MODELLING AND CONTROL OF MULTI-VEHICLE TRAFFIC NETWORKS USING AN INTEGRATED VISSIM-MATLAB PLATFORM

by

Shubhankar Chintamani Shindgikar

A thesis submitted to the faculty of
The University of North Carolina at Charlotte
in partial fulfillment of the requirements
for the degree of Master of Science in
Mechanical Engineering

Charlotte

2020

Approved by:

_____

Dr. Amir H. Ghasemi

_____

Dr. Srinivas Pulugurtha

_____

Dr. Jun Xu

ABSTRACT

SHUBHANKAR CHINTAMANI SHINDGIKAR. Modelling and Control of Multi-Vehicle Traffic Networks using an Integrated VISSIM-MATLAB Platform. (Under the direction of DR. AMIR H. GHASEMI)

This thesis aims to optimize the road traffic network by using different control algorithms and techniques to monitor road driving conditions and improve road safety. Intelligent transportation systems (ITS) have paved the way for opportunities in traffic network monitoring, adaptive signalized intersections, optimized traffic networks and predictive traffic analysis. This thesis aims to develop a platform for integrating PTV VISSIM and MATLAB SIMULINK to design and analyze the flow of traffic in a highway traffic network. As a case study, a 2-lane road network in Charlotte, North Carolina was designed and simulated in PTV VISSIM which is a microscopic multi modal traffic flow software. We model a non-signalized traffic network in VISSIM. The math model and network dynamics were formulated and implemented in MATLAB due to its mathematical prowess. From VISSIM, we take the inflow and outflow rates data and send them to the controller in MATLAB through a VISSIM Component Object Model (COM) interface. A COM-server was created in MATLAB to merge and integrate the software. By employing an extremum seeking approach, an optimal velocity is determined and sent back to VISSIM through the COM interface. Extremum seeking controller is a non model based controller and hence is used in the research due to limited knowledge of the system. The extremum seeking controller was designed in MATLAB SIMULINK. Integration of VISSIM-MATLAB-SIMULINK is done by the COM server. Numerical simulation demonstrates the effectiveness of the platform for testing different traffic control approaches and optimizing flow.

Keywords: MATLAB-VISSIM integrated, traffic networks,Component Object Model, Extremum Seeking Control

DEDICATION

I dedicate this thesis with love an affection to my family and friends. A special feeling of gratitude to my parents, Chintamani and Smita Shindgikar whose words of encouragement and continuous support led my to the fruitful completion of this thesis. My brother Abhijeet, sister Ishwari, uncle Shriram and aunt Shriya who never left my side throughout the process.

I also dedicate this thesis to my friends whose push for tenacity still ring in my ears. I will always appreciate what they have done, especially Vahid, Arjun and Pouria for helping me develop my skills, Sharoni for the many hours of proof reading and Shemal for keeping my spirits high when the times were tough.

I would like to dedicate this thesis to all the people who have been there for me throughout the entire Master's program. You have been the best.

ACKNOWLEDGEMENTS

I would like to thank and acknowledge Dr. Ghasemi, who has been a constant support and guide throughout the Master's program. Also, my committee members, Dr. Pulugurtha and Dr. Jun Xu, without whose support this would not be possible. And finally, UNC Charlotte, for giving me this opportunity to research utilizing one of the best facilities the university has to offer.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

CAV Connected Automated Vehicle

COM Component Object Model

ES Extremum Seeking

GUI Graphic User Interface

HPF High Pass Filter

ITS Intelligent Transportation Systems

KMPH Kilometers Per Hour

LPF Low Pass Filter

MFD Macroscopic Fundamental Diagrams

MPH Miles Per Hour

NCDOT North Carolina Department of Transportation

CHAPTER 1: INTRODUCTION

Simulation modeling has long been recognized as a very useful tool for designing upgrades to urban freeway systems. The simulation model allows the engineer to forecast the effects of a planned freeway system change before it is introduced and to determine the merits of competing models. Due to the increasing demands for traffic, continuous maintenance and improvement of road traffic control systems are essential. The preceding phase of such technology activities is the correct simulation. Also, our day's Smart Transportation Systems (ITS) concept requires advanced use of computer technology.

Economic growth and rapid development have to lead to exponential growth in the use of automobiles. We face an ever-growing problem of traffic congestion in large-scale urban traffic networks due to this. Many traffic control strategies have been tried and tested to reduce congestion and alleviate traffic jams [1].

Traffic jams cost US $87 billion in 2018 [2]. Different control strategies with different traffic flow models have been developed for the management of traffic networks [3–9]. Among these efforts, connected automated vehicle (CAV) technologies have received increasing attention recently [3, 9]. Recent studies have shown the positive impacts of CAVs technology on fuel consumption, reduced travel time, and improved safety [10–12]. However, for practical purposes, (i) no traffic network in the near future will consist entirely of automated vehicles, and (ii) vehicles (even automated ones, but especially the human-driven ones) will never behave entirely homogeneously.

Autonomous cars are looked upon in recent years to reduce traffic congestion due to the reduced gap and high speed-movement. Hence, in recent years, the concept of autonomous vehicles is growing fast. The two main differences between human-driven

vehicles and autonomous vehicles in traffic networks are (i)congestion and (ii)capacity. Much research has been done toward autonomous vehicles. Development in technology has made communication much more accessible. Implementing this communication technology into cars paves the way for information sharing. Information sharing between connected autonomous cars lead to more accurate predictions, making better decisions, and taking appropriate actions in a short amount of time. All these benefits lead to fewer safety distances and higher speeds. We can see the smooth flow of traffic with higher capacity highways.

Traffic network operations in the future may be implicated by allowing wireless communication between vehicles and the transportation infrastructure. Parameters such as vehicle speed, position, traffic congestion, obstacles, stoppage time, vehicle flow, etc. can be known by information sharing, which were earlier unknown or estimated [13]. Connected, autonomous vehicles have the potential to impact traffic networks [14, 15], positively.

The topic of modeling and controlling the traffic network has been studied both on the macroscopic and microscopic levels. Specifically, at the microscopic level, two groups of approaches have recently been presented. The first group, models traffic networks wherein human-driven vehicles are modeled and their behaviors are either predetermined [16–18] or bounded [19, 20]. Numerous distributed control approaches have been proposed to address the traffic management for these microscopic models [21]. On the other hand, the macroscopic models are typically formulated using a conservation law and an equilibrium relation between density and flow, better known as the fundamental diagram. While in a traffic network the flow-density relation is no longer expected to be a real-valued number, recently the concept of the fundamental macroscopic diagram (MFD) has been adopted to obtain an efficient and elegant way for control of large-scale urban traffic networks from an aggregated point of view [22–24]. For instance, in [22], an optimal perimeter control for a two-region

urban city based on the MFD is proposed to regulate the exchanged traffic flows. A mixed control strategy integrating perimeter control for urban roads and ramp metering for freeways has also been presented in [23]. Additional urban traffic control strategies based on the MFD have also been proposed in [1, 24–30]

Various traffic control algorithms and management strategies are found to be an effective solution to the ever-increasing demand for road vehicles and manage rapid growth in travel demand. However, to validate these models with real-life scenarios is still a challenge researchers are working on. A direct and effective way to analyze these control strategies is field experiments. However, field experiments are time-consuming, expensive, and require a lot of effort and human resources. [31] So, to validate these dynamic models, a real-life simulator platform consisting of a MATLAB-VISSIM interface is developed in this research. This platform consists of an interface between the real world simulator VISSIM and MATLAB SIMULINK to implement control strategies. A COM server interface provided by VISSIM is used to build this platform.

Vissim has powerful micro-simulation functions. However, the software is relatively closed. [32] Vissim is a microscopic road traffic simulator that can be used for the study and behavior testing of vehicles. Different vehicle classes like cars, trucks, buses, bikes, trams, and also pedestrians can be simulated and tested in this software. Thus a traffic network simulation can be analyzed in detail due to the accurate description of the traffic network [33]. Vissim's GUI sometimes does not offer dynamic manipulations, so the user has to turn to something where he can access and manipulate the objects during the simulation dynamically. This dynamic assignment can be done through Vissim's COM interface. The COM interface can communicate between different processes in between the software. Using this COM programming interface, the parameters in Vissim, which we had initially been defined through the GUI, can be manipulated by using programming. This programming can be done through several

languages that are suited to handle COM objects like MATLAB, C++, Java, VB, Python, etc.

This thesis aims to develop a platform that allows integration of VISSIM and MATLAB software as the first step towards testing and validating intelligent traffic management systems. To this end, we modeled a non-signalized traffic system in VISSIM. Additionally, we developed an extremum seeking controller in Simulink MATLAB. By connecting MATLAB and VISSIM and transferring the traffic states as well as the control action through a COM interface, we were allowed to maximize the throughput of the modeled traffic system.

This research talks about the macroscopic modeling tools that have been used in this study for simulating managed lane-freeway networks. A simplified model of the fundamental diagram is presented in Chapter 2. Then, this thesis discusses the control techniques, optimization problem and the cost function that were used for increasing the mobility of a homogeneous traffic network in Chapter 3. Chapter 4 illustrates the traffic network geometry modelling and their steps in detail while the COM interface and the working of the integrated VISSIM-MATLAB-SIMULINK platform is discussed in chapter 5. Simulation results are presented in Chapter 6 and Chapter 7 concludes the paper by summarizing the results and listing future ideas and work.

CHAPTER 2: DYNAMICS OF TRAFFIC SYSTEM

Modeling of the traffic network starts with the math model formulation and drafting of the equations of motion for the system. This research deals with homogeneous traffic networks. The homogeneous traffic networks consist of human driven vehicles only. The system equations and the behaviour of the system is further discussed in this chapter.

Consider a homogeneous traffic network shown in. Figure 2.1 where it consists of multiple sub-networks or nodes. We characterize each sub-network $i \in \{1, 2, \cdots, \ell\}$ by its density K_i and the flow Q_i.

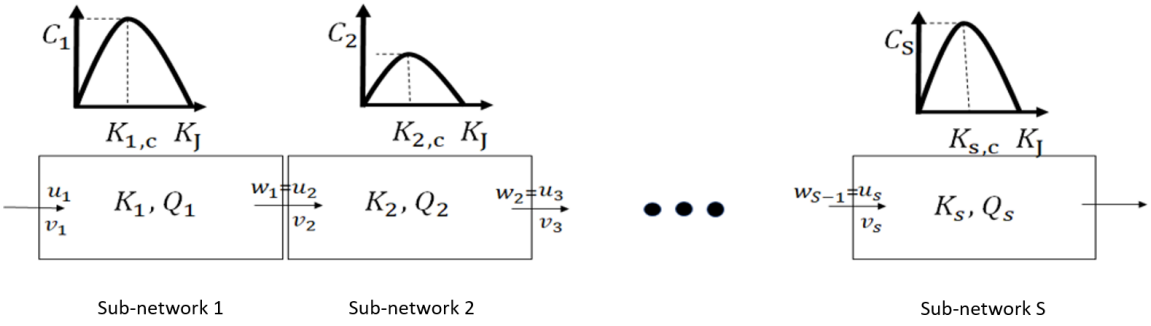

Figure 2.1: A S-sub-network linear transportation network with no intermediate on/off-ramps, where K_i, Q_i, u_i, v_i respectively presents the traffic density, flow, inflow of vehicles and the velocity of vehicles . For each sub-network a fundamental diagram can be derived which relates the flow within the sub-network to sub-network's density K

Density of sub-network i is updated according to the conservation law [34]. Specifically,

$$\dot{K}_i = u_i - w_i \tag{2.1}$$

where u_i is the flow (number of vehicles) entering the lane from a different node, w_i

is the flow leaving to another sub-network.

Following the lane transmission model [35], the inter-flows u_i and w_i are dependent on two intermediate quantities: the node demand and the sub-network supply. The sub-network demand is the number of vehicles that wish to exit the sub-network i at time t, and the supply is the number of vehicles that sub-network i can accept at time T. Both demand and supplies are functions of the density K_i and can be computed using a fundamental macroscopic diagram. Additionally, the average flow of a sub-network is a function of its density and can also be computed using the fundamental diagram.

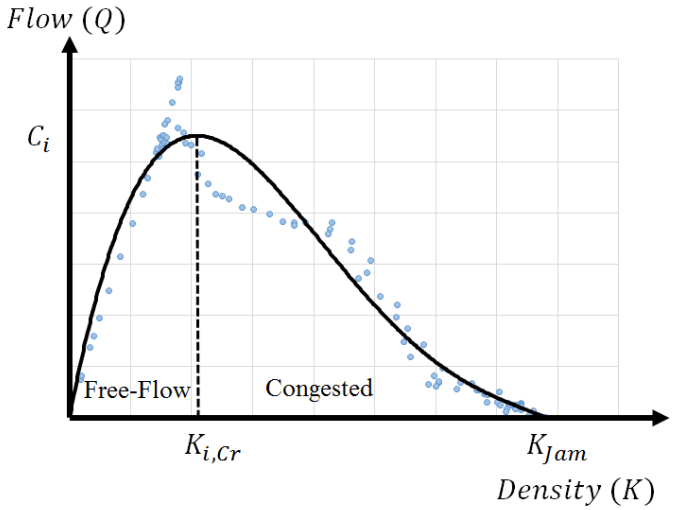

Figure 2.2: Flow density diagram of the system

Figure 2.2 represents a schematic of a fundamental diagram for a homogeneous sub-network i. We run a couple of simulations for a scenario in VISSIM to get the flows for each density. By fitting a fourth-order polynomial curve on the data points which are driven from VISSIM simulations, we can get the flow-density diagram. The critical density $K_{i,Cr}$ is the maximum number of vehicles on the road such that the average velocity of the vehicles is equal to the free flow velocity. The capacity C_i is the correspondent flow at the critical density. The MFD curve divides into two sections. The left sections represent the uncongested state, wherein the average velocity of the

road is equal to the free flow velocity (i.e., the slope of the curve represents the free-flow velocity). The right section corresponds to the congested and K_{J} represents the jam density where the average velocity of the traffic flow is zero.

