COOPERATIVE DECENTRALIZED INTERSECTION COLLISION AVOIDANCE USING EXTENDED KALMAN FILTERING

Ashil Sayyed Farahmand

Thesis submitted to the faculty of the Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
In
Electrical Engineering

December 5, 2008
Falls Church, VA

Keywords: cooperative collision avoidance, extended kalman filter (EKF), advanced vehicle control systems (AVCS), intelligent transportation systems (ITS), vehicular ad-hoc network (VANET)

COOPERATIVE DECENTRALIZED INTERSECTION COLLISION AVOIDANCE USING EXTENDED KALMAN FILTERING

Ashil Sayyed Farahmand

ABSTRACT

Automobile accidents are one of the leading causes of death and claim more than 40,000 lives annually in the US alone. A substantial portion of these accidents occur at road intersections. Stop signs and traffic signals are some of the intersection control devices used to increase safety and prevent collisions. However, these devices themselves can contribute to collisions, are costly, inefficient, and are prone to failure. This thesis proposes an adaptive, decentralized, cooperative collision avoidance (CCA) system that optimizes each vehicle's controls subject to the constraint that no collisions occur. Three major contributions to the field of collision avoidance have resulted from this research. First, a nonlinear 5-state variable vehicle model is expanded from an earlier model developed in [1]. The model accounts for internal engine characteristics and more realistically approximates vehicle behavior in comparison to idealized, linear models. Second, a set of constrained, coupled Extended Kalman Filters (EKF) are used to predict the trajectory of the vehicles approaching an intersection in real-time. The coupled filters support decentralized operation and ensure that the optimization algorithm bases its decisions on good, reliable estimates. Third, a vehicular network based on the new WAVE standard is presented that provides cooperative capabilities by enabling intervehicle communication. The system is simulated against today's common intersection control devices and is shown to be superior in minimizing average vehicle delay.

Acknowledgements

I would like to take this opportunity to thank the people who have been instrumental in helping me to complete the research and writing of this thesis.

First, I thank my advisor Dr. Lamine Mili for continually pushing me on and not allowing me to fall behind on my research. His constant encouragement is greatly appreciated and has been vital to completing this paper.

I am thankful to Dr. Shinya Kikuchi and Dr. Daniel Stilwell for serving on my committee and helping to prepare my thesis for the defense. Your suggestions were very helpful in directing my research.

I would also like to thank Dr. Yao Liang and Dr. Luiz DaSilva for providing expert advice on issues in mobile networking.

I thank Greg Schoenig and the rest of Dr. Mili's students that have provided recommendations for my research. I would like to especially thank Mital Gandhi for painstakingly editing this thesis.

Lastly, I would like to thank my family for always being there to support me.

Contents

ACKNOWLEDGEMENTS .... **III**

CONTENTS .... **IV**

LIST OF FIGURES .... **VI**

LIST OF TABLES .... **IX**

ACRONYMS .... **X**

CHAPTER 1 - INTRODUCTION **1**

1.1 BACKGROUND OF INTERSECTION CONTROL DEVICES 1
1.1.1 Stop Signs 1
1.1.2 Traffic Signals 2
1.1.3 Traffic Circles 3
1.2 LIMITATIONS OF CURRENT INTERSECTION CONTROL DEVICES 5
1.2.1 Safety 6
1.2.2 Efficiency 7
1.2.3 Cost 8
1.3 CONTRIBUTIONS 8
1.4 THESIS OVERVIEW 10

CHAPTER 2 - LITERATURE REVIEW **12**

2.1 HISTORY 12
2.2 OPTICAL COLLISION AVOIDANCE SYSTEMS 13
2.3 COOPERATIVE COLLISION AVOIDANCE (CCA) SYSTEMS 15
2.3.1 Microscopic Approach to CCA Research 15
2.3.2 Macroscopic Approach to CCA Research 16
2.4 TRENDS IN CCA RESEARCH 17

CHAPTER 3 - VEHICLE POWERTRAIN STATE SPACE MODEL **19**

3.1 PRINCIPLES OF AUTOMOBILE POWERTRAINS 19
3.2 STATE SPACE MODELING 21
3.2.1 Control Variables 21
3.2.2 State Variables and Dynamic Equations for the Vehicle 23
3.2.3 Assumed Model Parameters 29
3.3 SUMMARY 34

CHAPTER 4 - LINEARIZATION, DISCRETIZATION, AND THE EXTENDED KALMAN FILTER FOR VEHICLE MODELING **35**

4.1 KALMAN FILTER FOR LINEAR MODEL 35
4.1.1 Prediction Step 37
4.1.2 Filtering\Update Step 38
4.2 THE EXTENDED KALMAN FILTER 39
4.2.1 Linearization 40
4.2.2 Discretization 41
4.2.3 Prediction Step 45
4.2.4 Filtering\Update Step 45
4.2.5 EKF Implementation for Vehicle Modeling 48
4.3 NUMERICAL INTEGRATION TECHNIQUES 49
4.3.1 Euler Method 50
4.3.2 4th-order Runge-Kutta Method 50
4.4 SUMMARY 51

CHAPTER 5 - DEVELOPING A NETWORK FOR VEHICULAR CONTROL AT INTERSECTIONS 53

5.1 Wireless Access in the Vehicular Environment (WAVE) 53
5.1.1 IEEE 1609.1 - Resource Manager 54
5.1.2 IEEE 1609.2 - Security Services....................55
5.1.3 IEEE 1609.3 - Networking Services....................57
5.1.4 IEEE 1609.4 - Multi-channel Operations....................58
5.2 Cooperative Vehicle Optimization Algorithm.................... 58
5.2.1 Sparse Traffic....................59
5.2.2 Dense Traffic....................64
5.3 Summary.................... 69

CHAPTER 6 - STABILITY ANALYSIS OF VEHICLE MODEL AND NETWORK....................70

6.1 Vehicle State Space Model Stability.................... 70
6.1.1 Equilibrium Points....................70
6.1.2 Modes of Oscillation....................72
6.1.3 Determining the Equilibrium Points' Basins of Attraction 73
6.1.4 State Oscillations....................77
6.1.5 Transmission Instability....................80
6.2 Stability Analysis of the Sparse Traffic Optimization Algorithm 89
6.3 Summary.................... 93

CHAPTER 7 - SIMULATION RESULTS 94

7.1 Simulator Overview.................... 94
7.2 Comparison of the Euler and 4TH-order Runge-Kutta Methods on Integration of Vehicle Model.................... 96
7.3 Determination of System Parameters....................106
7.4 Determination of the Brake Constant of Proportionality....................107
7.5 Determination of Brake Angle as a Function of Deceleration....................109
7.6 Linear Approximation of Vehicle Acceleration 111
7.7 Optimization Algorithm Results....................113
7.7.1 Comparison of Stop Sign to Sparse Traffic Optimization Algorithm....................113
7.7.2 Comparison of Traffic Signal to Sparse Traffic Optimization Algorithm....................114
7.7.3 Comparison of Traffic Signal to Dense Traffic Optimization Algorithm....................116

CHAPTER 8 - CONCLUSIONS AND FUTURE RESEARCH....................120

CHAPTER 9 - REFERENCES 123

APPENDIX A 130

List of Figures

Figure 1.1: Average Delay of Vehicles at Roundabout and Traffic Signal Controlled Intersections as a Function of Approach Volume. The Approach Volume is Comparable to the Minimum Required to Justify Installation of a Traffic Signal. Plot Obtained from [15]. 5
Figure 1.2: Simplified Representation of Interaction Between Different System Stages. 9
Figure 3.1: Normalized Throttle Characteristics with Respect to Throttle Angle, plot of actual data and piecewise continuous fit copied from [1], plot of approximate fit overlaid 22
Figure 3.2: Definition of Steering Angle 23
Figure 3.3: Normalized Pressure Influence Function as a Function of the Ratio Between the Pressure in the Intake Manifold and the Atmosphere. Plot obtained from [1]. 25
Figure 3.4: Gear with Respect to Time for $a = 85^0$, and $b = 0^0$ 32
Figure 3.5: Vehicle Velocity with Respect to Time for $a = 85^0$, and $b = 0^0$ 33
Figure 3.6: Engine Angular Velocity with Respect to Time for $a = 85^0$, and $b = 0^0$. 34
Figure 5.1: Locations of WAVE Applications During a Typical Instant of Vehicle-to-Infrastructure Communication 55
Figure 5.2: Illustration of Optimization Variables 60
Figure 5.3: Vehicle Flow Rate With Respect to Time and Signal Phase Color 65
Figure 5.4: Example Intersection Scenario. 68
Figure 6.1: Phase Plot of Vehicle State Space Model Stable Equilibrium Points Determined Using Newton-Raphson Method For Throttle Angles 5-85^O, $b = 0^0$, $T_b = 0$, $R_g = \mathrm{R}_5$ and Error = 10^{-5}. 72
Figure 6.2: Engine Angular Velocity with Respect to Time for $a = 85^0$, $b = 0^0$, $R_g = \mathrm{R}_5$, and initial conditions m_a = 0.004 kg and w_e = -43 rad/s. 74
Figure 6.3: Engine Angular Velocity with Respect to Time for $a = 85^0$, $b = 0^0$, $R_g = \mathrm{R}_5$, and initial conditions m_a = 0.004 kg and w_e = -44 rad/s. 75
Figure 6.4: Boundary of Basin of Attraction for $a = 85^0$, $b = 0^0$, and $R_g = R_5$. The red line is an interpolation of the boundary. 76
Figure 6.5: Equilibrium Points and the Boundaries of Their Basins of Attraction for Various Throttle Angles, $b = 0^0$, and $R_g = R_5$ 77
Figure 6.6: Engine Angular Velocity vs. Mass of Air in Intake Manifold with Respect to Time for $a = 0.5^0$, $b = 0^0$, $R_g = \mathrm{R}_1$, and initial conditions m_a = 0.001 kg and w_e = 1 rad/s.. 78
Figure 6.7: Engine Angular Velocity vs. Mass of Air in the Intake Manifold with Respect to Time for $a = 0^0$, $b = 0^0$, $R_g = \mathrm{R}_1$, and initial conditions m_a = 0.001 kg and w_e = 1 rad/s. 79
Figure 6.8a: Gear with Respect to Time for $a = 4.5623^0$, $b = 0^0$ and initial conditions m_a = 0.00111331477934 kg and w_e = 29.61976701882556 rad/s. 81
Figure 6.8b: Gear with Respect to Time for $a = 4.5623^0$, $b = 0^0$ and initial conditions m_a = 0.00111331477934 kg and w_e = 29.61976701882556 rad/s. This plot is a magnification of the blue box in Figure 6.8a. 82
Figure 6.9a: Velocity with Respect to Time for $a = 4.5623^0$, $b = 0^0$ and initial conditions m_a = 0.00111331477934 kg and w_e = 29.61976701882556 rad/s. 83

FIGURE 6.9B: VELOCITY WITH RESPECT TO TIME FOR $a = 4.5623^0$, $b = 0^0$ AND INITIAL CONDITIONS m_a = 0.00111331477934 KG AND w_e = 29.61976701882556 RAD/S. THIS PLOT IS A MAGNIFICATION OF THE BLUE BOX IN FIGURE 6.9A. 84
FIGURE 6.10A: MASS OF AIR IN INTAKE MANIFOLD WITH RESPECT TO TIME FOR $a = 4.5623^0$, $b = 0^0$ AND INITIAL CONDITIONS m_a = 0.00111331477934 KG AND w_e = 29.61976701882556 RAD/S. 85
FIGURE 6.10B: MASS OF AIR IN INTAKE MANIFOLD WITH RESPECT TO TIME FOR $a = 4.5623^0$, $b = 0^0$ AND INITIAL CONDITIONS m_a = 0.00111331477934 KG AND w_e = 29.61976701882556 RAD/S. THIS PLOT IS A MAGNIFICATION OF THE BLUE BOX IN FIGURE 6.10A. 86
FIGURE 6.11A: ENGINE ANGULAR VELOCITY WITH RESPECT TO TIME FOR $a = 4.5623^0$, $b = 0^0$ AND INITIAL CONDITIONS m_a = 0.00111331477934 KG AND w_e = 29.61976701882556 RAD/S. 87
FIGURE 6.11B: ENGINE ANGULAR VELOCITY WITH RESPECT TO TIME FOR $a = 4.5623^0$, $b = 0^0$ AND INITIAL CONDITIONS m_a = 0.00111331477934 KG AND w_e = 29.61976701882556 RAD/S. PLOT IS A MAGNIFICATION OF THE BLUE BOX IN FIGURE 6.11A. 88
FIGURE 6.12: GEAR WITH RESPECT TO TIME FOR $a = 10^0$, $b = 0^0$ AND INITIAL CONDITIONS m_a = 0.00111331477934 KG AND w_e = 29.61976701882556 RAD/S. 89
FIGURE 6.13: VEHICLE 1 THROTTLE ANGLE WITH RESPECT TO TIME DETERMINED FROM OPTIMIZATION ALGORITHM EXECUTED NEAR 0 SECONDS. 91
FIGURE 6.14: VEHICLE 1 THROTTLE ANGLE WITH RESPECT TO TIME DETERMINED FROM OPTIMIZATION ALGORITHM EXECUTED AT THE 10 SECOND INTERVAL. 92
FIGURE 6.15: VEHICLE 1 THROTTLE ANGLE WITH RESPECT TO TIME DETERMINED FROM OPTIMIZATION ALGORITHM EXECUTED AT THE 20 SECOND INTERVAL. 93
FIGURE 7.1: SIMULATOR OPERATION FLOW CHART 95
FIGURE 7.2: ENGINE ANGULAR VELOCITY AS A FUNCTION OF TIME DETERMINED USING THE RK4 METHOD. THE INITIAL VALUE USED FOR m_a IS 0.00472741455026. 97
FIGURE 7.3: MASS OF AIR IN INTAKE MANIFOLD WITH RESPECT TO TIME FOR STEP SIZE OF 1 MS DETERMINED USING THE EULER METHOD 99
FIGURE 7.4A: MASS OF AIR IN INTAKE MANIFOLD WITH RESPECT TO TIME FOR STEP SIZE OF 2 MS DETERMINED USING THE EULER METHOD 100
FIGURE 7.4B: MASS OF AIR IN INTAKE MANIFOLD WITH RESPECT TO TIME FOR STEP SIZE OF 2 MS DETERMINED USING THE EULER METHOD. THIS PLOT IS A MAGNIFICATION OF THE BLUE BOX IN FIGURE 7.4A. 101
FIGURE 7.5A: MASS OF AIR IN INTAKE MANIFOLD WITH RESPECT TO TIME FOR STEP SIZE OF 3 MS DETERMINED USING THE EULER METHOD 102
FIGURE 7.5B: MASS OF AIR IN INTAKE MANIFOLD WITH RESPECT TO TIME FOR STEP SIZE OF 3 MS DETERMINED USING THE EULER METHOD. PLOT IS A MAGNIFICATION OF THE BLUE BOX SHOWN IN FIGURE 7.5A. 103
FIGURE 7.6: MASS OF AIR IN INTAKE MANIFOLD WITH RESPECT TO TIME FOR STEP SIZE OF 3 MS DETERMINED USING THE RK4 METHOD. 104
FIGURE 7.7A: MASS OF AIR IN INTAKE MANIFOLD WITH RESPECT TO TIME FOR STEP SIZE OF 4 MS DETERMINED USING THE RK4 METHOD. 105
FIGURE 7.7B: MASS OF AIR IN INTAKE MANIFOLD WITH RESPECT TO TIME FOR STEP SIZE OF 4 MS DETERMINED USING THE RK4 METHOD. PLOT IS A MAGNIFICATION OF THE BLUE BOX IN FIGURE 7.7A. 106
FIGURE 7.8: VEHICLE VELOCITY WITH RESPECT TO TIME DURING MAXIMUM BRAKING. THE BLUE SAMPLE POINTS HAVE BEEN ADDED TO SHOW THE SAMPLES USED TO DETERMINE THE SLOPE. THIS PLOT IS AN EDITED FORM OF FIGURE 20 FROM [1]. 108

FIGURE 7.9: SIMULATED VEHICLE VELOCITY WITH RESPECT TO TIME DURING MAXIMUM BRAKING FOR $K_b = 300 \frac{N \cdot m}{\deg rees}$ 109
FIGURE 7.10: ESTIMATE OF RELATIONSHIP BETWEEN BRAKE ANGLE AND DECELERATION USING LEAST-SQUARES REGRESSION. 110
FIGURE 7.11: ESTIMATE OF RELATIONSHIP BETWEEN ACCELERATION AND GEAR USING LEAST-SQUARES REGRESSION. 112
FIGURE 7.12: AVERAGE ADDITIONAL DELAY CAUSED BY STOP SIGNS AND TRAFFIC SIGNALS UNDER SPARSE TRAFFIC CONDITIONS IN COMPARISON TO THE SPARSE TRAFFIC OPTIMIZATION ALGORITHM. 115
FIGURE 7.13: NUMBER OF VEHICLES ABLE TO CROSS AN INTERSECTION FROM ONE APPROACH IN ONE GREEN CYCLE WITH RESPECT TO THE LENGTH OF ONE GREEN CYCLE. 118
FIGURE 7.14: VEHICLE FLOW RATE FROM ONE APPROACH WITH RESPECT TO LENGTH OF TIME A SIGNAL HAS SHOWN GREEN. 119

List of Tables

Table 3.1: Control Variables and Associated Notations . . . 21

Table 3.2: State Variables and Associated Notations 23

Table 3.3: Values of Vehicle Model Parameters [1] 30

Table 4.1: Variance of Simulated Noise for Each State Variable . . 49

Table 4.2: Summary of Major EKF Equations in Simplified Forms . . 52

Table 6.1: Eigenvalues of the Jacobian Matrix at Various Equilibrium Points 73

Table 6.2: Boundary of Basin of Attraction for Various Throttle Angles, $b = 0^0$, and $R_g = R_5$. Results are accurate to within +\- 0.5 76

Table 7.1: Numerical Integration Results 98

Table 7.2: Acceleration Estimates with Respect to Gear and Velocity Range 113

Table 7.3: Average Additional Delay Caused By Stop Signs and Traffic Signals Under Sparse Traffic Conditions in Comparison to the Sparse Traffic Optimization Algorithm 115

Acronyms

CCA - cooperative collision avoidance
DSRC - dedicated short-range communications
EKF - extended kalman filter
ITS - intelligent transportation systems
MAS - multi-agent systems
MUTCD - manual on uniform traffic control devices
RK4 - 4th-order Runge-Kutta
WAVE - wireless access in the vehicular environment
WSMP - WAVE short message protocol

Chapter 1 - Introduction

The invention of the automobile has allowed people to travel long distances at an unprecedented rate. Initially, vehicle ownership was limited to the rich. However, once mass production began in the 20th century, automobile ownership skyrocketed. The resulting traffic increase and contention at road junctions inspired the creation of new devices to prevent vehicle collisions. Stop signs, traffic signals, and traffic circles were installed at intersections.

The first section of this chapter begins with a historical background of stop signs, traffic signals, and traffic circles. Next, the limitations of these devices are examined. Afterwards, the contributions of this thesis to the field of collision avoidance are discussed followed by an outline of this thesis.

1.1 Background of Intersection Control Devices

Intersection control devices prevent collisions by assigning the right-of-way to conflicting traffic streams. Stop signs, traffic signals, and roundabouts are just some of the more common intersection control devices in use. The Federal Highway Administration, a division of the US Department of Transportation, recommends the installation of specific control devices primarily based on safety issues and traffic volumes. It periodically publishes these recommendations in the Manual on Uniform Traffic Control Devices (MUTCD) to ensure that the operation of the devices is standardized. In this section, the history of these devices is presented followed by their operating characteristics.

1.1.1 Stop Signs

The first recorded use of the stop sign was in Detroit, Michigan in 1915. Instead of the red and white configuration that is commonly used today, the original was black and white. The next few years saw the expanded use of the stop sign, and in 1922, a rural-dominated committee supported by the American Association of State Highway Officials met to standardize it to an octagonal shape. In 1924, the standard color of the signs was changed to black and yellow. The committee published the Manual and Specifications for the Manufacture, Display, and Erection of US Standard Road Markers and Signs, a precursor to the MUTCD, in 1925. Meanwhile, the National Conference on Street and Highway Safety, another primarily urban standards group, developed a similar manual that standardized stop signs to a different size with red and yellow coloring. In 1932, the two groups joined and published the first MUTCD. The standard stop sign underwent many changes related to its reflectivity, size, color, and mounting height until acquiring its present characteristics in 1971 [17].

The stop sign is well suited for low to medium levels of traffic [14]. It safely allows a minor street to join a road with heavy traffic without interrupting vehicle flow by forcing all vehicles to stop and proceed only when the major road is free of traffic. Four-way stop signs also work well at intersections where the traffic volume from each approach is approximately equal. These intersections generally have traffic volumes that are too low to warrant installing a traffic signal and too high to leave the intersection uncontrolled [13].

1.1.2 Traffic Signals

The first traffic signal was installed in 1868 near the Houses of Parliament in London using semaphore and gas lamps to safely deal with horse-drawn carriage and pedestrian traffic [18]. In 1910, Earnest Sirrine patented the first automatic signal. In 1912, Lester Wire invented an electric traffic light that used red and green lights [19]. James Hoge patented an electric light that was installed in Cleveland, Ohio in 1914 and could be controlled manually or by timer. A few years later around 1917, Charles Reading installed the first coordinated system of traffic signals in Salt Lake City, Utah [20]. By this time, police officers were having much difficulty seeing (and being seen by) the street traffic they were signaling to. In 1917, Detroit installed the first traffic tower, a tower in the middle of the intersection that the officer controlled traffic from. The next major contribution was made by William Potts in 1920. His invention of the first automatic 4-way signal to use red, yellow, and green lights was installed in Detroit, Michigan. As traffic began to increase after World War I, operating the manually controlled signals became problematic as police officers had to be deployed to each intersection. In the coming decade, automatic signals that were actuated by various methods such as car horns and vibrations were installed throughout the US. In 1935, the automatic traffic signal with three lights was standardized in the MUTCD. Today's traffic signals use larger lenses and are increasingly being illuminated by LEDs but have largely remained unchanged [18].

Traffic signals are effective at managing intersections with heavy traffic [13]. They tend to increase safety by reducing the number of severe broadside crashes [26]. The signal timings can be configured in a variety of ways to manage changing traffic conditions. In the past, the lights changed colors at preset intervals but newer traffic signals only change color when a vehicle is detected by sensors buried in the road. Additionally, multiple traffic lights can be coordinated so that vehicles may flow through a series of intersections without having to stop repeatedly at red lights. Traffic signals can also be programmed to detect emergency vehicles and give them priority over other vehicles.

1.1.3 Traffic Circles

A traffic circle is any intersection where the traffic moves in a rotational manner. Some implementations allow two-way traffic but most only allow movement in one direction. Early proposals also referred to the traffic circle as a roundabout, rotary, or gyratory. A modern roundabout is a specific type of traffic circle that requires entering vehicles to yield to those already in the intersection. Although confusing, the term "roundabout" is used interchangeably in the literature to refer to the older traffic circles by some and specifically to modern roundabouts by others. For a full explanation of the differences between traffic circles and modern roundabouts, the reader is referred to [16].

The first intersection that involved one-way rotary operation was invented by Eugene Henard in 1903. Soon afterward in 1905, Columbus Circle, the first traffic circle in America, was installed in New York by William Phelps Eno. The first Parisian traffic circle was built at the Place De l'Etoile around the Arc de Triomphe in 1907. Until the early 1950s, traffic circles continued to be designed throughout Europe and America using poor engineering judgment. Particularly, the right of way was not assigned to any specific traffic stream. This resulted in a large number of accidents during heavy traffic volumes. To solve this problem, the right of way was eventually given to incoming vehicles. Although accidents decreased, traffic circles became so congested that traffic came to a standstill and this method of traffic control soon lost popularity to other forms of intersection control. Many nations abandoned hope for rotaries. However, in 1966, the modern roundabout was introduced in the UK that gave priority to circulating traffic over entering vehicles. This solved the problem of gridlock and created a renewed interest in modern roundabouts throughout Europe in the following years. Since the 1990s, interest in modern roundabouts has increased in the US as well [16].

Unlike other intersection control devices, the current edition of the MUTCD does not specifically recommend the use of modern roundabouts or any traffic circles in general. It merely treats roundabouts as possible alternatives to traffic signals in certain circumstances [14]. Research at the state and federal levels indicates that modern roundabouts are effective at low to medium levels of traffic when the volume of vehicles entering from each approach is relatively equal. Figure 1.1 shows that modern roundabouts cause significantly less average delay than traffic signals when the approach volume of vehicles is at the minimum required for a traffic signal. However, as the percentage of vehicles taking left turns is increased, the average delay also increases. Hence, modern roundabouts are justified under special circumstances.

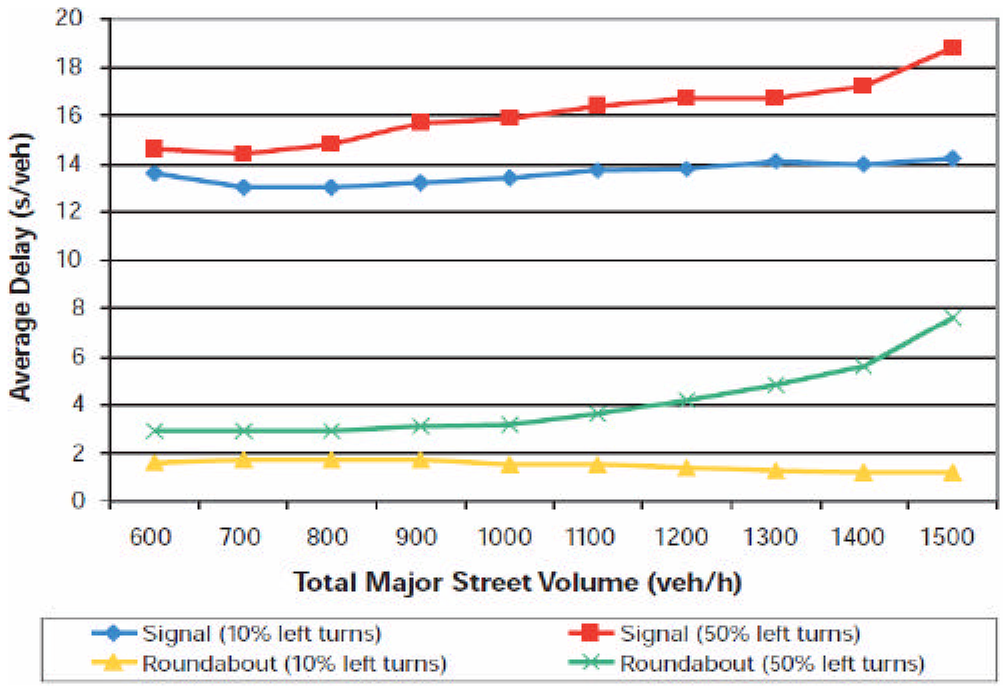


Figure 1.1: Average Delay of Vehicles at Roundabout and Traffic Signal Controlled Intersections as a Function of Approach Volume. The Approach Volume is Comparable to the Minimum Required to Justify Installation of a Traffic Signal. Plot Obtained from [15].

Modern roundabouts are also an ideal control device for intersections with more than four approaches since traffic signals in these configurations usually result in long delays [16]. Many more benefits are realized but a complete discussion is beyond the scope of this work. For more information, the reader is referred to [15] and [16].

1.2 Limitations of Current Intersection Control Devices

The intersection control devices currently in use are heavily dependent on physical infrastructure and are limited in their ability to adapt to changing conditions. These characteristics can lead to problems with safety, efficiency, and cost.

1.2.1 Safety

Today's intersection control devices are unsafe due to their reliance on physical and electrical infrastructure. Stop signs are especially vulnerable to being knocked down by unruly teenagers. This can create a dangerous situation where two vehicles on intersecting roads believe they have the right of way and collide. Although this situation may seem isolated, it has resulted in fatalities on numerous occasions [22]-[24]. In one incident, a stop sign at an intersection was illegally dismantled six times in only 17 days resulting in a fatal, high speed accident [25].

Traffic signals are not as easily tampered with as stop signs but pose their own safety hazards. When configuring the signal timings, special care must be given to the length of the yellow phase. If it is too short, drivers will not be able to cross the intersection before the signal turns red or will have to slam the brakes to avoid crossing the intersection. If it is too long, drivers will ignore it and continue on. The area where drivers are unsure whether to proceed or stop is known as the dilemma zone [13]. This area must be minimized to maintain a safe intersection. While generally increasing traffic flow, the installation of a traffic signal usually increases the number of rear-end collisions at an intersection [21], [26]. The dependence of traffic signals on the power grid also poses a problem. Whenever a power outage occurs, intersections can become dangerous and heavily congested as vehicles struggle to get through.

Since stop signs and traffic lights are dependent on outdoor infrastructure, they are susceptible to interruption from hurricanes and other forms of severe weather. Once the weather passes, returning evacuees may have to deal with nonfunctioning intersections. As time progresses and the impacts of climate change accelerate, instances of severe weather are forecasted to more frequently occur [28]. The dependence of the current intersection control devices on physical infrastructure for proper functionality makes them potential safety hazards. Ideally, a system that is independent of any infrastructure would be used.

1.2.2 Efficiency

The traffic control devices in use today are also highly inefficient since they are only designed to operate well under certain conditions. As mentioned earlier, four way stop signs are designed to operate well when the volume of each approach is approximately equal. However, when this is not the case, the stop sign introduces unnecessary delays in otherwise smooth flowing traffic. Consider a four way intersection with a queue of cars coming in on only one approach. Requiring each vehicle to stop at the intersection before proceeding wastes time and fuel since there are no other cars contending for use of the intersection. Now consider this same intersection under heavy traffic from all approaches. Again, the stop sign's requirement that all vehicles come to a complete stop impedes traffic flow.

Although traffic signals are effective at dealing with heavy traffic, they introduce unnecessary delays in certain cases. When the volume of vehicles on each approach of a four way intersection is approximately equal and is low, the time it takes for the signal to transition introduces delay and causes idling motorists to waste fuel. The Washington State Department of Transportation has found that "signals almost always create overall delay to drivers. In fact, minor side street traffic may experience excessive delay, particularly during off-peak hours" [21].

Although sudden changes in traffic are generally rare and isolated, an imminent natural disaster creates a significant demand on the transportation system. Unlike normal congestion, evacuation traffic is largely in one direction. The inability of the current intersection devices to adequately cope with this traffic means police officers are expected to manage traffic evacuations instead of serving the people in other ways [27]. The inability of the current intersection control devices to adapt to changing traffic conditions thus causes inefficient transport delays. Especially during disasters, it is critical that the delays be reduced as much as possible. Ideally, a system whose behavior adapts to traffic conditions at the time is desired.

1.2.3 Cost

Present day intersection control devices are costly. Stop sign placement incurs costs during purchase and installation. The vegetation surrounding the sign also needs to be maintained to prevent visual obscurity. These expenses are minor compared to traffic signals. The cost of purchasing and installing a single traffic signal ranges from \$250,000-\$500,000. Additionally, annual electricity and maintenance fees amount to about \$8000 for each signal [21]. Ideally, a system that is cheap would be used.

1.3 Contributions

The research conducted in this thesis seeks to improve upon the performance of current intersection control devices - specifically traffic signals and stop signs - by proposing an adaptive, decentralized, CCA system that acts as a traffic signal in dense traffic and as a yield sign in sparse traffic. The system's adaptive nature allows it to operate efficiently under a range of traffic conditions while decentralization allows operation independent of any outdoor, physical infrastructure. Vehicles use low-cost wireless equipment to cooperatively negotiate safe routes as opposed to using costly traffic signals and inefficient stop signs.

Each vehicle is modeled with a nonlinear five-state-variable model that accounts for the vehicle's dynamics and internal engine parameters. The system has each vehicle measure its position using GPS and its other state information using internal sensors. The state information of other vehicles is obtained by receiving data through a wireless communication link. The observations from all vehicles are passed to a set of constrained, coupled Extended Kalman Filters (EKF). The filtered state information is used to determine the optimal trajectory each vehicle should take through the intersection in order to minimize the average delay experienced by all vehicles. The vehicles' speed control then becomes automated to ensure they execute the optimal set of controls until they traverse the intersection. A simplified representation of the interaction between different stages of this system is shown in Figure 1.2. The contributions of this system to the field of collision avoidance are explained.

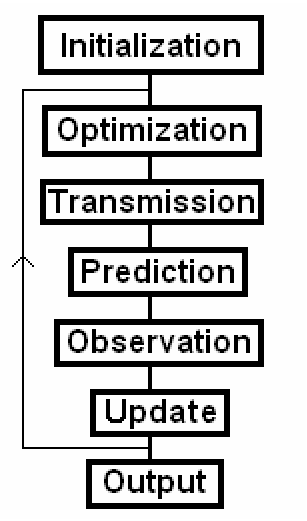


Figure 1.2: Simplified Representation of Interaction Between Different System Stages.

The nonlinear vehicle model expanded upon in this thesis improves on idealized linear tracker models by accounting for vehicle powertrain[1] characteristics. Most linear tracker models used by researchers only account for position, velocity, and acceleration and do not relate acceleration to the vehicle's states and controls. Thus, state dependent frictional forces arising from the engine and environment are unable to be accounted for. In addition, the optimization results stop short of solving the fundamental problem of finding the optimal set of vehicular controls and instead solve for optimal acceleration. The tracker model can be made comparable to the nonlinear vehicle model used here but only if the acceleration is a function of the vehicle's state vector.

The nonlinear vehicle model's five states include the vehicle position coordinates, mass of air in the intake manifold, engine angular velocity, and the brake torque. The state space model is based on data for a 1990 Lincoln Town Car. The vehicle is a typical 6-cylinder, front-wheel drive car with a Ford AOD transmission and a continuous, four-stroke, naturally-aspirated, spark-ignition engine. By accounting for the internal dynamic characteristics, the vehicle model more closely captures the dynamic response of a real vehicle.

[1] The major components of a powertrain are the engine, transmission, and drivetrain [2].

The use of coupled EKFs allows the state information of several vehicles approaching an intersection to be cooperatively estimated. However, good estimates are required to ensure that the optimization's decisions are based on accurate state information. Decentralized and coordinated control systems allow us to meet specific requirements.

The vehicular network provides the basis for CCA. It provides cooperative capabilities by enabling vehicles to communicate their information to other vehicles. This cooperative system is superior to vehicular-based radar and optical technologies since it can function over longer distances and does not require line of sight. At the present time, the vehicular network is unable to support the update frequency required for the CCA system. It is expected that future research will make this possible.

1.4 Thesis Overview

This thesis is organized as follows. Significant research activities in vehicular networks and collision avoidance that have contributed to the advancement of the concept of CCA systems are presented in Chapter 2. In Chapter 3, the nonlinear state space dynamic equations used to model each vehicle are introduced and analyzed. The basic operation of internal combustion engines is also discussed. The EKF is presented in Chapter 4 along with a brief description of some numerical integration techniques. In Chapter 5, issues related to the network of vehicles are discussed. The standards for DSRC, as determined by the US Department of Transportation, are introduced here. The interaction between multiple, cooperating vehicles is also explored through a study of the optimization algorithm. The stability of both the vehicle model and the network of vehicles are analyzed in Chapter 6. In Chapter 7, simulation results are presented. These results show that the CCA system is superior to stop signs and traffic signals in terms of minimizing delay while avoiding collisions. In Chapter 8, conclusions and suggestions for future research are provided.

Recent advances in wireless and vehicular technology have created a unique opportunity to develop CCA systems. There are many organizations throughout Europe, Japan, and the US that are extensively pursuing the deployment of Dedicated Short Range Communications (DSRC) [10], [12], [31], [32], [33], a project that supports vehicle-to-vehicle and vehicle-to-infrastructure communications. The project aims to define a set of standards to add Wireless Access in the Vehicular Environment (WAVE) to the popular IEEE 802.11 standard. DSRC is the foundation for a new generation of services to increase safety and capacity. The US standards are still being completed and new DSRC capable vehicles are expected to reach the US market as early as 2010 [29].

Chapter 2 - Literature Review

Collision avoidance technologies have long been of interest to researchers. In addition to motor vehicles, these technologies have been applied to numerous scenarios involving naval ships [34]-[36], airplanes [37]-[39], and robots [40]-[42]. Recent technological advances have made now an opportune time for research in vehicular CCA systems. In this chapter, past research focusing on optical and cooperative vehicular collision avoidance systems will be presented. Afterwards, current research and state of the art trends will be discussed.

2.1 History

Vehicular collision avoidance systems began to be researched in the mid-80s. Originally, they were designed to support the concept of Automated Highway Systems. By automating highway driving, researchers hoped to increase freeway capacity by decreasing the spacing between adjacent vehicles [43]. The use of laser and radar technologies for collision avoidance is popular for this application [49]-[51]. These form the basis of the Adaptive Cruise Control systems available today.

The European Union's PROMETHEUS project sought to demonstrate the feasibility of Automated Highway Systems by designing and evaluating an autonomous vehicle under normal highway traffic conditions [43]. The system was successfully tested in 1986 with a specially designed van using image processing to avoid other vehicles seen from a video camera. The use of expensive, dedicated hardware meant that it could not be immediately applied to normal road vehicles. However, the success of this project encouraged further research in autonomous vehicles and collision avoidance systems [45].

The PATH program at Berkeley undertook most of this research in the US. During a test in 1997, the feasibility of Automated Highway Systems using common passenger cars was demonstrated. The system's requirement of additional road infrastructure meant that it was not deployable until further research was completed [46], [47].

During the 1990s, policy makers realized that the limitations in wireless capabilities at the time made implementation of cooperative systems difficult [48]. It was also realized that Automated Highway Systems would require significant time and research so they began to encourage research in rapidly-deployable safety devices that performed only some Automated Highway Systems functions [43], [76]. However, key changes were occurring in wireless networks. The introduction of IEEE 802.11 gave consumers the ability to purchase and install low-cost wireless home networks for the first time. As the technology became cheaper in the following years, research into cooperative collision avoidance systems hastened.

2.2 Optical Collision Avoidance Systems

Optical collision avoidance is attracting greater interest from the research community as low-cost computers increase in performance [57], [58]. Optical systems typically use image processing algorithms to identify objects on a road way. The objects are then tracked to estimate their dynamics and warn drivers if a collision may occur. In addition to these functions, many systems provide other benefits as well.

Tsuji et al. [52] propose the use of a stereo infrared vehicle-mounted camera to track the movement of pedestrians at night. The system then provides a voice warning and highlights an infrared image of the pedestrian using a heads-up display on the windshield. Since twice as many pedestrian fatalities occur from accidents at night than during the day, this system looks promising from a safety perspective.

The use of video to maintain highway lane positioning by identifying lane markers was demonstrated in the PROMETHEUS project [53]. In [54], the use of video for lane keeping during vehicle following is applied to the urban environment. Urban roads are often more difficult to navigate due to their dynamic characteristics. Unlike freeways, they can have parked cars, pedestrians, or cyclists sharing the road. Gehrig and Stein use the elastic band approach to adjust for these obstacles. A vehicle's target path is established by observing a leading vehicle. This path is modeled as an elastic band. As the following vehicle travels, any aspects of the environment that no longer allow it to maintain the target path are modeled as forces acting on the elastic band. These forces perturb the band into the form of a new path [55].

Atev et al. [56] depart from conventional, vehicle-mounted cameras and use multiple video cameras installed at an intersection to predict if a collision is imminent. The paper discusses different methods of identifying vehicles from the video samples and considers the use of bounding rectangles and boxes. The results obtained with bounding boxes indicate that the system performs well in real-time with general purpose computing hardware. However, the paper does not attempt to consider the method of communicating a warning to drivers. This indicates a disadvantage of optical equipment that is not vehicle-mounted.

Pathirana et al. [59] attempt to improve upon common identification techniques and propose a robust EKF for tracking objects from camera images. The robust EKF is designed to handle the large uncertainties and measurement noise that are present in automotive systems, particularly in the initial conditions and image position localization of vision systems. Results demonstrate that using a robust EKF yields significantly improved results as compared to the EKF.

Optical collision avoidance technology looks promising for the future but presently possesses several disadvantages that need to be overcome. An inherent disadvantage is the need to have a clear line-of-sight to see an object. Thus vehicle-mounted cameras are unsuitable for more than warning devices for intersection collision avoidance since the whole intersection cannot be seen until it has already been entered. Although some authors have attempted to avoid this problem by installing cameras at the intersection, the inability of the camera to warn drivers makes this approach futile. There also remain image processing issues that need to be resolved. Many systems identify a vehicle by determining its edges. When two vehicles are close, they can appear as one vehicle to a vehicle-mounted camera [57]. More complex methods are needed to accommodate all possible driving situations.

2.3 Cooperative Collision Avoidance (CCA) Systems

Research in vehicle CCA systems has seen much interest since the standardization of DSRC at the 5.9 GHz band [65]-[69]. In cooperative systems, vehicles determine their status using internal sensors, and then base their decisions on road beacons and other vehicles' information obtained through network communication. Cooperative systems are designed differently than optical-based systems and present their own unique characteristics. Research in CCA systems is generally conducted in either a microscopic or macroscopic approach.

2.3.1 Microscopic Approach to CCA Research

The microscopic approach to CCA generally evaluates the interactions between a few vehicles to make conclusions about the use of the system on a large scale. This approach is most common among engineers. Some research following this approach is presented below, illustrating different aspects of CCA.

Li and Wang [60] propose the use of decentralized, cooperative driving to manage intersections that do not have traffic signals. Since heavy vehicles accelerate slowly, this system saves them from having to stop. Furthermore, line of sight is not needed for the system to function since vehicles are located by their wireless transmissions.

Morioka et al. [61] propose a similar collision warning system for use at intersections in mildly urban environments. The system uses Differential GPS and Gyroscope to determine accurate position information. The system requires vehicles to transmit their position, velocity, and other pertinent information when approaching an intersection. If a vehicle coming from another approach detects this signal, it sends its own information in reply. The authors emphasize the ability of the 5.8 GHz band to diffract and penetrate concrete structures common in cities. Results show that using an omni-directional antenna at the 5.8 GHz band is suitable for such intersection collision avoidance applications.

Sung et al. [62] use a cooperative system working in tandem with a sensor network to selectively provide warnings to oncoming drivers of dangerous road situations. The system uses sensors embedded in the road to measure a vehicle's velocity. The vehicle's position and velocity are then transmitted to a nearby station for analysis. If it is determined that the vehicle's speed could imperil the driver further along the road, a warning is transmitted. In mountainous areas, this system could warn drivers of sharp turns and approaching vehicles.

2.3.2 Macroscopic Approach to CCA Research

The macroscopic approach focuses on the performance of the system as a whole. This approach is commonly used by computer scientists and robotics researchers studying Multi-Agent Systems (MAS) [79], [63]. In MAS for networks of vehicles, each vehicle is modeled as an autonomous agent that is given simple instructions to follow in order for the network of vehicles to operate effectively. The field of MAS is generally interested in the protocol each vehicle should follow. A paper applying this approach to managing intersections is described next.

Dresner and Stone [63] propose a reservation-based system for optimizing the flow vehicles through an intersection while avoiding collisions. They model the intersection as a matrix of tiles with an intersection manager that can communicate wirelessly with approaching vehicle agents. Each approaching vehicle communicates its intended path to the manager. If the set of tiles that correspond to the vehicle's intended path are clear during the times it plans to traverse those tiles, then the manager acknowledges and reserves those tiles for the vehicle. Otherwise, the manager rejects the request and forces the vehicle to make another request at a later time. The reservation system is shown to be superior to traffic signals in terms of minimizing the average delay of vehicles.

Unlike optical systems, CCA is an effective way to manage intersection traffic since line of sight is not required. However, the use of CCA in the near future is hindered by the need for more infrastructure and other issues. This includes transceivers for every vehicle and in some cases, roadside transceivers as well. Furthermore, pedestrians and other obstacles are not accounted for in these systems. In order to optimize an intersection using CCA, the system needs accurate information from every vehicle. If any vehicle does not transmit information or transmits wrong information, the system as a whole can fail [64]. The system is heavily influenced by the performance of the network and thus requires a detailed network evaluation before being implemented.

2.4 Trends in CCA Research

The different collision avoidance technologies being researched today are likely to be widely deployed in the future. The technologies appear to be competing but are actually complementary in many ways. Radar and laser technologies will probably be improved in the near future but will be specialized for deployment of Adaptive Cruise Control and measuring the characteristics of adjacent vehicles. The next technology to see deployment will likely be vehicle mounted optical cameras. Initially, these will be used as safety devices to provide warnings to the driver. Since line of sight is needed and is unavailable in many cases, cooperative technologies complement optical systems. Cooperative systems require the most infrastructure and market penetration to operate effectively so they will likely be the last technology to see widespread installation. Additionally, the inability to sense pedestrians and other road obstacles means optical cameras will be necessary to complement cooperative systems. Once CCA is installed, however, the systems developed under MAS will probably be introduced.

The current barriers to the deployment of CCA are not technical but are more legal in nature [43]. Since cooperative driving is such a critical application where human lives are at risk, adequate testing must be continued to ensure system reliability. However, the systems are likely to have a profound impact on traffic and warrant more research.

