A SIMULATION OF TRAFFIC FLOW ON HIGHWAYS

BY

JESSICA LEES

A SENIOR RESEARCH PAPER PRESENTED TO THE DEPARTMENT OF MATHEMATICS AND COMPUTER SCIENCE OF STETSON UNIVERSITY IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF BACHELOR OF SCIENCE

STETSON UNIVERSITY

2003

TABLE OF CONTENTS

LIST OF TABLES

TABLE

LIST OF FIGURES

FIGURE

ABSTRACT

A SIMULATION OF TRAFFIC FLOW ON HIGHWAYS
By

Jessica Lees

May 2003

Advisor: Erich Friedman
Department: Mathematics and Computer Science

We consider one-way traffic flow on highways and, using collected data, create a model. We are unable to assume a normal distribution in the collected data, so we use a simulation to mimic the behavior of traffic. We run multiple simulations of the model over various traffic densities to determine how average traffic speed is related to traffic density. We find a linear relationship that fits the data extremely well. We also consider changes in the passing rules of the simulation and the addition of a third lane to analyze how these changes affect the results. We test whether there is a statistical difference between the different models.

CHAPTER 1
Relevant Research

Many models have been created to recreate traffic and learn more about it. Some models consider the psychology of drivers to predict behavior; others look for comparisons between traffic patterns and naturally-occurring phenomenon in order to make generalizations. Some models approach traffic from a microscopic approach, to see how an individual car may change traffic behavior, while others take a macroscopic perspective to see if patterns can be ascertained by examining behavior among traffic as a whole.

One way that highway traffic has been modeled recently is by considering cars as molecules in a gas that want to move in one direction with a constant velocity. The properties of gases can then be related to traffic to help predict behavior. Boris Kerner, a physicist at Daimler-Benz Research Institute, used this method to discover "synchronized traffic," a phenomena between free-flowing traffic and jams, where traffic moved slowly but steadily at the same speed in all three lanes. He found this occurred mostly at on-ramps, where a high number of cars merging caused traffic to completely synchronize and affected traffic flow for hours. [4]

Another model treats individual cars as cellular automatons. These cellular automatons then follow certain rules that are assigned in the program, such as passing when the car in front is moving too slowly. Some models also add "fudge factors" to allow for erratic behavior in order to mimic traffic patterns more realistically. [4]

Two theoretical physicists, Bernardo Huberman, of Xerox Palo Alto Research Center and Dirk Helbing, of the University of Stuttgart in Germany, used these two techniques to create a model for highway traffic. They replicated Kerner's results and found that synchronized traffic could occur in other situations, such as a slow car passing a slower car. They also expanded on the number of traffic phases, showing at least five distinct patterns that occur between free-flow and jams. They also showed that traffic can easily and quickly switch phases. According to Helbing, "At certain traffic densities, small causes have large effects. In particular, most types of

congestion are [caused by] small disturbances that grow and at some point cause the traffic to break down." [4]

Huberman and Helbing also used cellular automatons to simulate how long it took a mix of cars and trucks traveling at different speeds to cover a six mile stretch of highway. They found that 35 vehicles per mile of highway was a critical density where the traffic begins to synchronize and travel as a block. They found this block to be very fragile and could easily turn into stop-and-go traffic. Their solution to preventing this is to time on-ramp lights according to current traffic conditions rather than set time schedules. They claim that by allowing the light for the on-ramp to be green only when there are gaps in the highway traffic near that on-ramp, the block would be preserved and traffic would stay in a synchronized state. [4]

Another useful simulation of highway traffic was created by Martin Treiber and Dirk Helbing, both theoretical physicists at the University of Stuttgart in Germany. Treiber and Helbing examined results from this simulation to see how free-flowing highway traffic dissolves into stop-and-go patterns. They first identified three different states of traffic which can coexist on a single road, behind an inhomogeneity, such as an accident, on-ramp, etc. These traffic states, going in the upstream direction, are called:

- homogeneous congested traffic, where all lanes are approximately equally congested and therefore synchronized;
- inhomogeneous congested traffic, involving a 'pinch region' in one lane; and
- stop-and-go traffic.

Treiber and Helbing report that, in contrast to other studies, the phenomenon of this traffic behavior is caused by a region of very dense traffic that is stable in itself, but from which any instabilities or "shock waves" travel in an upstream direction. [8]

The simulation uses the Intelligent Driver Model, or IDM. The IDM simulates traffic with the idea that each car is an individual "molecule" of the larger traffic system. The parameters of the IDM are:

- desired velocity on a free road;
- desired safety time headway when following other vehicles;
- acceleration in everyday traffic;
- "comfortable" braking distance in everyday traffic;
- minimum bumper-to-bumper distance to the front vehicle;
- acceleration exponent. [9]

The IDM allows you to modify these parameters for the type of vehicle you want in the simulation; aggressive drivers have a lower bumper-to-bumper distance, higher acceleration, and lower safety time headway. The rate at which a vehicle is allowed to accelerate depends on the above parameters, the speeds of the vehicle and the vehicle in front, and the gap between the vehicles. [8]

For the simulation, Treiber and Helbing used a 20 km stretch of open freeway over a time interval of 120 minutes, with initial conditions of 1670 vehicles per hour. They assumed identical values in all parameters of the IDM for all cars in the simulation. In order to create an inhomogeneity, they manipulated the IDM to model a section of the freeway where people are forced to slow down and drive more carefully. This is created by increasing the desired safety time headway and decreasing the desired velocity. [8]

Treiber and Helbing found that after 10 minutes of free-flowing traffic, traffic broke down at the inhomogeneity, with homogeneous congested traffic occurring there. Smaller oscillations then developed upstream from the inhomogeneity and traveled further upstream and grew into stop-and-go waves. Then, the waves either dissipated or merged together, creating jams where traffic was at a standstill. Once these jams formed, they remained intact and traveled upstream without changing shape. Congested traffic at the inhomogeneity turned into free traffic downstream. [8]

Treiber and Helbing graphed the behavior of individual cars at six positions on the freeway: four upstream of the inhomogeneity, one at the inhomogeneity, and one downstream of it. The most interesting graphs are shown in Figure 1.

In D5, at the inhomogeneity, we see a gradual decline in the movement of free traffic, until it levels out to become homogeneous congested traffic. We can see that once cars enter this area, they will be moving at a constant, slower velocity. In D4, 0.5 km upstream of the inhomogeneity, we see the small oscillations that cause inhomogeneous congested traffic, where the traffic is still moving, but it is not at a constant velocity. In D3, 2.0 km upstream of the inhomogeneity, we see larger amplitudes of oscillation, where cars are slowing down more and getting closer to stop-and-go traffic. In D1, 5.2 km upstream of the inhomogeneity, we see stop-and-go traffic in the jams caused by the merging waves of congested traffic. [8]

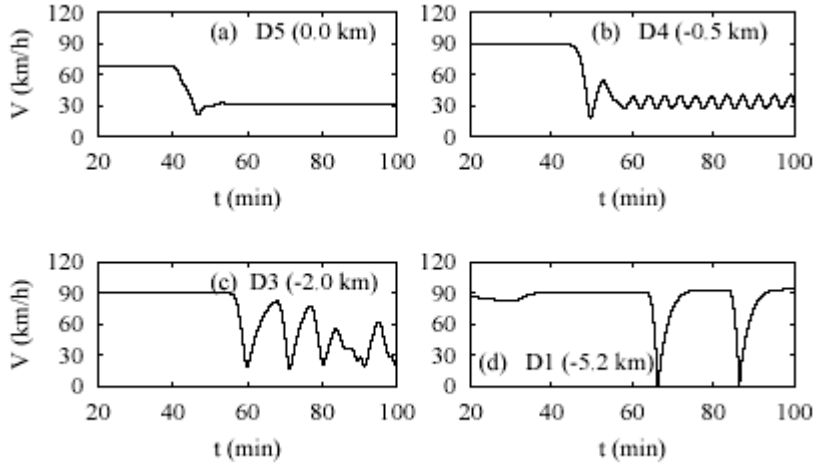

Figure 1 – Treiber and Helbing's Results

These graphs show interesting results, but since we will not be gathering results from individual vehicles in our simulation, we cannot compare our results for the effects of traffic density to those of a specific inhomogeneity.

These models will form a useful basis for our simulation. We will utilize cellular automatons to assign rules to the behavior of vehicles in our program. We would like to find out if our model has a critical traffic density, where the behavior and flow of traffic change dramatically. If this does occur, we will compare our critical density to that found by Huberman and Helbing.

We will also be graphing the results of our simulations to observe what the behavior looks like. A typical equilibrium speed vs. density curve, calculated using partial differential equations, is shown in Figure 2. The graph was created by Del Castillo and is presented by Rui in [7].

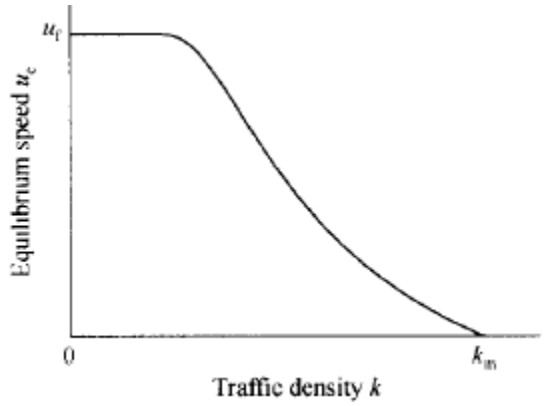

Figure 2 - Typical Traffic Speed vs. Density Curve

In this curve, u_t is the free flow speed and k_m is the jam density. We are interested to see if our model replicates these findings and if there are any similarities in the graphs.

There are also many models that are created to simulate traffic in cities for the purposes of easing congestion during rush hours, planning the layout of streets, or deciding on the timing of stoplights. While these models are interesting, they don't provide relevant information on the behavior of highway traffic, so we did not utilize these types of models for our research.

CHAPTER 2

Introduction of the Problem

We would like the model to predict speeds of traffic, and return data about average speed which we will statistically analyze to give us information about average traffic speed versus traffic density.

2.1. The Model

The model will involve a computer simulation that will be used to provide information about average speed and overall flow of traffic on highways. We will run the simulation using changing amounts of traffic over a certain length of time to find the average traffic speed for each traffic density.

2.1.1. Data

The data used to analyze traffic speeds was taken from Indiana highways from April to June 2002 by Purdue University as part of a joint transportation research program with the Indiana Department of Transportation [1]. This data was used because it contained relevant information about speed distribution on Indiana's highways. The data from each quarter shows roughly the same distribution so we can assume that the data is not influenced by weather, construction or other factors. All data is initially in miles per hour, but we will convert to feet per second for later ease of programming. The term highway is used throughout to refer to an interstate freeway, and can be urban or rural.

2.1.2. Assumptions

The assumptions we will be using throughout the model are:

- Speed patterns found on Indiana highways can be generalized to all U.S. highways.
- Our simulation runs on a highway with a 55 mph posted speed limit.

- Our simulation runs over a stretch of road that is 10 miles (or 52800 feet) long.
- All cars are the same length of 16 feet long.
- All traffic is traveling in one direction.
- The safe following distance between two cars is one car length. All vehicles are forced to leave at least this much distance between themselves and other cars.
- A car's position is measured at the front of the car.
- A car's initial speed will also be referred to as the desired speed. Cars can slow down from this speed when they are behind a slower car, but they never go faster than their desired speed.
- We will run the simulation for a fixed time of 1300 seconds to ensure that the only variables are number of lanes and traffic density, which will make it easier to identify trends in the data.

