INTEGRATING

COLLISION AVOIDANCE, LANE KEEPING,

AND CRUISE CONTROL

WITH AN OPTIMAL CONTROLLER

AND FUZZY CONTROLLER

WILLIAM K. GREFE

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 Computer Engineering
APPROVED:

Pushkin Kachroo
Krishnan Ramu
Daniel J. Stilwell

April 29, 2005 Blacksburg, Virginia

Keywords: Collision Avoidance, Smart Vehicles, Autonomous, Optimal control

INTEGRATING

COLLISION AVOIDANCE, LANE KEEPING,

AND CRUISE CONTROL

WITH AN OPTIMAL CONTROLLER

AND FUZZY CONTROLLER

WILLIAM K. GREFE

ABSTRACT

This thesis presents collision avoidance integrated with lane keeping and adaptive cruise control for a car. Collision avoidance is the ability to avoid obstacles that are in the vehicle's path, without causing damage to the obstacle or car. There are three types of collision avoidance controllers, passive, active, and semi-active. This thesis is designed using active collision avoidance controllers.

There are two controllers developed for collision avoidance in this paper. They are an optimal controller and a fuzzy controller. The optimal vehicle trajectory, which maximizes the distance to an obstacle and changes lanes, is derived. The optimal collision avoidance controller is a closed loop controller; with the decisions based on the current state. The fuzzy controller makes decisions based on the system rules. A simulation environment was created to compare these two controllers as viable solutions for collision avoidance.

The environment uses MATLAB/Simulink for simulation of the vehicle as well as the optimal and fuzzy controllers. The simulation incorporates system blocks of the kinematics of a car, navigation, states, control law, and velocity controller. Once the controllers are fully developed and tested in the simulation environment, they are implemented and tested on the platform vehicle. This verifies the real world performance of the controllers.

The platform vehicle is a modified radio controlled car. This car is completely autonomous. The car has onboard sensors that allow it to follow a white piece of tape as well as detect obstacles.

DEDICATIONS

Many thanks go to my adviser, Dr. Pushkin Kachroo, for allowing me to work on such an interesting topic.

Thank you to my family and friends for standing by me.

ACKNOWLEDGMENTS

Many thanks go to my adviser, Dr. Pushkin Kachroo, for allowing me to work on such an interesting topic.

Thank you to my family and friends for standing by me.

Also would like to thank the following companies for their donations

Hyde Park, Banner Engineering, Samtec, AMP/TYCO, Maxim, and Power One

TABLE OF CONTENTS

Page

LIST OF FIGURES

Figure **Page**

CHAPTER I

INTRODUCTION

Collision avoidance systems for cars are designed to reduce the number of accidents and fatalities on the roadways and highways. Safety systems are designed to helped safe lives like the seat belt when worn properly and the air bag.

In the United States in 2003, there were 42,643 people killed from motor vehicle accidents and 2,889,000 people injured in motor vehicle accidents. With any of the vehicle, accidents if there were a passive system to avoid an obstacle; it would have greatly decreased the number of fatalities and injuries. Driving is not a right but a privilege, we should treat driving seriously. The reason for researching collision avoidance is so the fraction of a second where a driver is not paying attention, a passive system could be implemented to keep the driver, passengers, and others safe. [9]

Currently, the marketplace is starting to see technology to help avoid motor vehicle collisions. There are three different types of collision avoidance systems: passive, active, and semi-active. Passive collision avoidance systems are typically audio or visual alarms indicating the potential for a collision. Active collision avoidance systems take control of the vehicle by controlling the throttle, braking, and steering to avoid or minimize a collision. Semi-active collision avoidance systems minimize the impact the collision has on the driver. These systems are starting to be available in the market today and the near future.

Passive collision avoidance systems that are either on the market or soon to be on the market Delphi has developed: lane change warning alarm, a vision system that detects roadway markers and warns of unintended lane changes. [16] Roadway Departure warning system that mimics the sound of rumble strips. The sound comes from the side toward which the car veers. [18] Blind Spot collision avoidance system, which is a combination of radar and vision systems to help a driver better sense his crash envelope. [16]

Active collision avoidance systems that are either on the market or soon to be on the market Delphi has developed: Forewarn Collision Avoidance Systems uses sensors strategically located around the vehicle to collect data and recognize hazards within their detection zone. Forewarn can then not only communicate when driver intervention is necessary, but take automatic action when appropriate. Smart Cruise Control detects traffic ahead, and using throttle control and limited braking, maintains a driver-selectable gap. [17] Another example is Jaguar's adaptive cruise control system. Jaguar's description of this system is: "Radar-based Adaptive Cruise Control (ACC) constantly maintains a comfortable gap to the car ahead, taking 40 individual measurements during each horizontal scan. ACC also offers Forward Alert, which provides a timely audible warning if traffic ahead starts to slow down. Adaptive Cruise Control is available on all models with automatic transmission." [4]

Semi-active collision avoidance systems that are either on the market or soon to be on the market Ford has developed: A video monitor embedded in the dashboard of a Ford Explorer concept vehicle shows a vehicle ahead of it, with a green box around it. As the concept vehicle gets too close for safety, the box turns to red, which senses that a crash may be imminent. This makes the seat belts tighten automatically and a computerized voice beckons, "Warning."

1.1 Controllers

The two controllers developed for collision avoidance are an optimal controller and a fuzzy controller. The optimal controller is an open loop controller; designed to make the obstacle avoidance decisions (control) by determining the optimal (minimal) cost based on the current state of the system. The controller is created using parameters to allow easy modification (tuning) of the operation of the controller. All possible useful vehicle states are passed to the optimal controller function to allow the incorporation of additional features easily. The distance to an obstacle in front of the vehicle, the velocity of an object behind the vehicle, the x-axis location of the vehicle and the lane the vehicle is in are the only states currently used in the optimal controller.

According to Gupta [10], "*Fuzzy logic*, which was introduced by Lotfi A. Zadeh in 1965 (Zadeh 1965), is a powerful tool for modeling human thinking and cognition. Instead of bivalent propositions, fuzzy logic systems deal with reasoning and multi-valued sets, stored rules, and estimated sampled functions from linguistic input to linguistic output."

The fuzzy controller makes decisions based on the system rules. These rules are based on natural language. The decisions, formed from membership functions shape the controller action. Membership functions can be looked on as filters, taking in certain inputs and sending out certain outputs to form the desired response.

1.2 Simulation

The simulation environment was created using MATLAB/Simulink because it is heavily used in science, industry, education and government. MATLAB/Simulink is used in a wide range of applications, including signal and image processing, communications, control design, test and measurement, financial modeling and analysis, and computational biology. MATLAB/Simulink is a high-level technical computing language and object orientated environment for algorithm development, data visualization, data analysis, and numerical computation. Add-on toolboxes extend the MATLAB/Simulink environment to solve particular classes of problems in various application areas. MATLAB/Simulink allows the development of a solution to technical computing problems faster than with traditional programming languages, such as C, C++, and Fortran. The easy of development along with the extensive toolboxes and functions available were the major reasons for selecting MATLAB/Simulink as the simulation environment.

The simulation environment starts as an overview of the vehicle controls and kinematic emulation of the vehicle in a Simulink model. This model incorporates the high-level system blocks representing, the kinematics of a car, navigation, states, control law, graphic system update and velocity controller. The model also contains parameters displays, constant blocks, as well as subsystems for two additional cars. The goals for the simulation environment were to be able to add obstacles, vehicles, and to change the path easily. The environment road map was created based on a grid capable of arbitrary lanes and different paths.

The road map grid contains binary data representing the road and the white lines by zeros and ones respectively. The line sensing sensors use this road map grid to detect the line beneath car. Functions were created to draw the lines and arcs, which form the path the vehicle, would follow on the road map. These functions allow the road map to be modified easily. This environment is setup to allow the extension to other moving obstacles not just vehicles following the path as well as changing the obstacle avoidance controller to another type or design. The reason for having three lanes in the simulation environment rather than two lanes is to test the controllers with a more complex arrangement of obstacles and traffic.

The simulation environment is flexible and if someone is interested in testing out another collision avoidance algorithm, the controller is a single function that can be replaced/changed easily. This was confirmed when, after implementing the optimal control first, the modifications needed to implement the fuzzy controller only consisted of writing the fuzzy controller.

1.3 Car

Figure 1 Platform for Obstacle Avoidance (Car)

The platform used as a final test of the obstacle avoidance controller is the car. The main hardware for the car is TI's TMS320C6000 DSP, IR and magnetic sensor board, PIC Microcontroller board, and the range finder. The sensors the car utilizes are the Fairchild Semiconductor QRD1114 IR emitter/detector pairs, the HAL506UA-E Hall effect sensors (magnetic sensors), and the Sharp GP2D12 range finder that is also IR based. The IR emitter/detector pairs locate the white tape that is on the black road surface. The magnetic sensors detect the magnetic line that is beneath the roadway by detecting the presence of a magnetic south pole on a series of magnets. Finally, the range finder sensor is used to detect any obstacles in front of the car. This IR range finder is unique in the way that it emits a modulated 40kHz infrared pulse. This way the range finder is less susceptible to ambient light.

1.4 Highlights, Contributions, and Outline

The following are the highlights and contributions made during this thesis work:

- A simulation environment to test various collision avoidance algorithms.
- Two controllers for collision avoidance optimal controller and fuzzy controller were developed.
- Development of the optimal lane change trajectory.

- Documentation of the simulation and controllers for collision avoidance.

- Successful testing of the collision avoidance controller on the car.

Outline of the thesis

CHAPTER II – Literature Review, discusses prior work related to collision avoidance

CHAPTER III – Modeling and Applying Collision Avoidance, begins discussing the optimal lane change trajectory and cost given various scenarios. The simulation is then discussed in detail regarding the optimal and fuzzy controllers, including examples. Next, a section discusses how to modify the simulation environment. Lastly, the platform kinematics, operation, collision avoidance, and digital signal processing source code is discussed.

CHAPTER IV– Results, discusses the simulation results and the car results with photographs of the car avoiding obstacles.

CHAPTER V – Conclusion

CHAPTER VI – Summary, and discussion of future work

CHAPTER II

REVIEW OF LITERATURE

There has been much interesting research done for collision avoidance. Below are descriptions of a few relevant research topics.

In [11], there is a formalization of human centered design principles and illustrate their application using an automation system that assists drivers to avoid unsafe lane departures. This paper recognizes the importance of the human-computer interaction as related to collision avoidance. That the safety and effectiveness of a collision avoidance system in an automobile is not only related to how well the automated system works, but how the entire human-computer system performs.

> "Technological advances have made plausible the design of automated systems that share responsibility with a human operator. The decision to use automation to assist or replace a human operator in safety-critical tasks must account for not only the technological capabilities of the sensor and control subsystems, but also the autonomy, capabilities, and preferences of the human operator. By their nature, such human-centered automation problems have multiple attributes." [11]

Making sensor-friendly vehicle and roadway systems would improve on the abilities of collision avoidance systems. In [15], work was done to show the improvements possible with complementary signal sensor and reflector technologies. These technologies can assist or replace single vehicle-based systems. The four most promising technologies passive license plates with enhanced radar return, roadside obstacle mounted radar-reflecting corner cubes, fluorescent paint for lane and obstacle marking, and light emitting diode brake light messaging are discussed especially on their improvement to the signal to noise ratio for the collision avoidance sensors. These sensor friendly systems should significantly improve collision avoidance systems.

In [12], a fuzzy logic enhanced car navigation and collision avoidance system has been designed. Essentially, the control of a car in this system is based on the flexible use of a fuzzy trajectory mapping unit that enables smooth trajectory management independent of

car's initial position or position of the destination. This was done with a fuzzy controller consisting of 28 rules and a state machine containing 4 states. For performing more demanding tasks, however, additional blocks of "intelligence" are required. The latter is quite possible thanks to the modular structure of the control system responsible for different task in separate without jeopardizing overall performance.

In [13], a multi-sensor collision avoidance system (CAS) is described in this paper. Measurements from radar, vision and sonar are combined using a fusion scheme that utilizes fuzzy clustering and estimation techniques to estimate relative motion between the vehicles. Fuzzy logic is used to generate audiovisual warnings for the driver. It also implements a throttle relaxer and brake actuator to slow the vehicle down. A prototype was implemented on a Humvee.

A fuzzy collision avoidance system for a fixed obstacle was designed and tested in [14]. This work describes a fuzzy trajectory controller with over 300 rules that is used with a specially designed car-driving robot. The rules were created based on the trajectories various drivers used to avoid a fixed obstacle. A laser was used as the obstacle detection device. While the robot and fuzzy controller worked successfully about 60% of the time, the reasons for failure are understood.

Using Game Theory as a basis for collision avoidance is a subject of much research. One example would be from [3]. This work describes mathematically how an evader (car) can avoid a pursuer (moving obstacle or static obstacle), using non-cooperative game theory. There is no path to follow or limitation as to where the vehicle can go to avoid the obstacle, outside its own physical path restrictions.

Another thesis [8] uses several methods discrete deterministic, stochastic, and non-cooperative dynamic models. One of the key conclusions is, "differential pursuit-evasion games are complicated to analyze even under the best circumstances, and that the introduction of realistic complexity makes them formally inflexible." [8] And, "Although differential game theory provides a framework for describing the important features of pursuit-evasion contests, and a set of normative results concerning optimal strategies in simple cases, it cannot generally provide optimal strategies for practically pursuit evasion problems, nor can it show how strategies can be implemented in a real control system subjected to limited sensory capacities, sensory and motor noise, component failure and constraints on processing speed and accuracy."[8]

Neural Control would be another viable solution to collision avoidance. Neural Control is using a network of simulated neurons, which learn the correct behavior when trained with a set of desired scenarios. MAMMOTH uses ALVINN's Neural Network for road following and extends it for use in cross-country applications. The motivation behind MAMMOTH is two parts: "vegetation is difficult to characterize using simple physical models, and multiple features are needed to distinguish vegetation from other natural objects. MAMMOTH uses a neural network to learn a model for vegetation by associating video data with a human-driven classification of terrain." [19]

The prototype vehicle platform used in this thesis is built on top of work done in [7]. This work included the lateral control of the car following the white piece of tape, which symbolized a lane. A MATLAB simulation was created for a car as well as the DSP software in the car to follow the white piece of tape. In the simulation $\dot{\theta}_p$ is never from the discrete version, it is always a continuous function which is never available from the real car, since the IR sensors work in discrete form. When using the calculated c(s) from the simulated sensors, the control was very unstable. The only solution that seemed to work was to disable the calculation of c(s) from the sensors. This work gave the starting parameters for the lateral controller for the car.

CHAPTER III

MODELING AND APPLYING COLLISION AVOIDANCE

3.1 Determining Optimal Lane Change Trajectory for the Car

In this section, the optimal lane trajectory is derived using Variational Calculus. This optimal trajectory will the be used by both the optimal and fuzzy lane change controllers. The starting point for this derivation are the vehicle kinematic state equations.

3.1.1 Dynamic System of the Car Model and State Equations

The figure of the car below shows the symbols being used in the dynamics of the car.

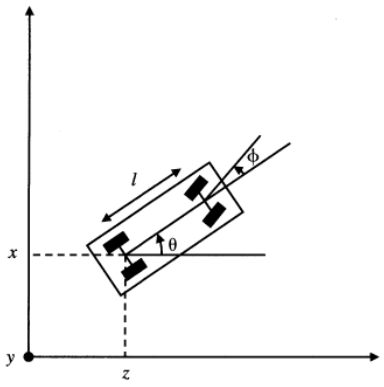


Figure 2 Coordinate System for the Car

The location and orientation of the car on a coordinate system can be described by using the State Equations 3.1-1. The states are described by: $x_1(t)$ is the x value of the center of the rear axle of the car, $x_2(t)$ is the y value of the center of the rear axle of the car, $x_3(t)$ is the heading of the car, $x_4(t)$ is the steering angle for the car, l is the length between the axles, u_1 is the car velocity, and $u_2(t)$ is the steering angle velocity. The following are the state equations and the state constraint for the car's kinematics.

$$\dot{\boldsymbol{x}} = \boldsymbol{A}(\boldsymbol{x}(t), \boldsymbol{u}(t))$$

$$\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \\ \dot{x}_3(t) \\ \dot{x}_4(t) \end{bmatrix} = \begin{bmatrix} \cos(x_3(t)) \\ \sin(x_3(t)) \\ \dfrac{\tan(x_4(t))}{l} \\ 0 \end{bmatrix} u_1 + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} u_2(t) \qquad [3.1\text{-}1]$$

$$-\frac{\pi}{3} \le x_4 \le \frac{\pi}{3} \qquad [3.1\text{-}2]$$

The state constraint 3.1-2 means the steering angle cannot exceed $\pm 60^{o}$.

3.1.2 Cost Function

The obstacle avoidance goal is to maximize the distance L between the car and the obstacle, minimize the control output u_2 and maximize velocity u_1. The cost function is defined where a minimal cost is optimal. This is accomplished by multiplying the distance squared and velocity u_1 squared by -1. The cost function is then:

$$J = \int_{t_0}^{t_f} -L^2(t) + 10u_2^2(t) - u_1^2 dt \qquad [3.1\text{-}3]$$

For this example, u_1 is a constant velocity so it will not be considered an input to the system. An optimal path is calculated using the kinematics of the car from the state equations and the cost function that describes what is being optimized.

3.1.3 Hamiltonian, Kinematics & Optimal Lane Change Trajectory

Hamiltonian was used because it is a convenient form to express the necessary conditions for optimality based on the principle of optimality.

The Hamiltonian needs to be calculated, so an optimal trajectory can be obtained. The Hamiltonian is defined in equation 3.1-4 as a function of the state equations and the cost function.

$$\boldsymbol{H}(\boldsymbol{x}(t),\boldsymbol{u}(t),\boldsymbol{p}(t),t) \triangleq \boldsymbol{g}(\boldsymbol{x}(t),\boldsymbol{u}(t),t) + \boldsymbol{p}^{T}(t)[\boldsymbol{a}(\boldsymbol{x}(t),\boldsymbol{u}(t),t)] \qquad [3.1\text{-}4]$$

Substituting equations 3.1-1and 3.1-3 into the Hamiltonian 3.1-4 yields:

$$\begin{aligned}\boldsymbol{H}(\boldsymbol{x}(t),\boldsymbol{u}(t),\boldsymbol{p}(t),t) = {} & -\boldsymbol{x}_1^2(t) - \boldsymbol{x}_2^2(t) + 10\boldsymbol{u}_2^2(t) + \boldsymbol{u}_1\boldsymbol{p}_1(t)\cos(\boldsymbol{x}_3(t)) + \\ & \boldsymbol{u}_1\boldsymbol{p}_2(t)\sin(\boldsymbol{x}_3(t)) + \frac{\boldsymbol{u}_1\boldsymbol{p}_3(t)\tan(\boldsymbol{x}_4(t))}{\boldsymbol{l}} + \boldsymbol{u}_2(t)\boldsymbol{p}_4(t)\end{aligned} \qquad [3.1\text{-}5]$$

The necessary conditions for optimal control expressed using the Hamiltonian 3.1-4 are:

$$\left.\begin{aligned}\dot{\boldsymbol{x}}^*(t) &= \frac{\partial \boldsymbol{H}}{\partial \boldsymbol{p}}(\boldsymbol{x}^*(t),\boldsymbol{u}^*(t),\boldsymbol{p}^*(t),t) \\ \dot{\boldsymbol{p}}^*(t) &= -\frac{\partial \boldsymbol{H}}{\partial \boldsymbol{x}}(\boldsymbol{x}^*(t),\boldsymbol{u}^*(t),\boldsymbol{p}^*(t),t) \\ 0 &= \frac{\partial \boldsymbol{H}}{\partial \boldsymbol{u}}(\boldsymbol{x}^*(t),\boldsymbol{u}^*(t),\boldsymbol{p}^*(t),t)\end{aligned}\right\} \text{for all } t \in [t_0, \boldsymbol{t}_f] \qquad [3.1\text{-}6]$$

Appling the necessary conditions of equations 3.1-6 to the Hamiltonian 3.1-5 yields the following optimal state equations, costate equations, and control:

$$\begin{aligned}\dot{\boldsymbol{x}}_1^*(t) &= \boldsymbol{u}_1\cos(\boldsymbol{x}_3^*(t)) \\ \dot{\boldsymbol{x}}_2^*(t) &= \boldsymbol{u}_1\sin(\boldsymbol{x}_3^*(t)) \\ \dot{\boldsymbol{x}}_3^*(t) &= \frac{\boldsymbol{u}_1\tan(\boldsymbol{x}_4^*(t))}{\boldsymbol{l}} \\ \dot{\boldsymbol{x}}_4^*(t) &= -\frac{1}{20}\boldsymbol{p}_4^*(t)\end{aligned} \qquad [3.1\text{-}7]$$

$$\dot{p}_1^*(t) = 2x_1^*(t)$$
$$\dot{p}_2^*(t) = 2x_2^*(t)$$
$$\dot{p}_3^*(t) = u_1 p_1^*(t)\sin(x_3^*(t)) - u_1 p_2^*(t)\cos(x_3^*(t)) \qquad [3.1\text{-}8]$$
$$\dot{p}_4^*(t) = -\frac{u_1 p_3^*(t)(1+\tan^2(x_4^*(t)))}{l}$$

$$u_2^*(t) = -\frac{1}{20} p_4^*(t) \qquad [3.1\text{-}9]$$

3.1.4 Derive the Optimal Trajectory for Lane Changing

The first step in deriving the optimal trajectory for lane changing was to apply the numerical method of variation of extremals. Even though this method does not allow for the state constraints, it was used to see how close the solution could get to the optimal trajectory. The variation of extremals algorithm from [6] is shown below.

The steps required to carry out the variation of extremals method:

1. Form the reduced differential equations by solving $\frac{dH}{du} = 0$ for ***u(t)*** in terms of ***x(t)***, ***p(t)***, and substituting in the state and costate equations [which then contain only ***x(t)***, ***p(t)***, and t].

2. Guess $p^{(0)}(t_0)$, an initial value for the costate, and set the iteration index ***i*** to zero.

3. Using $p(t_0) = p^{(i)}(t_0)$ and $x(t_0) = x_0$ as initial conditions, integrate the reduced state-costate equations and the influence function equations 3.1-10.

$$\frac{d}{dt}\left[P_x(p^{(i)}(t_0),t\right] = \left[\frac{\partial^2 H}{\partial p \partial x}(t)\right] P_x(p^{(i)}(t_0),t) + \left[\frac{\partial^2 H}{\partial p^2}(t)\right]_i P_p(p^{(i)}(t_0),t)$$
$$\frac{d}{dt}\left[P_p(p^{(i)}(t_0),t\right] = \left[\frac{\partial^2 H}{\partial x^2}(t)\right]_i P_x(p^{(i)}(t_0),t) + \left[-\frac{\partial^2 H}{\partial x \partial p}(t)\right]_i P_p(p^{(i)}(t_0),t) \qquad [3.1\text{-}10]$$

with initial conditions:

$$\boldsymbol{P}_x(\boldsymbol{p}^{(i)}(\boldsymbol{t}_0),\boldsymbol{t}_0)=\left.\frac{\boldsymbol{dx}(\boldsymbol{t}_0)}{\boldsymbol{dp}(\boldsymbol{t}_0)}\right|_{\boldsymbol{p}^{(i)}(\boldsymbol{t}_0)}=0 \text{ (the n x n zero matrix)}$$
$$\boldsymbol{P}_p(\boldsymbol{p}^{(i)}(\boldsymbol{t}_0),\boldsymbol{t}_0)=\left.\frac{\boldsymbol{dp}(\boldsymbol{t}_0)}{\boldsymbol{dp}(\boldsymbol{t}_0)}\right|_{\boldsymbol{p}^{(i)}(\boldsymbol{t}_0)}=\boldsymbol{I} \text{ (the n x n identity matrix)} \qquad [3.1\text{-}11]$$

from $\boldsymbol{t_0}$ to $\boldsymbol{t_f}$. Store only the values $\boldsymbol{p^{(i)}(t_f)}$, $\boldsymbol{x^{(i)}(t_f)}$, and the n x n matrices $\boldsymbol{P_p(p^{(i)}(t_0),t_f)}$ and $\boldsymbol{P_x(p^{(i)}(t_0),t_f)}$.

4. Check to see if the termination criterion $\left\|\boldsymbol{p}^{(i)}(\boldsymbol{t}_f)-\frac{\partial \boldsymbol{h}(\boldsymbol{x}^{(i)}(\boldsymbol{t}_f)}{\partial \boldsymbol{x}}\right\|<\gamma$ is satisfied. If it is, use the final iterate of $p^{(i)}(t_0)$ to reintegrate the state and costate equations and print out (or graph) the optimal trajectory. If the stopping criterion is not satisfied, use the iteration equation 3.1-12 to determine the value for $p^{(i+1)}(t_0)$, increase I by one, and return to step 3.

$$\boldsymbol{p}^{(i+1)}(\boldsymbol{t}_0)=\boldsymbol{p}^{(i)}(\boldsymbol{t}_0)-\left(\boldsymbol{P}_x^i(\boldsymbol{p}(\boldsymbol{t}_0),\boldsymbol{t}_f)\right)^{-1}(\boldsymbol{x}^i(\boldsymbol{t}_f)-\boldsymbol{x}(\boldsymbol{t}_f)) \qquad [3.1\text{-}12]$$

$\boldsymbol{P_p(p^{(i)}(t_0),\ t_f)}$ in equation 3.1-12, the n x n costate influence function matrix evaluated at $\boldsymbol{t=t_f}$ is shown below.

$$\boldsymbol{P}_p(\boldsymbol{p}^{(i)}(\boldsymbol{t}_0),\boldsymbol{t})\triangleq\left[\begin{array}{cccc}\frac{\partial \boldsymbol{p}_1(\boldsymbol{t})}{\partial \boldsymbol{dp}_1(\boldsymbol{t})} & \frac{\partial \boldsymbol{p}_1(\boldsymbol{t})}{\partial \boldsymbol{p}_2(\boldsymbol{t})} & \cdots & \frac{\partial \boldsymbol{p}_1(\boldsymbol{t})}{\partial \boldsymbol{p}_2(\boldsymbol{t})}\\ \vdots & \vdots & \vdots & \vdots\\ \frac{\partial \boldsymbol{p}_n(\boldsymbol{t})}{\partial \boldsymbol{p}_1(\boldsymbol{t})} & \frac{\partial \boldsymbol{p}_n(\boldsymbol{t})}{\partial \boldsymbol{p}_2(\boldsymbol{t})} & \cdots & \frac{\partial \boldsymbol{p}_n(\boldsymbol{t})}{\partial \boldsymbol{p}_n(\boldsymbol{t})}\end{array}\right]_{\boldsymbol{p}^{(i)}(\boldsymbol{t}_0)} \qquad [3.1\text{-}13]$$

$\boldsymbol{P_x(p^{(i)}(t_0),\ t_f)}$ in equation 3.1-12, the n x n state influence function matrix evaluated at $\boldsymbol{t=t_f}$ is shown below.

$$\boldsymbol{P}_x(\boldsymbol{p}^{(i)}(\boldsymbol{t}_0),\boldsymbol{t})\triangleq\left[\begin{array}{cccc}\frac{\partial \boldsymbol{x}_1(\boldsymbol{t})}{\partial \boldsymbol{dp}_1(\boldsymbol{t})} & \frac{\partial \boldsymbol{x}_1(\boldsymbol{t})}{\partial \boldsymbol{p}_2(\boldsymbol{t})} & \cdots & \frac{\partial \boldsymbol{x}_1(\boldsymbol{t})}{\partial \boldsymbol{p}_2(\boldsymbol{t})}\\ \vdots & \vdots & \vdots & \vdots\\ \frac{\partial \boldsymbol{x}_n(\boldsymbol{t})}{\partial \boldsymbol{p}_1(\boldsymbol{t})} & \frac{\partial \boldsymbol{x}_n(\boldsymbol{t})}{\partial \boldsymbol{p}_2(\boldsymbol{t})} & \cdots & \frac{\partial \boldsymbol{x}_n(\boldsymbol{t})}{\partial \boldsymbol{p}_n(\boldsymbol{t})}\end{array}\right]_{\boldsymbol{p}^{(i)}(\boldsymbol{t}_0)} \qquad [3.1\text{-}14]$$

The notation $[\ ]_i$ means the enclosed terms are evaluated on the i^{th} trajectory.