Chapter 3 - Vehicle Powertrain State Space Model

The major components of a powertrain are the engine, transmission, and drivetrain [2]. Conventional automobiles generate vehicle motion by burning a mixture of fuel and air. The internal combustion engine is responsible for converting the chemical energy of fuel into vehicle kinetic energy. The transmission and drivetrain convert the output of the engine to the appropriate torque and rotational speed before transferring the power to the wheels. This chapter will provide a detailed discussion of how this process occurs.

The first section of this chapter is intended to provide a presentation of the physical operation of automobile powertrains. It is not intended to be a thorough discussion of all internal devices but will provide a sufficient background applicable to this thesis. Subsequent sections will present the differential equations governing the dynamics of the vehicle and the values used for the parameters.

3.1 Principles of Automobile Powertrains

The process of converting chemical energy to kinetic energy begins at the accelerator. It adjusts the throttle valve which directly controls the amount of air being allowed into the engine's intake manifold. The intake manifold is responsible for regulating airflow to the individual engine cylinders. In naturally-aspirated engines, the pressure of the intake manifold never exceeds the ambient atmospheric pressure. The pressure difference creates a stream of air through the manifold to the cylinders.

The engine cylinders are the location where fuel combustion occurs. The cylinders receive air through a valve from the intake manifold. The electronic fuel injection system mixes the fuel with the air prior to combustion. The type of fuel injection determines when the fuel is mixed with air. In sequential electronic fuel injection (also known as multi-port fuel injection), fuel is injected at the exit of the intake manifold as air enters each cylinder for combustion.

The electronic fuel injection system is also tasked with deciding the amount of fuel to inject. Using an oxygen sensor to measure the composition of gases in the exhaust, it adjusts the amount of fuel to achieve the optimal air/fuel ratio (R_{af}) during combustion. In terms of mass, an R_{af} of 14.7:1 results in complete combustion of air and fuel assuming the fuel is pure heptane. Different ratios may be desired for a variety of reasons depending on the operating conditions.

After combustion occurs in the cylinders, the linear-reciprocating motion of the pistons is converted to rotational motion using a crankshaft. The crankshaft then connects to a flywheel. The flywheel acts as a stabilizing mechanism by smoothing the discrete firings of the pistons into a more continuous rotation. The engine angular velocity and output torque at the flywheel are often inappropriate for direct application to the wheels. The conversion to more appropriate values is performed by the transmission.

The transmission is composed of a gear box, along with a torque converter in automatic transmissions. The torque converter provides fluid coupling and increased torque during initial acceleration. It is composed of at least three rotating elements: a pump, a turbine, and a stator. The input of the converter, the pump, is directly connected to the engine and rotates at engine speed. The output of the converter, the turbine, is connected to the wheels through the gear box. The stator connects between the pump and turbine. When there is a substantial difference between the pump and turbine rotational velocities ($\frac{w_{turbine}}{w_{pump}} < 0.9$), the torque converter is in converter mode. In this mode, the torque converter provides torque multiplication to the output by acting as an additional gear. When the pump and turbine angular velocities are nearly equivalent, the torque converter locks and is in fluid coupling mode [1]. The fluid coupling increases the fuel efficiency of the vehicle. Similar to the function of the flywheel, the fluid coupling also smoothes the rotational speed applied from the pistons to the wheels [2].

The gear box of a typical car has four or five gears of fixed size. In an automatic transmission, the vehicle adjusts gears by itself. The gears are sized so that high torque is transferred to the wheels during low velocities and high rotational speed during high velocities. The highest gear is often called the overdrive gear. It significantly increases fuel efficiency at high speeds. When first introduced, the overdrive was a different device housed separately from the rest of the gears but in current vehicles the term is simply used to refer to the highest gear.

3.2 State Space Modeling

The state space vehicle model is based on tabulated data for a 1990 Lincoln Town Car. The model is basically a form of the 3-state variable model derived in [1] with portions from other sources. The vehicle modeled herein is a typical 6-cylinder, front-wheel drive car with a Ford AOD transmission and a continuous, four-stroke, naturally-aspirated, spark-ignition engine. This model expands upon the original model and uses 3 control and 5 state variables.

3.2.1 Control Variables

The model's control variables represent the accelerator, brake, and steering wheel. The variables and their notations are shown in Table 3.1.

Table 3.1: Control Variables and Associated Notations

Control Variable	Notation
Throttle Angle (Accelerator)	α
Brake Angle	β
Steering Direction	θ

The throttle directly controls the air entering the intake manifold. Since the actual relationship is highly nonlinear, an approximate fit based on empirical data is used. The approximate fitted normalized throttle characteristic is defined as

$$n_{tc} = \left(\frac{a}{85} \right)^2 . \qquad \textbf{(1)}$$

The effect of the throttle angle on the mass of air entering the intake manifold is shown in Figure 3.1. It is assumed that the throttle valve adjusts immediately to a given throttle angle. Although there is no time constant associated with the throttle, the engine still does not receive power until the engine's states gradually adjust.

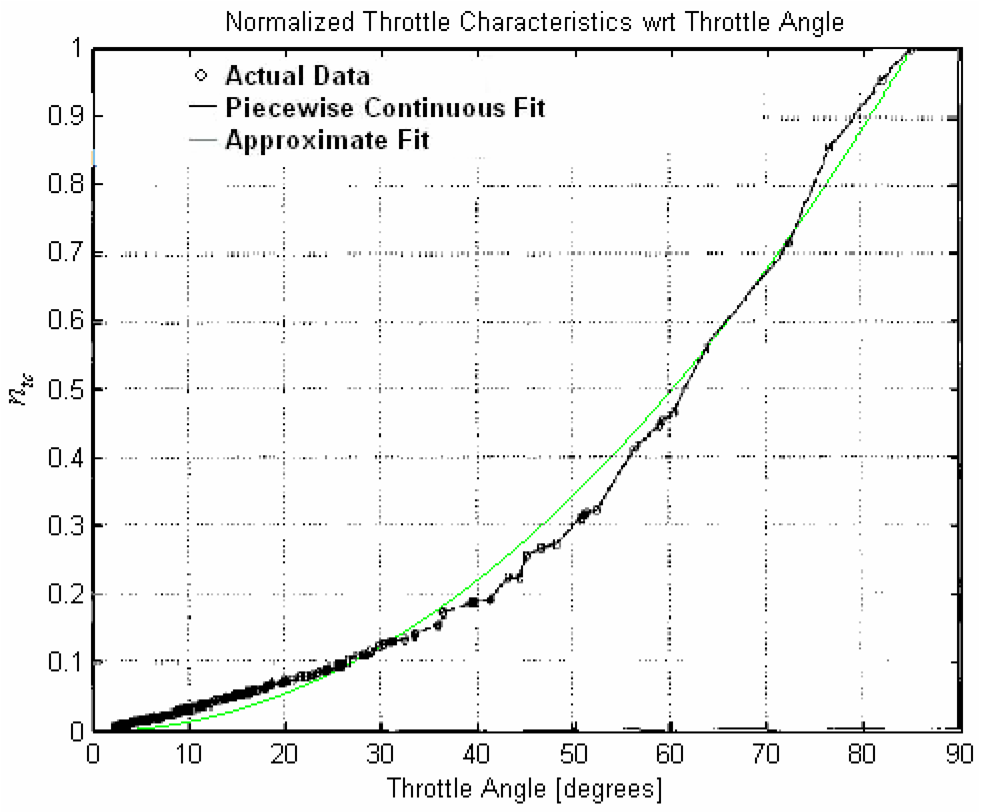


Figure 3.1: Normalized Throttle Characteristics with Respect to Throttle Angle, plot of actual data and piecewise continuous fit copied from [1], plot of approximate fit overlaid.

The brakes control the amount of pressure applied by the brake pads to the wheels. When the brakes are engaged, a torque opposing the motion of the car is created. In conventional vehicles, this causes the vehicle to dissipate energy in the form of heat and decelerate. Through simulations on the state space vehicle model used in this thesis, it was found that the brake angle

$$b \approx 10d\,, \tag{2}$$

where b is in degrees and d is the deceleration rate in m/s^2. In order to maintain agreement with the throttle angle a , b is defined over a range of 0-85^0. It is assumed that the brake pedal adjusts immediately to a given brake angle. Similar to the throttle, the vehicle still does not begin braking instantaneously until the state variables gradually adjust.

The steering direction controls the destination of the vehicle. Like the angles of the unit circle, it is defined as the angle from the positive x_1 axis as shown in Figure 3.2.

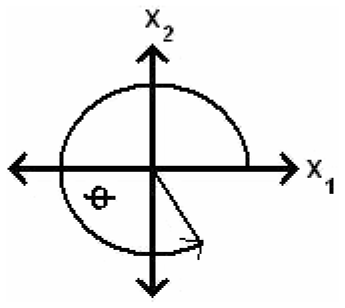


Figure 3.2: Definition of Steering Angle

3.2.2 State Variables and Dynamic Equations for the Vehicle

Vehicle dynamics is defined by five state variables; these include the x_1-coordinate position, x_2-coordinate position, mass of air in the intake manifold, engine angular velocity, and brake torque. The variables along with their notations are shown in Table 3.2.

Table 3.2: State Variables and Associated Notations

State Variable	Notation
x_1 Coordinate Position	x_1
x_2 Coordinate Position	x_2
Mass of Air in Intake Manifold	m_a
Engine Angular Velocity	ω_e
Brake Torque	T_b

Each vehicle determines its position using GPS. When vehicles come within wireless communication range, each periodically shares its states along with other necessary information with the others. Since these measurements are noisy and communication packets may be lost in transit, it is useful to independently track other vehicles' positions by using velocity for comparison with the GPS measurements. The velocity v of a vehicle is given by

$$v = R_g h w_e, \tag{3}$$

where R_g is the speed reduction ratio, a gear-dependent constant, w_e is the engine angular velocity, and h is the effective tire radius. Using this equation, it follows that the differential equations for the x_1 and x_2 positions are given by

$$\dot{x}_1 = R_g h w_e \cos(q), \tag{4}$$

$$\dot{x}_2 = R_g h w_e \sin(q). \tag{5}$$

The differential equation for the next state variable, the mass of air in the intake manifold, is given by

$$\dot{m}_a = \dot{m}_{ai} - \dot{m}_{ao}, \tag{6}$$

where $\dot{m}_{ai}$ and $\dot{m}_{ao}$ are the mass flow rates of air in and out of the intake manifold. The mass flow rate of air into the intake manifold is dependent on the throttle angle and the pressure difference between the atmosphere and the intake manifold. These dependencies are reflected in its equation

$$\dot{m}_{ai} = c_{mfr} n_{tc} n_{pif}, \tag{7}$$

where c_{mfr}, a vehicle-dependent constant, is defined as the maximum flow rate of air corresponding to a fully opened throttle valve. The effect of the throttle angle is represented by the normalized throttle characteristic defined in (1). n_{pif}, the normalized pressure influence function, represents the effect the pressure difference has on the incoming mass flow rate. A plot of n_{pif} for the specific vehicle used in this thesis is shown in Figure 3.3.

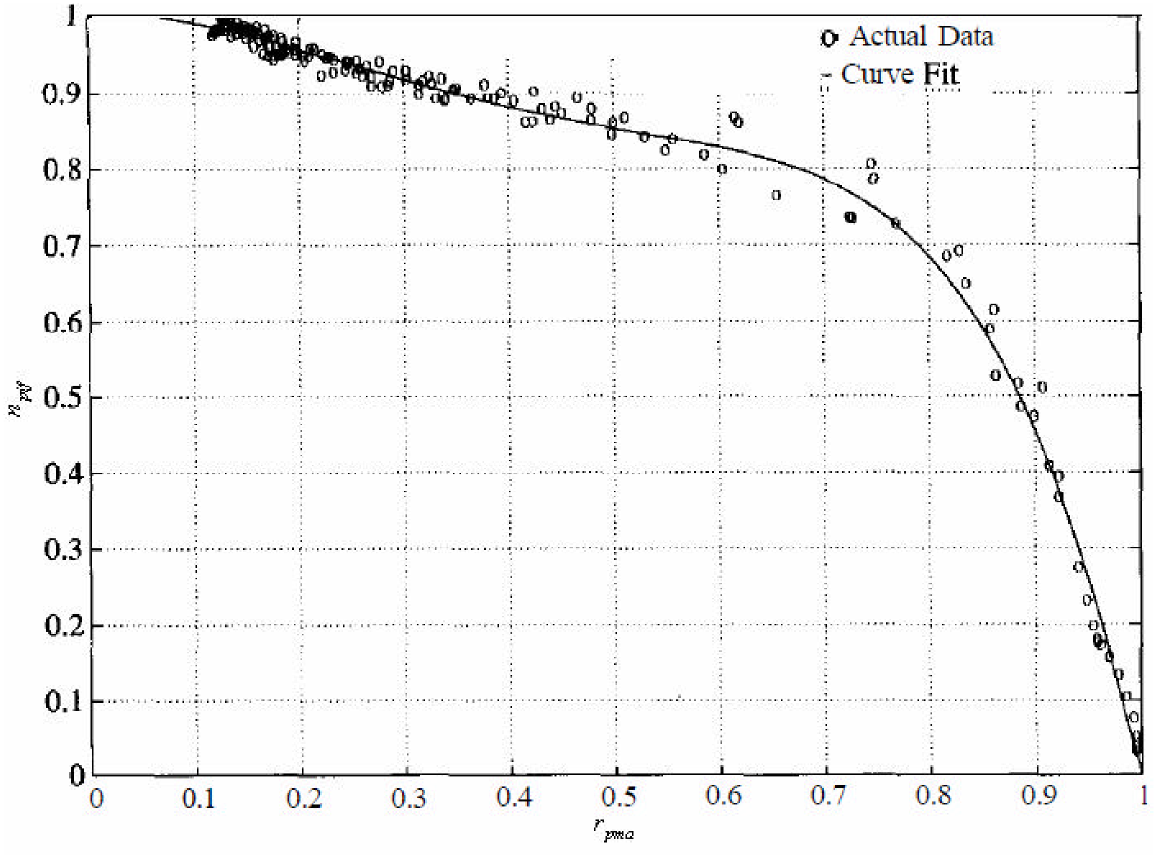


Figure 3.3: Normalized Pressure Influence Function as a Function of the Ratio Between the Pressure in the Intake Manifold and the Atmosphere. Plot obtained from [1].

Due to its nonlinear nature, the curve fit in Figure 3.3 has been used. Its functional form is given by the polynomial

$$n_{pif} = -4.9958r_{pma}^5 + 5.8832r_{pma}^4 - 1.1218r_{pma}^3 - 0.6579r_{pma}^2 - 0.1278r_{pma} + 1.0104 \quad \textbf{(8)}$$

where r_{pma} is defined as the ratio between the pressure in the intake manifold and the atmosphere, and is given by

$$r_{pma} = \frac{P_m}{P_{atm}}. \quad \textbf{(9)}$$

Using the Ideal Gas Law, the manifold pressure is given by

$$P_m = \frac{m_a}{M_{air}} \frac{RT}{V_m}, \tag{10}$$

where R is the ideal gas constant, T is the manifold temperature, M_{air} is the molecular mass of air, and V_m is the volume of the intake manifold.

Now that $\dot{m}_{ai}$ has been analyzed, the differential equations governing the mass flow rate of air exiting the manifold will be studied. Since the air leaving the manifold is directly used for combustion, its mass is dependent on the amount of work the engine is performing. The differential equation for the mass flow rate of air leaving the intake manifold is

$$\dot{m}_{ao} = c_1 h_{vol} m_a w_e, \tag{11}$$

where c_1 is a physical constant and h_{vol} is the volumetric efficiency [1]. c_1 is given by

$$c_1 = \frac{V_e}{4pV_m}, \tag{12}$$

where V_e is the constant engine displacement. The volumetric efficiency describes the engine's ability to draw fresh air into the cylinders for combustion. It is a measure of the percentage of air and fuel that is drawn into the cylinder relative to the maximum amount that can be drawn during static conditions. It is common for fuel-injected vehicles to achieve efficiencies greater than 90%. Due to the difficulty of precisely modeling the volumetric efficiency, an empirical approximation is used instead [2]. The approximation is given by the equation

$$h_{vol} = (24.5w_e - 3.10\times10^4)m_a{}^2 + (-0.167w_e + 222)m_a + (8.10\times10^{-4}w_e + 0.352). \tag{13}$$

The engine angular velocity is the next state variable to be analyzed. It is directly controlled by the brake angle and is indirectly controlled by the throttle angle through m_a. It is also directly proportional to the vehicle speed. Using Newton's Laws, the engine angular velocity is given as the solution of

$$\dot{w}_e = \frac{T_i - T_f - T_p - T_l}{J_e^*}, \qquad \textbf{(14)}$$

where T_i is the engine indicated torque, T_f is the engine friction torque, T_p is the engine pump torque, T_l is the load torque, and J_e^* is the effective engine inertia. The torque variables in (14) represent a variety of forces acting on the vehicle. T_i represents the gross power generated by combustion in the engine cylinders. T_f includes internal friction in the engine and powertrain while T_p accounts for losses in the torque converter. The load torque is dependent on a number of factors including the amount of pressure on the brakes, the rolling resistance of the vehicle, and aerodynamic drag. The effective engine inertia is a gear dependent constant given by

$$J_e^* = J_e + J_{t,g} + R_g^2(J_{wf} + J_{wr} + 4Mh^2), \qquad \textbf{(15)}$$

where J_e is the engine and torque converter inertia, $J_{t,g}$ is the transmission inertia in gear (a gear dependent constant), J_{wf} and J_{wr} are the combined inertias of the front and rear wheels respectively, and M is the mass of the vehicle. The equation yielding the engine indicated torque is

$$T_i = c_t n_{afi}(t - \Delta t_{it}) n_{si}(t - \Delta t_{st}) \frac{\dot{m}_{ao}(t - \Delta t_{it})}{w_e(t - \Delta t_{it})}, \qquad \textbf{(16)}$$

where c_t is the engine torque constant (the maximum torque capacity of an engine for a given air mass and engine speed), n_{afi} is the normalized air fuel influence function, n_{si} is the normalized spark influence function, $\Delta \mathrm{t}_{\mathrm{it}}$ is the intake-to-torque production delay, and $\Delta \mathrm{t}_{\mathrm{st}}$ is the spark-to-torque production delay [1]. Assuming negligible time delays associated with engine power production, Equation 16 simplifies to

$$T_i = c_t n_{afi} n_{si} \frac{\dot{m}_{ao}}{w_e}. \qquad \textbf{(17)}$$

The normalized air fuel influence function is directly dependent on r_{af}, the air/fuel ratio, and is given by

$$n_{afi} = \cos\left(7.3834(r_{af} - 13.5)\right). \quad \textbf{(18)}$$

n_{si} models the effect that ignition spark timing has on the engine torque production. The normalized spark influence function is given by

$$n_{si} = \cos(s_a - c_m)^{2.875}, \quad \textbf{(19)}$$

where s_a is the spark advance/retard from top dead center and the constant c_m is the minimum spark advance for best torque [2]. Now that the engine's torque generation has been studied, the sources of energy loss are analyzed. The engine friction torque has been approximated from empirical data and is given by [2]

$$T_f = 0.1056 w_e + 15.10 . \quad \textbf{(20)}$$

The torque converter's torque in converter mode is given by

$$T_{pump} = c_2 w_p^2 + c_3 w_p w_t + c_4 w_t^2, \quad \textbf{(21)}$$

$$T_{turb} = c_5 w_p^2 + c_6 w_p w_t + c_7 w_t^2 , \quad \textbf{(22)}$$

where c_2 through c_7 are constants. As the converter locks and transitions to fluid coupling mode, the torques become equal and are given by

$$T_{pump} = T_{turb} = c_8 w_p^2 + c_9 w_p w_t + c_{10} w_t^2, \quad \textbf{(23)}$$

with c_8, c_9, and c_{10} as constants. Since the pump rotates at the engine angular velocity and the turbine rotates as at the same rate as the pump in fluid coupling mode, the torque equation simplifies to

$$T_{pump} = T_{turb} = (c_8 + c_9 + c_{10}) w_e^2 . \quad \textbf{(24)}$$

The final torque in the engine angular velocity differential equation, the load torque, is given by

$$T_l = R_g\left(T_b + C_a R_g^2 h^3 w_e^2 + hF_r\right), \qquad \textbf{(25)}$$

where C_a is the aerodynamic drag coefficient and the constant F_r is the combined rolling resistance of the wheels.

The final state variable, the brake torque, is now analyzed. The brake torque is described by the following linear differential equation:

$$\dot{T}_b = \frac{K_b b - T_b}{t_{b,v}}. \qquad \textbf{(26)}$$

The brake torque constant of proportionality, K_b, is directly related to the sensitivity of the brakes to pressure from the brake pedal. $t_{b,v}$, the brake system time constant, models the delay between pressure on the brake pedal and the application of the brake pads to the wheels.

3.2.3 Assumed Model Parameters

The specific values used for the parameters of the state space model are primarily based on [1] with portions related to the torque converter from [2]. The parameter values that have been obtained from [1] are listed in Table 3.3. In the case that both papers neglect to mention the value of a parameter, an approximate value is derived.

The torque converter parameters in (24) are given by $c_8 = -6.7644\times10^{-3}$, $c_9 = 32.0084\times10^{-3}$, and $c_{10} = -25.2441\times10^{-3}$ [2]. For (18) and (19), it is assumed that the r_{af} is constant at the optimal level of 13.5 and s_a is equal to c_m. Although [1] and [2] are modeling different vehicles, it is assumed that the vehicles are similar enough for these values from [2] to suffice for the vehicle in [1] (since both papers use a front-wheel drive car with a 6 cylinder engine).

The temperature of the intake manifold in (10) is assumed to be 50^0 C (333 K). Since the manifold receives fresh air but is located near the combustion chamber, its temperature should be warmer than the atmosphere but much cooler than the combustion chamber. The molecular mass of air is assumed to be 28.97 g/mol [3].

Table 3.3: Values of Vehicle Model Parameters [1].

Parameter	**Value**
h	0.33 m
R_1	0.4167
R_2	0.6817
R_3	1.0000
R_4	1.4993
$R_5 = R_d$	2.3058
c_{mfr}	0.684 kg/s
V_m	0.00447 m^3
V_e	0.0049 m^3
J_e	0.2630 $kg \cdot m^2$
$J_{t,1}$	0.08202 $kg \cdot m^2$
$J_{t,2}$	0.07592 $kg \cdot m^2$
$J_{t,3}$	0.11388 $kg \cdot m^2$
$J_{t,4}$	0.13150 $kg \cdot m^2$
J_{wf}	2.565 $kg \cdot m^2$
J_{wr}	2.565 $kg \cdot m^2$
M	2148 kg
c_t	1,018,686 $N \cdot m \cdot rad / kg$
C_a	0.53384 kg/m
F_r	167.27 N

The vehicle shifts gears at 7, 15, 25, and 45 MPH. During periods of maximum acceleration (a = 85), the gears shift at 11, 20, 35, and 55 MPH. The shift speeds have been defined in two sets to more closely model the true behavior of real vehicles. The choice of shift speeds is made based on the engine angular velocity. One of the main purposes of gearing is to allow the engine to maintain a smaller operating range over all vehicle velocities. Figures 3.4-3.6 show the gear, vehicle velocity, and engine angular velocity when integrating the state space equations with respect to time (more on this topic in section 4.3). The plots are three different outputs for one simulation run. Notice how the range of w_e is relatively small compared to the range of the vehicle velocity. The value of the speed reduction ratio used for fifth gear is different from the value given in [1]. The ratio should increase for higher gears so the engine stays in its operating range. The value of 0.3058 given in [1] is deemed a scanning error and 2.3058 is used instead.

The brake constant of proportionality, K_b, is found to be 300 $N \cdot m$/degree. This value is derived through simulations and is further discussed in section 7.3.

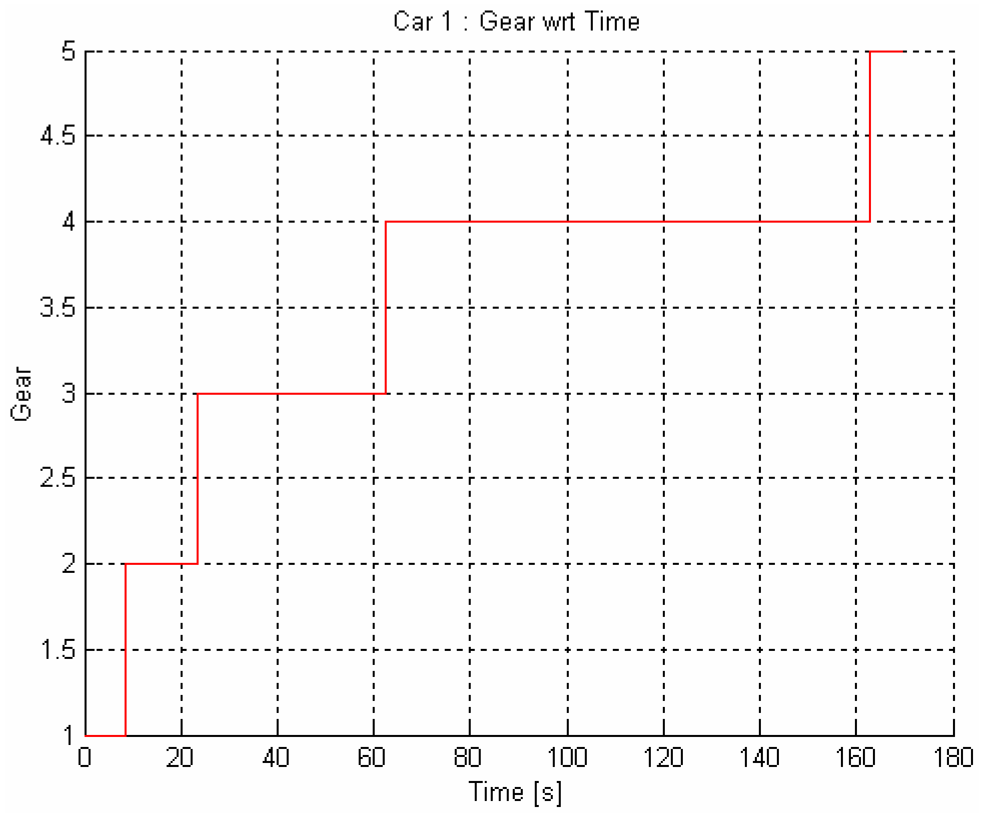


Figure 3.4: Gear with Respect to Time for a = 85^0, and b = 0^0.

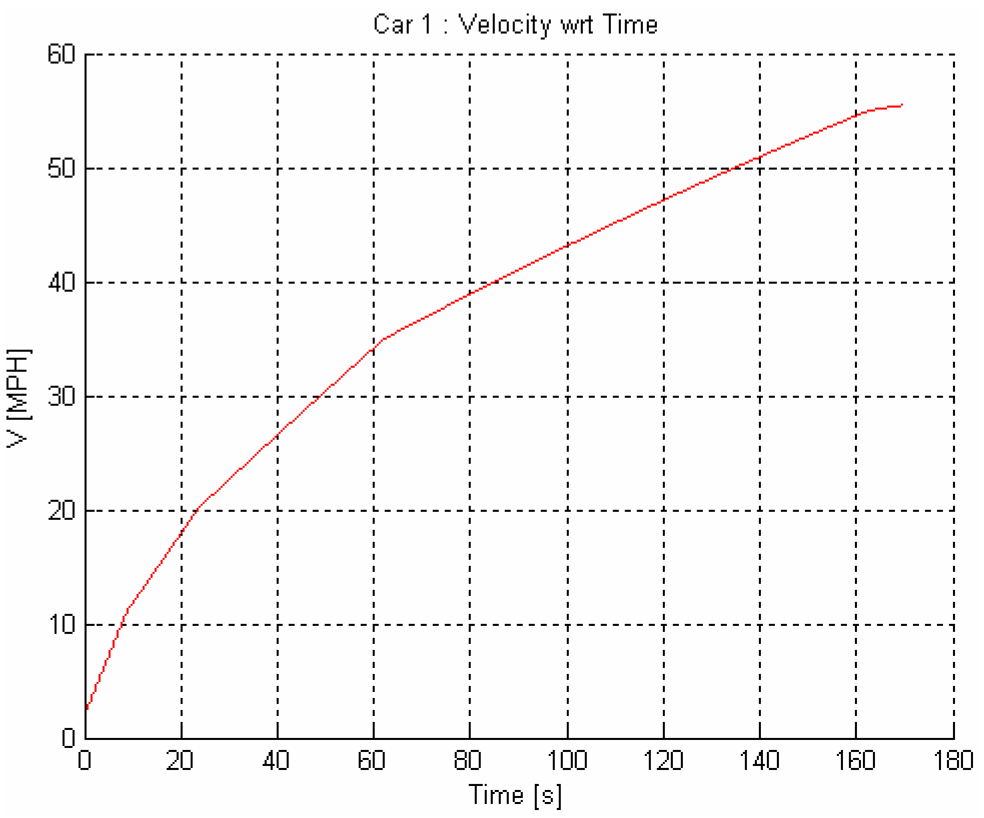


Figure 3.5: Vehicle Velocity with Respect to Time for $a = 85^0$*, and* $b = 0^0$.

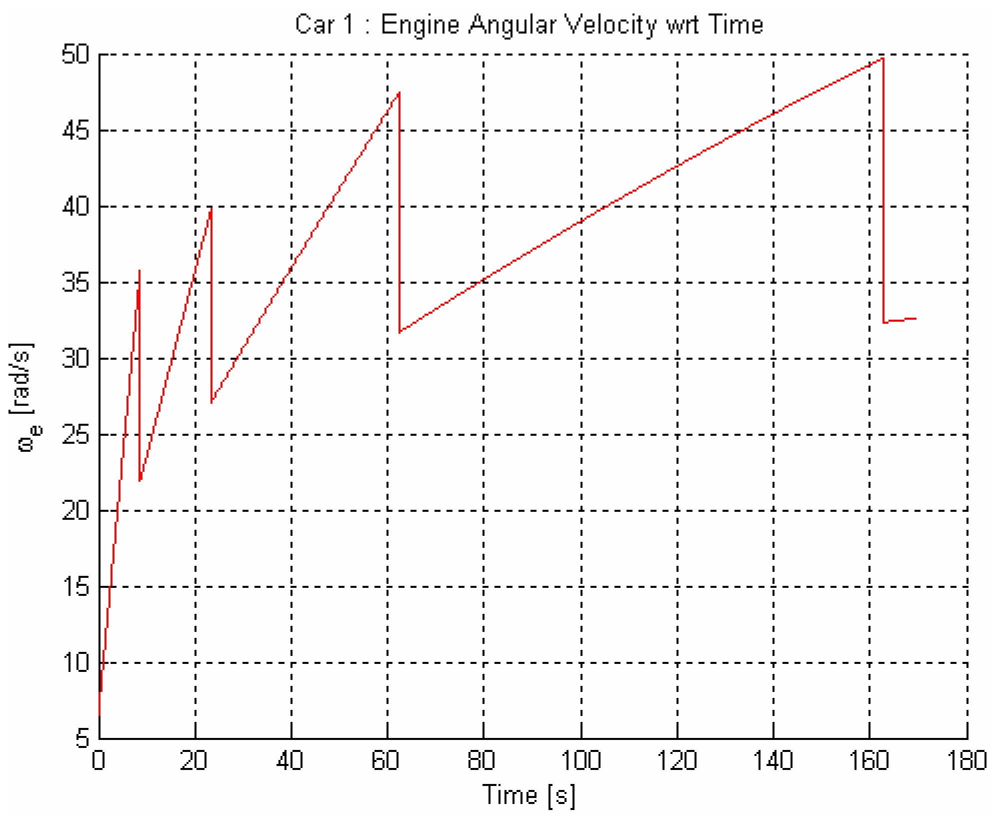


Figure 3.6: Engine Angular Velocity with Respect to Time for $a = 85^0$*, and* $b = 0^0$.

3.3 Summary

The dynamics equations and a background on vehicle powertrain operation have been presented in this chapter. In the next chapter, the vehicle states will be put into vector matrix form and an EKF will be derived that is capable of estimating the states from noisy observations.

Chapter 4 - Linearization, Discretization, and the Extended Kalman Filter for Vehicle Modeling

The Kalman Filter is a mathematical tool that estimates state information from noisy measurements. Although the filter may be formulated in different ways, this thesis will focus on the computationally-efficient recursive form. This chapter will introduce the Kalman Filter and discuss its governing equations. In the next section, the Extended Kalman Filter (EKF) will be presented and its application to the research conducted in this thesis will be explained. The following sections will discuss specific implementation methods.

4.1 Kalman Filter for Linear Model

The classical Kalman Filter [5] is a discrete-time filter used to estimate the states of linear, dynamic systems. At each time step, it predicts the next state of the system using all past observations, receives a new noisy observation, then makes a filtered estimate of the system's state by processing both the new observation and the prediction. The filter models each state transition as

$$\underline{x}_k = \underline{f}_{k,k-1}\underline{x}_{k-1} + \underline{B}_{k-1}\underline{u}_{k-1} + \underline{G}_{k-1}\underline{w}_{k-1}, \tag{27}$$

where $\underline{x}$ is the state vector, $\underline{f}$ is the state transition matrix, $\underline{B}$ is the control-input matrix, $\underline{u}$ is the control vector, $\underline{G}$ is the noise-coloring matrix, and $\underline{w}$ is the process noise. The state transition matrix describes how the system states change when no external inputs are applied. It determines the system's stability. The transition matrix has the following properties:

$$\underline{I} = \underline{f}_{k,k}, \tag{28}$$

$$\underline{x}_k = \underline{f}_{k,k-1}\underline{x}_{k-1}, \tag{29}$$

$$\underline{x}_k = \underline{f}_{k,0}\underline{x}_0, \tag{30}$$

$$\underline{f}_{k,k-2} = \underline{f}_{k,k-1}\underline{f}_{k-1,k-2}, \tag{31}$$

$$\underline{f}_{k-1,k} = \underline{f}_{k,k-1}^{-1}. \tag{32}$$

The control-input matrix controls the effect of the inputs on the states. The process noise is a vector of independent, white, normally-distributed random variables with zero mean and covariance defined by the matrix $\underline{Q}_k$. The process noise models state variations in a system. For example, a rolling vehicle's velocity varies slightly depending on the number of bumps on a road. Usually, the sources of variation are not large enough to warrant a more complex model so they are treated as noise. The Kalman Filter models each observation vector, $\underline{z}_k$, as

$$\underline{z}_k = \underline{H}_k\underline{x}_k + \underline{e}_k \tag{33}$$

where $\underline{H}_k$ is the observation matrix, and $\underline{e}_k$ is the observation noise vector. The observation matrix relates the states to the values of the parameters observed. Although the states are being estimated, the state may not be directly observable so another physical quantity may be observed instead (i.e. using a thermocouple to observe voltage as an indicator of the temperature). The observation noise, like the process noise, is a vector of independent, white, normally-distributed random variables with zero mean and covariance $\underline{R}_k$. The observation noise models variations in sensor measurements. The process and observation noises have the following properties:

$$E[\underline{w}_k] = \underline{0}, \tag{34}$$

$$E[\underline{e}_k] = \underline{0}, \tag{35}$$

$$E[\underline{w}_k\underline{w}_l^T] = 0 \quad \forall \quad k \neq l, \tag{36}$$

$$E[\underline{e}_k\underline{e}_l^T] = 0 \quad \forall \quad k \neq l, \tag{37}$$

$$E[\underline{w}_k\underline{e}_l^T] = 0 \quad \forall \quad k \neq l. \tag{38}$$

Before the Kalman Filter may begin estimating, the noise covariances and states must be initialized. The uncertainty in the initial state $\underline{x}_0$ is reflected in the covariance matrix $\underline{Q}_0$.

The Kalman Filter uses this matrix to decide how to filter its estimate during the update step. The uncertainty of the observations is reflected in the covariance matrix $\underline{R}_k$, which is usually assumed to be constant for all values of k.

4.1.1 Prediction Step

Under the foregoing assumptions, the Kalman Filter predicts the next state of the system by making the best possible estimate at the current time step. Under Gaussianity, the best possible estimate in the maximum-likelihood sense is characterized by minimizing the mean-squared error of the prediction. The optimal prediction satisfies the following property:

$$\underline{\hat{x}}_{k|k-1} = E[\underline{x}_k \mid \underline{z}_1, \underline{z}_2, \ldots \underline{z}_{k-1}]. \qquad \textbf{(39)}$$

The foregoing equation states that the predicted state estimate is the conditional expectation or mean value of $\underline{x}_k$ given all previous observations $\underline{z}_1$ through $\underline{z}_{k-1}$. Substituting the state equation (27) into the prediction equation (39) yields

$$\underline{\hat{x}}_{k|k-1} = \underline{f}_{k-1}\underline{\hat{x}}_{k-1|k-1} + \underline{B}_{k-1}\underline{u}_{k-1} + \underline{G}_{k-1}E[\underline{w}_{k-1}]. \qquad \textbf{(40)}$$

Since the process noise is defined as having zero mean, the final form of the prediction equation is given by

$$\underline{\hat{x}}_{k|k-1} = \underline{f}_{k-1}\underline{\hat{x}}_{k-1|k-1} + \underline{B}_{k-1}\underline{u}_{k-1}. \qquad \textbf{(41)}$$

After making the state prediction, the accuracy of the estimate is determined by means of the covariance matrix of the prediction state error, which is defined as

$$\underline{d}_{k|k-1} = \underline{x}_k - \underline{\hat{x}}_{k|k-1}. \qquad \textbf{(42)}$$

Substituting (41) into (42) and taking the expectation of $\underline{d}_{k|k-1}\underline{d}_{k|k-1}^T$ yields the following expression of the covariance matrix, $\underline{\Sigma}_{k|k-1}$, during the prediction stage:

$$\underline{\Sigma}_{k|k-1} = E[\underline{d}_{k|k-1}\underline{d}_{k|k-1}^T]. \qquad \textbf{(43)}$$

The Kalman Filter recalculates this quantity in the filtering step using observation information.

4.1.2 Filtering\Update Step

The Kalman Filter determines the filtered state estimate by comparing the prediction to the observation and then processing them. When the variances of prediction errors are lower than those of observations, the filtered estimates rely more on the predicted estimates rather than on the observations. On the contrary, heavier weights are given to the observations when the variances of the prediction errors are higher. The expression for the Kalman Filter's filtered state estimate is given by

$$\underline{\hat{x}}_{k|k} = \underline{\hat{x}}_{k|k-1} + \underline{K}_k \left(\underline{z}_k - \underline{H}_k \underline{\hat{x}}_{k|k-1}\right), \qquad \textbf{(44)}$$

where $\underline{K}_k$ is the Kalman gain matrix and the quantity $\left(\underline{z}_k - \underline{H}_k \underline{\hat{x}}_{k|k-1}\right)$ is termed the innovation. The Kalman gain matrix determines the weights that are assigned to the predicted estimate and to the observations when calculating the filtered state estimate. Its expression is given by

$$\underline{K}_k = \underline{\Sigma}_{k|k-1} \underline{H}_k^T \left(\underline{H}_k \underline{\Sigma}_{k|k-1} \underline{H}_k^T + \underline{R}_k\right)^{-1}, \qquad \textbf{(45)}$$

where the term $\left(\underline{H}_k \underline{\Sigma}_{k|k-1} \underline{H}_k^T + \underline{R}_k\right)$ is the covariance matrix of the innovation. The gain is considered optimal in the weighted least-squares sense since it minimizes the following objective function [6]:

$$J(\underline{x}_k) = (\underline{z}_k - \underline{H}_k \underline{x}_k)^T \underline{R}_k^{-1} (\underline{z}_k - \underline{H}_k \underline{x}_k) + (\underline{\hat{x}}_{k|k-1} - \underline{x}_{k-1})^T \underline{\Sigma}_{k|k-1}^{-1} (\underline{\hat{x}}_{k|k-1} - \underline{x}_{k-1}). \qquad \textbf{(46)}$$

Once the filtered estimate is found, the covariance matrix of the updated state estimation error is calculated by

$$\underline{\Sigma}_{k|k} = \left(\underline{I} - \underline{K}_k \underline{H}_k\right) \underline{\Sigma}_{k|k-1}. \qquad \textbf{(47)}$$

For a more thorough discussion of the classical Kalman Filter, the reader is referred to [77].

4.2 The Extended Kalman Filter

The EKF is a generalized form of the classical Kalman Filter used to estimate the dynamics of *nonlinear* systems. The formulation of the EKF presented here is based on [4]. It relies on both a nonlinear and linear model as opposed to the classical Kalman Filter which only uses linear models. The nonlinear model is used directly in the prediction stage. In the filtering stage, linearization and discretization are performed on the nonlinear model around the previous filtered state estimate to obtain a set of discretized, linear perturbation equations. The classical Kalman Filter filtering equations are then used as a basis for the EKF filtering equations.

Let us derive the main expression of the EKF. The state and observation equations are given by

$$\dot{\underline{x}}(t) = \underline{f}\left(\underline{x}(t), \underline{u}(t), t\right) + \underline{G}(t)\underline{w}(t), \qquad \textbf{(48)}$$

$$\underline{z}(t) = \underline{h}\left(\underline{x}(t), \underline{u}(t), t\right) + \underline{e}(t) \quad t = t_i \quad i = 1,2\ldots, \qquad \textbf{(49)}$$

respectively, where $\underline{f}\left(\underline{x}(t),\underline{u}(t),t\right)$ and $\underline{h}\left(\underline{x}(t),\underline{u}(t),t\right)$ are functions of the state vector $\underline{x}(t)$, the control vector $\underline{u}(t)$, and either implicitly or explicitly a function of time, t. It is also assumed that both $\underline{f}$ and $\underline{h}$ are continuous and continuously differentiable with respect to all elements of the state and control vectors. The state and control vectors are assumed to be continuous while the process noise is assumed to be a continuous, white noise process. On the other hand, the observations in (49) are treated as being obtained at discrete intervals. As for the process and observation noise covariance matrices, they are expressed as

$$E[\underline{w}(t)\underline{w}'(t)] = \underline{Q}(t)d(t - t), \qquad \textbf{(50)}$$

$$E[\underline{e}(t_k)\underline{e}'(t_l)] = \underline{R}(t_k)d_{kl}, \qquad \textbf{(51)}$$

where d_{kl} is defined as the Kronecker delta function. It is important to note that the quantity $\underline{Q}(t)d(t-t)$ and $\underline{R}(t_k)$ are the covariance matrices while $\underline{Q}(t)$ can be referred to as an intensity matrix [4]. The process and observation noise vectors also have the same characteristics shown in (34)-(38) for the classical Kalman Filter: they have zero mean, are uncorrelated to themselves in time, and are uncorrelated to each other in time.

4.2.1 Linearization

By applying a first-order Taylor series expansion, the EKF linearizes the nonlinear equations to obtain a set of linear perturbation equations. Formally, we have

$$\underline{f}\left(\underline{x}(t),\underline{u}(t),t\right)=\underline{f}\left(\underline{x}^*(t),\underline{u}^*(t),t\right)+\underline{F}_x\left(\underline{x}^*(t),\underline{u}^*(t),t\right)d\underline{x}(t)+\underline{F}_u\left(\underline{x}^*(t),\underline{u}^*(t),t\right)d\underline{u}(t), \quad (52)$$

where

$$\underline{F}_x\left(\underline{x}^*(t),\underline{u}^*(t),t\right)=\begin{pmatrix}\frac{\partial f_1}{\partial x_1^*} & \cdots & \frac{\partial f_1}{\partial x_n^*}\\ \vdots & \ddots & \vdots\\ \frac{\partial f_n}{\partial x_1^*} & \cdots & \frac{\partial f_n}{\partial x_n^*}\end{pmatrix}, \quad (53)$$

and the term $\underline{F}_u\left(\underline{x}^*(t),\underline{u}^*(t),t\right)$ is defined as

$$\underline{F}_u\left(\underline{x}^*(t),\underline{u}^*(t),t\right)=\begin{pmatrix}\frac{\partial f_1}{\partial u_1^*} & \cdots & \frac{\partial f_1}{\partial u_m^*}\\ \vdots & \ddots & \vdots\\ \frac{\partial f_n}{\partial u_1^*} & \cdots & \frac{\partial f_n}{\partial u_m^*}\end{pmatrix}. \quad (54)$$

Here, n and m are defined as the number of state and control variables, respectively. The partial derivatives used in (53) and (54) are respectively defined as

$$\frac{\partial f_i}{\partial x_j^*}=\left.\frac{\partial f_i\left(\underline{x}(t),\underline{u}(t),t\right)}{\partial x_j(t)}\right|_{\underline{x}(t)=\underline{x}^*(t),\underline{u}(t)=\underline{u}^*(t)}, \quad (55)$$

$$\frac{\partial f_i}{\partial u_j^*}=\left.\frac{\partial f_i\left(\underline{x}(t),\underline{u}(t),t\right)}{\partial u_j(t)}\right|_{\underline{x}(t)=\underline{x}^*(t),\underline{u}(t)=\underline{u}^*(t)}. \quad (56)$$

The vectors $\underline{x}^*(t)$ and $\underline{u}^*(t)$ denote the nominal state and control vectors, respectively. For the first iteration, these values are none other than the initial conditions. For successive iterations, the filtered estimates of the previous time step are assigned as nominal values. The perturbation from the nominal value represented by $d\,\underline{x}(t)$ is defined as

$$d\,\underline{x}(t) = \underline{x}(t) - \underline{x}^*(t), \tag{57}$$

where the nominal state vector satisfies the equation

$$\underline{\dot{x}}^*(t) = \underline{f}\left(\underline{x}^*(t), \underline{u}^*(t), t\right). \tag{58}$$

In the first-order Taylor series approximation given by (52), it is assumed that higher order terms are negligible. This assumption is justified when state and control vector perturbations are small. By substituting the linearized form (52) into the nonlinear state equation (48), it follows the linear perturbation state equation given by

$$d\,\underline{\dot{x}}(t) = \underline{F}_x\left(\underline{x}^*(t), \underline{u}^*(t), t\right)d\,\underline{x}(t) + \underline{F}_u\left(\underline{x}^*(t), \underline{u}^*(t), t\right)d\,\underline{u}(t) + \underline{G}(t)\underline{w}(t)\,. \tag{59}$$

It is necessary for the nominal value to provide a good approximation to the actual value; otherwise, the linear perturbation equations will not accurately model the true behavior of the system. Repeating the linearization on the nonlinear observation equation yields the following linear perturbation observation equation:

$$d\,\underline{z}(t) = \underline{H}_x\left(\underline{x}^*(t), \underline{u}^*(t), t\right)d\,\underline{x}(t) + \underline{H}_u\left(\underline{x}^*(t), \underline{u}^*(t), t\right)d\,\underline{u}(t) + \underline{e}(t)\,. \tag{60}$$

Now, to convert the perturbation state equation to a form that is applicable to the recursive EKF equations, the equation needs to be discretized. This step will be addressed next.

