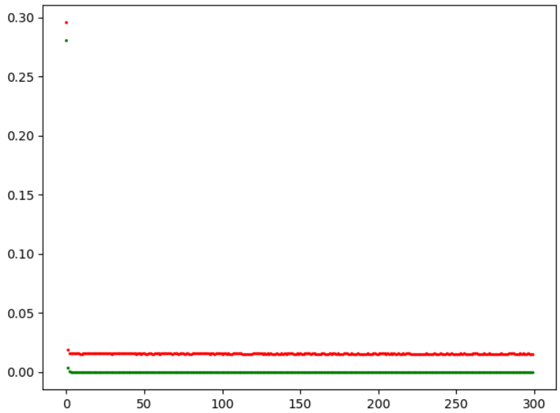

Figure 31: $C_{vv}(r)$ vs r, $\rho_{total} = 0.05$, lane switching on

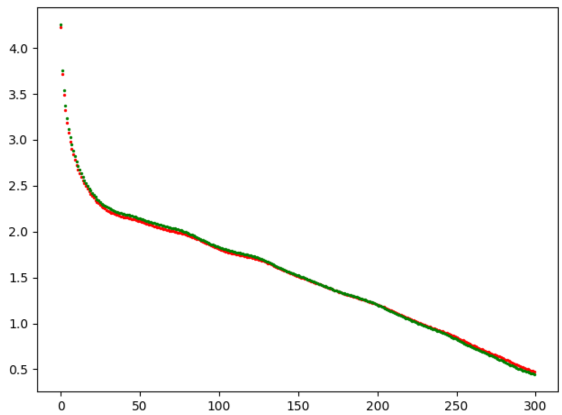

Figure 32: $C_{vv}(r)$ vs r, $\rho_{total} = 0.15$, lane switching on

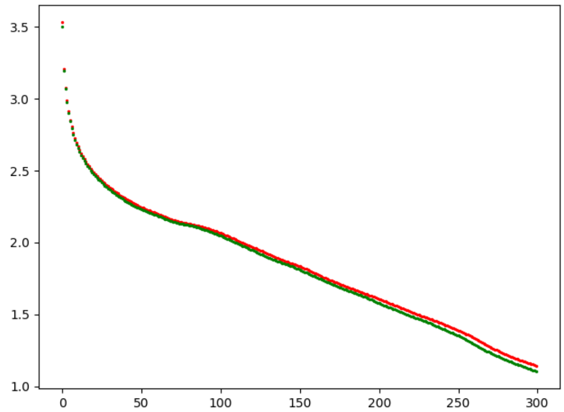

Figure 33: $C_{vv}(r)$ vs r, $\rho_{total} = 0.25$, lane switching on

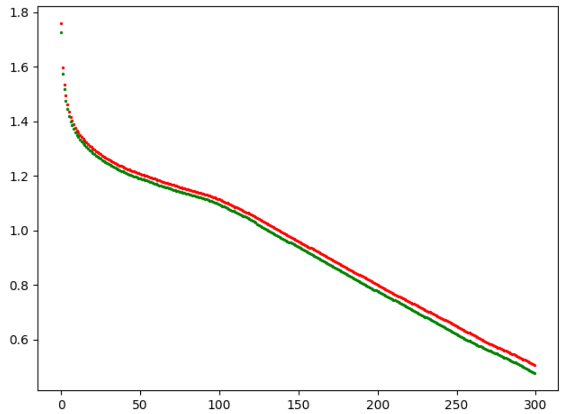

Figure 34: $C_{vv}(r)$ vs r, $\rho_{total} = 0.45$, lane switching on

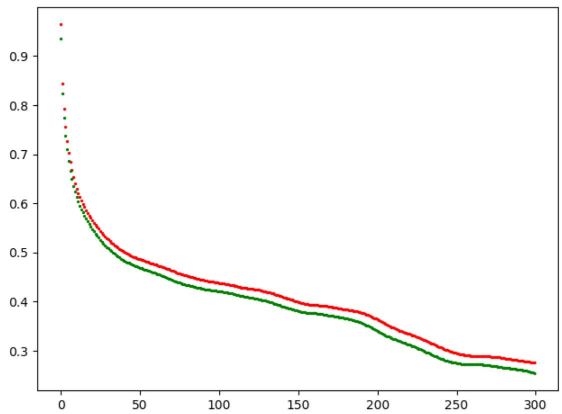

Figure 35: $C_{vv}(r)$ vs r, $\rho_{total} = 0.6$, lane switching on

The two lanes tend to have an almost exact C_{vv} function when the lane switching is enabled; this tells that the two lanes have a similar number of particles with similar velocities.