In this research, to find out the relationship between the average flow of the lane and its density, the fourth-order polynomial equation shows the continuous relation between the mean flow rate of Q and the density in the cell K. Specifically,

$$Q_i(K_i) = -4K_i^4 + 0.02K_i^3 - 2.5K_i^2 + 13K_i + 102 \tag{2.2}$$

CHAPTER 3: CONTROL OF TRAFFIC SYSTEM

We now pose an optimization problem to determine the optimal velocity which vehicles should have at the beginning of each segment such that the average flow of all sub-networks is maximized. For a traffic network shown in Figure 2.1, we define $\mathcal{X} = [K_1, \cdots, K_{\mathrm{S}}]$, and $U = [U_1, \cdots, U_{\mathrm{S}}]^{\mathrm{T}}$.

To maximize the mobility of the system, we consider the average of the mean-flow rate as the cost function. To this end, we defined an objective function that focuses on maximizing the throughput by keeping the number of vehicles in each lane as close as possible to their critical densities. To meet this goal, cost function J is defined as

$$\min_{U} J = \sum_{i=1}^{S} Q_i(k_i) - C_i \tag{3.1}$$

where $\mathcal{X}_i$, is the state of the sub-networks and U_i is the control inputs from the controller and Q_i is the mean-flow rate of lane i and C_i is the capacity of the subsection.

3.1 Extremum Seeking Controller

In this thesis, we employed the extremum seeking approach for solving the cost function (3.1). The reason behind selecting this specific approach is that ES control is a non-model-based real-time optimization approach that is suitable for systems with unknown or limited knowledge of the optimal operating point. A conventional ES scheme is illustrated in the context of the optimization of the throughput of a traffic network. To determine the optimal control input of $\mathcal{U}_i^*$, the signal U_i is formed by adding a sinusoidal perturbation $A_i \sin(\Omega_i t)$ to $\bar{\mathcal{U}}_i$, where $\bar{\mathcal{U}}_i$ is the estimated value of the optimal operating flow rate.

The calculated Q_i then passes through a high-pass filter $G_{i,\mathrm{HF}} = \frac{s}{s+\omega_{i,\mathrm{H}}}$, where $\omega_{i,\mathrm{H}}$ is

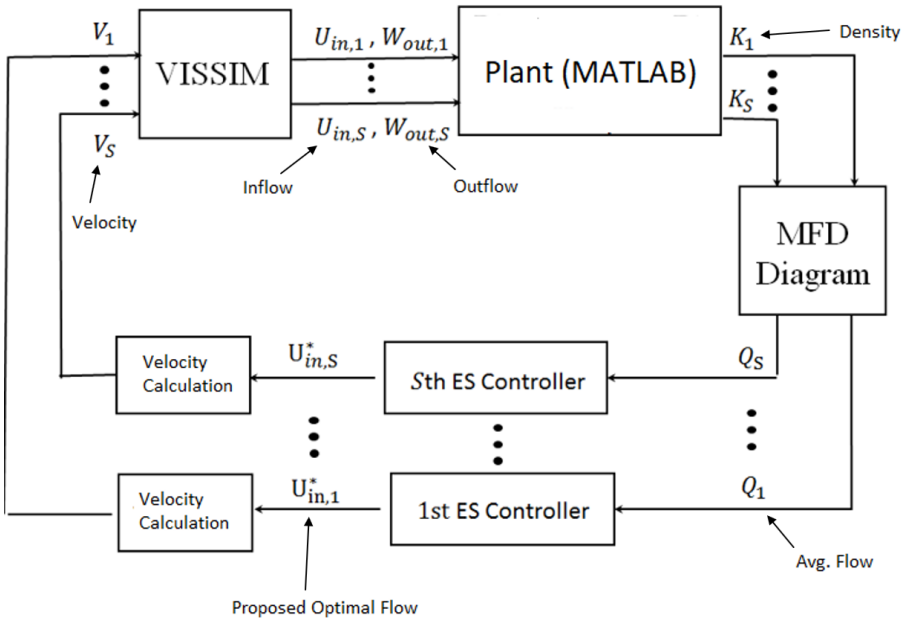

Figure 3.1: Schematic of traffic optimization control.

the frequency for the high-pass filter. The output of the high-pass filter passes the low-pass filter $G_{i,\mathrm{LF}} = \frac{\omega_{i,\mathrm{L}}}{s+\omega_{i,\mathrm{L}}}$, where $\omega_{i,\mathrm{L}}$ is the frequency for low pass filter. The output of the high pass filter is multiplied by the same perturbation signal, $A_i \sin(\omega t - \phi_1)$, where ϕ_i is a phase lag. The resulting signal passes through a low-pass filter, and then an integrator and a new estimate of $\bar{\mathcal{U}}_\rangle$ is produced in the direction of increasing Q_i (Figure 3.2).

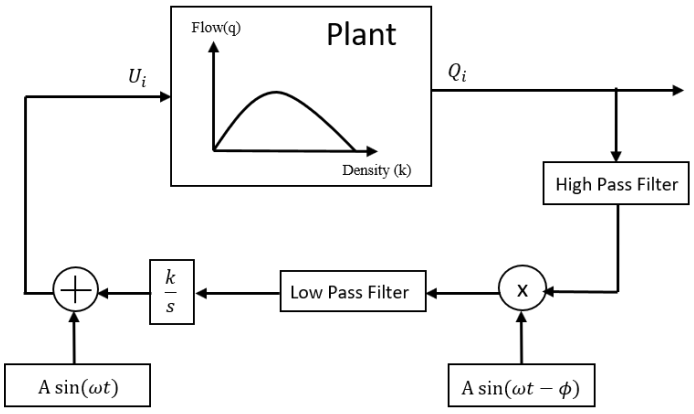

Figure 3.2: Extremum Seeking Controller

Figure 3.3 [36] explains the working of an extremum seeking controller. From the

graph we can see three operating conditions of the extremum seeking controller viz. when the operating point is larger, equal and smaller than the extremum seeking point.

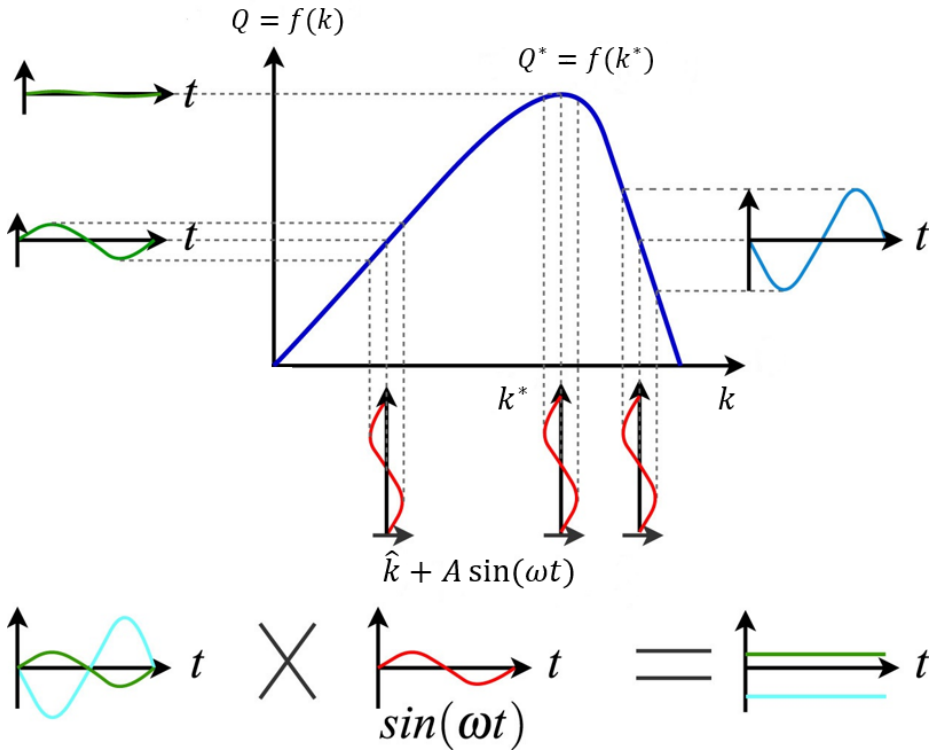

Figure 3.3: Working of Extremum Seeking Controller

The extremum seeking controller moves up and down the graph multiplying the two phase signals resulting in a signal with either a positive mean if the signals are in phase with each other or with a negative mean if the two signals are out of phase. This working of the extremum seeking controller enables us to reach an optimal point which is the critical density point where the derivative is zero.

CHAPTER 4: MODEL OF THE TRAFFIC SYSTEM

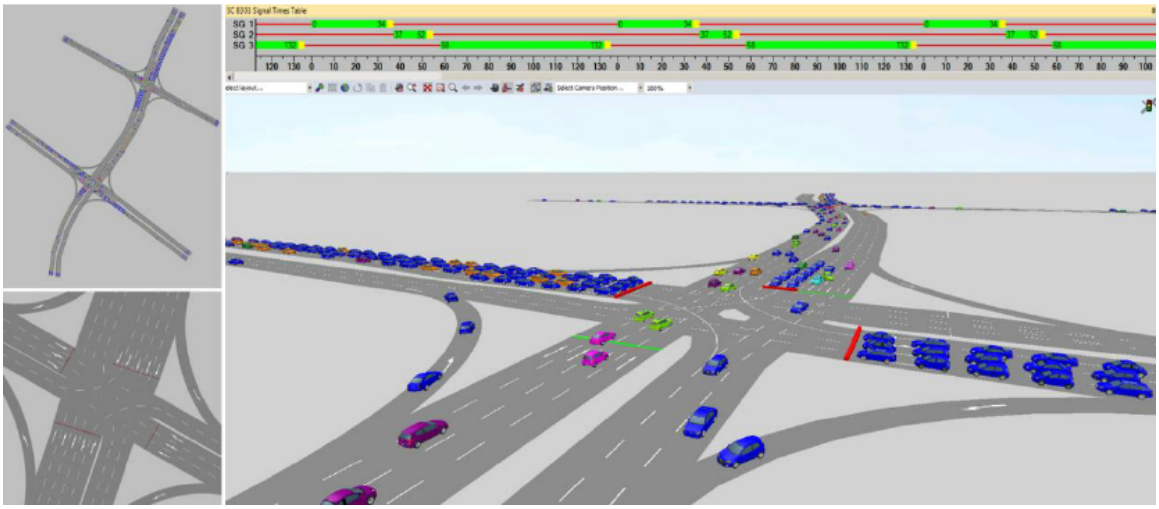

Figure 4.1: Sample Vissim Model

The model of the traffic system was designed in a microscopic traffic simulation software, PTV VISSIM (11.0). The model was set up in an urban setting in the city of Charlotte, NC, USA. Figure 4.1 [37] below shows a sample road network designed using the Vissim GUI. As a case study, a 4 node single non-signalized road link with 2 lanes traffic model was designed. Real, exact geometry and traffic parameters were used to recreate this road traffic network. This exact modeling can be possible in this software using background images or using the inbuilt map service. The width of each lane was set to be 12ft, according to the NCDOT. The road network designed was the North Tryon Street from JW Clay Boulevard to the intersection at East Mallard Creek Church Road, as seen in figure 4.3. VISSIM offers two map providers, viz. Bing Maps for an aerial view and Open Street view map provider Mapnik for generating and designing the traffic network accurately and realistically. For this case study, Mapnik was used to design the traffic network. The link was set to behave

as an urban motorized road with free lane selection. A Wiedemann 74 car following model was used for the car following behavior. The vehicle class to be studied were only cars. Hence all other vehicle classes were blocked for the traffic network. This model was used for all the tests and different control algorithms implemented to check feasibility, control, road network management, traffic flow, and road safety.