4.2.2 Discretization

The discretization is performed by first finding the solution of the linear perturbation state equation (59). The equation will be solved by first finding the homogenous solution, then the particular solution, then proceeding to find the complete solution. Removal of the inputs and noise yields the following homogeneous differential equation:

$$\frac{d\left[d\underline{x}_h(t)\right]}{dt} = \underline{F}_x\left(\underline{x}^*(t), \underline{u}^*(t), t\right)d\underline{x}_h(t). \quad \textbf{(61)}$$

By utilizing (30), the equation

$$d\underline{x}_h(t) = \underline{f}(t, t_0)d\underline{x}(t_0), \quad \textbf{(62)}$$

can be derived, where $\underline{f}(t, t_0)$ is the state transition matrix of the linear perturbation state equation from time t_0 to time t. By taking the first derivative, we get

$$\frac{d\left[d\underline{x}_h(t)\right]}{dt} = \dot{\underline{f}}(t, t_0)d\underline{x}(t_0). \quad \textbf{(63)}$$

By substituting (61) into (63), we obtain an expanded form expressed as

$$\underline{F}_x\left(\underline{x}^*(t), \underline{u}^*(t), t\right)d\underline{x}_h(t) = \dot{\underline{f}}(t, t_0)d\underline{x}(t_0). \quad \textbf{(64)}$$

Now, by substituting (62) into (64) and by simplifying out common terms on both sides of the equation, the differential equation yielding the homogeneous solution is derived. Formally, we have

$$\dot{\underline{f}}(t) = \underline{F}_x\left(\underline{x}^*(t), \underline{u}^*(t), t\right)\underline{f}(t). \quad \textbf{(65)}$$

To find the particular solution, the method of variation of parameters is used. For brevity, the terms $\underline{F}_x\left(\underline{x}^*(t), \underline{u}^*(t), t\right)$ and $\underline{F}_u\left(\underline{x}^*(t), \underline{u}^*(t), t\right)$ are denoted by $\underline{F}_x(*)$ and $\underline{F}_u(*)$, respectively. Let

$$d\underline{x}_p(t) = \underline{f}(t)d\underline{y}(t), \quad \textbf{(66)}$$

where $\underline{y}(t)$ is an unknown $n \times 1$ vector. Differentiation of (66) yields

$$\frac{d\left[d\underline{x}_p(t)\right]}{dt} = \underline{f}(t)\frac{d\left[d\underline{y}(t)\right]}{dt} + \frac{d\left[\underline{f}(t)\right]}{dt}d\underline{y}(t). \quad \textbf{(67)}$$

By equating (67) with the perturbation state equation given by (59), it follows

$$\underline{f}(t)\frac{d\left[d\,\underline{y}(t)\right]}{dt}+\frac{d\left[\underline{f}(t)\right]}{dt}d\,\underline{y}(t)=\underline{F}_x(*)d\,\underline{x}_p(t)+\underline{F}_u(*)d\,\underline{u}(t)+\underline{G}(t)\underline{w}(t). \quad \textbf{(68)}$$

Substituting (65) and (66) into (68) yields

$$\underline{f}(t)\frac{d\left[d\,\underline{y}(t)\right]}{dt}+\underline{F}_x(*)\underline{f}(t)d\,\underline{y}(t)=\underline{F}_x(*)\underline{f}(t)d\,\underline{y}(t)+\underline{F}_u[*]d\,\underline{u}(t)+\underline{G}(t)\underline{w}(t), \quad \textbf{(69)}$$

where the simplified form is expressed as

$$\underline{f}(t)\frac{d\left[d\,\underline{y}(t)\right]}{dt}=\underline{F}_u(*)d\,\underline{u}(t)+\underline{G}(t)\underline{w}(t). \quad \textbf{(70)}$$

Since $\underline{f}(t)$ is non-singular [4], its inversion followed by an integration yields the expression of $d\,\underline{y}(t)$, which is given by

$$d\,\underline{y}(t)=\int_{t_0}^{t}\underline{f}^{-1}(t)\left[\underline{F}_u\left(\underline{x}^*(t),\underline{u}^*(t),t\right)d\,\underline{u}(t)+\underline{G}(t)\underline{w}(t)\right]dt\ . \quad \textbf{(71)}$$

Substituting (71) into (66) yields the expression of the particular solution, which is given by

$$d\,\underline{x}_p(t)=\underline{f}(t)\int_{t_0}^{t}\underline{f}^{-1}(t)\left[\underline{F}_u\left(\underline{x}^*(t),\underline{u}^*(t),t\right)d\,\underline{u}(t)+\underline{G}(t)\underline{w}(t)\right]dt\ . \quad \textbf{(72)}$$

To find the complete solution, the homogeneous and particular solutions are added to yield

$$d\,\underline{x}(t)=\underline{f}(t,t_0)d\,\underline{x}(t_0)+\int_{t_0}^{t}\underline{f}(t,t)\left[\underline{F}_u\left(\underline{x}^*(t),\underline{u}^*(t),t\right)d\,\underline{u}(t)+\underline{G}(t)\underline{w}(t)\right]dt\,. \quad \textbf{(73)}$$

Now, the solution over one interval is given by

$$d\,\underline{x}(t_{k+1}) = \underline{f}(t_{k+1},t_k)d\,\underline{x}(t_k) + \left(\int_{t_k}^{t_{k+1}} \underline{f}(t_{k+1},t)\underline{F}_u\left(\underline{x}^*(t),\underline{u}^*(t),t\right)dt\right)d\,\underline{u}(t) + \int_{t_k}^{t_{k+1}} \underline{f}(t_{k+1},t)\underline{G}(t)\underline{w}(t)dt\ . \quad \textbf{(74)}$$

Converting (74) to discrete notation and decrementing an iteration to maintain agreement with the form of the EKF equations presented earlier gives the discrete perturbation state equation expressed as

$$d\,\underline{x}_k = \underline{f}_{k,k-1}(*)d\,\underline{x}_{k-1} + \underline{y}_{k,k-1}(*)d\,\underline{u}_{k-1} + \underline{w}_{d;k-1}, \quad \textbf{(75)}$$

where the asterisk indicates a functional dependence on the nominal values. The discretized form of the process noise covariance is given by

$$\underline{Q}_{d;k,k-1}(*) = \int_{t_{k-1}}^{t_k} \underline{f}(t_k,t;*)\underline{G}(t)\underline{Q}(t)\underline{G}^{'}(t)\underline{f}^{'}(t_k,t;*)dt\ . \quad \textbf{(76)}$$

Since it is assumed that the observations are being received periodically, simply changing the notation of (60) yields the discrete perturbation observation equation

$$d\,\underline{z}_k = \underline{H}_{x;k}(*)d\,\underline{x}_k + \underline{H}_{u;k}(*)d\,\underline{u}_k + \underline{e}_k\,. \quad \textbf{(77)}$$

If the expressions $\underline{F}_x\left(\underline{x}^*(t),\underline{u}^*(t),t\right)$, $\underline{F}_u\left(\underline{x}^*(t),\underline{u}^*(t),t\right)$, $\underline{G}(t)$, and $\underline{Q}(t)$ are approximately constant over one time interval, the solutions for $\underline{f}_{k,k-1}(*)$, $\underline{y}_{k,k-1}(*)$, and $\underline{Q}_{d;k,k-1}(*)$ can be greatly simplified. The simplified equations are given by

$$\underline{f}_{k,k-1}(*) = e^{\underline{F}_x[*]T} \approx \underline{I} + \underline{F}_x(*)T\ , \quad \textbf{(78)}$$

$$\underline{y}_{k,k-1}(*) \approx \underline{F}_u(*)T + \underline{F}_x(*)\underline{F}_u(*)\frac{T^2}{2} \approx \underline{F}_u(*)T\ , \quad \textbf{(79)}$$

$$\underline{Q}_{d;k,k-1}(*) \approx \underline{G}_{k-1}\underline{Q}_{k-1|k-1}\underline{G}^{'}_{k-1}T\ , \quad \textbf{(80)}$$

where T is the time of one interval.

4.2.3 Prediction Step

The EKF prediction equations are similar in form to those of the classical Kalman Filter. However, the EKF needs to be linearized at every time step around the filtered estimates of the previous state. This means that the nominal state is equivalent to the last filtered state

$$\underline{x}^*(t) = \underline{\hat{x}}(t \mid t_{k-1}), \tag{81}$$

and as a consequence

$$d\,\underline{x}(t) = 0. \tag{82}$$

Replacing the nominal state with the previous filtered estimate in the definition of the nominal state (58) yields the differential equation

$$\frac{d\left[\underline{\hat{x}}(t \mid t_{k-1})\right]}{dt} = \underline{f}\left(\underline{\hat{x}}(t \mid t_{k-1}), \underline{u}^*(t), t\right). \tag{83}$$

Subsequent integration results in the EKF prediction equation given by

$$\underline{\hat{x}}_{k|k-1} = \underline{\hat{x}}_{k-1|k-1} + \int_{t_{k-1}}^{t_k} \underline{f}\left(\underline{\hat{x}}(t \mid t_{k-1}), \underline{u}^*(t), t\right) dt. \tag{84}$$

$\underline{f}$ may not have an analytical solution. In section 4.3, some popular numerical integration methods are presented to approximate the definite integral. After calculating the predicted estimate, the covariance of the predicted state error can be determined by the equation

$$\underline{\Sigma}_{k|k-1} = \underline{f}_{k|k-1} \underline{\Sigma}_{k-1|k-1} \underline{f}'_{k|k-1} + \underline{Q}_{d;k,k-1}. \tag{85}$$

4.2.4 Filtering\Update Step

The EKF update equations are generally based on the discretized linear perturbation equations whereas the prediction equation was based on the nonlinear model. Using the discrete perturbation equations to derive a form similar to the filtering equation for the classical Kalman Filter (44) yields

$$d\,\underline{\hat{x}}_{k|k} = d\,\underline{\hat{x}}_{k|k-1} + \underline{K}_k(*)\left\{d\,\underline{z}_k - \left[\underline{H}_{x;k}(*)d\,\underline{\hat{x}}_{k|k-1} + \underline{H}_{u;k}(*)d\,\underline{u}_k\right]\right\}. \quad \textbf{(86)}$$

The term $d\,\underline{\hat{x}}_{k|k-1}$ is zero due to linearization around each corrected point. By definition, the perturbation of the update is given by

$$d\,\underline{\hat{x}}_{k|k} = \underline{\hat{x}}_{k|k} - \underline{x}_k^*, \quad \textbf{(87)}$$

where

$$\underline{x}_k^* = \underline{\hat{x}}_{k|k-1}. \quad \textbf{(88)}$$

Applying these changes gives the equation

$$\underline{\hat{x}}_{k|k} = \underline{\hat{x}}_{k|k-1} + \underline{K}_k(*)\left\{d\,\underline{z}_k - \underline{H}_{u;k}(*)d\,\underline{u}_k\right\}. \quad \textbf{(89)}$$

This process is repeated for the perturbation of the observation. By definition,

$$d\,\underline{z}_k = \underline{z}_k - \underline{z}_k^*, \quad \textbf{(90)}$$

where

$$\underline{z}_k^* = \underline{h}\left(\underline{\hat{x}}_{k|k-1}, \underline{u}_k\right). \quad \textbf{(91)}$$

By combining the results of (90) and (91) with (89), the final form of the filtering equation is obtained. Formally, we have

$$\underline{\hat{x}}_{k|k} = \underline{\hat{x}}_{k|k-1} + \underline{K}_k(*)\left\{\underline{z}_k - \underline{h}\left(\underline{\hat{x}}_{k|k-1}, \underline{u}_k\right) - \underline{H}_{u;k}(*)d\,\underline{u}_k\right\}, \quad \textbf{(92)}$$

where the gain is given by

$$\underline{K}_k(*) = \underline{\Sigma}_{k|k-1}\underline{H}'_{x;k}(*)\left(\underline{H}_{x;k}(*)\underline{\Sigma}_{k|k-1}\underline{H}'_{x;k}(*) + \underline{R}_k\right)^{-1}. \quad \textbf{(93)}$$

Using the gain and the observation information, the updated state error can be calculated via

$$\underline{\Sigma}_{k|k}(*) = \left[\underline{I} - \underline{K}_k(*)\underline{H}_{x;k}(*)\right] \cdot \underline{\Sigma}_{k|k-1}(*). \qquad \textbf{(94)}$$

The presentation of the EKF equations is complete. However, using the basic form of the EKF may result in convergence problems since performing only one correction may not maintain a small enough perturbation between the true and nominal states thereby causing the model assumptions for (52) to break down. The recursive form of the Iterated EKF is introduced here for speeding convergence. The EKF update step is iterated by using the result of the previous update as the nominal value of the next update stage. In this process, each iteration involves computation of the following vectors and matrices:

- $\underline{F}_x\left(\underline{x}^*(t), \underline{u}^*(t), t\right)$ and $\underline{F}_u\left(\underline{x}^*(t), \underline{u}^*(t), t\right)$,
- $\underline{H}_x\left(\underline{x}^*(t), \underline{u}^*(t), t\right)$ and $\underline{H}_u\left(\underline{x}^*(t), \underline{u}^*(t), t\right)$,
- $\underline{f}_{k,k-1}(*)$,
- $\underline{\Sigma}_{k|k-1}$,
- $\underline{K}_k(*)$.

The Iterated EKF update stage in recursive format is given by the expression

$$\underline{x}_{k|k}^{(i+1)} = \hat{\underline{x}}_{k|k-1} + \underline{K}_k^{(i)}(*)\left\{\underline{z}_k - \underline{h}\left(\hat{\underline{x}}_{k|k-1}, \underline{u}_k\right) - \underline{H}_{u;k}^{(i)}(*)d\underline{u}_k\right\}. \qquad \textbf{(95)}$$

The iterations cease when the difference between the current update and previous update are within a predefined error. The specific stopping method that we have applied is given by

$$\max\left|\hat{\underline{x}}_{L;k|k} - \hat{\underline{x}}_{L-1;k|k}\right| < e \;, \qquad \textbf{(96)}$$

where e is the error and max is the highest valued element of the absolute difference between the updated state vectors. Other forms of the iterated EKF may opt for the norm of the difference between the two vectors or some other property but the maximum is chosen here to ensure that every state is within the desired error. The filter is designed to produce good results when the system is not highly nonlinear. Otherwise, the EKF can suffer from convergence problems and another form of the Kalman Filter may be preferable. Researchers have shown that the Unscented Kalman Filter often outperforms the EKF for highly nonlinear systems [78].

4.2.5 EKF Implementation for Vehicle Modeling

The EKF formulation used to model the vehicle in this thesis is presented here. Since each vehicle estimates its own states as well as the states of each other vehicle, the EKFs of each vehicle need to be coupled. The coupled EKF state vector is just each vehicle's state vector placed together in series. It is defined as

$$\underline{x} = [x_{11}, x_{21}, m_{a1}, w_{e1}, T_{b1}, \quad x_{12}, x_{22}, m_{a2}, w_{e2}, T_{b2}, \quad \cdots]^T, \tag{97}$$

where the sixth element is the x_1 coordinate of the second vehicle. Other EKF vectors and matrices are similarly redefined for the coupled EKF.

The prediction stage utilizes the RK4 method to implement (84). The predicted state error is calculated using (85). The matrix exponential form of (78) is used to determine the transition matrix. It is assumed that $\underline{u}(t)$, $\underline{F}_x(*)$, $\underline{F}_u(*)$, $\underline{G}(t)$, and $\underline{Q}(t)$ are approximately constant over one time interval and that the period between EKF updates is sufficiently small to use the simplified forms of (79) and (80). The predicted estimate of m_a is constrained to a minimum of 10^{-10} kg. The predicted estimate of w_e is constrained to a minimum of 6.5 rad/s when the brakes are not applied to model the automatic movement of a vehicle when the accelerator is not being used. This value corresponds to a vehicle velocity of approximately 2 MPH. When the brakes are applied, the engine angular velocity is constrained to 10^{-5} rad/s corresponding to a vehicle velocity of approximately 0 MPH. It is not set to zero since one of the state equations is divided by this variable. The estimate of T_b is constrained to a minimum of zero $N \cdot m$ so that the brakes are unable to contribute to forward motion.

When a vehicle receives observations, all states of each vehicle are directly observed. The goal of using the EKF is to obtain accurate, real-time position and velocity information to avoid collisions. Since the vehicle velocity is dependent on the engine angular velocity, and the engine angular velocity is a function of the mass of air in the intake manifold and the brake torque, all states must be observed. Directly observing the states reduces the real-time computational burden since $\underline{H}_x$ does not have to be calculated separately; it is simply equal to the identity matrix.

In the simulations, the process and observation noises for a given state variable are independent and identically distributed according to a normal distribution. The variance of the simulated noise for each state variable is shown in Table 4.1.

Table 4.1: Variance of Simulated Noise for Each State Variable.

State Variable	x_1[m]	x_2 [m]	m_a [kg]	ω_e[rad/s]	T_b [$N \cdot m$]
Noise Variance	10^{-2}	10^{-2}	10^{-10}	10^{-2}	10^{2}

The implemented update stage uses a modified form of (92). Since the states are directly observed and assuming the controls do not change over one period, the perturbation of the control vector reduces to zero simplifying the update equation to

$$\hat{\underline{x}}_{k|k} = \hat{\underline{x}}_{k|k-1} + \underline{K}_k(*)[\underline{z}_k - \hat{\underline{x}}_{k|k-1}]. \quad \textbf{(98)}$$

The constraints applied to the predicted estimates of m_a, w_e, and T_b are applied to the filtered estimates of these variables as well. The other calculations in the update are left unchanged and implemented as presented.

4.3 Numerical Integration Techniques

Numerical integration techniques are necessary in situations where certain nonlinear equations make finding analytical solutions impossible. In this section, the Euler and 4th order Runge-Kutta (RK4) methods are presented. The algorithms described here are based on [8].

4.3.1 Euler Method

The Euler method is the simplest (and most error-prone) numerical integration technique. Beginning with some initial conditions, it uses a first-order Taylor series approximation of the differential equation to find successive points in the solution. Given a differential equation and a set of initial conditions expressed as

$$y' = \frac{dy}{dt} = f(t, y), \tag{99}$$

$$y_0 = y(0), \tag{100}$$

$$y_0' = f\left(0, y(0)\right) = y'(0), \tag{101}$$

the Euler method numerically integrates (66) by using the recursive equation

$$y_{n+1} = y_n + hy_n' + O(h^2), \tag{102}$$

where h is the step size. In essence, the Euler method assumes that the slope over one step is constant and equivalent to the value of the slope at the beginning of the step. Any deviation from this assumption is accounted for as error. The local error, which is the error introduced in a single iteration, is represented by the term $O(h^2)$ in (102). The Euler method introduces an amount of local error on the order of h^2. Over numerous iterations, the accumulated local errors produce a global error on the order of h.

4.3.2 4th-order Runge-Kutta Method

The RK4 method greatly improves on the Euler method. By approximating the differential equation with a higher order Taylor series expansion, the local and global errors can be significantly reduced. Although it uses a more complex approximation of the differential equation than the Euler method, the ability to reduce the number of iterations while maintaining the same level of error yields a higher computational efficiency. The most common form of the RK4 method begins with a differential equation of the type shown in (99) and initial conditions as in (100). It performs numerical integration using the recursive equation

$$y_{n+1} = y_n + \frac{h}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right) + O(h^5)\,, \tag{103}$$

where

$$k_1 = f(t_n, y_n)\,, \tag{104}$$

$$k_2 = f(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_1)\,, \tag{105}$$

$$k_3 = f(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_2)\,, \tag{106}$$

$$k_4 = f(t_n + h, y_n + hk_3)\,. \tag{107}$$

By calculating the slope at multiple points in the interval h using the k values, the RK4 method produces a better approximation than the Euler method.

4.4 Summary

The Kalman Filter has been presented in this chapter. The formulation of the classical Kalman Filter is used as a basis for deriving the equations for the EKF. The major EKF equations in their simplified form are summarized in Table 4.2.

Table 4.2: Summary of Major EKF Equations in Simplified Forms.

Nonlinear Model	$\dot{\underline{x}}(t) = \underline{f}(\underline{x}(t), \underline{u}(t), t) + \underline{G}(t)\underline{w}(t)$ $\underline{z}_i(t) = \underline{h}(\underline{x}(t), \underline{u}(t), t) + \underline{e}_i(t)$
Discretized Linear Perturbation Model	$\delta \underline{x}_k = \underline{\phi}_{k,k-1}(*)\delta \underline{x}_{k-1} + \underline{w}_{d;k-1}$ $\delta \underline{z}_k = \underline{H}_{x;k}(*)\delta \underline{x}_k + \underline{e}_k$
Prediction	$\hat{\underline{x}}_{k\mid k-1} = \hat{\underline{x}}_{k-1\mid k-1} + \int_{t_{k-1}}^{t_k} \underline{f}(\hat{\underline{x}}(t \mid t_{k-1}), \underline{u}^*(t), t)dt$
Jacobians	$\underline{F}_x(\underline{x}^*(t), \underline{u}^*(t), t) = \frac{d\underline{f}}{d\underline{x}} = \begin{pmatrix} \partial f_1/\partial x_1^* & \cdots & \partial f_1/\partial x_n^* \\ \vdots & \ddots & \vdots \\ \partial f_n/\partial x_1^* & \cdots & \partial f_n/\partial x_n^* \end{pmatrix}$ $\underline{F}_u(\underline{x}^*(t), \underline{u}^*(t), t) = \frac{d\underline{f}}{d\underline{u}} = \begin{pmatrix} \partial f_1/\partial u_1^* & \cdots & \partial f_1/\partial u_m^* \\ \vdots & \ddots & \vdots \\ \partial f_n/\partial u_1^* & \cdots & \partial f_n/\partial u_m^* \end{pmatrix}$ $\underline{H}_x(\underline{x}^*(t), \underline{u}^*(t), t) = \frac{d\underline{h}}{d\underline{x}} = \begin{pmatrix} \partial h_1/\partial x_1^* & \cdots & \partial h_1/\partial x_n^* \\ \vdots & \ddots & \vdots \\ \partial h_n/\partial x_1^* & \cdots & \partial h_n/\partial x_n^* \end{pmatrix}$ $\underline{H}_u(\underline{x}^*(t), \underline{u}^*(t), t) = \frac{d\underline{h}}{d\underline{u}} = \begin{pmatrix} \partial h_1/\partial u_1^* & \cdots & \partial h_1/\partial u_m^* \\ \vdots & \ddots & \vdots \\ \partial h_n/\partial u_1^* & \cdots & \partial h_n/\partial u_m^* \end{pmatrix}$
Transition Matrix	$\underline{\phi}_{k,k-1}(*) = e^{\underline{F}_x[\underline{x}^*(t),\underline{u}^*(t),t]T} \approx \underline{I} + \underline{F}_x(\underline{x}^*(t), \underline{u}^*(t), t)T$
Process Noise Covariance	$\underline{Q}_{d;k,k-1}(*) \approx \underline{G}_{k-1}\underline{Q}_{k-1\mid k-1}\underline{G}'_{k-1}T$
Prediction State Error Covariance	$\underline{\Sigma}_{k\mid k-1} = \underline{\phi}_{k,k-1}\underline{\Sigma}_{k-1\mid k-1}\underline{\phi}'_{k,k-1} + \underline{Q}_{d;k,k-1}$
Kalman Gain	$\underline{K}_k(*) = \underline{\Sigma}_{k\mid k-1}\underline{H}'_{x;k}(*)(\underline{H}_{x;k}(*)\underline{\Sigma}_{k\mid k-1}\underline{H}'_{x;k}(*) + \underline{R}_k)^{-1}$
Updated State Error	$\underline{\Sigma}_{k\mid k}(*) = [\underline{I} - \underline{K}_k(*)\underline{H}_{x;k}(*)] \cdot \underline{\Sigma}_{k\mid k-1}(*)$

Chapter 5 - Developing a Network for Vehicular Control at Intersections

This chapter will discuss how multiple vehicles interact and communicate at an intersection using DSRC at the 5.9 GHz band, a project that allows communication between vehicles for Intelligent Transportation Systems (ITS) applications [11]. DSRC is based on several standards that add Wireless Access in the Vehicular Environment (WAVE) to the popular IEEE 802.11 standard [10]. This chapter will begin by presenting the WAVE standards. The purpose is to show how vehicles establish the network that is used to transmit information for the optimization algorithm. Next, the operation of the optimization algorithm will be explained in both sparse and dense traffic. WAVE is currently unable to support the update frequency required for the collision avoidance system but it is expected that this will be possible with future improvements.

5.1 Wireless Access in the Vehicular Environment (WAVE)

WAVE has special characteristics that make it well suited for vehicular environments. It has been allocated 75 MHz of frequency between 5.850-5.925 GHz. This allows data rates ranging from 6-27 Mbps at distances up to 1 km when vehicles communicate with road side units or other vehicles. WAVE supports vehicle speeds up to 100 MPH and a low latency of 50 ms [70].

WAVE is defined by the IEEE 1609 family of standards and uses the IEEE 802.11p amendment to extend the use of 802.11 to vehicles. The IEEE 1609 family is composed of four standards describing the resource manager, security services, networking services, and multi-channel operations. The trial-use version of these standards will be further described in this section.

5.1.1 IEEE 1609.1 - Resource Manager

The IEEE 1609.1 standard defines the architecture and data flows of WAVE. It also describes command messages and data formats. The standard explains how data communication between road side units and vehicle on board units occurs [71]. The discussion of this standard's operation will be based on [72].

The standard defines applications residing on the on board unit as Resource Command Processors and those residing in road side units or elsewhere as Resource Manager Applications. The Resource Manager is the focus of this standard and is also the application that is responsible for managing communication between multiple Resource Manager Applications and Resource Command Processors. Figure 5.1 illustrates the communication scheme. It depicts the location of where different applications reside in a typical instance of vehicle-to-infrastructure communication.

WAVE communication imitates a client-server architecture that is managed by the Resource Manager. For example, in the case where a company wants to provide traffic updates by analyzing vehicle speed statistics in a stretch of highway, the application that analyzes the traffic data (a Resource Manager Application) would reside on the road side unit or a remote server that is connected to a road side unit. When the Resource Manager Application sends a request for the speed of the vehicle in Figure 5.1, the Resource Manager application in the road side unit receives the request then forwards it to the vehicle's Resource Command Processor application using WAVE. The vehicle then replies to the Resource Manager which forwards the message to the Resource Manager Application. If another passing vehicle asks for traffic updates by sending a request to the road side unit, the roles of client and server from the previous case are switched and the Resource Manager would reside in the vehicle to manage its requests to the traffic server.

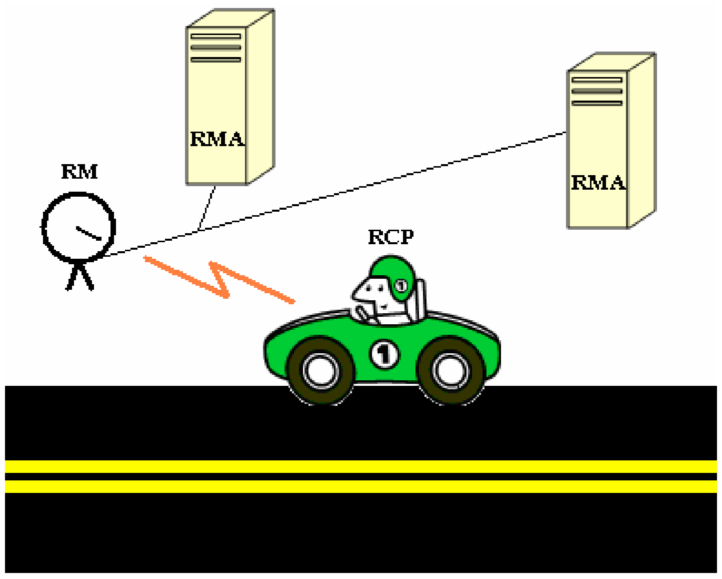


Figure 5.1: Locations of WAVE Applications During a Typical Instant of Vehicle-to-Infrastructure Communication.

WAVE is designed to provide secure communications and minimize the cost of on board units by minimizing the amount of processing required by them. When an Resource Manager sends a request to the Resource Command Processor, it is simply a set of commands that tell the Resource Command Processor to get information from a specific interface or memory location. Each Resource Manager Application is assigned to have access to only certain memory locations for security. This system differs from most general purpose computers where incoming network data is analyzed by the processor and then handled accordingly. In WAVE, the reduced burden on the on board unit reduces cost and increases security.

5.1.2 IEEE 1609.2 - Security Services

The IEEE 1609.2 standard defines secure processing and message formats. Security is important in WAVE because vehicles transmit sensitive information that could constitute a violation of privacy if accessed by unauthorized parties. WAVE maintains security by ensuring confidentiality and authenticity in message transmissions. The final standard is expected to address privacy issues with the current version. The discussion of the security services offered by WAVE will be based on [73].

WAVE ensures confidentiality through encryption. WAVE systems use both symmetric and asymmetric encryption methods. In symmetric encryption, the same key is employed to both encrypt and decrypt the message. In asymmetric encryption (also known as public key cryptography), a widely distributed public key is usually used to encrypt messages while a private key that is kept secret by the receiver is used to decrypt messages. A good private key is defined such that it is computationally impractical to be determined from the public key. Although the latter is usually used for encryption and the private key for decryption, the two keys can alternate roles. Since asymmetric encryption often requires more computation than symmetric encryption, WAVE uses asymmetric encryption to establish communications and then utilize Advanced Encryption Standard, a symmetric encryption standard, for actual communications.

WAVE ensures accurate identification by authenticating transmitting units before opening an encrypted communication session. When a vehicle initiates a transmission to another vehicle, it applies its private key to encrypt a digital signature to identify itself in the message. The receiving vehicle can then use that vehicle's public key for decryption to verify the signature. In order to prevent another vehicle from acting like the sender, the public key is certified with a trusted certificate authority.

WAVE communications present some privacy issues that must be solved before widespread implementation. When a vehicle sends a broadcast to other vehicles, the vehicle should somehow be authenticated but not have its identity leaked to other vehicles. The current standard does not address this issue. The standard also recommends the utilization of permanent MAC addresses for each vehicle. This potentially allows a vehicle to be tracked by observing the location of its transmissions over time. While there are ongoing discussions related to periodically assigning new addresses to solve this issue, we should recognize that WAVE deployment is still an area of concern for privacy advocates.

5.1.3 IEEE 1609.3 - Networking Services

The IEEE 1609.3 standard discusses transport and network layer services such as addressing, routing, and WAVE data exchange [71]. It also discusses the behavior of the system when dealing with messages of different application priorities (not channel priorities). The different communication channels in WAVE are also introduced although this topic is covered in greater depth in the IEEE 1609.4 standard. The presentation of this standard is based on the trial-use standards in [74].

Every WAVE device is given a 48-bit MAC address for each interface and at least one globally unique IPv6 address. Since on board units only have a single wireless interface, they have one MAC address. Road side units typically require a MAC address for each of their wired and wireless interfaces. IPv6 addressing allows WAVE to handle a large network without worrying about address exhaustion. IPv6 is designated as the future replacement for today's internet addressing scheme. It supports 2^{128} unique addresses and is not expected to be completely used in the foreseeable future. WAVE designers have also allowed the use of other types of network addresses through the use of the Wave Short Message Protocol (WSMP).

WAVE defines two types of channels: control and service channels. The control channel is designed to have high speed and low latency to support high priority, safety applications. This channel is also used to establish communications for uncritical applications before switching to a service channel. The service channels are designed to handle general purpose communications. Unlike control channels that only support WSMP, the service channels support both WSMP and IPv6 for transmissions.

WSMP is optimized to operate over the WAVE network. It allows units to directly control the channel and power used during transmission. During heavy traffic, reducing transmission power can lower the amount of interference experienced by other vehicles allowing WAVE to handle large vehicular networks. WSMP also allows non-IPv6 traffic to be encapsulated in messages.

The ability of WSMP to communicate from one on board unit to another is of particular use for decentralized CCA. Using WAVE Short Messages, vehicles can transmit observation information for other vehicles to employ in their EKFs. The standard demonstrates that WAVE is an adequate means of transmitting information to support this CCA system.

5.1.4 IEEE 1609.4 - Multi-channel Operations

IEEE 1609.4 is focused on lower layer networking functions. It describes in detail how WAVE deals with control and service channels. Additionally, it describes aspects of the IEEE 802.11p amendment. The standard also discusses the operational details of some of the WAVE features that have already been presented. These details are beyond the scope of this thesis, interested readers should refer to [75] for more details.

5.2 Cooperative Vehicle Optimization Algorithm

The cooperative vehicle optimization algorithm utilizes each vehicle's model parameters, acceleration capabilities, intended direction, and the EKF's filtered vehicle state estimates at a given time to determine the optimal set of controls for each vehicle. Since the optimization determines the controls for each vehicle from the current time until it travels through the intersection, it need only be executed once each time a new vehicle is observed. The vehicle controls are optimized in one of two methods depending on the number of vehicles. If the number of vehicles is less than 6, the sparse algorithm is used, otherwise, the dense algorithm is executed. This threshold has been chosen to balance the performance of the optimization with the capacity of the wireless network. The amount of network traffic generated is directly related to the number of vehicles included in the optimization. Since the network cannot support an infinite number of transmissions, a safe threshold of 6 vehicles is chosen.

Upon execution of the optimization, each vehicle compares its optimization results to the results obtained by other vehicles. Since all vehicles have the same information, each vehicle’s optimization results are identical. Afterwards, the controls of each vehicle become automated. Automation is necessary to ensure compliance with optimization results. This section will present the optimization algorithm’s behavior along with the conditions that determine which approach is used.

5.2.1 Sparse Traffic

The sparse traffic algorithm determines the optimal set of controls for each vehicle by minimizing the average delay of the vehicles approaching the intersection. The delay is defined as the amount of additional time required for a vehicle to traverse the intersection due to the priority of other vehicles being yielded the right-of-way and is given by the objective function

$$\arg\min t = \frac{1}{n}\sum_{k=1}^{n}\left(t_{o,k} - t_{a,k}\right), \tag{108}$$

where t is defined as delay, n is the number of vehicles, $t_{o,k}$ is the amount of time taken for the k-th vehicle to cross the intersection according to the result of the optimization algorithm, and $t_{a,k}$ is the amount of time taken for the k-th vehicle to cross the intersection if no other vehicles are present. The process used by the algorithm to calculate these times is discussed in the remainder of this section. The optimization models each vehicle as a circle with a known, fixed radius that is dependent on the vehicle size. The minimization is subject to the constraint

$$(x_{1,j} - x_{1,i})^2 + (x_{2,j} - x_{2,i})^2 > (R_i + R_j + m_s)^2, \tag{109}$$

where $x_{1,j}$ is defined as the x_1 position of the center of the j-th vehicle, R_i is the radius of the i-th vehicle, and m_s is the system safety margin. This constraint forces the optimization algorithm to maintain a safe distance between each pair of vehicles. The optimization algorithm’s view of a scenario with two vehicles is illustrated in Figure 5.2.

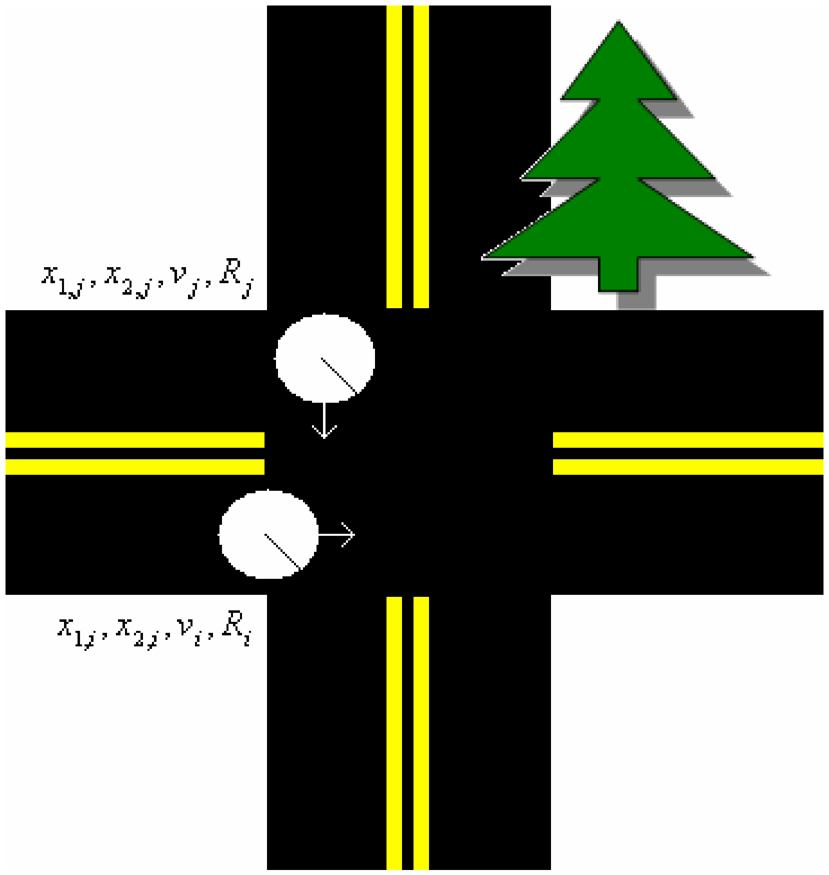


Figure 5.2: Illustration of Optimization Variables.

The operation of the Sparse Traffic Optimization Algorithm generally emulates that of a yield sign. Whenever two vehicles approach an intersection, the vehicle that is able to traverse the intersection first if the other vehicle is not present is given right of way. The other vehicle must cross the intersection after the first vehicle. The operation of the algorithm works similarly for multiple vehicles by forcing lower priority vehicles to delay in order to allow preceding vehicles to cross.

The intersection is assumed to be square with four single lane approaches situated in a rural area. The time to cross is calculated by predicting the vehicle's future trajectory based on its current states and its intended direction. Prediction involves integrating the vehicles' state equations. However, since the state equations do not have an analytical solution and are codependent, they are difficult to integrate over a long period of time. Instead, a piecewise constant approximation for the vehicle acceleration is derived through simulations. The purpose of the safety margin in (109) is to account for differences between the approximated and real vehicle acceleration. The approximation is discussed in Section 7.6.

All vehicles approaching the intersection perform certain common actions. If a vehicle is going straight through the intersection, it will accelerate to the maximum allowed speed on that approach, and then continue at that speed. If turning, it will first accelerate, and then slow down as it nears the intersection so it does not turn at an excessively high speed. The vehicle will turn gradually as it enters the intersection and accelerate once again when exiting. This cycle of accelerating, braking, cruising on the turn, then accelerating again is accounted for when predicting the time it takes for turning vehicles to cross the intersection.

All vehicles are optimized with a throttle angle of 85^0 during periods of acceleration since the acceleration capabilities of the vehicle model are low in comparison to its deceleration. During the braking portion before the turn, a deceleration of 4.00 m/s^2 is used. This deceleration rate is chosen because it approximates real vehicle behavior. Garber states that a value of 11.2 ft/sec^2 is used as a typical deceleration rate for vehicles [13]. This value is equivalent to 3.45 m/s^2 which is close to the value used here. Since the relationship between the deceleration and brake angle is slightly nonlinear, a linear approximation has been used that yields a brake angle of 37.6696^0. The approximation is further explored in Section 7.5. The vehicle's speed is kept constant on turns and its steering angle changes uniformly to allow a smooth turn. Once the vehicle completes a turn and begins accelerating, it finishes crossing the intersection and elapsed time is recorded.

Once the predicted time for each vehicle has been calculated, modifications are made to this time in some cases when multiple vehicles are coming from one approach or if a vehicle is already within a certain distance from the intersection when the optimization begins. If two vehicles coming from one approach are positioned near each other and the following vehicle has a sufficiently high speed, the following vehicle can have a lower predicted time to cross the intersection than the leading vehicle. In this case, the algorithm modifies the following vehicles time to cross to one millisecond higher than the leading vehicle's. Whenever a vehicle has begun braking in preparation for a turn, the vehicle's controls will no longer be changed if the optimization algorithm is run again. Since the range of WAVE is sufficient to find vehicles 1 km away, it is assumed that any vehicles that could possibly collide with a vehicle this near have already been accounted for. After making modifications, the vehicles' predicted times to cross the intersection are arranged in ascending order. If two vehicles have identical times, the vehicle with the lower valued globally unique MAC address will be given priority. Next, this order is used to decide which vehicles' controls are optimized first.

The optimization determines each vehicle's controls for every EKF iteration after the present time until it leaves the intersection. For vehicles going straight through the intersection and initially for those turning, the acceleration phase is optimized by accelerating with a throttle angle of 85^0 until reaching the speed limit, then adjusting the throttle angle to maintain that speed. Once turning vehicles arrive at the braking area preceding a turn, the throttle is released and the brake angle is set to 37.6696 degrees. The algorithm models the time constant of the brake when it is pressed and released. When the vehicle begins to turn, the throttle angle is set to a constant value to maintain the desired speed on the turn, the brakes are released, and the steering angle begins to change at a constant rate. When the vehicle completes the turn, the elapsed time is recorded as the optimal time for this vehicle since other vehicles did not cause delay. The throttle is set to 85^0 and the steering angle is set to the direction the vehicle is now traveling in.

Lower priority vehicles repeat the above procedure but perform a safety check at the end. The vehicle checks to see if its optimal path violates the minimum distance constraint between itself and all higher priority vehicles. If the optimization is within the constraints and a collision does not occur, the set of controls are kept and the next vehicle is optimized. Otherwise, the optimization is run again but is preceded by a delay phase. The vehicle's maximum allowed initial velocity is reduced by 0.5 MPH from the speed limit in this phase. If the vehicle's initial velocity is above this new limit, it will be forced to depress the brake at an angle of 19^0 until reaching this speed before optimizing again. If this optimization still results in a collision, then the maximum allowed initial velocity is reduced by another 0.5 MPH. Adjusting this value of 0.5 MPH upwards results in a faster algorithm. However, it also means that the vehicles may be slowing down more than needed and therefore, introducing more delay. The value chosen here strikes a balance between these design considerations. The process of optimizing and reducing speed is repeated until a collision free trajectory is produced. The time required to complete the collision free path is recorded.

The difference between the collision free time and the optimal time determines a vehicle's delay. The average delay of all vehicles determines the cost of the particular vehicle ordering used. All possible vehicle order combinations are applied to the optimization algorithm and the one yielding the minimum cost is chosen. A procedure for determining the possible combinations is presented by Li and Wang [60]. Notice, however, that the number of combinations increases rapidly as more vehicles are included. Since the maximum range of WAVE is 1 km and the speeds of the vehicles being optimized are near 50 MPH, the optimization algorithm only has approximately 22 seconds to make a decision in the worst case scenario. In the implementation of a traffic optimization algorithm, the performance of the optimization must be balanced with the required processing time. The number of combinations that must be considered along with the time intensive nature of the optimization make this algorithm unsuitable for dense traffic. Therefore, a limit of 5 vehicles has been allowed for the Sparse Traffic Optimization Algorithm.

The Dense Traffic Optimization Algorithm is designed to make decisions with less information and less time than its sparse counterpart. The sparse algorithm assumes the vehicle network is fully meshed so each vehicle is able to communicate and receive information from every other vehicle. However, the network congestion created by many communicating vehicles means this configuration is difficult to achieve in dense traffic. The sparse algorithm converts to the dense algorithm when the number of vehicles exceeds 5. The dense algorithm is used when the sparse algorithm is unlikely to be executed quickly enough or is unlikely to be supported by the network.