CHAPTER 3

Statistical Analysis of Traffic Speed Data

In order to better predict the speeds of traffic flow, we would like to fit a normal distribution to it to see if it follows a predictable pattern. We must do a statistical analysis of the data we are given to see if it is indeed normally distributed.

3.1. Mean and Standard Deviation

The mean speed, μ, on a highway with 55 mph posted speed limit, was 62.9 mph. In order to simplify our calculation of standard deviation, we will assume that any vehicle that falls in a speed range is traveling at the midpoint of that speed range, with the exception of the end categories. Since these categories encompass more speeds, we use a speed of 35 mph for the lowest category and 95 mph for the highest category. We compute the variance, σ^2, using the equation:

$$\sigma^2 = \frac{\sum_{i=1}^{M}(c * x_i - \mu)}{(n-1)}$$

where c is the number of data points in the speed category, M is the number of speed categories, x is the midpoint of the speed category, and n is the total number of data points. Our data has σ^2 = 69.124 mph, so σ = 8.314 mph. We then convert these numbers to feet per second, giving us μ = 90.787 ft/sec. and σ = 12.197 ft/sec.

3.2. Chi-Squared Test

We must then use the mean and standard deviation to do a chi-squared test on the data, in order to tell us if the distribution of data points in each speed category is normal. First, we must calculate the expected frequency for each category and then compare it to the observed frequency.

We calculate the expected frequency by finding the probability that, in a normal distribution, a data point will lie in a certain speed category. This is done by computing a test-statistic Z, which can then be looked up in a Z-table, or table of areas under a standard normal curve. The equation for this statistic is:

$$Z = \frac{(x - \mu)}{\sigma}$$

For example, to compute $P(40 \leq x < 45)$, we first rewrite this as $P(-2.63 \leq Z < -2.03)$. From a normal probability table we find that the probability is .0169. Multiplying this probability by the total number of cars gives an expected number of cars for this speed category. We repeat this method for each speed category. The observed frequency is simply the number of cars observed in that speed category, given by the data.

The data used in computing the chi-squared statistic is listed in Table 1.

x-value	Z-value	Expected Frequency	Observed Frequency
$\mathbf{x} \leq 40$	$\mathbf{Z} \leq -2.63$	1421	7284
$40 < \mathbf{x} \leq 45$	$-2.63 < \mathbf{Z} \leq -2.03$	5587	4360
$45 < \mathbf{x} \leq 50$	$-2.03 < \mathbf{Z} \leq -1.43$	18248	8652
$50 < \mathbf{x} \leq 55$	$-1.43 < \mathbf{Z} \leq -.83$	41950	30663
$55 < \mathbf{x} \leq 60$	$-.83 < \mathbf{Z} \leq -.23$	67999	63045
$60 < \mathbf{x} \leq 65$	$-.23 < \mathbf{Z} \leq .37$	77784	89946
$65 < \mathbf{x} \leq 70$	$.37 < \mathbf{Z} \leq .97$	62710	74450
$70 < \mathbf{x} \leq 75$	$.97 < \mathbf{Z} \leq 1.58$	36000	37944
$75 < \mathbf{x} \leq 80$	$1.58 < \mathbf{Z} \leq 2.18$	14049	10105
$80 < \mathbf{x} \leq 85$	$2.18 < \mathbf{Z} \leq 2.78$	3934	2749
$85 < \mathbf{x}$	$2.78 < \mathbf{Z}$	893	1377

Table 1- Chi-Squared Data

The chi-squared statistic will tell us how closely these actual values are to the expected values in a normal distribution. The equation for this statistic is:

$$\chi^2 = \sum_{i=1}^{n} \frac{(O_i - E_i)}{E_i}$$

where O is the observed value and E is the expected value. With this calculation, we got a value of $\chi^2 = 38834.93$. We compare this to the test statistic, with 5% error, of 15.51, and, since our

chi-squared statistic is well above this value, we conclude that the speeds on these highways are not normally distributed.

This calculation tells us that our data cannot be predicted based on a normal curve, so we must find a new method to predict speeds in our simulation. After experimenting with other curves to fit the data, we chose to use the proportions given by the data as a guide for predicting the speeds of all of the cars.

CHAPTER 4
Computer Simulation

4.1. Basic Multi-Lane Simulation

Our first program simulates multiple lanes of one-way traffic. We used Mathematica to write and run this program. The code for the basic program is listed in Appendix A. In the basic simulation, passing rules are not followed. Vehicles are allowed to travel in any lane at any given speed, and if a vehicle intends to pass a slower vehicle in front, it may use any lane that is available. The data we will collect from each vehicle will be the total amount of time it took that vehicle to move over 52800 feet. We do not collect data from the vehicles that are initially in the simulation. Instead, we add vehicles according to a Poisson distribution and collect this information once they reach the end of the highway interval. Vehicles that reach the end of the highway are deleted from the simulation.

4.1.1. Setting the Initial Conditions

We begin by setting the initial parameters of the simulation: number of vehicles, number of lanes, and number of seconds. We then set parameters for each vehicle in the simulation.

We randomize the position of each of the initial vehicles and then check to make sure that two vehicles don't overlap on a 16 foot interval of highway. The vehicles are then sorted in descending position for each lane. They are sorted using Mathematica's automatic sorting feature.

We determine a car's desired velocity using proportions identified by research from the Indiana Department of Transportation and Purdue University. [1] We obtain a random number for each vehicle and assign it a speed according to the parameters shown in Table 2. The actual speed value is calculated by standardizing the random number over the given speed interval. For the

end speed intervals, we set a lower and upper speed limit. In the lowest interval, the lower limit is 30 mph. In the highest interval, the upper limit is 100 mph.

Speed Category	Parameter for Random Number, r
40 mph and below	$0 \leq r < .022$
41 to 45 mph	$.022 \leq r < .035$
46 to 50 mph	$.035 \leq r < .061$
51 to 55 mph	$.061 \leq r < .154$
56 to 60 mph	$.154 \leq r < .345$
61 to 65 mph	$.345 \leq r < .617$
66 to 70 mph	$.617 \leq r < .842$
71 to 75 mph	$.842 \leq r < .957$
76 to 80 mph	$.957 \leq r < .988$
81 to 85 mph	$.988 \leq r < .996$
86 mph and above	$.996 \leq r < 1$

Table 2- Parameters for Speed Category

4.1.2. Time Elapse

Once the vehicles have been ordered, we can begin to move them by lane. The vehicles move in one-second increments according to their speed. We move vehicles in descending order from the furthest position, beginning in the right-most lane. If a car's desired velocity moves it to a

position that still allows for a 16 foot space behind the vehicle in front of it, it is allowed to move at its desired velocity. If not, the program checks each adjacent lane to see if there is a gap that would accommodate the car and 16 feet of following distance. Since there are no passing rules, the program checks both right and left lanes, if they exist. If there is a gap, the car is moved to the new lane. If no gap exists, the car is forced to change its speed in order to remain 16 feet behind the car ahead.

New vehicles are added to the simulation by means of a Poisson distribution. The probability that x cars get added during a given second is:

$$P(x) = \frac{e^{-\lambda}\lambda^{x}}{x!},$$

where λ is the average traffic density per second, which can be calculated by:

$$\lambda = \frac{C}{10miles} * 61.9\frac{miles}{hour} * \frac{1}{3600},$$

where C is the initial number of cars and 61.9 miles/hour is the average speed for 55 mph highways. The positions of new cars are also randomized and checked for duplicates.

When a vehicle reaches 52800 feet or greater in position, the program checks to see whether the car was part of the original traffic density or was added during the running of the simulation. If the vehicle was in the simulation before the time elapse began, we will not include its data in the final analysis, and we simply delete its parameters from the simulation. We neglect these cars because they entered the stretch of road before the simulation began, and we cannot generalize about their behavior over that part of the road. Instead, they will create the desired amount of traffic for the vehicles whose behavior we are going to analyze.

If the vehicle was added at some time during the simulation, the program calculates and saves the time difference between its initial position, somewhere in the first 100 ft., and its new position,

somewhere greater than 52800 ft., and saves it as an average speed. It also saves the desired velocity for that vehicle. These two numbers will be used later to determine what proportion of its desired velocity at which the vehicle was able to travel. The other parameters for the vehicle are deleted from the simulation.

4.2. Improved Multi-Lane Simulation

This program was also written using Mathematica and the code can be found in Appendix B. This simulation also uses multiple lanes of one-way traffic. Again, passing is allowed, but vehicles are now forced to follow highway driving and passing rules. The rules used in this program are:

- All vehicles must drive in the right-most lane, if the space is available;
- If a vehicle intends to pass a slower-moving vehicle, it can only pass in the adjacent left lane, if there is space in that lane.

As in the basic simulation, the data we will collect from each vehicle will be the total amount of time it took that vehicle to move over 52800 feet. The program uses the same method to set initial conditions and add new cars, but uses different logic to move each car.

4.2.1. Time Elapse

Since the rule is that all vehicles must be in the rightmost lane if there is space available, we must begin by trying to move any vehicle that is not in the rightmost lane to its adjacent right lane. We also want to move the vehicle to the forward position according to its speed, so we will check the right lane to see if it has enough space for the vehicle to move forward. If the space is available, the car is moved to that position; if not, we will check directly in front of the car to see if there is room for it to move forward in its own lane. If there is room, it is moved to that position; if not, the program checks the adjacent left lane for the necessary space. If there is no space in the left

lane, the car is forced to stay in its own lane and reduce its speed so it stays one car length behind the slower car in front. This method is continued for every vehicle in the simulation.

When a vehicle reaches 52800 feet, the program follows the same procedure in the basic model for saving its average and desired speed.

CHAPTER 5
Results

5.1. Running the Simulation

We ran simulations over highways with two and three lanes, in each program. Simulations began at 100 vehicles, were increased by 25 vehicles per lane, and reached a total number of cars of at least 1000 vehicles. We obtained the average traffic speed and the proportion of average speed over desired speed for every run of the simulation. We will look at this proportion subtracted from one, which we will call the slow down, because it gives us information about how much traffic as a whole had to slow down from its desired speed due to congestion.

5.2. Linear Regression and R^2 Statistic

Statistics that will be useful for analyzing our data are a line of regression and the R^2 statistic. The line of regression fits the data with a line, so an approximate slope can be calculated. The R^2 statistic is related to this line because it tells us how well this line fits our data.

In order to calculate a line of regression, we minimize the square of the distance from each point to the line of regression. We use the equation,

$$D = \sum_{i=1}^{n} \left(y_i - f(x_i)\right)^2 ,$$

where y_i is the observed value and $f(x_i)$ is the value on the line of regression. We can square this equation to get rid of the square root, and do a substitution for,

$$f\left(x_i\right) = mx_i ,$$

since the equation for linear regression is of this form. In order to minimize the distance, we take the derivative with respect to m, the unknown constant, and set it equal to zero. This becomes,

$$\frac{dD}{dm} = \sum_{i=1}^{n} 2\left(y_i - mx_i\right) * \left(-x_i\right) = 0.$$

We then solve for m, which produces the equation,

$$m = \frac{\sum_{i=1}^{n} x_i y_i}{\sum_{i=1}^{n} x_i^2}.$$

Once we have obtained this line of regression, we can calculate how closely our data points follow this line, which is called the R^2 statistic. To calculate R-squared, we must first calculate the sum of the squares of error, or SSE. This gives us the residual between the y values of the observed points and the y values of the predicted points. The SSE is given by the equation,

$$SSE = \sum_{i=1}^{n} \left(y_i - \hat{y}_i \right)^2,$$

where $\hat{y}$ is the y-value of the predicted points.