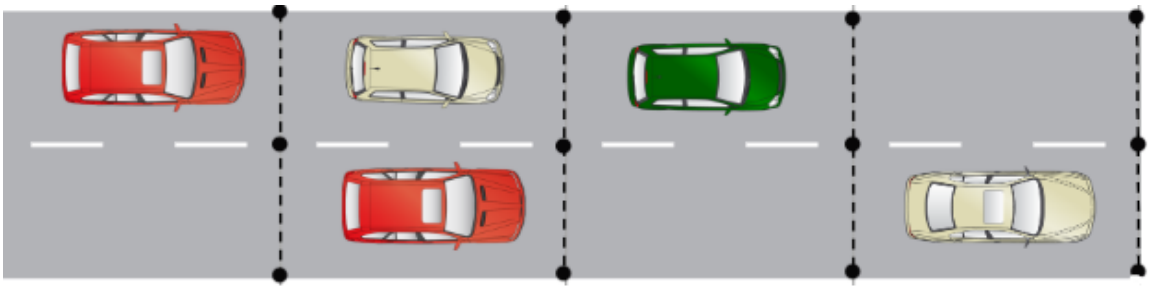

Figure 4.2: Homogeneous Traffic Network

The modeling of the traffic network has to be done in the Vissim GUI, after which it can be called into your desired software for use in COM programming. The step by step details of the Vissim GUI and the various functions and attributes used to design the road network geometry for this case study is explained ahead in this chapter. This research deals with a multi-vehicle homogeneous traffic network. Human-driven

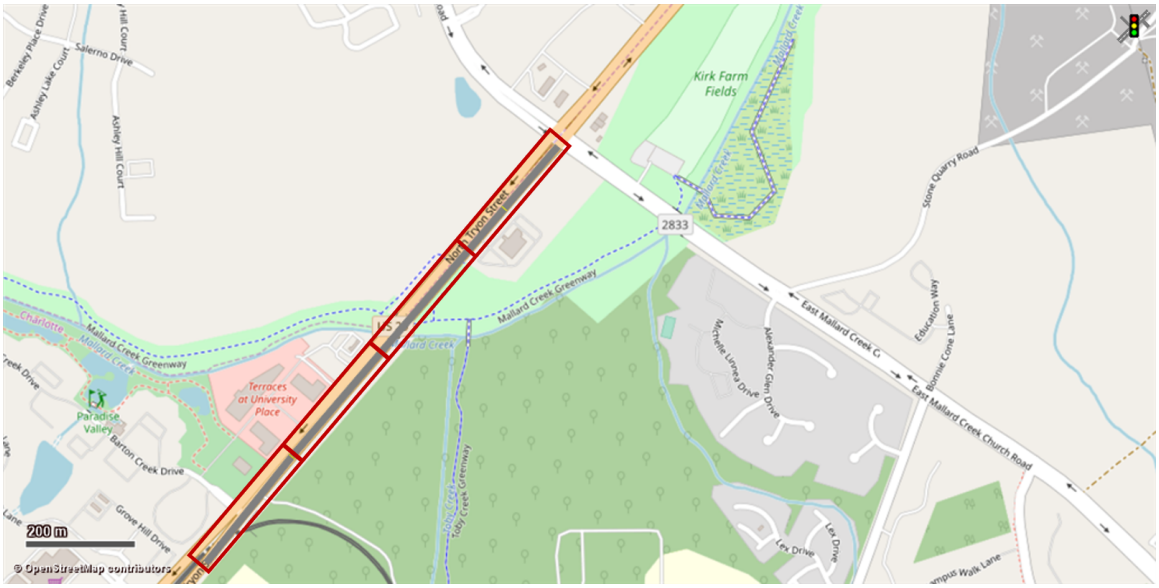

Figure 4.3: Traffic Network

vehicles are considered in this research. A similar road network was designed for analysis and optimization in the Vissim GUI. All the steps followed while designing, and the Vissim interface and attributes are explained hence.

4.1 Graphical User Interface

Vissim offers an extensive graphical user interface (GUI) as seen in figure 4.4 [38] which consists of a Network Editor, Network Objects sidebar, a quick view sidebar, and a list menu to show and edit each object attribute data like vehicle inputs, desired speeds, vehicle routes, reduced speed areas, etc. The network editor is where all the Vissim attributes like vehicle inputs; signal heads, areas, etc. can be found and used to design the traffic network. The network editor is where all the objects are placed and edited to design the network. The background of the network editor can be plain grey with no maps or inbuilt mapping options like Bing maps can be used to design the traffic network. For this case study, the traffic network was designed using the inbuilt maps provided by Vissim.

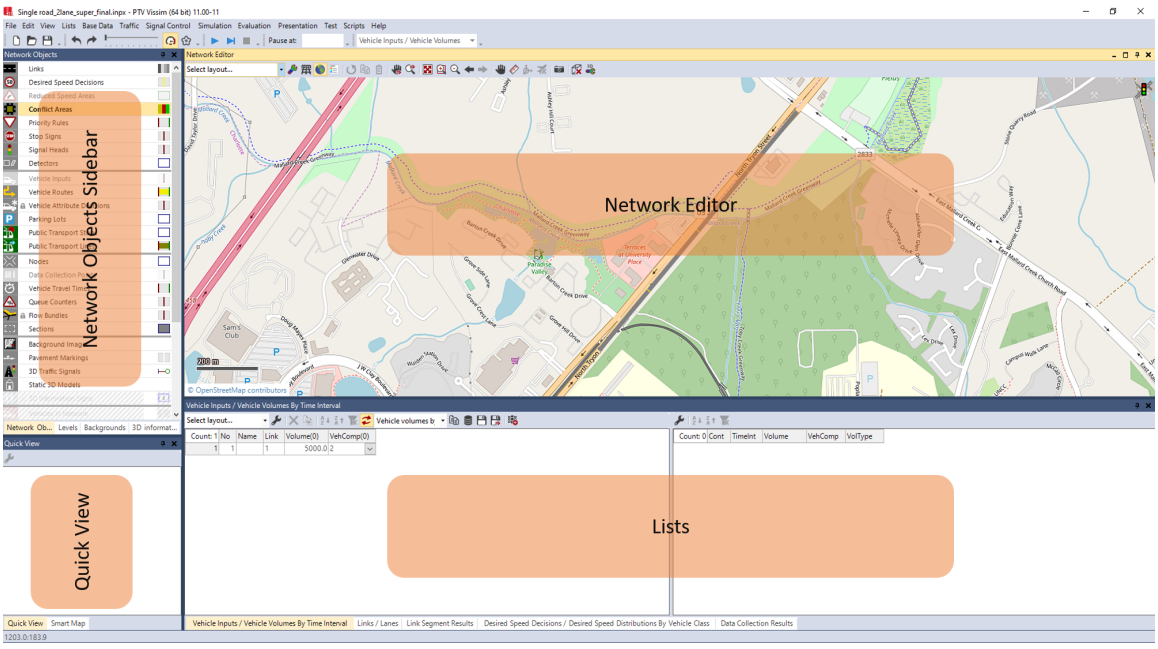

Figure 4.4: Vissim GUI

4.2 Designing of Traffic Network

4.2.1 Links

Links will be forming up the roads for our traffic network. A link can be a single lane link or have multiple lanes. It can be a motorized road or a gravel road or a train/tram line. The types of vehicles allowed on the road and if it is a turning lane or not can also be set. So, a lot of options can be selected and set while designing the road links.

- To start the designing of the link, select the links button (shown by a road) from the objects sidebar

Figure 4.5: Link in Network Objects

- Once that is selected, go to the desired starting point. To add a new link: Press Ctrl + to right-click and drag the mouse from the starting point to the ending point of the while holding down the right-click button
- The graphic screen shows a grey road link in the driving direction of the vehicles
- A dialog box opens up where you can edit three base parameters viz. Lanes, Pedestrian Areas, Display and Others
- In lanes, the editable attributes are the number of lanes, a width of each lane, the name of the link, its behavior, allowed/blocked vehicles on the link, and the lane change policies.
- You can define the lane as a pedestrian area too b selecting that option.
- A 3D rendered lane can be designed by defining the start and end Z-offset values and the thickness of the lane

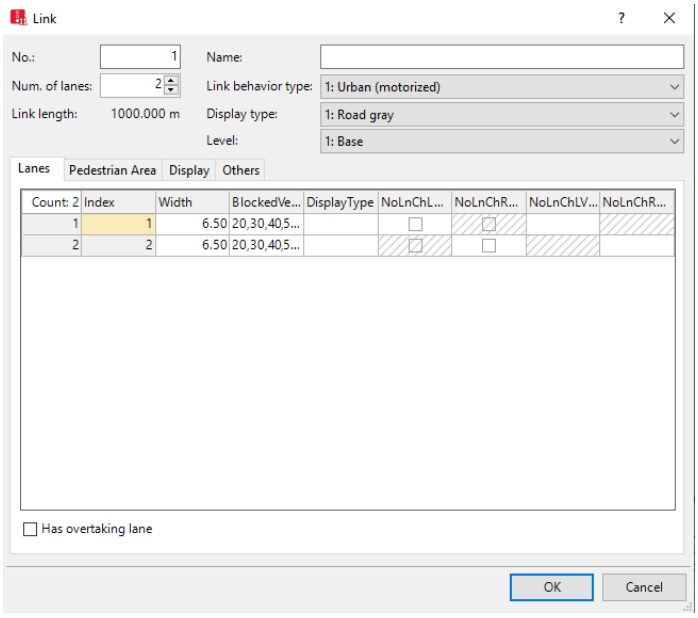

Figure 4.6: Link Dialog Box

- Link segment evaluation parameters and attributes, dynamic assignment, look ahead distances, and other basic and advanced assumptions too can be set up in the dialog box
- To generate the opposite direction, right-click inside the link and select generate opposite direction and select the number of lanes
- To adjust the link according to the road geometry, VISSIM offers spline points which can be generated by pressing Ctrl + right-clicking inside the link
- The spline point that we generated in the previous step now acts as a pin so that the link curvature is adjusted only for the section downstream of that spline point.
- Use spline points to adjust the curvature of links so that the traffic model is accurate and has the same geometry

4.2.2 Connectors

In VISSIM, vehicles cannot travel from one link to another just if there is a geometrical connection between the links. For this purpose, connectors are necessary. Connectors join two road links and makes travelling between links possible. Link-connector-link is the typical travel path. The steps to design and place a connector are listed below.

- Select the links insert mode

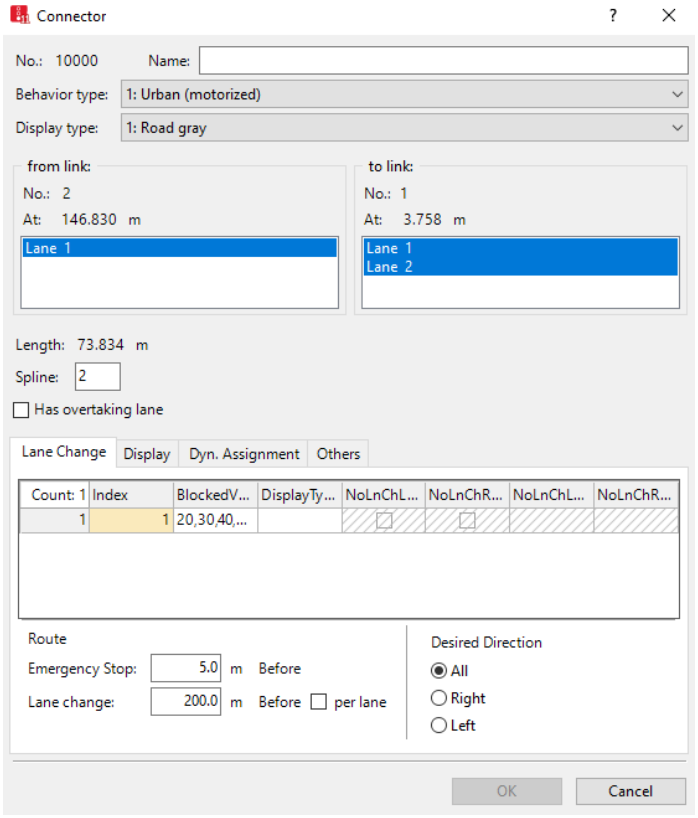

Figure 4.7: Connector Dialog Box

- Pan to the end of the link and the start of the link you want to connect it to
- CTRL + right-click inside link and keep the mouse button pressed and drag the mouse into link you wish to connect
- The connector dialog box opens on releasing the mouse button

- For turning connectors, select the number of splines points
- Select the lanes you want to connect and hit okay

4.2.3 Vehicle Inputs

The next step used to design the road geometry and attributes in the research are configuring the vehicle inputs. These inputs define which road link should have motorized traffic travelling through them and the vehicle volumes for that link. Each link can have a different vehicle volume and composition. Different vehicle volumes for the same link can be defined based on time interval. Vehicles traveling on public transport lines are modeled separately and must not be included here. In this research we shall only be simulating human driven cars and the volume is 3000 vehicles/hour. The steps used to model the vehicle inputs is explained below.

- Select the Vehicle Inputs attributes from the network objects sidebar as shown in figure 4.8

Figure 4.8: Vehicle Inputs

- To place a vehicle input, Ctrl + right click on the link which adds a vehicle input at the start of the link shown in black
- A list opens which allows you to select the vehicle flow volume (3000 veh/hr)

Different volume types can be set viz. Exact and Stochastic. In this research, random volume fluctuations occur since the stochastic volume type was selected. This means that vehicle inputs will have a random probability distribution pattern.