Substituting the Hamiltonian equation 3.1-5 into 3.1-10 results in the differential influence function matrix $\boldsymbol{P_x}$ shown below:

$$\dot{\boldsymbol{P}}_{\boldsymbol{x}}(\boldsymbol{t})=\begin{bmatrix} -\boldsymbol{u}_1\sin(\boldsymbol{x}_3(\boldsymbol{t}))\boldsymbol{P}_{\boldsymbol{x}(3,1)}(\boldsymbol{t}) & -\boldsymbol{u}_1\sin(\boldsymbol{x}_3(\boldsymbol{t}))\boldsymbol{P}_{\boldsymbol{x}(3,2)}(\boldsymbol{t}) & -\boldsymbol{u}_1\sin(\boldsymbol{x}_3(\boldsymbol{t}))\boldsymbol{P}_{\boldsymbol{x}(3,3)}(\boldsymbol{t}) & -\boldsymbol{u}_1\sin(\boldsymbol{x}_3(\boldsymbol{t}))\boldsymbol{P}_{\boldsymbol{x}(3,4)}(\boldsymbol{t}) \\ \boldsymbol{u}_1\cos(\boldsymbol{x}_3(\boldsymbol{t}))\boldsymbol{P}_{\boldsymbol{x}(3,1)}(\boldsymbol{t}) & \boldsymbol{u}_1\cos(\boldsymbol{x}_3(\boldsymbol{t}))\boldsymbol{P}_{\boldsymbol{x}(3,2)}(\boldsymbol{t}) & \boldsymbol{u}_1\cos(\boldsymbol{x}_3(\boldsymbol{t}))\boldsymbol{P}_{\boldsymbol{x}(3,3)}(\boldsymbol{t}) & \boldsymbol{u}_1\cos(\boldsymbol{x}_3(\boldsymbol{t}))\boldsymbol{P}_{\boldsymbol{x}(3,4)}(\boldsymbol{t}) \\ \dfrac{\boldsymbol{u}_1(1+\tan^2(\boldsymbol{x}_4(\boldsymbol{t}))\boldsymbol{P}_{\boldsymbol{x}(4,1)}(\boldsymbol{t})}{\boldsymbol{l}} & \dfrac{\boldsymbol{u}_1(1+\tan^2(\boldsymbol{x}_4(\boldsymbol{t}))\boldsymbol{P}_{\boldsymbol{x}(4,2)}(\boldsymbol{t})}{\boldsymbol{l}} & \dfrac{\boldsymbol{u}_1(1+\tan^2(\boldsymbol{x}_4(\boldsymbol{t}))\boldsymbol{P}_{\boldsymbol{x}(4,3)}(\boldsymbol{t})}{\boldsymbol{l}} & \dfrac{\boldsymbol{u}_1(1+\tan^2(\boldsymbol{x}_4(\boldsymbol{t}))\boldsymbol{P}_{\boldsymbol{x}(4,4)}(\boldsymbol{t})}{\boldsymbol{l}} \\ -\dfrac{1}{20}\boldsymbol{P}_{\boldsymbol{p}(4,1)} & -\dfrac{1}{20}\boldsymbol{P}_{\boldsymbol{p}(4,2)} & -\dfrac{1}{20}\boldsymbol{P}_{\boldsymbol{p}(4,3)} & -\dfrac{1}{20}\boldsymbol{P}_{\boldsymbol{p}(4,4)} \end{bmatrix}$$

Equation above is [3.1-15]

Substituting the Hamiltonian equation 3.1-5 into 3.1-10 results in the differential influence function matrix $\boldsymbol{P_p}$ shown below:

$$\begin{aligned} \dot{\boldsymbol{P}}_{\boldsymbol{p}(1,1)}(\boldsymbol{t}) &= 2\boldsymbol{P}_{\boldsymbol{x}(1,1)}(\boldsymbol{t}) \\ \dot{\boldsymbol{P}}_{\boldsymbol{p}(1,2)}(\boldsymbol{t}) &= 2\boldsymbol{P}_{\boldsymbol{x}(1,2)}(\boldsymbol{t}) \\ \dot{\boldsymbol{P}}_{\boldsymbol{p}(1,3)}(\boldsymbol{t}) &= 2\boldsymbol{P}_{\boldsymbol{x}(1,3)}(\boldsymbol{t}) \\ \dot{\boldsymbol{P}}_{\boldsymbol{p}(1,4)}(\boldsymbol{t}) &= 2\boldsymbol{P}_{\boldsymbol{x}(1,4)}(\boldsymbol{t}) \end{aligned} \qquad [3.1\text{-}16]$$

$$\begin{aligned} \dot{\boldsymbol{P}}_{\boldsymbol{p}(2,1)}(\boldsymbol{t}) &= 2\boldsymbol{P}_{\boldsymbol{x}(2,1)}(\boldsymbol{t}) \\ \dot{\boldsymbol{P}}_{\boldsymbol{p}(2,2)}(\boldsymbol{t}) &= 2\boldsymbol{P}_{\boldsymbol{x}(2,2)}(\boldsymbol{t}) \\ \dot{\boldsymbol{P}}_{\boldsymbol{p}(2,3)}(\boldsymbol{t}) &= 2\boldsymbol{P}_{\boldsymbol{x}(2,3)}(\boldsymbol{t}) \\ \dot{\boldsymbol{P}}_{\boldsymbol{p}(2,4)}(\boldsymbol{t}) &= 2\boldsymbol{P}_{\boldsymbol{x}(2,4)}(\boldsymbol{t}) \end{aligned} \qquad [3.1\text{-}17]$$

$$\begin{aligned} \dot{\boldsymbol{P}}_{\boldsymbol{p}(3,1)}(\boldsymbol{t}) &= \left(\boldsymbol{u}_1\boldsymbol{p}_1(\boldsymbol{t})\cos(\boldsymbol{x}_3(\boldsymbol{t})+\boldsymbol{u}_1\boldsymbol{p}_2(\boldsymbol{t})\sin(\boldsymbol{x}_3(\boldsymbol{t}))\right)\boldsymbol{P}_{\boldsymbol{x}(3,1)}(\boldsymbol{t})+\boldsymbol{u}_1\boldsymbol{P}_{\boldsymbol{p}(1,1)}(\boldsymbol{t})\sin(\boldsymbol{x}_3(\boldsymbol{t}))-\boldsymbol{u}_1\boldsymbol{P}_{\boldsymbol{p}(2,1)}\cos(\boldsymbol{x}_3(\boldsymbol{t})) \\ \dot{\boldsymbol{P}}_{\boldsymbol{p}(3,2)}(\boldsymbol{t}) &= \left(\boldsymbol{u}_1\boldsymbol{p}_1(\boldsymbol{t})\cos(\boldsymbol{x}_3(\boldsymbol{t})+\boldsymbol{u}_1\boldsymbol{p}_2(\boldsymbol{t})\sin(\boldsymbol{x}_3(\boldsymbol{t}))\right)\boldsymbol{P}_{\boldsymbol{x}(3,2)}(\boldsymbol{t})+\boldsymbol{u}_1\boldsymbol{P}_{\boldsymbol{p}(1,2)}(\boldsymbol{t})\sin(\boldsymbol{x}_3(\boldsymbol{t}))-\boldsymbol{u}_1\boldsymbol{P}_{\boldsymbol{p}(2,2)}\cos(\boldsymbol{x}_3(\boldsymbol{t})) \\ \dot{\boldsymbol{P}}_{\boldsymbol{p}(3,3)}(\boldsymbol{t}) &= \left(\boldsymbol{u}_1\boldsymbol{p}_1(\boldsymbol{t})\cos(\boldsymbol{x}_3(\boldsymbol{t})+\boldsymbol{u}_1\boldsymbol{p}_2(\boldsymbol{t})\sin(\boldsymbol{x}_3(\boldsymbol{t}))\right)\boldsymbol{P}_{\boldsymbol{x}(3,3)}(\boldsymbol{t})+\boldsymbol{u}_1\boldsymbol{P}_{\boldsymbol{p}(1,3)}(\boldsymbol{t})\sin(\boldsymbol{x}_3(\boldsymbol{t}))-\boldsymbol{u}_1\boldsymbol{P}_{\boldsymbol{p}(2,3)}\cos(\boldsymbol{x}_3(\boldsymbol{t})) \\ \dot{\boldsymbol{P}}_{\boldsymbol{p}(3,4)}(\boldsymbol{t}) &= \left(\boldsymbol{u}_1\boldsymbol{p}_1(\boldsymbol{t})\cos(\boldsymbol{x}_3(\boldsymbol{t})+\boldsymbol{u}_1\boldsymbol{p}_2(\boldsymbol{t})\sin(\boldsymbol{x}_3(\boldsymbol{t}))\right)\boldsymbol{P}_{\boldsymbol{x}(3,4)}(\boldsymbol{t})+\boldsymbol{u}_1\boldsymbol{P}_{\boldsymbol{p}(1,4)}(\boldsymbol{t})\sin(\boldsymbol{x}_3(\boldsymbol{t}))-\boldsymbol{u}_1\boldsymbol{P}_{\boldsymbol{p}(2,4)}\cos(\boldsymbol{x}_3(\boldsymbol{t})) \end{aligned}$$

Equation above is [3.1-18]

$$\dot{\boldsymbol{P}}_{\boldsymbol{p}(4,1)}(\boldsymbol{t}) = -\frac{2\boldsymbol{u}_1\boldsymbol{p}_3(\boldsymbol{t})\tan(\boldsymbol{x}_4(\boldsymbol{t}))(1+\tan^2(\boldsymbol{x}_4(\boldsymbol{t})))\boldsymbol{P}_{\boldsymbol{x}(4,1)}(\boldsymbol{t})}{\boldsymbol{l}} - \frac{\boldsymbol{u}_1(1+\tan^2(\boldsymbol{x}_4(\boldsymbol{t})))\boldsymbol{P}_{\boldsymbol{p}(3,1)}(\boldsymbol{t})}{\boldsymbol{l}}$$

$$\dot{\boldsymbol{P}}_{\boldsymbol{p}(4,2)}(\boldsymbol{t}) = -\frac{2\boldsymbol{u}_1\boldsymbol{p}_3(\boldsymbol{t})\tan(\boldsymbol{x}_4(\boldsymbol{t}))(1+\tan^2(\boldsymbol{x}_4(\boldsymbol{t})))\boldsymbol{P}_{\boldsymbol{x}(4,2)}(\boldsymbol{t})}{\boldsymbol{l}} - \frac{\boldsymbol{u}_1(1+\tan^2(\boldsymbol{x}_4(\boldsymbol{t})))\boldsymbol{P}_{\boldsymbol{p}(3,2)}(\boldsymbol{t})}{\boldsymbol{l}}$$

$$\dot{\boldsymbol{P}}_{\boldsymbol{p}(4,3)}(\boldsymbol{t}) = -\frac{2\boldsymbol{u}_1\boldsymbol{p}_3(\boldsymbol{t})\tan(\boldsymbol{x}_4(\boldsymbol{t}))(1+\tan^2(\boldsymbol{x}_4(\boldsymbol{t})))\boldsymbol{P}_{\boldsymbol{x}(4,3)}(\boldsymbol{t})}{\boldsymbol{l}} - \frac{\boldsymbol{u}_1(1+\tan^2(\boldsymbol{x}_4(\boldsymbol{t})))\boldsymbol{P}_{\boldsymbol{p}(3,3)}(\boldsymbol{t})}{\boldsymbol{l}}$$

$$\dot{\boldsymbol{P}}_{\boldsymbol{p}(4,4)}(\boldsymbol{t}) = -\frac{2\boldsymbol{u}_1\boldsymbol{p}_3(\boldsymbol{t})\tan(\boldsymbol{x}_4(\boldsymbol{t}))(1+\tan^2(\boldsymbol{x}_4(\boldsymbol{t})))\boldsymbol{P}_{\boldsymbol{x}(4,4)}(\boldsymbol{t})}{\boldsymbol{l}} - \frac{\boldsymbol{u}_1(1+\tan^2(\boldsymbol{x}_4(\boldsymbol{t})))\boldsymbol{P}_{\boldsymbol{p}(3,4)}(\boldsymbol{t})}{\boldsymbol{l}}$$

Equation above is [3.1-19]

Initial Conditions for differential influence function matrices $\boldsymbol{P_x}$ & $\boldsymbol{P_p}$

$$\boldsymbol{P}_{\boldsymbol{x}}(0) = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad \boldsymbol{P}_{\boldsymbol{p}}(0) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad [3.1\text{-}20]$$

Initial Conditions of ***x(0)*** & ***p(0)*** for step 3.

$$\boldsymbol{x}(0) = \begin{bmatrix} -1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad \boldsymbol{p}(0) = \begin{bmatrix} 1 \\ 1 \\ -1 \\ -1 \end{bmatrix} \qquad [3.1\text{-}21]$$

Final Conditions of $\boldsymbol{x(t_f)}$ for step 4.

$$\boldsymbol{x}(\boldsymbol{t}_f) = \begin{bmatrix} \textit{free} \\ 0.3 \\ 0 \\ 0 \end{bmatrix} \qquad [3.1\text{-}22]$$

Variation of Extremals (Convergence)

The method of variation of extremals will generally converge quite rapidly; however if the initial guess for $\boldsymbol{p(t_0)}$ is poor, the method may not converge at all. Making a good initial guess is a difficult matter, because we have no insight to guide us in selecting

$p^{(0)}(t_0)$. This was found to be true in this case as the system did not converge; this may also have resulted because this method does not allow the state constraint to be used.

The next attempted method to solve the two-point boundary value problem was Quasilinearization, but one of the matrices was always singular no matter what the initial conditions of ***p*** were.

For the final attempt, MATLAB/Simulink was used to model the State and Costate equations including the state constraints and the initial conditions. Below is the overview of the Simulink model of the variation of extremals method including the state constraints. Following the overview are the subsystem models of state and costate equation blocks.

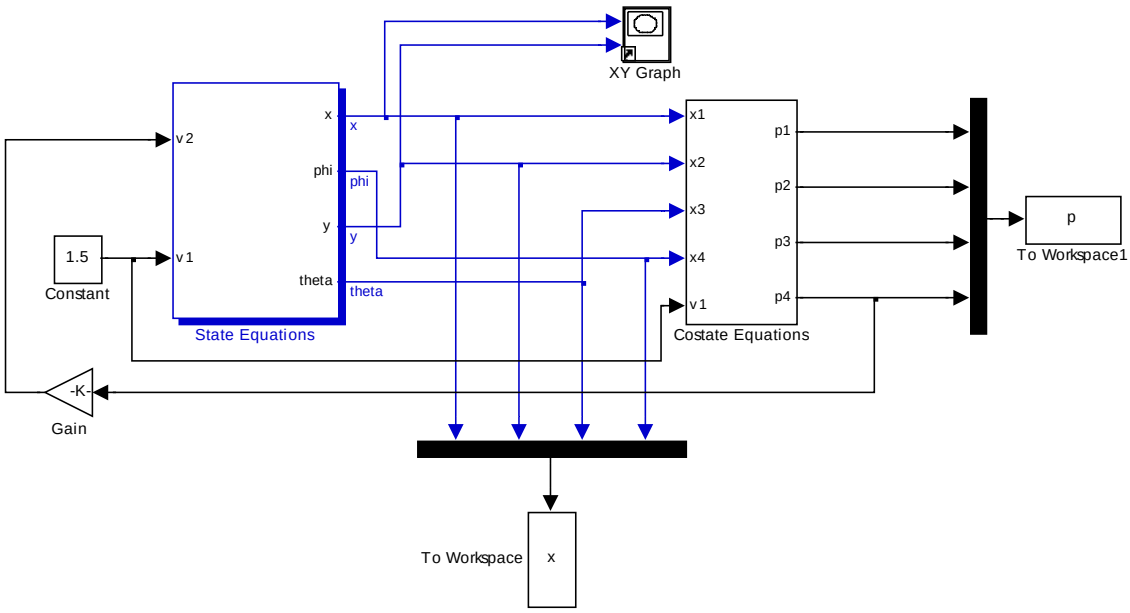


Figure 3 Model of State and Costate Equations with State Constraints

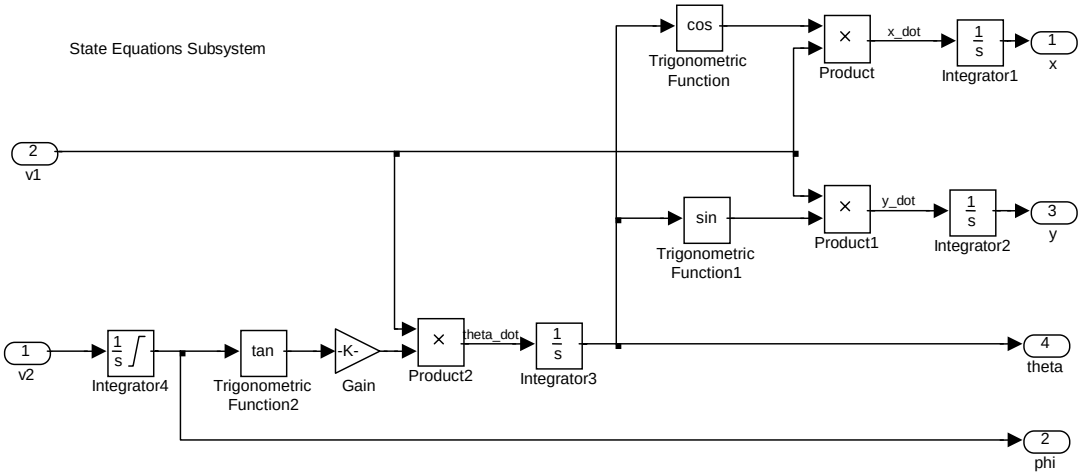


Figure 4 Model of State Equations with State Constraints Subsystem

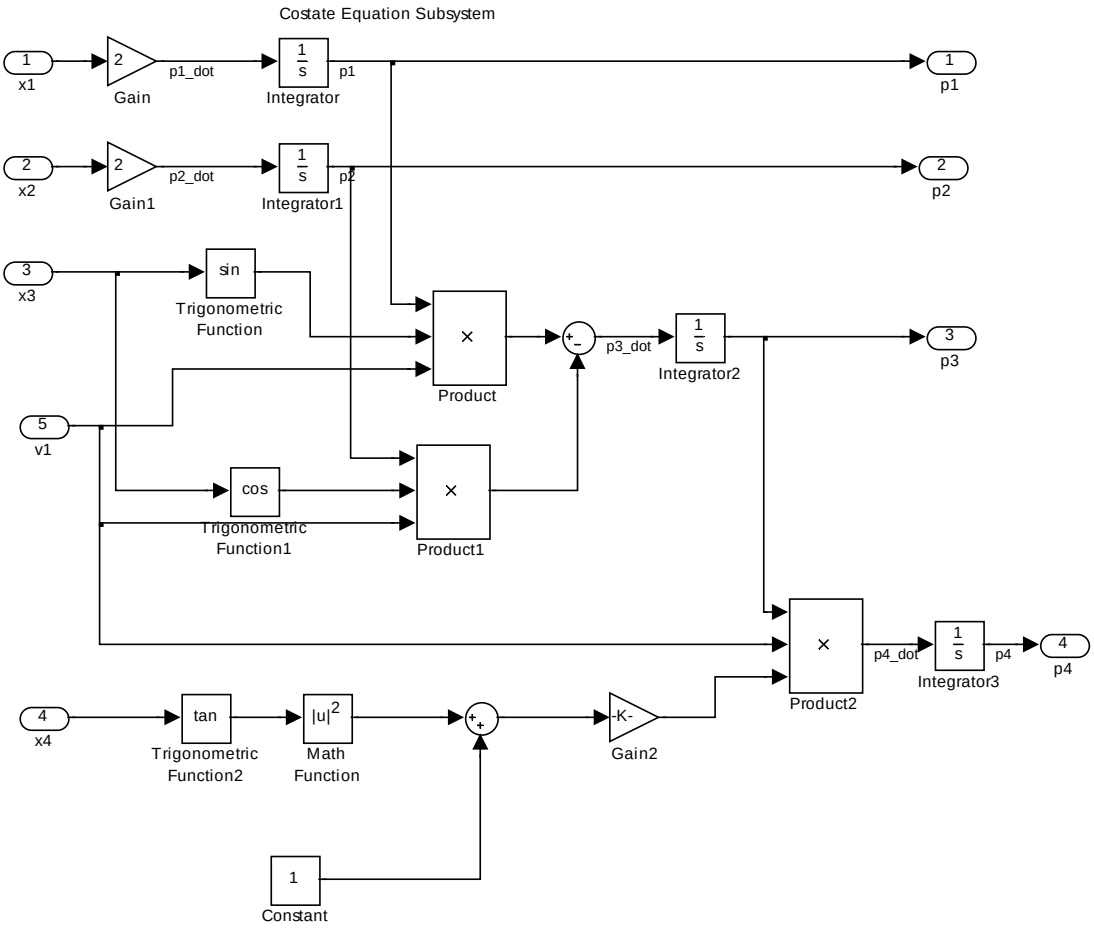


Figure 5 Model of Costate Equations Subsystem

The Simulink model above was run and the costate initial conditions ***p(0)*** were adjusted until the results came close to the desired $\boldsymbol{x(t_f)}$ see equation 3.1-22. The final values for ***p(0)*** are: $\boldsymbol{p}(0) = \begin{bmatrix} 400,000 \\ 2,000 \\ -18,000 \\ -2,685 \end{bmatrix}$ yielding the final values of $\boldsymbol{x(t_f)}$: $\boldsymbol{x}(t_f) = \begin{bmatrix} -0.54771 \\ 0.30313 \\ 0.0014119 \\ -0.7854 \end{bmatrix}$ with t_f=0.3856 seconds.

Below are graphs one showing the values of x vs. y from the Simulink model above run with the final values of ***p(0)*** and the other showing the steering angle $\boldsymbol{x_4(t)}$ vs. time.

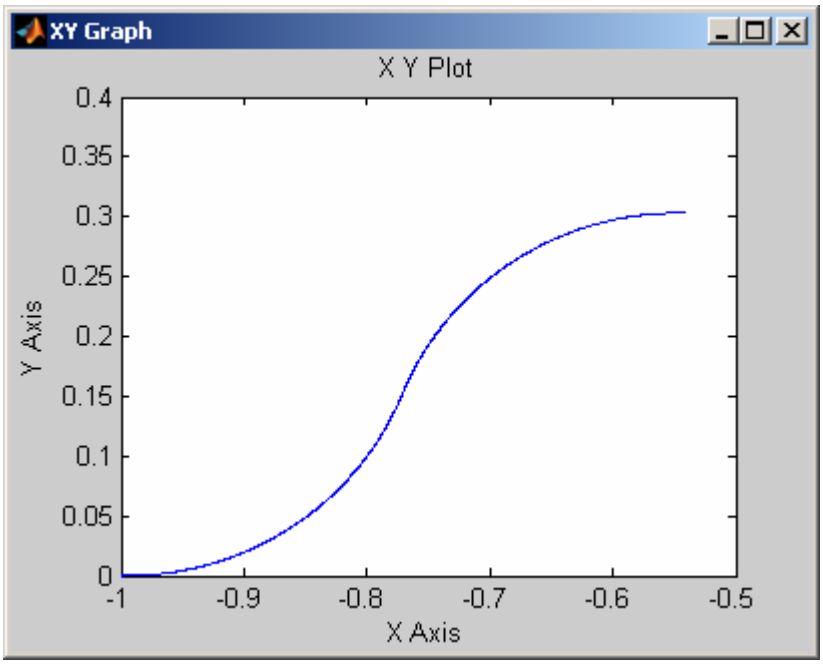


Figure 6 Optimal Path the Car takes when Changing Lanes

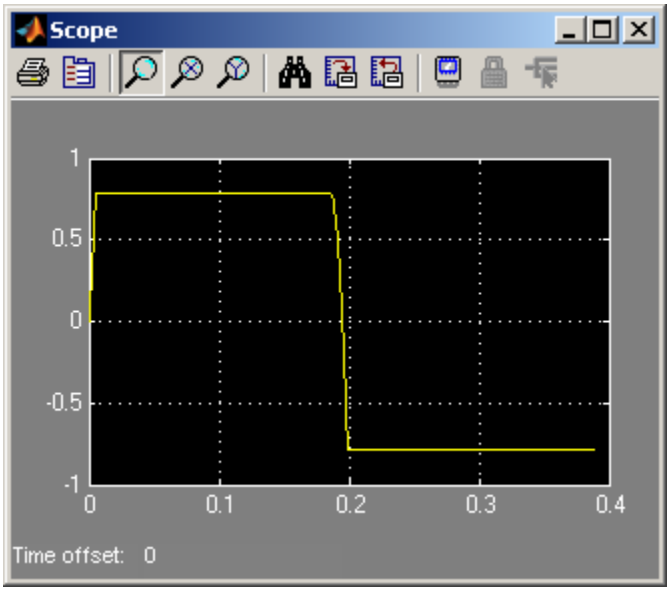


Figure 7 Steering Angle $x_4(t)$ (φ)

Obviously, state $\boldsymbol{x_4(t)}$ is "Bang Bang" to reach an optimal trajectory. To maneuver the car in an optimal trajectory when changing lanes, the car turns its wheel hard when making the first turn of the lane change maneuver then hard the other way when making the second turn of the lane change maneuver. Finally, the car is orientated in the new lane and follows it. The next section proves that this lane-changing maneuver has the optimal cost based on the previous definition of optimality.

3.1.5 Evaluated Cost Function of Various Trajectories

Now with the optimal lane change trajectory derived, various car-driving scenarios are explored. There are nine scenarios with static obstacles and moving cars. The cost function is numerically evaluated for each scenario.

3.1.5.1 Vehicle Stopped With One Fixed Obstacle

One option would have the car stop in front of an obstacle 1 meter away. The distance to the obstacle would be $\boldsymbol{L(t)}$=1 and $\boldsymbol{u_1}$=0 and $\boldsymbol{u_2}$=0. Using $\boldsymbol{t_0}$=0 and $\boldsymbol{t_f}$=0.66667 the same time the next example will use. The cost function shown in equation 3.1-23 is when the car is at rest.

$$J_{stop} = \int_{t_0}^{t_f} -L^2(t) + 10u_2^2 - u_1^2 dt = -0.66667 \qquad [3.1\text{-}23]$$

3.1.5.2 *Vehicle Driving Into One Fixed Obstacle*

Equation 3.1-24 is the distance from the car to an obstacle one meter in front of it, if the car drives straight into the obstacle. Figure 8 shows the trajectory of the car.

$$L(t) = 1 - u_1(t - t_0) \qquad [3.1\text{-}24]$$

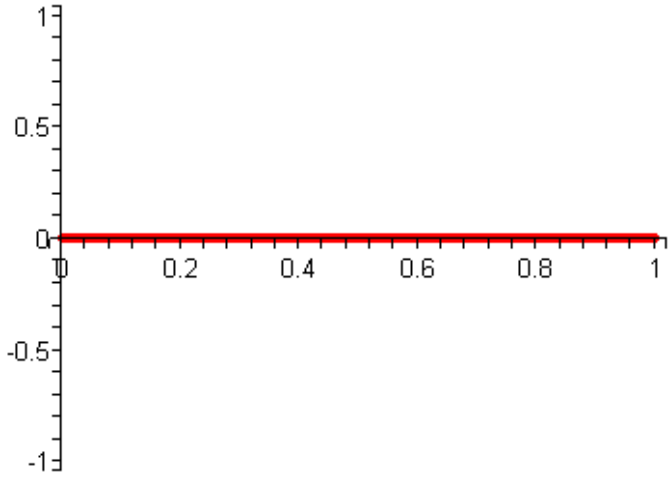


Figure 8 Path of Car Driving Straight into Obstacle

Equation 3.1-25 is the time when the car collides with the object Using t_0=0, and u_1=1.5 yields t_f=0.66667.

$$t_f = t_0 + \frac{1}{u_1} \qquad [3.1\text{-}25]$$

The cost function equation 3.1-26 uses 3.1-24, t_0=0, u_1=1.5, t_f=0.66667, and u_2=0.

$$J_{straight} = \int_{t_0}^{t_f} -L^2(t) + 10u_2^2 - u_1^2 dt = -1.722222222 \qquad [3.1\text{-}26]$$

3.1.5.3 *Vehicle Avoiding One Fixed Obstacle*

Next, the car uses "Bang Bang" control on state $x_4(t)$, which avoids obstacles, by changing lanes to maximize the distance between the obstacle and the car.