Next we find the sum of squares of total, or SST. This statistic tells us how far away the observed y values fall from the mean of the predicted y-values. The SST is given by,

$$SST = \sum_{i=1}^{n} \left(y_i - \bar{y} \right)^2,$$

where $\bar{y}$ is the mean of the observed y-values.

When we divide the SSE by the SST, we get a proportion of how much the residuals make up the total discrepancy between the predicted and observed values. If we subtract this proportion from one, we get the R^2 statistic, or,

$$R^2 = 1 - \frac{SSE}{SST}.$$

This tells us that if the proportion SSE/SST is low, meaning the residuals between the predicted and observed y-values were smaller than the residuals between the observed data and its mean,

we get a higher R^2. If R^2 is closer to one, we know the line of regression fits the data well, and as R^2 gets closer to zero, the line of best fit is less accurate. If R^2 is not very good, we know that the data is not fit well by a line, and we would try some other regression to get a higher R^2.

5.3. Results for Two Lanes

5.3.1. Basic Model

For two lanes without passing rules, we plotted the proportion that traffic was slowed down, versus the traffic density. The data points resemble a linear function so we fitted them with such a function. We chose a linear function with a y-intercept at the origin because a traffic density that is very small or zero, should have a zero slow down proportion. This also gives us a common point for the graphs for each model, which will be useful in later statistical analysis. We also converted traffic density to cars per mile for easier comparison between two and three lanes. The data is plotted in Figure 3 with its linear regression.

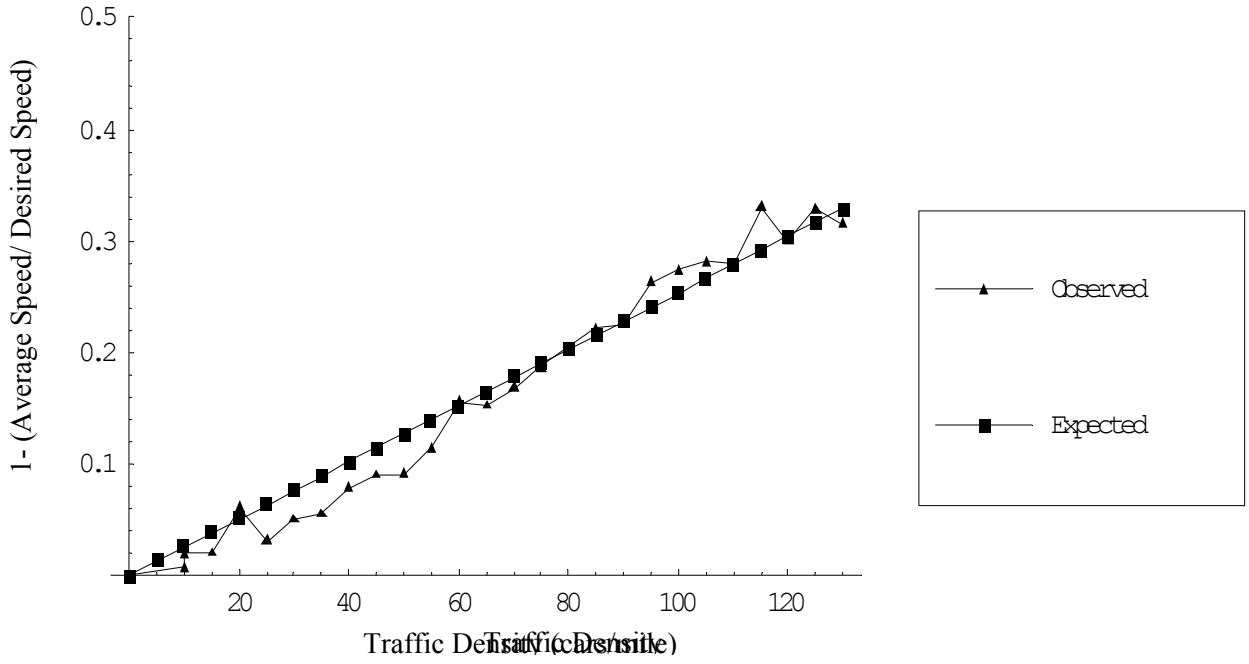

Figure 3- Basic Model, Two Lanes

We calculated a line of regression for this data which was y = .00253x . Since the slope of this line is positive, we know the relationship is positive, which tells us that for every increase in traffic density, there was a proportional increase in slow down.

We found an R^2 of .9801, meaning the data has a strong linear relationship and is approximated well by this linear function. This result tells us that in two lanes without passing rules, the slow down experienced by traffic has a strong linear relationship with the traffic density.

5.3.2. Improved Model

For two lanes with passing rules, we again plotted the proportion that the traffic was slowed down, versus the traffic density. The data points seemed to exhibit a linear relationship so we fitted the points with a linear function. We again used a linear function with a y-intercept at the origin. The plotted data and the linear regression are shown in Figure 4.

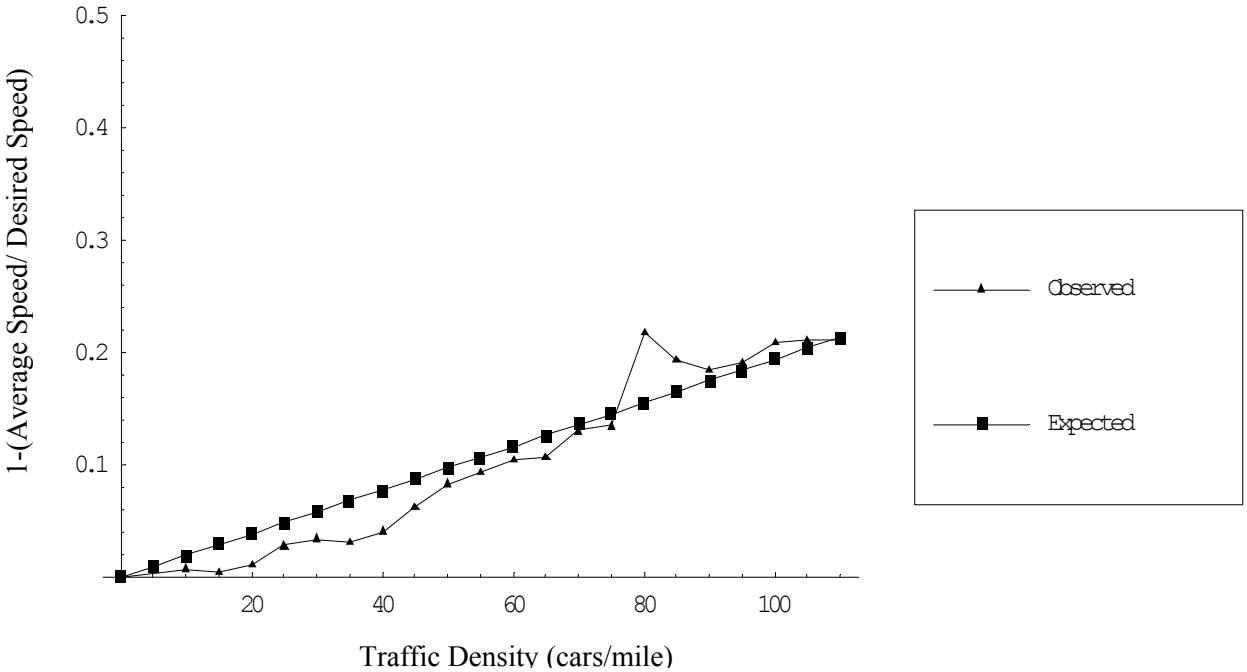

Figure 4- Improved Model, Two Lanes

The line of regression for this data was y = .00194x. The slope is positive, which again tells us of the positive relationship between traffic slow down and traffic density.

We calculated an R^2 of .9482, which tells us the line of regression fits the data pretty strongly. So, in two lanes, with passing rules, higher traffic density means vehicles must travel slower than they would like to.

5.4. Results for Three Lanes

5.4.1. Basic Model

We analyzed the results from simulations in three lanes with no passing rules in the same way as in two lanes. We plotted the slow down versus traffic density and found a line of regression that passed through the origin. This is shown in Figure 5.

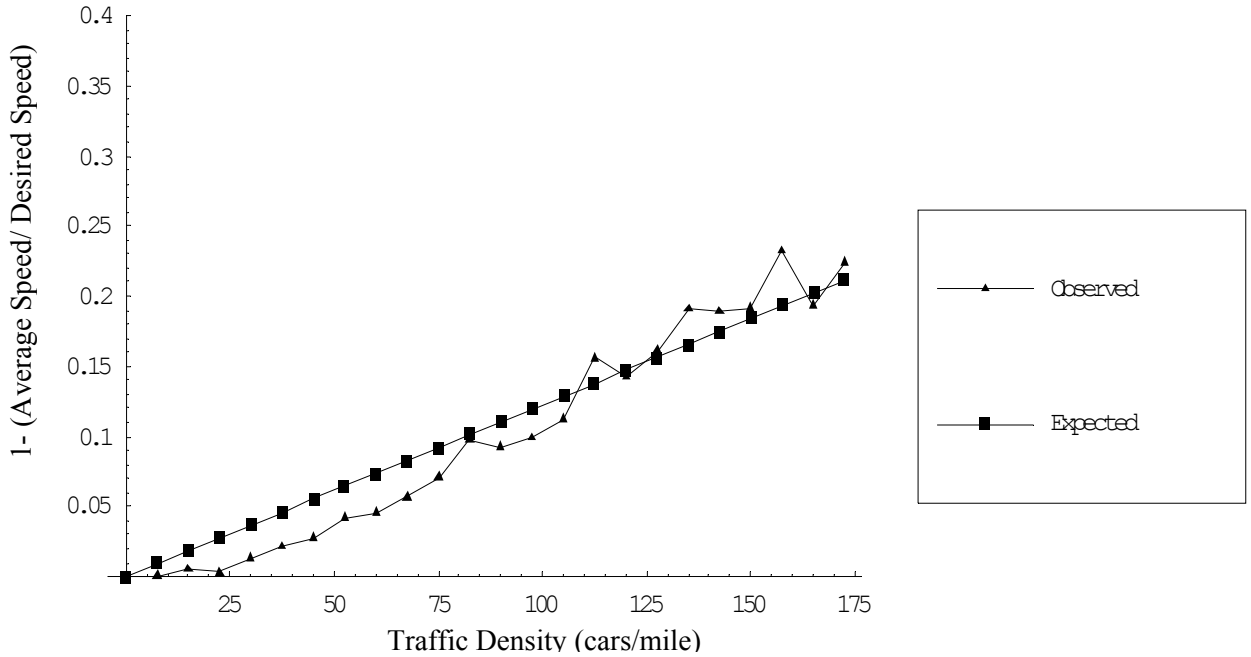

Figure 5- Basic Model, Three Lanes

The line of regression was y = .00123x, again showing an obviously positive relationship for three lanes without passing rules.