4.2.4 Vehicle Routes

In this research, a single non signalized road section was designed as part of the case study. However, for more complex traffic networks, an array of road sections and connectors consisting of different turns and routes may be possible. For this reason, it is necessary when modelling a traffic network to assign vehicle routes. If vehicle routes are not assigned, the vehicles either follow a straight path or disappear at the end of a link even with turning connectors present and connecting links at different turns. Thus, at road branches, the vehicle path is defined by routes. Each vehicle without a routing decision is assigned a path in VISSIM when it arrives at a routing decision. Route proportions for all the routes are taken into consideration when assigning paths to vehicles. Vehicles are neither generated, nor are they taken out of the network at this point.

- To insert vehicle routes, from the network objects sidebar select the Vehicle Routes (Static) mode as shown in figure 4.9

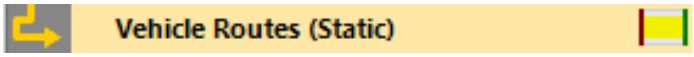

Figure 4.9: Vehicle Routes

- To add a new routing decision, Ctrl + right click on the starting point of the link which creates a purple bar which signifies the starting point of the vehicle routing decision
- Start moving the mouse along the routes you want to the final location and click the left mouse button
- The route destination is defined and shown as a yellow band
- Similarly, add other route destination markers for the same link for the vehicles

- Double-clicking outside of the Vissim network exits the vehicle routing decision for the particular link

4.2.5 Reduced Speed Areas

Since this research deals with a non-signalized traffic network with a single two lane link, analysing congestion in the networks required an initialization criteria that caused congestion in the network. The two approaches thought of to initialize congestion were (i) Reduced Speed Areas and (ii) Desired speed decisions. Reduced speed areas enable modelling speed reductions for selected vehicle classes for a selected duration of time. In this research, a temporary speed reduction to simulate a block or stop sign was used as an alternate strategy. At the end of sub-network 3, a reduced speed area was set up for each lane for 450 seconds where the speed was set to be 3 mph or 5 kmph. This created a jam scenario so that we could analyse and test the working of the extremum seeking controller and the platform. The steps used to design and set up reduced speed decisions are explained below.

- To set up reduced speed areas, first select the Reduced Speed Area mode from the network objects sidebar as shown in figure 4.10

Figure 4.10: Reduced Speed Areas

- To add a reduced speed area, Ctrl + Right click on the desired start point of the area and drag the mouse for the desired length which pressing down the right click mouse button

- The dimensions of the reduced speed area are indicated by a yellow polygon

- Once the dialog box opens, for each relevant vehicle class assign speed distribution and deceleration

4.2.6 Right of Way/Conflict Areas

To model non-signalized road networks, conflict areas are needed to set priority right of way decisions for vehicles. Conflict areas are widely used for turning vehicle right of way decisions. They are preferred way to model non-signlazied intersections. The method to model this is explained below.

- All potential conflict areas are displayed as yellow (passive) once the conflict areas button is selected from the network objects sidebar as shown in figure 4.11

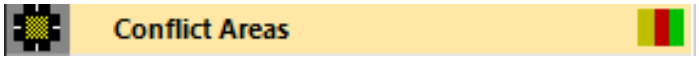

Figure 4.11: Conflict Areas

- Right-click on the conflict area opens the context menu where select the desired right of way such that the area is marked as green and the other one as red
- For branching, to avoiding vehicle overlap when queuing select set status to undermined

4.2.7 Desired Speed Decisions

Desired speed decisions are basically speed limits. They are used to define vehicle speeds at certain user defined points for desired time intervals. Desired speed decisions act like variable speed limits based on the control instructions from the extremum seeking controller and hence are used extensively in this research. The desired speed decisions are set up at every 250m for the road link in the geometry for each lane. There are 4 desired speeds that will be changed dynamically in between simulations. There are a total of 6 desired speed decisions. Initially to create congestion and attain jam density the speed decisions at the outflow of node 3 are set to be 3 mph/5kmph for the first 450 seconds. The controller kicks in at each 225 seconds and changes these

velocities dynamically depending on the traffic scenario condition at that moment. The steps to set up the speed limits is explained below.

- From the network objects sidebar, select the desired speed decisions button as shown in figure 4.12

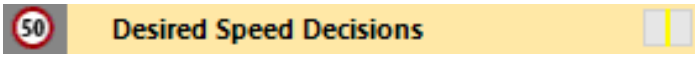

Figure 4.12: Desired Speed Decisions

- Ctrl + right click on the road link and the specific lane you wish to set the desired speed limit for
- A yellow bar signifies that a desired speed decision has been set
- Now, set the position where you want the desired speed decision which is at 250m for this research
- Add the vehicle classes and their respective speed limits by clicking add
- Similarly repeat these steps and set up desired speed decisions for every lane of each sub-network of the entire road link

4.2.8 Data Collection Points

Datapoint collection measurements are assigned points in the VISSIM road network where the user wishes to collect data. This research utilizes this functionality extensively to collect and analyse data and give appropriate control instructions. There are various attributes that can be collected like the queue delays, number of vehicles in the network, occupancy rate, speeds (arithmetic and harmonic averages) and acceleration. In this thesis, the queue delays and the number of vehicles are collected every 225 seconds. This data is analysed and control instructions are given to improve flow and reduce congestion. The steps used to setup the data collection measurements are explained below.

- Select the Data Collection Measurement button from the network objects sidebar as shown in figure 4.13

Figure 4.13: Data Collection Points

- Ctrl + right click on the link and lane where you want to collect data
- A brown bar denotes that a data collection point has been set up at that position and a list opens showing the created data collection point
- Like the desired speed decisions, select the position where you would like the data collection point to be at which for this research is at every 250m
- Create data collection points at the entry and exit of each node
- By default the data collection point collects all possible data and to refine what data should be collected, select the attribute selection button from the list and unselect all the data not needed to be collected for the simulation
- Once the data collection points are set up, define them in the data collection measurements and then configure them to be collected at every 225 seconds
- Defining data collection points can be done through evaluation, measurement definition, data collection measurement and configuration can be done by evaluation, configuration
- Until the above two steps are done, the data collection points will not collect data
- Once the data collection points are set up, results of the data collection points can be further filtered to just show the required data

In this thesis we are collecting only number of vehicles and the queue delays results from the data collection points measurements. The number of vehicles entering and exiting the node are used to calculate the flow of that node while the queue delays measurements are used to define a condition for the controller to run. The controller only exports new speed limits if the queue delay is over 5 seconds. If the queue delay is less than 5 seconds the system keeps running signifying that there is no congestion in the network.

4.2.9 Displaying and Analysing Results

VISSIM offers a data analysis package which is used for post processing and analysis of results. In this thesis, we are using this module of VISSIM to validate our platform and see if the controller is able to reduce congestion in the road network. After each simulation, the data collection results and the link collection results are plotted as a line graph to see how the densities, speeds and the flows fluctuates after every simulation time interval. These results are discussed in brief in Section 6. The steps to display the results are listed and explained below.

- After the simulation is complete, go to the data collection measurements list and select the data to be graphed
- Right click the data and select "plot data for selected attributes"
- A graphical window will open showing bar graphs of the selected attribute
- Change the graph parameters to desired values and representation

4.2.10 Simulation

After performing all the steps above, the network is ready for simulation. VISSIM offers single step and continuous simulation options. This thesis utilizes the continuous simulation with breaks at every 225 seconds when the network is analysed. To just get data and perform analysis when the graphical representation of the states of

the system is not required, VISSIM offers a quick mode simulation which suspends the updating of the graphic network window and the simulation runs quicker. For this thesis we do not use the quick mode as we are interested in the visual state of the system too. The steps to run the simulation are explained below.

- To run the simulation and set up the parameters for the simulation like simulation time, step time and random seed parameters, select simulation, parameters and input the desired attribute values
- Set the simulation break times
- Save the network file
- Click on the RUN CONTINUOUS button in the main toolbar.

The simulation will run for the first 225 seconds and break after which on pressing the RUN CONTINUOUS button again, the simulation will start. In this thesis, these parameters are set through the COM server. The detailed working of the COM server and how the integrated platform works is explained further in Chapter 5

CHAPTER 5: PTV MODELLING AND CONNECTING TO MATLAB

Vissim is microscopic road traffic simulator which can be used for the study and behaviour testing of vehicles. Different vehicle classes like cars, trucks, buses, bikes, trams and also pedestrians can be simulated and tested in this software. Thus a traffic network simulation can be analysed in detail due to the accurate description of the traffic network [33]. The road network geometry was designed and developed in Vissim's user friendly and easy to understand graphical user interface (GUI). However, Vissim's GUI sometimes does not offer dynamic manipulations so the user has to turn to something where he can access and manipulate the objects during the simulation dynamically. This dynamic assignment can be done through Vissim's COM interface. The COM interface can communicate between different processes in between the software. Using this COM programming interface, the parameters in Vissim which we had originally defined through the GUI can be manipulated by using programming. This programming can be done through a number of languages that are suited to handle COM objects like Matlab,C++,Java,VB,Python, etc.

5.1 COM Interface

In this research, we have used MATLAB to program GUI parameters and manipulate them dynamically. Parameters that are aimed to be changed dynamically in this research are the desired speeds, in flows or vehicle inputs at each link and data collection parameters and collection points. For this, a road network was first designed in the Vissim GUI using the integrated maps that it offers. To do this dynamically, a component object (COM) server was set up in MATLAB and all commands were passed through this interface to VISSIM. Using Matlab programming language, a

COM client i.e. an ActiveX server was created.

A strict object hierarchy structure are followed by the Vissim COM programming and it consists of two kinds of object types [33] viz. collections (array,lists) and containers. Objects are linked to one another and addition or removal of objects is only possible in the container due to which this distinction is necessary [38]. In the figure below a hierarchical structure of the VISSIM COM object is shown. The interface of objects are always represented by an 'I'. The head is always the Vissim object which is followed by different interfaces and objects as seen in figure 5.1.

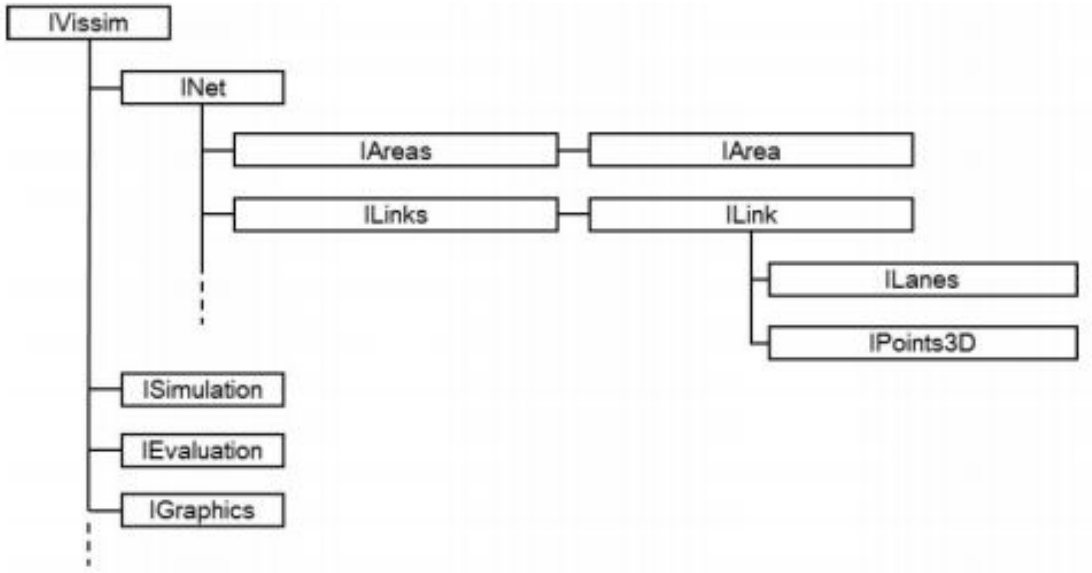

Figure 5.1: Vissim COM Hierarchy

These interfaces can be acessed through the Vissim help within the software which can be found in Help > Vissim-COM > IVissim as shown in figure 5.2.

Also, for more help with the COM object methods, the Matlab command window can be used. For example, to check the object methods created via the Vissim-COM server, type "vis.methods" in the command window. Matlab will provide you with a list of the methods that can be used in the command window.