Integrating the state equations 3.1-1 gives equations for $x_1(t)$ and $x_2(t)$ in terms of $x_3(t)$.

$$\begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} = \int \left(\begin{bmatrix} \cos(x_3(t)) \\ \sin(x_3(t)) \end{bmatrix} u_1 + \begin{bmatrix} 0 \\ 0 \end{bmatrix} u_2(t) \right) dt \qquad [3.1\text{-}27]$$

$x_3(t)$ is expressed in terms of $x_4(t)$ in the equation below:

$$x_3(t) = \int \left(\left(\frac{\tan(x_4(t))}{l} \right) u_1 + (0) u_2(t) \right) dt \qquad [3.1\text{-}28]$$

The "Bang Bang" control of $x_4(t)$ can be expressed piecewise as:

$$x_4(t) = \begin{cases} \frac{\pi}{3} & t_0 \le t \le t_1 \\ -\frac{\pi}{3} & t_1 < t \le t_2 \\ 0 & t_2 < t \le t_f \end{cases} \qquad [3.1\text{-}29]$$

t_1, t_2, t_f, are calculated from the equations below given u_1=1.5, t_0=0, l=0.2413, lane distance=0.3, and the equations 3.1-27, 3.1-28, and 3.1-29

$$\begin{aligned} \frac{lane\_distance}{2} = x_2(t_1) \quad & \Rightarrow t_1 = 0.1902674305 \\ t_2 = 2(t_1 - t_0) + t_0 \quad & \Rightarrow t_2 = 0.3805348610 \\ 1 = x_1(t_f) \quad & \Rightarrow t_f = 0.7495067304 \end{aligned} \qquad [3.1\text{-}30]$$

The distance to an obstacle 1 meter in front of the car is shown in equation 3.1-31 and the path is shown in Figure 9:

$$L(t) = \sqrt{(x_1(t)-1)^2 + x_2^2(t)} \qquad [3.1\text{-}31]$$

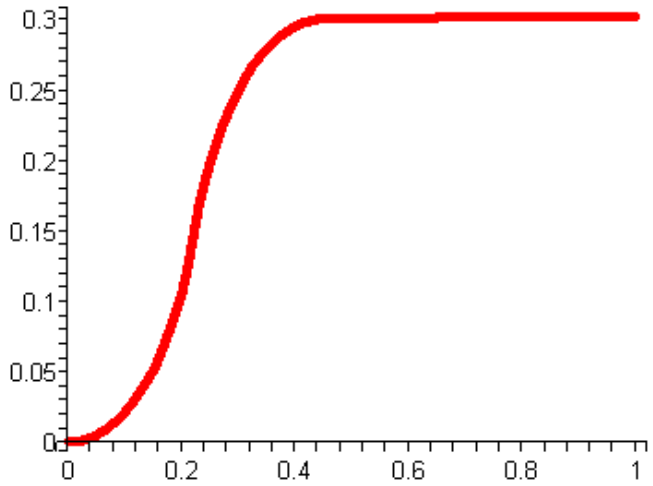


Figure 9 Path of the Car Avoiding the Obstacle while changing lanes

The cost function using equations 3.1-27, 3.1-28, 3.1-29, 3.1-30, and 3.1-31 is shown below.

$$J_{lane\_change} = \int_{t_0}^{t_f} -L^2(t) + 10u_2^2 - u_1^2 dt = -2.005327312 \qquad [3.1\text{-}32]$$

This cost function is lower than the preceding two examples.

3.1.5.4 Vehicle Stopped With Two Fixed Obstacles

The fourth option would have the car stop in front of two obstacles. One obstacle is 1 meter directly in front of the car and the other obstacle is in another lane 0.5 meters in front of the car in the x direction and -0.3 in the y direction. The sum of the distances to the two obstacles would be $L(t)$= 1.583095190 and u_1=0 and u_2=0. Using t_0=0 and t_f=0.66667 the same time the next example will use. The cost function shown in equation 3.1-34 is when the car is at rest.

$$L(t) = 1 + \sqrt{.5^2 + .3^2} = 1.583095190 \qquad [3.1\text{-}33]$$

$$J_{stop2} = \int_{t_0}^{t_f} -L^2(t) + 10u_2^2 - u_1^2 dt = \text{-}1.670793397 \qquad [3.1\text{-}34]$$

3.1.5.5 Vehicle Driving Into One Of Two Fixed Obstacles

Equation 3.1-35 is the sum of the distances from the car to the two obstacles as described above, assuming the car drives straight into the obstacle in front of it. Figure 10 shows the trajectory of the car.

$$\boldsymbol{L}(\boldsymbol{t}) = 1 - \boldsymbol{u}_1(\boldsymbol{t} - \boldsymbol{t}_0) + \sqrt{\left(1 - \boldsymbol{u}_1(\boldsymbol{t} - \boldsymbol{t}_0) - .5\right)^2 + (-0.3)^2} \qquad [3.1\text{-}35]$$

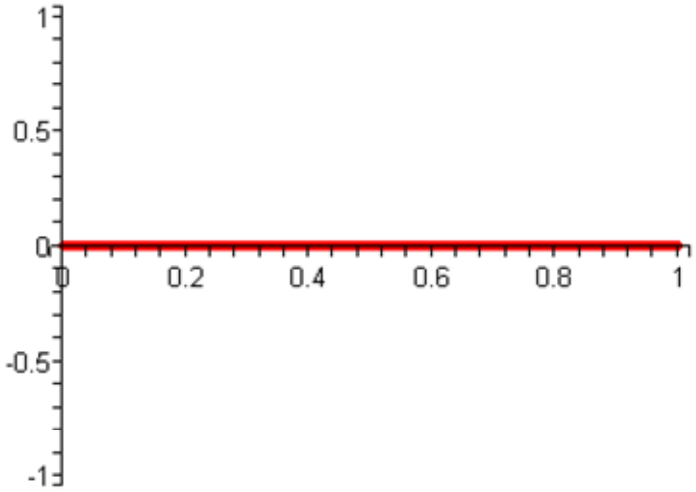


Figure 10 Path of a Car Driving Straight into Obstacle

Equation 3.1-36 is the time when the car collides with the object. Using $\boldsymbol{t_0}$=0, and $\boldsymbol{u}_1$=1.5 yields $\boldsymbol{t_f}$=0.66667.

$$\boldsymbol{t}_f = \boldsymbol{t}_0 + \frac{1}{\boldsymbol{u}_1} \qquad [3.1\text{-}36]$$

The cost function, equation 3.1-37 uses 3.1-35, $\boldsymbol{t_0}$=0, $\boldsymbol{u}_1$=1.5, $\boldsymbol{t_f}$=0.66667, and $\boldsymbol{u}_2$=0.

$$\boldsymbol{J}_{straight\,2} = \int_{t_0}^{t_f} -\boldsymbol{L}^2(\boldsymbol{t}) + 10\boldsymbol{u}_2^2 - \boldsymbol{u}_1^2 \boldsymbol{dt} = -2.109170581 \qquad [3.1\text{-}37]$$

3.1.5.6 Vehicle Avoiding Two Fixed Obstacles

Next, the car uses "Bang Bang" control on state $x_4(t)$, which avoids obstacles, by changing lanes to maximize the distance between the obstacle and the car.

Integrating the state equations 3.1-1 gives equations for $x_1(t)$ and $x_2(t)$ in terms of $x_3(t)$.

$$\begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} = \int \left(\begin{bmatrix} \cos(x_3(t)) \\ \sin(x_3(t)) \end{bmatrix} u_1 + \begin{bmatrix} 0 \\ 0 \end{bmatrix} u_2(t) \right) dt \qquad [3.1\text{-}38]$$

$x_3(t)$ is expressed in terms of $x_4(t)$ in the equation below:

$$x_3(t) = \int \left(\left(\frac{\tan(x_4(t))}{l} \right) u_1 + (0)\, u_2(t) \right) dt \qquad [3.1\text{-}39]$$

The "Bang Bang" control of $x_4(t)$ can be expressed piecewise as:

$$x_4(t) = \begin{cases} \frac{\pi}{3} & t_0 \le t \le t_1 \\ -\frac{\pi}{3} & t_1 < t \le t_2 \\ 0 & t_2 < t \le t_f \end{cases} \qquad [3.1\text{-}40]$$

t_1, t_2, t_f, are calculated from the equations below given u_1=1.5, t_0=0, l=0.2413, lane distance=0.3, and the equations 3.1-27, 3.1-28, and 3.1-29

$$\begin{aligned} \frac{lane\_distance}{2} = x_2(t_1) \quad & \Rightarrow t_1 = 0.1902674305 \\ t_2 = 2(t_1 - t_0) + t_0 \quad & \Rightarrow t_2 = 0.3805348610 \\ 1 = x_1(t_f) \quad & \Rightarrow t_f = 0.7495067304 \end{aligned} \qquad [3.1\text{-}41]$$

Equation 3.1-42 is the sum of the distances from the car to the two obstacles as described above. Figure 11 shows the trajectory of the car.

$$L(t) = \sqrt{(x_1(t) - 1)^2 + x_2^2(t)} + \sqrt{(x_1(t) - .5)^2 + (x_2(t) + 0.3)^2} \qquad [3.1\text{-}42]$$

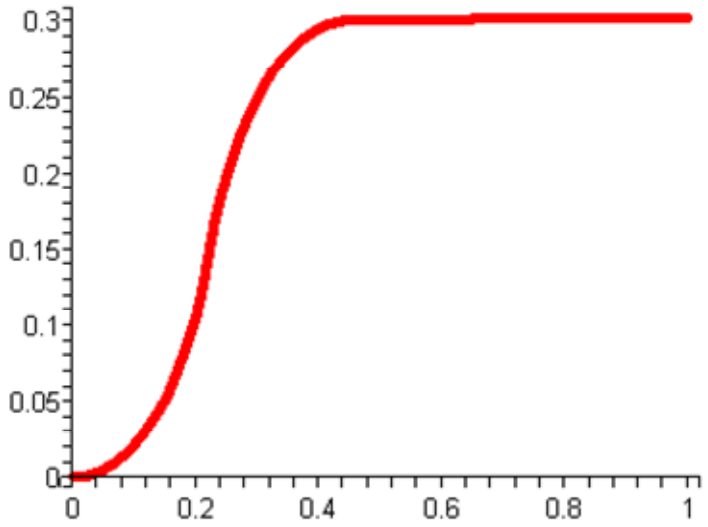


Figure 11 Path of a Car Avoiding the Obstacle while changing lanes

The cost function using equations 3.1-38, 3.1-39, 3.1-40, 3.1-41, 3.1-42 is shown below.

$$\boldsymbol{J}_{lane\_change2} = \int_{t_0}^{t_f} -\boldsymbol{L}^2(\boldsymbol{t}) + 10\boldsymbol{u}_2^2 - \boldsymbol{u}_1^2 \boldsymbol{dt} = \text{-2.820447717} \qquad [3.1\text{-}43]$$

This cost is lower than the cost of the two preceding examples.

3.1.5.7 Vehicle Stopped With One Fixed Obstacle and One Moving Obstacle

Another option would have the car stop in front of an obstacle that is in front of it and have another vehicle keep going down the highway. The static obstacle is 1 meter directly in front of the car and the other vehicle is in another lane starting 0.5 meters in front of the car in the x direction with a velocity of 1.75 m/s and -0.3 in the y direction. The sum of the distances to the two obstacles would be ***L(t)*** in equation 3.1-44 and $\boldsymbol{u_1}$=0 and $\boldsymbol{u_2}$=0. Using $\boldsymbol{t_0}$=0 and $\boldsymbol{t_f}$=0.66667 the same time the next example will use. The cost function is shown in equation 3.1-45 when the car is at rest.

$$\boldsymbol{L}(\boldsymbol{t}) = 1 + \sqrt{(-1 - 1.75\boldsymbol{t} + 0.5)^2 + 0.3^2} \qquad [3.1\text{-}44]$$

$$\boldsymbol{J}_{stop3} = \int_{t_0}^{t_f} -\boldsymbol{L}^2(\boldsymbol{t}) + 10\boldsymbol{u}_2^2 - \boldsymbol{u}_1^2 \boldsymbol{dt} = \text{-3.089128653} \qquad [3.1\text{-}45]$$

3.1.5.8 Vehicle Driving into a fixed Obstacle with a moving obstacle

Equation 3.1-46 is the sum of the distances from the car to the two obstacles as described above. Assuming the car drives straight into the obstacle in front of it. Figure 12 shows the trajectory of the car.

$$\boldsymbol{L}(\boldsymbol{t}) = 1 - \boldsymbol{u}_1(\boldsymbol{t} - \boldsymbol{t}_0) + \sqrt{\left(1 - \boldsymbol{u}_1(\boldsymbol{t} - \boldsymbol{t}_0) - \left(1.75\left(\boldsymbol{t} - \boldsymbol{t}_0\right) - 0.5\right)\right)^2 + (-0.3)^2} \qquad [3.1\text{-}46]$$

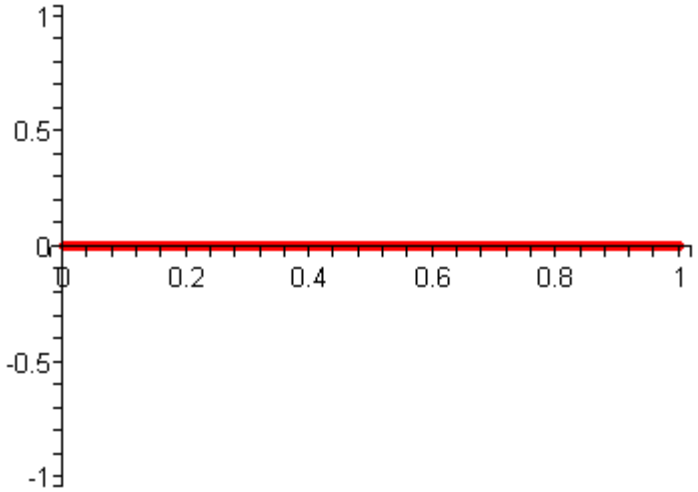


Figure 12 Path of a Car Driving Straight into Obstacle

Equation 3.1-47 is the time when the car collides with the object. Using $\boldsymbol{t_0}$=0, and $\boldsymbol{u}_1$=1.5 yields $\boldsymbol{t_f}$=0.66667.

$$\boldsymbol{t}_f = \boldsymbol{t}_0 + \frac{1}{\boldsymbol{u}_1} \qquad [3.1\text{-}47]$$

The cost function, equation 3.1-48 uses 3.1-46, $\boldsymbol{t_0}$=0, $\boldsymbol{u}_1$=1.5, $\boldsymbol{t_f}$=0.66667, $\boldsymbol{u}_2$=0, and the moving vehicle going 1.75 m/s.

$$\boldsymbol{J}_{straight\,4} = \int_{t_0}^{t_f} -\boldsymbol{L}^2(\boldsymbol{t}) + 10\boldsymbol{u}_2^2 - \boldsymbol{u}_1^2 \boldsymbol{dt} = -2.756857316 \qquad [3.1\text{-}48]$$

3.1.5.9 Vehicle Avoiding One Fixed Obstacle and One Moving Obstacle

Next, is the car uses "Bang Bang" control on state $x_4(t)$, which avoids obstacles, by changing lanes to maximize the distance between the obstacle and the car.

Integrating the state equations 3.1-1 gives equations for $x_1(t)$ and $x_2(t)$ in terms of $x_3(t)$.

$$\begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} = \int \left(\begin{bmatrix} \cos(x_3(t)) \\ \sin(x_3(t)) \end{bmatrix} u_1 + \begin{bmatrix} 0 \\ 0 \end{bmatrix} u_2(t) \right) dt \qquad [3.1\text{-}49]$$

$x_3(t)$ is expressed in terms of $x_4(t)$ in the equation below:

$$x_3(t) = \int \left(\left(\frac{\tan(x_4(t))}{l} \right) u_1 + (0)\, u_2(t) \right) dt \qquad [3.1\text{-}50]$$

The "Bang Bang" control of $x_4(t)$ can be expressed piecewise as:

$$x_4(t) = \begin{cases} \frac{\pi}{3} & t_0 \le t \le t_1 \\ -\frac{\pi}{3} & t_1 < t \le t_2 \\ 0 & t_2 < t \le t_f \end{cases} \qquad [3.1\text{-}51]$$

t_1, t_2, t_f, are calculated from the equations below given u_1=1.5, t_0=0, l=0.2413, lane distance=0.3, and the equations 3.1-27, 3.1-28, and 3.1-29

$$\begin{aligned} \frac{lane\_distance}{2} = x_2(t_1) \quad & \Rightarrow t_1 = 0.1902674305 \\ t_2 = 2(t_1 - t_0) + t_0 \quad & \Rightarrow t_2 = 0.3805348610 \\ 1 = x_1(t_f) \quad & \Rightarrow t_f = 0.7495067304 \end{aligned} \qquad [3.1\text{-}52]$$

Equation 3.1-53 is the sum of the distances from the car to the two obstacles as described above. Figure 13 shows the trajectory of the car.

$$L(t) = \sqrt{(x_1(t)-1)^2 + x_2^2(t)} + \sqrt{\left(x_1(t) - \left(1.75(t - t_0) - 0.5\right)\right)^2 + (x_2(t) + 0.3)^2} \qquad [3.1\text{-}53]$$

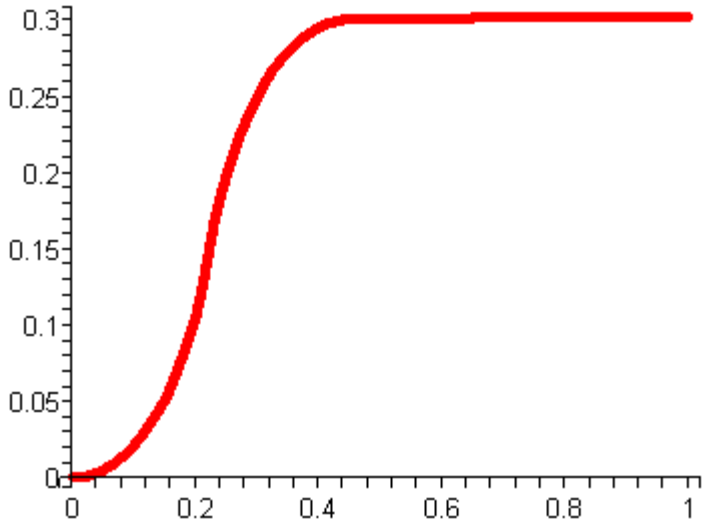


Figure 13 Path of a Car Avoiding the Obstacle while changing lanes

The cost function using equations 3.1-49, 3.1-50, 3.1-51, 3.1-52, 3.1-53 is shown below.

$$J_{lane\_change3} = \int_{t_0}^{t_f} -L^2(t) + 10u_2^2 - u_1^2 dt = -3.695199822 \qquad [3.1\text{-}54]$$

This cost function is lower than the preceding two examples.

3.2 MATLAB/Simulink Simulation Environment

Starting with [7], a MATLAB/Simulink simulation environment was created for the car. The simulation environment was designed for the car to go around an oval track with 3 lanes. The track was created as a 3 m by 8 m grid of 1 mm square points. The white line was simulated as ones in this 3000 by 8000 matrix and black as zeros. Most of the simulation functions in [7] were written as MATLAB M files and the simulations were run from MATLAB. These functions were rewritten in Simulink and the simulation runs from the Simulink model. This graphical approach with multiple subsystems allows for an easy understanding and easy of use as well as ease of debugging. It also makes it clearer as to exactly what is going on from signal to signal. The top-level overview of the Simulink model is shown in Figure 14 below.

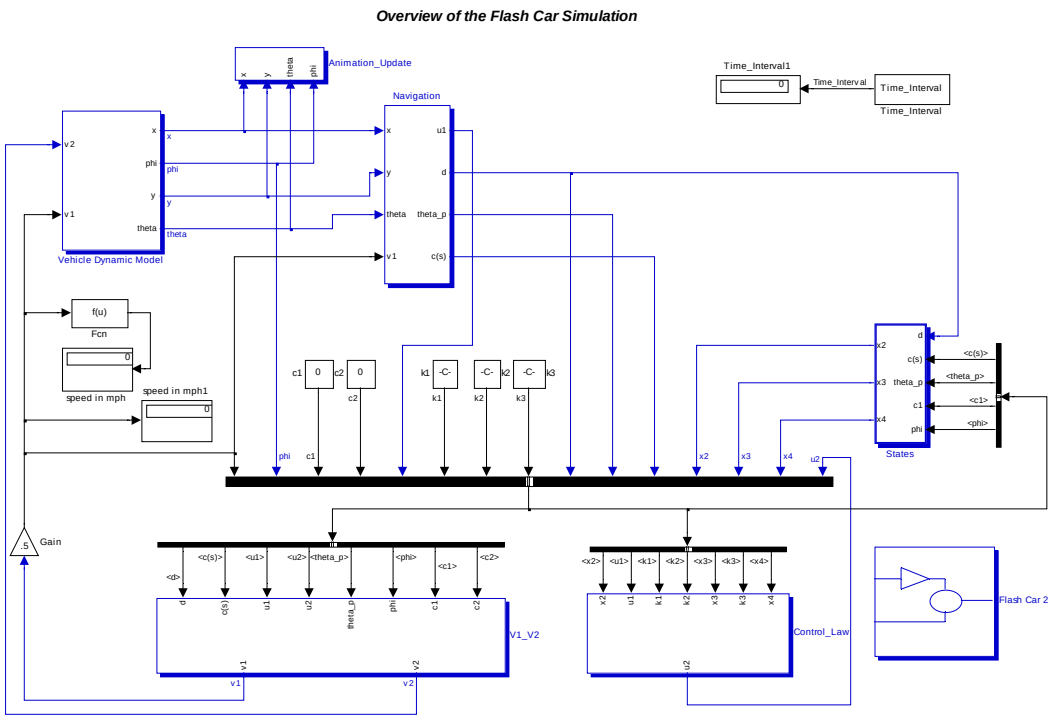


Figure 14 Overview of the Car Simulation

The rest of the models and subsystems are included in Appendix A Simulink

The simulation of lane changing, car behavior when changing lanes, obstacle detection, obstacle avoidance and control law are discussed next.

3.2.1 Car Path Control

The complete kinematic model for each car is based on equation (9.10) in [5] and is in the subsystem *Car Dynamic Model* shown in Appendix A:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} \sin\theta \\ \cos\theta \\ \dfrac{\tan\theta}{l} \\ 0 \end{bmatrix} v_1 + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} v_2 \qquad [3.2\text{-}1]$$

The lateral controller used to control each car is based on a state space feedback system. These states, shown below are defined by equations (10.4-10.6) in [5] and are implemented in *Overview of the States* subsystem shown in Appendix A:

$$\boldsymbol{x}_2 = -\boldsymbol{c}'(\boldsymbol{s})\boldsymbol{d}\tan\theta_p - \boldsymbol{c}(\boldsymbol{s})(1-\boldsymbol{d}\,\boldsymbol{c}(\boldsymbol{s}))\frac{1+\sin^2\theta_p}{\cos^2\theta_p} + \frac{(1-\boldsymbol{d}\,\boldsymbol{c}(\boldsymbol{s}))^2\tan\phi}{\boldsymbol{l}\cos^2\theta_p} \qquad [3.2\text{-}2]$$

$$\boldsymbol{x}_3 = (1-\boldsymbol{d}\,\boldsymbol{c}(\boldsymbol{s}))\tan\theta_p \qquad [3.2\text{-}3]$$

$$\boldsymbol{x}_4 = \boldsymbol{d} \qquad [3.2\text{-}4]$$

The control law equations, shown below are defined by equation (10.13) in [5] and are included in the subsystem *Control Law* shown in Appendix A:

$$\boldsymbol{u}_2 = -\boldsymbol{k}_1\left|\boldsymbol{u}_1(\boldsymbol{t})\right|\boldsymbol{x}_2 - \boldsymbol{k}_2\boldsymbol{u}_1(\boldsymbol{t})\boldsymbol{x}_3 - \boldsymbol{k}_3\left|\boldsymbol{u}_1(\boldsymbol{t})\right|\boldsymbol{x}_4 \qquad [3.2\text{-}5]$$

Below are the best values of the gains that were determined experimentally for the controller:

$\boldsymbol{k}_1 = 4000$

$\boldsymbol{k}_2 = 1200$

$\boldsymbol{k}_3 = 60$

The inputs to the car kinematic equations are $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ as defined by equations (10.7 & 10.8) in [5] where $\boldsymbol{v}_1$ is the linear velocity of the rear wheels and $\boldsymbol{v}_2$ is the angular velocity of the steering wheels. The equations for $\boldsymbol{a}_1$ and $\boldsymbol{a}_2$ are given in [5] following equations (10.7 & 10.8). These equations below are contained in the *Overview of V1 and V2* subsystem shown in Appendix A.

$$\boldsymbol{v}_1 = \frac{1-\boldsymbol{d}\,\boldsymbol{c}(\boldsymbol{s})}{\cos\theta_p}\boldsymbol{u}_1 \qquad [3.2\text{-}6]$$

$$\boldsymbol{v}_2 = \boldsymbol{a}_2(\boldsymbol{u}_2 - \boldsymbol{a}_1\boldsymbol{u}_1) \qquad [3.2\text{-}7]$$

$$\boldsymbol{a}_1 = \frac{\partial \boldsymbol{x}_2}{\partial \boldsymbol{s}} + \frac{\partial \boldsymbol{x}_2}{\partial \boldsymbol{d}}(1-\boldsymbol{d}\,\boldsymbol{c}(\boldsymbol{s}))\tan\theta_p + \frac{\partial \boldsymbol{x}_2}{\partial \theta_p}\left[\frac{\tan\phi(1-\boldsymbol{d}\,\boldsymbol{c}(\boldsymbol{s})}{\boldsymbol{l}\cos\theta_p} - \boldsymbol{c}(\boldsymbol{s})\right] \qquad [3.2\text{-}8]$$

$$\boldsymbol{a}_2 = \frac{\boldsymbol{l}\cos^3\theta_p \cos^2\phi}{(1-\boldsymbol{d}\,\boldsymbol{c}(\boldsymbol{s}))^2} \qquad [3.2\text{-}9]$$

The following equations came from [7], and are included in the subsystem above.

$$\frac{\partial \boldsymbol{x}_2}{\partial \boldsymbol{s}} = -\ddot{\boldsymbol{c}}(\boldsymbol{s})\boldsymbol{d}\tan\theta_p - (\dot{\boldsymbol{c}}(\boldsymbol{s}) + 2\boldsymbol{d}\,\boldsymbol{c}(\boldsymbol{s})\dot{\boldsymbol{c}}(\boldsymbol{s}))\frac{1+\sin^2\theta_p}{\cos^2\theta_p} - \frac{2(1-\boldsymbol{d}\,\boldsymbol{c}(\boldsymbol{s}))\boldsymbol{d}\,\dot{\boldsymbol{c}}(\boldsymbol{s})\tan\phi}{\boldsymbol{l}\cos^3\theta_p} \qquad [3.2\text{-}10]$$

$$\frac{\partial \boldsymbol{x}_2}{\partial \boldsymbol{d}} = \boldsymbol{c}^2(\boldsymbol{s})\frac{1+\sin^2\theta_p}{\cos^2\theta_p} - \frac{2\,\boldsymbol{c}(\boldsymbol{s})(1-\boldsymbol{d}\,\boldsymbol{c}(\boldsymbol{s}))\tan\phi}{\boldsymbol{l}\cos^3\theta_p} \qquad [3.2\text{-}11]$$

$$\frac{\partial \boldsymbol{x}_2}{\partial \theta_p} = -\boldsymbol{c}(\boldsymbol{s})\frac{1-\boldsymbol{d}\,\boldsymbol{c}(\boldsymbol{s})4\tan\theta_p}{\cos^2\theta_p} + \frac{3(1-\boldsymbol{d}\,\boldsymbol{c}(\boldsymbol{s}))^2\tan\phi\tan\theta_p}{\boldsymbol{l}\cos^3\theta_p} \qquad [3.2\text{-}12]$$

3.2.2 Lane Changing

The *Lane Changing* subsystem shown in Appendix A takes in all the sensor data, and the System Mode.