5.2.2 Dense Traffic

The operation of the Dense Traffic Optimization Algorithm resembles that of a traffic signal. Traffic signals are optimal control devices in heavy traffic because they allow the maximum rate of vehicle flow - known as the saturation flow rate - through an intersection. The objective of the Dense Traffic Optimization Algorithm is to maximize the vehicle flow rate. This section will begin by explaining some basic characteristics of vehicle flow in signalized intersections. Then the operation of the Dense Traffic Optimization Algorithm will be described.

The flow rate of one approach through a signalized intersection is characterized by Figure 5.2. When a signal turns green, there is some delay before the vehicles achieve the saturation flow rate. The delay exists for a number of reasons including driver perception time, creation of sufficient headway, and vehicle acceleration characteristics. When a signal turns green, it takes a small amount of time for drivers to observe it and press the throttle. Another cause of delay is that human drivers require a relatively large separation from preceding cars in order to allow for sufficient reaction time. When a signal turns green, vehicles delay accelerating until some headway is created from the vehicle directly in front of them. The accumulation of each vehicle's waiting time makes this a significant source of delay. The third source of delay is caused by the time required to accelerate to the speed limit. The Dense Traffic Optimization Algorithm seeks to improve upon current traffic signals by reducing the delays from the first two sources.

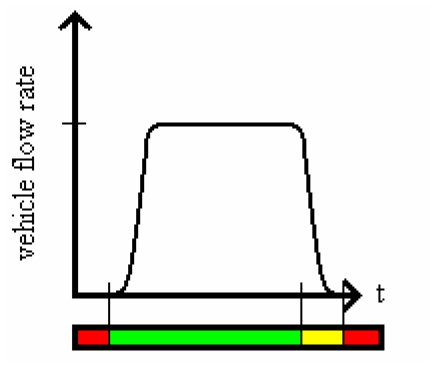


Figure 5.3: Vehicle Flow Rate With Respect to Time and Signal Phase Color.

The Dense Traffic Optimization Algorithm is initialized when the number of vehicles exceeds 5. The algorithm first assigns some vehicles leadership roles. These roles serve to minimize interference by defining how communication takes place between vehicles. A vehicle may be designated as a Primary, Head, Tail, or any combination thereof. A Primary is responsible for most communication between vehicles from its intersection approach and those at other approaches. This role is generally assigned to the first vehicle at each approach. However, if a vehicle is turning right, it will only become a Primary if vehicles going to other directions are not present on the approach. The first vehicle in each lane is a Head. Heads communicate amongst each other and the other vehicles in their lane. They also act as leaders of their platoons. A platoon is defined here as a group of vehicles that move through the intersection together during one traffic cycle. The final vehicle in a platoon is the Tail. The Tail generally communicates with vehicles further behind in an intersection queue and generates confirmation messages. These roles effectively turn the fully meshed network of the Sparse Traffic Optimization Algorithm into a hierarchical network. This configuration is meant to limit which vehicles can generate transmissions, not which vehicles can listen.

Priorities are assigned to different approaches once roles have been assigned. They decide the order each approach will be allocated access to the intersection. The order that each approach will be allowed to enter the intersection is based on the order which Primaries would enter the intersection first based on their current position and speed information. Normally for the dense algorithm, each vehicle does not maintain information about vehicles outside its approach. However, recall that upon transitioning to dense optimization, each vehicle still has information about every other vehicle. The priorities are only calculated once during this transition and are maintained until either an approach is absent of any vehicles when its turn is due or the algorithm transitions back to sparse mode.

The amount of time allocated to each approach is determined once priorities have been assigned. Each Head uses the position and velocity information of vehicles in its lane to determine its platoon's preliminary intersection crossing time. In [13], Garber recommends avoiding cycle lengths greater than 120 seconds so this is established as the maximum cycle length. A Head predicts whether it is able to cross the intersection to its intended destination within 120 seconds based on its position and acceleration characteristics. If it is able, then the Head performs the same prediction for the next vehicle in its lane using that vehicle's position information and the *minimum* of the two vehicle's acceleration capabilities. The minimum is used to account for trucks and other low acceleration vehicles that can restrict following vehicles to slow speeds. The Head also calculates if the vehicle is able to cross the intersection within 3 seconds of the previous vehicle (the vehicle's full acceleration is used to calculate this instead of the minimum). The purpose of this additional check is to increase the efficiency of the intersection. Without this rule, the algorithm could possibly allot a long cycle to only a few vehicles that are sufficiently spread out but able to wirelessly communicate. This process is repeated until the 3 second rule is not met, all vehicles in the lane have been analyzed, or the 120 second limit has been reached. The preliminary intersection crossing time for each lane is then transmitted to the Primary of that approach.

The Primary calculates the maximums and averages of the preliminary crossing times for the left turn lanes, straight lanes, and right turn lanes. It is assumed that the vehicles in each lane only desire to go in one direction unless the approach only has one lane (i.e. there are no lanes where a vehicle may choose to turn or go straight and there are no U-turns). Unoccupied lanes are ignored. The average of the preliminary crossing times for lanes moving in a direction are used to prevent unequal vehicle distribution from drastically reducing the capacity of the intersection. The averages and maximums are exchanged among Primaries then distributed among the Heads of each lane.

The Heads combine this data with earlier approach priority information to make a schedule detailing the time each lane is allowed to cross the intersection. If the average crossing time for the right turn lanes is more than twice the maximum of the other two averages, the average preliminary crossing times for all lanes on the approach are once again averaged to determine the crossing time for this approach. Right turns can usually be taken without needing to plan separate time. When the same approach is going straight, or when the approach on the right is taking a left, vehicles are able to safely take right turns. If the average for the right turn lanes does not meet this criterion, then the maximum of the averages for the left and straight lanes determines the crossing times for this approach.

The Heads propagate this information to other cars in their lanes and form a platoon with the vehicles that are able to meet this crossing time. Each vehicle independently determines its set of controls to comply with the chosen acceleration rate considered in the preliminary calculation. The Tails of these new platoons notify the next vehicle (if one is present) that it will be the Head of the next platoon. The new Head that is positioned closest to the intersection entrance becomes the new Primary when the current Primary crosses the intersection. If there is a conflict, the vehicle with the lowest value MAC address assumes this role. Whenever a platoon finishes traversing the intersection, the Tail notifies all Heads. The Heads on an approach begin moving when they hear notifications from the previous approach's Tails and determine it is their turn. When each Head begins moving, it immediately signals following vehicles in its lane to accelerate.

Communicating information for each lane allows vehicles from other approaches to take advantage of normally unused intersection capacity. Some common situations are illustrated in Figure 5.3. Assume that vehicles are distributed as shown and it is the South approach's turn to cross. When the South approach begins crossing, it is possible for the North approach to take a left turn. By communicating the planned time for each approach, the vehicle can cross if it determines it is able to. Now assume the approach from the West is scheduled to cross. Since the vehicles are going straight, the vehicle on the East approach can cross during the same period as can the right turning vehicle on the North approach. In these situations, the algorithm prioritizes left turns first, straight movements, then lastly, right turns.

The Dense Traffic Optimization Algorithm improves on the performance of today's traffic signals in many respects. By communicating crossing times beforehand, the vehicles do not have to guess when they can cross. The algorithm saves fuel since vehicles will no longer accelerate to an intersection only to have the signal turn red once they are nearby. The algorithm also removes the dilemma zone since there are no yellow lights. Recall how the vehicle flow rate in Figure 5.2 takes quite some time to reach the saturation flow rate during human driving. This algorithm greatly minimizes the amount of delay in the upwards curve by signaling when to begin acceleration and using speed automation to control vehicle movement. Since the Head immediately begins moving once it is its turn, the delay in the driver perceiving a signal change is also removed. The amount of improvement achieved with this algorithm in comparison to conventional traffic signals is analyzed in Chapter 7.

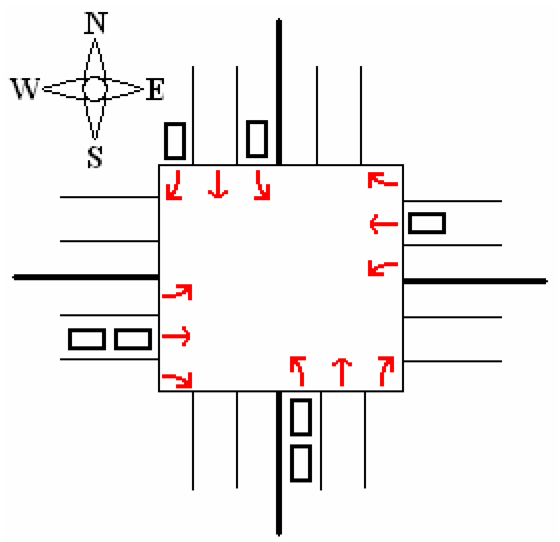


Figure 5.4: Example Intersection Scenario.

5.3 Summary

The operation of the vehicular network was presented in this chapter. The new IEEE 1609 family of standards have been introduced and are used to facilitate transmission of EKF observations and to support the optimization algorithm. The Sparse and Dense Traffic Optimization Algorithms have also been analyzed.

Chapter 6 - Stability Analysis of Vehicle Model and Network

This chapter will analyze the stability of the vehicle state equations and the optimization algorithm of the cooperative vehicle network. For the stability analysis, any physical constraints placed on the states (i.e. $m_a \geq 0$, $w_e \geq 0$) have been relaxed to observe the system's response. Although the vehicles in this system are never allowed to violate these constraints, future applications of this model may cause the states to venture outside of the constrained area. For example, it is possible that the vehicle model presented here may be applied to a boat engine where a negative engine angular velocity may be permitted. The analysis here is presented for completeness. The findings presented here only *indicate* possible shortfalls of modeling a real vehicle with these state equations since some state responses may not be physically possible.

6.1 Vehicle State Space Model Stability

Analysis of the vehicle model's stability shows how the system behaves at any given set of states. It is important to ensure that the model's operating range is free from unstable points and areas that may result in undesired operation. The stability is analyzed by finding the eigenvalues of some of the model's equilibrium points. The states are then perturbed in different ways to view how the system reacts.

6.1.1 Equilibrium Points

An equilibrium point is a state vector $\underline{x}_{eq}$ that satisfies the equation

$$\dot{\underline{x}} = \underline{f}[\underline{x}_{eq}, \underline{u}, t] = 0 \quad \forall \quad t, \qquad \textbf{(110)}$$

for a fixed control vector $\underline{u}$. The point is considered unstable or repellent if a small perturbation causes the system to depart from the point. It is considered stable or attractive if a small perturbation results in the eventual return of the system to that point. In terms of the vehicle model, unstable points are dangerous because drivers expect stable vehicles. Imagine a vehicle where a slight depression of the gas pedal and subsequent release - analogous to perturbing from an equilibrium point - causes the vehicle to accelerate rapidly. Instability is clearly undesirable and steps must be taken to avoid its occurrence.

The equilibrium points for the vehicle model can generally be found using any root-finding method. The Newton-Raphson method was chosen to find the equilibrium points shown in the phase plot of Figure 6.1. Note that the equilibrium points form a continuous line but have only been evaluated for discrete integer values of the throttle angle then interpolated. The equilibrium points for small throttle angles ($a < 5^{o}$) have not been shown due to the increased computational time required to solve for them with this method.

Figure 6.1 shows that as the throttle angle increases, the engine's ability to receive air into the intake manifold approaches a limit. The equilibrium point corresponding to a throttle angle of 85 degrees gives the states of the vehicle at maximum velocity.

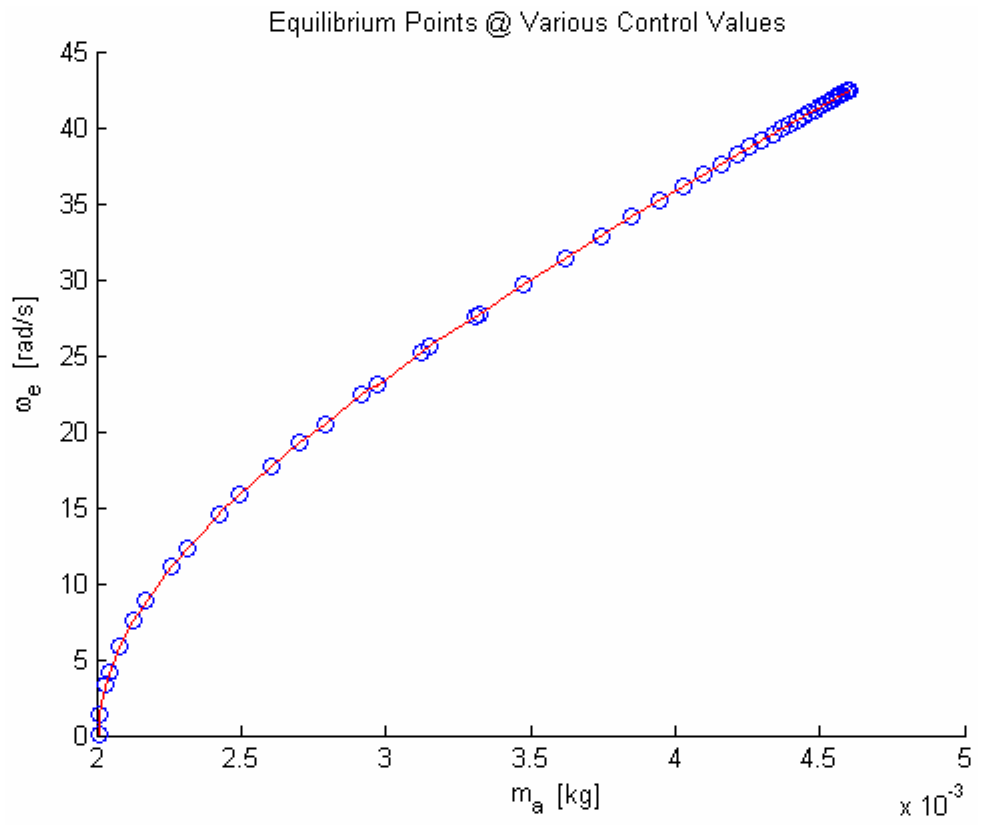


Figure 6.1: Phase Plot of Vehicle State Space Model Stable Equilibrium Points Determined Using Newton-Raphson Method For Throttle Angles 5-85^O, $b = 0^0$, $T_b = 0$, $R_g = \mathrm{R}_5$ *and Error* = 10^{-5}.

6.1.2 Modes of Oscillation

The stability of each equilibrium point can be determined by finding the eigenvalues of the Jacobian matrix at that point. If the real part of the eigenvalues are on the left hand side of the complex plane, then the equilibrium point is stable. Eigenvalues are found for a subset of throttle angles and are shown in Table 6.1. Results show that all equilibrium points are stable.

Table 6.1: Eigenvalues of the Jacobian Matrix at Various Equilibrium Points.

Equilibrium Point	Non-Zero Eigenvalues
$\alpha = 10$ [deg] $m_a = 0.00299988000001$ [kg] $\omega_e = 23.49456305508211$ [rad/s]	-2.08847680245321 +20.50408656510694i -2.08847680245321 -20.50408656510694i -10.00000000000000
$\alpha = 25$ [deg] $m_a = 0.00437648305663$ [kg] $\omega_e = 40.01957422208155$ [rad/s]	-42.45689015940916 -13.69559299102246 -10.00000000000000
$\alpha = 40$ [deg] $m_a = 0.00459331634648$ [kg] $\omega_e = 42.39547321296866$ [rad/s]	-171.063700088545 -3.547931987130 -10.000000000000
$\alpha = 60$ [deg] $m_a = 0.00467033624071$ [kg] $\omega_e = 43.23145107707402$ [rad/s]	-419.939545265642 -1.468851979803 -10.000000000000
$\alpha = 85$ [deg] $m_a = 0.00470114670659$ [kg] $\omega_e = 43.56481006480795$ [rad/s]	-869.142632185972 -0.715912820180 -10.000000000000

6.1.3 Determining the Equilibrium Points' Basins of Attraction

After finding the stable equilibrium points, the ability of the points to return to their stable states after being perturbed is studied. The area on the phase plot that converges to a particular equilibrium point is defined as that point's basin of attraction. This area is found by setting the initial states of the vehicle to some perturbed point, letting the throttle angle equal a fixed value, then integrating the state equations with respect to time. If the states approach the equilibrium point as time tends to infinity, then the perturbed point is located in the basin of attraction. Otherwise, if the states depart, the initial point is located in an unstable area.

The equilibrium point corresponding to a throttle angle of 85^{O} is initially chosen for simulation. First the initial conditions are perturbed from the equilibrium point but kept inside the vehicle's Normal Operating Area. The Normal Operating Area is defined from 0-0.005 kg for m_a and 0-50 rad/s for w_e and is dependent on the vehicle model parameters. This area encompasses all physically possible states the vehicle should take by natural adjustment of its controls. It is important to ensure that this area is within the basin of attraction, otherwise normal driving may result in unstable behavior. After conducting numerous simulations, it is surmised that this area is stable for all initial combinations of m_a and w_e for all throttle angles.

Next, the equilibrium point is perturbed below the Normal Operating Area. With the initial condition for m_a fixed at 0.004 kg, the initial value of w_e is repeatedly reduced at the beginning of each simulation run until the vehicle exhibits unstable behavior. The boundary of the basin of attraction is found to lie between -43 and -44 rad/s where all values less than or equal to -44 display instability and all values greater than or equal to -43 but below the Normal Operating Area display stable behavior. The engine angular velocity of the stable case is shown in Figure 6.2 and the unstable case in Figures 6.3. The mass of air in the intake manifold remains constant throughout both simulations.

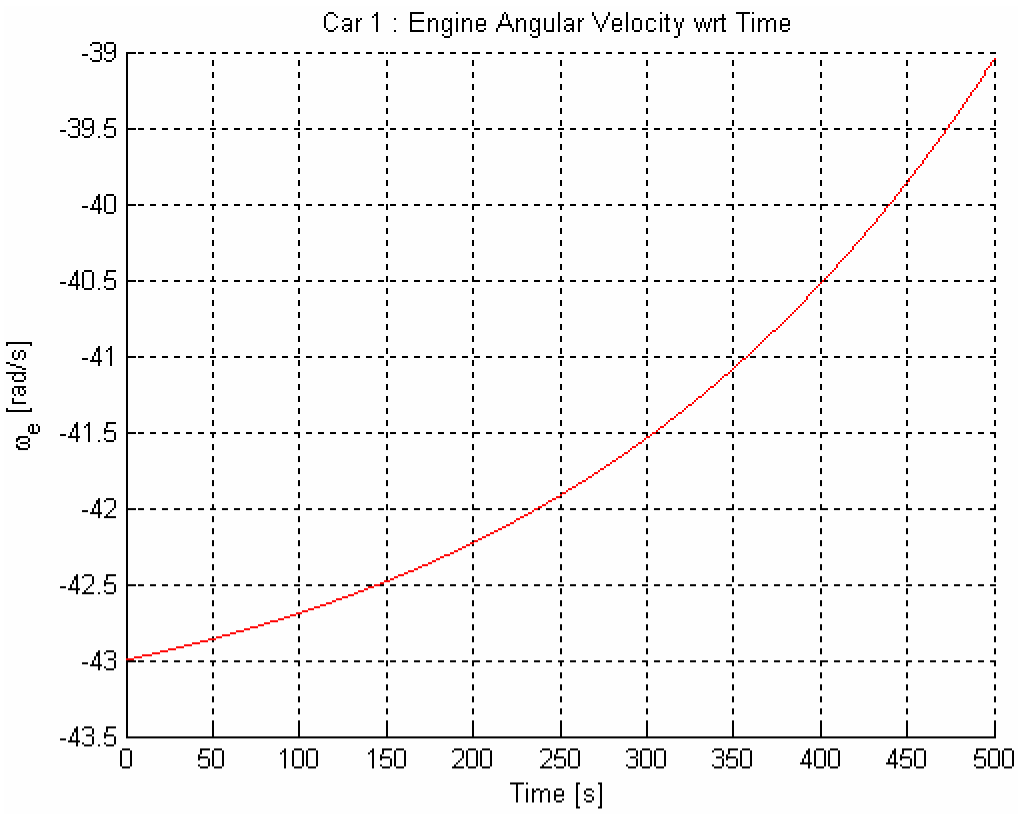


Figure 6.2: Engine Angular Velocity with Respect to Time for $a = 85^0$, $b = 0^0$, $R_g = R_5$, and initial conditions m_a = 0.004 kg and w_e = -43 rad/s.

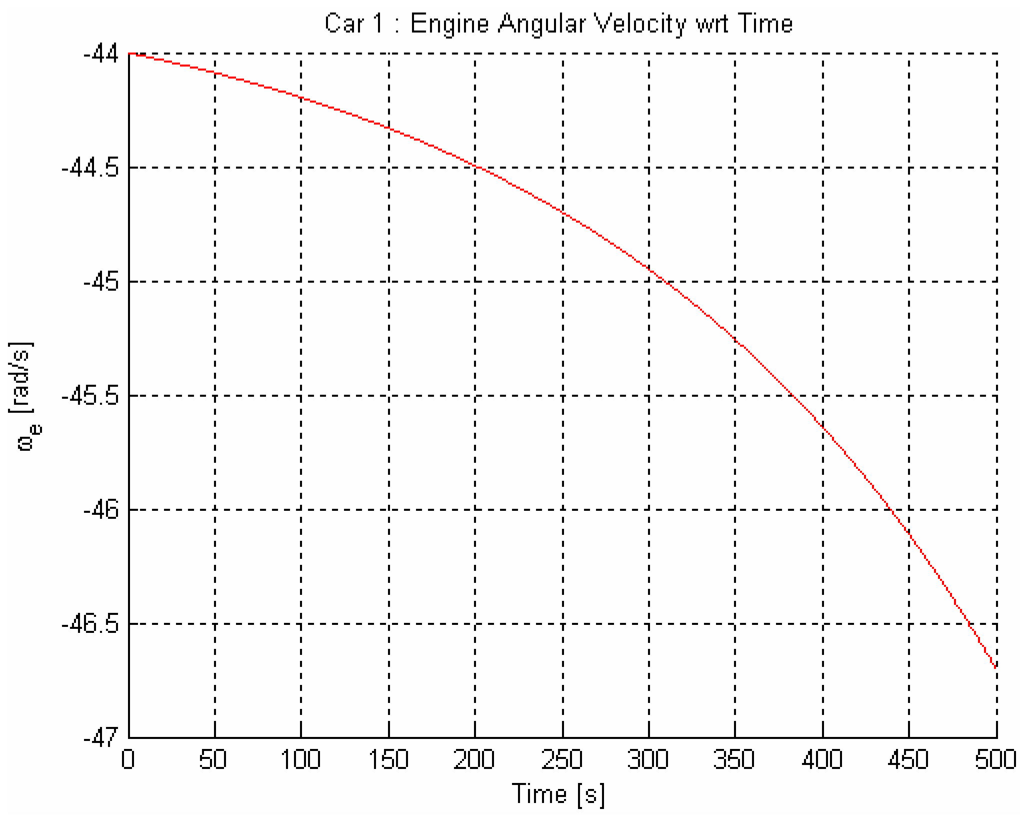


Figure 6.3: Engine Angular Velocity with Respect to Time for $a = 85^0$, $b = 0^0$, $R_g = \mathrm{R}_5$*, and initial conditions* m_a *= 0.004 kg and* w_e *= -44 rad/s.*

This process used to find the boundary of the basin of attraction is repeated for different values of m_a and the results are shown in Figure 6.4 and listed in Table 6.2. The point is then perturbed to the left, right, and upwards direction from the Normal Operating Area but the increasing computational load is unable to be processed in a timely fashion. Perturbations applied over to these areas on the phase plot require the integration algorithm to use smaller and smaller step sizes. These areas are neglected and the lower portion of the plots is focused on. The basins of attraction for other throttle angles are shown in Figures 6.5 and listed in Table 6.2. Since the equilibrium points for high throttle angles are very close together, their basins of attraction are very similar.

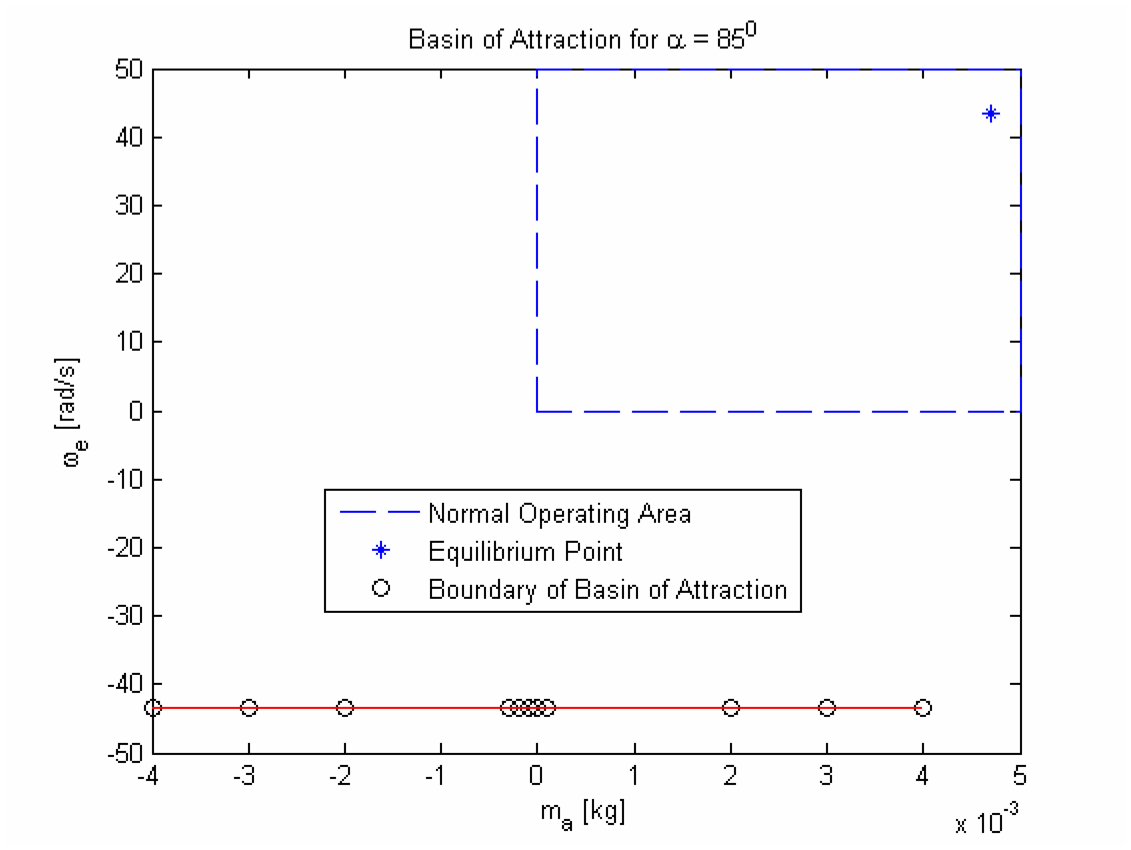
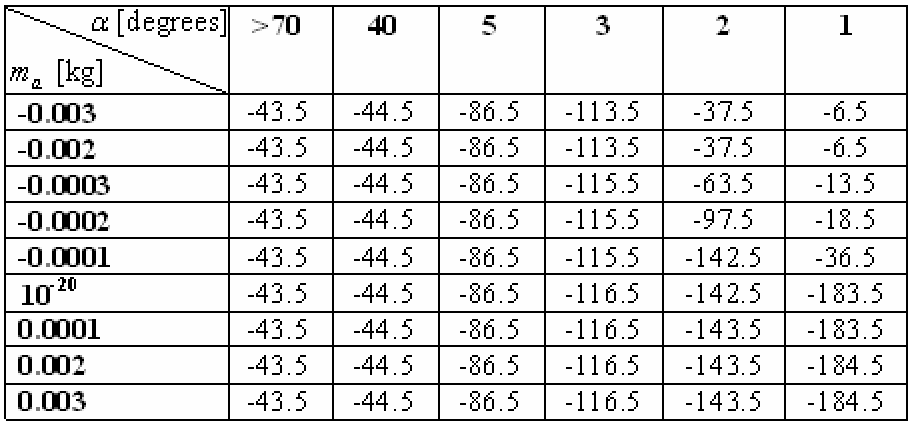


Figure 6.4: Boundary of Basin of Attraction for $a = 85^0$, $b = 0^0$, and $R_g = R_5$. The red line is an interpolation of the boundary.

Table 6.2: Boundary of Basin of Attraction for Various Throttle Angles, $b = 0^0$, and $R_g = R_5$. Results are accurate to within +\- 0.5.



α [degrees] / m_a [kg]	>70	40	5	3	2	1
-0.003	-43.5	-44.5	-86.5	-113.5	-37.5	-6.5
-0.002	-43.5	-44.5	-86.5	-113.5	-37.5	-6.5
-0.0003	-43.5	-44.5	-86.5	-115.5	-63.5	-13.5
-0.0002	-43.5	-44.5	-86.5	-115.5	-97.5	-18.5
-0.0001	-43.5	-44.5	-86.5	-115.5	-142.5	-36.5
10^{-20}	-43.5	-44.5	-86.5	-116.5	-142.5	-183.5
0.0001	-43.5	-44.5	-86.5	-116.5	-143.5	-183.5
0.002	-43.5	-44.5	-86.5	-116.5	-143.5	-184.5
0.003	-43.5	-44.5	-86.5	-116.5	-143.5	-184.5

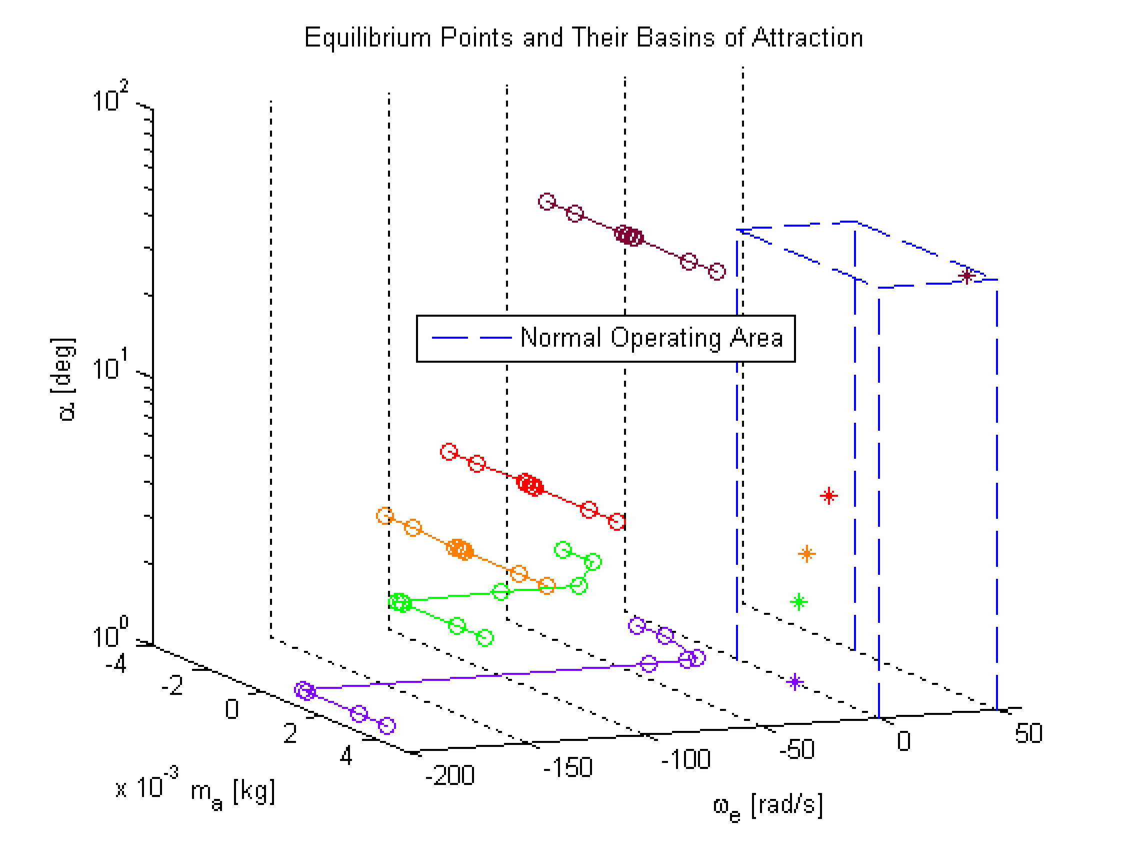


Figure 6.5: Equilibrium Points and the Boundaries of Their Basins of Attraction for Various Throttle Angles, $b = 0^0$, and $R_g = R_5$.

Figure 6.5 shows that as the throttle angle increases, the basin of attraction contracts and approaches the normal operating area. For throttle angles below 3^0, the lower boundary of the basin of attraction rises sharply in the area where m_a becomes negative. In the next section, the throttle angle is further reduced to observe how the vehicle reacts.

6.1.4 State Oscillations

Limit cycles are self-sustained oscillations that can occur in nonlinear systems. When the throttle angle is small, the response of the engine angular velocity with respect to the mass of air in the intake manifold can become under damped over time as shown in Figure 6.6. Although this is not a limit cycle, it has been shown to illustrate the dampening effect of the throttle angle. A possible response when the throttle angle is completely released is shown in Figure 6.7. This plot demonstrates a limit cycle since releasing the throttle causes the system to operate indefinitely without any additional energy. From Figures 6.6 and 6.7, it is clear that the throttle angle is directly related to the amount of dampening experienced by the system.

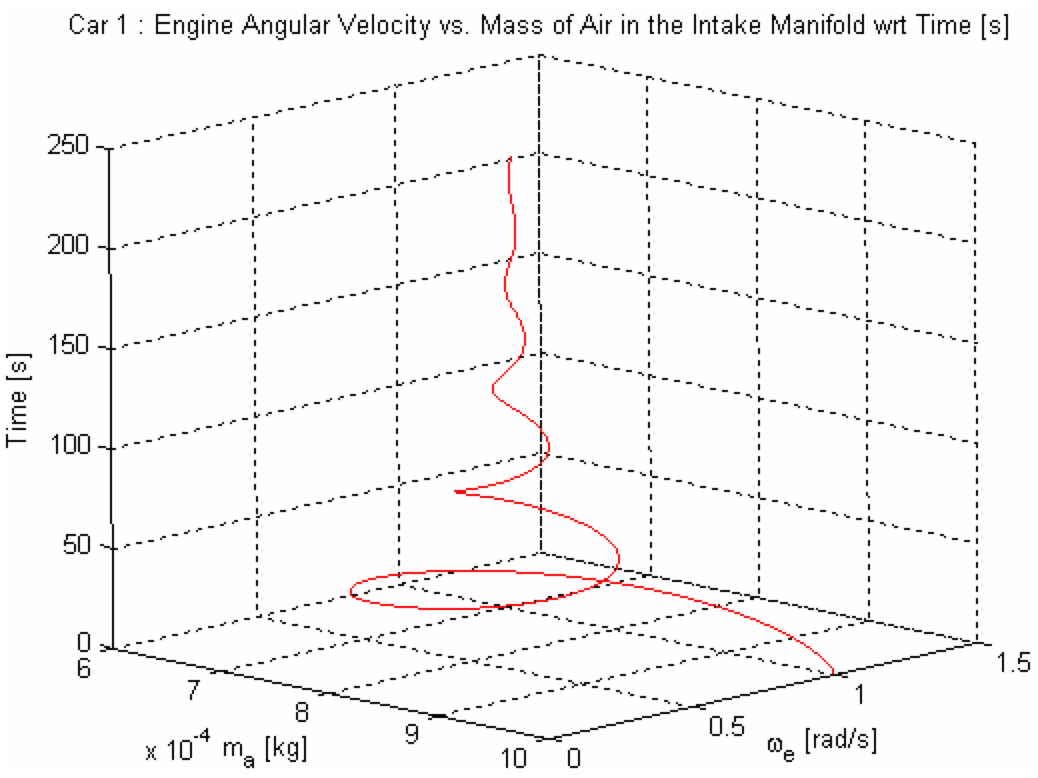


Figure 6.6: Engine Angular Velocity vs. Mass of Air in Intake Manifold with Respect to Time for $a = 0.5^0$, $b = 0^0$, $R_g = \mathrm{R}_1$, *and initial conditions* m_a *= 0.001 kg and* w_e *= 1 rad/s.*

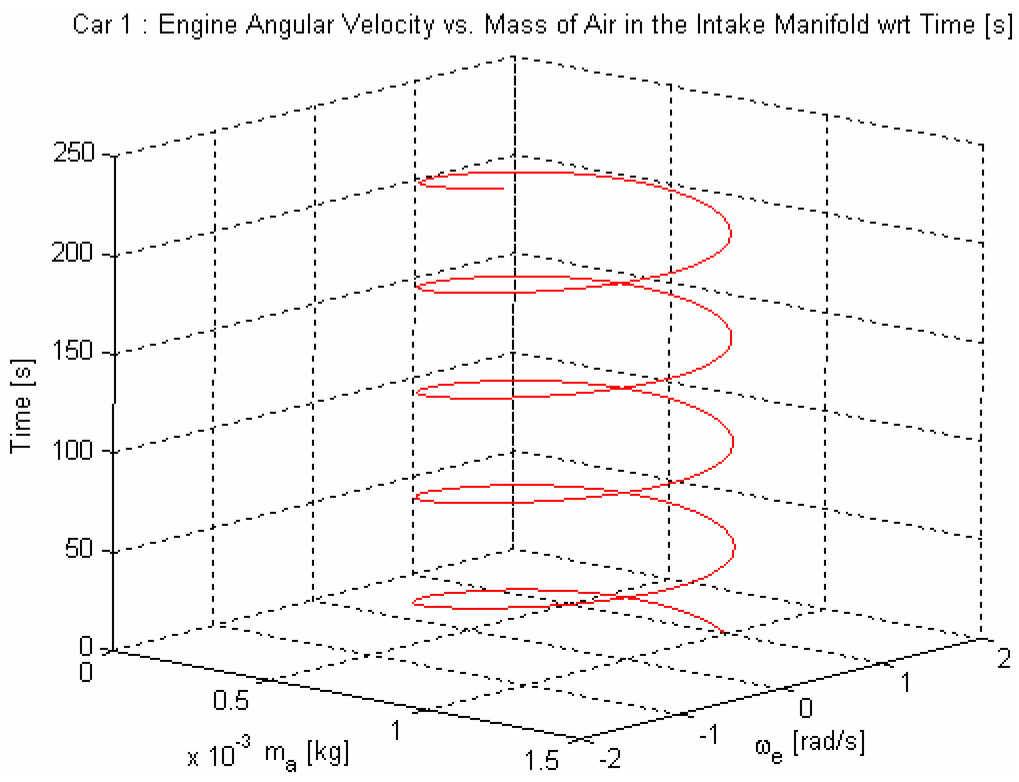


Figure 6.7: Engine Angular Velocity vs. Mass of Air in the Intake Manifold with Respect to Time for $a = 0^0$, $b = 0^0$, $R_g = \mathrm{R}_1$, *and initial conditions* m_a *= 0.001 kg and* w_e *= 1 rad/s.*

The oscillations in Figures 6.6 and 6.7 are the vehicle's attempt to maintain the mass of air and the engine angular velocity in equilibrium. When w_e is greater than the equilibrium value and m_a is at the equilibrium value at a fixed throttle angle, the engine combusts too much air causing m_a to decrease while w_e moves toward equilibrium. m_a and w_e eventually decrease to the point where the throttle is allowing in more air than the vehicle is exhausting, which allows m_a to increase again. This enables w_e to increase since more air causes the engine to do more work. This process repeats until a steady state is reached.

It is important to note that these results *can* naturally occur in the system but are usually transient and the result of being perturbed from equilibrium by an initial condition mismatch. Under normal circumstances when m_a and w_e are in equilibrium, releasing the throttle angle causes the states to gradually decay as opposed to oscillate.

The results in Figures 6.6 and 6.7 simply do not occur in real vehicles and something about the model must be changed. Fortunately, forcing the model to obey the basic physical constraints prevents most of these problems. By maintaining $m_a \geq 0$ and $w_e \geq 0$, most unstable situations are avoided. Additionally, for the collision avoidance simulations in this thesis, the behavior of the engine at low throttle angles is not of great concern since all accelerations take place at high throttle angles.

6.1.5 Transmission Instability

The vehicle's transmission can display instability around the points where gear shifts occur. It can be made to repeatedly alternate between an arbitrary gear and the next higher gear for a small period of time causing the states m_a and w_e to oscillate. The reasons why this occurs and the conditions that must exist will be discussed in this section.

At each iteration of the EKF, the transmission decides which gear to use based on the velocity of the vehicle (assume $a \neq 85$). Recall the vehicle velocity given by (3). Since h is constant, whenever R_g is increased during a gear shift, w_e must be reduced to maintain the same velocity. This large, discontinuous change in w_e can break the equilibrium with m_a and become the source of oscillations when the vehicle acceleration is very low.

A simulation has been designed to demonstrate the instability of the transmission in this case. The vehicle has been initialized slightly below the point that it normally shifts to third gear and the initial values of m_a and w_e are in equilibrium to prevent oscillations from mismatch. The throttle angle is set to a value $\left(4.5623^0\right)$ that is only slightly more than what is needed to maintain the vehicle at zero acceleration $\left(4.4623^0\right)$. The results of this simulation are displayed in Figures 6.8-6.11.