This closed loop will continue running till we have reached critical velocity which will give us maximum flow or free flow. The step by step procedure followed while

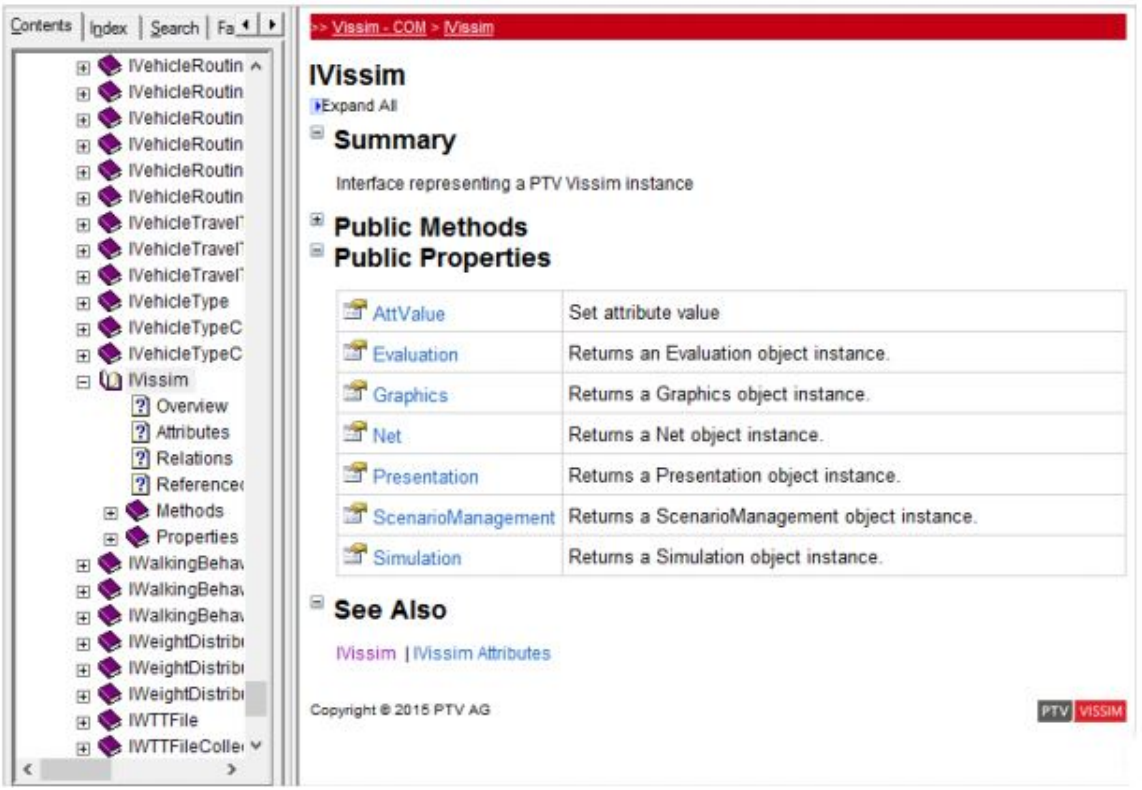

Figure 5.2: Vissim COM Hierarchy

integrating Matlab and Vissim is explained hencefurther.

5.2 Data Transfer and Communication between Packages

There is data and control signal command transfer between three different software in this platform as seen in figure 5.3. Each communication link acts as a port for data and command transfer between VISSIM and MATLAB.

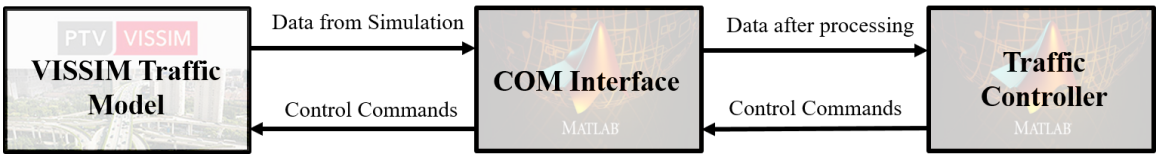

Figure 5.3: Data Transfer between Vissim-COM-Controller

5.2.1 VISSIM-To-COM Link

The simulation is designed and set up and simulation is started using the COM interface. A time step of 1 second is used to simulate the network for 2475 seconds.

The vehicle inflows, outflows and density data is collected at each node of the road network. This data in the form of a list is saved and exported to the COM server through the data collection measurement interface at each time step. The ItemByKey syntax is used to get the flows for each data collection point. The number of vehicles passing each data collection point are recorder by VISSIM and saved in the list. This list can be accessed in MATLAB using:

```
DC = vis.Net.DataCollectionMeasurements.ItemByKey(m);
NoVeh= DC.get( 'AttValue','Vehs(Current,Last,All)');
```

Once the inflow and outflow are in MATLAB, the density is further calculated by using equation 2.1 in MATLAB. These calculated densities are stored in the MATLAB workspace which will be further exported to the controller in SIMULINK.

5.2.2 COM-To-Controller Link

The density data saved in the workspace is processed and conditioned into a time-series that is readable by the traffic controller and exported to it. These attribute values are imported into SIMULINK using inbuilt "From Workspace" blocks which import the densities for each time step. These go through the fundamental diagram and eventually to the controller. The controller gives the optimal values as discussed in section 3. The controller is designed in SIMULINK so the data transfer between MATLAB workspace and SIMULINK goes smoothly without any discrepancies or latency.

5.2.3 Controller-To-COM Link

The controller calculates the optimal flows and velocities according to the traffic scenario as explained above. These new attribute values are then sent to the COM interface i.e. MATLAB workspace using predefined SIMULINK blocks again and those variables are saved in the MATLAB workspace. Here the "To Workspace" block is used to transmit the data generated by the controller.

5.2.4 COM-To-VISSIM Link

The COM server i.e. MATLAB now reconditions the data again in a format that is readable by VISSIM and transfers the data using predefined COM objects using the "set.AttValue" method as shown below using MATLAB script. Here, our attribute is velocity and this is set using the desired speed decisions method in VISSIM. This method is under the DesSpeedDecision interface.

desiredspeed=vnet.DesSpeedDecisions; speed=desiredspeed.ItemByKey(w); speed.set('AttValue'

Since we are just dealing with cars, the results for only cars' vehicle class (10) are set and exported throughout the simulation.

5.3 Vissim-COM Server Creation

We start the programming by the creation of the ActiveX server. To do this we need to start MATLAB and create a new script file('.m' extension). A new network editor will open which is where the programming will be done. Start by clearing the command window, clearing the workspace variables and their values and closing all the open MATLAB windows. This can be done by using the following three commands:

```
clc;
clear all;
close all;
```

We want MATLAB to pass only single dimension arrays to Vissim. To do this use the feature which defines the attribute for dimensions of array and set it to one. This can be done by using the following command:

```
feature=('COM_SafeArraySingleDim',1);
```

Next, create the COM server using the 'actxserver' command in MATLAB. This can be done using the following command:

```
vsim = actxserver('Vissim.Vissim');
```

If there are multiple versions of Vissim installed on your device, you can call out the particular version on Vissim you would like to use. This can be done by adding the version number like shown below for version 11.

For Vissim version 11:

```
vsim = actxserver('VISSIM.vissim.11');
```

Here, 'vsim' is the variable for the ActiveX server which we will be using further throughout the programming when we have to output to Vissim through MATLAB using the COM-server. actxserver is the command for calling the ActiveX feature and Vissim.Vissim is the syntax for calling Vissim. 11 is the version of Vissim you would like to use.

5.4 Vissim Network and Layout

Creation of the Vissim-COM server is done and the next step is to load your network layout into the MATLAB workspace. We cannot create the network objects and geometry through the COM programming. This has to be done using Vissim's graphical user interface (GUI). Create the geometry and save it in your defined workspace which is the same as where your Matlab file is being saved to reduce further complication. If you are unable to save it in the same location, alternate syntax are provided further to access those files. There should be able to identify two files associated with your Vissim network. A project file with the .inpx extension and a layout file with a .layx extension.

Now, incorporate them in the COM program code using predefined commands. To load the network use the command "LoadNet and to load the layout use the 'LoadLayout' method. The syntax to do this is shown below:

```
PathCOM = cd;
```

Here, cd means current directory. You could also use 'pwd' which means it shows the access path of the current directory and use that path displayed in the command window.

```
Netfile= fullfile(PathCOM, 'COMsetup.inpx');
flag_read_additionally = false;
```

If you would like to also access and read the network elements then set the above to true.

```
vsim.LoadNet(Netfile, flag_read_additionally);
Layfile = fullfile(PathCOM, 'COMsetup.layx');
vsim.LoadLayout(Layfile);
```

Alternately, you could also define the path, the file name and use the following syntax:

```
vsim.LoadNet('D:\VissimCOM\COMsetup.inpx');
vsim.LoadNet('D:\VissimCOM\COMsetup.layx');
```

5.5 Simulation Settings

In order for our simulation to run for a specific period of time and with a desired step time, we need to set these parameters in our COM program. We can run the simulation for a single time step or run it continuously. We can also break the break at a certain time and set new parameters accordingly. There are again different ways to do this as shown below:

```
period_time = 3600;
step_time = 10;
random_seed = 42;
```

Variation 1:

```
sim = vsim.Simulation;
sim.set('AttValue','SimPeriod',period_time);
sim.set('AttValue','SimPeriod',step_time);
sim.set('AttValue','RandSeed',random_seed);
```

Variation 2:

```
set(Vissim.Simulation, 'AttValue', 'SimPeriod', period_time);
```

```
set(Vissim.Simulation, 'AttValue', 'SimPeriod', step_time);
set(Vissim.Simulation, 'AttValue', 'RandSeed', random_seed);
```

5.6 Defining and Setting the Network Objects

Network objects and attributes can defined through COM programming and set to desired values. First, we define a network object that will be used to set or get various attributes. This can be done using the command:

```
netob=vsim.Net;
```

Now we define the needed attributes. Suppose, you want to set vehicle inputs for different links. You start by defining the vehicle input number, selecting the new volume and setting it to that link using the following syntax:

```
vin=netob.VehicleInputs;
```

For link 1:

Set the vehicle input number:

```
vin_no=1;
```

Setting the new volume:

```
vol_1=1500;
vin_1 = vin.ItemByKey(vin_no);
vin_1.set('AttValue','Volume(1),vol_1');
```

For link 2:

Set the vehicle input number:

```
vin_no=2;
```

Setting the new volume:

```
vol_2=2500;
vin_2 = vin.ItemByKey(vin_no);
vin_2.set('AttValue','Volume(1),vol_2');
```

If you would like to have different volumes for the same link but after a set time interval, first define time intervals in the Vissim GUI and then do the same process

as said above but with a change in the volume numbers i.e. Volume(1), Volume(2), Volume(3), ... and so on. So, if you have 4 time intervals each of 100 seconds (0-100,101-200,201-300,301-400) and wish to set different volumes for each, it can be done using the syntax shown below.

```
vin_no=1;
```

For 0-100:

Setting the new volume:

```
vol_1=1500;
vin_1 = vin.ItemByKey(vin_no);
vin_1.set('AttValue','Volume(1),vol_1');
```

For 101-200:

Setting the new volume:

```
vol_2=2500;
vin_2 = vin.ItemByKey(vin_no);
vin_2.set('AttValue','Volume(2),vol_2');
```

For 201-300:

Setting the new volume:

```
vol_2=3500;
vin_3 = vin.ItemByKey(vin_no);
vin_3.set('AttValue','Volume(3),vol_3');
```

For 301-400:

Setting the new volume:

```
vol_4=4500;
vin_4 = vin.ItemByKey(vin_no);
vin_4.set('AttValue','Volume(4),vol_4');
```

So, Volume(1) basically means the first defined time interval. However, you can set different volumes for different time intervals only if continuous is deactivated. If it is

activated, the above is not possible and you will get an error saying that volume is not subject to change. Hence, if you wish to have varying volumes for different time intervals for a single link, continuous has to be deactivated.

Similarly, you can change desired speeds or any other attribute you wish to change. The command identifier for each attribute which can be changed can be found in the help section. For example, for vehicle inputs, it can be found in Help>Vissim-COM>IVehicleInput

5.7 Simulation

We now run the simulation using for loops or different syntax's. It is preferable to eliminate for loops as they require more computation time and memory.However, both the syntax's are explained below. We run the simulation for the set desired period time, random seed and step time. It is possible to break the simulation at a certain time to change parameters dynamically in between a simulation using this feature of the COM programming. This can be done using the following syntax:

```
for i=0:(period_time*step_time)
sim.RunSingleStep;
end
```

Alternately,

```
vsim.Simulation.RunSingleStep
```

To run a simulation continuously,

```
set(vsim.Simulation, 'AttValue', 'SimPeriod', period_time);
break = 200;
set(vsim.Simulation, 'AttValue', 'SimBreakAt', break);
```

To set the maximum speed of the simulation:

```
set(vsim.Simulation, 'AttValue', 'UseMaxSimSpeed', true);
```

Hint: to change the speed use:

```
set(vsim.Simulation, 'AttValue', 'SimSpeed', 10);
```

Where, 10 = 10 Sim. sec./s

Run the simulation continuously:

vsim.Simulation.RunContinuous;

To stop the simulation:

vsim.Simulation.Stop;

5.8 Datapoint Collection and Accessing Results

Vissim offers a variety of data collection results that can be collected any user defined location or coordinate. Different parameters like vehicle average speeds, queue length, occupancy rate, queue delays, travel times, etc. can be calculated and used for post-processing. Once the datapoints are defined and set, the results are exported in a attribute file (.att) which can be accessed through different software like excel and even in MATLAB. We first define what parameters we would like to obtain by programming it in our editor and we can use it in our for loop for optimization problems. For example, in our case study we could collect densities of road sections and if it is above a certain value and there is congestion in the road network, we could reduce the flow or increase the desired speed decisions dynamically at every time step if required using if else statements. The syntax to accessing datapoints and collecting required data is shown below:

For number of vehicles:

datapoint.get('AttValue',Vehs(Current,Last,All)')

For speed:

datapoint.get('AttValue',Speed(Current,Last,All)')

Here, AttValue means the attribute value we would like to get, Vehs,Speed being the attributes we want to obtain for the Current simulation, Last time step and for All type of vehicles. We can change this by changing the syntax according to our needs. For example for the current simulation, average of all time steps and for cars, the syntax would be:

For number of Cars in the network:

```
datapoint.get('AttValue',Vehs(Current,Avg,10)')
```

Where 10 is the vehicle class for cars and avg means the average of all time steps.