When the ***System_Mode*** is set to 0, then ***sensor_d***, ***sensor_theta_p***, and ***sensor_c(s)*** go to the car controller and the car continues on its way following the white line.

When the ***System_Mode*** is set to 1, then the lane_change.m file is called to make the car change lanes. The sensor data includes the IR sensor and the range finder. The state machine that controls the car lane changing maneuver is the ***lane_change*** function. This function takes in ***turn_left_or_right***, ***front_error_dist***, ***theta***, ***System_Mode***, ***lane_we_are_in***, ***In_Lane_Change, theta1***, ***lane_change_direction_latched*** and outputs the curvature:

$$\boldsymbol{c(s)} = \frac{\boldsymbol{lane\_change\_direction\_latched}}{\boldsymbol{Rad\_of\_Lane\_Change\_Turn}}$$

The car controller and states use this curvature, so that the car turns its wheel hard when making the first turn of the lane change maneuver, then hard the other way when making the second turn of the lane change maneuver, and finally changes to the new lane and follows it.

The car turns until ***theta*** and ***theta1*±*Turn_Alpha*** are approximately equal; it is ± because it depends on whether the car is turning left or right. The car repeats the curve executed at the beginning of the lane change maneuver except in the opposite direction to align the car with the white line. In addition, the lane_change.m file updates the ***lane_we_are_in***. The reason for the delays (1/Z) is to retain the previous values of the ***lane_we_are_in***, ***In_Lane_Change***, ***theta1***, ***lane_change_direction_latched***. Since there are multiple cars, the delays remember the previous values for each car separately. This way the same functions can be used for each car.

3.2.3 Obstacle Detection

The obstacle detection system models ultrasonic or range finder distance sensors, and when there is an obstacle in the region the sensor detects it. The region for each sensor is defined by 360 degrees divided by the number of sensors. The ***obstacle_sensor_function*** uses the current ***x***,***y***, and ***theta*** and returns an array with the distance the closest obstacle is away from each sensor. If there is no obstacle closer than ***max_dist*** the ***obstacle_sensor_function*** outputs ***max_dist***. There is an array ***static_obstacles***, which includes obstacles that do not move, and the cars' positions. It is a 2 dimensional array which has a row for each obstacle, in each row there is a column for x-position, y-position, and obstacle radius.

3.2.4 Obstacle Avoidance

The *Obstacle Avoidance* subsystem shown in Appendix A, takes in ***x***, ***y***, and ***theta*** which goes to the obstacle sensor function which simulates the distance sensors, using the MATLAB function ***obstacle_sensor_function.m*** The discrete derivative is used to calculate the velocity of the obstacles. ***Impulse_to_zero*** function eliminates the impulse that occurs when a sensor rotates and detects a nearby object. The ***obstacle_avoidance_cost_function_control_law.m*** has twenty-one different states, each ***distance sensor***, ***velocity from each distance sensor***, ***lane we are in*** (1=outer 3=inner), car velocity ***V1***, car heading ***theta***, ***System Mode***, and ***x*** location. The outputs for the cost function are ***u1*** (car linear velocity), ***turn left or right***, and ***System Mode***. The cost function calculates using optimal control, the optimal control outputs. For the fuzzy obstacle avoidance controller the function ***fuzzy_law.m*** is substituted for ***obstacle_avoidance_cost_function_control_law.m***

3.2.5 Control Law

The control law 3.2-13 is the heart of the optimal control obstacle avoidance controller. It determines which way the car turns when there is an obstacle in the way.

$$\boldsymbol{u}^{*} = \min_{u_2=[1,0,-1]} \begin{bmatrix} \boldsymbol{x}_1\boldsymbol{k}_1(\boldsymbol{u}_2^2\text{-}1)\text{-}(\boldsymbol{x}_1+\boldsymbol{k}_2)\boldsymbol{u}_2^2+\boldsymbol{k}_3(\boldsymbol{x}_{13}<\text{-}.5)\boldsymbol{u}_2^2+(\boldsymbol{u}_2==\text{-}1)(\boldsymbol{x}_{17}==1)+ \\ (\boldsymbol{u}_2==1)(\boldsymbol{x}_{17}==3)+5\boldsymbol{u}_2^2(\boldsymbol{x}_{21}>\boldsymbol{maxcurvex}) \mid (\boldsymbol{x}_{21}<\boldsymbol{mincurvex}) \end{bmatrix} \quad [3.2\text{-}13]$$

This term changes lanes if obstacle gets too close in the front $\boldsymbol{x_1}$

$$\boldsymbol{x}_1\boldsymbol{k}_1(\boldsymbol{u}_2^2\text{-}1)\text{-}(\boldsymbol{x}_1+\boldsymbol{k}_2)\boldsymbol{u}_2^2 \quad [3.2\text{-}14]$$

This term changes lanes if an obstacle is behind the car and is approaching at a speed of less than -0.5

$$\boldsymbol{k}_3(\boldsymbol{x}_{13}<\text{-}.5)\boldsymbol{u}_2^2 \quad [3.2\text{-}15]$$

This term increases the cost so if the car is in lane 1 then it won't make a right turn

$$(\boldsymbol{u}_2==\text{-}1)(\boldsymbol{x}_{17}==1) \quad [3.2\text{-}16]$$

This term increases the cost so if the car is in lane 3 then it won't make a left turn

$$(\boldsymbol{u}_2==1)(\boldsymbol{x}_{17}==3) \quad [3.2\text{-}17]$$

This term increases the cost so the car will not try to change lanes on the curves at either end of the track since the lane change maneuver does not currently work on the curves.

$$5\boldsymbol{u}_2^2(\boldsymbol{x}_{21}>\boldsymbol{maxcurvex}) \mid (\boldsymbol{x}_{21}<\boldsymbol{mincurvex}) \quad [3.2\text{-}18]$$

Figure 15, Figure 16, and Figure 17 below show the costs for the three choices of changing lanes with the cost as the Y-axis and the Distance an Object is in front of the car as the X-axis, all three graphs have no car approaching from the rear.

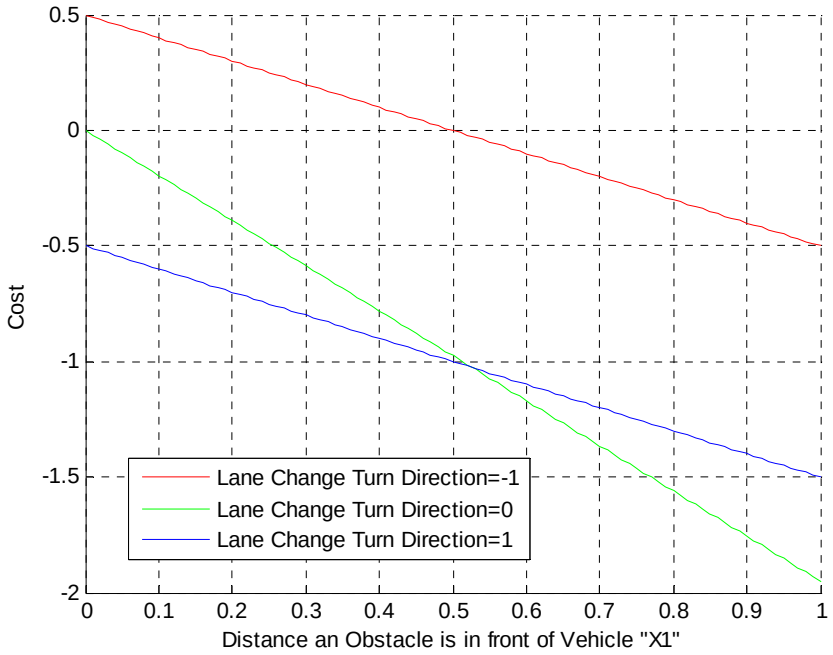


Figure 15 Cost Function for Lane 1 and Rear Velocity=0

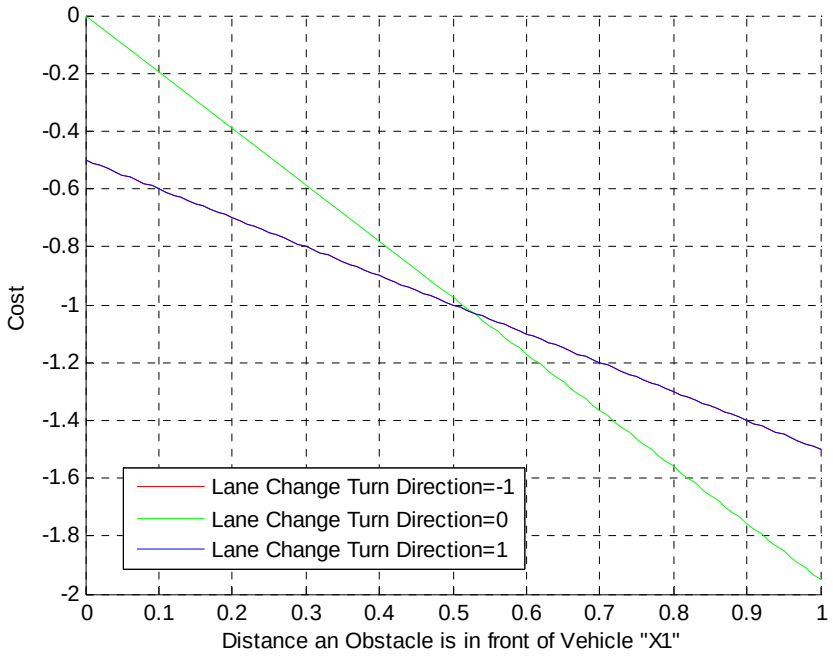


Figure 16 Cost Function for Lane 2 and Rear Velocity=0

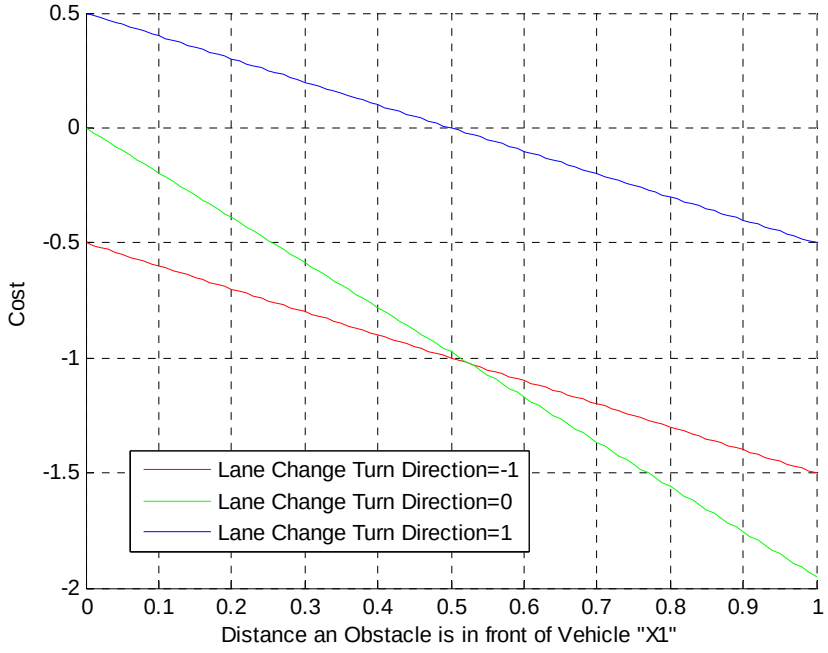


Figure 17 Cost Function for Lane 3 and Rear Velocity=0

3.2.6 Adaptive Cruise Control

Adaptive cruise control is included in the model, because it is an important part of collision avoidance. This make the simulation more realistic, in the sense that a driver will not always changes lanes when they see a car in front of them that is going slower than they are. When the car in front of them slows down too much then changing lanes is preferable. The following equation is what adjusts the car's velocity based on the distance to the obstacle in front of it.

In equation, 3.2-19 $\boldsymbol{x_1}$ is distance the obstacle is in front of the vehicle and once the obstacle gets closer than 1 meter, the speed of the vehicle decreases to 0 when the obstacle gets within 0.2 meters. The max function makes sure the vehicle does not travel backwards. This is included in ***obstacle_avoidance_cost_function_control_law.m***

$$\boldsymbol{u}_1=\max\left(\frac{3(\boldsymbol{x}_1-0.2)}{0.8}\ 0\right) \qquad [3.2\text{-}19]$$

3.2.7 Car Behavior when avoiding obstacles and changing lanes

When the car detects an obstacle and it needs to change lanes it performs the following maneuver. The car turns its wheel hard when making the first turn of the lane change maneuver, then hard the other way when making the second turn of the lane change maneuver, and finally changes to the new lane and follows it. This path is shown below in Figure 18.

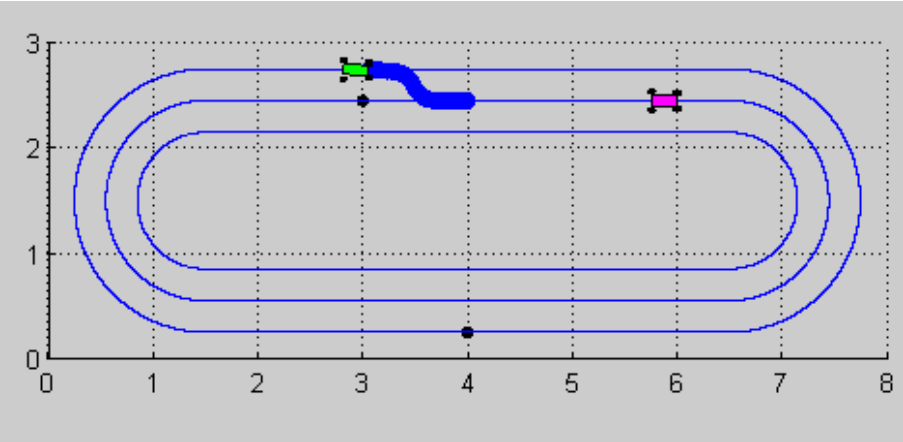


Figure 18 Lane Changing Path

The obstacle avoidance is based on a set of states that are used to create optimal control. The input states for the obstacle avoidance controller are:

$$\begin{bmatrix} x_1 \\ \vdots \\ x_8 \\ x_9 \\ \vdots \\ x_{16} \\ x_{17} \\ x_{18} \\ x_{19} \\ x_{20} \\ x_{21} \end{bmatrix} = \begin{bmatrix} \text{distance sensor 1} \\ \vdots \\ \text{distance sensor 8} \\ \text{velocity from distance sensor 1} \\ \vdots \\ \text{velocity from distance sensor 8} \\ \text{lane we are in} \\ v_1 \\ \theta \\ \text{System Mode} \\ \text{x location of vehicle} \end{bmatrix} \qquad [3.2\text{-}20]$$

The control variables are:

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \begin{bmatrix} u_1 \\ \text{turn\_direction} \\ \text{System Mode} \end{bmatrix} \quad [3.2\text{-}21]$$

The cost function that needs to be minimized over u_2=[-1,0,1] to optimally control the obstacle avoidance/lane changing behavior as defined by:

$$J^* = \min_{u_2=[1,0,-1]} \begin{bmatrix} x_1 k_1 (u_2^2 - 1) - (x_1 + k_2) u_2^2 + k_3 (x_{13} <= .5) u_2^2 + (u_2 == -1)(x_{17} == 1) + (u_2 == 1)(x_{17} == 3) \\ +5 u_2^2 (x_{21} > maxcurvex) \mid (x_{21} < mincurvex) \end{bmatrix} \quad [3.2\text{-}22]$$

Below are the optimal gains for the cost function:

$k_1 = 2.2$

$k_2 = 0.5$

$k_3 = -1.0$

A state machine with four states is used to control the lane change behavior and its behavior is described below.

The first state is when the car is following the white line and is not changing lanes. The state machine sends d, θ_p, and $c(s)$, to the car controller (state and calculations for v_1 and v_2).

The second state is when the obstacle avoidance system first determines the car needs to change lanes. This state initializes the lane change parameters, and d, θ_p, and $c(s)$ are now calculated by the lane change function and are not based on the simulated IR sensors. The following variables are set as shown below.

In_Lane_Change=1

$\theta_i = \theta$

$c(s)=0$

$d=0$

$\theta_p=0$

lane_change_direction_latched=lane_change_direction

The third state is when the car makes the first turn. This turn is executed by forcing the path curvature value ***c(s)*** such that the car controller believes that it is following a curved path that is smaller then the limit on ***phi*** will allow. This forces the wheel to the limit as fast as possible for the "Bang Bang" optimal control on ***phi***. The state is terminated when the car θ has turned by ***Turn_Alpha*** radians, the state machine then advances to the next state.

$$c(s) = \frac{lane\_change\_direction\_latched}{Rad\_of\_Lane\_Change\_Turn} \qquad [3.2\text{-}23]$$

The fourth state is when the car makes the second turn in the opposite direction, to come back to the original θ. This turn is executed by forcing the path curvature value ***c(s)*** such that the car controller believes that it is following a curved path that is smaller then the limit on ***phi*** will allow. This forces the wheel to the limit as fast as possible for the "Bang Bang" optimal control on ***phi***.

$$c(s) = \frac{-lane\_change\_direction\_latched}{Rad\_of\_Lane\_Change\_Turn} \qquad [3.2\text{-}24]$$

The state is terminated when the car θ has turned back to θ_i, the state machine is then reset to the first state. The lane the car is in state is now updated. The following variables are set as shown below.

In_Lane_Change=0

System_Mode=0

lane_we_are_in=lane_we_are_in+lane_change_direction_latched [3.2-25]

3.2.8 Example of 3 Cars and 3 Static Obstacles

The example shown in Figure 19 below shows the paths of three vehicles each one has adaptive cruise control along with the optimal control collision avoidance controller. When the vehicle detects the static obstacle and there is a car in another lane that it will

slow down and change lanes to avoid the static obstacle and the vehicle that is in the next lane over.

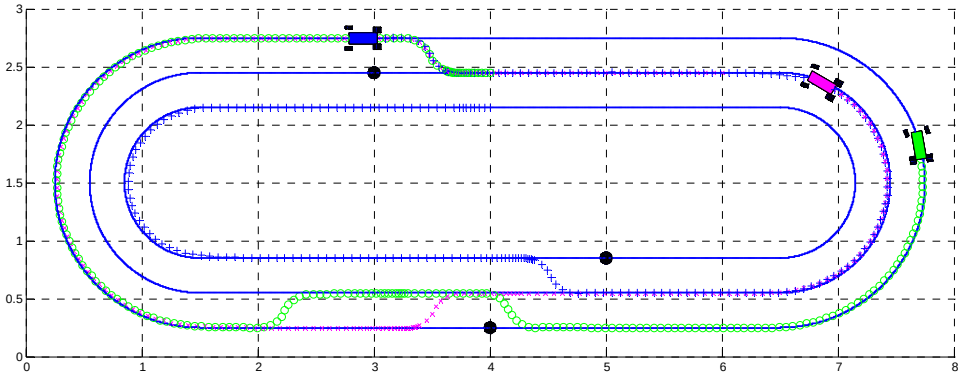


Figure 19 Lane Changing Path with 3 Cars and 3 Static Obstacles

with Optimal control

3.2.9 Fuzzy Control

The fuzzy controller was created using the FIS Editor in MATLAB. The four input states that are currently used in the optimal control subroutine ***obstacle_avoidance_cost_function_control_law.m*** were defined as inputs for the fuzzy controller as were the three outputs. Membership functions for each input and output were created so that rules could be written to perform the tasks required for collision avoidance and adaptive cruise control. The fuzzy controller overview in the FIS editor is shown below.

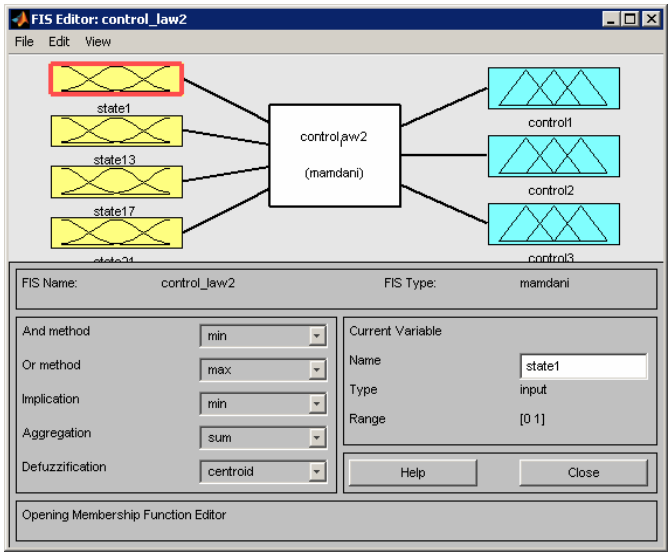


Figure 20 Fuzzy Controller Overview

The input ***state1*** is the distance an obstacle is in front of the car has four membership functions: ***turn***, ***do not turn***, ***acc***, and ***slow down***. These membership functions are shown in the FIS Editor below. The reason for the zmf membership function for ***turn*** is because the car needs to turn when the obstacle distance is less than 0.4 meters away. The reason for the sigmf for ***do not turn*** is because the car does not need to turn when the obstacle distance is more than 0.4 meters away. See Figure 29 on page 49. The ***slow down*** and the ***acc*** memberships functions are pimf and trimf respectively. These membership functions were chosen so that the closer the car got to an obstacle the slower the vehicle would go (slow down) and the farther the car was from an obstacle the faster it would go (acc). The shapes and positions were adjusted such that automatic cruise control would respond as desired. See Figure 31 on page 50.

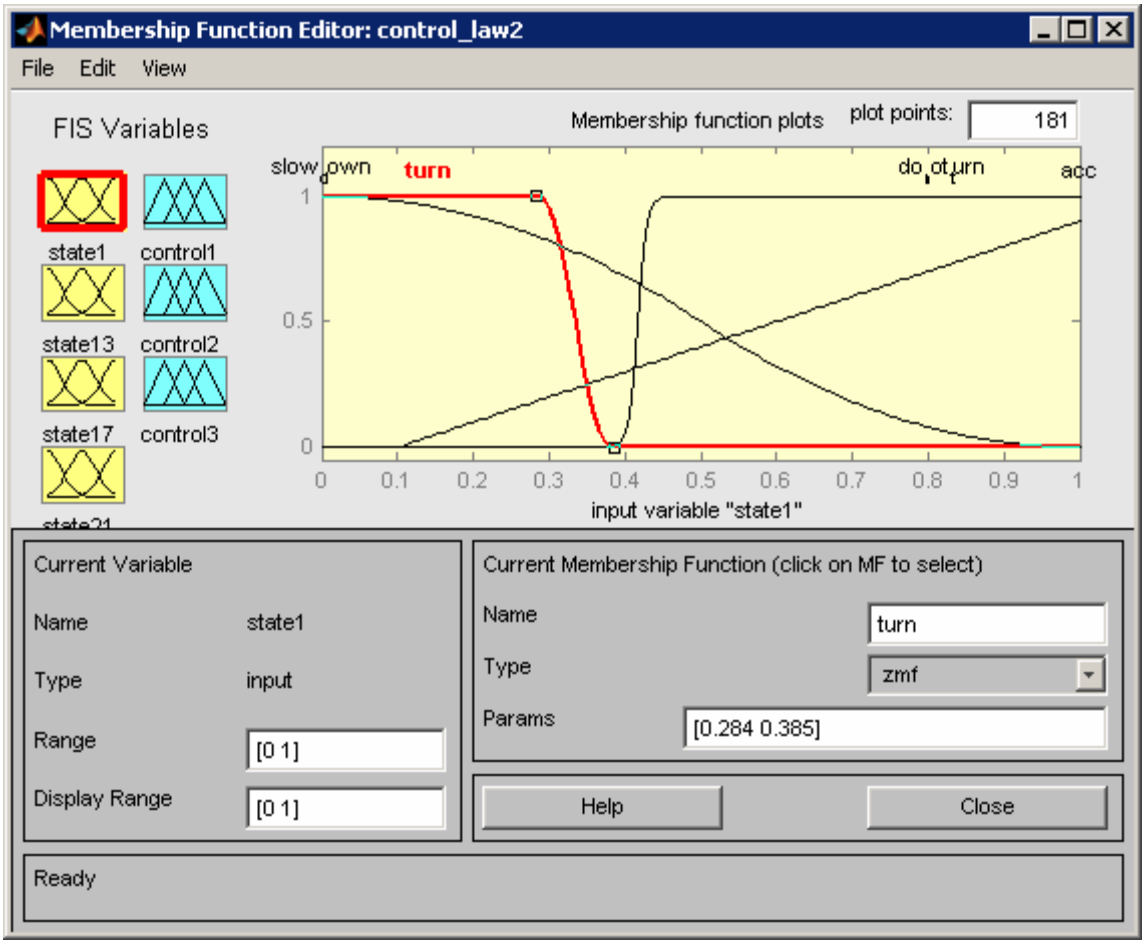


Figure 21 Fuzzy Control Input ***State1***

The input ***state13*** is the velocity of objects from the rear of the vehicle. There is one membership function ***turn_rear_speed***. The membership function is shown in the FIS Editor below. The reason for the zmf membership function for the ***turn_rear_speed*** is because the car needs to turn when another car approaches from the rear with a velocity less than -0.35 meters per second. See Figure 30 on page 49.