For this data, the R^2 statistic is .9668 so the linear relationship is strong. Once again, we see that increased traffic density causes an increased slow down proportion.

5.4.2. Improved Model

The data from three lanes with passing rules is shown below with the line of best fit. It also appears to have a linear relationship, which we will again verify with a calculation of R^2. The graph is shown in Figure 6.

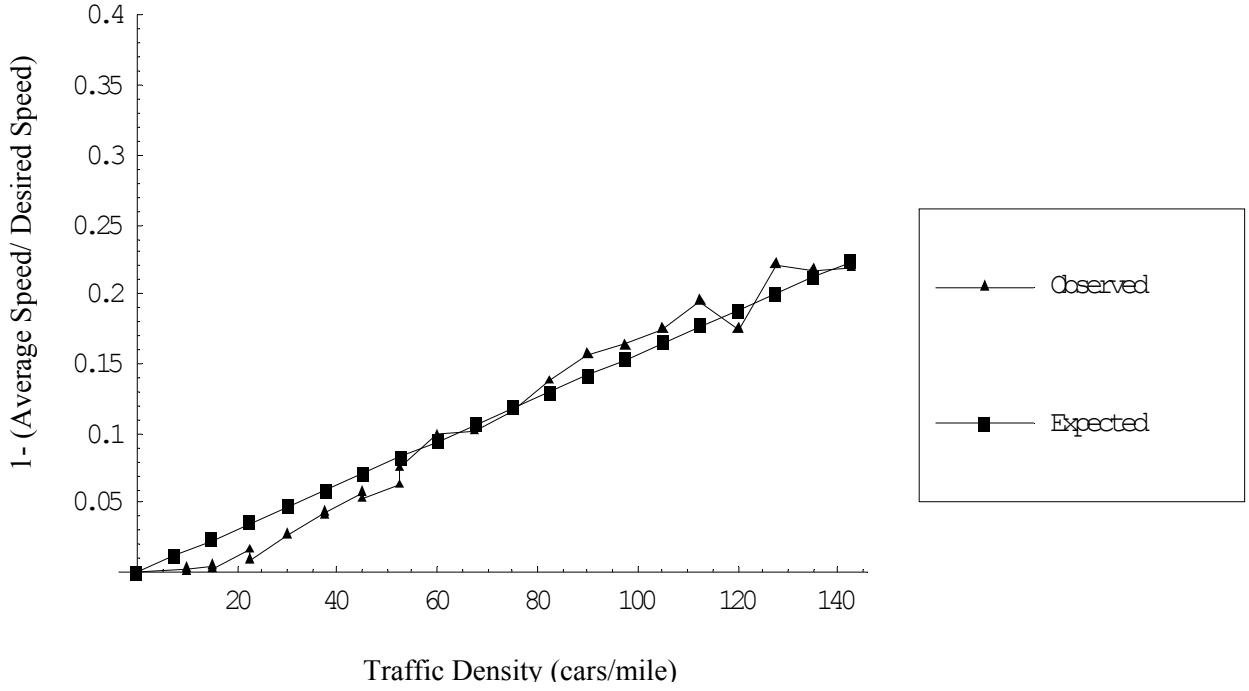

Figure 6- Improved Model, Three Lanes

The linear regression produced a line with the equation y = .00157x, so the positive relationship is still present in three lanes with passing rules.

The R^2 statistic is .9833 so this data fits the line of regression very accurately.

In all models for two and three lanes, the data exhibited a strong, positive linear relationship, so we can generalize these results. Using this simulation, we found that the slow down of traffic increases proportionally for every increase in traffic density. Now we would like to examine the effects of adding passing rules.

CHAPTER 6

Statistical Analysis of Results

6.1. Comparison of Passing Rules and No Passing Rules

One of the reasons for creating a model that uses passing rules and one that does not, was the purpose of comparing how these rules might affect the slow down of traffic at the same density. In order to see if these rules have an effect, or if any difference is due to chance, we must test if the difference between the two functions is statistically significant.

6.1.1. T-test

Since we are using a small sample, we will be using a t-test to look for statistical significance. The functions we are testing are the linear functions we obtained from the linear regression and are of the form:

$$y = \beta x,$$

where y is the slow down proportion, x is the traffic density, and β is the slope of the line of best fit. We want to see whether the slopes of the two functions are statistically significant. In order to use the t-test on each slope, we must normalize each slope to some $\hat{\beta}$. Once we find $\hat{\beta}$, we will calculate its mean and variance. To calculate $\hat{\beta}$, we want to minimize the distance from the data points to the line of regression. The equation for the distance from the data points to the line is given by:

$$L = \sum_{i=1}^{n} \left(y_i - \hat{\beta}_i x_i \right)^2 .$$

The derivative of this equation is:

$$\frac{dL}{d\hat{\beta}} = \sum_{i=1}^{n} -2x_i\left(y_i - \hat{\beta}x_i\right).$$

Setting this equal to zero and solving for $\hat{\beta}$, we get:

$$\hat{\beta} = \frac{\sum y_i x_i}{\sum x_i^2}. \quad (1)$$

The mean, or expected value, for the slope, is in fact, just the slope itself, β. The variance of $\hat{\beta}$ is the variance of equation (4).

$$V\left(\hat{\beta}\right) = V\left(\frac{\sum y_i x_i}{\sum x_i^2}\right).$$

We can simplify this down to,

$$V\left(\hat{\beta}\right) = \frac{1}{\left(\sum x_i^2\right)^2}\sum x_i^2 V(y).$$

We know the variance of y is the mean of the distance from the observed value of y to the function, which is given by,

$$V(y) = \frac{\sum_{i=1}^{n}\left(y_i - \hat{\beta}x_i\right)^2}{(n-1)},$$

where y_i is the observed value, $\hat{\beta}$ x_i is the expected value given by the function, and n is the total number of data points. After cancellation and substitution, we obtain a final equation for variance of $\hat{\beta}$,

$$V\left(\hat{\beta}\right) = \frac{\sum_{i=1}^{n}\left(y_i - \hat{\beta}x_i\right)^2}{(n-1)\left(\sum_{i=1}^{n} x_i^2\right)}. \quad (2)$$

Since we now know both the mean and variance for each slope β, we can use the t-test to see if the difference between the slopes is statistically significant. We are testing against the null hypothesis, that the two slopes are equal, or H_o: $\beta_1 = \beta_2$. We will obtain a test statistic and compare it to a given value for statistical significance. The formula for this test-statistic, t, is:

$$t_{2n-2} = \frac{\left(\hat{\beta}_1 - \hat{\beta}_2\right) - \left(\beta_1 - \beta_2\right)}{\sqrt{\left(\frac{V(y_1)}{\sum_{i=1}^{n} x_{i1}^2} + \frac{V(y_2)}{\sum_{i=1}^{n} x_{i2}^2}\right)}}, \quad (3)$$

where V(y) is given by equation (8) and, due to the null hypothesis, β_1-$\beta_2 = 0$. If this t-value is larger than the critical value given by the table, then we know the difference between the two slopes is statistically significant and the passing rules do have an effect on how much vehicles must slow down in a certain traffic density.

6.1.2. T-test for Two Lanes

To begin our analysis of how the addition passing rules affected traffic slow down, we graph the observed data from the basic and improved model, which is shown in Figure 7.

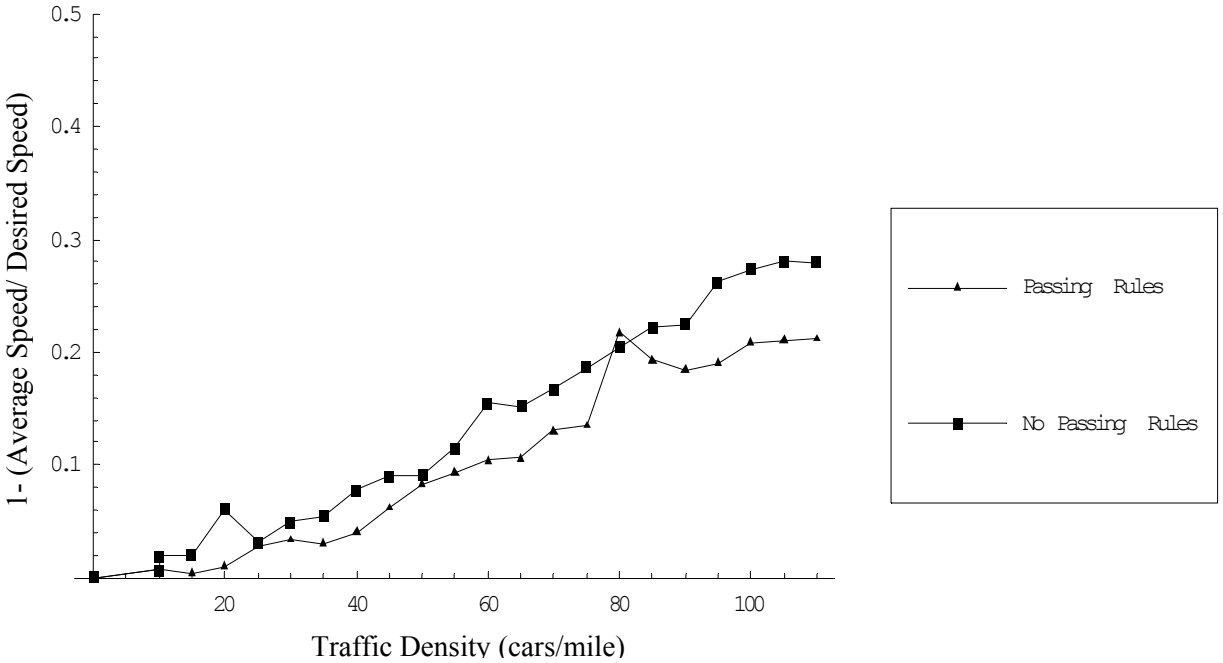

Figure 7- Basic and Improved Model for Two Lanes

From the graph, it appears that simulations with passing rules experienced a greater traffic slow down than simulations with no passing rules at the same traffic density. This implies that passing rules helped traffic flow better by allowing vehicles to go closer to their desired speed. In order to see if our intuition is correct, we must use the t-test to see if the difference is statistically significant.

We know that in equation (1), β is the value of the slope of the line of best fit we found for each model by using a linear regression. Next, we use equation (1) with the data from each model to find $\hat{\beta}$. In the basic model, we calculated $\hat{\beta}_B$ = .000249; in the improved model, we found $\hat{\beta}_I$ = .000194. We then used equation (2) to find the variance for each model's data points. In the basic model, $V(\hat{\beta}_B)$ = .000363; in the improved model $V(\hat{\beta}_I)$ = .000576. We then used

equation (3) with these four values to find a t-statistic of 5.5625. At a 95% significance level, the critical t-value for 40 degrees of freedom is 1.684. Since our t-statistic is above this level, our data is statistically significant, so we know that passing rules had a significant effect on traffic speed. We can conclude that in the case of two lanes, using passing rules cut down on traffic slow down significantly compared to not using passing rules at the same traffic density.

6.1.3. T-test for Three Lanes

We are also interested in the effect of implementing passing rules to traffic in three lanes. The graph of the basic and improved models in three lanes is shown below.