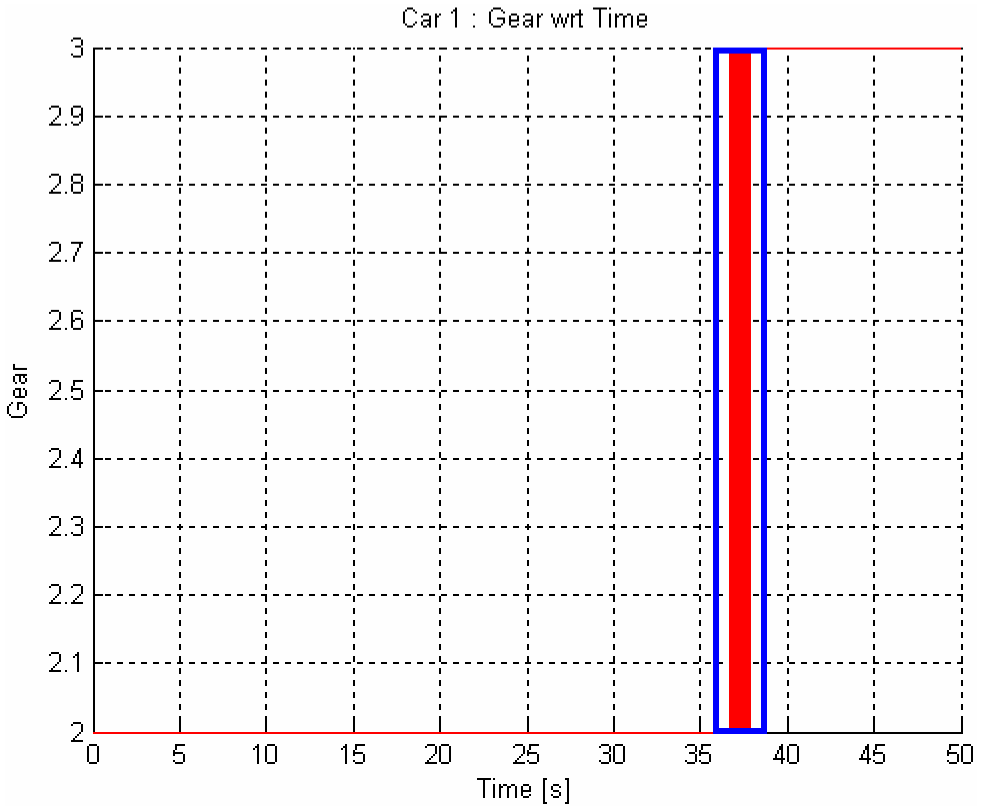


Figure 6.8a: Gear with Respect to Time for $a = 4.5623^0$, $b = 0^0$ *and initial conditions* m_a *= 0.00111331477934 kg and* w_e *= 29.61976701882556 rad/s.*

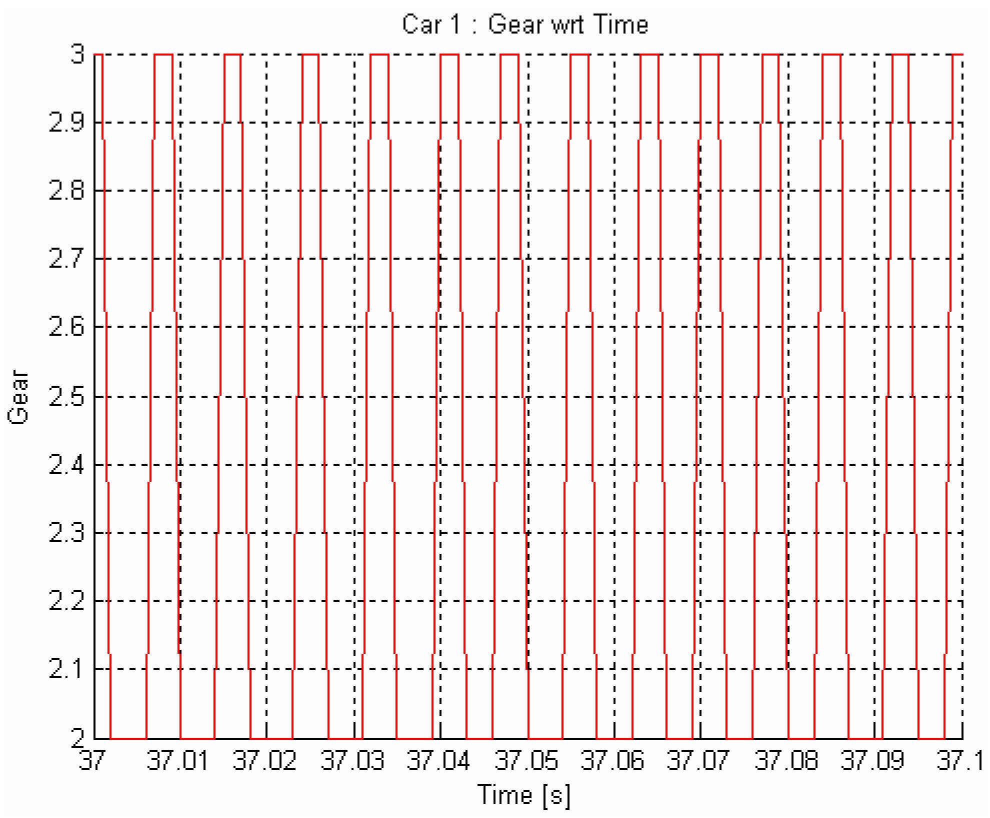


Figure 6.8b: Gear with Respect to Time for $a = 4.5623^0$, $b = 0^0$ *and initial conditions* m_a *= 0.00111331477934 kg and* w_e *= 29.61976701882556 rad/s. This plot is a magnification of the blue box in Figure 6.8a.*

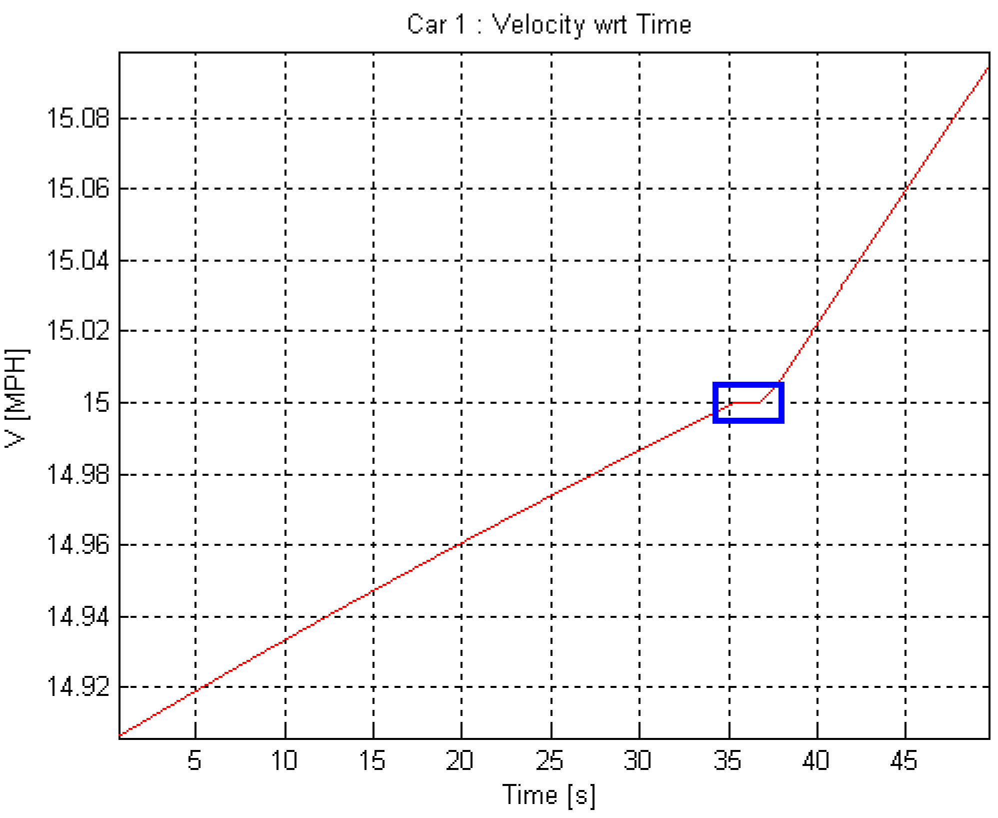


Figure 6.9a: Velocity with Respect to Time for $a = 4.5623^0$, $b = 0^0$ *and initial conditions* m_a *= 0.00111331477934 kg and* w_e *= 29.61976701882556 rad/s.*

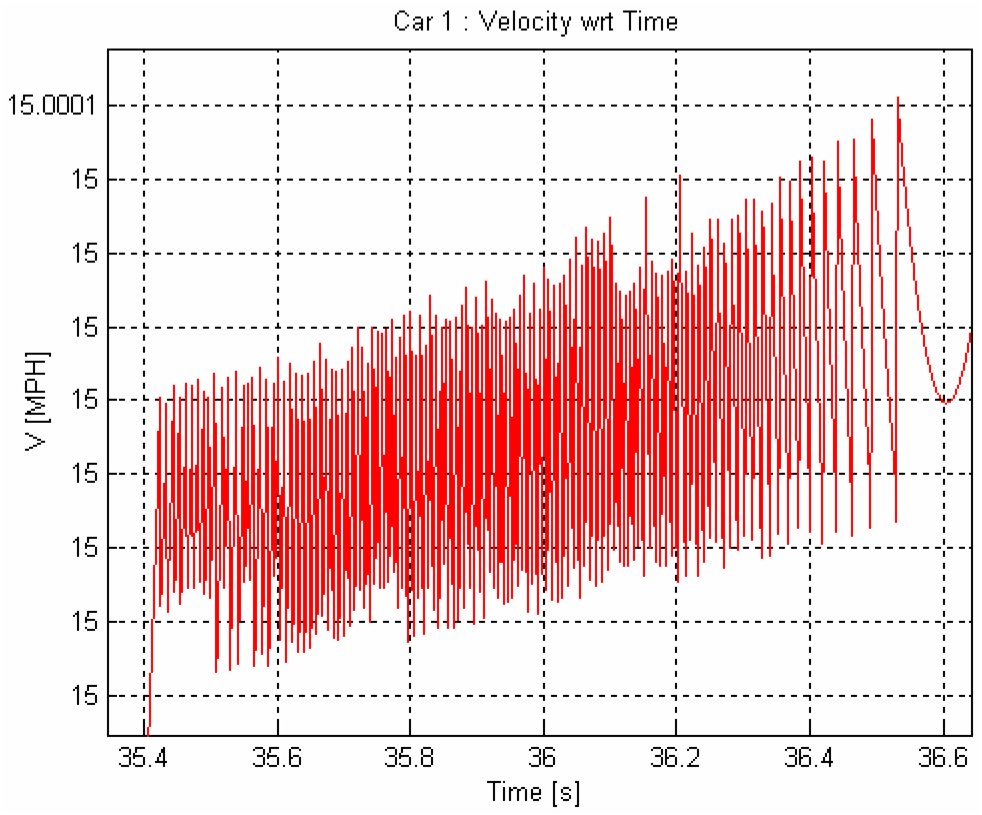


Figure 6.9b: Velocity with Respect to Time for $a = 4.5623^0$, $b = 0^0$ *and initial conditions* m_a *= 0.001113331477934 kg and* w_e *= 29.61976701882556 rad/s. This plot is a magnification of the blue box in Figure 6.9a.*

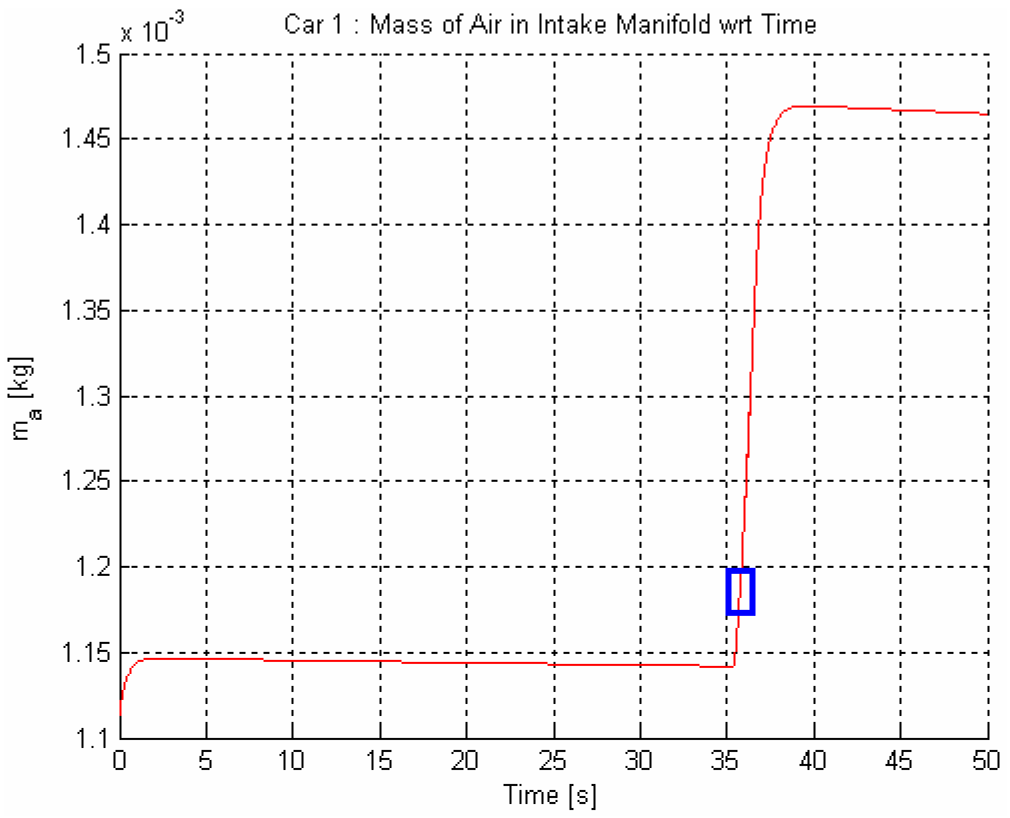


Figure 6.10a: Mass of Air in Intake Manifold with Respect to Time for $a = 4.5623^0$, $b = 0^0$ *and initial conditions* m_a *= 0.00111331477934 kg and* w_e *= 29.61976701882556 rad/s.*

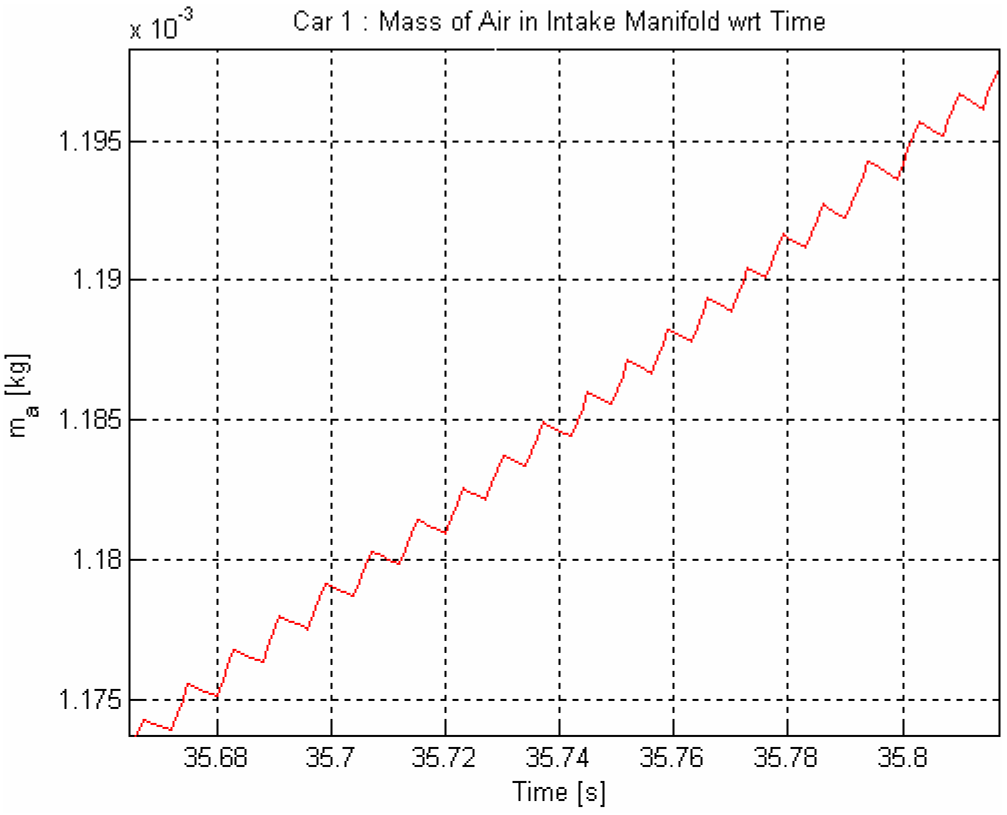


Figure 6.10b: Mass of Air in Intake Manifold with Respect to Time for $a = 4.5623^0$, $b = 0^0$ *and initial conditions* m_a *= 0.00111331477934 kg and* w_e *= 29.61976701882556 rad/s. This plot is a magnification of the blue box in Figure 6.10a.*

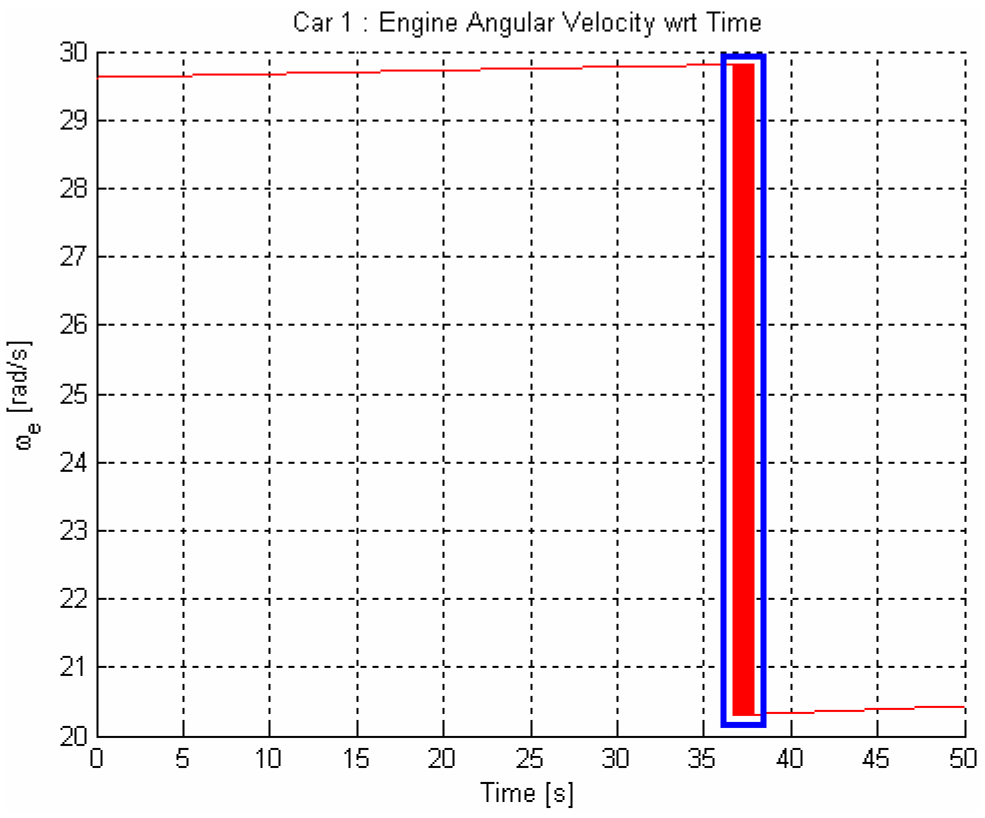


Figure 6.11a: Engine Angular Velocity with Respect to Time for $a = 4.5623^0$, $b = 0^0$ *and initial conditions* m_a *= 0.00111331477934 kg and* w_e *= 29.61976701882556 rad/s.*

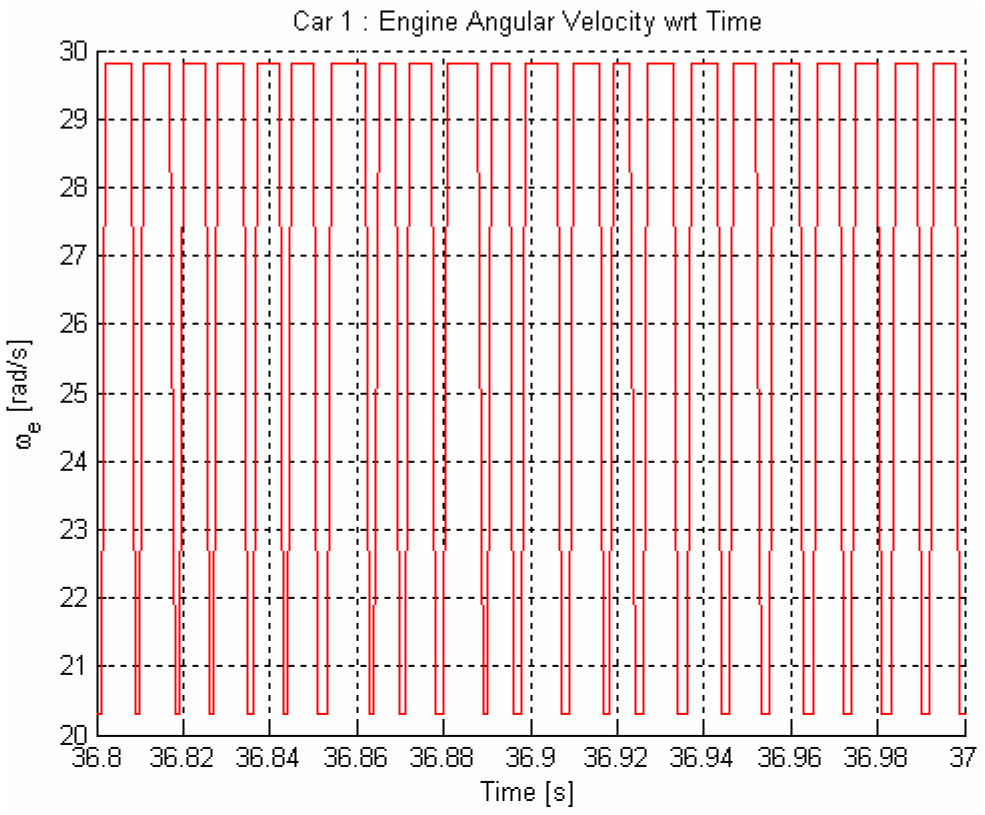


Figure 6.11b: Engine Angular Velocity with Respect to Time for $a = 4.5623^0$, $b = 0^0$ *and initial conditions* m_a *= 0.00111331477934 kg and* w_e *= 29.61976701882556 rad/s. Plot is a magnification of the blue box in Figure 6.11a.*

The thick line in Figure 6.8 shows that the transmission is repeatedly alternating between second and third gear. Normally, the velocity of the vehicle increases linearly for a constant throttle angle. However, the instability from the gear oscillations causes the velocity to stop increasing and become noisy as shown in Figures 6.9a and 6.9b. It seems that this instability is caused by the sudden change of w_e while shifting. The engine attempts to return m_a and w_e to equilibrium but small changes to the states result in gear shifts as shown in Figures 6.10a, 6.10b, and 6.11. This problem is best prevented by making sure that the vehicle's acceleration is sufficiently high to avoid oscillations. The absence of oscillations during high acceleration is illustrated by Figure 6.12.

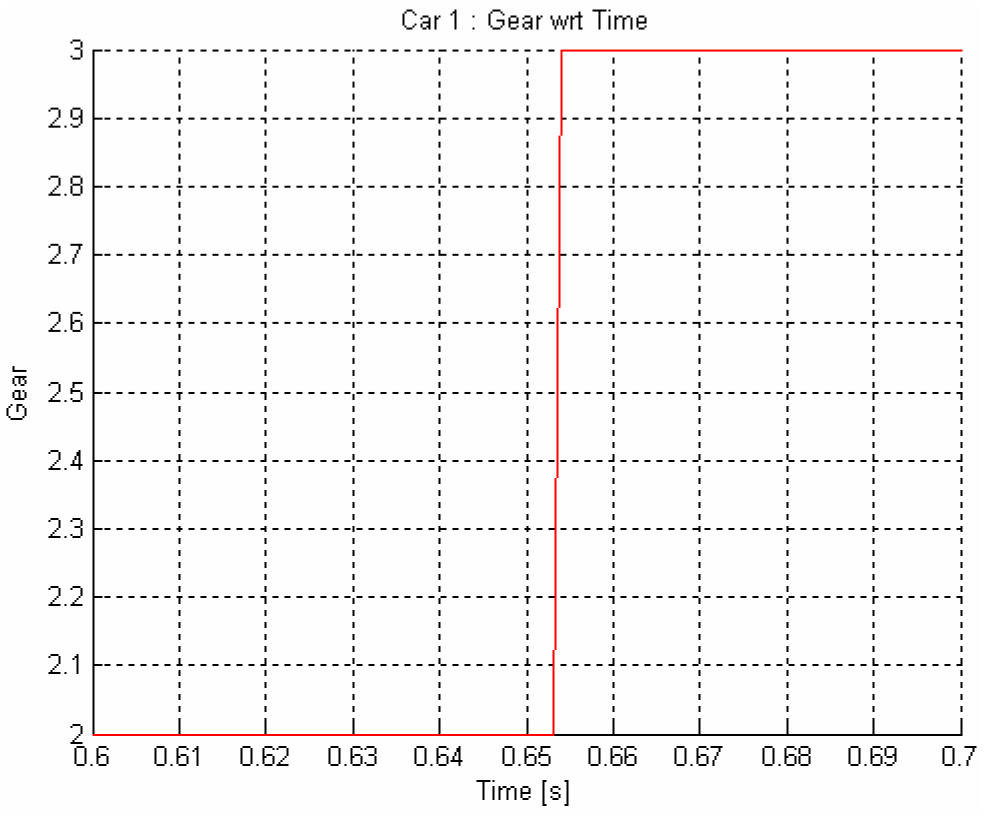


Figure 6.12: Gear with Respect to Time for $\alpha = 10^0$, $\beta = 0^0$ *and initial conditions* m_a = *0.00111331477934 kg and* w_e = *29.61976701882556 rad/s.*

6.2 Stability Analysis of the Sparse Traffic Optimization Algorithm

Simulation results showed that the Sparse Traffic Optimization Algorithm can produce instability under certain conditions. We will analyze these conditions next. The algorithm is executed when a new vehicle enters the communication range of the other vehicles approaching the intersection. If the vehicles that are added to the network later exhibit a lower cost, then they will cause the controls of vehicles present earlier to alternate between braking and accelerating. As more vehicles approach, it is possible that these oscillations will affect every vehicle. In this section, an example is presented to reveal how this instability can occur.

Instability can be illustrated by studying a scenario of three cars moving toward a four approach, single lane intersection in a rural area with a speed limit of 50-MPH. Let the first vehicle approach from the north at an initial velocity of 45 MPH, the second vehicle approach from the west at a velocity of 45.25 MPH, and the third vehicle approach from the south at 45.50 MPH. Assume all vehicles have the same internal characteristics, want to take left turns, and are initially positioned 1 km away from the center of the intersection. This scenario will now be analyzed from the perspective of vehicle 1.

Initially, vehicles 2 and 3 are outside the communication range of vehicle 1; consequently, the latter assumes it is the only one approaching the intersection. Vehicle 1 executes the optimization algorithm and produces the sequence of throttle values shown in Figure 6.13. Notice that the optimization results show that it undergoes maximum acceleration for 25 seconds until reaching the speed limit. It then maintains the speed limit until the 43 second mark after which it brakes, and then turns. Finally, it crosses the intersection and begins to accelerate at the maximum rate.

Let vehicle 1 and vehicle 2 come within communication range at 10 seconds. Now the optimization algorithm executes again and the output for vehicle 1 is shown in Figure 6.14. Note that all values before the 10 second mark are past values that have already been completed. The values afterwards are planned future actions that are output from the optimization algorithm. Also note the release of the throttle at the 10 second interval. Since vehicle 1 now has a higher cost than vehicle 2, it is forced to brake to avoid a collision.

Now, let vehicle 3 join the network at the 20 second interval. Since vehicle 3 now has the lowest cost of the three vehicles, vehicle 1 and 2 are forced to press the brakes. The output of the optimization for the throttle angle of vehicle 1 is shown in Figure 6.15. Whenever a new vehicle is added to the existing network that has a slightly lower cost than an existing vehicle and whose intended path intersects that of an existing vehicle, the optimization algorithm will cause the old vehicle to brake then accelerate. The vehicular response resembles a set of masses connected by springs. Whenever, a new mass is added, the positions of the other masses are perturbed and they begin to oscillate.

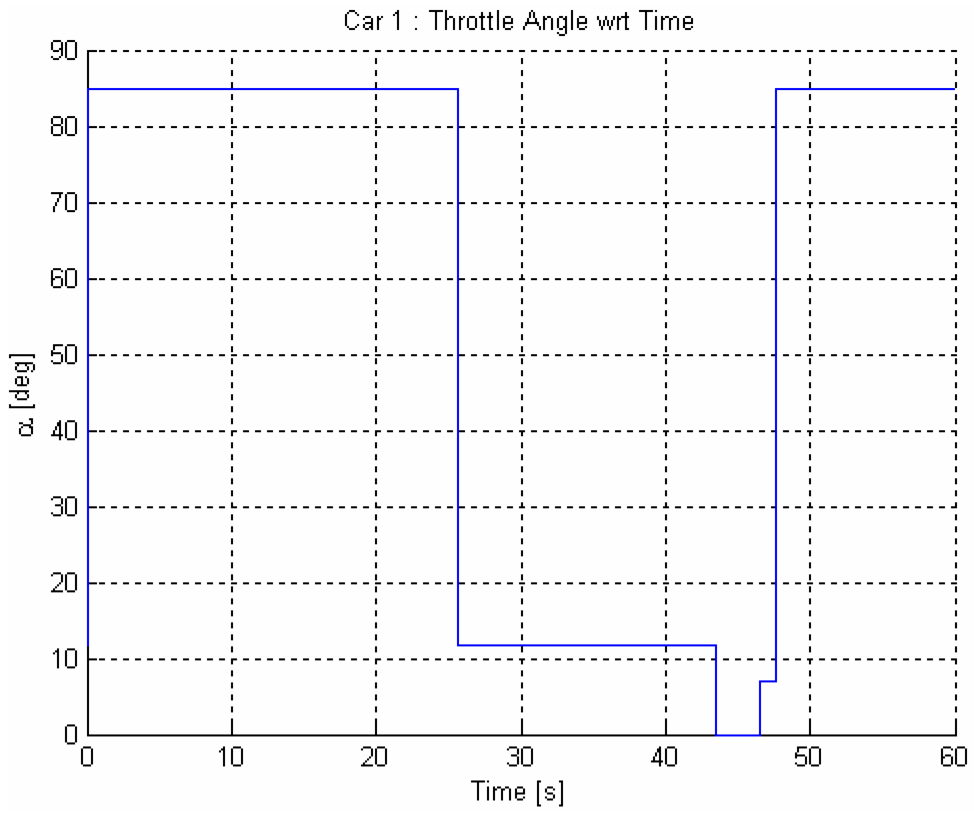


Figure 6.13: Vehicle 1 Throttle Angle With Respect to Time Determined from Optimization Algorithm Executed Near 0 Seconds.

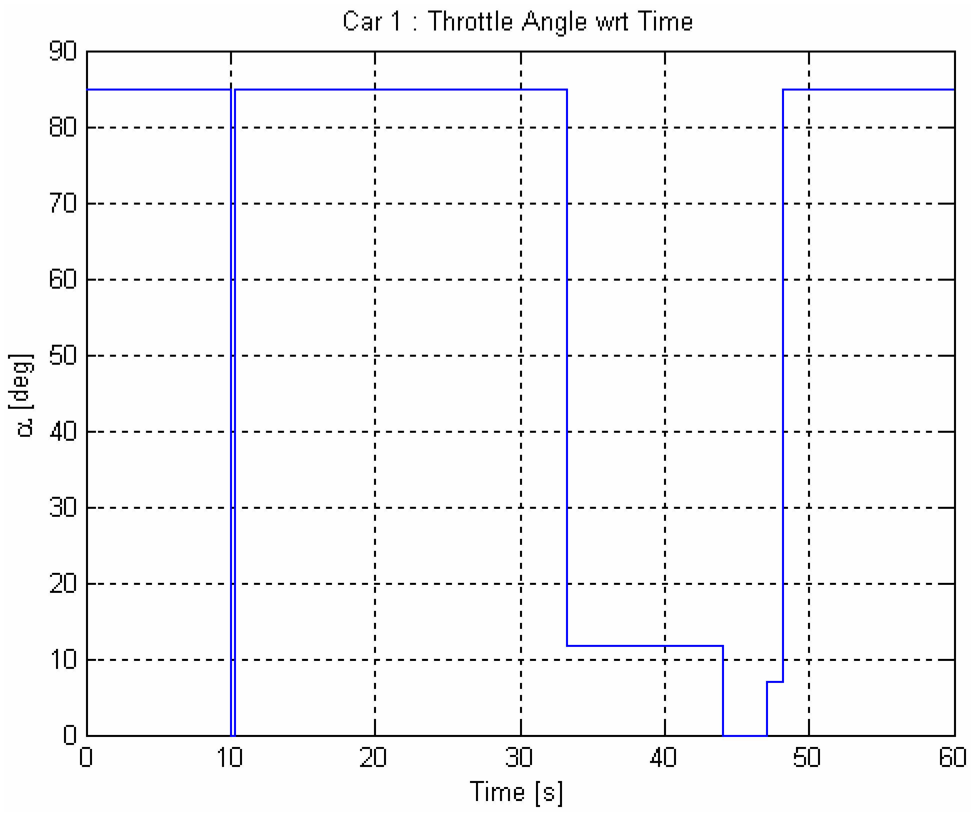


Figure 6.14: Vehicle 1 Throttle Angle With Respect to Time Determined from Optimization Algorithm Executed At the 10 Second Interval.

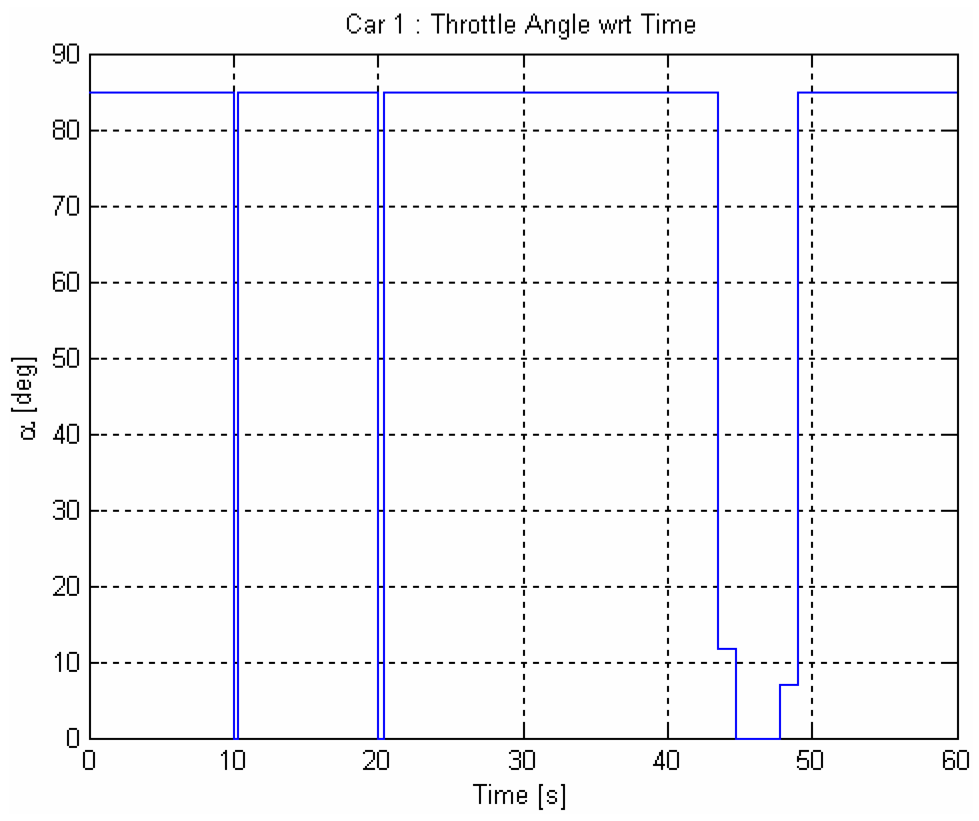


Figure 6.15: Vehicle 1 Throttle Angle With Respect to Time Determined from Optimization Algorithm Executed At the 20 Second Interval.

6.3 Summary

The stability of the vehicle state equations has been analyzed in this chapter by finding the eigenvalues of the Jacobian matrix at some equilibrium points and observing the system response around these points. In addition, the existence of limit cycles and transmission instabilities has been explored. The vehicle state equations were generally found to display stable behavior within the normal operating area if some basic constraints are enforced. It was also found that the Sparse Traffic Optimization Algorithm can become unstable in a special case where subsequent vehicles of lower velocity quickly enter the wireless coverage area.

Chapter 7 - Simulation Results

This chapter will present the results of simulations performed on the vehicle model and the network of vehicles. First, an overview of the simulator will be provided. The next few sections will explain how certain system parameters and design decisions were determined. In the final section, the optimization algorithm will be simulated.

7.1 Simulator Overview

The simulations have been coded and run using MATLAB. The main simulation is described by the flow chart in Figure 7.1 and the MATLAB code given in Appendix A. The flow chart will now be described.

The initialization stage allows several simulation parameters to be adjusted. The number of vehicles, the number of EKF iterations, the initial conditions of the vehicle, and the quantities that are plotted are just some of the options that may be adjusted. After the simulation has initialized, the optimization algorithm may be executed. It is usually desirable not to perform optimization using the initial conditions but to wait until the EKF has completed a few cycles so it can converge.

The transmission stage changes the vehicle's gear ratio if its speed has entered the range for a certain gear. The preset speeds that designate certain gears may be overridden by adjusting a parameter in the initialization. The prediction stage uses the RK4 method to integrate the vehicle state equations. It is possible for multiple integrations may be performed in one EKF iteration. If the vehicle's speed changes sufficiently to allow it to change gears, it will not do so until the transmission stage of the next iteration.

The noise is specified in the next stage. The noise may be white or may be specified to be colored. The noise can also be disabled if it is desired to remove randomness from the results. If the noise is removed, the correction is equated to the prediction and the simulator iterates, otherwise, observation information is used in the correction stage. The recursive correction stage repeats until achieving the desired estimation error specified in the initialization.

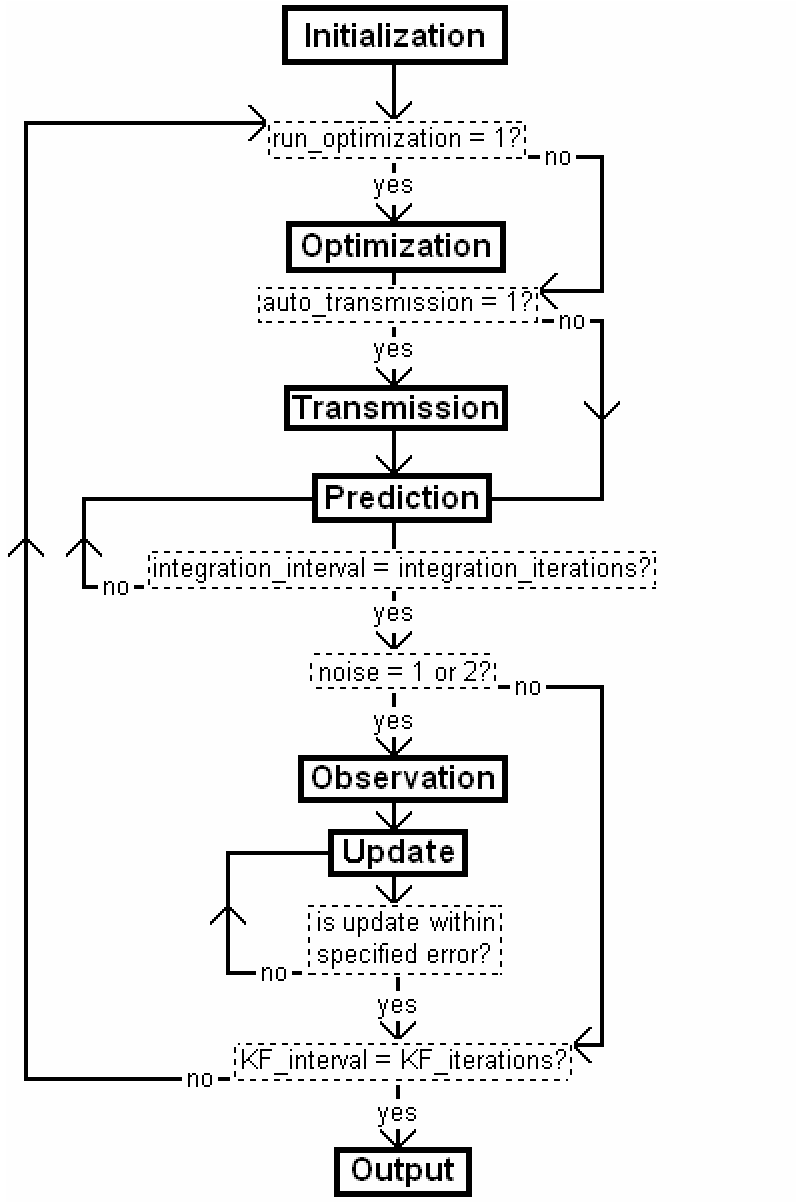


Figure 7.1: Simulator Operation Flow Chart.

7.2 Comparison of the Euler and 4th-order Runge-Kutta Methods on Integration of Vehicle Model

The performance of the Euler and RK4 methods are compared when solving for w_e. Since the vehicle velocity is directly related to w_e, the error present in its solution figures prominently into the accuracy of the vehicle velocity. It is important to keep the errors to a minimum because the optimization algorithm needs accurate velocity information to avoid collisions.

The Euler and RK4 methods are compared in terms of error when the vehicle is at maximum acceleration. The solution of w_e yielded by the RK4 method is shown in Figure 7.2 and the numerical integration results are given in Table 7.1. The data is displayed over three different time intervals to show the local and global errors. Since the model has no analytical solution, the error has been defined as the difference of the Euler from the RK4 results. Table 7.1 shows that the difference between the Euler and RK4 methods is quite small and indicates that the Euler method is sufficient for use in this collision avoidance application.

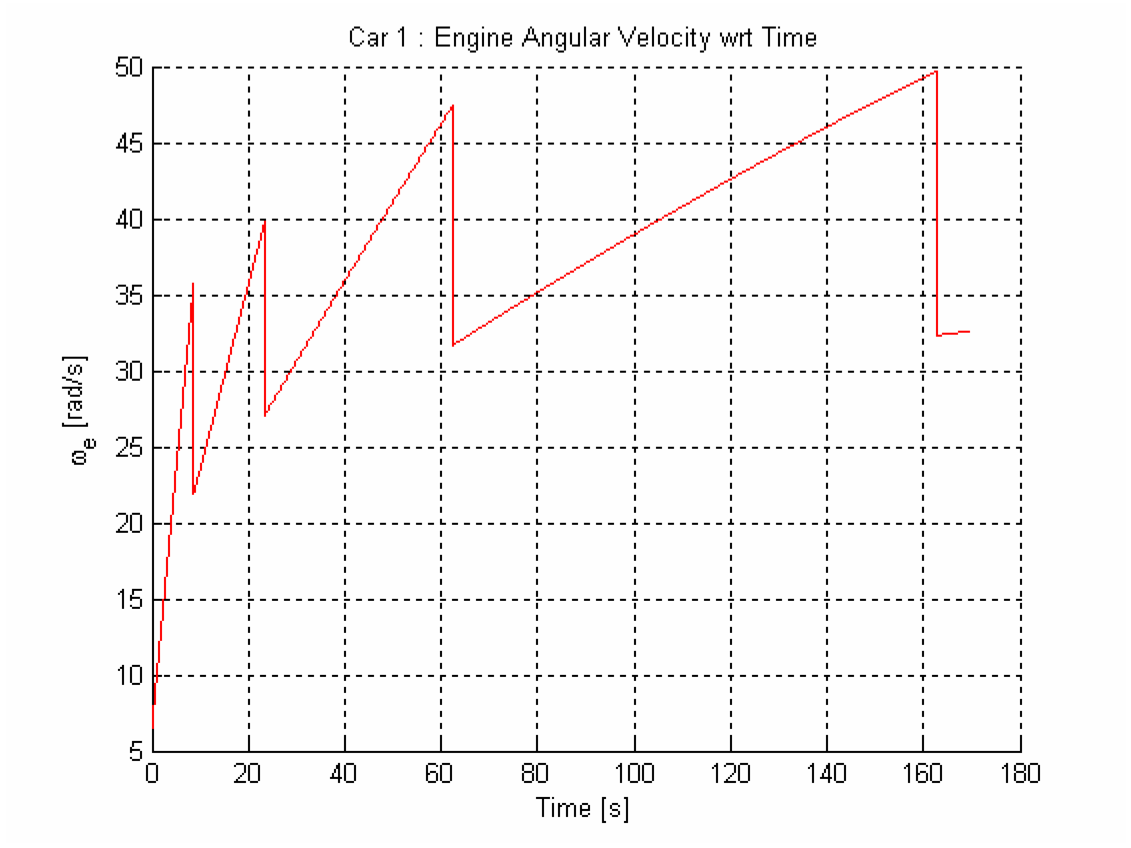
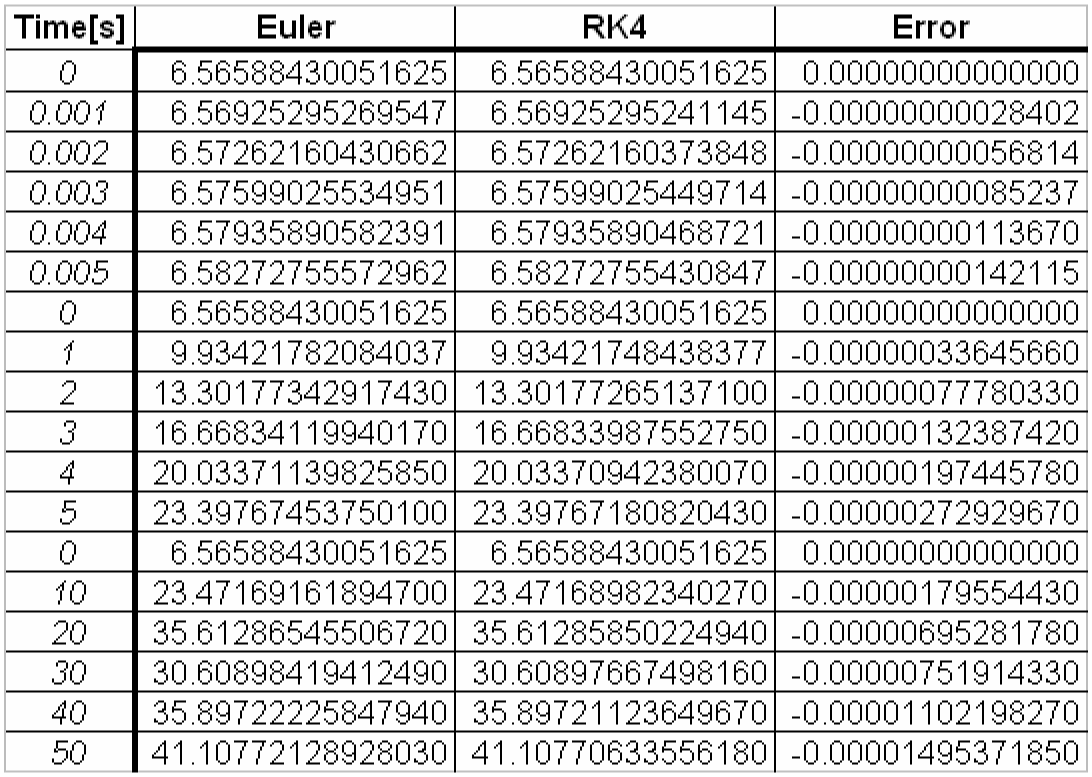


Figure 7.2: Engine Angular Velocity as a Function of Time Determined Using the RK4 Method. The initial value used for m_a *is 0.00472741455026.*

Table 7.1: Numerical Integration Results



Time[s]	Euler	RK4	Error
0	6.56588430051625	6.56588430051625	0.00000000000000
0.001	6.56925295269547	6.56925295241145	-0.00000000028402
0.002	6.57262160430662	6.57262160373848	-0.00000000056814
0.003	6.57599025534951	6.57599025449714	-0.00000000085237
0.004	6.57935890582391	6.57935890468721	-0.00000000113670
0.005	6.58272755572962	6.58272755430847	-0.00000000142115
0	6.56588430051625	6.56588430051625	0.00000000000000
1	9.93421782084037	9.93421748438377	-0.00000033645660
2	13.30177342917430	13.30177265137100	-0.00000077780330
3	16.66834119940170	16.66833987552750	-0.00000132387420
4	20.03371139825850	20.03370942380070	-0.00000197445780
5	23.39767453750100	23.39767180820430	-0.00000272929670
0	6.56588430051625	6.56588430051625	0.00000000000000
10	23.47169161894700	23.47168982340270	-0.00000179554430
20	35.61286545506720	35.61285850224940	-0.00000695281780
30	30.60898419412490	30.60897667498160	-0.00000751914330
40	35.89722225847940	35.89721123649670	-0.00001102198270
50	41.10772128928030	41.10770633556180	-0.00001495371850

The Euler and RK4 methods are now compared in terms of model breakdown. When integrating, smaller time steps yield more accurate results at the cost of more complexity. As the steps get larger, eventually the Euler and RK4 methods break down and produce highly inaccurate solutions. Since m_a usually breaks down before w_e, it is focused on in these simulations. Figures 7.3-7.5 show the solution of m_a using the Euler method. In Figure 7.3 with a time step of 1 ms, the Euler method yields a satisfactory solution. However, when the time step is doubled, it breaks down and displays oscillations at points where the gears shift as shown in Figures 7.4a and b. When the time step is increased further to 3 ms, the output shown in Figure 7.5 does not even resemble the original solution having large oscillations throughout the simulation.