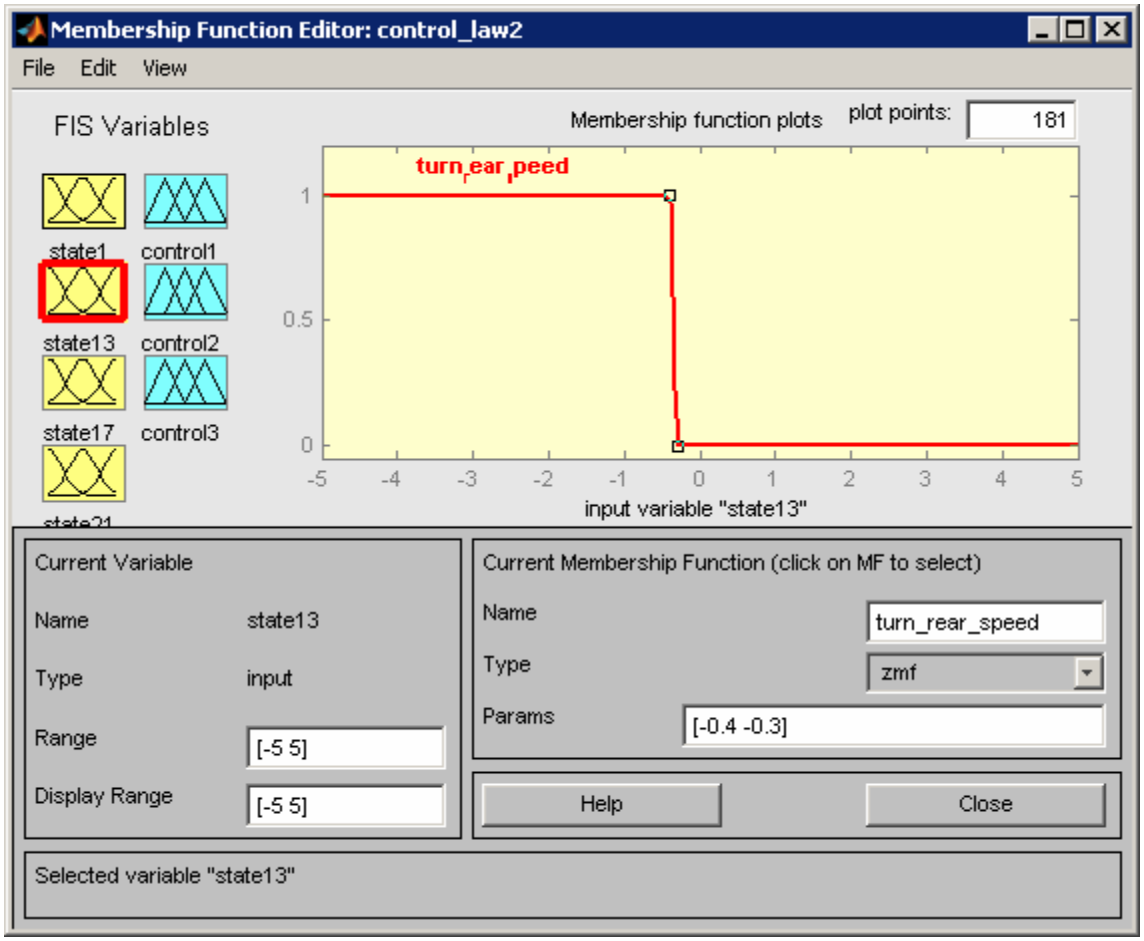


Figure 22 Fuzzy Control Input ***State13***

The input ***state17*** is the lane we are in. There are three membership functions for the car ***inlane1***, ***lane2***, and ***inlane3***. The membership functions are shown in the FIS Editor below. The membership function for ***inlane1*** is trimf so ***inlane1*** would be true when ***state17*** equals 1. The membership function for ***lane2*** is trapmf so ***lane2*** would be true when ***state17*** equals 2. The membership function for ***inlane3*** is trimf so ***inlane3*** would be true when ***state17*** equals 3. These are used to ensure the car turns in the correct direction based on the lane it is in.

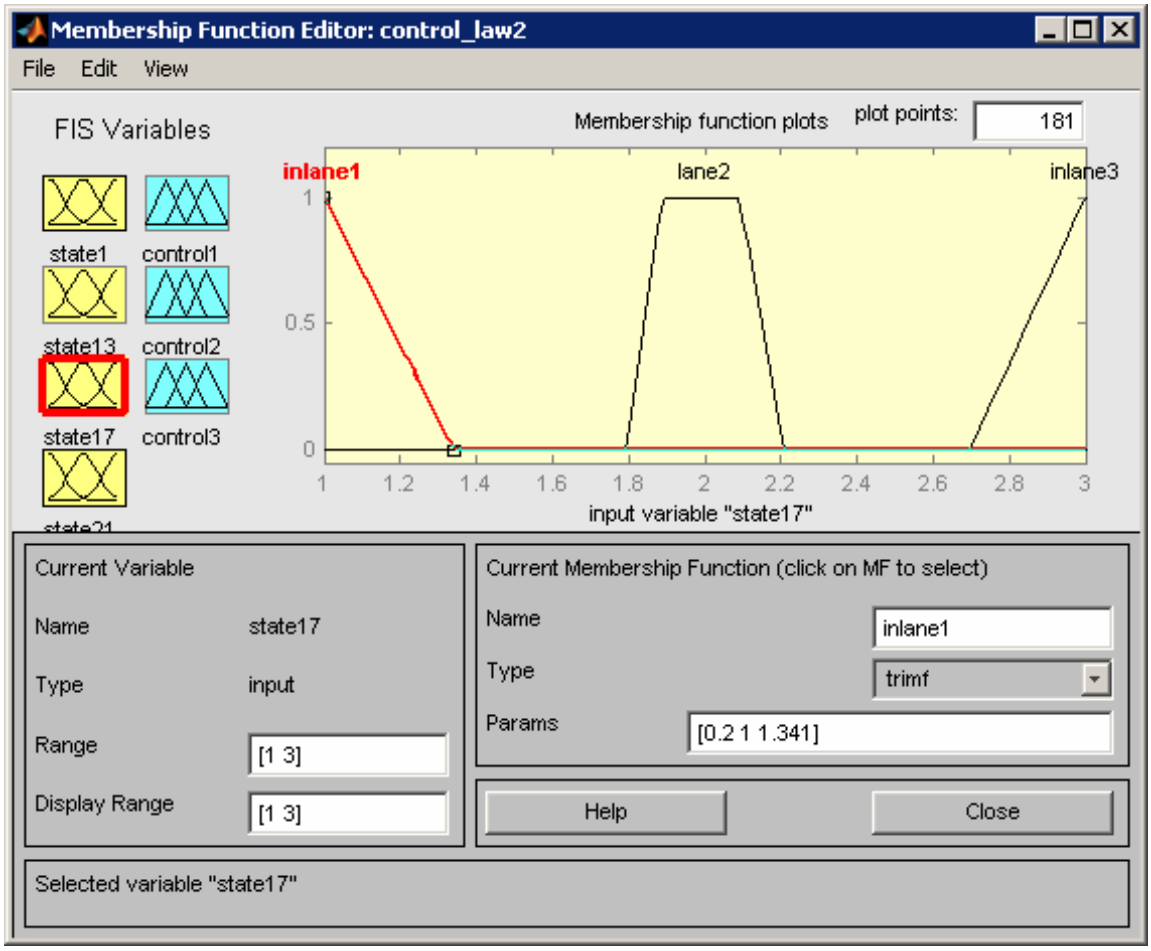


Figure 23 Fuzzy Control Input ***State17***

The input ***state21*** is the x location of the car. There are two membership functions ***no_turn_x_curve1*** and ***no_turn_x_curve2***. The membership functions are shown in the FIS Editor below. The reason for the zmf membership function for ***no_turn_x_curve1*** is so the car does not turn on the curve on the left side of the track. The reason for the sigmf membership function for ***no_turn_x_curve2*** is so the car does not turn on the curve on the right side of the track.

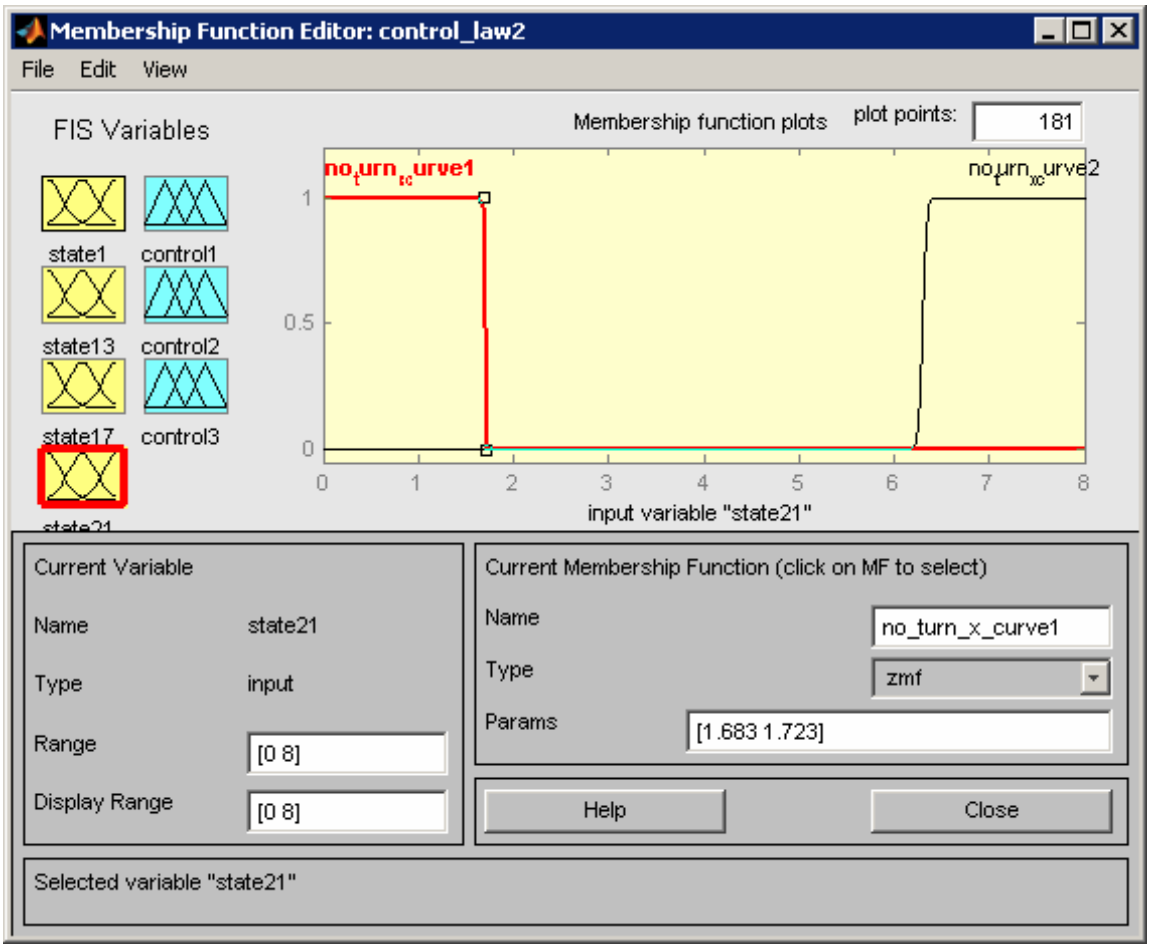


Figure 24 Fuzzy Control Input ***State21***

The output ***Control1*** is the vehicle speed of the car. There are two membership functions ***brake*** and ***speed***. The membership functions are shown in the FIS Editor below. The membership function for ***speed*** is smf because for the automatic cruise control, it needs to increase speed according the fuzzy logic rules. The membership function for ***brake*** is zmf because for the automatic cruise control, it needs to decrease speed according the fuzzy logic rules. See Figure 31 on page 50.

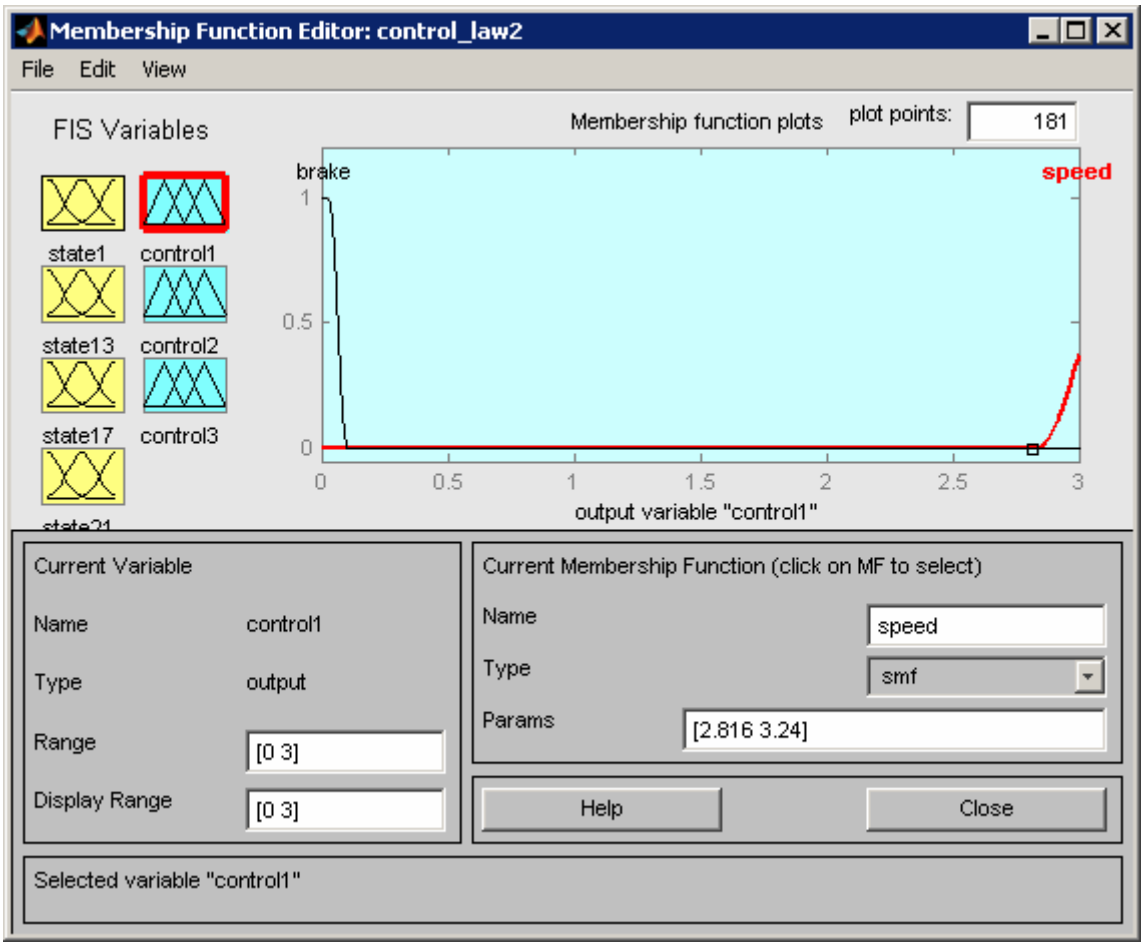


Figure 25 Fuzzy Control Output ***Control1***

The output ***Control2*** is the turn direction of the car. There are three membership functions ***turn_right, do_not_turn*** and ***turn_lef***t. The membership functions are shown in the FIS Editor below. The membership function for ***turn_right*** is trimf, so when the fuzzy logic rules determine that the car should turn right, ***control2*** will have a value of -1. The membership function for ***do_not_turn*** is trimf, so when the fuzzy logic rules determine that the car should not turn, ***control2*** will have a value of 0. The membership function for ***turn_left*** is trimf, so when the fuzzy logic rules determine that the car should turn left, ***control2*** will have a value of 1.

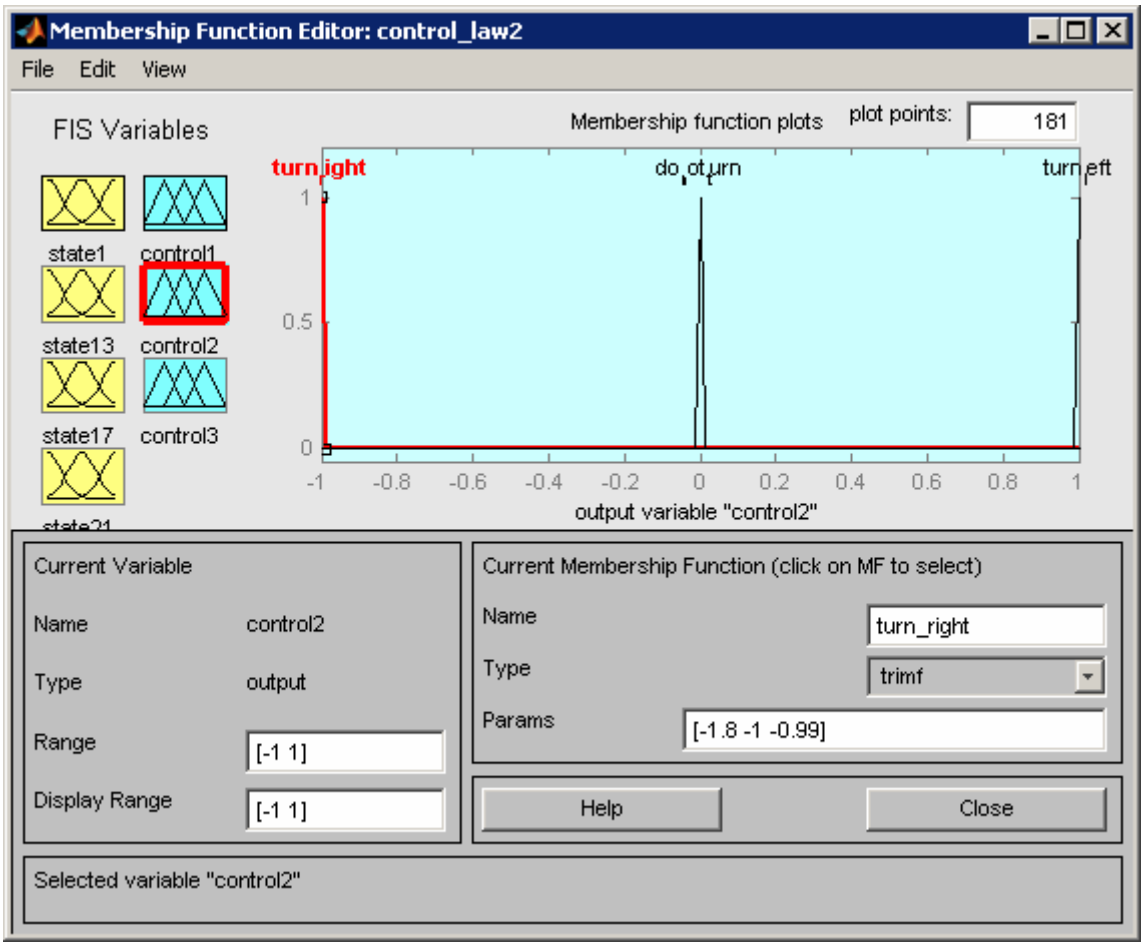


Figure 26 Fuzzy Control Output ***Control2***

The output ***Control3*** is the System Mode for the car. There are two membership functions ***no_lane_change*** and ***lane_change***. The membership functions are shown in the FIS Editor below. The membership function for ***no_lane_change*** is zmf, so when the fuzzy logic rules determine that the car should not change lanes, ***control3*** will have a value of 0. The membership function for ***lane_change*** is sigmf, so when the fuzzy logic rules determine that the car should change lanes, ***control3*** will have a value of 1.

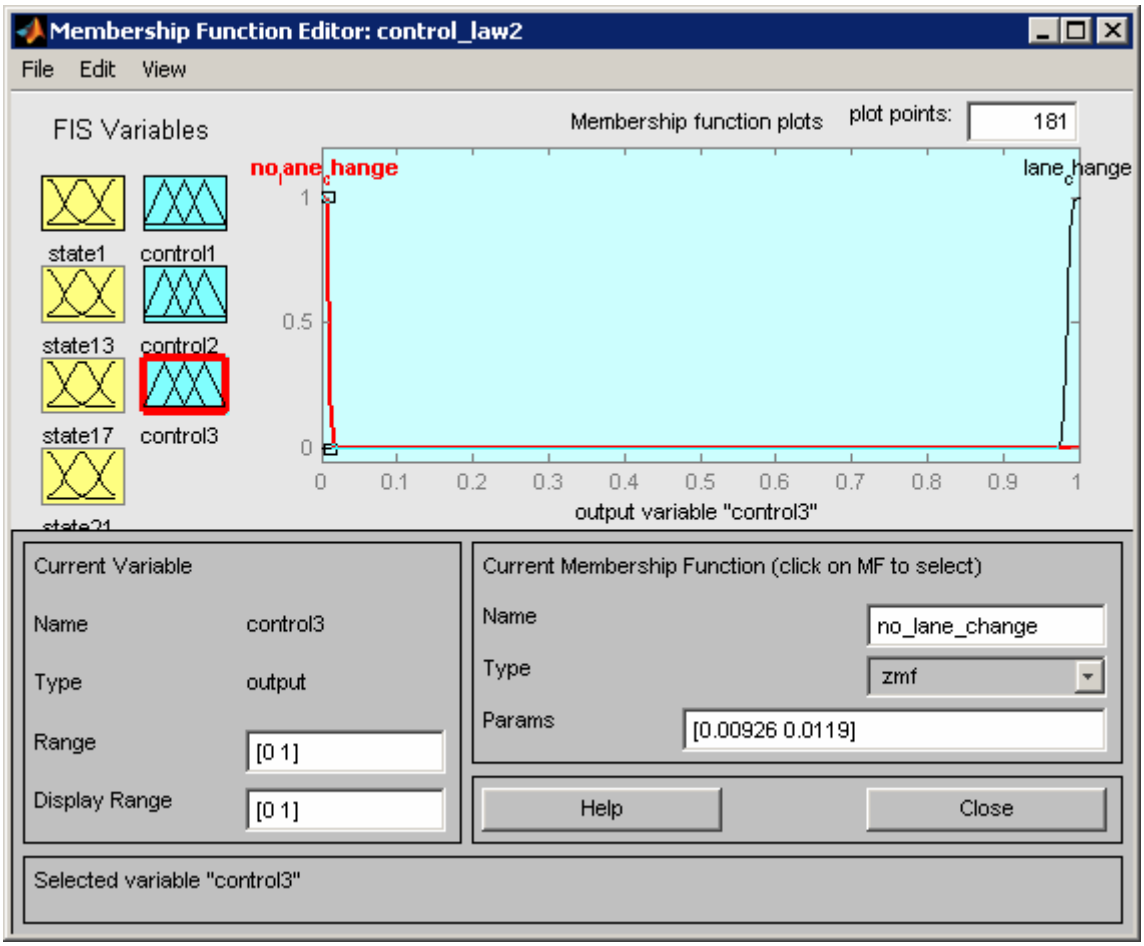


Figure 27 Fuzzy Control Output ***Control3***

The rules that describe how the fuzzy logic calculates the outputs given the inputs are shown in the FIS editor below.

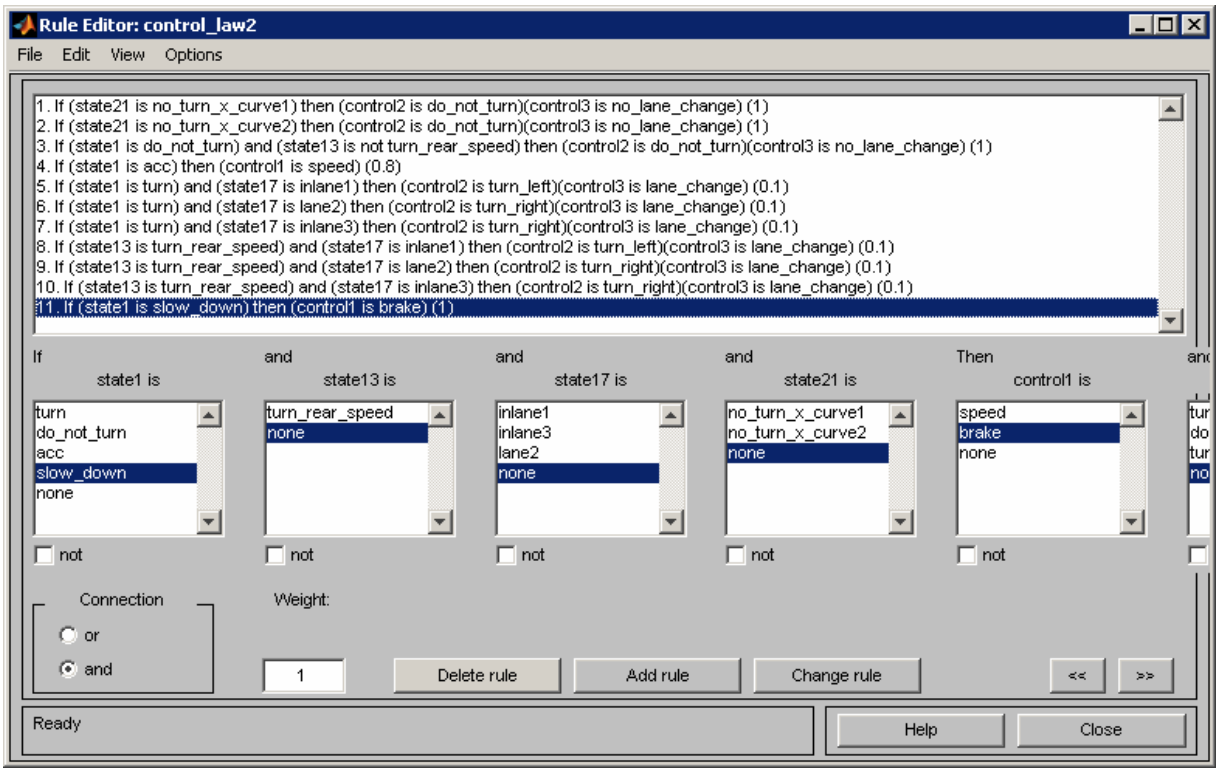


Figure 28 Fuzzy Control Rules

The figure below shows the output control 2 lane change direction varies versus state 1 (distance from obstacle in front of the car) and state 21 (x location of the car)

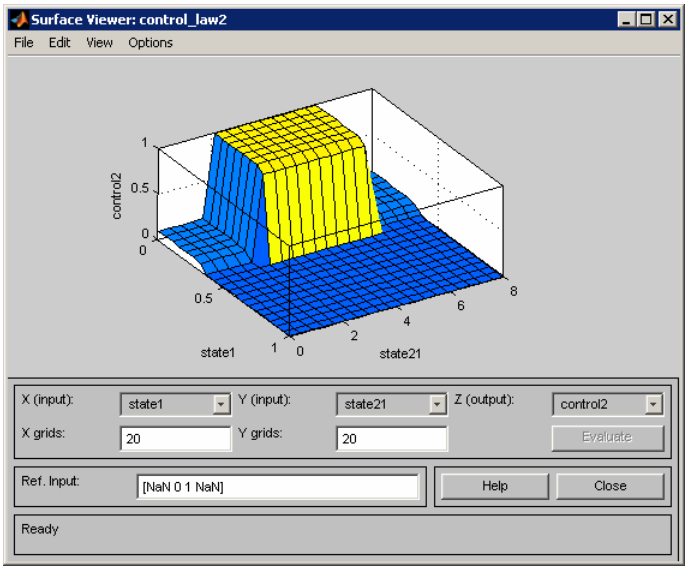


Figure 29 Fuzzy Controller Output (***Control2*** vs ***State1*** and ***State21***)

The figure below shows the output ***control2*** lane change direction versus ***State13*** (velocity of the car approaching from the rear) and ***State21*** (x location of the car)

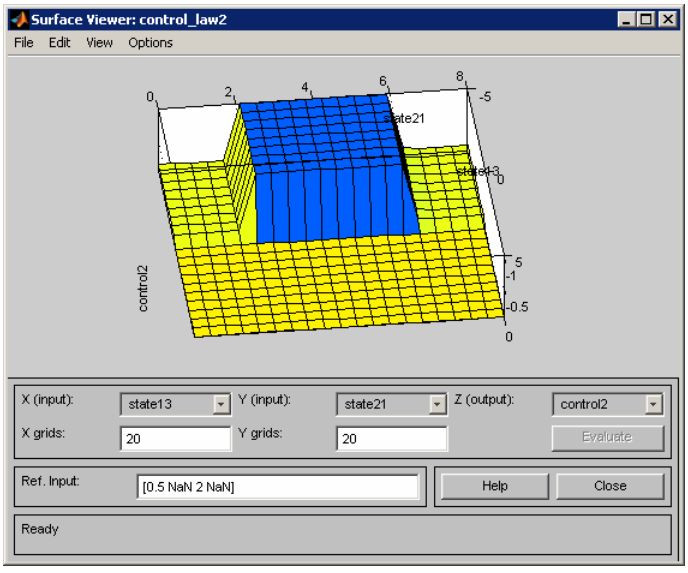


Figure 30 Fuzzy Controller Output (***Control2*** vs ***State13*** and ***State21***)

The figure below shows the output ***control1*** car velocity versus ***State1*** (distance the obstacle is in front of the car) and ***State13*** (velocity of the car approaching from the rear)

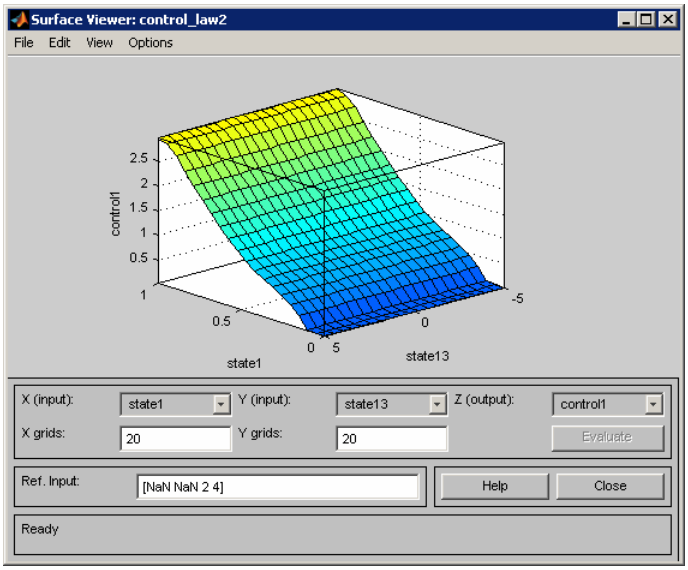


Figure 31 Fuzzy Controller Output (***Control1*** vs ***State1*** and ***State13***)

The optimal control obstacle avoidance controller was replaced by the fuzzy controller in the simulation. The simulation was run again with the same settings and the results (below) were virtually the same.