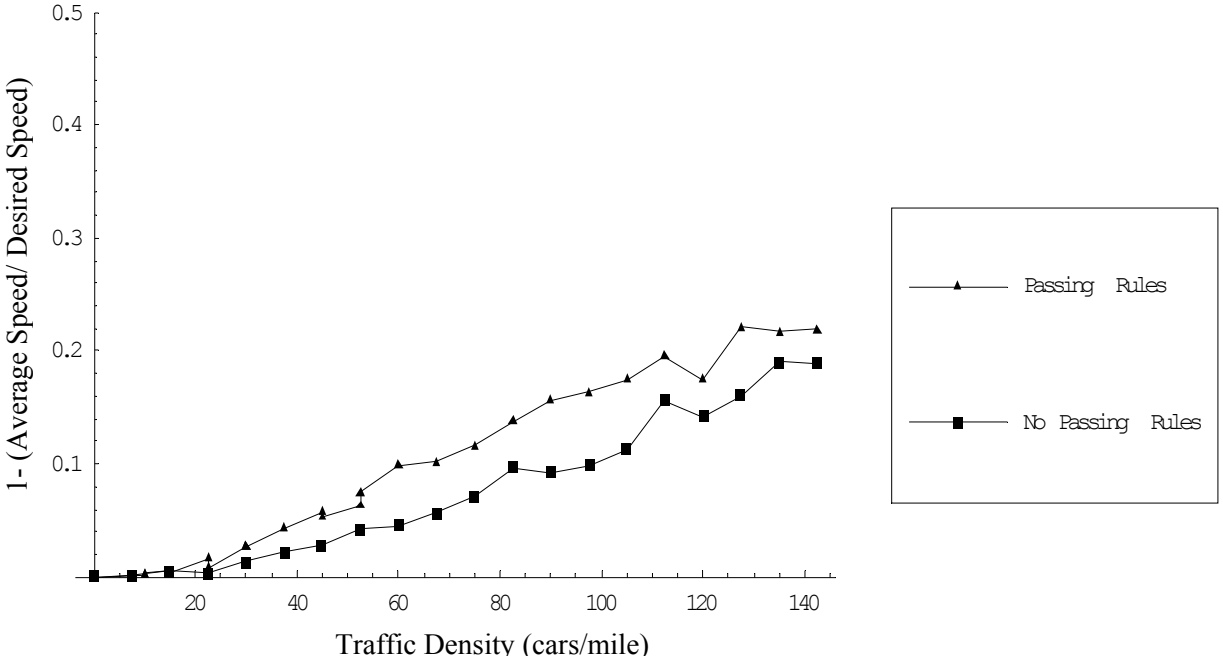

Figure 8- Basic and Improved Model for Three Lanes

This graph shows us that, in the case of three lanes, the situation seems to be reversed. From this graph, we see that simulations that were run with passing rules, tended to have a higher slow down proportion than simulations run without passing rules. This implies that passing rules slowed down traffic in three lanes. We will use the t-test to see if this is implication is statistically significant.

We find $\hat{\beta}_B = .000123$, $\hat{\beta}_I = .000157$, $V\left(\hat{\beta}_B\right) = .000418$, and $V\left(\hat{\beta}_I\right) = .000243$. We find a t-statistic of 5.92317, which, at a 95% confidence level, is significant.

This result tells us that, three lanes, there was a statistically significant difference between simulations with passing rules and simulations without passing rules. However, unlike the two

lane case, the result of adding passing rules was that traffic was slowed down more, instead of less. Possible reasons for this unusual result are discussed in the next chapter.

6.2. Comparison of Two and Three Lanes

We would also like to examine the effects of adding a third lane for the same traffic density in both the basic and improved models. We will use the same t-test to compare the slopes of each line of regression.

6.2.1. Basic Model

The data from the basic model in two lanes and three lanes is graphed in Figure 9.

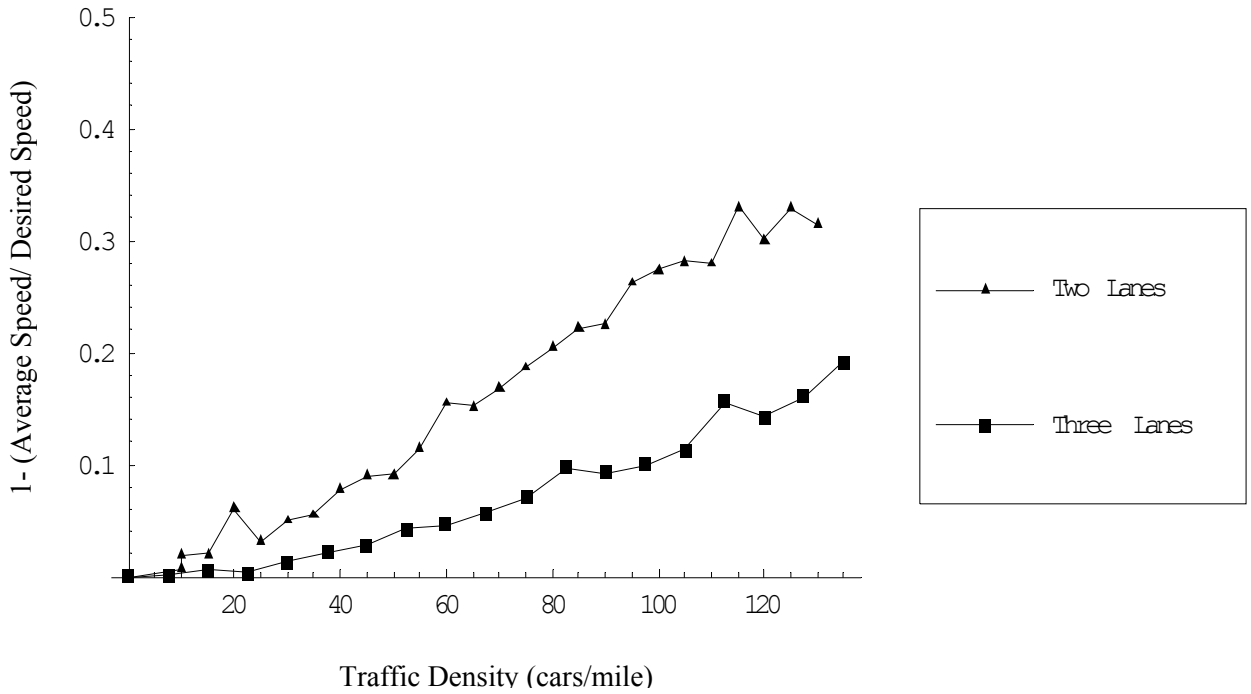

Figure 9- Basic Model in Two and Three Lanes

This graph shows that traffic in two lanes experienced a greater slow down than traffic in three lanes, when no passing rules were used. We used the t-test to see if this result is statistically significant.

We find $\hat{\beta}_2 = .00253$, $\hat{\beta}_3 = .00114$, $V(\hat{\beta}_2) = .00038$, and $V(\hat{\beta}_3) = .00036$. Using these values, we find a t-statistic of 18.6003. This result is highly significant for 43 degrees of freedom at 95% significance. Thus, adding a third lane had a very significant impact on increasing the overall speed of traffic, when no passing rules were used.

6.2.2. Improved Model

We did the same comparison for the model that incorporates passing rules. The graph of two and three lanes for this model is shown in Figure 10.

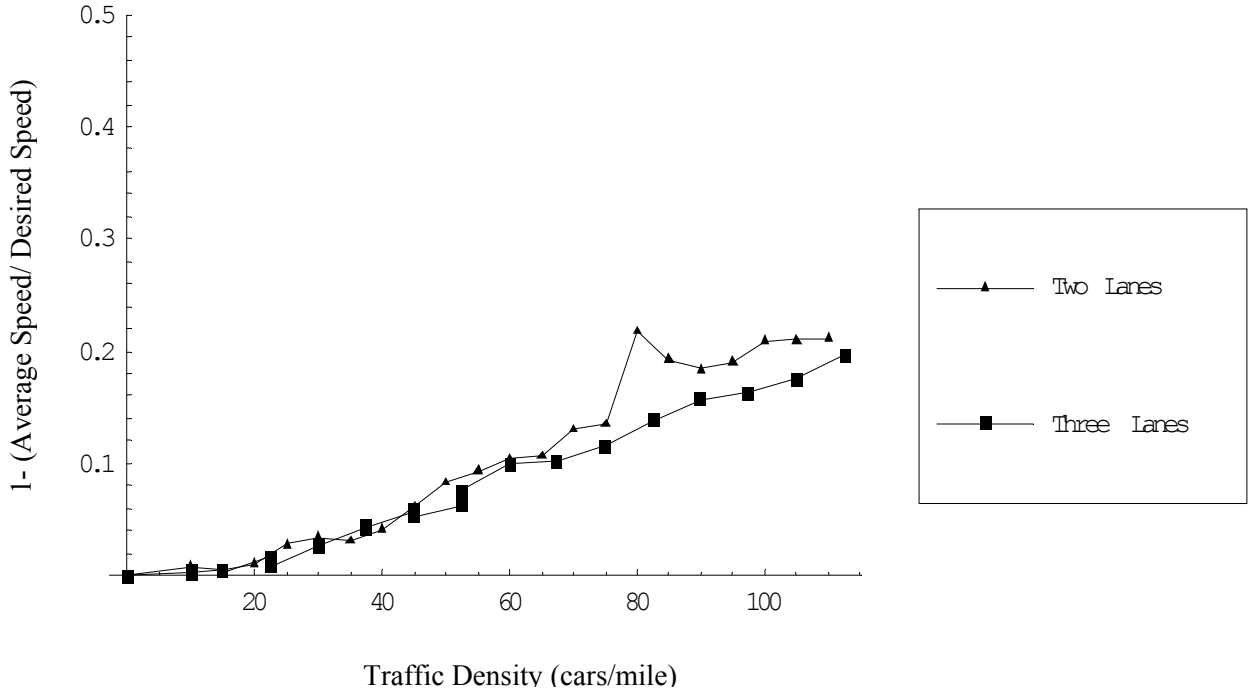

Figure 9- Improved Model in Two and Three Lanes

In the improved model, the data points for two and three lanes are much closer, although for higher traffic densities, the third lane still appears to decrease the slow down. We tested to see if these were significant.

We obtained $\hat{\beta}_2 = .00194$, $\hat{\beta}_3 = .00156$, $V\left(\hat{\beta}_2\right) = .00058$, and $V\left(\hat{\beta}_3\right) = .00025$. Using these values, we found a t-statistic of 3.9752 which is significant for 43 degrees of freedom at the 95% significance level. According to these statistics, adding a third lane also had a significant effect on traffic with passing rules and allowed traffic to travel at a higher speed.

CHAPTER 7
Discussion of the Model and Conclusions

7.1. Discussion of the Model

Our model was created to simulate a real-life phenomenon of how traffic moves on highways. The goal of this model was to recreate this behavior, and find a function that modeled it best. We were able to simulate how traffic moved along a highway and we found that it was fit very well by a linear regression. Since all of our R^2 statistics were very close to one, we know that the data we produced was modeled very well by a linear function.

Also, we were able to show that passing rules do have a significant effect on how well traffic flows. Based on our simulations, these passing rules were helpful in allowing two lanes of traffic to travel at higher speeds for higher densities. Unfortunately, these passing rules also significantly hurt three lanes of traffic, which is a weakness that will be discussed later.

We were also able to use our model to show the significance of adding a third lane of traffic. Adding this third lane significantly helped lower the slow down that the vehicles experienced, whether or not passing rules were implemented.