IX. APPENDIX

```
#include <cstdio>
#include <iostream>
#include <cstdlib>
#include <vector>
#include <string>
#define V_MAX 5

static const int l = 5000;
static float meanVTemp;

struct particle{
    int pos;
    int vel;
    particle(int pos, int vel): pos(pos), vel(vel) {}
    particle(const particle& p): pos(p.pos), vel(p.vel) {}
    inline void move() {
        pos = (pos + vel) % l;
    }
    inline void accelarate() {
        vel++;
        if (vel > V_MAX) vel = V_MAX;
    }
    inline void deaccelarate() {
        if (vel > 0) vel--;
    }
    inline int operator-(const particle& p) {
        return pos - p.pos;
    }
    inline bool operator>(const particle& p) {
        return pos > p.pos;
    }
    inline bool operator<(const particle& p) {
        return pos < p.pos;
    }
};

typedef std::vector<particle> vector;

float getRandom() {
    return (float)(std::rand()) / RAND_MAX;
}

int getAheadParticle(const particle& p, const vector& road) {
    const int goal = p.pos;
    const int size = road.size();
    if (size == 0) return -1;
    int left = 0; int right = size;
    int middle;
    while (left < right) {
        middle = (right + left) / 2;
        if (road[middle].pos < goal) {
            left = middle + 1;
        } else {
            right = middle;
        }
    }
    return left == size && road[left].pos < goal ? 0: left;
}

void doOperations(vector& thisRoad, vector& otherRoad, const float sto, const float sw) {
    int size = thisRoad.size();
    if (size == 0) {
        meanVTemp = V_MAX;
        return ;
    }
    meanVTemp = 0;
    for (int i = 0; i < size; i++) {
        particle& p = thisRoad[i];
        p.accelarate();
        int gap = (thisRoad[(i + 1)%size] - p + l) % l - 1;
        if (gap < p.vel) {
            if (getRandom() < sw) {
                int x = getAheadParticle(p, otherRoad);
                int otherGap = x == -1? V_MAX: (otherRoad[x] - p + l) % l - 1;
                bool backWontSlow = x == -1 || x == 0 || ((otherRoad[(x - 1)].pos + otherRoad[(x - 1)].vel) % l ) < p.pos ;
                if (otherGap > gap && otherRoad.size() < l - 1 && backWontSlow) {
                    if (x == -1) otherRoad.emplace_back(particle(p));
                    else if (x == 0) {
                        if (otherRoad[0] > p) {
                         otherRoad.insert(otherRoad.begin(), particle(p));
                        }
                        else {
                         otherRoad.emplace_back(particle(p));
                        }
                    } else {
                         otherRoad.insert(otherRoad.begin() + x, particle(p));
                    }
                    thisRoad.erase(thisRoad.begin() + i);

                    i--;
                    size--;
                    continue;
                }
            }
            p.vel = gap;
        }
        if (getRandom() < sto) p.deaccelarate();
        meanVTemp += p.vel;
        p.move();
        if (p.vel > V_MAX)
            std::cout<<p.vel<<std::endl;
    }
    if (size > 1) {
        if (thisRoad[size - 1].pos < thisRoad[size - 2].pos) {
            thisRoad.insert(thisRoad.begin(), particle(thisRoad[size - 1]));
            thisRoad.pop_back();
        }
    }
    meanVTemp /= size;
}

void appendToFile(const char* name, const char* data) {
    FILE* file = fopen(name, "a");
    if (file == nullptr) return (void)(std::cout<<"couldn't open file "<<name<<std::endl);
    fprintf(file, "%s", data);
    fclose(file);
```

```
}

void start(const int n, const float sto, const
    float sw) {
    vector lane1, lane2;
    bool insertTo = true;
    for (int i = 0; i < n; i++) {
        if (getRandom() < 0.5f && lane1.size() <
     l)
            lane1.emplace_back(particle(lane1.
    size(), 0));
        else
            lane2.emplace_back(particle(lane2.
    size(), 0));
        insertTo = !insertTo;
    }
    const int simulationTime = 4000;
    float meanV1 = 0, meanO1 = 0, meanV2 = 0,
    meanO2 = 0;
    int t = 0;
    while (t < simulationTime) {
        doOperations(lane1, lane2, sto, sw);
        meanV1 = (meanVTemp + meanV1 * t) / (t +
     1);
        meanO1 = ((float)(lane1.size()) / l +
    meanO1 * t) / (t + 1);
        doOperations(lane2, lane1, sto, sw);
        meanV2 = (meanVTemp + meanV2 * t) / (t +
     1);
        meanO2 = ((float)(lane2.size()) / l +
    meanO2 * t) / (t + 1);
        t++;
    }
    std::string str = "N = " + std::to_string(n)
     + "\t sto = " + std::to_string(sto) + "\t
    sw = " + std::to_string(sw);
    str += "\tMeanV1 = "+std::to_string(meanV1)+
    "\tMeanO1 = "+std::to_string(meanO1);
    str += "\tMeanV2 = "+std::to_string(meanV2)+
    "\tMeanO2 = "+std::to_string(meanO2)+"\n";
    appendToFile("data.txt", str.c_str());
}


int main(int argc, char **argv) {
    std::srand(time(nullptr));
    int n = argc == 2? std::stoi(argv[1]): 5;
    for (n; n < 1.9f * l; n+=20) {
        start(n, 0.f, 0.f);
        start(n, 0.2f, 1.f);
        start(n, 0.2f, 0.f);
        start(n, 0.2f, 1.f);
    }
}
```

Listing 1: "The Nagel-Schreckenberg Model for 2 lane"