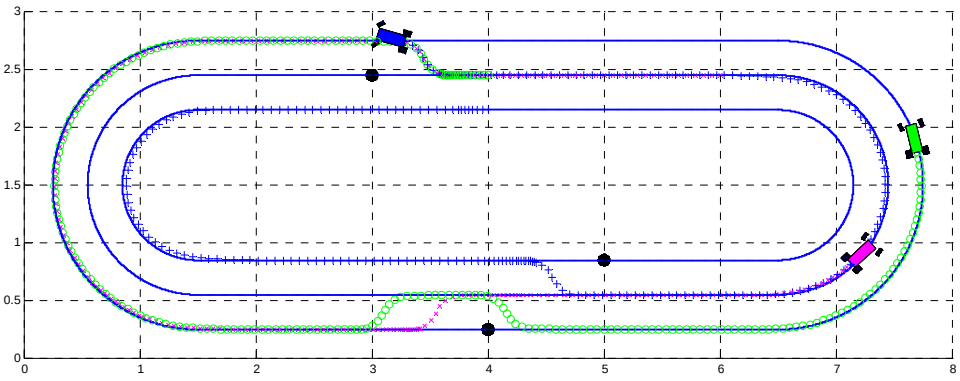


Figure 32 Lane Changing Path with 3 Cars and 3 Static Obstacles

with a Fuzzy Controller

3.2.10 How to Work in the MATLAB/Simulink Simulation Environment

3.2.10.1 Adding a Car

The way to add more cars is to copy either the car 2 or 3 subsystem and paste it anywhere in the main window of Simulink.

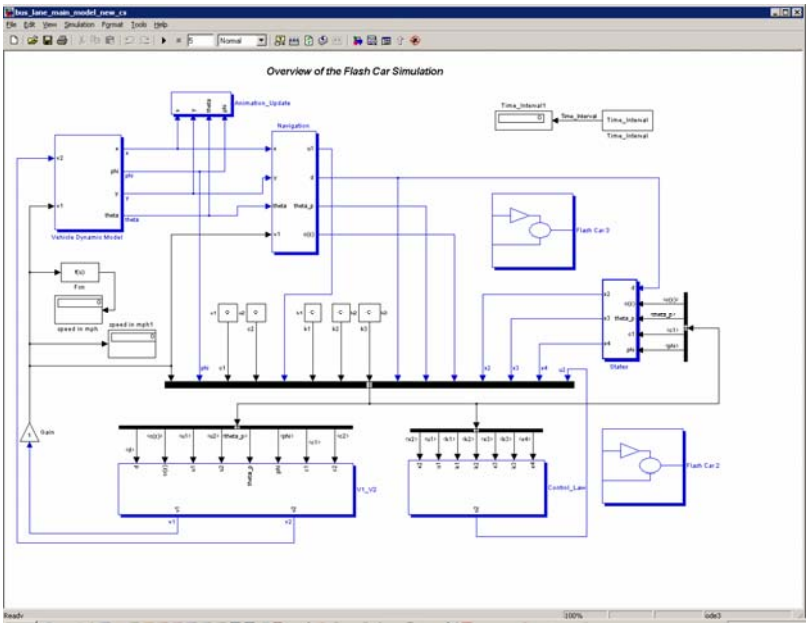


Figure 33 Main Screen for the Car Simulation

Next, go into the vehicle dynamics of the new car model that was just created and change the initial condition of each the integrators to a new variable. See example below.

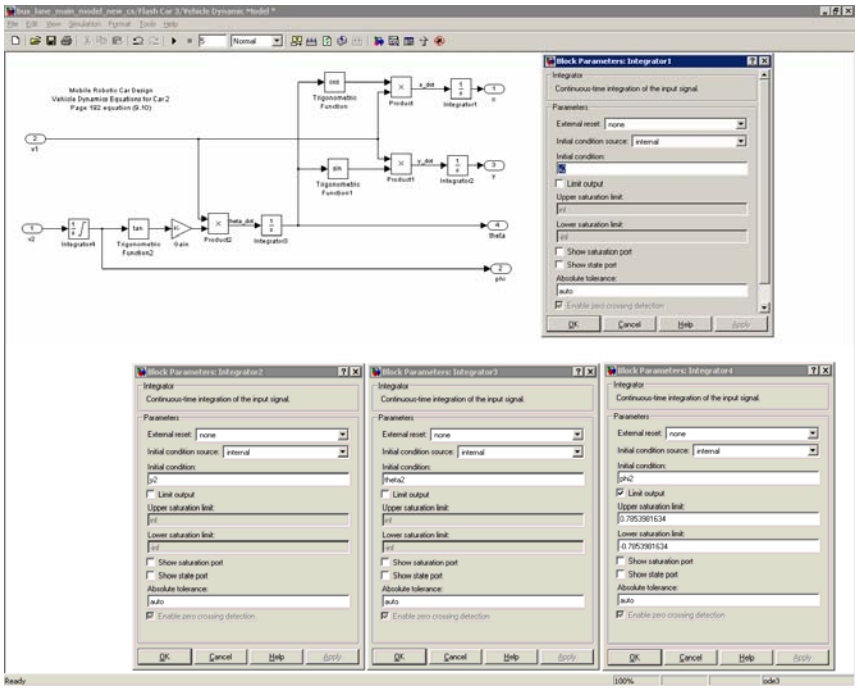


Figure 34 Vehicle Dynamics for New Car

Then go to the lane changing block and change the initial conditions of the unit delay by the red note. Again, see the example below.

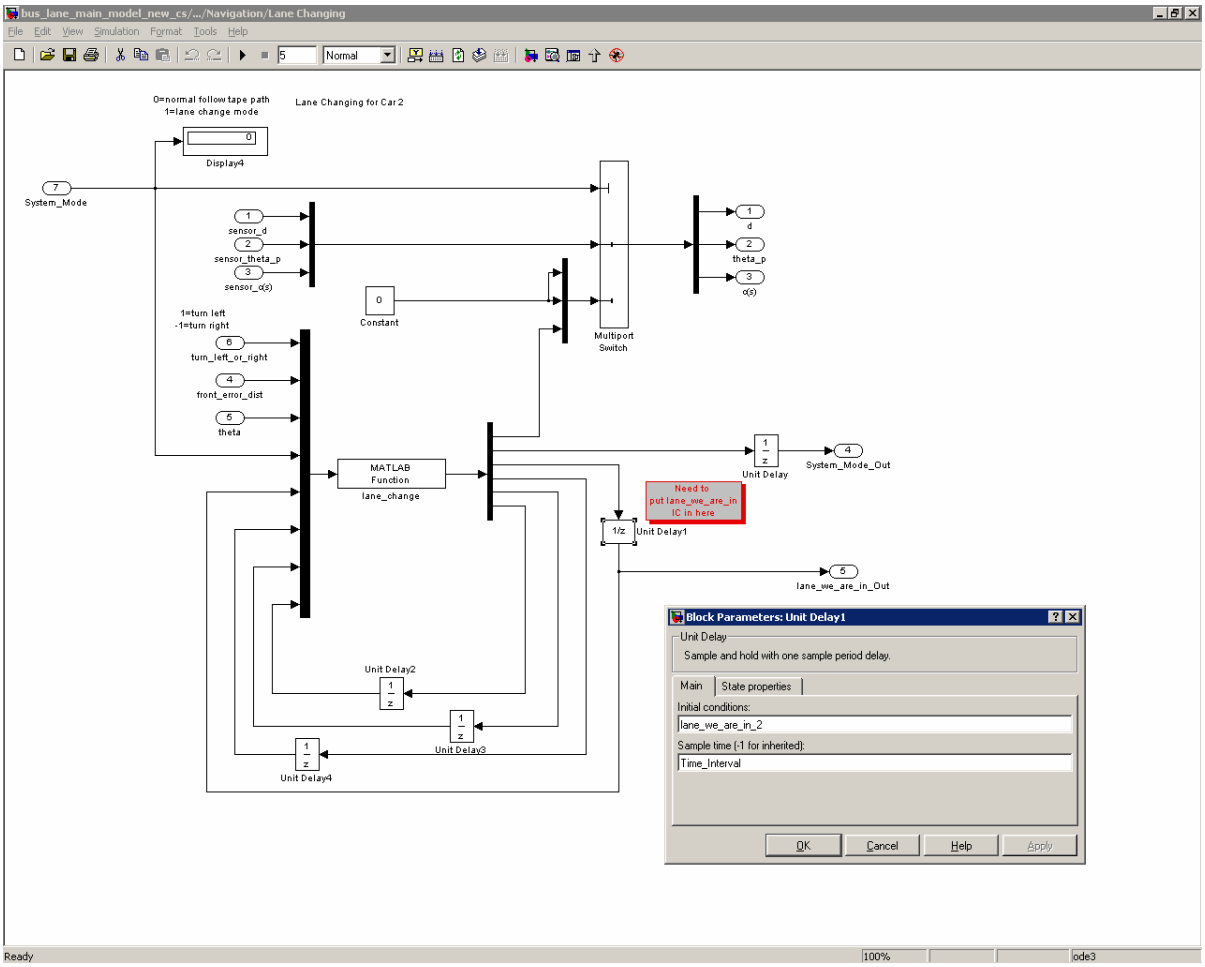


Figure 35 Initial Lane for New Car

Go to update animation block and change the constant number to the number of the car just created. See details below.

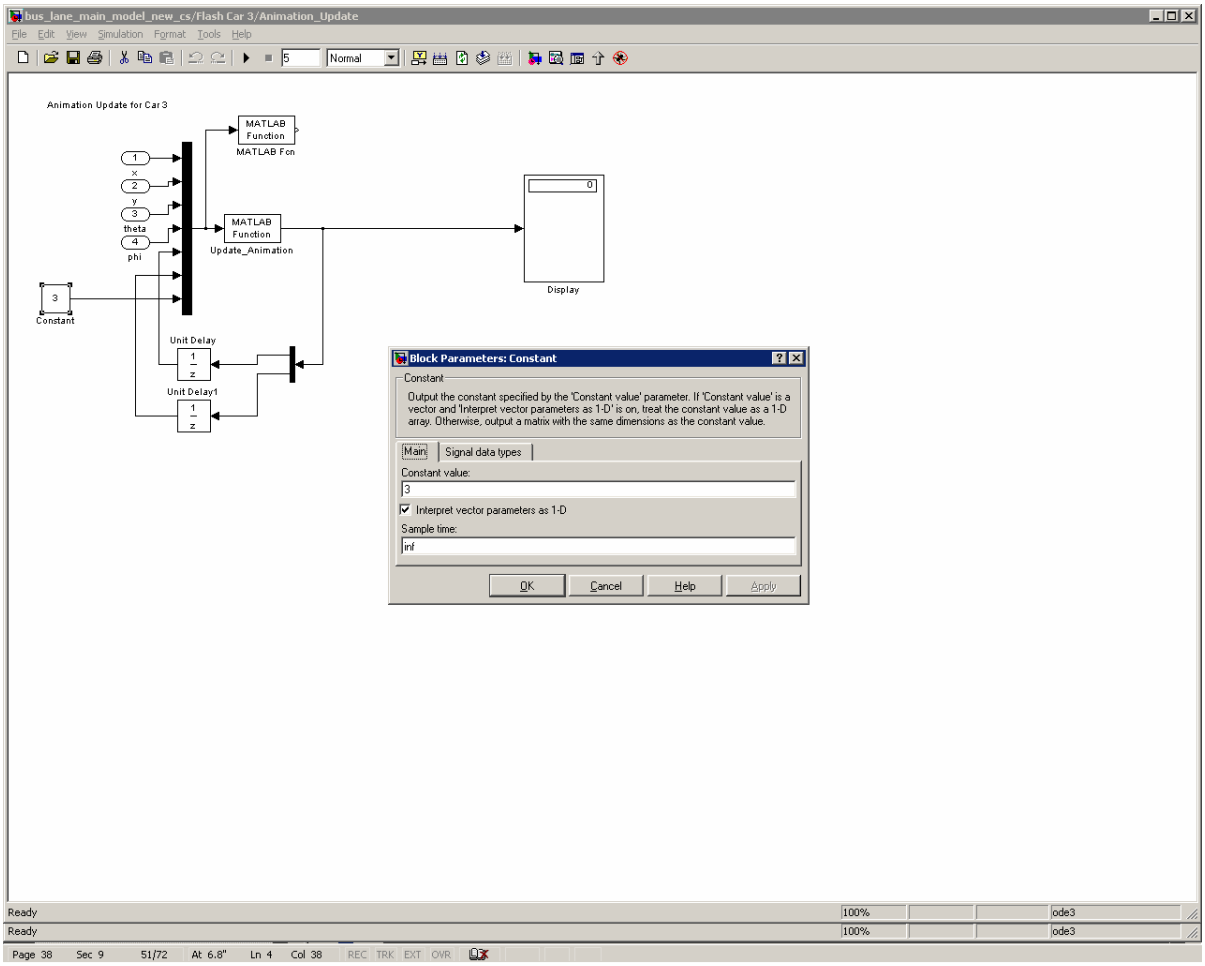


Figure 36 Animation Update Vehicle Number for New Car

Update the MATLAB function initialize_function_graphic.m by adding the car# in the global variables. Copy Car 2 location and paste it below car 2 renaming all variables car#. Go to the end of the initialize function and copy the create car information, re-labeling the location and car# where it is highlighted in yellow below.

```
car3 = CreateCar(Width_Between_Rear_Wheels, Width_Between_Front_Wheels,
Length_between_axles, Height_of_Car,...Diameter_of_Rear_Wheels,
Diameter_of_Front_Wheels,Width_of_Front_Wheels, Width_of_Rear_Wheels,'blue');
locate(car3,[x2 y2 0]);
aim(car3,[x2+cos(theta2) y2+sin(theta2) 0]);
turn(car3.tire_fl,'z',phi2*180/pi);
turn(car3.tire_fr,'z',phi2*180/pi);
```

Next, edit the update_obstacle.m file by copying and pasting the statements below

```
if (inputvar(7)==3)
   static_obstacle(maxobstacle+3,1)=inputvar(1);
   static_obstacle(maxobstacle+3,2)=inputvar(2);
   static_obstacle(maxobstacle+3,3)=Length_between_axles/2;
   end
```

Changing what is highlighted to the next number.

Lastly, change update_animation.m file, copy what is below, changing what is highlighted to the next number and the '+b' is the color and shape of the trails for the vehicle.

```
if car_number==3
    plot(x0,y0,'+b');
   display_car=car3;

end
```

3.2.10.2 Adding an Obstacle

Add another x,y, and radius coordinate location as another row to the variable below in the initialize_function_graphic.m file.

```
static_obstacle=[3 2.45 .050;4 .25 .050; 5 .85 .050]; %Creates an obstacle on the track
```

3.2.10.3 Modifying the Track Shape

The create_road_map.m file uses two subroutines to create the white lines. One creates horizontal lines and an example is shown below

```
temp=h_line(1.5,.85,6.5,.85,tape_width);
x_road=cat(2,x_road,temp(1,1:length(temp)));
y_road=cat(2,y_road,temp(2,1:length(temp)));
```

The half circles at either end of the track are created with another routine, as in the example shown below.

```
temp=curves(6.5,1.5,.950,3*pi/2,tape_width);
x_road=cat(2,x_road,temp(1,1:length(temp)));
y_road=cat(2,y_road,temp(2,1:length(temp)));
```

3.3 Model Car

3.3.1 Model Car Operations

As it stands the car can detect and avoid an obstacle with the IR sensor mounted in the front. The car changes lanes if the obstacle gets too close to the front. The car will not go off the road since the cost function increases the cost to make a right turn from the right lane or a left turn from the left lane. This makes the cost of those choices not optimal.

3.3.2 Model Car Dynamics

Since the original car software did not have any need to estimate the car's heading based on the car's dynamics, there needed to be a function added that estimated the heading so the car could use the same lane changing function the simulation used.

Equation 3.3-1 is from [5] and Equation 3.3-2 is the discrete time integral of Equation 3.3-1.

$$\textit{theta\_dot}=\textit{actual\_velocity}*\tan(\textit{phi})/\textit{L}; \quad [3.3\text{-}1]$$

$$\textit{theta}=\textit{theta\_dot}*\textit{T}+\textit{theta}; \quad [3.3\text{-}2]$$

3.3.3 Lane Changing

The car's lane changing worked just like the simulation except that the vehicle's maximum value of phi was different depending on if the vehicle was turning to the left or right. A second variable ***turn_alpha1*** was created to compensate for this discrepancy. Parameters that are different compared to the simulation are ***turn_alpha*** = 0.82, ***turn_alpha1*** = 1, and ***LANE_CHANGE_FE_DIST*** = 0.002.

3.3.4 Obstacle Detection

There is an IR sensor on the front of the car that detects the obstacles. The car uses the Sharp GP2D12 range finder, which can detect obstacles from 10 to 80 cm away. Unlike the simulation the car has only one sensor that detects obstacles in the front.

3.3.5 Obstacle Avoidance

Obstacle Avoidance is done using optimal control. There is a cost function that calculates the cost of whether to turn left or right or stay in the same lane when it sees an obstacle in its way. The function selects the control with the lowest cost.

3.3.6 Car DSP Code

The car DSP code was taken from [5] and [7]. Changes were made to the control.c program along with adding the subroutines lane_change.c and obstacle_avoidance_cost_function_control_law.c. The two subroutines were adapted from the MATLAB m files of the same names used in the simulation. The existing range finder that is on the front of the vehicle was used for obstacle detection. The steering control in the existing car code had the calculated value for phi_dot being used as phi. This is modified to integrate phi_dot to calculate phi, and then to send it to the servo.

3.3.6.1 Control.c

The changes to the control.c are described below:

When the vehicle is following the white line, ie. it is not changing lanes, then theta_p and d are calculated from the sensors. When the car is changing lanes (System Mode=1) then theta_p and d are set to 0. Theta_p_dot was changed to be always 0. The code for this is shown below.

```
if (System_Mode==0)
{
theta_p = atan((front_error-back_error)/L);
d = back_error;
}
else
{
theta_p=0;
d=0;
```

```
}
```

The original car had no need for calculating the vehicle heading theta, however in order to do the lane change maneuver; we need to have an arbitrary heading when the vehicle starts the maneuver in order to control the vehicle during the lane change maneuver. The code below is the discretized version of the vehicle state equations for theta.

```
theta_dot=actual_velocity*tan(phi)/L;
theta=theta_dot*T+theta;
```

Next, the states for the obstacle_avoidance_cost_function_control_law for the subroutine are setup. These states match the states in the MATLAB simulation.

```
states[1]=distance;  //front distance
states[5]=1.0; // rear distance
states[DISTANCE_SENSORS+1]=(distance-p_distance)/T; //front vel
states[DISTANCE_SENSORS+5]=0.0;  //rear vel.
p_distance=distance;
states[2*DISTANCE_SENSORS+1]=(float)(lane_we_are_in);
states[2*DISTANCE_SENSORS+2]=actual_velocity;
states[2*DISTANCE_SENSORS+3]=theta;
states[2*DISTANCE_SENSORS+4]=System_Mode;
```

The obstacle_avoidance_cost_function_control_law is called and returns the control variables in the variable controls. Again, this corresponds to the MATLAB simulation. The controls outputs are stored in the variables System_Mode, lane_change_direction, and u1.

```
obstacle_avoidance_cost_function_control_law(states,controls);
System_Mode=controls[3];
lane_change_direction=controls[2];
u1=controls[1];
```

The lane_change subroutine is called to change lanes as directed by the obstacle_avoidance_cost_function_control_law. This subroutine corresponds to the MATLAB function with the same name. The values in return_val are then assigned to the appropriate variables.

```
lane_change(lane_change_direction,front_error,theta,System_Mode,
            lane_we_are_in, In_lane_Change, theta1,
            lane_change_direction_latched,loop_counter, return_val);
c=return_val[1];
System_Mode=return_val[2];
lane_we_are_in=return_val[3];
In_lane_Change=return_val[4];
theta1=return_val[5];
```

```
lane_change_direction_latched=return_val[6];
```

3.3.6.2 Obstacle_avoidance_cost_function_control_law.c

The obstacle_avoidance_cost_function_control_law subroutine is virtually the same in the MATLAB with the same name. The differences are due to C starting with an array index 0 instead of 1 for the simulation and the find function in MATLAB had to be created for the C program. MATLAB allowed for matrix arithmetic so in the C program, each element of the vector J was calculated separately

```
void obstacle_avoidance_cost_function_control_law(float *state, float *control)

{
int i, imin;
float k1, k2, k3, J[3], Jmin, u2[3];

/*
   control[3]=state[2*DISTANCE_SENSORS+4];
   i=1;
   k1=2.5;k2=.5;k3=-1; //weights for cost function k1 was 1.95
   u2[0]=-1.0; //all possible values for control(2)
   u2[1]=0.0; //all possible values for control(2)
   u2[2]=1.0; //all possible values for control(2)

   J[0]=state[1]*k1*(u2[0]*u2[0]-1)+(state[1]+k2)*(-u2[0]*u2[0])+k3*(float)(state[13]<-
.5)*(u2[0]*u2[0])+(float)(u2[0]==-1)*(float)(state[2*DISTANCE_SENSORS+1]
==1.0)+(float)(u2[0]==1.0)*(float)(state[2*DISTANCE_SENSORS+1]==MAX_NUM_
LANES);

   J[1]=state[1]*k1*(u2[1]*u2[1]-1)+(state[1]+k2)*(-u2[1]*u2[1])+k3*(float)(state[13]<-
.5)*(u2[1]*u2[1])
     +(float)(u2[1]==-1)*(float)(state[2*DISTANCE_SENSORS+1]
==1.0)+(float)(u2[1]==1.0)*(float)(state[2*DISTANCE_SENSORS+1]==MAX_NUM_
LANES);

   J[2]=state[1]*k1*(u2[2]*u2[2]-1)+(state[1]+k2)*(-u2[2]*u2[2])+k3*(float)(state[13]<-
.5)*(u2[2]*u2[2]) +(float)(u2[2]==-1)*(float)(state[2*DISTANCE_SENSORS+1]
==1.0)+(float)(u2[2]==1.0)*(float)(state[2*DISTANCE_SENSORS+1]
==MAX_NUM_LANES);
```

The code below replaces the find MATLAB function.

```
   state(1)*k1*(u2.*u2-1)+(state(1)+k2)*(-u2.*u2)
         Jmin=500.0;
         for (i=0;i<=2;i=i+1)
                  {
```

```
                if (J[i]<=Jmin)
                        {
                        Jmin=J[i];
                        imin=i;
                        }
                }
        control[2]=u2[imin];
```

The following code differs only in syntax to the code in MATLAB.

```
   if (control[2]!=0)
      {
      control[3]=1;
      }
   if ((state[1]-.2)>0)
      {
         control[1]=CAR_SPEED*(state[1]-.2)/.8; //acc
      }
         else
         {control[1]=0;}
}
```

3.3.6.3 Lane Change.c

The lane_change subroutine is virtually the same in the MATLAB with the same name. The differences are due to C starting with an array index 0 instead of 1 for the simulation. During testing it was discovered that the vehicle turns with a different radius, when turning right or left. This causes the vehicle to not be able to change from right lane to left lane and back again, and still follow the white line. To overcome this, two different values of turn_alpha are used depending on the direction of the lane change.

```
void lane_change(int lane_change_direction,float front_error_dist,float theta,
            int System_Mode, int lane_we_are_in, int In_Lane_Change,float theta1,
            int lane_change_direction_latched,int loop_counter,float *c_val )
{
float Turn_angle;
if (System_Mode!=1)
   {c_val[1]=0;}
if (System_Mode==1)
{
```

Below is the code for the state machine that controls the lane change maneuver. Case 0 initializes the state machine, state 1 is the first turn, state 2 is the second turn of the lane change maneuver.

```
   switch (In_Lane_Change)
   {
      case 0: //initalize lane change parameters
         In_Lane_Change=1;
         theta1=theta;
         c_val[1]=0;
         lane_change_direction_latched=lane_change_direction;
         break;
      case 1: //1st curve of lane change
         Turn_angle=Turn_Alpha;
         if (lane_change_direction_latched==-1)
         Turn_angle=Turn_Alpha1;
         c_val[1]=lane_change_direction_latched/Rad_of_Lane_Change_Turn;
         if (fabs((theta1+lane_change_direction_latched*Turn_angle)-
            theta)<=TURN_COMPLETE_ERROR)
            {
            In_Lane_Change=2;
            }
         break;
      case 2: //2nd curve of lane change
         c_val[1]=-1*lane_change_direction_latched/Rad_of_Lane_Change_Turn;
         if (fabs(theta1-theta)<=TURN_COMPLETE_ERROR)
            {
            In_Lane_Change=0;
            System_Mode=0;
            lane_we_are_in=lane_we_are_in+lane_change_direction_latched;
            }
         break;
   }
}
```

The variables below are returned to the calling program control.c.

```
c_val[2]=System_Mode;
c_val[3]=lane_we_are_in;
c_val[4]=In_Lane_Change;
c_val[5]=theta1;
c_val[6]=lane_change_direction_latched;
}
```

CHAPTER IV

RESULTS

4.1 Simulation Results

After all the work deriving the optimal lane change trajectory, the simulation was a lot easier to complete. The simulation car avoids nonmoving obstacles as well as moving obstacles coming from behind it. When the simulated car avoids obstacles it takes the optimal trajectory when changing lanes. The simulated ultrasonic sensor behaves as the actual sensor does. The simulation was run under various conditions of obstacles and various speeds of the cars. Updating the vehicle position on the screen every 20ms makes a great improvement on the speed of the simulation over the simulation time interval of 0.4 ms. Running the simulation with a fixed step time in Simulink approximated the actual program of the real car, which calculates its parameters at the same fixed time interval of 0.4 ms. The simulation was run with the vehicle in all lanes and thoroughly tested obstacles and pursuing vehicles in various lanes, and everything worked as was expected.