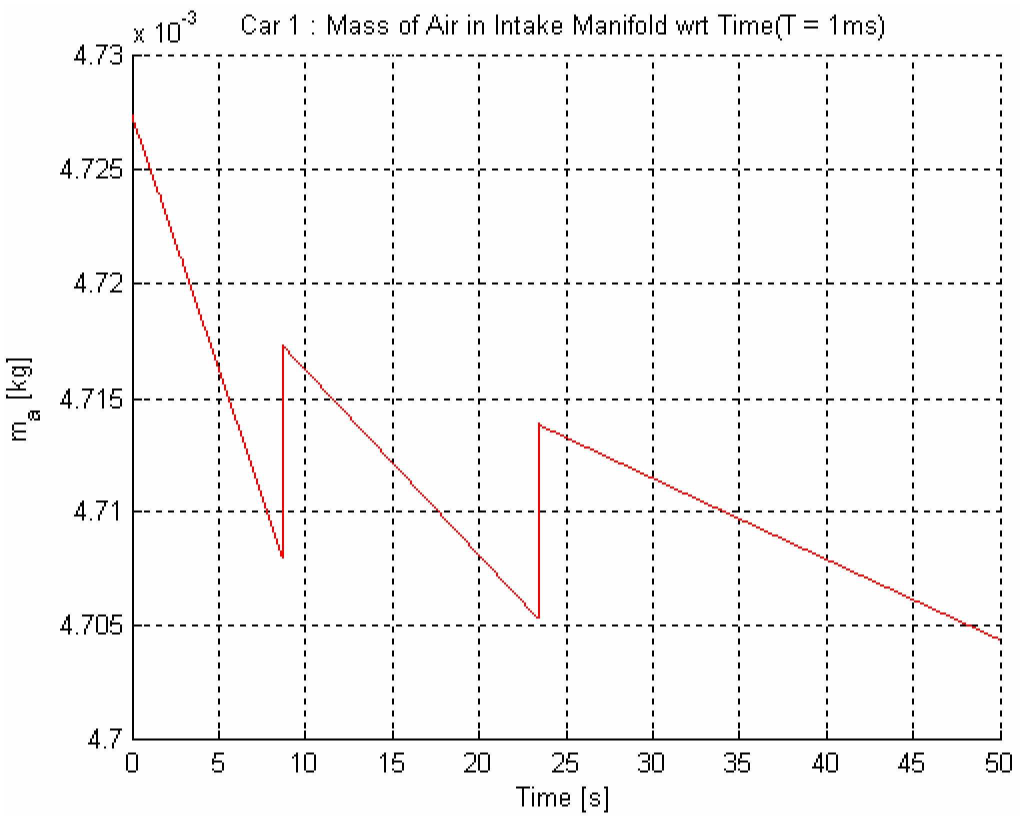


Figure 7.3: Mass of Air in Intake Manifold with Respect to Time for Step Size of 1 ms Determined using the Euler Method.

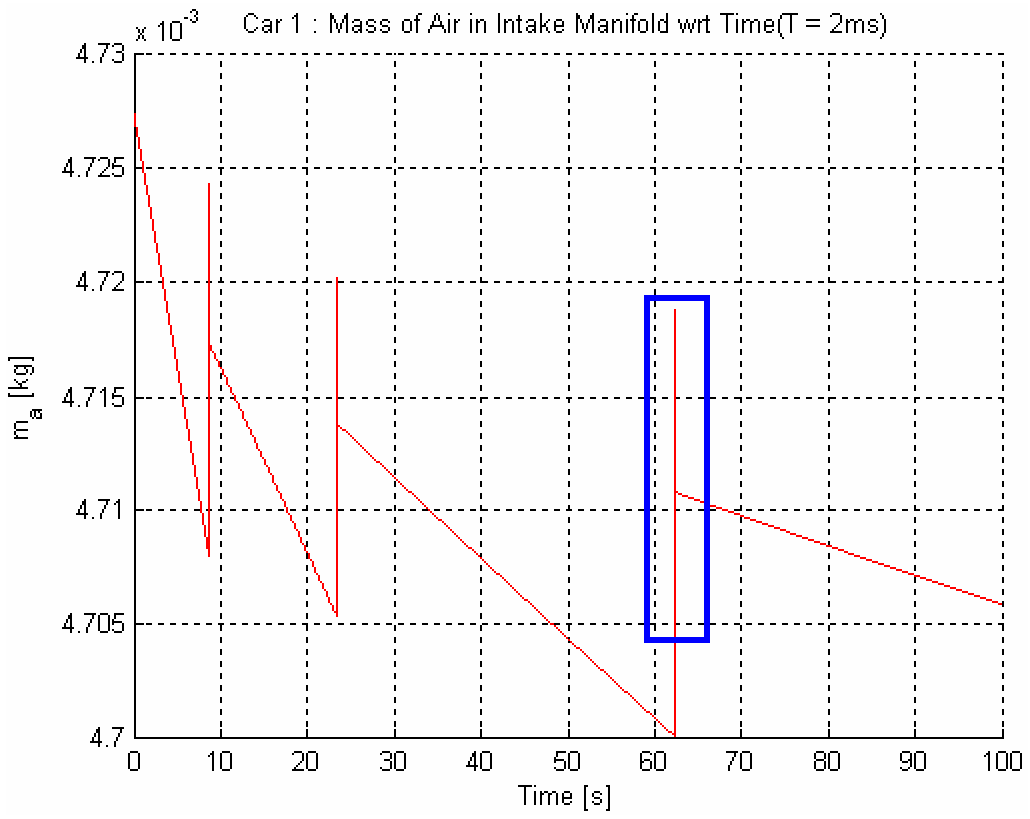


Figure 7.4a: Mass of Air in Intake Manifold with Respect to Time for Step Size of 2 ms Determined using the Euler Method.

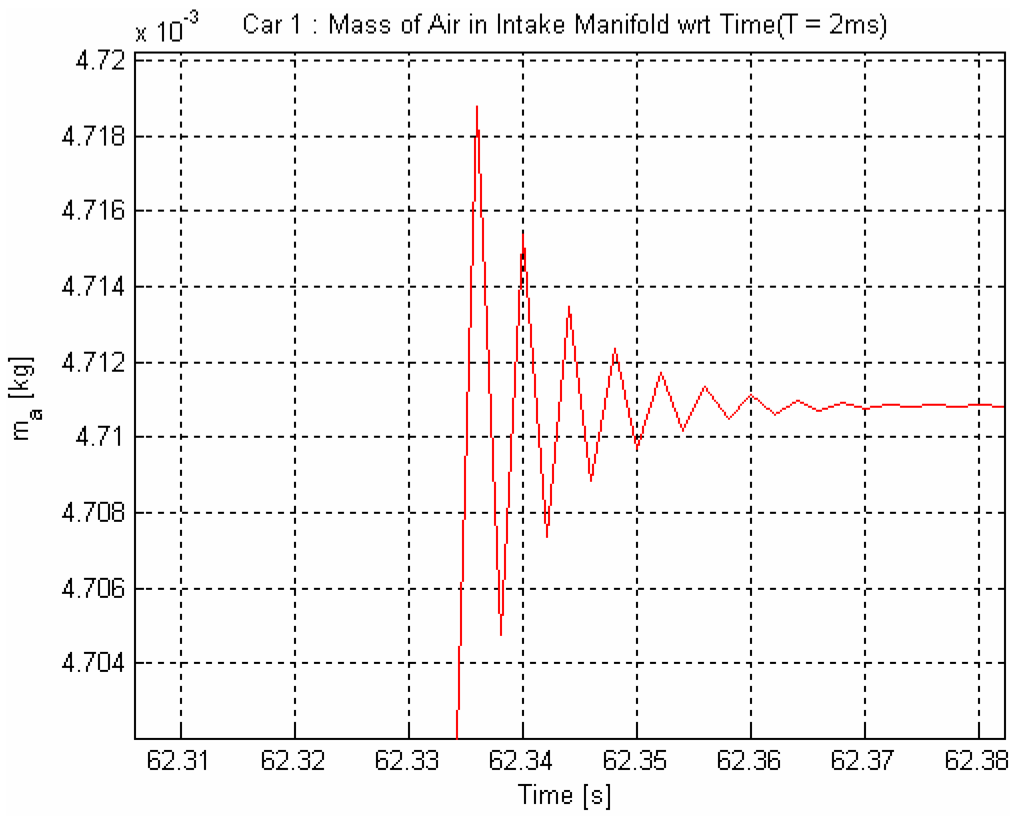


Figure 7.4b: Mass of Air in Intake Manifold with Respect to Time for Step Size of 2 ms Determined using the Euler Method. This plot is a magnification of the blue box in Figure 7.4a.

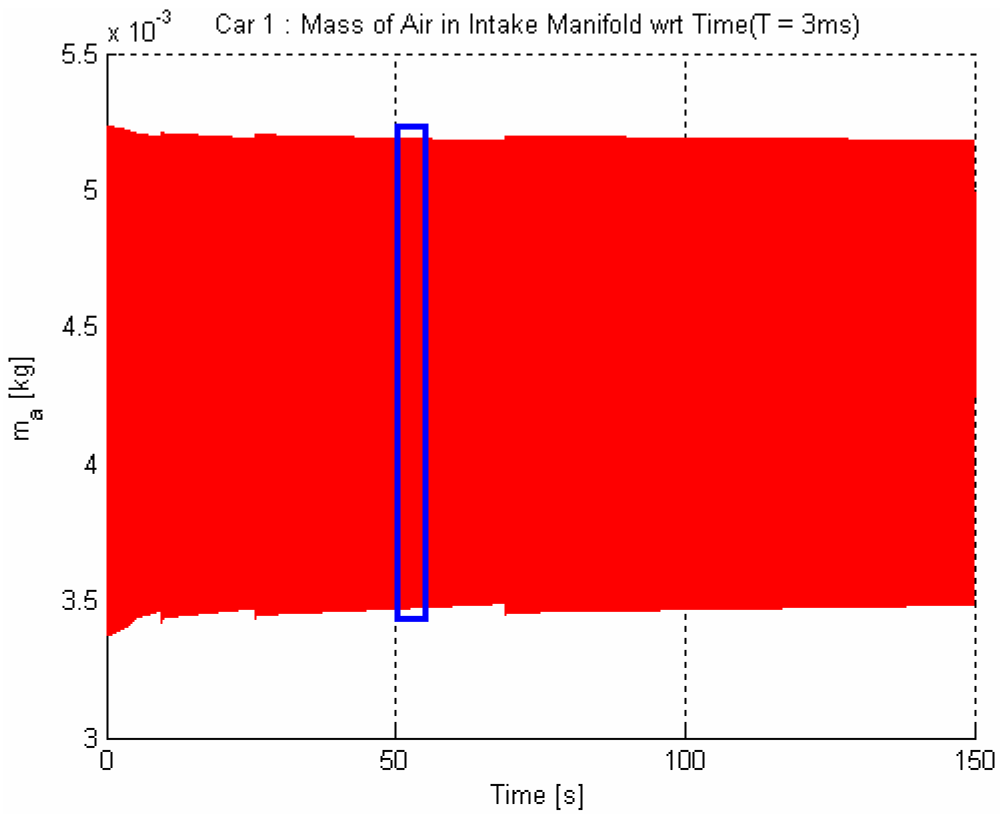


Figure 7.5a: Mass of Air in Intake Manifold with Respect to Time for Step Size of 3 ms Determined using the Euler Method.

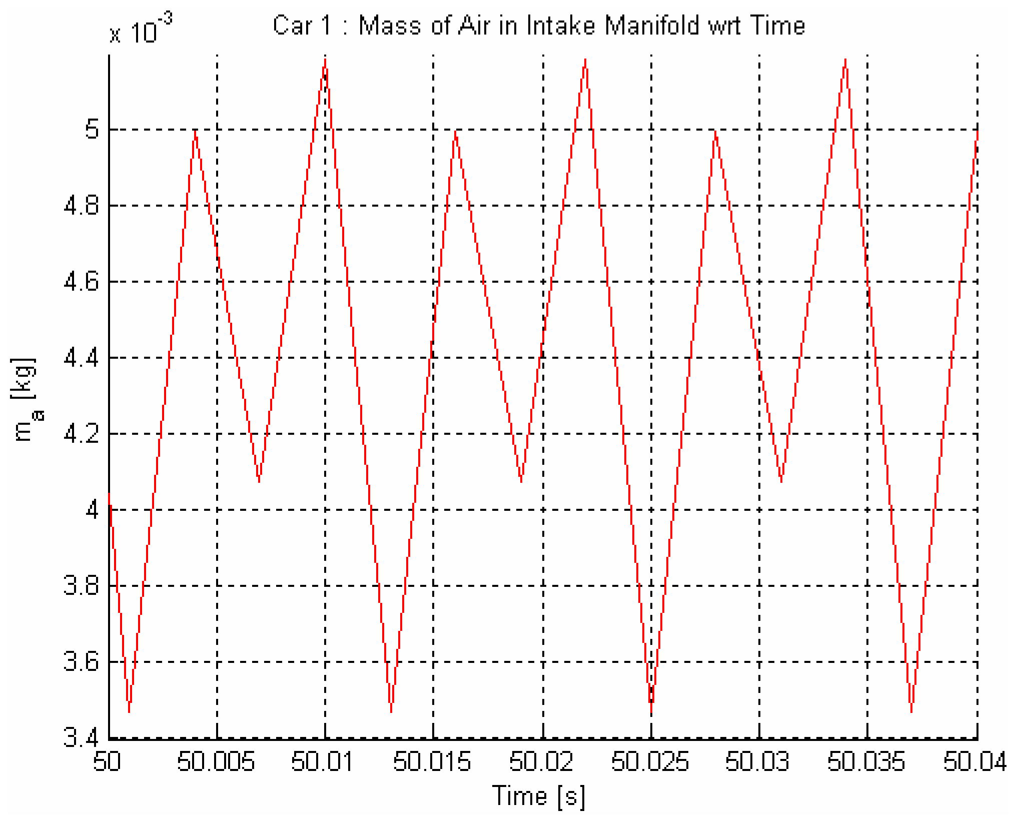


Figure 7.5b: Mass of Air in Intake Manifold with Respect to Time for Step Size of 3 ms Determined using the Euler Method. Plot is a magnification of the blue box shown in Figure 7.5a.

Now the breakdown of the RK4 method is tested. Figures 7.6 and 7.7 show the performance of the RK4 method with a step size of 3 and 4 ms, respectively. The plots clearly show that the RK4 method is better at maintaining model integrity than the Euler method since its breakdown at 4 ms is approximately twice as much as that of the Euler method.

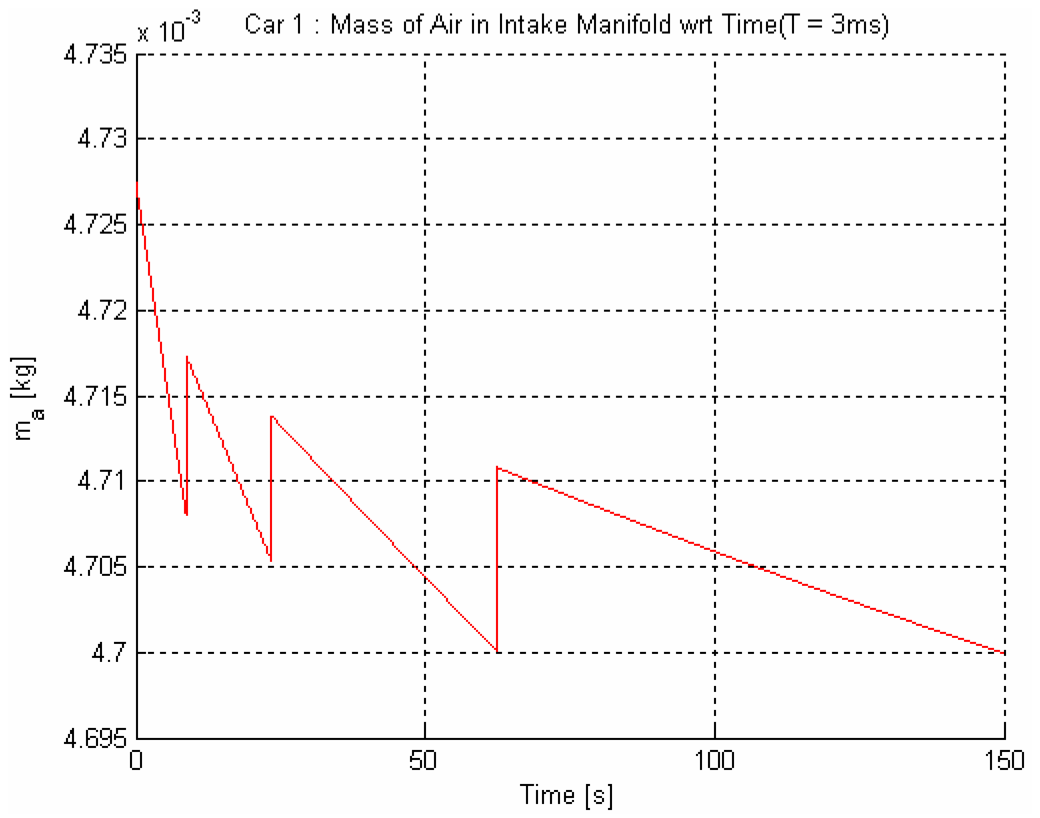


Figure 7.6: Mass of Air in Intake Manifold with Respect to Time for Step Size of 3 ms Determined using the RK4 Method.

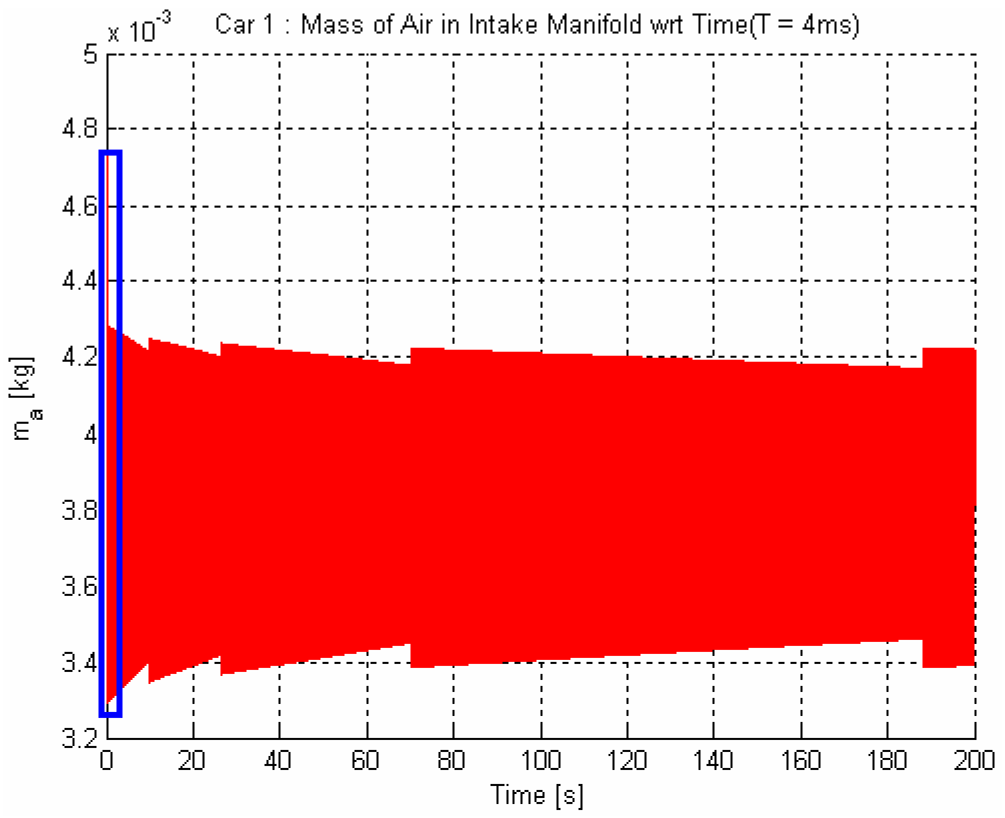


Figure 7.7a: Mass of Air in Intake Manifold with Respect to Time for Step Size of 4 ms Determined using the RK4 Method.

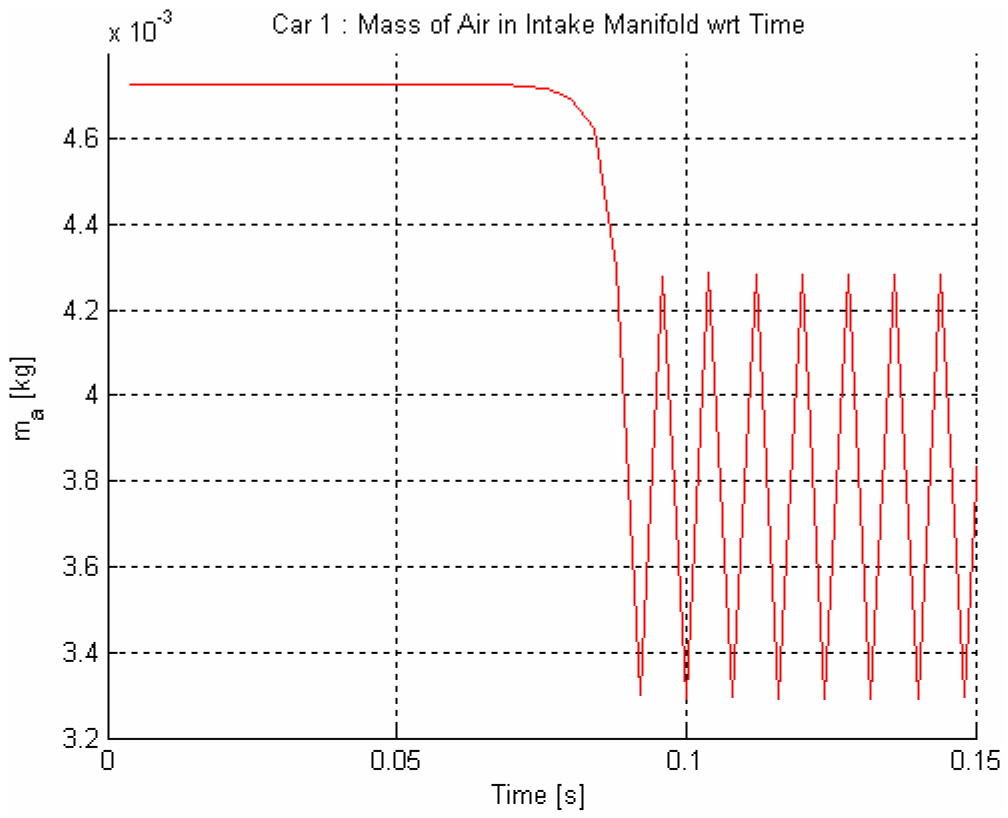


Figure 7.7b: Mass of Air in Intake Manifold with Respect to Time for Step Size of 4 ms Determined using the RK4 Method. Plot is a magnification of the blue box in Figure 7.7a.

Numerical analysis is a vast field and there are many other methods available. However, the RK4 method was chosen for use in this simulator due to its widespread popularity, low error, and resistance to model breakdown.

7.3 Determination of System Parameters

The EKF update frequency, the number of predictions in each EKF iteration, and the allowable correction error are determined through simulations. They are presented in this section along with an explanation of why they are specifically chosen.

The EKF update frequency is mainly determined by the EKF's convergence and practical implementation issues. When the EKF converges, the update stages of subsequent iterations usually finish after two or three iterations. Ensuring that convergence occurs as quickly as possible greatly speeds up processing. Although high update frequencies help the EKF converge, it also makes it necessary to use more complex and expensive hardware to do the calculations. Additionally, if the update frequency is too fast, the wireless network may not be capable of transmitting observations on time. Since this system is designed to be implemented in the near future, the goal is to find a suitable frequency that current hardware can implement while ensuring quick convergence. After conducting numerous simulations, an EKF update frequency of 1 kHz is chosen. Although lower frequencies are suitable for most driving conditions, this frequency ensures satisfactory results for all conditions. An adaptive frequency would be better for a real implementation but was not simulated here. With a 1 kHz frequency, the system is able to tolerate 1 ms of delay. Since WAVE has a typical delay of 50 ms, more research needs to be done to reduce the update frequency before implementing a real system.

The integration frequency of the prediction stage is chosen to be 1 kHz. This decision is mainly based on the integration frequency causing the RK4 method to break down and the EKF update frequency. Since each EKF iteration must execute at least one prediction, the choice of the EKF update frequency mainly decided the integration frequency. Although the RK4 method was shown to break down using 4 ms time steps in Figure 7.7, under certain conditions, it has been observed to break down earlier as well.

The allowable EKF correction error is mainly determined by the properties of m_a since it is more susceptible to errors than the other state variables. Since the mass of air is generally on the order of 10^{-3}, the correction error must be a few orders of magnitude lower than this for the EKF to yield adequate, consistent results. The value used in simulations for the allowable correction error is 10^{-6}.

7.4 Determination of the Brake Constant of Proportionality

The brake constant of proportionality, K_b, is determined from the deceleration characteristics of the vehicle. Figure 7.8 shows the deceleration of the vehicle during maximum braking.

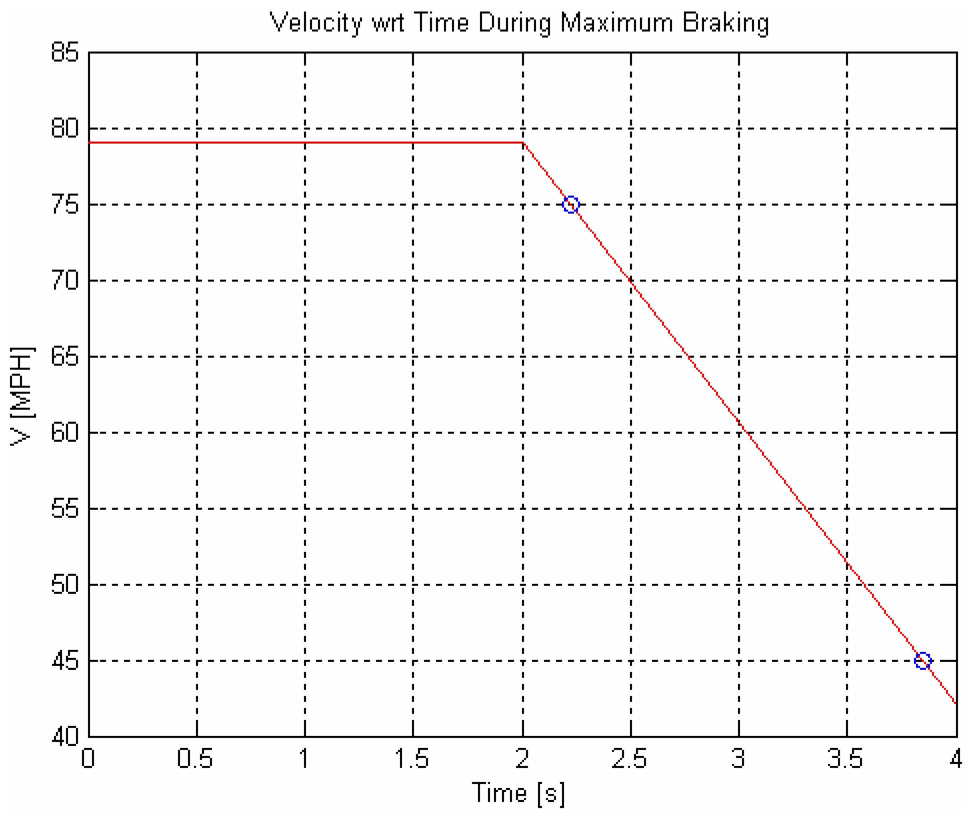


Figure 7.8: Vehicle Velocity with Respect to Time during Maximum Braking. The blue sample points have been added to show the samples used to determine the slope. This plot is an edited form of Figure 20 from [1].

Two sample points at (2.3, 75) and (3.8, 45) are used to solve for the deceleration which is found to be $-\left(\frac{45-75}{3.8-2.3}\right)=\frac{30}{1.5}=20\frac{MPH}{s}=8.9815\frac{m}{s^2}$. The value for K_b is found by conducting a set of simulations where the brakes are fully applied, K_b is varied, and the deceleration is observed. When it approximates the value found from Figure 7.8 to within 1% error, then that value of K_b is used. Figure 7.9 shows that a deceleration rate of 20.07 is produced when $K_b = 300\frac{N\cdot m}{\deg rees}$. The deceleration is within the specified error bounds so it is used as the value for the brake constant of proportionality.

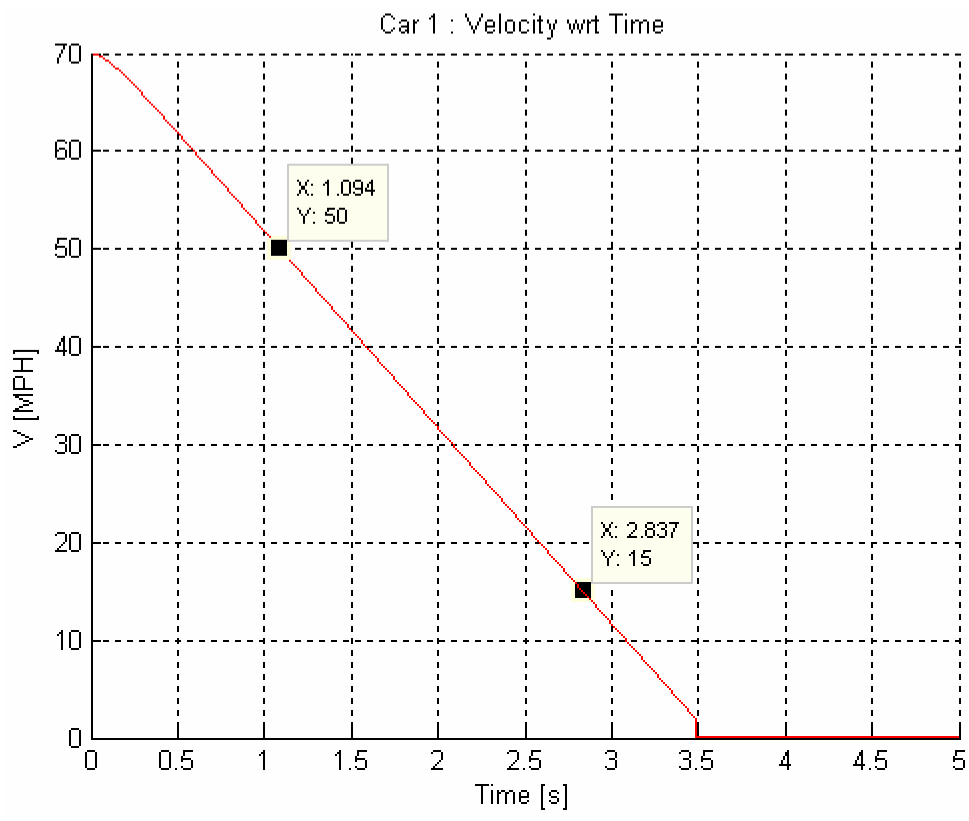


Figure 7.9: Simulated Vehicle Velocity with Respect to Time during Maximum Braking for $K_b = 300\frac{N \cdot m}{\deg rees}$.

7.5 Determination of Brake Angle as a Function of Deceleration

In this section, the findings of the last section are extended and an empirical relationship between the brake angle and deceleration is derived. In Figure 7.9, notice how the deceleration rate is approximately constant throughout the simulation. This indicates that it is independent of the velocity and only dependent on the brake angle. Although losses from drag are modeled, they are usually very small compared to the brake torque. By simulating the vehicle under a variety of brake angles, the extent of the deceleration's dependence on variables other than the brake angle is investigated.

Several simulations are conducted to find the resulting deceleration under multiple brake angles. The deceleration is measured by finding the slope of the velocity from the 50 and

15 MPH points. An empirical estimate is made using least-squares regression and is shown along with the estimation error in Figure 7.10.

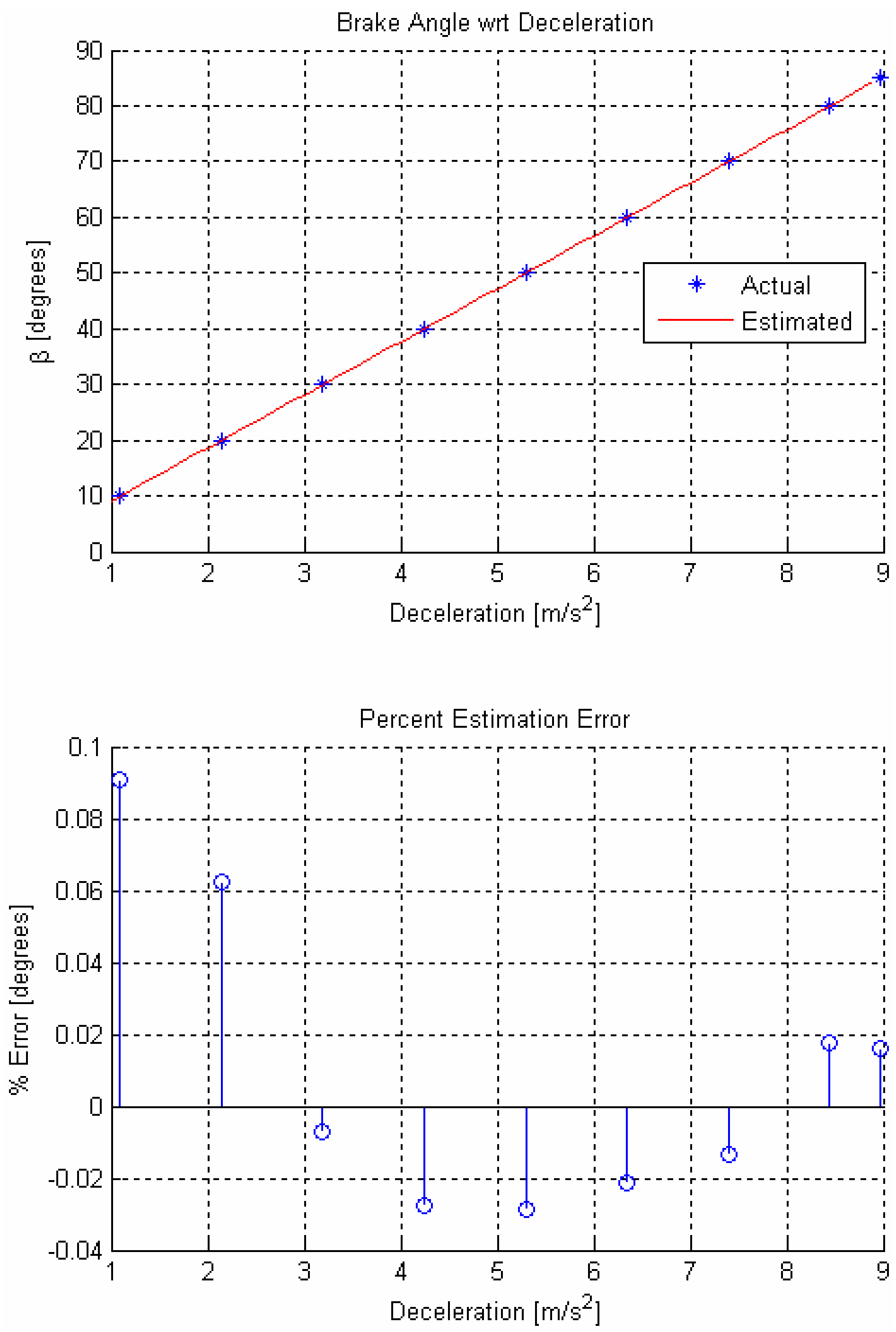


Figure 7.10: Estimate of Relationship Between Brake Angle and Deceleration using Least-Squares Regression.

Figure 7.10 is arrived at by varying the brake angle in increments of 10 degrees and observing the deceleration as the dependent variable. The deceleration is then plotted on the x-axis and the brake angle on the y-axis. The deceleration resulting from maximum braking is also included. The least-squares estimate is given by

$$\hat{b} = 9.511\text{d} - 0.3744 \,, \qquad \textbf{(111)}$$

where $\hat{b}$ is in degrees and d is the deceleration rate in m/s^2. As Figure 7.10 shows, the linear estimate is very accurate with all errors below 0.1%. Eventually, as the brake angle approaches zero, the vehicle's deceleration will primarily be caused by forces other than the brakes and this estimate will no longer hold.

7.6 Linear Approximation of Vehicle Acceleration

The procedure to determine the throttle angle in terms of the acceleration is similar to finding the brake angle in terms of the deceleration but is considerably more complex. Since the vehicle's ability to accelerate is much weaker than its ability to decelerate, drag forces and other losses are not always negligible. The gear's influence on acceleration means the function is multivariate. And to add further difficulties, the normalized throttle characteristics indicate that acceleration is nonlinearly related to throttle angle. In order to avoid nonlinearities introduced by the throttle, the optimization algorithm only uses the maximum throttle angle of 85^0 when accelerating. This allows the acceleration to be approximated as a piecewise function.

The simulator is run with the vehicle accelerating from idle to 60 MPH at maximum acceleration. The acceleration at higher speeds is not considered since those speeds are not used in simulations. For each gear, an estimate of the acceleration is made using least-squares regression. Due to increased estimation error caused by nonlinearities, the fourth gear is treated as two separate gears. The regression estimates along with the associated errors are shown in Figure 7.11 and listed in Table 7.2.

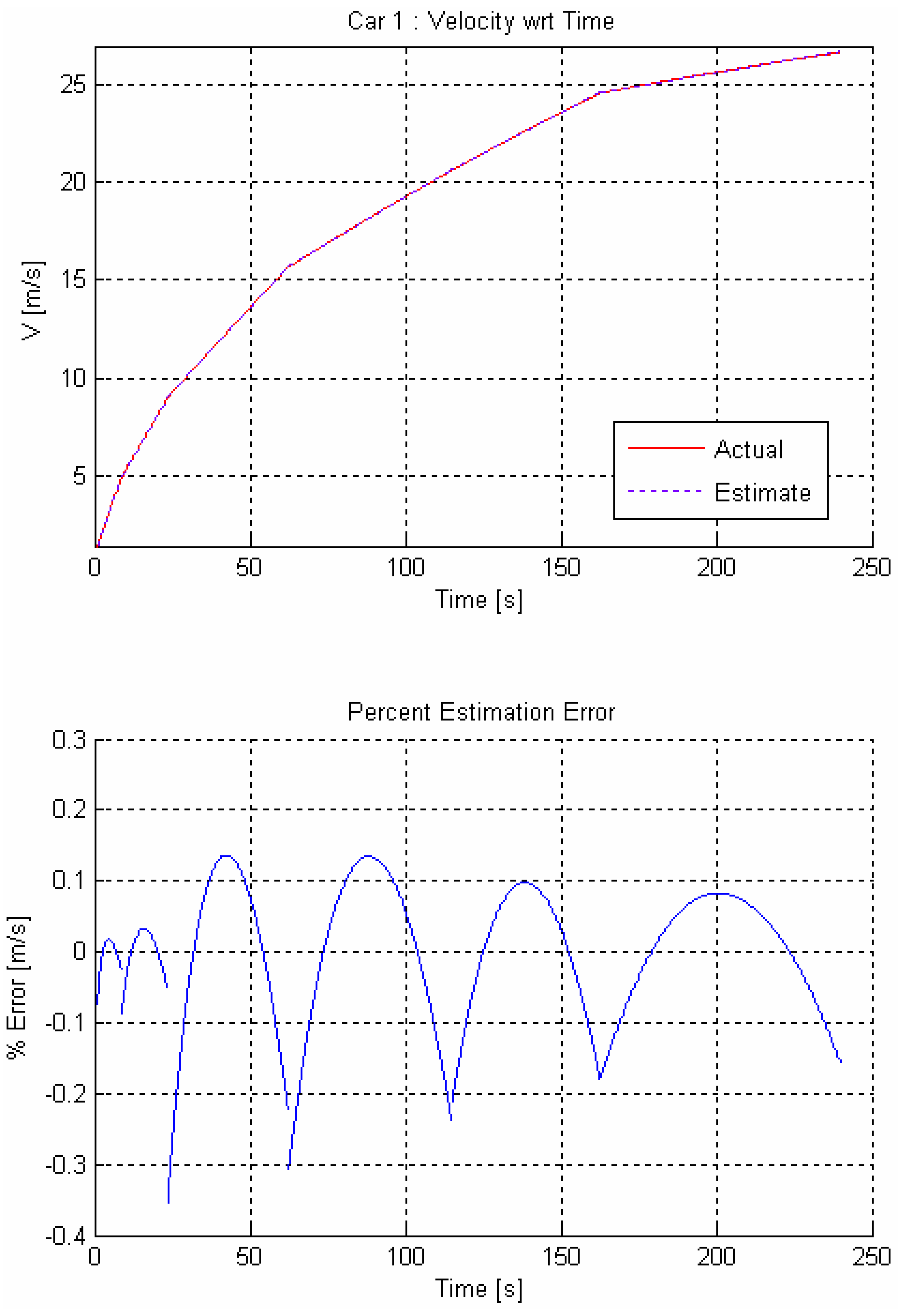


Figure 7.11: Estimate of Relationship Between Acceleration and Gear using Least-Squares Regression.

Table 7.2: Acceleration Estimates with Respect to Gear and Velocity Range.

Gear	Velocity [m/s]	Velocity [MPH]	Acceleration [m/s^2]
1	[0-4.92]	[0-11]	0.4624
2	(4.92-8.94]	(11-20]	0.2728
3	(8.94-15.6]	(20-35]	0.1724
4	(15.6-20.5]	(35-45]	0.09438
4	(20.5-24.6]	(45-55]	0.08311
5	(24.6-26.8]	(55-60]	0.02717

The low errors in Figure 7.11 indicate that the acceleration estimates are very accurate. The size of the errors of the 4th and 5th curves also justifies the decision to model the 4th gear with two estimates.

7.7 Optimization Algorithm Results

The performance of the optimization algorithm is evaluated by comparing it against conventional intersection control devices. The delay of this algorithm is compared to the delay introduced by a stop sign and a typical traffic signal. The simulations occur at a four approach, single lane intersection. The width of each lane is set to 3.6 m. The x_1-axis is defined as the east-west direction where east is in the positive direction. And the x_2- axis is defined as the north-south direction where north is in the positive direction. The results are presented in this section.

7.7.1 Comparison of Stop Sign to Sparse Traffic Optimization Algorithm

The stop sign is compared to the optimization algorithm in five simulations. The purpose of these simulations is to quantify the delay caused by stop signs even when other vehicles are not present. A vehicle is simulated approaching an intersection, stopping, and then accelerating to cross straight-through. The time to cross using a stop sign is compared to the time to cross using the optimization algorithm developed here which does not require braking if other vehicles are not present. The first simulates the optimization at a rural intersection with a speed limit of 50-MPH on each approach. The subsequent simulations model the stop sign controlled intersection with speed limits of 46, 40, 35, and 30-MPH. The vehicles use a braking angle of 37.6696^0 to decelerate, a throttle angle of 85^0 to accelerate, and have a radius parameter of 2.5 m. The vehicles are initially traveling at the speed limit as they approach the intersection. Table 7.3 and Figure 7.12 show the amount of additional time needed to cross the intersection because of the stop sign.

7.7.2 Comparison of Traffic Signal to Sparse Traffic Optimization Algorithm

The same scenario used to show the delay caused by a stop sign is utilized for a traffic signal as well. Determining this delay is more difficult since signals are often customized for specific intersections. A common configuration on low traffic roads is a fully-actuated two-phase traffic signal. It functions by either displaying green for east-west or north-south traffic and changing only when vehicles are detected. In the case that the vehicle arrives from an approach that is already signaling green, the vehicle is not delayed at all. However, if the vehicle is coming from the approach that is signaling red, the vehicle must first come to a complete stop, wait for the signal to change from green, to yellow, to red, then green for the approach the vehicle is coming from. The delay in this case is calculated by adding the delay caused by stopping to the delay caused by waiting through the signal's yellow time.