However, we built assumptions into the model in order to make simulation easier. These assumptions may have caused certain weaknesses in how true the model is to real life.

One of the weaknesses of our model was the use of passing rules. Our data for three lanes showed that passing rules slowed down the flow of traffic. We viewed an animation of a simulation that used passing rules in three lanes and found that traffic tended to be concentrated in the two right lanes, with few vehicles utilizing the left-most lane. This is due to our rule that traffic move to the right-most lane as much as possible. This causes an increased density in two lanes, which interferes with the ability of cars to pass each other and forces traffic to slow down. One possible solution to this problem would be to amend the passing rules for three or more lanes

so that vehicles tend to travel more in the center lanes, which would allow the left lanes to be used more for passing and would help to spread out the traffic, rather than concentrating it.

Another weakness was the assumption that all cars are exactly the same length. In real-life it is obvious that this assumption is false. Since our model deals with passing, this assumption affects the amount of time it takes for one vehicle to pass another. Passing a shorter vehicle would take less time than passing a longer vehicle, so changing this assumption could possibly have an effect on the slow down of traffic. One possible solution would be setting vehicle length as a parameter for each car. The length of a vehicle could be assigned using some appropriate distribution. This length would then have an impact on whether or not a vehicle was allowed to change lanes.

7.2. Conclusions

Based on this model, we were able to conclude that the traffic slow down proportion and traffic density have a positive, linear relationship. We also found that passing rules have a significant effect on the traffic slow down proportion, although this effect is not clear for all situations. We are also able to find that adding a third lane has a significant effect on improving traffic flow. We can speculate that this result is true for higher numbers of lanes, but more simulations are required to test this theory.

In order to compare our results to those obtained by partial differential equations, we graph our data in terms of traffic speed versus traffic density and compare it to the distribution found by Del Castillo in [7]. This type of graph for the improved model over three lanes is shown in figure 9.

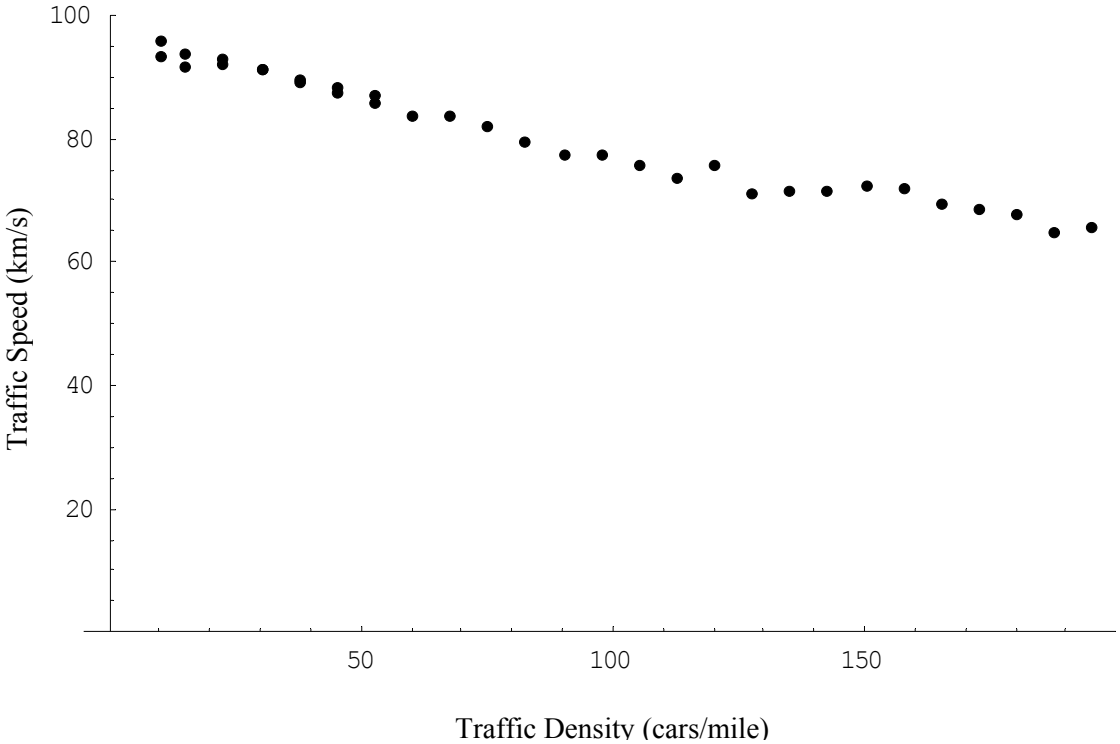

Figure 10- Improved Model, Three Lanes

This graph shows that our data follows a linear relationship instead of the curved, nonlinear relationship found by Del Castillo. However, we have not reached a traffic density where the traffic speed was close to zero. This means that more runs of the simulation may produce a more non-linear relationship.

Also, our data does not show a critical density that causes the traffic speed to decrease sharply. The critical density in Huberman and Helbing's model was 35 vehicles per mile of highway, where traffic became very fragile and unstable. Our simulation does not produce a critical density at this point, so our model does not replicate the results of Huberman and Helbing. Further runs of the simulation may have produced a different critical density for our model.

Even though our model does not replicate the results of Del Castillo or Huberman and Helbing, it gives interesting results of a linear relationship between average traffic speed and traffic density.

APPENDIX A.

Code for Basic Model (Mathematica)

```
//Creating and initializing variables
numcars = 100;
numlanes = 2;
maxtime = 1300;
nplace = Table[52800*Random[], {i, 1, numcars}];
lane = Table[Random[Integer, {1, numlanes}], {i, 1, numcars}];
speed = Table[a = Random[];
      Which[a < .022, 44 + (2000/3)*a,
     a < .035, (176/3) + (22/3)*a,
      a < .061, 66 + (22/3)*a,
      a < .154, (220/3) + (22/3)*a,
      a < .345, (242/3) + (22/3)*a,
      a < .617, 88 + (22/3)*a,
      a < .842, (286/3) + (22/3)*a,
      a < .957, (308/3) + (22/3)*a,
      a < .988, 110 + (22/3)*a,
      a < .996, (352/3) + (22/3)*a,
      a < 1, (374/3) + 22*a], {i, 1, numcars}];
timein[k_] := Table[0, {Length[lanes[k]]}];
timeout[k_] := Table[0, {Length[lanes[k]]}];
avgspeed := {};
newlane := {};
carsadded = 0;
carsleft = 0;
savedvelocity := {};
newpos := {};

//Sorting the vehicles
lanes[i_] := nplace[[Flatten[Position[lane, i]]]];
order[k_] := Reverse[Sort[lanes[k]]];
initialorder[k_] := Reverse[Sort[lanes[k]]];
velocity[k_] := Reverse[Sort[lanes[k]]];
numbers[k_] := Length[lanes[k]];
```

```
//Assigning velocities to respective vehicles
For[L = 1, L ≤ numlanes, L++,
  For[c = 1, c ≤ Length[lanes[L]], c++,
    For[k = 1, k ≤ numcars, k++,
      If[order[L][[c]] == nplace[[k]],
        velocity[L] = ReplacePart[velocity[L], speed[[k]], c]]]]];

//Checking for duplicate positions
For[L = 1, L ≤ numlanes, L++,
  For[c = 1, c ≤ numbers[L], c++,
    While[Select[order[L], (order[L][[c]] - 32) < # < (order[L][[c]]) &] != {},
      x = 52800*Random[];
      order[L] = ReplacePart[order[L], x, c];
      initialorder[L] = ReplacePart[initialorder[L], x, c];
      w = Transpose[Reverse[Sort[Transpose[{order[L], velocity[L],
      initialorder[L]}]]]];
      order[L] = w[[1]];
      velocity[L] = w[[2]];
      initialorder[L] = w[[3]]]]];

//Moving the vehicles
For[t = 0, t ≤ maxtime, t++,
  For[L = 1, L ≤ numlanes, L++,
    For[c = 1, c ≤ numbers[L], c++,
      If[c == 1,
        If[(velocity[L][[c]] + order[L][[c]]) ≤ 52800,
          order[L] = ReplacePart[
      order[L], (order[L][[c]] + velocity[L][[c]]), c],
//Saving a vehicle's statistics
          If[(timein[L][[c]]) > 0,
            timeout[L] = ReplacePart[timeout[L], t, c];
            AppendTo[savedvelocity, velocity[L][[c]]];
            AppendTo[avgspeed, ((order[L][[c]] + velocity[L][[c]] –
                    initialorder[L][[c]])/(timeout[L][[c]] - timein[L][[c]]))];
          carsleft = carsleft + 1;
          numbers[L] = numbers[L] - 1;
          velocity[L] = Drop[velocity[L], {c}];
```

```
        initialorder[L] = Drop[initialorder[L], {c}];
        order[L] = Drop[order[L], {c}];
        timein[L] = Drop[timein[L], {c}];
        timeout[L] = Drop[timeout[L], {c}];
        c = c - 1],
      If[(velocity[L][[c]] + order[L][[c]]) ≤ (order[L][[c - 1]] - 32),
//Moving a vehicle forward to its desired position in its own lane
        order[L] = ReplacePart[order[L], (velocity[L][[
          c]] + order[L][[c]]), c],
//Checking the adjacent lane, if vehicle is unable to move to desired position
        If[L == 1,
//In lane 1, checking the adjacent lane
          If[Select[order[L + 1], (order[L][[c]] - 32) < # < (order[L][[c]] +
                velocity[L][[c]]) &] == {},
            AppendTo[timein[L + 1], timein[L][[c]]];
            AppendTo[timeout[L + 1], timeout[L][[c]]];
            AppendTo[order[L + 1], order[L][[c]]];
            AppendTo[initialorder[L + 1], initialorder[L][[c]]];
            AppendTo[velocity[L + 1], velocity[L][[c]]];
            numbers[L + 1] = numbers[L + 1] + 1;
            timein[L] = Drop[timein[L], {c}];
            timeout[L] = Drop[timeout[L], {c}];
            order[L] = Drop[order[L], {c}];
            initialorder[L] = Drop[initialorder[L], {c}];
            numbers[L] = numbers[L] - 1;
            velocity[L] = Drop[velocity[L], {c}];
            y = Transpose[Reverse[Sort[Transpose[{order[L + 1], velocity[L + 1], timein[L + 1],
                  initialorder[L + 1], timeout[L + 1]}]]]];
            order[L + 1] = y[[1]];
            velocity[L + 1] = y[[2]];
            timein[L + 1] = y[[3]];
            initialorder[L + 1] = y[[4]];
            timeout[L + 1] = y[[5]];
            c = c - 1,
            order[L] = ReplacePart[order[L], (order[L][[c - 1]] - 32), c]],
          If[L == numlanes,
```

```
//In the leftmost lane, checking the adjacent lane
        If[Select[order[L - 1], (order[L][[c]] - 32) < # < (order[L][[c]] +
              velocity[L][[c]]) &] == {},
          AppendTo[timein[L - 1], timein[L][[c]]];
          AppendTo[timeout[L - 1], timeout[L][[c]]];
          AppendTo[order[L - 1], (order[L][[c]] + velocity[L][[c]])];
          AppendTo[initialorder[L - 1], initialorder[L][[c]]];
          AppendTo[velocity[L - 1], velocity[L][[c]]];
          numbers[L - 1] = numbers[L - 1] + 1;
          timein[L] = Drop[timein[L], {c}];
          timeout[L] = Drop[timeout[L], {c}];
          order[L] = Drop[order[L], {c}];
          initialorder[L] = Drop[initialorder[L], {c}];
          numbers[L] = numbers[L] - 1;
          velocity[L] = Drop[velocity[L], {c}];
          w = Transpose[Reverse[Sort[Transpose[{order[L - 1], velocity[L - 1], timein[L - 1],
                initialorder[L - 1], timeout[L - 1]}]]]];
          order[L - 1] = w[[1]];
          velocity[L - 1] = w[[2]];
          timein[L - 1] = w[[3]];
          initialorder[L - 1] = w[[4]];
          timeout[L - 1] = w[[5]];
          c = c - 1,
          order[L] = ReplacePart[order[L], (
          order[L][[c - 1]] - 32), c]],
//In any middle lane, checking left adjacent lane
        If[Select[order[L + 1], (order[L][[c]] - 32) < # < (order[L][[c]] +
              velocity[L][[c]]) &] == {},
          AppendTo[timein[L + 1], timein[L][[c]]];
          AppendTo[timeout[L + 1], timeout[L][[c]]];
          AppendTo[order[L + 1], order[L][[c]]];
          AppendTo[initialorder[L + 1], initialorder[L][[c]]];
          AppendTo[velocity[L + 1], velocity[L][[c]]];
          numbers[L + 1] = numbers[L + 1] + 1;
          timein[L] = Drop[timein[L], {c}];
          timeout[L] = Drop[timeout[L], {c}];
          order[L] = Drop[order[L], {c}];
```

```
        initialorder[L] = Drop[initialorder[L], {c}];
        velocity[L] = Drop[velocity[L], {c}];
        numbers[L] = numbers[L] - 1;
        u = Transpose[Reverse[Sort[Transpose[{order[L + 1], velocity[L + 1], timein[L + 1],
              initialorder[L + 1], timeout[L + 1]}]]]];
        order[L + 1] = u[[1]];
        velocity[L + 1] = u[[2]];
        timein[L + 1] = u[[3]];
        initialorder[L + 1] = u[[4]];
        timeout[L + 1] = u[[5]];
        c = c - 1,
```