The fuzzy controller works as well as the optimal control, but it is significant slower than the optimal control model. There are many advantages of fuzzy control over optimal control. The fuzzy control gives you a clearer understanding of the inputs and outputs. The rules are easier to understand than the numerical formulas of optimal control. Expanding the rules to include other collision avoidance behaviors are much easier using fuzzy control. The tools that are available in MATLAB to optimize the behavior of the controller are easier to use. There are no tools for the optimal control; you have to create your own tools. It is easier to see the different shapes and graphs because they are built into the fuzzy logic toolbox that you would otherwise have to create yourself for the optimal control. The disadvantages of fuzzy control over optimal control are; the

program runs much slower, at times there seems to be too many options to have the optimal solution, they are not as precise, and the decisions points are less clear.

4.2 Model Car Results

After all the simulation work was finished, converting the MATLAB/Simulink simulation to c code and integrating the code with the existing code was not difficult. The biggest issue was the car not being symmetric. When changing lanes to the left or right the car behaved differently. The servo was not calibrated correctly, which makes changing lanes to the right or left different. The minimum radius the car could turn is different if it turns right or left. The car track was limited to two lanes instead of the three lanes that were used in the simulation. Below are several photographs of the car as it changes lanes.

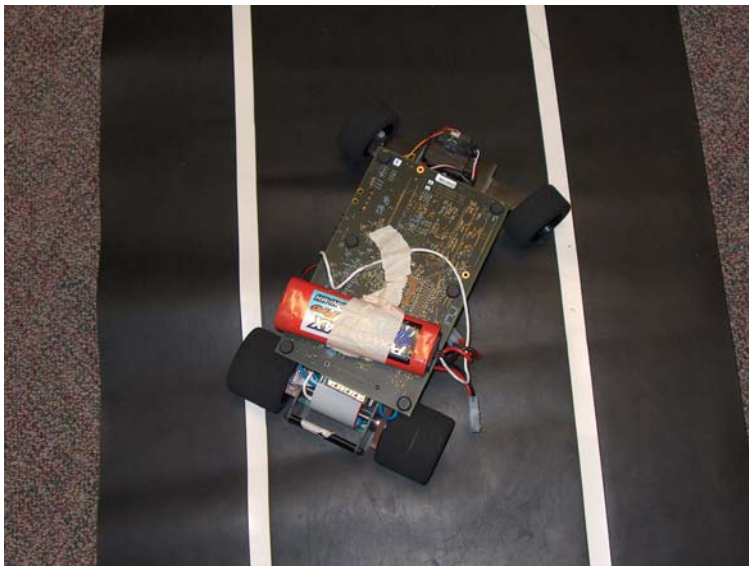


Figure 37 Car making first turn to avoid an obstacle

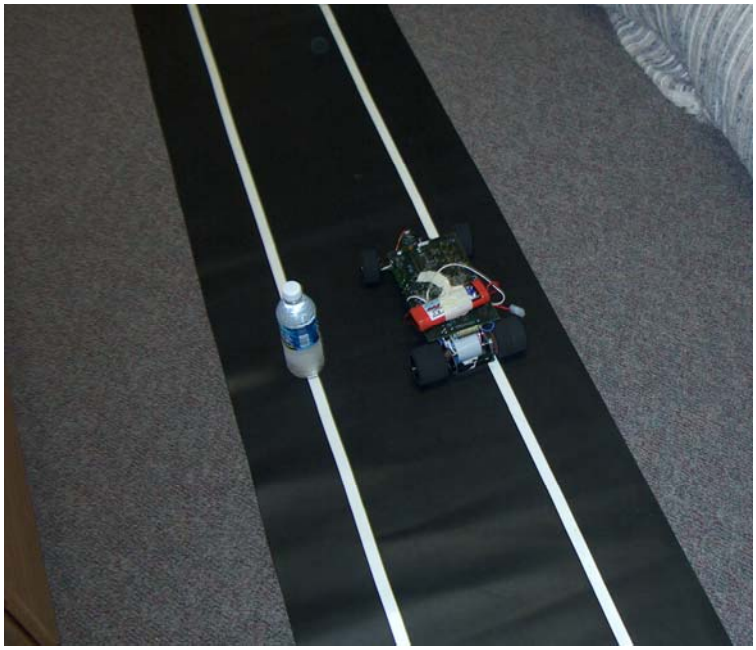


Figure 38 Car changed lanes to avoid an obstacle

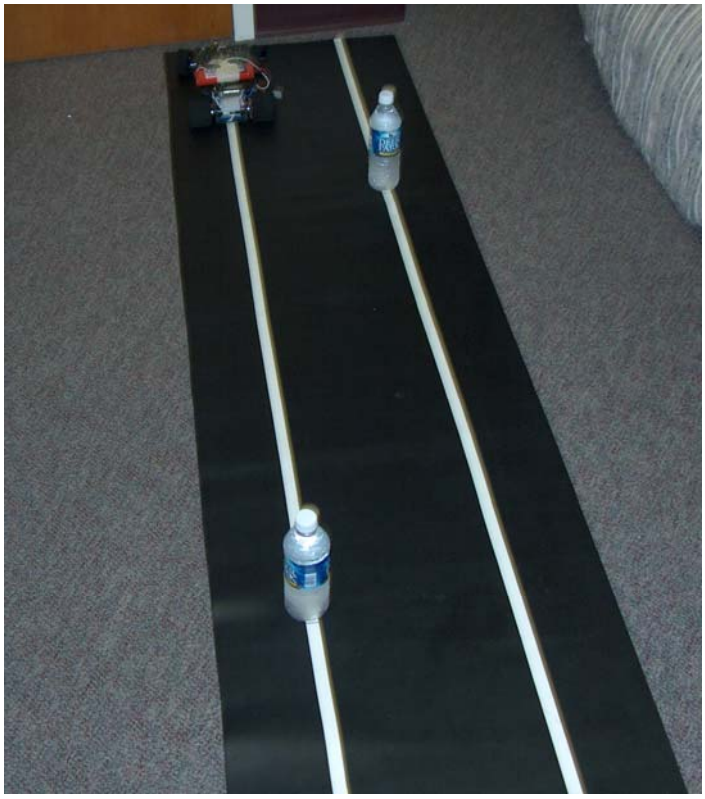


Figure 39 Car stopped after avoiding both obstacles

CHAPTER V

CONCLUSION

The problem that was looked into was collision avoidance with lane changing and lane keeping. The open loop lane change maneuver was derived from the principal of optimality and resulted in bang bang control of the car's steering. One of the proposed solutions was using an open loop lane change maneuver with optimal control as the decision base. Another was using Fuzzy controller as the decision base. The fuzzy controller was designed after the optimal controller. The desired final states from the optimal control were used in the design of the fuzzy controller. This was first worked out in a simulation using MATLAB/Simulink.

Working out all the bugs and problems in the simulation made it easy to implement in the prototype car. For the DSP code in the car the MATLAB m files just had to be converted line by line to c code. The MATLAB m files were written with similar function routines as the c code, so the converting was not very difficult.

The optimal controller was designed and tested first. As more functions were added to the optimal controller, it became more difficult to come up with the formulas to have the car behave correctly. When deciding to turn or not to turn the distances could be set exactly. The next controller that was worked on was the Fuzzy controller. Fuzzy programming was easier to use and manipulate and to describe the rules needed for obstacle avoidance. With the Fuzzy controller, it was easy to add more rules and functions. Although setting exactly when to turn or not turn was difficult, because of the fuzzy nature of the controller. Also, it is so flexible, and has so many adjustments that it takes time to setup optimally.

The curvature estimation formula included from [7], made the vehicle perform poorly when it was used in both the simulation and the model car. The reason was because of the discrete nature of the sensors which cause the term **Error! Objects cannot be created from editing field codes.** to generate noisy data. The car and the simulation

behaved significantly better with a higher gain and the curvature output going to the controller set to zero except when doing the lane change maneuver.

We are now at the forefront of designing Smarter/Safer vehicles, when it comes to avoiding obstacles. We have hybrid-powered cars now, maybe one day soon hybrid (Human/Computer) controlled steering vehicles will be on the roads.

CHAPTER VI

SUMMARY

This project has been very interesting and opened other possibilities for making cars safer. Optimal control seems like the one of the better methods when it comes to collision avoidance, another idea would be a neural system or a hybrid system (optimal control and neural system). When starting to implement a neural system it gets complex rather quickly. Trying to get a system that learns and grows with each experience would be ideal, it is just complex to create as well as difficult to verify. What would happen in a real vehicle if the learning system, decides that crashing is a "better" solution? After getting the simulation working properly and then implementing it on the model car and seeing it work well, it was very interesting to watch.

Future work on improving the software for collision avoidance system would include, creating an optimal lane change trajectory for the curves, designing a Neural / Fuzzy Hybrid Collision Avoidance with Human factor control system, and an auto tuning for the Fuzzy Controller. Adding feedback to the lane change trajectory would make the system more robust to compensate for differences in the real-world car and lane characteristics. Another future software improvement would be to incorporate a factor for the comfort of the ride, because with bang bang steering then everything would be shaking around in the car. There needs to be a comfortable ride controller added to the collision avoidance for normal collision avoidance. In an emergency, the bang bang controller might still be needed to assure collision avoidance. Changing the cost function so the vehicle will not change lanes into an obstacle next to it rather then relying on adaptive cruise control to avoid the collision is one last area for future software work.

Future work on improving the hardware for collision avoidance system would include adding ultrasonic sensors and the supporting hardware for the sensors. The ultrasonic sensors would be mounted to all four sides of the car so the vehicle will not change lanes into an obstacle next to it rather then relying on adaptive cruise control to avoid the collision. Adding the hardware to the car is more difficult given there is only one spare analog to digital input channel. Adding an additional analog to digital converters and DC-DC converters would require a new PIC printed circuit board to be designed and manufactured.

LITERATURE CITIED

[1] Balkcom, Devin J. and Mason, Matthew T., "Extremal Trajectories for Bounded Velocity Mobile Robots," ICRA, May, 2002.

[2] Benli, Ömer S., INTERNATIONAL JOURNAL OF INDUSTRIAL ENGINEERING. INDUSTRIAL ENGINEERING APPLICATIONS AND PRACTICE: USERS ENCYCLOPEDIA® http://benli.bcc.bilkent.edu.tr/~omer/research/dynprog.html Copyright 1999

[3] Howells, Christopher C., "Game-Theoretic Approach with Cost Manipulation to Vehicular Collision Avoidance," Thesis. Virginia Polytechnic Institute and State University, 2004.

[4] Jaguar, "Adaptive Cruise Control," http://www.jaguar.com/uk/en/_glossary/acc_2.htm

[5] Kachroo, Pushkin and Mellodge, Patricia, "Mobile Robotic Car Design," The McGraw-Hill Companies Inc., New York, Copyright 2005.

[6] Kirk, Donald E., "Optimal Control Theory An Introduction," Dover Publications, Mineola, New York, Copyright 1998.

[7] Mellodge, Patricia, "Feedback Control for a Path Following Robotic Car," Thesis. Virginia Polytechnic Institute and State University, 2002.

[8] Shedied, Samy A., "Optimal Control for a Two Player Dynamic Pursuit Evasion Game; The Herding Problem," Thesis. Virginia Polytechnic Institute and State University, 2002.

[9] "Traffic Safety Facts 2003," National Highway Traffic Safety Administration, Report #DOT HS 809 640.

[10] Gupta, Madan M., Jin, Liang, and Homma, Noriyasu. "Static and Dynamic Neural Networks: From Fundamentals to Advanced Theory," John Wiley & Sons, Inc. Hoboken, New Jersey. Copyright 2003

[11] Goodrich, Michael A. and Boer, Erwin R., "Designing Human-Centered Automation Tradeoffs in Collision Avoidance System Design" IEEE Transactions on Intelligent Transportation Systems, Vol. 1, No. 1, March 2000.

[12] Riid, Andri, Pahhomov, Dmitri and Rustern, Ennu, "Car Navigation and Collision Avoidance with Fuzzy Logic," Fuzzy Systems, 2004. Proceedings. 2004 IEEE International Conference on Volume 3, 25-29 July 2004 Page(s):1443 - 1448 vol.3

[13] Cheok , Ka C., Smid, G. E.and McCune, D.J., "A Multisensor-Based Collision Avoidance System With Application To A Military HMMWV," 2000 IEEE Intelligent Transportation Systems Conference Proceedings Dearborn (MI), USA Octoberl-3, 2000

[14] Lages, Ulrich, "Collision Avoidance System for fixed Obstacles - Fuzzy Controller Network for Robot Driving of an Autonomous Vehicle-," 2001 IEEE Intelligent Transportation Systems Conference Proceedings - Oakland (CA), USA - August 25-29, 2001

[15] Griffiths, P., Langer, D., Misener, J.A., Siegel, M., and Thorpe, C., "Sensor-friendly vehicle and roadway systems," Instrumentation and Measurement Technology Conference, 2001. IMTC 2001. Proceedings of the 18th IEEE Volume 2, 21-23 May 2001 Page(s):1036 - 1040 vol.2

[16] Runkle, Donald, Sensors Expo and Conference, http://www.delphi.com/pdf/media/DonRunkle_SensorsExpo_June8_2004.pdf

[17] Delphi, Safety Warning Systems http://www.delphi.com/products/auto/safety/warning/

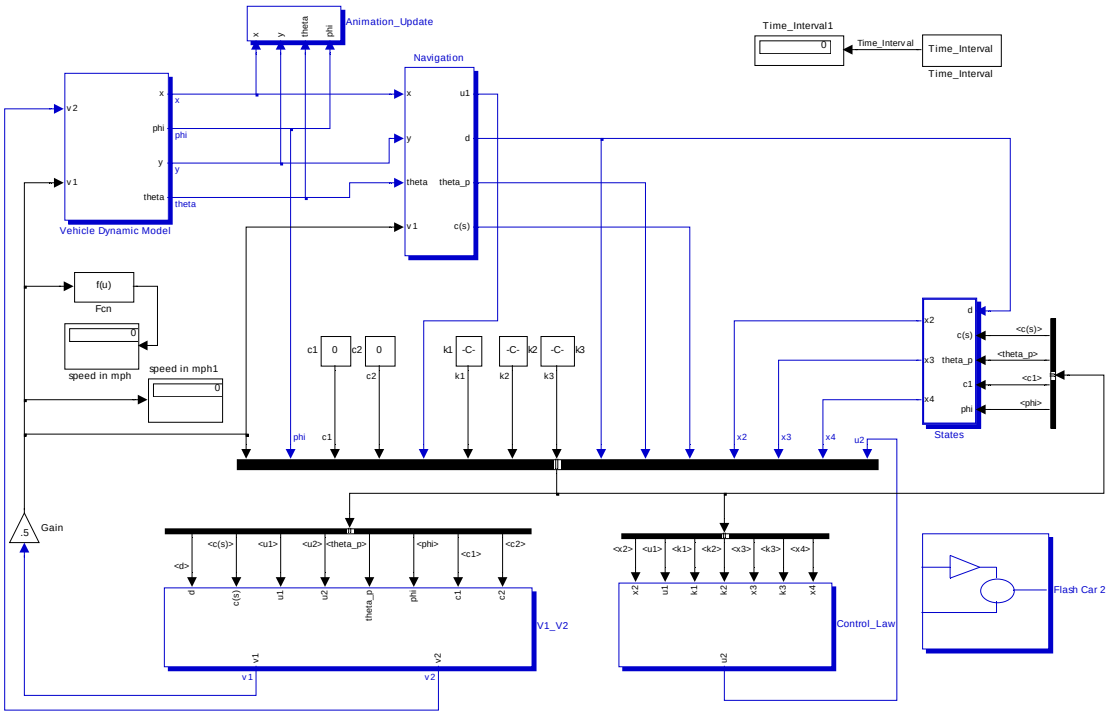
[18] Smart Car http://www.memagazine.org/backissues/mar03/features/smartcar/smartcar.html

[19] Autonomous Cross-Country Navigation http://www.frc.ri.cmu.edu/~axs/cross_country.html