5.9 Saving the Files and Releasing the Server

Once the simulation is complete, MATLAB can be used to save the changes made to the network and layout. Once the files are saved, we can release the COM server which will also close the Vissim GUI and the simulation will be over. This can be done using the following syntax:

To save the network and layout:

```
Netfile= fullfile(PathCOM, 'COMsetup.inpx');
vsim.SaveNetAs(Netfile)
Layfile = fullfile(PathCOM, 'COMsetup.layx');
vsim.SaveLayout(Layfile)
```

To release and end Vissim:

```
vsim.release
```

This brings an end to the Vissim-COM program. As mentioned earlier, the Vissim-COM is beneficial in changing required parameters dynamically. However, you cannot change the read-only parameters in between the simulation or once the simulation has begun. You can only change parameters which are already programmed in to be changed at stipulated times where there is a break in simulation, at the beginning or at the end of the simulation.

CHAPTER 6: A CASE STUDY

To verify the performance of the developed platform, we designed a case study. For this case study, a vehicle input of 3000 vehicles per hour was initially set for the road network. Also, the vehicles are travelling from west to east. The road network is discretized into 4 nodes and data collection points are set up at the start and end of each road to collect the number the vehicles. For this paper, we shall be analyzing node 2 and node 3 of the traffic network. The simulation was set to run for 2475 seconds with step time 1 sec and a random seed of 42. We break the simulation at each 225 seconds to study the traffic scenario for that time period. Data collection points at the entry and exit to each node collect data after each break for every 15 seconds. The data being collected is the number of vehicles passing the data collection point. We also evaluate every link for its density, speed and volume of cars at each node using the link evaluation feature of VISSIM.

In this case study, a non signalized road network is designed as mentioned in section 4. A jam scenario is created by introducing a block for the first break. The controller kicks in and analyses the jam using the queue delay results collected at the end of each 225 seconds. If the queue delay is more than 5 seconds for any node, the controller switches on, analyses the traffic scenario for each node and suggests a new optimal velocity and flow for each node. The system is in a state of congestion for the first and second break after which from the new speeds limits, the network is optimized.The simulation was run for 2475 seconds and the results for the second node and third node were graphed. The results are displayed below. The results are shown for each node. First we display results for node 2 and then for node 3. Speed, flow, and density graphs are shown at each time step and the characteristics are analysed in

detail.

6.1 Traffic Scenarios

The traffic networks were setup to simulate a jam in the initial phases of the simulation to test the extremum seeking controller working and check whether the traffic controller is maximizing flow earlier than expected. Figure 6.1 depicts the traffic scenario at 250 seconds. As seen from the figure, there is congestion in both the nodes that we are aiming to analyse.

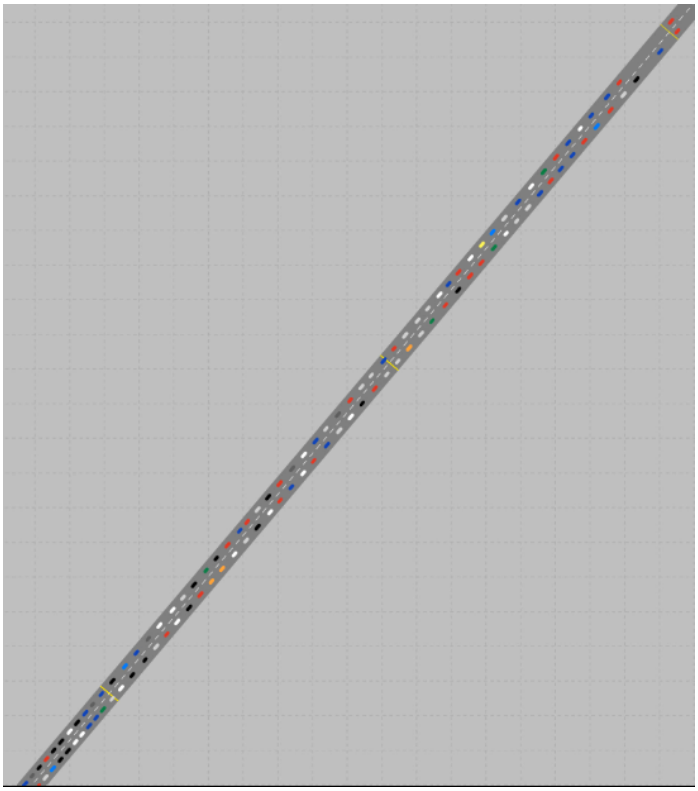

Figure 6.1: Comparison of Traffic Network Scenarios at 450 seconds

Further moving ahead in the simulation, figure 6.2 compares the traffic scenario for the baseline (without controller) and optimized (working controller) cases. This concludes that the extremum seeking controller is working in conjunction with the real time traffic model and maximizing the flow rate.

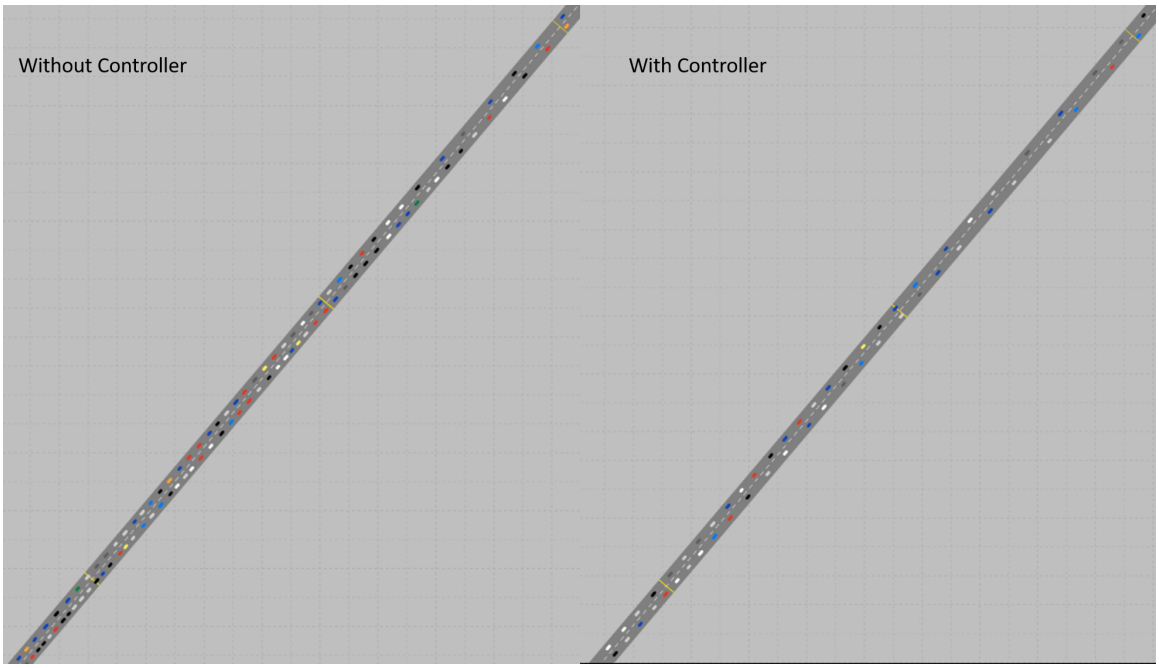

Figure 6.2: Comparison of Traffic Network Scenarios at 450 seconds

6.2 Node 2 Results

Density, flow and velocity for the first node at every 75 seconds are displayed below for each scenario viz. with and without the controlled speed limits. As seen, when we use the traffic controller, the simulation reached optimal free flow quicker. Here, the graphs are limited to 900 seconds i.e for 4 breaks after which the simulation stabilizes.

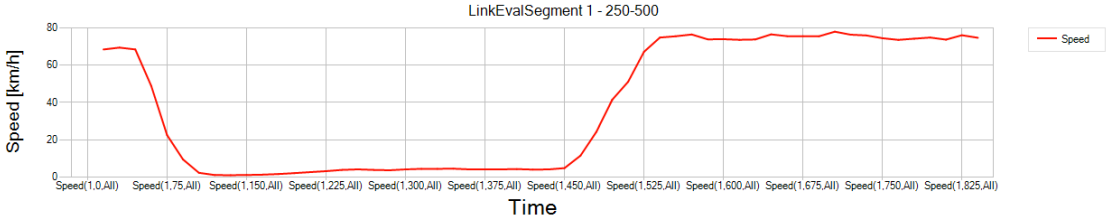

Figure 6.3: Node 2: Speed vs. Time Without Controller

Shown above are the speed changes with time from the second node for both the baseline and optimized case. As seen from the graphs, the vehicles velocity start at 70 km/hr, stay constant and then reduce over time to about 5 km/hr by 100 seconds for both the cases. These remain same for different amount of time for the two cases.

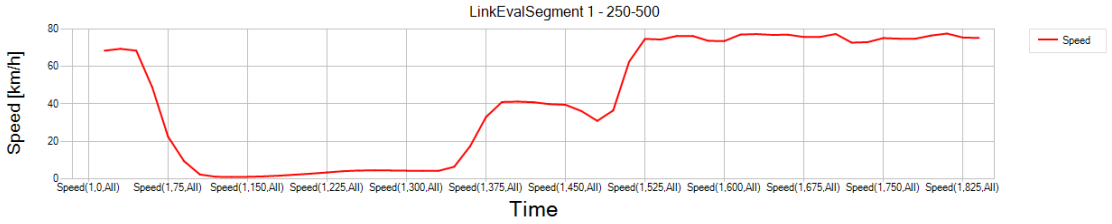

Figure 6.4: Node 2: Speed vs. Time With Controller

For the baseline case with no controller, the speeds remain at 5 km/hr for 450 seconds which is the initialization condition for creating the jam. After this the speeds start increasing and attain maximum free flow velocity for the road network. However, for the optimized case, the controller kicks in at 225 seconds due to the network state feedback from VISSIM and sets new speeds to increase traffic flow. As seen from the graphs, the speeds increase to 40 km/hr based on the controller and remain so until the next break where data transfer takes places i.e. at 450 seconds. Again the controller gives new outputs depending on the states of the system and increases the velocity further to reach optimal velocity earlier than the baseline case.

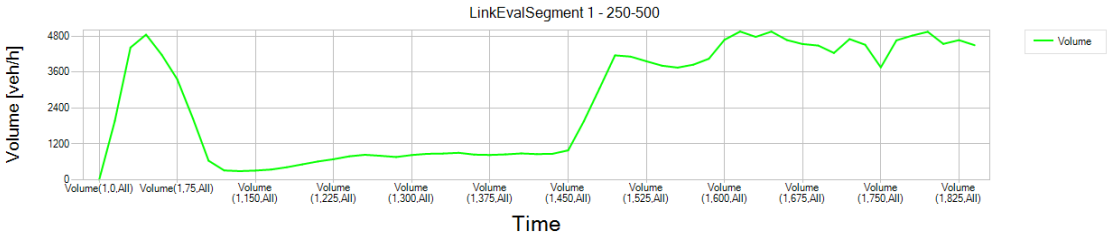

Figure 6.5: Node 2: Volume vs. Time Without Controller

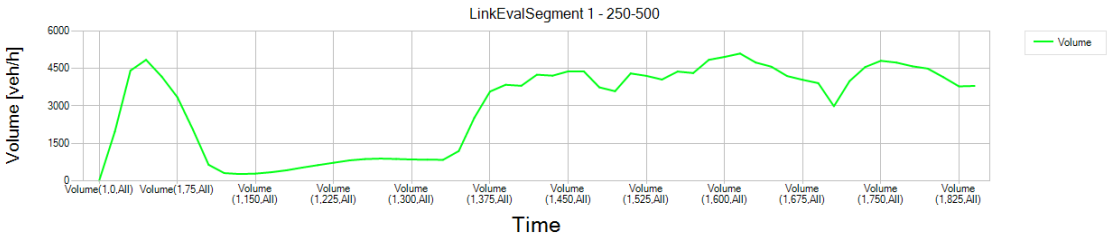

Figure 6.6: Node 2: Volume vs. Time With Controller

A similar trend as the speeds is shown above by the volume changes with time

graphs. For both the baseline and the optimized case, the system initializes, reaches maximum volume when the speeds are higher and similarly drops down to a couple hundred vehicles per hour by the end of 75 seconds. This volume remains constant for up to 450 seconds too for the baseline case after which the system starts attaining free flow. However, again for the optimized case, after the controller gives control instructions, the system volume start increasing earlier than that of the baseline case.