//In any middle lane, checking right adjacent lane

```
        If[Select[order[L - 1], (order[L][[c]] - 32) < # < (order[L][[c]] +
              velocity[L][[c]]) &] == {},
          AppendTo[timein[L - 1], timein[L][[c]]];
          AppendTo[timeout[L - 1], timeout[L][[c]]];
          AppendTo[order[L - 1], (order[L][[c]] + velocity[L][[c]])];
          AppendTo[initialorder[L - 1], initialorder[L][[c]]];
          AppendTo[velocity[L - 1], velocity[L][[c]]];
          numbers[L - 1] = numbers[L - 1] + 1;
          timein[L] = Drop[timein[L], {c}];
          timeout[L] = Drop[timeout[L], {c}];
          order[L] = Drop[order[L], {c}];
          initialorder[L] = Drop[initialorder[L], {c}];
          velocity[L] = Drop[velocity[L], {c}];
          numbers[L] = numbers[L] - 1;
          z = Transpose[Reverse[Sort[Transpose[{order[L -1], velocity[L - 1], timein[L - 1],
              initialorder[L - 1], timeout[L - 1]}]]]];
          order[L - 1] = z[[1]];
          velocity[L - 1] = z[[2]];
          timein[L - 1] = z[[3]];
          initialorder[L - 1] = z[[4]];
          timeout[L - 1] = z[[5]];
          c = c - 1,
```

//If no space is available in other lanes, moving vehicle to new position behind slower vehicle

```
        order[L] = ReplacePart[order[L], (order[L][[c - 1]] - 32),
        c]]]]]]]]];
```

```
//Adding new cars
  lambda = (numcars/10)*(61.9)*(1/3600);
  r = Random[];
  newcars = 0;
  newlane = {};
  newpos = {};
  While[r > (Sum[(E^(-lambda)*lambda^j)/(j!), {j, 0, newcars}]), newcars++];
  carsadded = carsadded + newcars;
//Checking for duplicate positions
  If[newcars ≠ 0,
   For[k = 1, k ≤ newcars, k++,
    AppendTo[newlane, Random[Integer, {1, numlanes}]];
    AppendTo[newpos, 100*Random[]];
    While[Select[order[newlane[[k]]], (newpos[[k]] -
          32) < # < (newpos[[k]]) &] ≠ {},
     newlane = ReplacePart[newlane, Random[Integer, {1, 2}], k];
     newpos = ReplacePart[newpos, 120*Random[], k]];
    AppendTo[order[newlane[[k]]], newpos[[k]]];
    AppendTo[initialorder[newlane[[k]]], newpos[[k]]];
    AppendTo[timein[newlane[[k]]], t];
    AppendTo[timeout[newlane[[k]]], 0];
    AppendTo[velocity[newlane[[k]]], a = Random[];
     Which[a < .022, 44 + (2000/3)*a,
          a < .035, (176/3) + (22/3)*a,
          a < .061, 66 + (22/3)*a,
          a < .154, (220/3) + (22/3)*a,
          a < .345, (242/3) + (22/3)*a,
          a < .617, 88 + (22/3)*a,
          a < .842, (286/3) + (22/3)*a,
          a < .957, (308/3) + (22/3)*a,
          a < .988, 110 + (22/3)*a,
          a < .996, (352/3) + (22/3)*a,
          a < 1, (374/3) + 22*a]];
//Sorting new cars into respective lanes
     x = Transpose[Reverse[Sort[Transpose[{order[newlane[[k]]], velocity[newlane[[k]]],
          initialorder[newlane[[k]]], timein[newlane[[k]]], timeout[newlane[[k]]]}]]];
     order[newlane[[k]]] = x[[1]];
```

```
velocity[newlane[[k]]] = x[[2]];
initialorder[newlane[[k]]] = x[[3]];
timein[newlane[[k]]] = x[[4]];
timeout[newlane[[k]]] = x[[5]];
numbers[newlane[[k]]] = numbers[newlane[[k]]] + 1]]]
```

APPENDIX B.