A minimum yellow interval of 4 seconds is recommended for single lane intersections with 30-MPH speed limits on the approaches and 5 seconds for 50-MPH limits according to the method presented by Garber in [13]. Yellow intervals of 4.00, 4.25, 4.50, 4.75, and 5.00 s have been used for the 30, 35, 40, 46, and 50-MPH simulations, respectively. Therefore, the total delay may be computed by summing the delay caused by the stop sign at a certain speed (since this accounts for the stopping delay) with the delay caused by the yellow interval and dividing by two. The calculated delays are shown in Table 7.3 and Figure 7.12. Note that this calculation is only valid for very sparse traffic. It is assumed that the last crossing of the intersection by another vehicle before the arrival of the vehicle considered here happened sufficiently long ago that this vehicle does not need to wait for the signal to stay green on the other approach. If another vehicle just crossed the intersection from another phase before this vehicle stopped at the signal, the delay caused by waiting for the other phase to change from green to yellow would significantly increase the delay of the traffic signal.

Table 7.3: Average Additional Delay Caused By Stop Signs and Traffic Signals Under Sparse Traffic Conditions in Comparison to the Sparse Traffic Optimization Algorithm.

Velocity [MPH]	Delay for Stop Sign [s]	Delay for Traffic Signal [s]
30	6.38	5.19
35	6.79	5.52
40	7.16	5.83
46	7.51	6.13
50	7.86	6.43

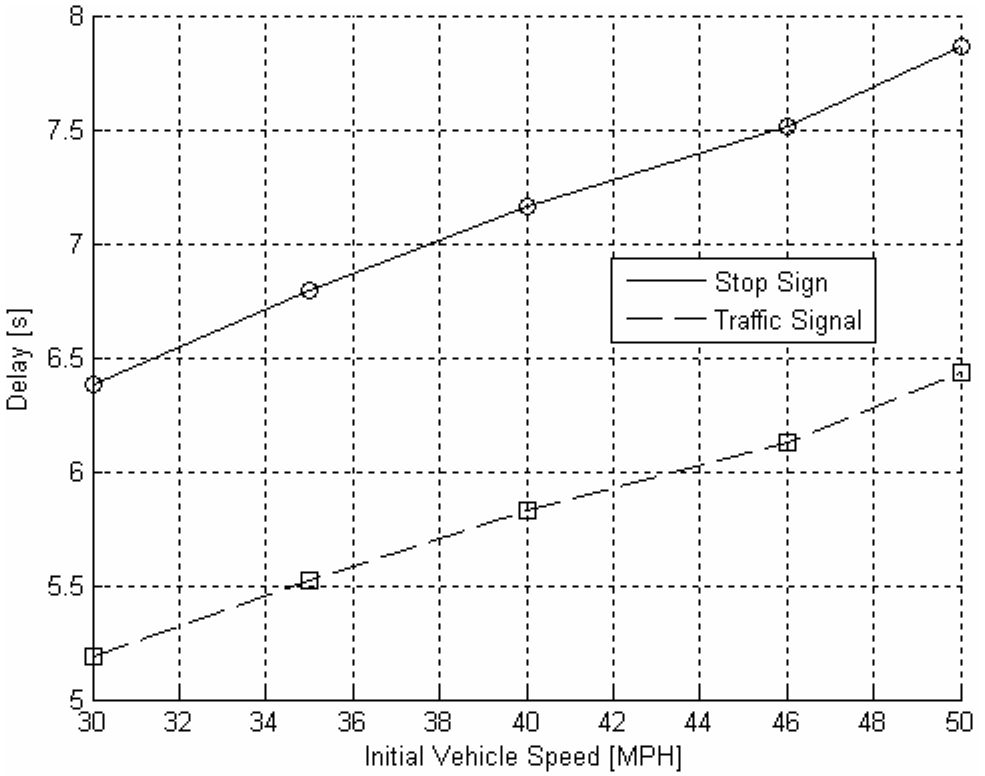


Figure 7.12: Average Additional Delay Caused By Stop Signs and Traffic Signals Under Sparse Traffic Conditions in Comparison to the Sparse Traffic Optimization Algorithm.

7.7.3 Comparison of Traffic Signal to Dense Traffic Optimization Algorithm

The traffic signal is compared to the Dense Traffic Optimization Algorithm by analyzing the number of vehicles that traverse the intersection in one green cycle, the intersection capacity, and the saturation flow rates for each control method. Assume that an intersection has an infinite number of vehicles stopped at each approach, the length of each vehicle is 5 m, and there is 1.5 m of separation between adjacent vehicles. Also assume that the maximum allowable speed on the road is 30 MPH, the vehicles linearly accelerate to this speed in 3 seconds, and all vehicles travel straight. In this section, this scenario is used to determine the effects of human driver delays at traffic signal controlled intersections on the number of vehicles that are able to cross in one green cycle, the intersection capacity, and the saturation flow rate.

The number of human driven vehicles that can cross a traffic controlled intersection in one green cycle is given by

$$n_{TS} = 2\left\lfloor \frac{\tfrac{1}{2}at_1^2 + v\left(t_{green} - t_1 - t_2\right) - 2\left(w_{lane}\right) - l_{vehicle}}{l_{vehicle} + d_{separation}} \right\rfloor, \tag{112}$$

where a is the acceleration rate, t_1 is the time spent accelerating, v is the maximum allowable speed, t_{green} is the amount of combined green and yellow time, t_2 is the time the nth vehicle delays before accelerating, w_{lane} is the lane width, $l_{vehicle}$ is the length of each vehicle, and $d_{separation}$ is the distance between each successive vehicle. t_{green} is used as the independent variable and t_2 is determined by considering the events at a traffic signal. Human drivers delay accelerating when traffic signals turn green because they need time to perceive signal changes and to allow the preceding vehicle to create sufficient headway. According to Garber, a value of 1.0 second is adequate for the driver perception delay of the first vehicle in a lane [13]. For the separation delay, it is assumed that each successive vehicle waits 1 second to accelerate after the preceding vehicle does.

Using these assumptions, t_2 is assigned a value of n_{TS} seconds. Now, (112) is simplified and the final form of the equation is given by

$$n_{TS} = 2\left\lfloor \frac{\frac{1}{2}at_1^2 + v\left(t_{green} - t_1\right) - 2\left(w_{lane}\right) - l_{vehicle}}{v + d_{separation} + l_{vehicle}} \right\rfloor. \tag{113}$$

This calculation is repeated for the Dense Traffic Optimization Algorithm. The algorithm eliminates perception delays by planning vehicle movements ahead of time and can eliminate separation delay by using automation to maintain tight spacing. By reducing separation delay, the flow rate of vehicles at the intersection can be increased. However, at the same time, safety is reduced as there is a low margin for error. A separation delay of 0.112 s is introduced which results in a 3 m safety margin between adjacent vehicles at the speed limit of 30 MPH. Using (112) as a basis and assuming that there are no perception delays, the number of vehicles that can cross the intersection in one green cycle using the Dense Traffic Optimization Algorithm is given by

$$n_O = 2\left\lfloor \frac{\frac{1}{2}at_1^2 + v\left(t_{green} - t_1 - 0.112\right) - 2\left(w_{lane}\right) - l_{vehicle}}{0.112v + d_{separation} + l_{vehicle}} \right\rfloor. \tag{114}$$

Equations (113) and (114) are used to compare the performance of the traffic signal and Dense Traffic Optimization Algorithm. Figure 7.13 shows the number of vehicles that are able to cross the intersection from one approach as the green time is varied. The plot clearly shows that the separation delay required by human drivers greatly reduces the flow rate of an intersection. It demonstrates that the performance of the Dense Traffic Optimization Algorithm is slightly more than 2 times greater than that of the traffic signal.

The DTOA is now compared to the traffic signal by analyzing the maximum capacity of an intersection under each type of control. Using the same assumptions as the preceding simulation, the intersection capacity is determined here by analyzing a two-phase signal with 60 s green times. The capacity of the traffic signal controlled intersection is determined to be 4,560 vehicles per hour and the capacity of the Dense Traffic Optimization Algorithm controlled intersection is 11,520 vehicles per hour.

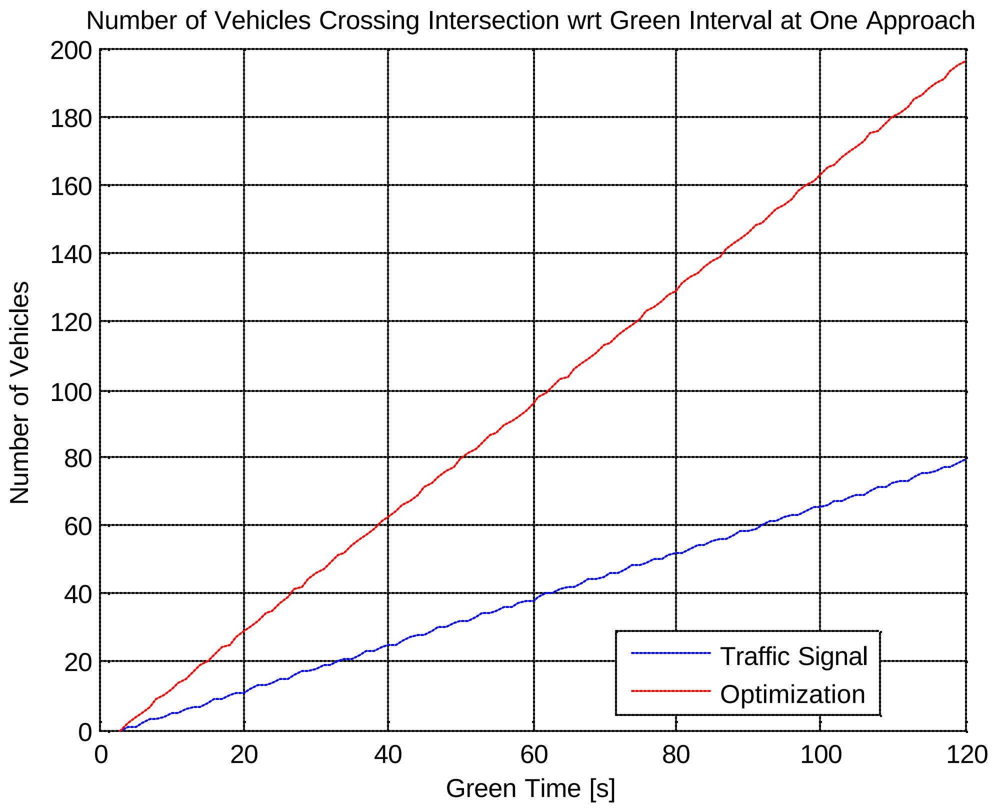


Figure 7.13: Number of Vehicles Able to Cross an Intersection From One Approach in One Green Cycle with Respect to the Length of One Green Cycle.

The traffic signal and Dense Traffic Optimization Algorithm are now compared in terms of saturation flow rate, which is reached when all vehicles in the intersection are at the maximum allowable speed. Using the same assumptions as before, when the 4th vehicle enters the intersection, all preceding vehicles are at the maximum speed while crossing the intersection and the saturation flow rate is reached. With a traffic signal, this rate is reached 7.000 seconds after the signal turns green due to perception, separation, and vehicle acceleration delays. Saturation is reached in only 3.224 seconds with the Dense Traffic Optimization Algorithm yielding better performance. The saturation flow rate can be found by calculating the amount of time subsequent vehicles cross a fixed point. The Dense Traffic Optimization Algorithm maintains a spacing of 3 m between vehicles during movement. At 30 MPH, this means vehicles cross a point in intervals of 0.59657 seconds. In the traffic signal case, the additional separation delay means vehicles cross every 1.4847 seconds. The resulting saturation flow rates are displayed in Figure 7.14. The plot shows that the Dense Traffic Optimization Algorithm allows a flow rate of more than 2 times that of the traffic signal and reaches saturation in less than half the time.

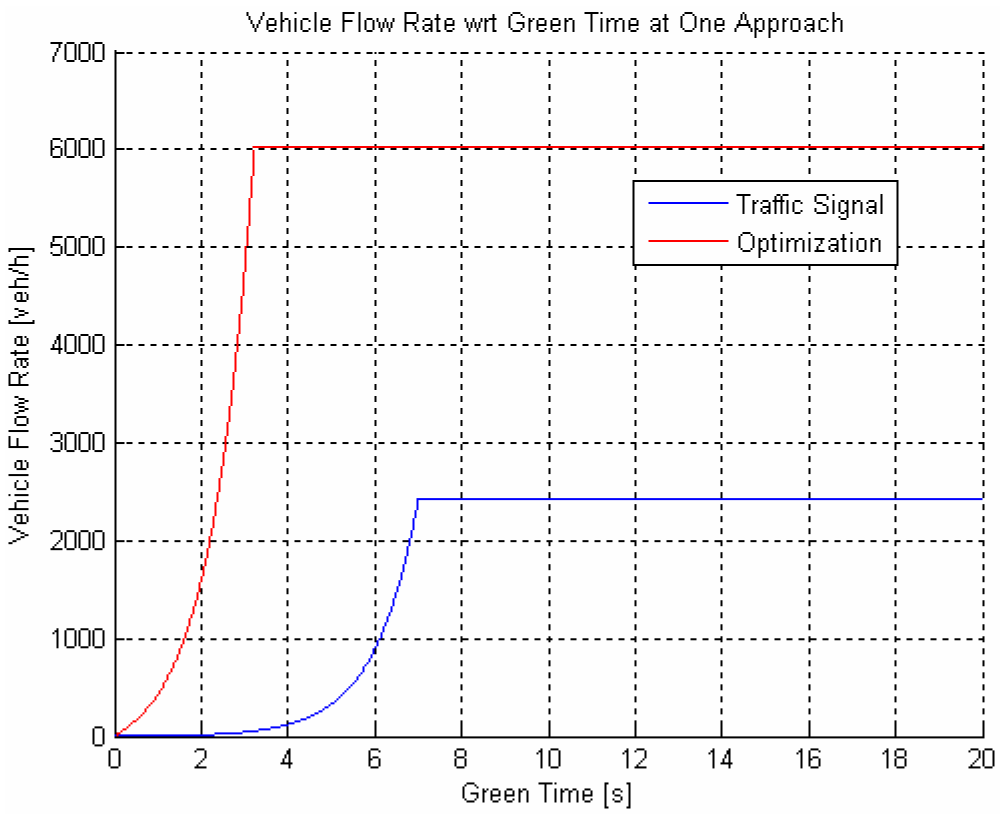


Figure 7.14: Vehicle Flow Rate From One Approach With Respect to Length of Time a Signal has Shown Green.

Chapter 8 - Conclusions and Future Research

This thesis has proposed a novel collision avoidance system that adapts to changing traffic conditions. The system applies a constrained, coupled EKF to estimate the states of a realistic, nonlinear vehicle model. The estimates are transmitted wirelessly to other vehicles using the new WAVE standards. Vehicles use these estimates to optimize their future controls by minimizing the average vehicle delay. Results show that this cooperative intersection collision avoidance system outperforms today's common intersection control devices. The contributions of this research to the field of collision avoidance are explained followed by some avenues for future research.

The expanded nonlinear vehicle model improves on idealized linear models by accounting for internal vehicle characteristics and a variety of forces acting on the vehicle. The model's states include the vehicle position, mass of air in the intake manifold, engine angular velocity, and the brake torque. By modeling the internal characteristics, the model more closely resembles the operation of real vehicles than idealized linear models. The ability of the coupled EKF to predict the trajectories of vehicles approaching an intersection was shown to be sufficient for the optimization. Additionally, decentralized operation achieved by coupling vehicle EKFs provides system reliability. The vehicle network provides the basis for cooperative collision avoidance by enabling vehicles to communicate their information to other vehicles. This cooperative system is superior to vehicular-based radar and optical technologies since it can function over longer distances and does not require line of sight.

Future Research

The work presented here opens several research opportunities in related areas. Most vehicular optimization research focuses on minimizing time or delay. However, this optimization algorithm could be modified to account for fuel efficiency as well. Since the state space model enables the amount of fuel to be inferred from the mass of air in the intake manifold, the fuel efficiency can be studied. With the growth of green engineering and the push toward more efficient technologies also comes interest in hybrid vehicles. The nonlinear vehicle model presented here could be extended to model hybrids.

There are certain aspects of the EKF that are in need of improvement before being implemented. The EKF analysis performed in this thesis assumes an unconstrained filter. Future research should investigate a different method for computing the covariance matrices to reflect these constraints.

In this research, it has been assumed that observation information is instantly transmitted and received by other vehicles. One difficult problem that has yet to be solved is determining how the optimization algorithm handles delayed EKF observations. The current EKF formulation requires an update frequency of 1 kHz. In order to reliably use a WAVE network to transmit observations, the EKF needs to be modified to support an update frequency of 20 Hz.

The current optimization algorithm uses an arbitrary, safe threshold to decide when to switch from sparse to dense mode. Investigating the capacity of a fully-meshed WAVE network can provide information detailing when it is appropriate to switch based on communication and processing delay as more vehicles join the network. The optimization algorithm is currently designed for four-way intersections. In the future, the algorithm could be applied to traffic circles. In order for this system to control traffic at a normal intersection, the optimization algorithm needs to be modified to account for any pedestrians. This could be accomplished by fusing optical camera and cooperative communicated information when optimizing vehicle controls.

The optimization algorithm currently does not account for weather. During heavy precipitation, braking distances and vehicle separation need to be increased to prevent collisions. Additionally, the magnitude of the vehicle acceleration needs to be limited so that traction with the ground is not lost. Limitations in wireless communication distances also should be considered.

The decentralized, cooperative collision avoidance system presented in this thesis is generally able to be implemented with today's technology. However, more work is still needed to solve some of the remaining technical issues and ensure high reliability and low cost.

Chapter 9 - References

[1] Hedrick, J.K., McMahon, D.H., and Swaroop, D., "Vehicle Modeling and Control for Automated Highway Systems". California PATH Program, UC Berkeley, November 1993.

[2] Cho, D. and Hedrick, J.K., "Automotive Powertrain Modeling for Control", Transactions of the ASME: *Journal of Dynamic Systems, Measurement, and Control*, Vol. 111, No. 4, December 1989a.

[3] Williams, D.R., "Earth Fact Sheet". *NASA*. April 19 2007. Accessed February 13 2008 <http://nssdc.gsfc.nasa.gov/planetary/factsheet/earthfact.html> 2007.

[4] Mendel, J. M., *Lessons in Estimation Theory for Signal Processing, Communications, and Control*. Prentice Hall, Inc., Englewood Cliffs, NJ 1995.

[5] Analytic Sciences Corporation Technical Staff and Gelb, A., *Applied Optimal Estimation.* M.I.T. Press, Cambridge, MA, 1974.

[6] Mili, L., Robust Estimation and Filtering, Ch 8 Class Notes, 2007.

[7] Grewal, M.S. and Andrews, A.P., *Kalman Filtering: Theory and Practice*. Prentice Hall, Inc., Englewood Cliffs, NJ, 1993.

[8] Gerald, C.F. and Wheatley, P.O., *Applied Numerical Analysis*, 6th Ed. Addison-Wesley Publishing Company, Inc, Reading, MA, 1997.

[9] Mathews, J.H., *Numerical Methods for Mathematics, Science, and Engineering*, 2nd Ed. Prentice Hall, Inc., Englewood Cliffs, NJ 1992.

[10] Berger, I. "Standards for Car Talk". *IEEE: The Institute*. March 7 2007. Accessed February 13 2008. <http://www.theinstitute.ieee.org/portal/site/tionline/menuitem.130a3558587d56e8fb2275875bac26c8/index.jsp?&pName=institute_level1_article&TheCat=2201&article=tionline/legacy/inst2007/mar2007/featurewire.xml&>.

[11] US Department of Transportation. "Dedicated Short Range Communications (DSRC)". *ITS Standards Advisory*. April 2003 Advisory No.3. Accessed February 13 2008. <http://www.standards.its.dot.gov/Documents/advisories/dsrc_advisory.htm>.

[12] DSRC Industry Consortium. *DSRC Technology and the DSRC Industry Consortium (DIC) Prototype Team*. White paper. 2005.

[13] Garber, N.J. and Hoel, L.A., *Traffic and Highway Engineering*, 3rd Ed. The Wadsworth Group, Pacific Grove, CA, 2002.

[14] US Department of Transportation. *Manual on Uniform Traffic Control Devices*, rev. 2. Federal Highway Administration. 2007.

[15] US Department of Transportation. *Roundabouts: An Informational Guide*. Federal Highway Administration. 2000.

[16] Taekratok, T. *Modern Roundabouts for Oregon*. Oregon Department of Transportation. 1998.

[17] Rietveld, J. "History of the Stop Sign in America". 2007. Last Accessed February 26 2008. <http://signalfan.freeservers.com/road%20signs/stopsign.htm> .

[18] Regenold, M. "From Waving Arms to LED's: A Brief History of the Traffic Signal". *UC Berkeley: Tech Transfer Newsletter*. Fall 2007. Last Accessed February 26 2008. <http://www.techtransfer.berkeley.edu/newsletter/07-4/traffic_signals.php>.

[19] Bellis, M. "The History of Roads and Asphalt". Last Accessed February 26 2008. <http://inventors.about.com/library/inventors/blasphalt.htm>.

[20] Mueller, E.A. "Aspects of the History of Traffic Signals". *IEEE Transactions on Vehicular Technology, Vol. VT-19, NO. 1, February 1970*.

[21] Washington State Department of Transportation. "Traffic Signals". 2007. Last Accessed February 28 2008. <http://www.wsdot.wa.gov/biz/trafficoperations/traffic/signals.htm>.

[22] The New York Times. "3 Are Sentenced to 15 Years in Fatal Stop Sign Prank". June 21 1997. Last Accessed February 28 2008. <http://query.nytimes.com/gst/fullpage.html?res=9E07E3DA103EF932A15755C0A961958260>.

[23] DeKoster, L. "Missing Stop Sign Causes Car Accident on Stanton". Iowa State Daily. October 14 1998. Last Accessed February 28 2008. <http://media.www.iowastatedaily.com/media/storage/paper818/news/1998/10/14/UndefinedSection/Missing.Stop.Sign.Causes.Car.Accident.On.Stanton-1073826.shtml>.

[24] Lakey, J. "Downed Stop Sign Putting Pedestrians at Serious Risk". TheStar.com. January 28 2008. Last Accessed February 28 2008. <http://www.thestar.com/News/article/296962>.

[25] TransSafety, Inc. "When Stop Sign Was Downed Six Times in Seventeen Days, Texas DOT May Be Liable for Not Correcting Sign's Susceptibility to Vandalism". Road Injury

Prevention & Litigation Journal. June 29 2000. Last Accessed February 28 2008. <http://www.usroads.com/journals/rilj/0107/ri010701.htm>.

[26] Arizona Department of Transportation. "Are Traffic Signals Really a Cure-All?". Last Accessed February 28 2008. <http://www.azdot.gov/Highways/traffic/Signal.asp>.

[27] Post, Buckley, Schuh & Jernigan, Inc. *Lower Southeast Florida Hurricane Evacuation Study Technical Data Report: Transportation Analysis Chapter (Broward Version)*. January 1991.

[28] BBC. "Weather Disasters 'Getting Worse'". November 25 2007. Last Accessed February 29 2008. <http://news.bbc.co.uk/2/hi/in_depth/7111623.stm>.

[29] Underwood, S. *Wireless Vehicle Communication Systems*. Center for Automotive Research: Management Briefing Seminar. August 8 2006.

[30] US Department of Transportation: Research and Innovative Technology Administration. "Current Activities". December 5 2007. Last Accessed March 2 2008. <http://www.its.dot.gov/cicas/cicas_current_act.htm>.

[31] European Commission. *Standardisation Mandate Forwarded to the European Standardisation Bodies in the Field of Road Transport Telematics*. April 24 1998.

[32] Oyama, S. *DSRC at 5.8 GHz in Japan*., DSRC International Task Force. March 29 2005.

[33] Eichler, S., Schroth, C., and Eberspächer, J. *Car-to-Car Communications*. August 29 2006.

[34] Yonkjin, L. and Yicheng, J., "MAS and AIS Based Vessel Automatic Collision Avoidance System". *International Conference on Transportation Engineering 2007*.

[35] Shi, C., Zhang, M., and Peng, J. "Vessel Collision Avoidance in Close-Quarter Situation Using Differential Games". *International Conference on Transportation Engineering 2007*.

[36] Wilson, P.A., Harris, C.J., and Hong, X., "A Line of Sight Counteraction Navigation Algorithm for Ship Encounter Collision Avoidance". *The Journal of Navigation*, Vol. 56, No. 1, pp. 111-121, 2003.

[37] Richards, A. and How, J.P., "Aircraft Trajectory Planning With Collision Avoidance Using Mixed Linear Programming". *Proceedings of the American Control Conference*, Vol. 3, pp. 1936-1941, 2002.

[38] Ikeda, Y., Nguyen, B., Barfield, A. Sundqvist, B., and Jones, S., "Automatic Air Collision Avoidance System". *Proceedings of the 41st SICE Annual Conference*, Vol. 1, pp. 630-635, 2002.

[39] Zhao, Y. and Schultz, R.L., "Deterministic Resolution of Two Aircraft Conflict in Free Flight". *AIAA Guidance, Navigation, and Control Conference*, 1997.

[40] Barfoot, T.D. and Clark, C.M., "Motion Planning for Formations of Mobile Robots". *Robotics and Autonomous Systems*, Vol. 46, No. 2, pp. 65-78, 2004.

[41] Mastellone, S., Stipanovic, D.M., Graunke, C.R., Intlekofer, K.A., and Spong, M.W., "Formation Control and Collision Avoidance for Multi-agent Non-holonomic Systems: Theory and Experiments". *The International Journal of Robotics Research*, Vol. 27, No. 1, pp. 107-126, January 2008.

[42] Fox, D., Burgard, W., Thrun, S., and Cremers, A.B., "A Hybrid Collision Avoidance Method For Mobile Robots". *Proceedings of the IEEE International Conference on Robotics & Automation*, May 1998.

[43] Vahidi, A. and Eskandarian, A., "Research Advances in Intelligent Collision Avoidance and Adaptive Cruise Control". *IEEE Transactions on Intelligent Transportation Systems*, Vol. 4, No. 3, pp. 143-153, September 2003.

[44] Shladover, S. E., "Review of the State of Development of Advanced Vehicle Control Systems (AVCS)". *Vehicle System Dynamics*, Vol. 24, No. 6, pp. 551-595, July 1995.

[45] Behringer, R. and Müller, N., "Autonomous Road Vehicle Guidance from Autobahnen to Narrow Curves". *IEEE Transactions on Robotics and Automation*, Vol. 14, No. 5, pp. 810-815, October 1998.

[46] California PATH, "Vehicle Platooning and Automated Highways". September 1998.

[47] California PATH, "Automated Driving Mini-Demonstration". September 1998.

[48] Hedrick, J.K., Chen, Y., and Mahal, S., "Optimized Vehicle Control/Communication Interaction in an Automated Highway System". California PATH Program, UC Berkeley, October 2001.

[49] Kajiwara, A., "Vehicular Stepped-FM Coded Radar for Collision Avoidance". *Vehicular Technology Conference*, 1998.

[50] Sanmartin-Jara, J., Burgos-Garcia, M., and Retamosa-Sanchez, J., "SS-FH Signals Used for Very Low Interference in Vehicular Cruising Control Systems". *Vehicular Technology Conference*, 1999.

[51] Wang, J. and Rajamani, R., "Should Adaptive Cruise-Control Systems be Designed to Maintain a Constant Time Gap Between Vehicles?". *IEEE Transactions on Vehicular Technology*, Vol. 53, No. 5, pp. 1480-1490, September 2004.

[52] Tsuji, T., Hattori, H., Watanabe, M., and Nagaoka, N., "Development of Night-Vision System". *IEEE Transactions on Intelligent Transportation Systems*, Vol. 3, No. 3, pp. 203-209, September 2002.

[53] Ulmer, B., "VITA - An Autonomous Road Vehicle(ARV) for Collision Avoidance in Traffic". *Proceedings of the Intelligent Vehicles Symposium*, pp.36-41, July 1992.

[54] Gehrig, S.K. and Stein, F.J., "Collision Avoidance for Vehicle-Following Systems". *IEEE Transactions on Intelligent Transportation Systems*, Vol. 8, No. 2, pp. 233-244, June 2007.

[55] Quinlan, S. and Khatib, O., "Elastic Bands: Connecting Path Planning and Control". *Proceedings of IEEE International Conference on Robotics and Automation*, Vol. 2, pp. 802-807, May 1993.

[56] Atev, S., Arumugam, H., Masoud, O., Janardan, R., and Papanikolopoulos, N.P., "A Vision-Based Approach to Collision Prediction at Traffic Intersections". *IEEE Transactions on Intelligent Transportation Systems*, Vol. 6, No. 4, pp. 416-423, December 2005.

[57] Sun, Z., Bebis, G., and Miller, R., "On-Road Vehicle Detection Using Optical Sensors: A Review". *Proceedings of IEEE Conference on Intelligent Transportation Systems*, pp. 585-590, October 2004.

[58] Sun, Z., Bebis, G., and Miller R., "On-Road Vehicle Detection: A Review". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 28, No. 5, pp. 694-711, May 2006.

[59] Pathirana, P.N., Lim, A.E.K., Savkin, A.V., and Hodgson, P.D., "Robust Video/Ultrasonic Fusion-Based Estimation for Automotive Applications". *IEEE Transactions on Vehicular Technology*, Vol. 56, No. 4, pp. 1631-1639, July 2007.

[60] Li, L. and Wang, F.Y., "Cooperative Driving at Blind Crossings Using Intervehicle Communication". *IEEE Transactions on Vehicular Technology,* Vol. 55, No. 6, pp. 1712-1724, November 2006.

[61] Morioka, Y., Sota, T., and Nakagawa, M., "An Anti-Car Collision System Using GPS and 5.8 GHz Inter-Vehicle Communication at an Off-Sight Intersection". *IEEE Vehicular Technology Conference*, Vol. 5, pp. 2019-2024, September 2000.

[62] Sung, K., Yoo, J., and Kim, D., "Collision Warning System on a Curved Road Using Wireless Sensor Networks". *IEEE Vehicular Technology Conference*, pp. 1942-1946, September 2007.

[63] Dresner, K. and Stone, P., "Multiagent Traffic Management: An Improved Intersection Control Mechanism". *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems*, pp. 471-477, July 2005.

[64] Tan, H.S. and Huang, J., "DGPS-Based Vehicle-to-Vehicle Cooperative Collision Warning: Engineering Feasibility Viewpoints". *IEEE Transactions on Intelligent Transportation Systems*, Vol. 7, No. 4, pp. 415-428, December 2006.

[65] Miller, R. and Huang, Q., "An Adaptive Peer-to-Peer Collision Warning System". *IEEE Vehicular Technology Conference*, Vol. 1, pp. 317-321, 2002.

[66] Naumann, R. and Rasche, R., "Intersection Collision Avoidance by Means of Decentralized Security and Communications Management of Autonomous Vehicles". *Proceedings of the 30th ISATA Conference on ATT/ITS*, pp. 1-8, June 1997.

[67] Ueki, J., Mori, J., Nakamura, Y., Horii, Y., and Okada, H., "Development of Vehicular-Collision Avoidance Support System by Inter-Vehicle Communications - VCASS". *IEEE Vehicular Technology Conference*, Vol. 5, pp. 2940-2945, May 2004.

[68] Shladover, S.E. and Tan, S.K., "Analysis of Vehicle Positioning Accuracy Requirements for Communication-Based Cooperative Collision Warning". *Journal of Intelligent Transportation Systems*, Vol. 10, No. 3, pp. 131-140, September 2006.

[69] Sengupta, R., Rezaei, S., Shladover, S.E., Cody, D., Dickey, S., and Krishnan, H., "Cooperative Collision Warning Systems: Concept Definition and Experimental Implementation". *Journal of Intelligent Transportation Systems*, Vol. 11, No. 3, pp. 143-155, July 2007.

[70] Jones, B., "DSRC - Linking the Vehicle and the Road". ITS Joint Programs Office. US Department of Transportation. February 14 2005.

[71] US Department of Transportation: Research and Innovative Technology Administration. "Intelligent Transportation Systems Standards Fact Sheet". January 9 2006. Last Accessed March 19 2008. <http://www.standards.its.dot.gov/fact_sheet.asp?f=80>.

[72] IEEE Vehicular Technology Society (VTS), "IEEE Trial-Use Standard for Wireless Access in Vehicular Environments (WAVE) - Resource Manager". October 13 2006.

[73] Intelligent Transportation Systems Committee, "IEEE Trial-Use Standard for Wireless Access in Vehicular Environments (WAVE) - Security Services for Applications and Management Messages". July 6 2006.

[74] Intelligent Transportation Systems Committee, "IEEE Trial-Use Standard for Wireless Access in Vehicular Environments (WAVE) - Networking Services". April 20 2007.

[75] IEEE Vehicular Technology Society, "IEEE Trial-Use Standard for Wireless Access in Vehicular Environments (WAVE) - Multi-channel Operation". November 29 2006.

[76] Shladover, S.E. Private Communication. October 6 2006.

[77] Jazwinski, A.H., *Stochastic Processes and Filtering Theory*. Academic Press, Inc., New York, NY, 1970.

[78] Julier, S.J. and Uhlmann, J.K., "A New Extension of the Kalman Filter to Nonlinear Systems." *Int. Symp. Aerospace/Defense Sensing, Simul. and Controls*, 1997.

[79] Burmeister, B., Haddadi, A., and Matylis, G., "Application of Multi-Agent Systems in Traffic and Transportation." *IEEE Proceedings on Software Engineering*, Vol. 144, No. 1, pp. 51-60, February 1997.

[80] Farahmand, A.S. and Mili, L., "Cooperative Decentralized Intersection Collision Avoidance Using Extended Kalman Filtering (submitted for publication)." *IEEE Intelligent Vehicles Symposium, 2009.*