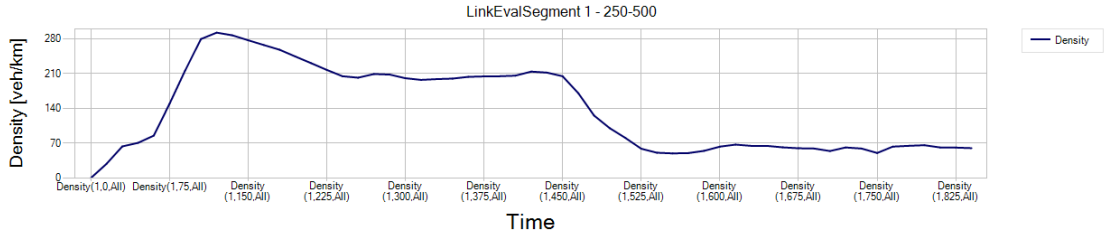

Figure 6.7: Node 2: Density vs. Time Without Controller

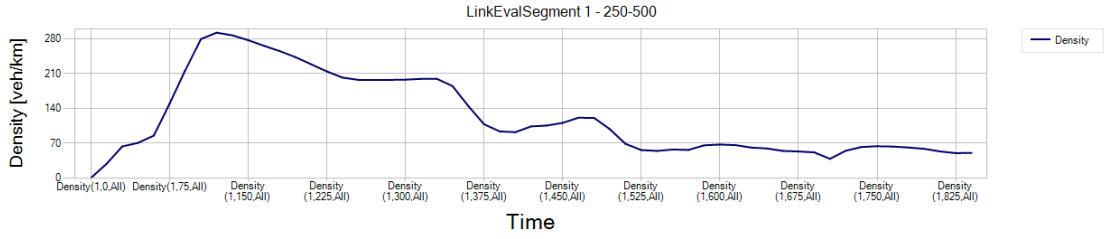

Figure 6.8: Node 2: Density vs. Time With Controller

The density graphs depicted in the figures above, show the density fluctuations over time. The densities up to 225 seconds before the controller kicks in are the same but once the controller starts when the queue delays reach higher levels, the densities for the optimized case start reducing and the system starts its path towards attaining free flow. The densities reach a maximum of 292 vehicles per kilometer (veh/km) which is the jam density at 135 seconds and reduce to 210 veh/km by the end of 225 seconds. The densities remain at 210 veh/km for 450 seconds and 350 seconds for the baseline and optimized cases respectfully. The densities reach critical density or free flow density quicker for the optimized case.

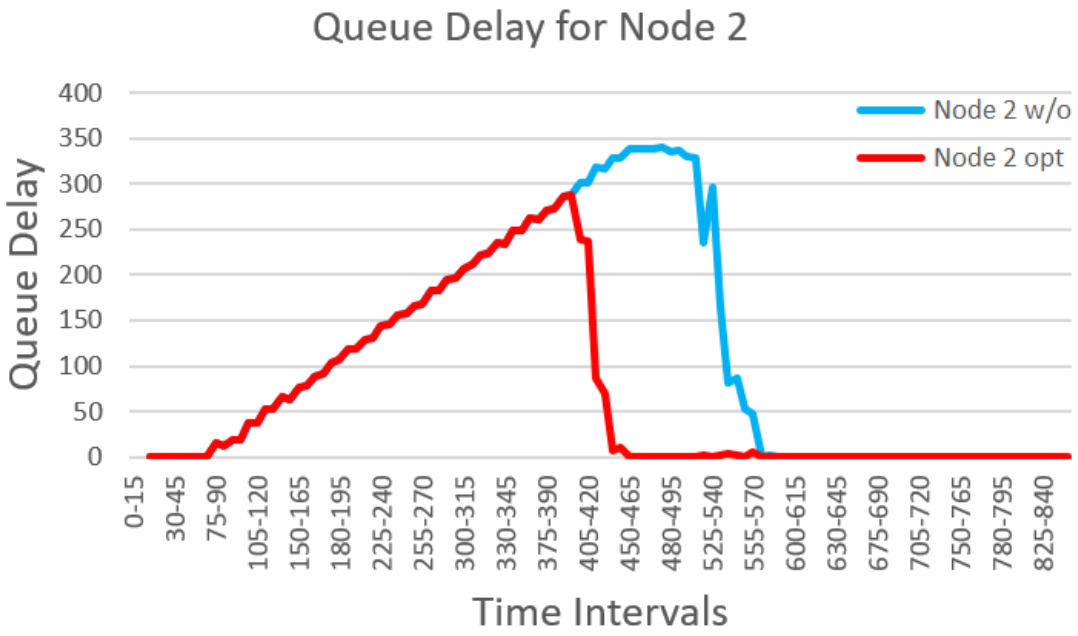

Figure 6.9: Node 2: Queue Delays

The queue delay results are shown in figure 6.9. As seen from the graph, for the second node, the blue line graph denotes the queue delay times for the baseline case without the controller whereas the red line graph denotes the queue delay times for the optimized case with the controller. This graph shows us clearly that the optimized case queue delays were significantly lower than than the baseline case. For the optimized case, the maximum delay time value was around 250 seconds whereas for the baseline case the maximum delay time was around 350 seconds. That is almost a difference of a 100 seconds. Also, the delay times for the optimized case dropped to near zero by 465 seconds whereas the baseline case took around 570 seconds to drop to a near zero value. This graph shows us that the integrated VISSIM-MATLAB platform has managed to integrate our controller with the traffic model in VISSIM and through its controlled speed decisions, has lowered queue delays and eventually reduced congestion and increased flow.

6.3 Node 3 Results

Similarly like for node 2, density, flow and velocity for the second node at every 75 seconds are displayed below for each scenario viz. with and without the controlled speed limits. From the graphs, when we use the traffic controller, the simulation reached optimal free flow quicker. Here, the graphs are limited to 900 seconds i.e for 4 breaks after which the simulation stabilizes. A prominent difference between the baseline and optimized cases are seen in the results for node 3. This is due to the vehicles exiting the second node at higher speeds set by the controller than for the first node.

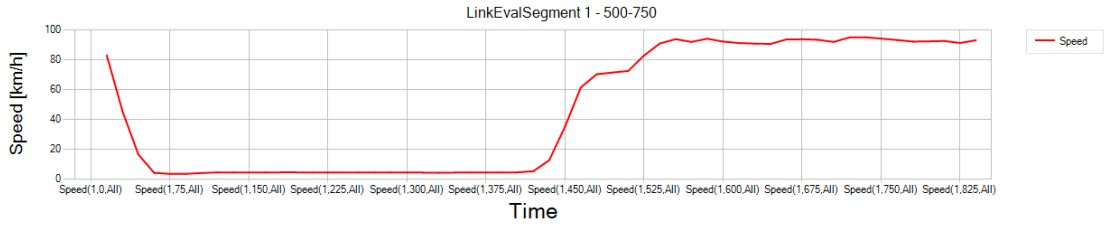

Figure 6.10: Node 3: Speed vs. Time Without Controller

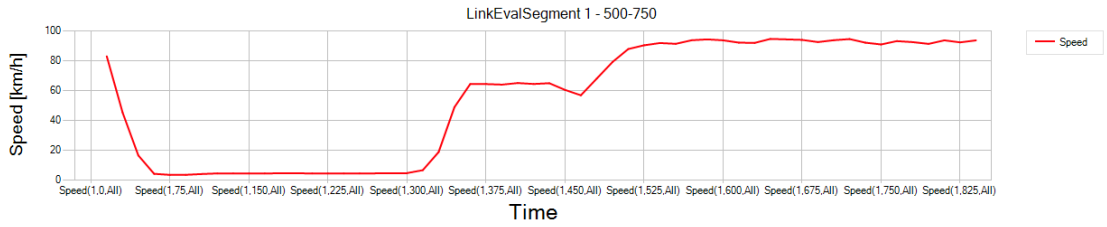

Figure 6.11: Node 3: Speed vs. Time With Controller

Shown above are the speed changes with time from the third node for both the baseline and optimized case. As seen from the graphs, the vehicles velocity start at 80 km/hr, stay constant and then reduce over time to about 5 km/hr by 60 seconds for both the cases. These remain same for different amount of time for the two cases. For the baseline case with no controller, the speeds remain at 5 km/hr for 400 seconds which is the initialization condition for creating the jam. After this the speeds start

increasing and attain maximum free flow velocity for the road network. However, for the optimized case, the controller kicks in at 225 seconds due to the network state feedback from VISSIM and sets new speeds to increase traffic flow. As seen from the graphs, the speeds increase to 60 km/hr based on the controller and remain so until the next break where data transfer takes places i.e. at 450 seconds. Again the controller gives new outputs depending on the states of the system and increases the velocity further to reach optimal velocity earlier than the baseline case.

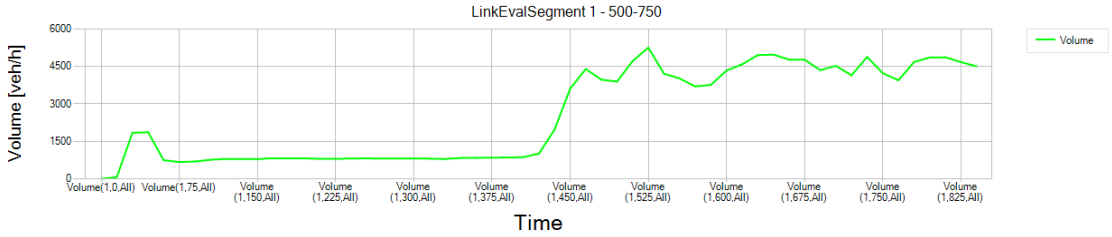

Figure 6.12: Node 3: Volume vs. Time Without Controller

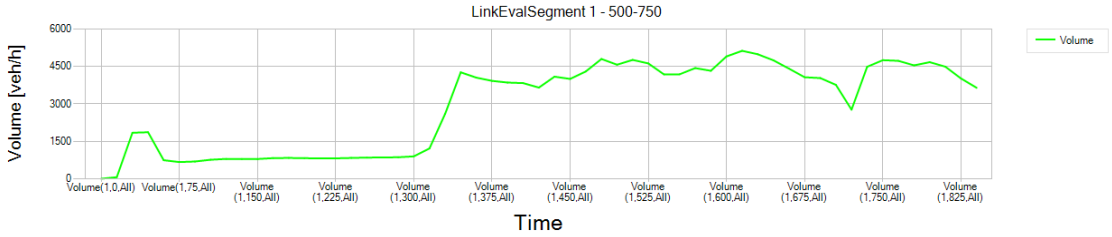

Figure 6.13: Node 3: Volume vs. Time With Controller

A similar trend as the speeds is shown above by the volume changes with time graphs. For both the baseline and the optimized case, the system initializes, reaches maximum volume when the speeds are higher and similarly drops down to a couple hundred vehicles per hour by the end of 60 seconds. This volume remains constant for up to 400 seconds for the baseline case after which the system starts attaining free flow. However, again for the optimized case, after the controller gives control instructions, the system volume start increasing earlier (at about 300 seconds) than that of the baseline case.

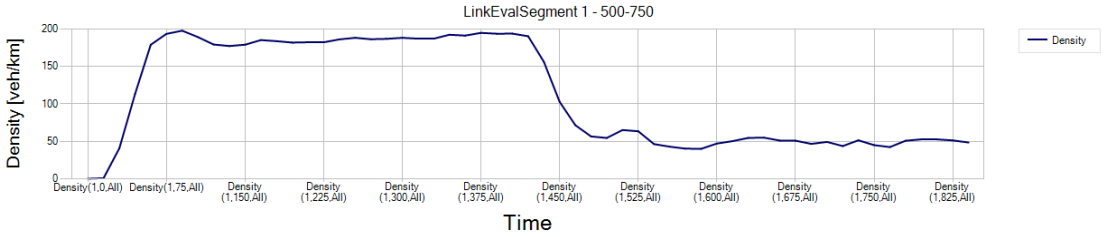

Figure 6.14: Node 3: Density vs. Time Without Controller

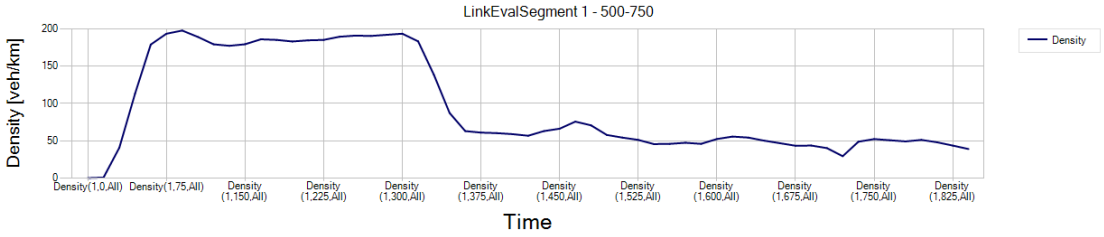

Figure 6.15: Node 3: Density vs. Time With Controller

The density graphs depicted in the figures 6.14 and 6.15, show the density fluctuations over time. The densities up to 300 seconds are the same but once the controller starts when the queue delays reach higher levels, the densities for the optimized case start reducing and the system starts its path towards attaining free flow. The densities reach a maximum of 200 vehicles per kilometer (veh/km) which is the jam density at 100 seconds and reduce to 50 veh/km by the end of 375 seconds. The densities remain at 200 veh/km for 400 seconds and 300 seconds for the baseline and optimized cases respectfully. The densities reach critical density or free flow density quicker for the optimized case.