Code for Improved Model (Mathematica)

```
//Creating and initializing variables
numcars = 100;
numlanes=2;
maxtime=1300;
nplace=Table[52800*Random[],{i,1,numcars}];
lane=Table[Random[Integer,{1,numlanes}],{i,1,numcars}];
speed=Table[a=Random[];
    Which[a<.022, 44+(2000/3)*a,
    a<.035, (176/3)+(22/3)*a,
    a<.061, 66+(22/3)*a,
    a<.154, (220/3)+(22/3)*a,
    a<.345, (242/3)+(22/3)*a,
    a<.617, 88+(22/3)*a,
    a<.842, (286/3)+(22/3)*a,
    a<.957, (308/3)+(22/3)*a,
    a<.988, 110+(22/3)*a,
    a<.996, (352/3)+(22/3)*a,
    a<1, (374/3)+22*a],{i,1,numcars}];
timein[k_]:=Table[0,{Length[lanes[k]]}]
timeout[k_]:=Table[0,{Length[lanes[k]]}]
avgspeed:={}
newlane:={}
carsadded=0;
carsleft=0;
savedvelocity:={}
newpos:={}

//Sorting vehicles
lanes[i_]:=nplace[[Flatten[Position[lane,i]]]]
order[k_]:=Reverse[Sort[lanes[k]]]
initialorder[k_]:=Reverse[Sort[lanes[k]]]
velocity[k_]:=Reverse[Sort[lanes[k]]]
numbers[k_]:=Length[lanes[k]]
```

```
//Assigning velocities to respective vehicles
For[L=1, L ≤ numlanes, L++,
 For[c=1, c ≤ Length[lanes[L]], c++,
  For[k=1, k ≤ numcars, k++,
   If[order[L][[c]]==nplace[[k]],
    velocity[L]=ReplacePart[velocity[L],speed[[k]],c]]]]]

//Checking for duplicate positions
For[L=1, L ≤ numlanes, L++,
 For[c=1, c ≤ numbers[L], c++,
  While[Select[order[L], (order[L][[c]]-32)<#<(order[L][[c]])&]!={},
   x=52800*Random[];
   order[L]=ReplacePart[order[L], x, c];
   initialorder[L]=ReplacePart[initialorder[L], x, c];
   w=Transpose[Reverse[Sort[Transpose[{order[L],velocity[L],initialorder[L]}]]]];
   order[L]=w[[1]];
   velocity[L]=w[[2]];
   initialorder[L]=w[[3]]]]]

//Moving the vehicles
For[t = 0, t ≤ maxtime, t++,
 For[L = 1, L ≤ numlanes, L++,
  For[c = 1, c ≤ numbers[L], c++,
   If[c == 1,
    If[(velocity[L][[c]] + order[L][[c]]) ≤ 52800,
     If[L == 1,
        order[L] = ReplacePart[order[L], (order[L][[c]] + velocity[L][[c]]), c],
        If[Select[order[L - 1], (order[L][[c]] + velocity[L][[c]] - 32) < # < (order[L][[c]]
             + velocity[L][[c]]) &] == {},
        AppendTo[timein[L - 1], timein[L][[c]]];
        AppendTo[timeout[L - 1], timeout[L][[c]]];
        AppendTo[order[L - 1], (order[L][[c]] + velocity[L][[c]])];
        AppendTo[initialorder[L - 1], initialorder[L][[c]]];
        AppendTo[velocity[L - 1], velocity[L][[c]]];
        numbers[L - 1] = numbers[L - 1] + 1;
        timein[L] = Drop[timein[L], {c}];
        timeout[L] = Drop[timeout[L], {c}];
```

```
        order[L] = Drop[order[L], {c}];
        initialorder[L] = Drop[initialorder[L], {c}];
        numbers[L] = numbers[L] - 1;
        velocity[L] = Drop[velocity[L], {c}];
        w = Transpose[Reverse[Sort[Transpose[{order[L - 1], velocity[L - 1], timein[L - 1],
              initialorder[L - 1], timeout[L - 1]}]]]];
        order[L - 1] = w[[1]];
        velocity[L - 1] = w[[2]];
        timein[L - 1] = w[[3]];
        initialorder[L - 1] = w[[4]];
        timeout[L - 1] = w[[5]];
        c = c - 1,
        order[L] = ReplacePart[order[L], (order[L][[c]] + velocity[L][[c]]), c]]],
//Saving a vehicle's statistics
      If[(timein[L][[c]]) > 0,
       timeout[L] = ReplacePart[timeout[L], t, c];
       AppendTo[savedvelocity, velocity[L][[c]]];
       AppendTo[avgspeed, ((order[L][[c]] + velocity[L][[c]] -
        initialorder[L][[c]])/(timeout[L][[c]] - timein[L][[c]]))]];
      carsleft = carsleft + 1;
      numbers[L] = numbers[L] - 1;
      velocity[L] = Drop[velocity[L], {c}];
      initialorder[L] = Drop[initialorder[L], {c}];
      order[L] = Drop[order[L], {c}];
      timein[L] = Drop[timein[L], {c}];
      timeout[L] = Drop[timeout[L], {c}];
      c = c - 1],
     If[L == 1,
//In the rightmost lane, moving a vehicle to its desired position
      If[(velocity[L][[c]] + order[L][[c]]) ≤ (order[L][[c - 1]] - 32),
       order[L] = ReplacePart[order[L], (velocity[L][[c]] + order[L][[c]]), c],
//In the rightmost lane, checking the left adjacent lane
       If[Select[order[L + 1], (order[L][[c]] - 32) < # < (order[L][[c]] +
            velocity[L][[c]]) &] == {},
        AppendTo[timein[L + 1], timein[L][[c]]];
        AppendTo[timeout[L + 1], timeout[L][[c]]];
        AppendTo[order[L + 1], order[L][[c]]];
```

```
        AppendTo[initialorder[L + 1], initialorder[L][[c]]];
        AppendTo[velocity[L + 1], velocity[L][[c]]];
        numbers[L + 1] = numbers[L + 1] + 1;
        timein[L] = Drop[timein[L], {c}];
        timeout[L] = Drop[timeout[L], {c}];
        order[L] = Drop[order[L], {c}];
        initialorder[L] = Drop[initialorder[L], {c}];
        numbers[L] = numbers[L] - 1;
        velocity[L] = Drop[velocity[L], {c}];
        y = Transpose[Reverse[Sort[Transpose[{order[L + 1], velocity[L + 1], timein[L + 1],
            initialorder[L + 1],timeout[L + 1]}]]]];
        order[L + 1] = y[[1]];
        velocity[L + 1] = y[[2]];
        timein[L + 1] = y[[3]];
        initialorder[L + 1] = y[[4]];
        timeout[L + 1] = y[[5]];
        c = c - 1,
//In the rightmost lane, moving to a position behind the slower vehicle
        order[L] = ReplacePart[order[L], (order[L][[c - 1]] - 32), c]]],
      If[L == numlanes,
//In the leftmost lane, attempting to move to the right adjacent lane
       If[Select[order[L - 1], (order[L][[c]] + velocity[L][[c]] - 32) < # < (order[L][[c]] +
            velocity[L][[c]]) &] == {},
        AppendTo[timein[L - 1], timein[L][[c]]];
        AppendTo[timeout[L - 1], timeout[L][[c]]];
        AppendTo[order[L - 1], (order[L][[c]] + velocity[L][[c]])];
        AppendTo[initialorder[L - 1], initialorder[L][[c]]];
        AppendTo[velocity[L - 1], velocity[L][[c]]];
        numbers[L - 1] = numbers[L - 1] + 1;
        timein[L] = Drop[timein[L], {c}];
        timeout[L] = Drop[timeout[L], {c}];
        order[L] = Drop[order[L], {c}];
        initialorder[L] = Drop[initialorder[L], {c}];
        numbers[L] = numbers[L] - 1;
        velocity[L] = Drop[velocity[L], {c}];
        w = Transpose[Reverse[Sort[Transpose[{order[L - 1], velocity[L - 1], timein[L - 1],
            initialorder[L - 1], timeout[L - 1]}]]]];
```

```
        order[L - 1] = w[[1]];
        velocity[L - 1] = w[[2]];
        timein[L - 1] = w[[3]];
        initialorder[L - 1] = w[[4]];
        timeout[L - 1] = w[[5]];
        c = c - 1,
//In the leftmost lane, moving to the desired position
        If[(order[L][[c]] + velocity[L][[c]]) ≤ (order[L][[c - 1]] - 32),
          order[L] = ReplacePart[order[L], (order[L][[c]] + velocity[L][[c]]), c],
//In the leftmost lane, moving to a position behind the slower vehicle
          order[L] = ReplacePart[order[L], (order[L][[c - 1]] - 32), c]]],
//In any middle lane, attempting to move to the right lane
       If[Select[order[L - 1], (order[L][[c]] + velocity[L][[c]] - 32) < # < (order[L][[c]] +
            velocity[L][[c]]) &] == {},
        AppendTo[timein[L - 1], timein[L][[c]]];
        AppendTo[timeout[L - 1], timeout[L][[c]]];
        AppendTo[order[L - 1], (order[L][[c]] + velocity[L][[c]])];
        AppendTo[initialorder[L - 1], initialorder[L][[c]]];
        AppendTo[velocity[L - 1], velocity[L][[c]]];
        numbers[L - 1] = numbers[L - 1] + 1;
        timein[L] = Drop[timein[L], {c}];
        timeout[L] = Drop[timeout[L], {c}];
        order[L] = Drop[order[L], {c}];
        initialorder[L] = Drop[initialorder[L], {c}];
        velocity[L] = Drop[velocity[L], {c}];
        numbers[L] = numbers[L] - 1;
        z = Transpose[Reverse[Sort[Transpose[{order[L - 1], velocity[L - 1], timein[L - 1],
            initialorder[L - 1], timeout[L - 1]}]]]];
        order[L - 1] = z[[1]];
        velocity[L - 1] = z[[2]];
        timein[L - 1] = z[[3]];
        initialorder[L - 1] = z[[4]];
        timeout[L - 1] = z[[5]];
        c = c - 1,
//In any middle lane, moving to the desired position
        If[(order[L][[c]] + velocity[L][[c]]) ≤ (order[L][[c - 1]] - 32),
          order[L] = ReplacePart[order[L], (order[L][[c]] + velocity[L][[c]]), c],
```

```
//In any middle lane, checking the left adjacent lane
            If[Select[order[L + 1], (order[L][[c]] - 32) < # < (order[L][[c]] +
                  velocity[L][[c]]) &] == {},
              AppendTo[timein[L + 1], timein[L][[c]]];
              AppendTo[timeout[L + 1], timeout[L][[c]]];
              AppendTo[order[L + 1], order[L][[c]]];
              AppendTo[initialorder[L + 1], initialorder[L][[c]]];
              AppendTo[velocity[L + 1], velocity[L][[c]]];
              numbers[L + 1] = numbers[L + 1] + 1;
              timein[L] = Drop[timein[L], {c}];
              timeout[L] = Drop[timeout[L], {c}];
              order[L] = Drop[order[L], {c}];
              initialorder[L] = Drop[initialorder[L], {c}];
              velocity[L] = Drop[velocity[L], {c}];
              numbers[L] = numbers[L] - 1;
              u = Transpose[Reverse[Sort[Transpose[{order[L + 1], velocity[L + 1],
                  timein[L + 1], initialorder[L + 1], timeout[L + 1]}]]]];
              order[L + 1] = u[[1]];
              velocity[L + 1] = u[[2]];
              timein[L + 1] = u[[3]];
              initialorder[L + 1] = u[[4]];
              timeout[L + 1] = u[[5]];
              c = c - 1,
//In any middle lane, moving to a position behind the slower vehicle
              order[L] = ReplacePart[order[L], (order[L][[c - 1]] - 32), c]]]]]]]];
//Adding new cars
  lambda = (numcars/10)*(61.9)*(1/3600);
  r = Random[];
  newcars = 0;
  newlane = {};
  newpos = {};
  While[r > (Sum[(E^(-lambda)*lambda^j)/(j!), {j, 0, newcars}]), newcars++];
  carsadded = carsadded + newcars;
  If[newcars ≠ 0,
//Checking for duplicate positions
    For[k = 1, k ≤ newcars, k++,
      AppendTo[newlane, Random[Integer, {1, numlanes}]];
```

```
    AppendTo[newpos, 100*Random[]];
    While[Select[order[newlane[[k]]], (newpos[[k]] - 32) < # < (newpos[[k]]) &] ≠ {},
     newlane = ReplacePart[newlane, Random[Integer, {1, 2}], k];
     newpos = ReplacePart[newpos, 120*Random[], k]];
    AppendTo[order[newlane[[k]]], newpos[[k]]];
    AppendTo[initialorder[newlane[[k]]], newpos[[k]]];
    AppendTo[timein[newlane[[k]]], t];
    AppendTo[timeout[newlane[[k]]], 0];
    AppendTo[velocity[newlane[[k]]], a = Random[];
          Which[a < .022, 44 + (2000/3)*a,
          a < .035, (176/3) + (22/3)*a,
          a < .061, 66 + (22/3)*a,
          a < .154, (220/3) + (22/3)*a,
          a < .345, (242/3) + (22/3)*a,
          a < .617, 88 + (22/3)*a,
          a < .842, (286/3) + (22/3)*a,
          a < .957, (308/3) + (22/3)*a,
          a < .988, 110 + (22/3)*a,
          a < .996, (352/3) + (22/3)*a,
          a < 1, (374/3) + 22*a]];
//Sorting the new cars into respective lanes
    x = Transpose[Reverse[Sort[Transpose[{order[newlane[[k]]],velocity[newlane[[k]]],
              initialorder[newlane[[k]]], timein[newlane[[k]]], timeout[newlane[[k]]]}]]]];
    order[newlane[[k]]] = x[[1]];
    velocity[newlane[[k]]] = x[[2]];
    initialorder[newlane[[k]]] = x[[3]];
    timein[newlane[[k]]] = x[[4]];
    timeout[newlane[[k]]] = x[[5]];
    numbers[newlane[[k]]] = (numbers[newlane[[k]]] + 1)]]]
```

APPENDIX C

Data for Traffic Speed

Recorded Speeds	Freeways posted at 55 mph Urban Interstate	Freeways posted at 65 mph Rural Interstate
40 mph and below	7284	798
41 to 45 mph	4360	286
46 to 50 mph	8652	1054
51 to 55 mph	30663	6813
56 to 60 mph	63045	22506
61 to 65 mph	89946	50199
66 to 70 mph	74450	79003
71 to 75 mph	37944	59945
76 to 80 mph	10105	21729
81 to 85 mph	2749	5122
86 mph and above	1377	1398
Total	**330,575**	**248,853**

REFERENCES

[1] D.L. Cochran, *Speed Monitoring on Indiana Highways- April thru Jun, 2002.* Indiana Department of Transportation and Purdue University, West LaFayette, IN, 2002. http://bridge.ecn.purdue.edu/~speed/Reports/2002sum2.pdf .

[2] S. Ghahramani, *Fundamentals of Probability, Second Edition*. Prentice Hall, Upper Saddle River, NJ, 2000.

[3] R.V. Hogg and E.A. Tanis, *Probability and Statistical Inference, Fifth Edition*. Prentice Hall, Upper Saddle River, NJ, 1997.

[4] R. Kunzig, "Curing Congestion." *Discover*, 20(3), pp31-32, 1999.

[5] H. Pulapaka, Personal Communication. September 25, 2002.

[6] J. Rasp, Personal Communication. April 1, 2003.

[7] J. Rui, W. Qingsong, and Z. Zuojin, "A New Dynamics Model for Traffic Flow." *Chinese Science Bulletin,* 46(4), pp.345-350, 2001.

[8] M. Treiber and D. Helbing, *Explanation of observed features of self-organization in traffic flow* 2002.. http://xxx.uni-augsburg.de/PS_cache/cond-mat/pdf/9901/9901239.pdf . February 2, 2003.

[9] M. Treiber, *Traffic Model used for the Simulation.* 1999. http://vwisb7.vkw.tu-dresden.de/~treiber/MicroApplet1_0/IDM.html . February 3, 2003.

BIOGRAPHICAL SKETCH

Jessica Lees is a senior who is looking forward to graduation. Her activities at Stetson include involvement in Student Ambassadors, Alpha Kappa Psi, and the study abroad program. Jessica plans to move to Nottingham, England after graduation, but she is undecided about a career. She enjoys traveling, trying new foods, and watching movies.