Appendix A

```
% Ashil Farahmand
% Recursive Iterated EKF using Modified UCB Engine Model

tic; clc; clear; format long;

KF_iterations = 5*10^4;              % Number of iterations for KF
integration_iterations = 1;          % Number of integrations per each KF_iteration
T = 1*10^-3; num_of_cars = 1;

auto_transmission = logical(1);                 % 0 = (diagnostic)Manual, 1 = Automatic
noise = 1;                                        % 0 = (diagnostic)None, 1 = White, 2 = Colored
rough = logical(0);                               % 0 = random noise, x = noise multiplied by value of rough alternates :/- each iteration
run_optimization = logical([1; 0; 0; 0]);       % 0 = (diagnostic) Do not change inputs, 1 = change inputs according to cost function
min_iterations_to_show_corrections = 6;  % If iterating less than this margin, do not display
show_optimization = 1;                          % 0 = (diagnostic) Do not plot optimization, 1 = Plot Optimization
max_corrections = 1*10^2;                        % Prevents infinite loop in correction
e_correction = 10^-6;                             % Allowable Correction Error

mode_output = [0, 0, 0, 1, 1,    1, 0, 0, 1, 1,    1, 0, 0, 1];
%mode_output = ones(14);

% Output Modes
% -----------------------------------------------------
% Index   |   Property
% 1             throttle angle
% 2               brake angle
% 3          steering direction
% 4            position (x2 vs. x1)
% 5            position (x vs. t)
% 6                  v_mph
% 7                  v_m/s
% 8          distance b/w cars
% 9                    ma
% 10                   we
% 11                   Tb
% 12                  gear
% 13     phase plot (ma vs. we wrt t)
% 14        corrector iterations

w = zeros(5*num_of_cars, 1); e = zeros(5*num_of_cars, 1); z = zeros(5*num_of_cars, 1); Fx = zeros(5*num_of_cars, 5*num_of_cars);

nominal_controls = zeros(3*num_of_cars);                              % Control Vector during one KF iteration
predictions = zeros(5*num_of_cars, integration_iterations + 1);   % Holds the predictions during integration
corrections = zeros(5*num_of_cars, 2);                                 % Holds the corrected states when iterating the correction step (holds
previous and next corrections)
correction_iterations = ones(KF_iterations, 1);                       % Holds the number of iterations the corrector performs/KF iteration

predicted_states = zeros(5*num_of_cars, KF_iterations); corrected_states = zeros(5*num_of_cars, KF_iterations); gear_ratios =
zeros(num_of_cars, KF_iterations);
predicted_opt_states = zeros(4*num_of_cars, KF_iterations);

% Topology Parameters
speed_limits = [22; 50; 10];           % Speed limits (left turn, straight, right turn)
lane_width  = 3.6; line_separation = .1;

% Used by cost function
iteration_to_start_optimization = 10;                          % KF_iteration to start optimization
safe_distance = [2.5; 2.5; 2.5; 2.5];                              % Safe distance to maintain around each respective vehicle
safety_margin = .5;                                                  % Additional distance to maintain between vehicles
direction = [0; 0; 0; 0];                                           % 1 - Left turn, 2 - Straight, 3 - Right turn (Set in vehicle ICs)
order = zeros(num_of_cars, 1);                                   % Order of cars going through the intersection
steady_speed_t_angles = [6.91049; 11.77215; 3.52779];    % Throttle angles to maintain a steady speed at the speed limit
acc_profile = [0.4624; 0.2728; 0.1724; 0.09438; .08311];    % Acceleration values for different velocity ranges
braking_distances = [0; 0];                                          % Distance from intersection to begin braking in preparation for turn
(calculated below)
time_optimal = zeros(num_of_cars, 1);                         % The amount of time needed to traverse the intersection when no other
vehicles are present
```

```
time_to_cross = zeros(num_of_cars, 1);                          % The amount of time needed to traverse the intersection for a given order
target_speed = speed_limits(2);                                  % Used in optimization to signal a necessary acceleration or brake
brake_decel_rates = [-2.037; -4];                                 % Steady state deceleration at corresponding brake angle
brake_decel_angles = [19; 37.6696];                               % Brake angles to achieve corresponding brake decel rates
steering_angle_increment = [ (90*T*4*speed_limits(1)*1609.344)/(3*pi*lane_width*3600);
(90*T*4*speed_limits(3)*1609.344)/(pi*lane_width*3600) ];

% Assign Initial Conditions
K = I(5*num_of_cars); G = I(5*num_of_cars); Q = I(5*num_of_cars);
q = [.01; .01; 10^-10; .01; 100]; % Covariances of process and observation noise
for i = 1:num_of_cars    Q(5*i - 4, 5*i - 4) =  q(1);    Q(5*i - 3, 5*i - 3) = q(2);    Q(5*i - 2, 5*i - 2) = q(3);    Q(5*i - 1, 5*i - 1) = q(4);
Q(5*i - 0, 5*i - 0) = q(5); end
sigma = Q; last_sigma = Q; Rk = Q;

delta_t = T/integration_iterations;                       % One KF_interval time is integration_iterations*delta_t = T
k = zeros(4, 5*num_of_cars);                               % RK4 variables; one col/SV
RK_states = zeros(5*num_of_cars, 1);                       % state vector
u = zeros(3*num_of_cars, KF_iterations);                   % control vector
Jtg_current = zeros(num_of_cars, 1);
MAX = 0.684; R = 8.3145; Tm = 333; Mair = .02897; Vm = .00447; Patm =  101325; c_PR = (R*Tm)/(Patm*Mair*Vm);
Ve = .0049; c_1 = Ve/(4*pi*Vm);
normal_shift_speeds = [0; 7; 15; 25; 45];      % Speed to shift
fast_shift_speeds = [11; 20; 35; 55];          % Speeds to shift under max acceleration
Rg = [0.4167; 0.6817; 1; 1.4993; 2.3058];
c_2 = 1018686; Ca = 0.53384; h = 0.33; Fr = 167.27; Je = 0.2630; Jtg = [0.08202; 0.07592; 0.11388; 0.13150]; Jw = 5.13; M = 2148; Kb =
300; Tau_bv = 0.1;

% Pre-Processing

% Assign braking distances
x = ( (speed_limits(2) - speed_limits(1:2:3))*(1609.344/3600))/(-brake_decel_rates(2));
braking_distances = -.5*brake_decel_rates(2)*x.^2 + speed_limits(1:2:3)*(1609.344/3600).*x + lane_width;

% Car 1
car = 1;
direction(car) = 1;
gear_ratios(car, 1) = Rg(1);
predictions(5*car - 4, 1) = -200;
predictions(5*car - 3, 1) = -lane_width/2;
predictions(5*car - 2, 1) = .0033558;
predictions(5*car - 1, 1) = 5 * (1609.344/3600) *(1/(h*gear_ratios(car, 1)));
u(3*car - 2, 1:KF_iterations) = 11.77215;
%u(3*car - 2, 1:KF_iterations) = 85;
u(3*car - 0, 1:KF_iterations) = 0;

% Car 2
car = 2;
if(num_of_cars > 1)
direction(car) = 2;
gear_ratios(car, 1) = Rg(5);
predictions(5*car - 4, 1) = -lane_width/2;
predictions(5*car - 3, 1) = 200;
predictions(5*car - 2, 1) = .00033558;
predictions(5*car - 1, 1) = 45 * (1609.344/3600) * (1/(h*gear_ratios(car, 1)) );
u(3*car - 2, 1:iteration_to_start_optimization) = 85;
u(3*car - 0, 1:KF_iterations) = 270;
end

% Car 3
car = 3;
if(num_of_cars > 2)
direction(car) = 2;
gear_ratios(car, 1) = Rg(1);
predictions(5*car - 4, 1) = -lane_width/2;
predictions(5*car - 3, 1) = 16 + lane_width;
predictions(5*car - 2, 1) = .00033558;
predictions(5*car - 1, 1) = 2 * (1/(h*gear_ratios(car, 1)) );
u(3*car - 2, 1:iteration_to_start_optimization) = 85;
u(3*car - 0, 1:KF_iterations) = 270;
end

% Car 4
```

```
car = 4;
if(num_of_cars > 3)
direction(car) = 2;
gear_ratios(car, 1) = Rg(1);
predictions(5*car - 4, 1) = -lane_width/2;
predictions(5*car - 3, 1) = 30 + lane_width;
predictions(5*car - 2, 1) = .00033558;
predictions(5*car - 1, 1) = 2 * (1/(h*gear_ratios(car, 1)) );
u(3*car - 2, 1:iteration_to_start_optimization) = 85;
u(3*car - 0, 1:KF_iterations) = 270;
end

predicted_states(:, 1) = predictions(:, 1); corrected_states(:, 1) = predictions(:, 1);

% Begin to iterate (index 1 holds the ICs)
for KF_interval = 1:KF_iterations

   % Run Optimization
   if((sum(run_optimization) > 0) & (KF_interval > iteration_to_start_optimization))       % At least one car needs to be optimized then
true
      ETA = zeros(num_of_cars, 1);

      % Find ETA of each car based on current states (Cost Function)
      for i = 1:num_of_cars
         d = max( abs(corrected_states(5*i - 4, KF_interval)), abs(corrected_states(5*i - 3, KF_interval)) ) - lane_width - safe_distance(i);
         v = corrected_states(5*i - 1, KF_interval)*h*gear_ratios(i, KF_interval);

         % Too close for changing plans (ETA < 0 then no change of control vars)
         if( (direction ~= 2) & (d < braking_distances(ceil(direction(i)*.6))) )             ETA(i) = -i;             run_optimization(i) = 0;
         elseif(direction(i) == 1)

            if( (v + 0.1) > (speed_limits(2)*1609.344/3600) )        % At speed limit
               d = max( abs(corrected_states(5*i - 4, KF_interval)), abs(corrected_states(5*i - 3, KF_interval)) ) - safe_distance(i) -
braking_distances(1);
               k(1) = d/(corrected_states(5*i - 1, KF_interval)*gear_ratios(i, KF_interval)*h);
               k(2) = 0;
            elseif( v > (45*1609.344/3600) )                          % High 4th Gear
               % Find position of car after accelerating to speed limit
               k(1) = (speed_limits(2)*1609.344/3600 - v)/acc_profile(5);
               x = .5*acc_profile(5)*k(1)^2 + v*k(1) - max( abs(corrected_states(5*i - 4, KF_interval)), abs(corrected_states(5*i - 3,
KF_interval)) );

               % Reach Speed Limit
               if(x < -braking_distances(1))                      k(2) = (abs(x) - braking_distances(1))/(speed_limits(2)*1609.344/3600);
               elseif(x > -braking_distances(1))             % Do not reach speed limit before line
                  k(1) = ( sqrt(v^2 + 2*acc_profile(5)*(max( abs(corrected_states(5*i - 4, KF_interval)), abs(corrected_states(5*i - 3,
KF_interval)) )- braking_distances(1)) ) - v )/acc_profile(5);
                  % Find speed of vehicle @ lane_width, then average
                  % over whole time to find v
               end

            % ------------------------------------------------------------------------------------
            elseif( v > (35*1609.344/3600) )                          % Lo 4th Gear
               k(2) = (45 - 35)*(1609.344/3600)/acc_profile(4);
               x = .5*acc_profile(4)*k(2)^2 + v*k(2) - max( abs(corrected_states(5*i - 4, KF_interval)), abs(corrected_states(5*i - 3,
KF_interval)) );

               % Reach High 4th Gear
               if(x < -braking_distances(1))
                  k(2) = (speed_limits(2) - 45)*(1609.344/3600)/acc_profile(5);
                  x  = .5*acc_profile(5)*k(2)^2 + 45*(1609.344/3600)*k(2) + x;

                  % Reach Speed Limit
                 if(x <  -braking_distances(1))                      k(1) = (abs(x) - braking_distances(1))/(speed_limits(2)*1609.344/3600);
                 elseif(x > -braking_distances(1))             % Do not reach speed limit before line
                     k(1) = (sqrt((45*1609.344/3600)^2 + 2*acc_profile(5)*(lane_width + abs(x))) - 45*1609.344/3600)/acc_profile(5);
                  % Find speed of vehicle @ lane_width, then average
                  % over whole time to find v
                 end

               elseif(x > -braking_distances(1))             % Do not reach High 4th Gear
```

```
                  k(1) = ( sqrt(v^2 + 2*acc_profile(4)*(lane_width + max( abs(corrected_states(5*i - 4, KF_interval)),
abs(corrected_states(5*i - 3, KF_interval)) ))) - v )/acc_profile(4);
                 % Find speed of vehicle @ lane_width, then average
                  % over whole time to find v
              end
          else
              if(d > braking_distances(1))
                  k(1) = (d - braking_distances(1))/v;
              end
          end

          k(3) = (braking_distances(1) - lane_width)/((speed_limits(2)/2 - speed_limits(1)/2)*1609.344/3600);
          k(4) = (3*lane_width*pi/4)/(speed_limits(1)*1609.344/3600);
          ETA(i) = k(1) + k(2) + k(3) + k(4);

      elseif(direction(i) == 3)

          if( (v + 0.1) > (speed_limits(2)*1609.344/3600) )          % At speed limit
              d = max( abs(corrected_states(5*i - 4, KF_interval)), abs(corrected_states(5*i - 3, KF_interval)) ) - safe_distance(i) -
braking_distances(2);
              k(1) = d/(corrected_states(5*i - 1, KF_interval)*gear_ratios(i, KF_interval)*h);
              k(2) = 0;
          elseif( v > (45*1609.344/3600) )                             % High 4th Gear
             % Find position of car after accelerating to speed limit
              k(1) = (speed_limits(2)*1609.344/3600 - v)/acc_profile(5);
              x = .5*acc_profile(5)*k(1)^2 + v*k(1) - max( abs(corrected_states(5*i - 4, KF_interval)), abs(corrected_states(5*i - 3,
KF_interval)) );

              % Reach Speed Limit
              if(x < -braking_distances(2))                   k(2) = (abs(x) - braking_distances(2))/(speed_limits(2)*1609.344/3600);
              elseif(x > -braking_distances(2))             % Do not reach speed limit before line
                  k(1) = ( sqrt(v^2 + 2*acc_profile(5)*(max( abs(corrected_states(5*i - 4, KF_interval)), abs(corrected_states(5*i - 3,
KF_interval)) )- braking_distances(2)) ) - v )/acc_profile(5);
                 % Find speed of vehicle @ lane_width, then average
                  % over whole time to find v
              end



          % ------------------------------------------------------------------------------------
          elseif( v > (35*1609.344/3600) )                             % Lo 4th Gear
              k(2) = (45 - 35)*(1609.344/3600)/acc_profile(4);
              x = .5*acc_profile(4)*k(2)^2 + v*k(2) - max( abs(corrected_states(5*i - 4, KF_interval)), abs(corrected_states(5*i - 3,
KF_interval)) );

              % Reach High 4th Gear
              if(x < -braking_distances(2))
                  k(2) = (speed_limits(2) - 45)*(1609.344/3600)/acc_profile(5);
                  x  = .5*acc_profile(5)*k(2)^2 + 45*(1609.344/3600)*k(2) + x;

                  % Reach Speed Limit
                 if(x <  -braking_distances(2))                    k(1) = (abs(x) - braking_distances(2))/(speed_limits(2)*1609.344/3600);
                 elseif(x > -braking_distances(2))             % Do not reach speed limit before line
                     k(1) = (sqrt((45*1609.344/3600)^2 + 2*acc_profile(5)*(lane_width + abs(x))) - 45*1609.344/3600)/acc_profile(5);
                 % Find speed of vehicle @ lane_width, then average
                 % over whole time to find v
                 end

              elseif(x > -braking_distances(2))             % Do not reach High 4th Gear
                 k(1) = ( sqrt(v^2 + 2*acc_profile(4)*(lane_width + max( abs(corrected_states(5*i - 4, KF_interval)),
abs(corrected_states(5*i - 3, KF_interval)) ))) - v )/acc_profile(4);
                 % Find speed of vehicle @ lane_width, then average
                  % over whole time to find v
              end
          else
              if(d > braking_distances(2))
                  k(1) = (d - braking_distances(2))/v;
              end
          end

          k(3) = (braking_distances(2) - lane_width)/((speed_limits(2)/2 - speed_limits(3)/2)*1609.344/3600);
          k(4) = (lane_width*pi/4)/(speed_limits(3)*1609.344/3600);
          ETA(i) = k(1) + k(2) + k(3) + k(4);
```

```
        else

          if( (v + 0.1) > (speed_limits(2)*1609.344/3600) )         % At speed limit
             d = max( abs(corrected_states(5*i - 4, KF_interval)), abs(corrected_states(5*i - 3, KF_interval)) ) - lane_width -
safe_distance(i) + 2*lane_width;
             ETA(i) = d/(corrected_states(5*i - 1, KF_interval)*gear_ratios(i, KF_interval)*h);

          elseif( v > (45*1609.344/3600) )                           % High 4th Gear
             % Find position of car after accelerating to speed limit
             k(1) = (speed_limits(2)*1609.344/3600 - v)/acc_profile(5);
             x = .5*acc_profile(5)*k(1)^2 + v*k(1) - max( abs(corrected_states(5*i - 4, KF_interval)), abs(corrected_states(5*i - 3,
KF_interval)) );

             % Reach Speed Limit
             if(x < lane_width)               ETA(i) = k(1) + (abs(x) + lane_width)/(speed_limits(2)*1609.344/3600);
             elseif(x > lane_width)           % Do not reach speed limit before line
                ETA(i) = ( sqrt(v^2 + 2*acc_profile(5)*(lane_width + max( abs(corrected_states(5*i - 4, KF_interval)),
abs(corrected_states(5*i - 3, KF_interval)) ))) - v )/acc_profile(5);
                % Find speed of vehicle @ lane_width, then average
                % over whole time to find v
             end

          % ---------------------------------------------------------------------------------------
          elseif( v > (35*1609.344/3600) )                           % Lo 4th Gear
             k(2) = (45 - 35)*(1609.344/3600)/acc_profile(4);
             x = .5*acc_profile(4)*k(2)^2 + v*k(2) - max( abs(corrected_states(5*i - 4, KF_interval)), abs(corrected_states(5*i - 3,
KF_interval)) );

             % Reach High 4th Gear
             if(x < lane_width)
                k(1) = (speed_limits(2) - 45)*(1609.344/3600)/acc_profile(5);
                x  = .5*acc_profile(5)*k(1)^2 + (45*1609.344/3600)*k(1) + x;

                % Reach Speed Limit
                if(x < lane_width)               ETA(i) = k(1) + k(2) + (abs(x) + lane_width)/(speed_limits(2)*1609.344/3600);
                elseif(x > lane_width)           % Do not reach speed limit before line
                   ETA(i) = k(2) + (sqrt((45*1609.344/3600)^2 + 2*acc_profile(5)*(lane_width + abs(x))) -
45*1609.344/3600)/acc_profile(5);
                % Find speed of vehicle @ lane_width, then average
                % over whole time to find v
                end

             elseif(x > lane_width)              % Do not reach High 4th Gear
                ETA(i) = ( sqrt(v^2 + 2*acc_profile(4)*(lane_width + max( abs(corrected_states(5*i - 4, KF_interval)),
abs(corrected_states(5*i - 3, KF_inte rval)) ))) - v )/acc_profile(4);
                % Find speed of vehicle @ lane_width, then average
                % over whole time to find v
             end
          else
             ETA(i) = (d + lane_width)/v;
          end

        end
      end

      toc

      % Sort the ETAs in ascending optimization order (duplicateETAs yield error)
      ETA
      order = sort(ETA)
      for i = 1:num_of_cars           order(i) = find(order(i) == ETA);        end

      % Perform Optimization (assume only 1 car first)
      x = [0; 0];        v = 0;        a = 0;        k = zeros(1, 2);

      for i = 1:num_of_cars
        delay = 0;                          % Time delay to avoid collision
        collision_points = 0;               % Indices where a collision will occur with current trajectory plan
        while(run_optimization(order(i)) == 1)
          x = [corrected_states(5*order(i) - 4, KF_interval); corrected_states(5*order(i) - 3, KF_interval)];
          v = corrected_states(5*order(i) - 1, KF_interval)*h*gear_ratios(order(i), KF_interval);
        v_mph = v*3600/1609.344;              target_speed = speed_limits(2)*1609.344/3600;              a = -brake_decel_rates(1)/100;
k(2) = 1;
        source = find(max(abs(x)) == abs(x));               sink = find(min(abs(x)) == abs(x));
        u(3*order(i) - 2, KF_interval:KF_iterations) = 0;              u(3*order(i) - 1, KF_interval:KF_iterations) = 0;
u(3*order(i) - 0, KF_interval:KF_iterations) = u(3*order(i) - 0, 1);

        if( (direction(order(i)) == 1) | | (direction(order(i)) == 3) )
          for j = KF_interval:KF_iterations
            if((sum(size(collision_points)) - 2) > 0) % If original trajectory is invalid, delay some then proceed normally
              a = a*exp(-T/Tau_bv) + brake_decel_rates(1)/100;                         u(3*order(i) - 1, j) = brake_decel_angles(1);
k(1) = a;

              if(v_mph < (speed_limits(2) - delay) )                           collision_points = 1;                          end
            elseif( abs(x(source)) > braking_distances(ceil(direction(order(i))*.6)) )   % Have not met decel area yet
              u(3*order(i) - 2, j) = 85;

              if(collision_points == 1)                              k(1) = k(1)*exp(-T/Tau_bv);
              else                          k(1) = 0;                          end

              if(v > target_speed)                                                    a = 0;                            u(3*order(i) - 2, j) =
steady_speed_t_angles(2);
              elseif(v_mph > 47)                                                      a = k(1) + acc_profile(5);
              elseif(v_mph > fast_shift_speeds(3))                            a = k(1) + acc_profile(4);
              elseif(v_mph > fast_shift_speeds(2))                            a = k(1) + acc_profile(3);
              elseif(v_mph > fast_shift_speeds(1))                            a = k(1) + acc_profile(2);
              else                                                                        a = k(1) + acc_profile(1);                          end
            elseif( (abs(x(source)) > lane_width) && (v < target_speed - 0.391) )      % In decel area and below target speed
              u(3*order(i) - 2, j) = 85;                         target_speed = brake_decel_rates(2)*T + target_speed;                          k(1) =
k(1)*exp(-T/Tau_bv);

              if(v_mph > 47)                                                          a = k(1) + acc_profile(5);
              elseif(v_mph > fast_shift_speeds(3))                            a = k(1) + acc_profile(4);
              elseif(v_mph > fast_shift_speeds(2))                            a = k(1) + acc_profile(3);
              elseif(v_mph > fast_shift_speeds(1))                            a = k(1) + acc_profile(2);
              else                                                                        a = k(1) + acc_profile(1);                          end
            elseif( (abs(x(source)) > lane_width) && (v >= target_speed - 0.391) )      % In decel area and at target speed
              if(k(1) > 0)                              a = -brake_decel_rates(2)/100 + k(1)/100;                              k(1) = -1;
'*',                  end

              a = a*exp(-T/Tau_bv) + brake_decel_rates(2)/100;                         u(3*order(i) - 1, j) = brake_decel_angles(2);
              target_speed = a*T + target_speed;                       k(3) = 1;
            elseif( (abs(x(source)) <= lane_width) && (abs(x(sink)) <= lane_width) )   % Begin turning manuever(inside Isection)
              if(k(3) == 1)
                steering_angle_increment = [ (90*T*4*(v_mph - 0.5)*1609.344)/(3*pi*lane_width*3600); (90*T*4*(v_mph -
0.5)*1609.344)/(pi*lane_width*3600) ];
                k(3) = -1;
              end

              k(1) = 0;
              a = a*exp(-T/Tau_bv);                          target_speed = speed_limits(2)*1609.344/3600;
              u(3*order(i) - 2, j) = steady_speed_t_angles(direction(order(i)));
              u(3*order(i), j) = u(3*order(i), j - 1) - (direction(order(i)) - 2)*steering_angle_increment( ceil(direction(order(i))*.6) );
            elseif(abs(x(sink)) > lane_width)                                                                    % Out of intersection moving away
              u(3*order(i) - 2, j) = 85;                          u(3*order(i) - 0, j) = u(3*order(i), 1) - (direction(order(i)) - 2)*90;

              % Assigns time to cross
              if(k(1) == 0)
                k(1) = -2;
                time_to_cross(order(i)) = (j - KF_interval)*T;

                % Assign costs of this iteration's trajectory
                if(collision_points == 0)                  time_optimal(order(i)) = time_to_cross(order(i));                end
              end

              if(v > target_speed)                                                    a = 0;                            u(3*order(i) - 2, j) =
steady_speed_t_angles(2);
              elseif(v_mph > 47)                                                      a = acc_profile(5);
              elseif(v_mph > fast_shift_speeds(3))                            a = acc_profile(4);
              elseif(v_mph > fast_shift_speeds(2))                            a = acc_profile(3);
              elseif(v_mph > fast_shift_speeds(1))                            a = acc_profile(2);
              else                                                                        a = acc_profile(1);                          end
            end
```

```
                x(1) = (.5*a*T^2 + v*T)*cos(u(3*order(i), j)*pi/180) + x(1);
                x(2) = (.5*a*T^2 + v*T)*sin(u(3*order(i), j)*pi/180) + x(2);
                v = a*T + v;                v_mph = v*3600/1609.344;
                if( (a < 0) & (v < 0) )              v = 0;              v_mph = 0;              end
                predicted_opt_states((4*order(i) - 3:4*order(i)), j) = [x(1); x(2); v_mph; a];
            end
        else      % If not turning, then go straight
            k(3) = 0;
            for j =  KF_interval:KF_iterations
                if((sum(size(collision_points)) - 2) > 0)     % If original trajectory is invalid, delay some then proceed normally
                    a = a*exp(-T/Tau_bv) + brake_decel_rates(1)/100;              u(3*order(i) - 1, j) = brake_decel_angles(1);
k(1) = a;

                    if(v_mph < (speed_limits(2) - delay))              collision_points = 1;              end
                elseif( (k(3) == 0) & (abs(x(source)) > lane_width) & (abs(corrected_states(5*order(i) - 2 - sink, KF_interval) - x(source)) >
abs(corrected_states(5*order(i) - 2 - sink, KF_interval))) )
                    k(3) =  -1;
                    time_to_cross(order(i)) = (j - KF_interval)*T;

                    % Assign costs of this iteration's trajectory
                    if(collision_points == 0)          time_optimal(order(i)) = time_to_cross(order(i));          end
                else
                    u(3*order(i) - 2, j) = 85;

                    if(collision_points == 1)              k(1) = k(1)*exp(-T/Tau_bv);
                    else              k(1) = 0;              end

                    if(v > target_speed)                         a = 0;              u(3*order(i) - 2, j) =
steady_speed_t_angles(2);
                    elseif(v_mph > 47)                           a = k(1) + acc_profile(5);
                    elseif(v_mph > fast_shift_speeds(3))         a = k(1) + acc_profile(4);
                    elseif(v_mph > fast_shift_speeds(2))         a = k(1) + acc_profile(3);
                    elseif(v_mph > fast_shift_speeds(1))         a = k(1) + acc_profile(2);
                    else                                          a = k(1) + acc_profile(1);              end
                end

                x(1) = (.5*a*T^2 + v*T)*cos(u(3*order(i), j)*pi/180) + x(1);
                x(2) = (.5*a*T^2 + v*T)*sin(u(3*order(i), j)*pi/180) + x(2);
                v = a*T + v;                v_mph = v*3600/1609.344;
                if( (a < 0) & (v < 0) )              v = 0;              v_mph = 0;              end
                predicted_opt_states((4*order(i) - 3:4*order(i)), j) = [x(1); x(2); v_mph; a];
            end
        end


        % Compare trajectory plan to plans of all preceding cars
        if(i > 1)
            for car = 1:(i - 1)
                x = sqrt( (predicted_opt_states(4*order(i) - 3, 1:KF_iterations) - predicted_opt_states(4*order(car) - 3, 1:KF_iterations)).^2
+ (predicted_opt_states(4*order(i) - 2, 1:KF_iterations) - predicted_opt_states(4*order(car) - 2, 1:KF_iterations)).^2 );
                collision_points = find(x < (safe_distance(order(i)) + safe_distance(order(car)) + safety_margin));

                if((sum(size(collision_points)) - 2) > 0)              delay = delay + 0.5;
                else              run_optimization(order(i)) = 0;              end
            end
        else              run_optimization(order(i)) = 0;              end

      end
    end
    time_optimal
    time_to_cross - time_optimal
    'Cost = '
    mean(time_to_cross - time_optimal)
    toc
  elseif((show_optimization == 0) | (KF_interval <= iteration_to_start_optimization))
    for i = 1:num_of_cars
      predicted_opt_states(4*i - 3, KF_interval) = corrected_states(5*i - 4, KF_interval);
      predicted_opt_states(4*i - 2, KF_interval) = corrected_states(5*i - 3, KF_interval);
      predicted_opt_states(4*i - 1, KF_interval) = corrected_states(5*i - 1, KF_interval)*h*gear_ratios(i, KF_interval)*3600/1609.344;
      predicted_opt_states(4*i - 0, KF_interval) = 0;
    end
    end

    % Assign nominal controls
    predictions(:, 1) = corrected_states(:, KF_interval);
    nominal_controls = u(:, KF_interval);

    if(auto_transmission == 1)
        % Transmission
        for car = 1:num_of_cars
            v =  gear_ratios(car, KF_interval)*h*predictions(5*car - 1, 1);
            v_mph = v*(3600/1609.344);                          % Convert m/s -> mph

            if( (nominal_controls(3*car - 2) == 85) && (v_mph > normal_shift_speeds(2)) )
                % Down shift for better acceleration
                if(v_mph > fast_shift_speeds(4))                        gear_ratios(car, KF_interval + 1) = Rg(5);                      Jtg_current(car) =
Jtg(4);
                elseif(v_mph > fast_shift_speeds(3))                    gear_ratios(car, KF_interval + 1) = Rg(4);                      Jtg_current(car) =
Jtg(4);
                elseif(v_mph > fast_shift_speeds(2))                    gear_ratios(car, KF_interval + 1) = Rg(3);                      Jtg_current(car) =
Jtg(3);
                elseif(v_mph > fast_shift_speeds(1))                    gear_ratios(car, KF_interval + 1) = Rg(2);                      Jtg_current(car) =
Jtg(2);
                else                                                        gear_ratios(car, KF_interval + 1) = Rg(1);
Jtg_current(car) = Jtg(1);
                end
            elseif(v_mph > normal_shift_speeds(5))                  gear_ratios(car, KF_interval + 1) = Rg(5);                  Jtg_current(car) =
Jtg(4);
            elseif(v_mph > normal_shift_speeds(4))                  gear_ratios(car, KF_interval + 1) =Rg(4);                   Jtg_current(car) =
Jtg(4);
            elseif(v_mph > normal_shift_speeds(3))                  gear_ratios(car, KF_interval + 1) = Rg(3);                  Jtg_current(car) =
Jtg(3);
            elseif(v_mph > normal_shift_speeds(2))                  gear_ratios(car, KF_interval + 1) = Rg(2);                  Jtg_current(car) =
Jtg(2);
            else                                                        gear_ratios(car, KF_interval + 1) = Rg(1);                  Jtg_current(car) =
Jtg(1);
            end

            predictions(5*car - 1, 1) = v/(gear_ratios(car, KF_interval + 1)*h);                % Adjusts the engine velocity if gear shift occurs
        end
    else        for car = 1:num_of_cars            gear_ratios(car, KF_interval + 1) = gear_ratios(car, KF_interval);        end,    end

    % Prediction
    for integration_interval = 1:integration_iterations
        RK_states = predictions(:, integration_interval);

        for j = 1:4
            k(1:5:5*num_of_cars, j) = gear_ratios(:,
KF_interval)*h.*RK_states(4:5:5*num_of_cars).*cos(nominal_controls(3:3:3*num_of_cars)*pi/180);
            k(2:5:5*num_of_cars, j) = gear_ratios(:,
KF_interval)*h.*RK_states(4:5:5*num_of_cars).*sin(nominal_controls(3:3:3*num_of_cars)*pi/180);

            d_ma_i = MAX*( (nominal_controls(1:3:3*num_of_cars)/85).^2 ).*( (-4.9958)*(c_PR*RK_states(3:5:5*num_of_cars)).^5 +
(5.8832)*(c_PR*RK_states(3:5:5*num_of_cars)).^4 + (-1.1218)*(c_PR*RK_states(3:5:5*num_of_cars)).^3 + (-
.6579)*(c_PR*RK_states(3:5:5*num_of_cars)).^2 + (-.1278)*(c_PR*RK_states(3:5:5*num_of_cars)) + 1.0104 );
            d_ma_o = c_1*(24.5*RK_states(4:5:5*num_of_cars).*RK_states(3:5:5*num_of_cars).^2 -
0.167*RK_states(4:5:5*num_of_cars).*RK_states(3:5:5*num_of_cars) + (8.10*10^(-4))*RK_states(4:5:5*num_of_cars) - (3.10*10^(-
4))*RK_states(3:5:5*num_of_cars).^2 + 222*RK_states(3:5:5*num_of_cars) +
0.352).*RK_states(3:5:5*num_of_cars).*RK_states(4:5:5*num_of_cars);

            k(3:5:5*num_of_cars, j) = d_ma_i - d_ma_o;
            k(4:5:5*num_of_cars, j) = ( ((c_2*d_ma_o)./RK_states(4:5:5*num_of_cars)) - (0.1056*RK_states(4:5:5*num_of_cars) + 15.10) -
((1*10^-4)*RK_states(4:5:5*num_of_cars).^2) - (gear_ratios(:, KF_interval).*(RK_states(5:5:(5*num_of_cars)) + Ca*(gear_ratios(:,
KF_interval).^2).*(h^3).*(RK_states(4:5:(5*num_of_cars)).^2) + h*Fr)))./(Je + Jtg_current + (gear_ratios(:, KF_interval).^2)*(Jw +
4*M*h^2));
            k(5:5:5*num_of_cars, j) = (Kb*nominal_controls(2:3:3*num_of_cars) - RK_states(5:5:5*num_of_cars))/Tau_bv;

            if(j < 3)               RK_states = RK_states + (delta_t/2)*k(:, j);
            else                    RK_states = RK_states + delta_t*k(:, j);            end
        end

        predictions(:, integration_interval + 1) = predictions(:, integration_interval) + (delta_t/6)*( k(:, 1) + 2*k(:, 2) + 2*k(:, 3) + k(:, 4) );  %
State Transition
```

```
    end

    for i = 1:num_of_cars
        % we constraints (if braking limit we = 0 else w = idle speed)
        if( (predictions(5*i - 1, integration_iterations + 1) < 6.5) && (u(3*i - 1, KF_interval) > 0) )        predictions(5*i - 1,
integration_iterations + 1) = 10^-5;
        elseif(predictions(5*i - 1, integration_iterations + 1) < 6.5)        predictions(5*i - 1, integration_iterations + 1) = 6.5;    end
        % ma constraints (ma > 0)
        if( (predictions(5*i - 2, integration_iterations + 1) < 10^-10) )        predictions(5*i - 2, integration_iterations + 1) = 10^-10;    end
    end

    predicted_states(:, KF_interval + 1) = predictions(:, integration_iterations + 1);

    if(noise ~= 0)    % If noise == 0, skip correction step
        % Assign new values to noise dependent variables
        for i = 1:num_of_cars
            if(noise == 2)
                G(5*car - 4, 5*car - 1) = gear_ratios(car, KF_interval)*h*cos(nominal_controls(3*car - 0)*pi/180);
                G(5*car - 3, 5*car - 1) = gear_ratios(car, KF_interval)*h*sin(nominal_controls(3*car - 0)*pi/180);
                G(5*car - 1, 5*car - 2) = ( (.352)*c_2*c_1 )/( Je + Jtg_current(car) + (gear_ratios(car, KF_interval).^2)*(Jw + 4*M*h^2) );
                G(5*car - 1, 5*car - 0) = -gear_ratios(car, KF_interval)/( Je + Jtg_current(car) + (gear_ratios(car, KF_interval).^2)*(Jw +
4*M*h^2) );
            end

            w(5*i - 4) = randn*sqrt(q(1)); w(5*i - 3) = randn*sqrt(q(2)); w(5*i - 2) = randn*sqrt(q(3)); w(5*i - 1) = randn*sqrt(q(4)); w(5*i - 0) =
randn*sqrt(q(5));
            e(5*i - 4) = randn*sqrt(q(1)); e(5*i - 3) = randn*sqrt(q(2)); e(5*i - 2) = randn*sqrt(q(3)); e(5*i - 1) = randn*sqrt(q(4)); e(5*i - 0) =
randn*sqrt(q(5));
        end

        % Creates alternating noise +\- if enabled
        if(rough ~= 0)
            if (KF_interval == 2*floor(KF_interval/2))        w = -rough*abs(w);
            else        w = rough*abs(w);        end
        end

        % Get new observations
        x = predicted_states(:, KF_interval + 1) + G*w;    z = x + e;

        % Iterated Corrections
        corrections(:, 1) = 0;    corrections(:, 2) = predicted_states(:, KF_interval + 1);
        last_sigma = sigma;    Q = G*Q*G'*T;

        i = 0;
        while( (max( abs(corrections(:, 2) - corrections(:, 1)) ) > e_correction) && (i < max_corrections) )
            i = i + 1;        corrections(:, 1) = corrections(:, 2);

            % Compute Jacobian (non-zero along block diagonal)
            for car = 1:num_of_cars
                Fx(5*car - 4, 5*car - 4) = 0;        Fx(5*car - 4, 5*car - 3) = 0;        Fx(5*car - 4, 5*car - 2) = 0;
                Fx(5*car - 4, 5*car - 1) = gear_ratios(car, KF_interval)*h*cos(nominal_controls(3*car - 0)*pi/180);
                Fx(5*car - 4, 5*car - 0) = 0;

                Fx(5*car - 3, 5*car - 4) = 0;        Fx(5*car - 3, 5*car - 3) = 0;        Fx(5*car - 3, 5*car - 2) = 0;
                Fx(5*car - 3, 5*car - 1) = gear_ratios(car, KF_interval)*h*sin(nominal_controls(3*car - 0)*pi/180);
                Fx(5*car - 3, 5*car - 0) = 0;

                Fx(5*car - 2, 5*car - 4) = 0;        Fx(5*car - 2, 5*car - 3) = 0;
                Fx(5*car - 2, 5*car - 2) = MAX*((nominal_controls(3*car - 2)/85)^2)*(5*(-4.9958)*c_PR^5*corrections(5*car - 2, 1)^4 +
4*(5.8832)*c_PR^4*corrections(5*car - 2, 1)^3 + 3*(-1.1218)*c_PR^3*corrections(5*car - 2, 1)^2 + 2*(-.6579)*c_PR^2*corrections(5*car -
2, 1) + (-.1278)*c_PR) - (c_1*(3*24.5*corrections(5*car - 1, 1)^2*corrections(5*car - 2, 1)^2 - 2*0.167*corrections(5*car - 1,
1)^2*corrections(5*car - 2, 1) + 8.10*10^-4*corrections(5*car - 1, 1)^2 - 3*3.10*10^-4*corrections(5*car - 1, 1)*corrections(5*car - 2, 1)^2
+ 2*222*corrections(5*car - 1, 1)*corrections(5*car - 2, 1) + 0.352*corrections(5*car - 1, 1)) );
                Fx(5*car - 2, 5*car - 1) = -c_1*(2*24.5*corrections(5*car - 1, 1)*corrections(5*car - 2, 1)^3 - 2*0.167*corrections(5*car - 1,
1)*corrections(5*car - 2, 1)^2 + 2*8.10*10-4*corrections(5*car - 1, 1)*corrections(5*car - 2, 1) - 3.10*10^-4*corrections(5*car - 2, 1)^3 +
222*corrections(5*car - 2, 1)^2 + 0.352*corrections(5*car - 2, 1));
                Fx(5*car - 2, 5*car - 0) = 0;

                Je_star = Je + Jtg_current(car) + (gear_ratios(car, KF_interval)^2)*(Jw + 4*M*h^2);

                Fx(5*car - 1, 5*car - 4) = 0;        Fx(5*car - 1, 5*car - 3) = 0;
            Fx(5*car - 1, 5*car - 2) = (c_1*c_2/Je_star)*( 3*24.5*corrections(5*car - 1, 1)*corrections(5*car - 2, 1)^2 -
2*0.167*corrections(5*car - 1, 1)*corrections(5*car - 2, 1) + 8.10*10^-4*corrections(5*car - 1, 1) - 3*3.10*10^-4*corrections(5*car - 2, 1)^2
+ 2*222*corrections(5*car - 2, 1) + 0.352 );
            Fx(5*car - 1, 5*car - 1) = (1/Je_star)*( (c_1*c_2*(24.5*corrections(5*car - 2, 1)^3 - 0.167*corrections(5*car - 2, 1)^2 + 8.10*10^-
4*corrections(5*car - 2, 1) ) - 0.1056 - 2*(4.3*10^-6)*corrections(5*car - 1, 1) - 2*Ca*gear_ratios(car, KF_interval)^3*h^3*corrections(5*car
- 1, 1)) );
            Fx(5*car - 1, 5*car - 0) = -gear_ratios(car, KF_interval)/Je_star;

            Fx(5*car - 0, 5*car - 4) = 0;          Fx(5*car - 0, 5*car - 3) = 0;          Fx(5*car - 0, 5*car - 2) = 0;          Fx(5*car - 0,
5*car - 1) = 0;
            Fx(5*car - 0, 5*car - 0) = -1/Tau_bv;
          end

          %phi = I(5*num_of_cars) + Fx*T;
          phi = expm(Fx*T);        sigma = phi*last_sigma*phi' + Q;        K = sigma*inv(sigma + Rk);

          corrections(:, 2) = predicted_states(:, KF_interval + 1) + K*(z - predicted_states(:, KF_interval + 1));
        end
        correction_iterations(KF_interval) = i;
        sigma = (I(5*num_of_cars) - K)*last_sigma;

        if(i > min_iterations_to_show_corrections)        [KF_interval        i],     end

        % Force physical and model constraints
        for i = 1:num_of_cars
          if(corrections(5*i, 2) < 0)          corrections(5*i, 2) = 0;        end        % Force Tb > 0

          % we constraints (if braking limit we = 0 else w = idle speed)
          if( (corrections(5*i - 1, 2) < 6.5) && (u(3*i - 1, KF_interval) > 0) )          corrections(5*i - 1, 2) = 10^-5;
          elseif(corrections(5*i - 1, 2) < 6.5)          corrections(5*i - 1, 2) = 6.5;        end

          % ma constraints (ma > 0)
          if( (corrections(5*i - 2, 2) < 10^-10) )        corrections(5*i - 2, 2) = 10^-10;     end
        end

        corrected_states(:, KF_interval + 1) = corrections(:, 2);
    else
        corrected_states(:, KF_interval + 1) = predicted_states(:, KF_interval + 1);
    end

end

% Post Processing
i = 0;

if(mode_output(1) == 1)
    i = i + 1;    figure(i);

    % Plot Throttle
    for j = 1:num_of_cars
        subplot(num_of_cars, 1, j);      hold on;      grid;
        plot((1:KF_iterations)*T, u(3*j - 2, 1:KF_iterations), 'blue');
        title(['Car ', int2str(j), ' : Throttle Angle wrt Time']);    xlabel('Time [s]');    ylabel('\alpha [deg]');
    end
end

if(mode_output(2) == 1)
    i = i + 1;    figure(i);

    % Plot Brake Angle
    for j = 1:num_of_cars
        subplot(num_of_cars, 1, j);      hold on;      grid;
        plot((1:KF_iterations)*T, u(3*j - 1, 1:KF_iterations), 'blue');
        title(['Car ', int2str(j), ' : Brake Angle wrt Time']);    xlabel('Time [s]');    ylabel('\beta [deg]');
    end
end

if(mode_output(3) == 1)
    i = i + 1;    figure(i);

    % Plot Steering Angle
    for j = 1:num_of_cars
```

```
        subplot(num_of_cars, 1, j);        hold on;        grid;
        plot((1:KF_iterations)*T, u(3*j - 0, 1:KF_iterations), 'blue');
        title(['Car ', int2str(j), ' : Steering Angle wrt Time']);    xlabel('Time [s]');    ylabel('\theta [deg]');
    end
end

if(mode_output(4) == 1)
    i = i + 1;    figure(i);

    % Plot Positions (x2 vs. x1)
    for j = 1:num_of_cars
        subplot(num_of_cars, 1, j);        hold on;

        for k = 1:num_of_cars
            % North
            plot([ -lane_width, -lane_width ],             [ 1000, (0 + lane_width) ], 'black');            plot([ -line_separation, -line_separation ], [
1000, (0 + lane_width) ], 'yellow');
            plot([ 0, 0 ],                                 [ 1000, (0 + lane_width) ], 'black');            plot([ line_separation, line_separation ],    [
1000, (0 + lane_width) ], 'yellow');
            plot([ lane_width, lane_width ],               [ 1000, (0 + lane_width) ], 'black');

            % South
            plot([ -lane_width, -lane_width ],             [ -1000, (0 - lane_width) ], 'black');           plot([ -line_separation, -line_separation ], [ -
1000, (0 - lane_width) ], 'yellow');
            plot([ 0, 0 ],                                 [ -1000, (0 - lane_width) ], 'black');           plot([ line_separation, line_separation ],    [ -
1000, (0 - lane_width) ], 'yellow');
            plot([ lane_width, lane_width ],               [ -1000, (0 - lane_width) ], 'black');

            % West
            plot([ -1000, (0 - lane_width) ], [ lane_width, lane_width ], 'black');            plot([ -1000, (0 - lane_width) ], [ line_separation,
line_separation ], 'yellow');
            plot([ -1000, (0 - lane_width) ], [ 0, 0 ], 'black');            plot([ -1000, (0 - lane_width) ], [ -line_separation, -line_separation ], 'yellow');
            plot([ -1000, (0 - lane_width) ], [ -lane_width, -lane_width ], 'black');

            % East
            plot([ (0 + lane_width), 1000 ], [ lane_width, lane_width ], 'black');            plot([ (0 + lane_width), 1000 ], [ line_separation,
line_separation ], 'yellow');
            plot([ (0 + lane_width), 1000 ], [ 0 0 ], 'black');            plot([ (0 + lane_width), 1000 ], [ -line_separation, -line_separation ], 'yellow');
            plot([ (0 + lane_width), 1000 ], [ -lane_width, -lane_width ], 'black');
        end

        plot(corrected_states(5*j - 4, 1), corrected_states(5*j - 3, 1), 'red O');
        p = plot(predicted_states(5*j - 4, 1:KF_iterations), predicted_states(5*j - 3, 1:KF_iterations), 'blue');
        c = plot(corrected_states(5*j - 4, 1:KF_iterations), corrected_states(5*j - 3, 1:KF_iterations), 'red');
        o = plot(predicted_opt_states(4*j - 3, 1:KF_iterations), predicted_opt_states(4*j - 2, 1:KF_iterations), 'green');
        title(['Car ', int2str(j), ' : Position']);    xlabel('x_1 [m]');    ylabel('x_2 [m]');
        legend([p c o], 'Predicted', 'Filtered', 'Optimization', 'Location', 'Best');
    end
end

if(mode_output(5) == 1)
    i = i + 1;    figure(i);    count = 0;

    % Plot Position [x vs. t]
    for j = 1:num_of_cars
        for k = 1:2
            count = count + 1;
            subplot(num_of_cars, 2, count);        grid;        hold on;
            p = plot((1:KF_iterations)*T, predicted_states(5*j - 5 + k, 1:KF_iterations), 'blue');
            c = plot((1:KF_iterations)*T, corrected_states(5*j - 5 + k, 1:KF_iterations), 'red');
            o = plot((1:KF_iterations)*T, predicted_opt_states(4*j - 4 + k, 1:KF_iterations), 'green');
            title(['Car ', int2str(j), ' : x', int2str(k), ' wrt Time']);    xlabel('Time [s]');    ylabel(['x', int2str(k)]);
            %legend([p c o], 'Predicted', 'Filtered', 'Optimization', 'Location', 'Best');
        end
    end
end

if(mode_output(6) == 1)
    i = i + 1;    figure(i);

    % Plot Velocity [MPH]
    for j = 1:num_of_cars
```

```
        subplot(num_of_cars, 1, j);       grid;       hold on;
        p = plot((1:KF_iterations)*T, predicted_states(5*j - 1, 1:KF_iterations)*h.*gear_ratios(j, 1:KF_iterations)*(3600/1609.344), 'blue');
        c = plot((1:KF_iterations)*T, corrected_states(5*j - 1, 1:KF_iterations)*h.*gear_ratios(j, 1:KF_iterations)*(3600/1609.344), 'red');
        o = plot((1:KF_iterations)*T, predicted_opt_states(4*j - 1, 1:KF_iterations), 'green');
        title(['Car ', int2str(j), ' : Velocity wrt Time']);    xlabel('Time [s]');    ylabel('V [MPH]');
        legend([p c o], 'Predicted', 'Filtered', 'Optimization', 'Location', 'Best');
    end
end

if(mode_output(7) == 1)
    i = i + 1;    figure(i);

    % Plot Velocity [m/s]
    for j = 1:num_of_cars
        subplot(num_of_cars, 1, j);    grid;    hold on;
        p = plot((1:KF_iterations)*T, predicted_states(5*j - 1, 1:KF_iterations).*gear_ratios(j, 1:KF_iterations)*h, 'blue');
        c = plot((1:KF_iterations)*T, corrected_states(5*j - 1, 1:KF_iterations).*gear_ratios(j, 1:KF_iterations)*h, 'red');
        o = plot((1:KF_iterations)*T, predicted_opt_states(4*j - 1, 1:KF_iterations)*1609.344/3600, 'green');
        title(['Car ', int2str(j), ' : Velocity wrt Time']);    xlabel('Time [s]');    ylabel('V [m/s]');
        legend([p c o], 'Predicted', 'Filtered', 'Optimization', 'Location', 'Best');
    end
end

if(mode_output(8) == 1)
    if(num_of_cars > 1)
        i = i + 1;    figure(i);
        count = 0;

        if(num_of_cars > 3)       rows = num_of_cars*(num_of_cars - 1)/4;       cols = 2;
        else       rows = num_of_cars*(num_of_cars - 1)/2;       cols = 1;
        end

        % Plot Distance between Cars
        for j = 1:(num_of_cars - 1)
            for k = (j + 1):num_of_cars
                predicted_distance = sqrt(( predicted_states(5*j - 4, 1:KF_iterations) - predicted_states(5*k - 4, 1:KF_iterations) ).^2 + (
predicted_states(5*j - 3, 1:KF_iterations) - predicted_states(5*k - 3, 1:KF_iterations) ).^2);
                corrected_distance = sqrt(( corrected_states(5*j - 4, 1:KF_iterations) - corrected_states(5*k - 4, 1:KF_iterations) ).^2 + (
corrected_states(5*j - 3, 1:KF_iterations) - corrected_states(5*k - 3, 1:KF_iterations) ).^2);
                optimization = sqrt(( predicted_opt_states(4*j - 3, 1:KF_iterations) - predicted_opt_states(4*k - 3, 1:KF_iterations) ).^2 + (
predicted_opt_states(4*j - 2, 1:KF_iterations) - predicted_opt_states(4*k - 2, 1:KF_iterations) ).^2);

                count = count + 1;
                subplot(rows, cols, count);
                hold on;           grid;
                p = plot((1:KF_iterations)*T, predicted_distance(1:KF_iterations), 'blue');
                c = plot((1:KF_iterations)*T, corrected_distance(1:KF_iterations), 'red');
                o = plot((1:KF_iterations)*T, optimization(1:KF_iterations), 'green');
                collision = plot([0 KF_iterations*T], ( safe_distance(j) + safe_distance(k) )*[1 1], 'black');
                title(['Distance b/w Cars ', int2str(j), ' and ', int2str(k) ' wrt Time']);    xlabel('Time [s]');    ylabel('Distance [m]');
                legend([p c o collision], 'Predicted', 'Filtered', 'Optimization', 'Collision Threshold', 'Location', 'Best');
            end
        end
    end
end

if(mode_output(9) == 1)
    i = i + 1;    figure(i);

    % Plot Mass of Air in Intake Manifold
    for j = 1:num_of_cars
        subplot(num_of_cars, 1, j);       grid;    hold on;
        p = plot((1:KF_iterations)*T, predicted_states(5*j - 2, 1:KF_iterations), 'blue');
        c = plot((1:KF_iterations)*T, corrected_states(5*j - 2, 1:KF_iterations), 'red');
        title(['Car ', int2str(j), ' : Mass of Air in Intake Manifold wrt Time']);    xlabel('Time [s]');    ylabel('m_a [kg]');
        legend([p c], 'Predicted', 'Filtered', 'Location', 'Best');
    end
end

if(mode_output(10) == 1)
    i = i + 1;    figure(i);
    % Plot Engine Angular Velocity
    for j = 1:num_of_cars
        subplot(num_of_cars, 1, j);    grid;    hold on;
        p = plot((1:KF_iterations)*T, predicted_states(5*j - 1, 1:KF_iterations), 'blue');
        c = plot((1:KF_iterations)*T, corrected_states(5*j - 1, 1:KF_iterations), 'red');
        title(['Car ', int2str(j), ' : Engine Angular Velocity wrt Time']);    xlabel('Time [s]');    ylabel('\omega_e [rad/s]');
        legend([p c], 'Predicted', 'Filtered', 'Location', 'Best');
    end
end

if(mode_output(11) == 1)
    i = i + 1;    figure(i);

    % Plot Brake Torque
    for j = 1:num_of_cars
        subplot(num_of_cars, 1, j);    grid;    hold on;
        p = plot((1:KF_iterations)*T, predicted_states(5*j - 0, 1:KF_iterations), 'blue');
        c = plot((1:KF_iterations)*T, corrected_states(5*j - 0, 1:KF_iterations), 'red');
        title(['Car ', int2str(j), ' : Brake Torque wrt Time']);    xlabel('Time [s]');    ylabel('T_b [N*m]');
        legend([p c], 'Predicted', 'Filtered', 'Location', 'Best');
    end
end

if(mode_output(12) == 1)
    i = i + 1;    figure(i);

    for j = 1:num_of_cars        for k = 1:KF_iterations            gear_ratios(j, k) = find(Rg == gear_ratios(j, k));        end,    end

    % Plot Gear
    for j = 1:num_of_cars
        subplot(num_of_cars, 1, j);    grid;    hold on;
        p = plot((1:KF_iterations)*T, gear_ratios(j, 1:KF_iterations), 'red');
        title(['Car ', int2str(j), ' : Gear wrt Time']);    xlabel('Time [s]');    ylabel('Gear');
    end
end

if(mode_output(13) == 1)
    i = i + 1;    figure(i);

    % Plot Phase Plot
    for j = 1:num_of_cars
        subplot(num_of_cars, 1, j);        grid;
        hold on;
        p = plot3(predicted_states(5*j - 2, 1:KF_iterations), predicted_states(5*j - 1, 1:KF_iterations), (1:KF_iterations)*T, 'blue');
        c = plot3(corrected_states(5*j - 2, 1:KF_iterations), corrected_states(5*j - 1, 1:KF_iterations), (1:KF_iterations)*T, 'red');
        title(['Car ', int2str(j), ' : Engine Angular Velocity vs. Mass of Air in the Intake Manifold wrt Time [s]']);    xlabel('m_a [kg]');
ylabel('\omega_e [rad/s]');        zlabel('Time [s]');
        legend([p c], 'Predicted', 'Filtered', 'Location', 'Best');
    end

    %plot(corrected_states(5*j - 2, 1:KF_iterations/10:KF_iterations), corrected_states(5*j - 1, 1:KF_iterations/10:KF_iterations), 'O red');
end

if(mode_output(14) == 1)
    i = i + 1;    figure(i);    grid;    hold on;

    % Plot Correction Iterations
    plot((1:KF_iterations)*T, correction_iterations(1:KF_iterations, 1), 'red');
    m = plot([1 KF_iterations]*T, [mean(correction_iterations(1:KF_iterations, 1)) mean(correction_iterations(1:KF_iterations, 1))], 'green');
    title(['Corrector Iterations wrt Time']);    xlabel('Time [s]');    ylabel('Corrector Iterations');
    legend([m], 'Mean', 'Location', 'Best');
end

toc
```