The queue delay results are shown in figure 6.16. As seen from the graph, similarly like node 2 for node 3 too, the blue line graph denotes the queue delay times for the baseline case without the controller whereas the red line graph denotes the queue delay times for the optimized case with the controller. This graph in figure 6.16 too shows us clearly that the optimized case queue delays were significantly lower than than the baseline case. For the optimized case, the maximum delay time value was around 250 seconds whereas for the baseline case the maximum delay time was around

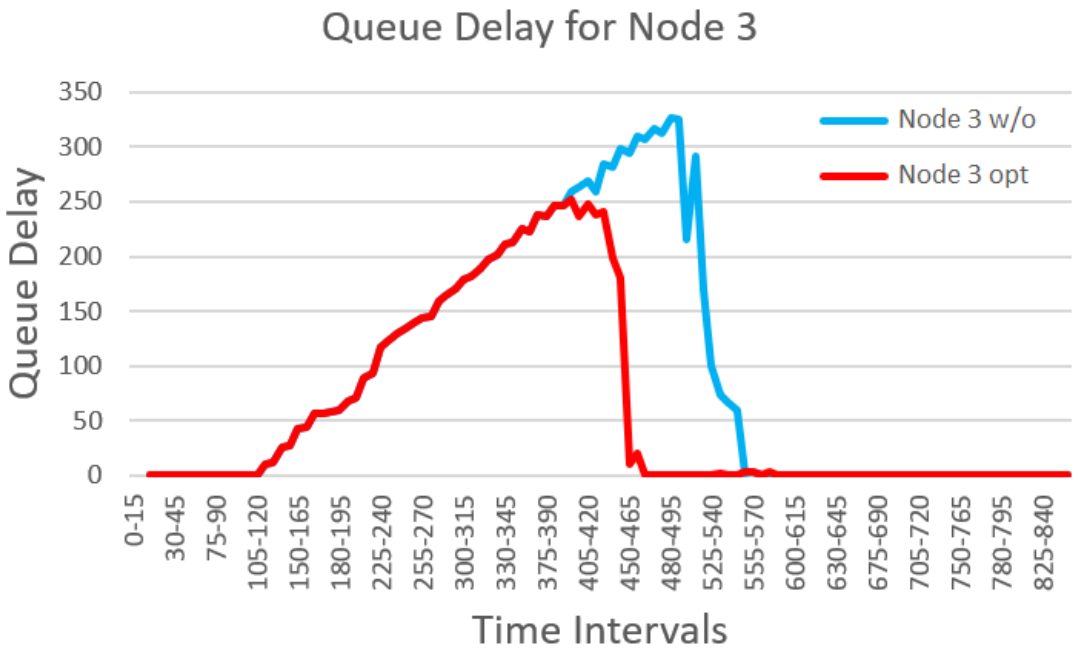

Figure 6.16: Node 3: Queue Delays

350 seconds. That is almost a difference of a 100 seconds. Also, the delay times for the optimized case dropped to near zero by 465 seconds whereas the baseline case took around 570 seconds to drop to a near zero value. This graph shows us that the integrated VISSIM-MATLAB platform has managed to integrate our controller with the traffic model in VISSIM and through its controlled speed decisions, has lowered queue delays and eventually reduced congestion and increased flow.

CHAPTER 7: CONCLUSION AND FUTURE WORK

This thesis is focused on developing a platform between VISSIM and MATLAB to employ different types of traffic control strategies. The math model and dynamics of the system in Chapter 2 focused on the equations of motion of the system and this model was successfully developed, designed and tuned in MATLAB. The real world traffic model was designed in PTV VISSIM such that exact geometry and traffic conditions were replicated. Microsoft developed an idea of a binary interface standard for software components which is known as the component object model (COM). Inter-process communication object creation is possible using a COM. A component object model (COM) server was created in MATLAB called ActiveX and was used as a virtual port between the real world testing software and our controller model. This virtual port or COM integrated MATLAB-VISSIM-SIMULINK to validate our control techniques and check for data and control command transfer between software. In particular, in this thesis, we modeled a non-signalized traffic system in VISSIM. To maximize the throughput of a non-signalized traffic network, an extremum seeking controller is developed. An extremum seeking approach is used in this research as the extremum seeking controller is a non model based controller and due to the limited knowledge of the system states. By using COM programming, we were able to transfer the flow rate between different sub networks from VISSIM to MATLAB. Knowing the current state of the traffic network, an optimal velocity is generated in MATLAB and transferred to VISSIM. The case study demonstrated that the integration was successful and shows that the controller is optimizing the flow of traffic effectively and the data and control signal transfer between VISSIM-MATLAB-CONTROLLER is smooth and without any discrepancies and latency's.

In the future, work towards the development of a managed lane highway with managed lane changes and variable speed limits is underway. A heterogeneous traffic network consisting of connected autonomous vehicle (CAV) and human driven vehicles is being designed in VISSIM and the dynamics and equations of motion of the system are being formulated. Also, a plan of simulating for a more complex road geometry with more number of lanes, signalized intersections and heterogeneous traffic is thought of. Different control strategies including model predictive control are thought to be implemented to control various different aspects of the road network. Different reasons for congestion will be worked on and optimized using this platform. This platform will form a base in implementing control strategies for traffic models. Hence a collaboration of control engineering and transportation engineering will pave way to intelligent transportation systems (ITS). ITS will make room for connected autonomous vehicles (CAVs) and hence more effective travel, added safety, lesser fuel consumption, lower travel times will be achievable. Ultimately the cost to the country will reduce. A model predictive control technique is planned to be implemented to predict possible outcomes and reduce congestion.

REFERENCES

[1] Z. Zhou, B. De Schutter, S. Lin, and Y. Xi, "Two-level hierarchical model-based predictive control for large-scale urban traffic networks," *IEEE Transactions on Control Systems Technology*, vol. 25, no. 2, pp. 496–508, 2017.

[2] G. Cookson and B. Pishue, "Inrix global traffic scorecard–appendices," *INRIX research*, 2017.

[3] W. Sun, J. Zheng, and H. X. Liu, "A capacity maximization scheme for intersection management with automated vehicles," *Transportation research procedia*, vol. 23, pp. 121–136, 2017.

[4] A. Ghiasi, O. Hussain, Z. S. Qian, and X. Li, "A mixed traffic capacity analysis and lane management model for connected automated vehicles: A markov chain method," *Transportation Research Part B: Methodological*, vol. 106, pp. 266–292, 2017.

[5] G. C. Petrisor, "Traffic lane management system," Jan. 15 2009. US Patent App. 12/080,721.

[6] M. G. Jabori, R. V. Lakdawala, and Q. Chen, "Dynamic lane management system and method," Nov. 4 2008. US Patent 7,447,824.

[7] M. Papageorgiou and A. Kotsialos, "Freeway ramp metering: An overview," *IEEE transactions on intelligent transportation systems*, vol. 3, no. 4, pp. 271–281, 2002.

[8] J. Ma, X. Li, S. Shladover, H. A. Rakha, X.-Y. Lu, R. Jagannathan, and D. J. Dailey, "Freeway speed harmonization," *IEEE Transactions on Intelligent Vehicles*, vol. 1, no. 1, pp. 78–89, 2016.

[9] D. A. Lazar, R. Pedarsani, K. Chandrasekher, and D. Sadigh, "Maximizing road capacity using cars that influence people," in *2018 IEEE Conference on Decision and Control (CDC)*, pp. 1801–1808, IEEE, 2018.

[10] C. Ross and S. Guhathakurta, "Autonomous vehicles and energy impacts: a scenario analysis," *Energy Procedia*, vol. 143, pp. 47–52, 2017.

[11] L. Ye and T. Yamamoto, "Modeling connected and autonomous vehicles in heterogeneous traffic flow," *Physica A: Statistical Mechanics and its Applications*, vol. 490, pp. 269–277, 2018.

[12] J. Liu, K. Kockelman, and A. Nichols, "Anticipating the emissions impacts of smoother driving by connected and autonomous vehicles, using the moves model," in *Transportation Research Board 96th Annual Meeting*, 2017.

[13] N. J. Goodall, B. L. Smith, and B. Park, "Traffic signal control with connected vehicles," *Transportation Research Record*, vol. 2381, no. 1, pp. 65–72, 2013.

[14] P. Bansal and K. M. Kockelman, "Forecasting americansâ long-term adoption of connected and autonomous vehicle technologies," *Transportation Research Part A: Policy and Practice*, vol. 95, pp. 49–63, 2017.

[15] A. Talebpour and H. S. Mahmassani, "Influence of connected and autonomous vehicles on traffic flow stability and throughput," *Transportation Research Part C: Emerging Technologies*, vol. 71, pp. 143–163, 2016.

[16] M. P. Vitus and C. J. Tomlin, "A probabilistic approach to planning and control in autonomous urban driving," in *Decision and Control (CDC), 2013 IEEE 52nd Annual Conference on*, pp. 2459–2464, IEEE, 2013.

[17] D. Sadigh and A. Kapoor, "Safe control under uncertainty," *arXiv preprint arXiv:1510.07313*, 2015.

[18] R. Szalai and G. Orosz, "Decomposing the dynamics of heterogeneous delayed networks with applications to connected vehicle systems," *Physical Review E*, vol. 88, no. 4, p. 040902, 2013.

[19] A. Gray, Y. Gao, J. K. Hedrick, and F. Borrelli, "Robust predictive control for semi-autonomous vehicles with an uncertain driver model," in *Intelligent vehicles symposium (IV), 2013 IEEE*, pp. 208–213, IEEE, 2013.

[20] V. Raman, A. Donzé, D. Sadigh, R. M. Murray, and S. A. Seshia, "Reactive synthesis from signal temporal logic specifications," in *Proceedings of the 18th international conference on hybrid systems: Computation and control*, pp. 239–248, ACM, 2015.

[21] F.-Y. Wang *et al.*, "Agent-based control for networked traffic management systems," 2005.

[22] N. Geroliminis, J. Haddad, and M. Ramezani, "Optimal perimeter control for two urban regions with macroscopic fundamental diagrams: A model predictive approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 1, pp. 348–359, 2013.

[23] K. Aboudolas and N. Geroliminis, "Perimeter and boundary flow control in multi-reservoir heterogeneous networks," *Transportation Research Part B: Methodological*, vol. 55, pp. 265–281, 2013.

[24] M. Hajiahmadi, J. Haddad, B. De Schutter, and N. Geroliminis, "Optimal hybrid perimeter and switching plans control for urban traffic networks," *IEEE Transactions on Control Systems Technology*, vol. 23, no. 2, pp. 464–478, 2015.

[25] S. Lin, T. Ling, and Y. Xi, "Model predictive control for large-scale urban traffic networks with a multi-level hierarchy," in *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*, pp. 211–216, IEEE, 2013.

[26] S. Lin, Q.-J. Kong, and Q. Huang, "A simulation analysis on the existence of network traffic flow equilibria," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 4, pp. 1706–1713, 2014.

[27] S. Lin and Q.-J. Kong, "A model-based demand-balance control for complex urban traffic networks," in *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pp. 2900–2905, IEEE, 2014.

[28] A. Ghasemi, R. Nikbakhti, A. Ghasemi, F. Hedayati, and A. Malvandi, "Parallelized numerical modeling of the interaction of a solid object with immiscible incompressible two-phase fluid flow," *Engineering Computations*, vol. 34, no. 3, pp. 709–724, 2017.

[29] A. Ghesemi, J. Fox, and A. Husic, "Predicting macroturbulence energy and timescales for flow over a gravel bed: Experimental results and scaling laws," *Geomorphology*, vol. 332, pp. 122–137, 2019.

[30] J. Haddad, M. Ramezani, and N. Geroliminis, "Cooperative traffic control of a mixed network with two urban regions and a freeway," *Transportation Research Part B: Methodological*, vol. 54, pp. 17–36, 2013.

[31] P. Bansal, "Matlab-vissim interface for online optimization of green time splits," *21st World Congress on Intelligent Transport Systems, ITSWC 2014: Reinventing Transportation in Our Connected World*, 01 2014.

[32] H. Cao and J. Luo, "Research on vissim-matlab integrated traffic simulation platform based on com interface technology," in *2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, pp. 305–309, IEEE, 2019.

[33] T. Tettamanti and I. Varga, "Development of road traffic control by using integrated vissim-matlab simulation environment," *Periodica Polytechnica Civil Engineering*, vol. 56, no. 1, pp. 43–49, 2012.

[34] X. Zotos, F. Naef, and P. Prelovsek, "Transport and conservation laws," *Physical Review B*, vol. 55, no. 17, p. 11029, 1997.

[35] C. F. Daganzo, "The cell transmission model, part ii: network traffic," *Transportation Research Part B: Methodological*, vol. 29, no. 2, pp. 79–93, 1995.

[36] H. Malek, S. Dadras, and Y. Chen, "Performance analysis of fractional order extremum seeking control," *ISA transactions*, vol. 63, pp. 281–287, 2016.

[37] N. Xiao, Y. Li, Y. Luo, E. Frazzoli, W. yu, and D. Wang, "Vissim simulation for extended back-pressure traffc signal control strategy," 06 2015.

[38] T. Tettamanti and M. T. Horváth, "A practical manual for vissim com programming in matlab-for vissim 9 and 10," 2018.