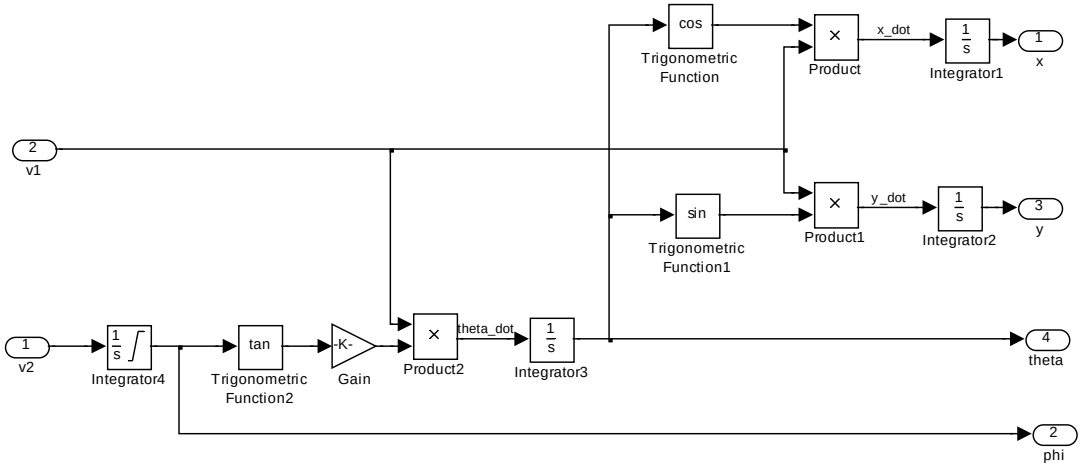
APPENDIX A SIMULINK

Overview of Car Simulation

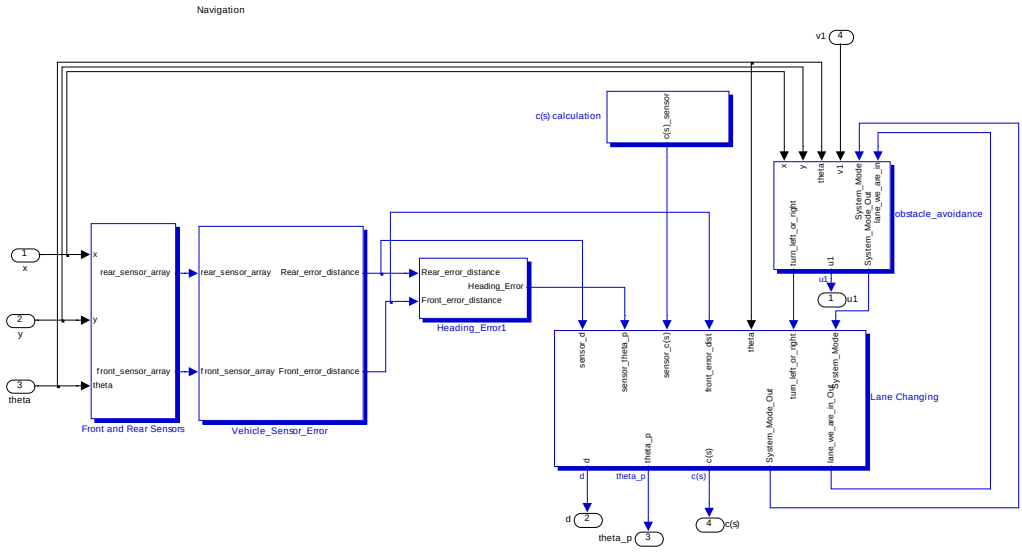
Overview of the Flash Car Simulation



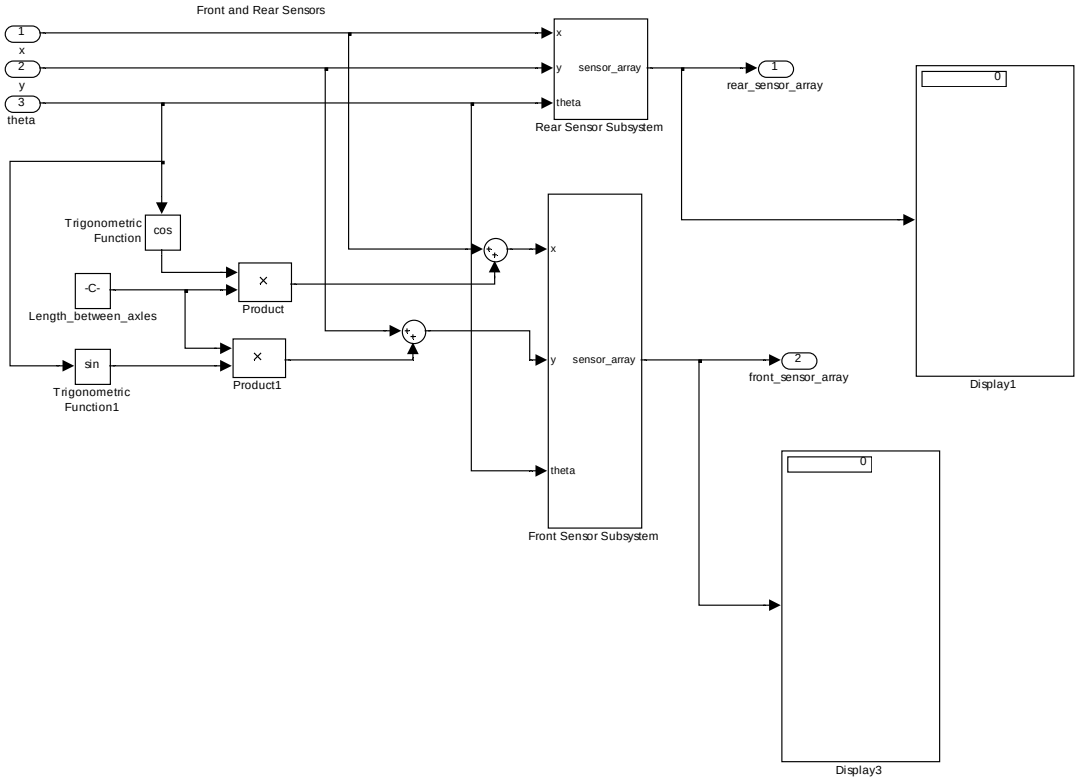
Car Dynamic Model



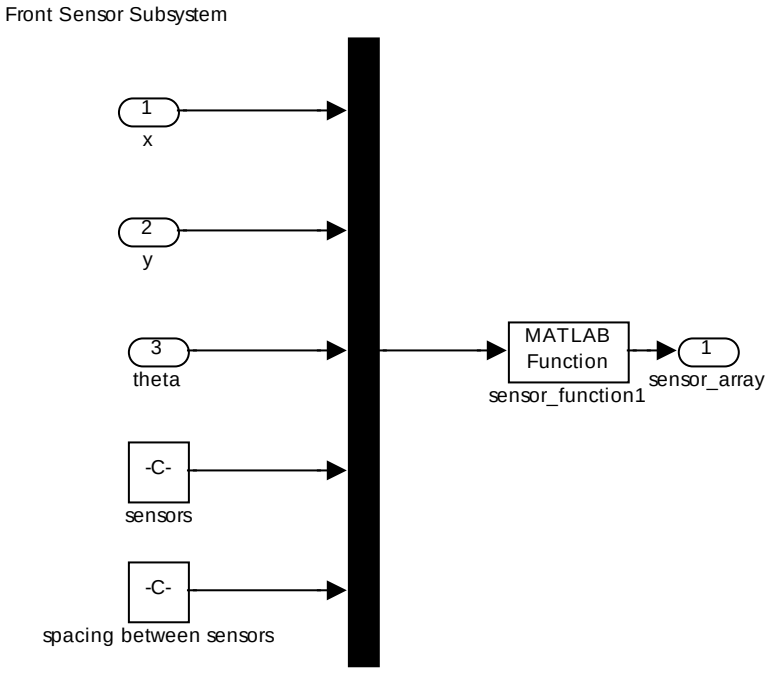
Overview of Navigation



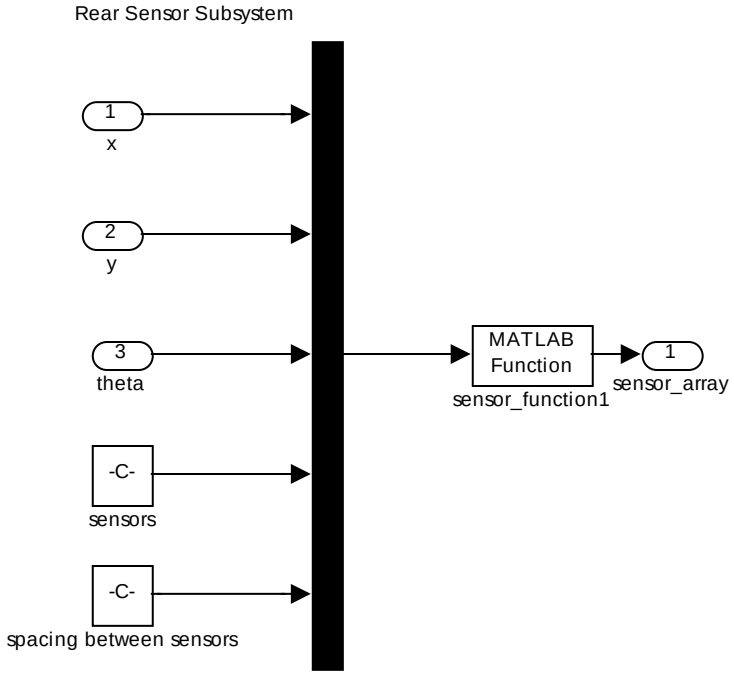
Overview of Front and Rear IR Sensors



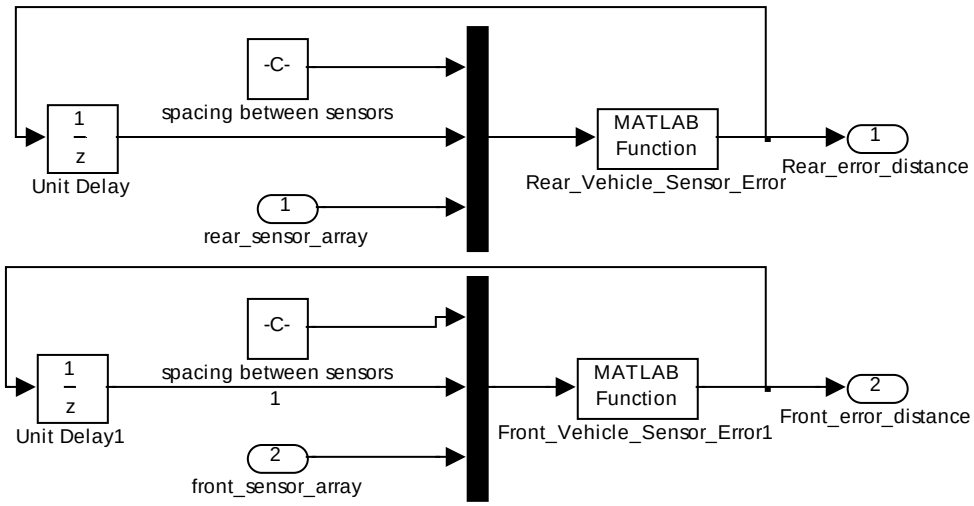
Front IR Sensor Subsystem



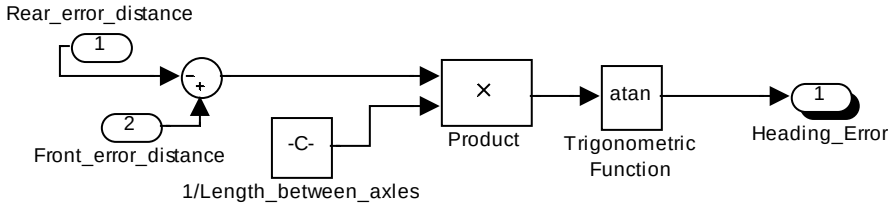
Rear IR Sensor Subsystem



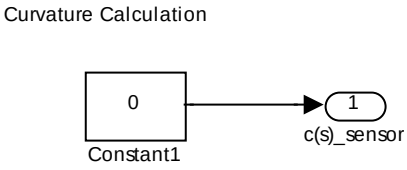
Car Sensor Error



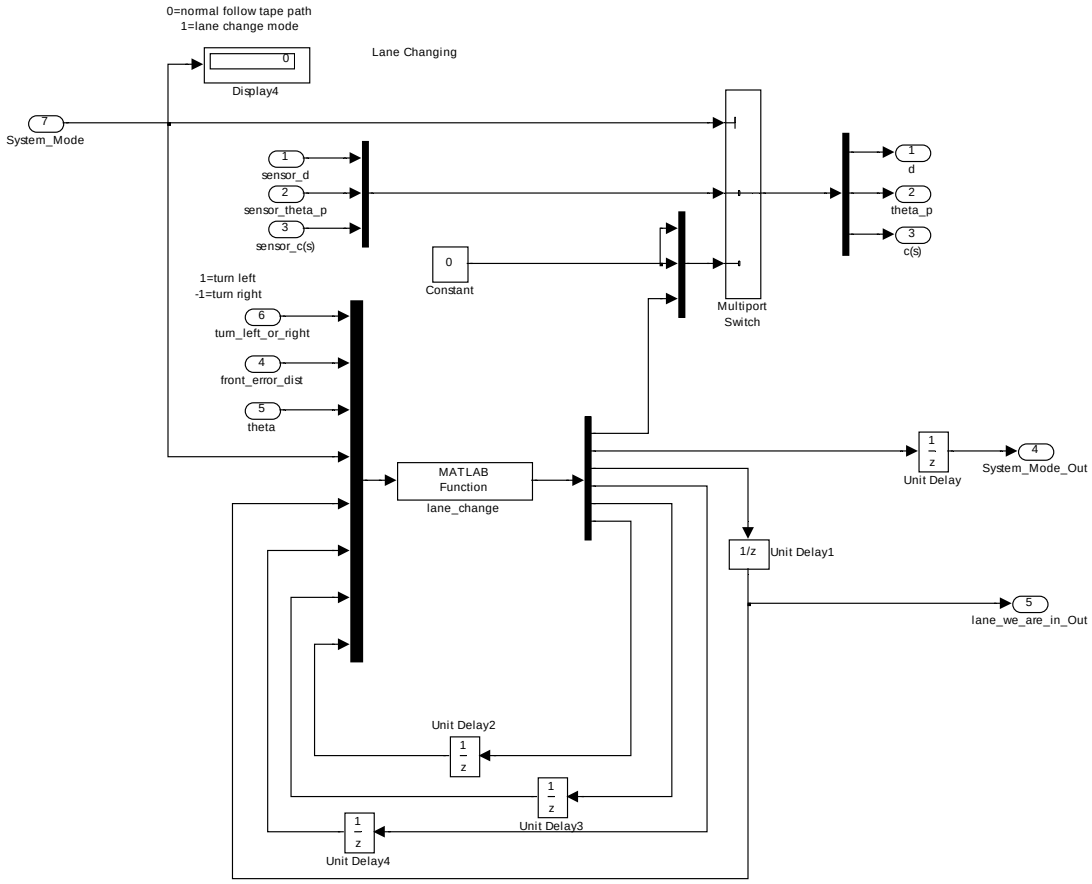
Heading Angle Calculation



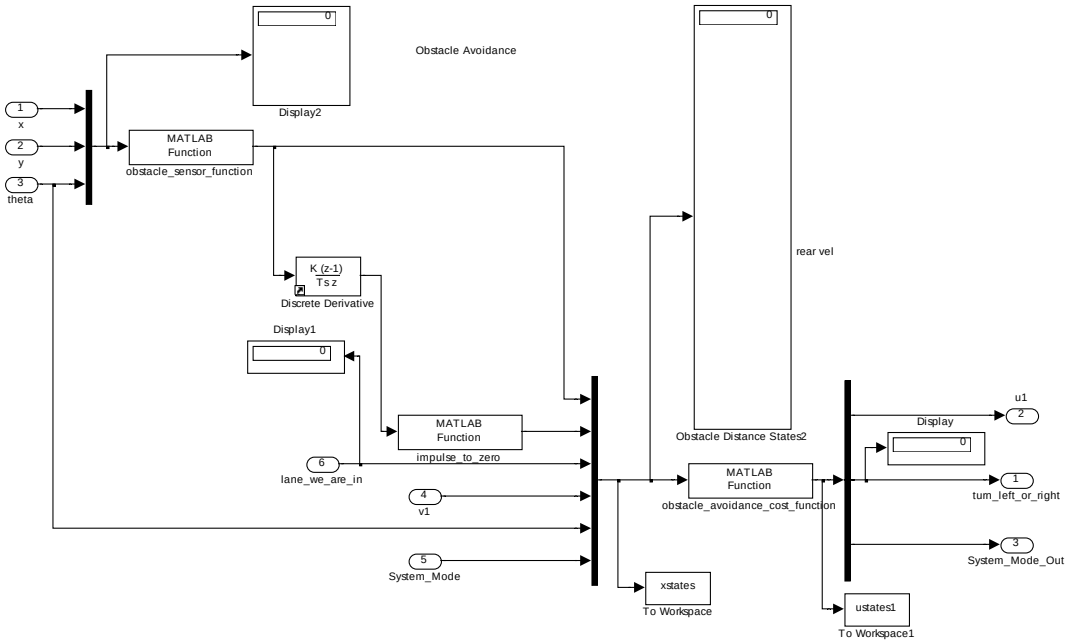
Curvature Calculation



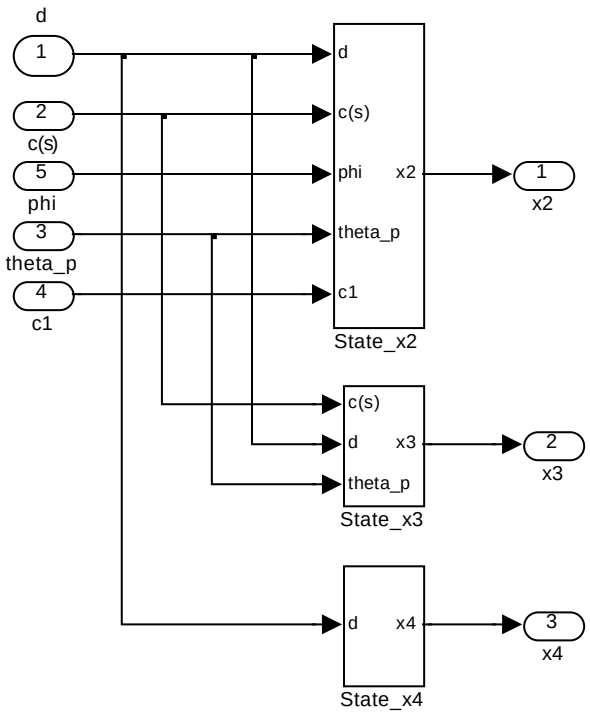
Lane Changing



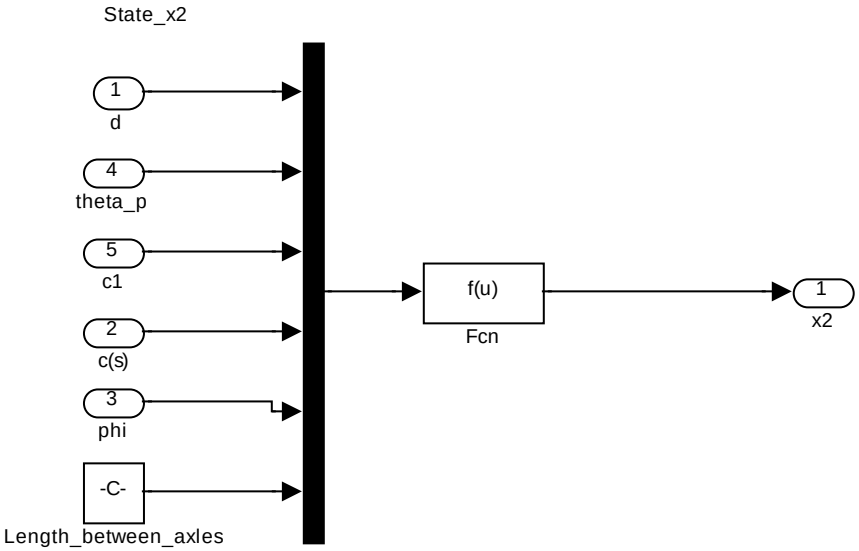
Obstacle Avoidance



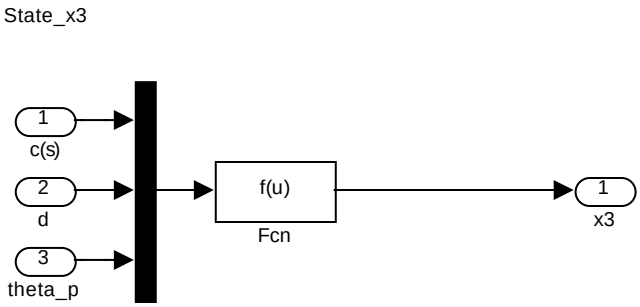
Overview of the States



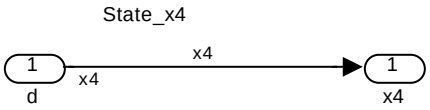
State $x_2(t)$



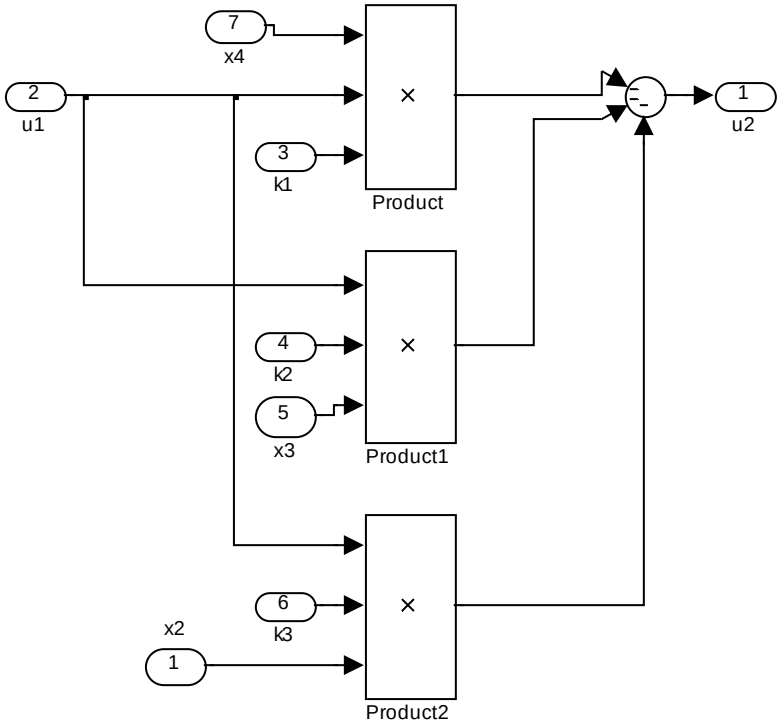
State $x_3(t)$



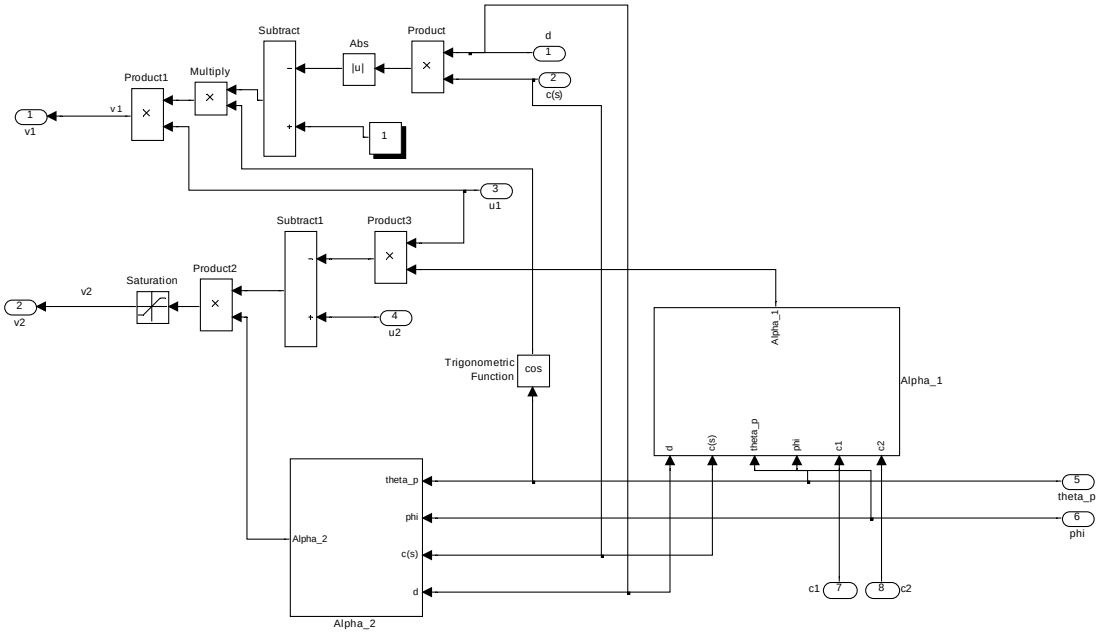
State $x_4(t)$



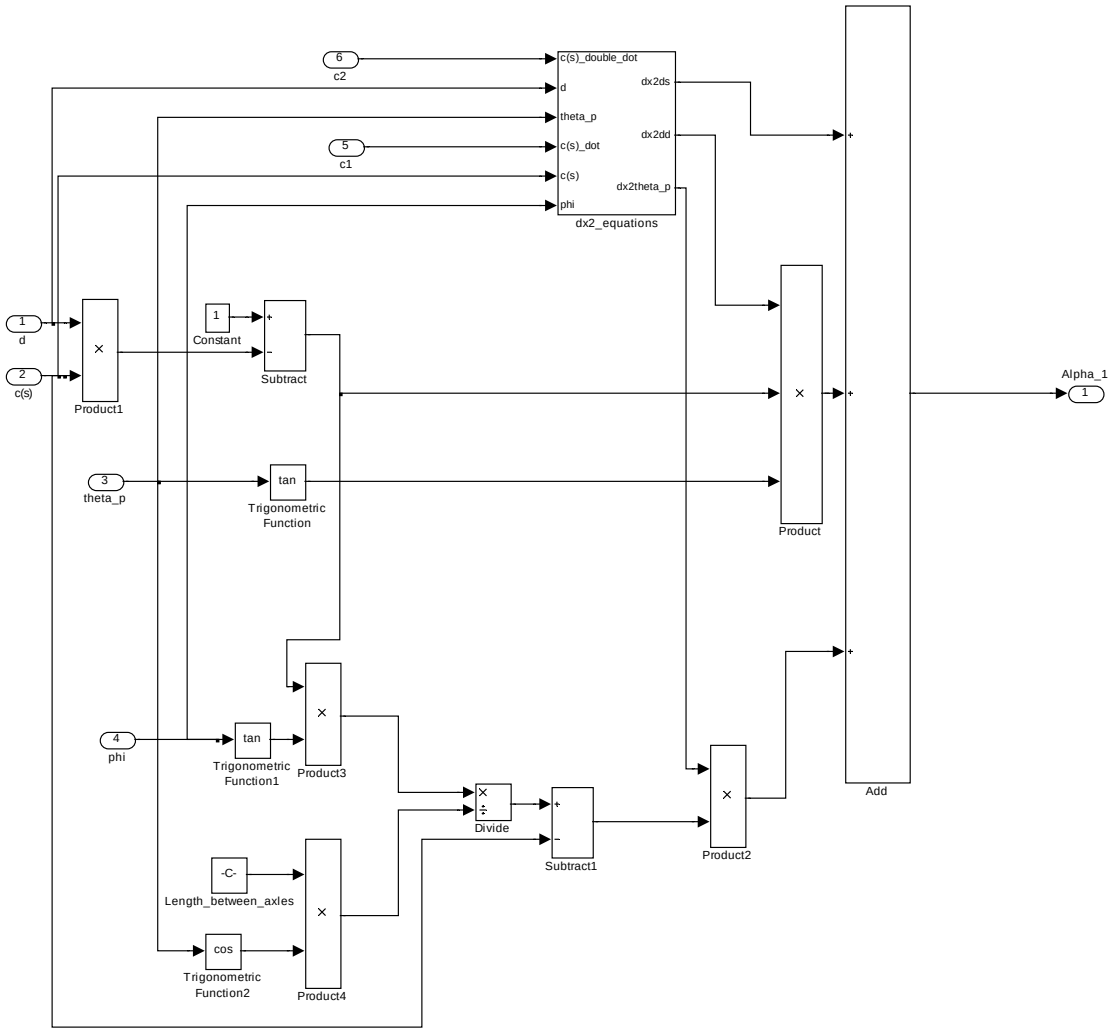
Control Law



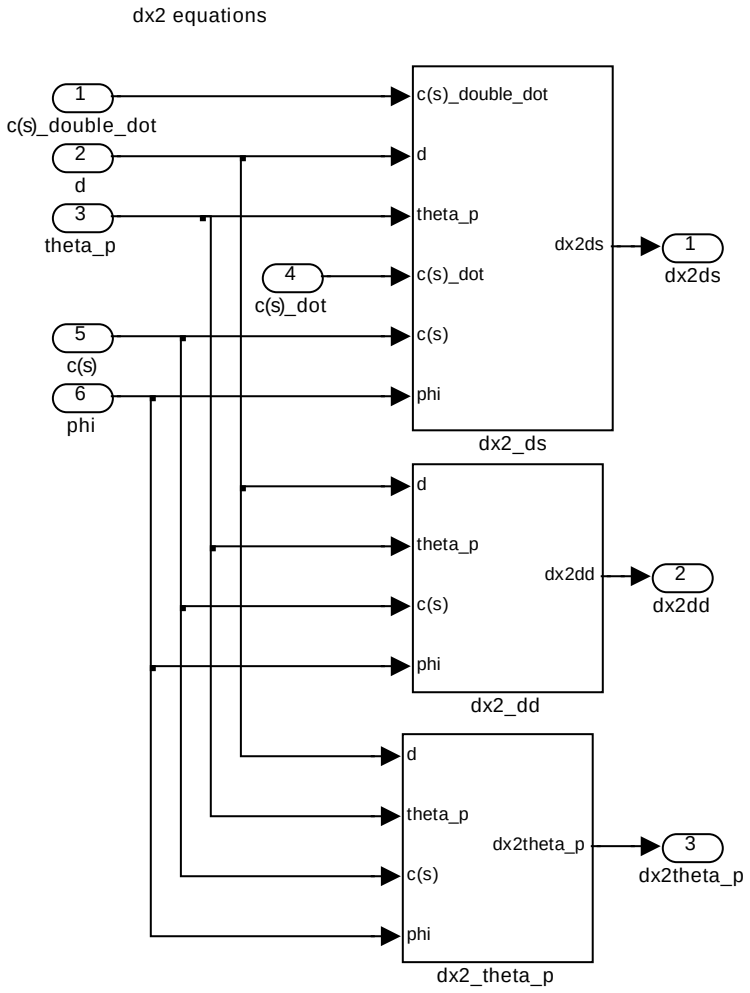
Overview of V1 and V2



Alpha 1

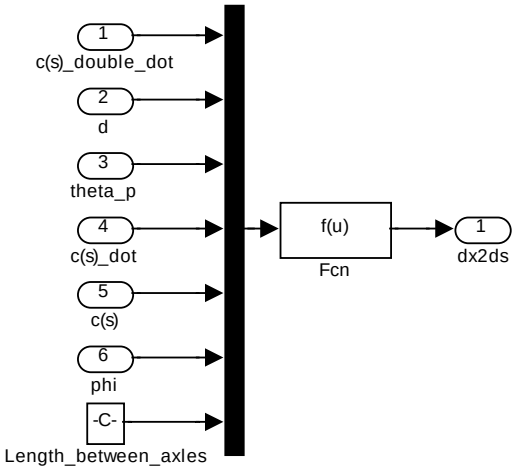

6
c2
5
c1
c(s)_double_dot
d
theta_p
c(s)_dot
c(s)
phi
dx2ds
dx2dd
dx2theta_p
dx2_equations
1
d
2
c(s)
Product1
1
Constant
Subtract
3
theta_p
tan
Trigonometric Function
Product
Alpha_1
1
4
phi
tan
Trigonometric Function1
Product3
Divide
Subtract1
Product2
Add
-C-
Length_between_axles
cos
Trigonometric Function2
Product4

dx2(t) Equations



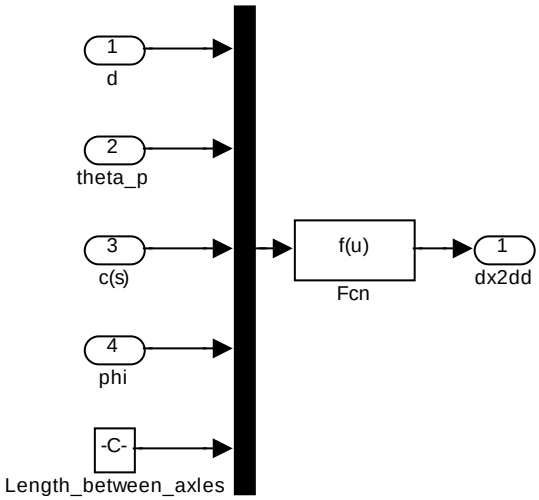
dx2(t)/ds Equations

dxds = -c2*d*tan(th)-(c1+2*d*c*c1)*((1+sin(th)*sin(th))/(cos(th)*cos(th)))-(2*(1-d*c)*d*c1*tan(phi0))/(Length_between_axles*(cos(th)^3));



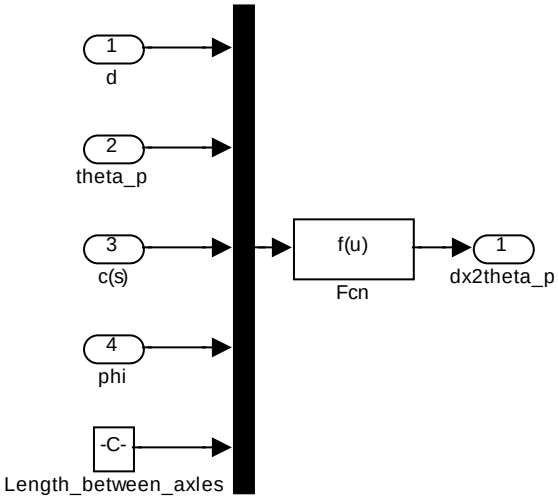
dx2(t)/dd Equations

dxdd = c*c*(1+sin(th)*sin(th))/(cos(th)*cos(th))-2*c*(1-d*c)*tan(phi0)/(Length_between_axles*(cos(th))^3);

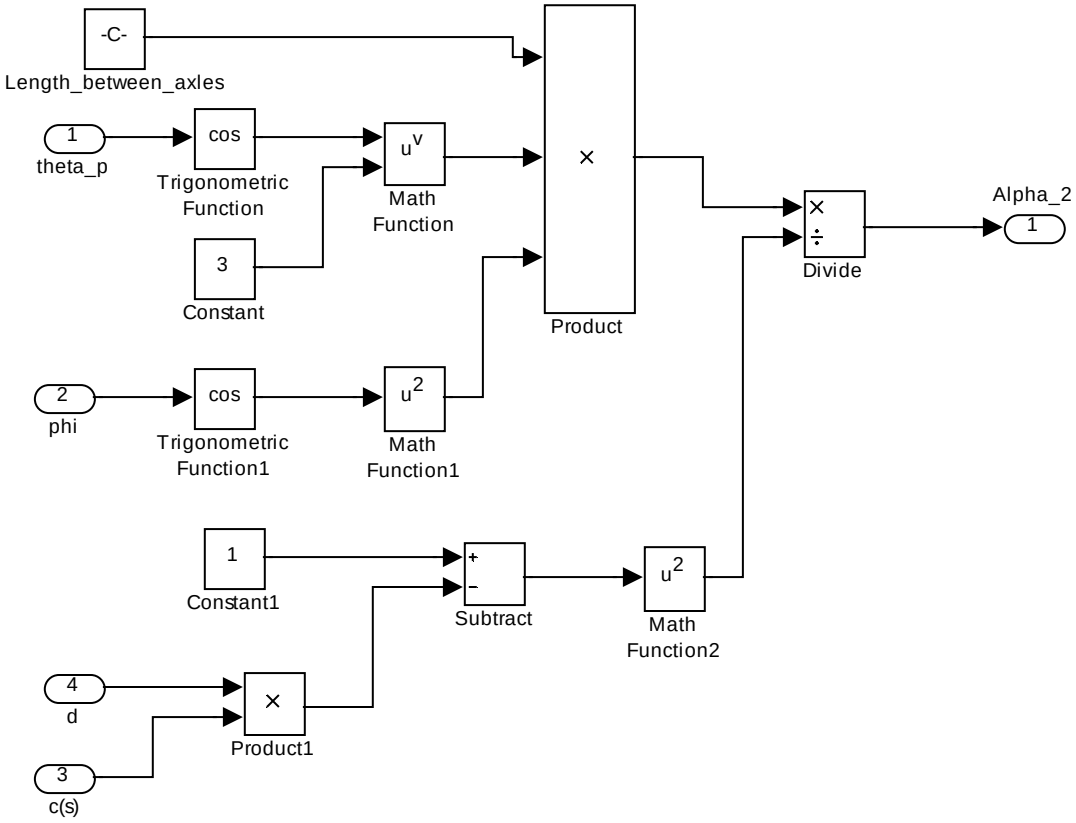


dx2(t)/dtheta_p Equations

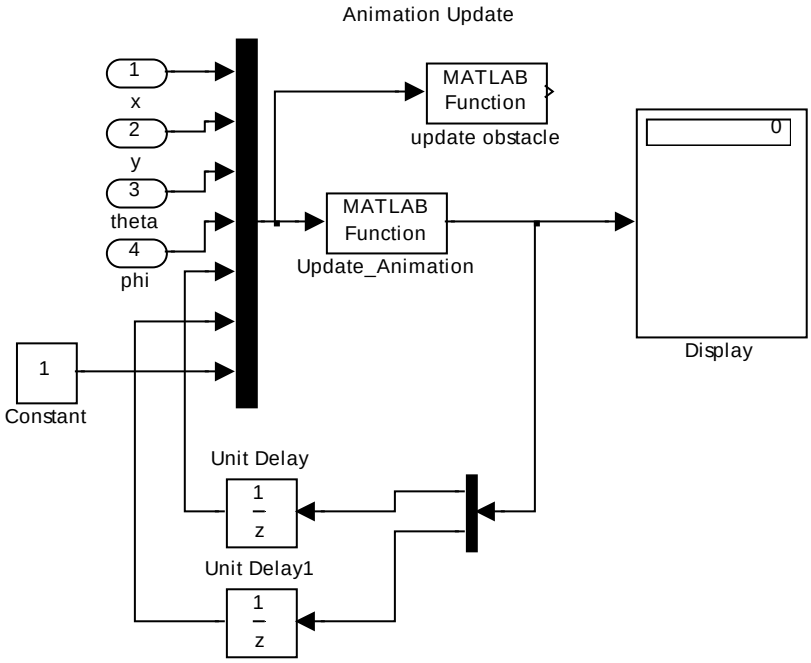
dxdtheta = -c*(1-d*c)*4*tan(th)/(cos(th))^2+3*(1-d*c)^2*tan(phi0)*tan(th)/(Length_between_axles*(cos(th))^3);



Alpha 2



Animation Update



Vita

William K. Grefe was born in Livingston, New Jersey and graduated from Roxbury High School in 1999. In 2003 he received a Bachelors degree in Electrical Engineering from Rensselaer Polytechnic Institute (RPI). After graduating from RPI he decided to continue onto the Masters Program at Virginia Tech. His research interests are Research & Design, Autonomous Vehicles, Obstacle Avoidance, & Automated Control Systems.

The motto he lives by is

"Why Not Change the World"