# Obstacle Avoidance of an Unmanned Ground Vehicle using a Combined Approach of Model Predictive Control and Proportional Navigation

by

Ryan Shaw

A thesis submitted to the Graduate Faculty of
Auburn University
in partial fulfillment of the
requirements for the Degree of
Master of Science

Auburn, Alabama
December 15, 2018

Keywords: Model Predictive Control, Proportional Navigation, Vehicle Dynamics

Approved by

## Abstract

This thesis presents a new approach for the guidance and control of a UGV (Unmanned Ground Vehicle). A special focus was placed on moving obstacles that interfere with the planned path of the vehicle, this is due to the fact that the majority of obstacle avoidance research has been completed on stationary objects. An obstacle avoidance algorithm was developed using an integrated system involving Proportional Navigation (Pro-Nav) and a Nonlinear Model Predictive Controller (NMPC). An obstacle avoidance variant of the ideal proportional navigation law generates command lateral accelerations to avoid obstacles, while the NMPC is used to track the reference trajectory given by the Pro-Nav. The NMPC utilizes a lateral vehicle dynamic model along with a nonlinear tire model in order to issue control inputs. In this application an obstacle avoidance algorithm can take over the control of a vehicle until the obstacle is no longer a threat. Another application of a Pro-Nav and NMPC algorithm was tested for leader/follower situations. The performance of the leader/follower and obstacle avoidance algorithm is evaluated through different simulations.

Simulation of the performance of the PNCAG and NMPC algorithm was conducted using two different simulation environments; MATLAB and Simulink Vehicle Dynamics Blockset. The MATLAB simulation validated the algorithm showing that it could be used to accomplish obstacle avoidance. With the algorithm shown to be effective, it was placed into the Vehicle Dynamics Blockset. The Vehicle Dynamics Blockset provided a higher fidelity vehicle model to provide a more realistic simulation environment. In addition to obstacle avoidance, simulation results verified the performance of a modified version of the PNCAG and NMPC algorithm in a leader/follower scenario. The results show, the algorithm handled the leader/follower and collision avoidance with reasonable error. Overall the algorithm was also able to follow a lead vehicle throughout a double lane change as well as avoid collision with a moving obstacle in four different scenarios.

## Acknowledgments

I would like to thank my parents for their support through my entire education, from helping with school projects in elementary school all the way through helping proofread this thesis. Thank you for your support through all the years of school, especially throughout my master's program. My father has always shown me how to work hard and how to make sure I keep everything in perspective. My mother has always helped me overcome obstacles and deal with all the challenges that are a part of life.

Also I would like to thank Joe Selikoff and Nick Williams. We all started in the GAVLAB at the same time, and have all been on the same research project throughout our graduate school careers. Thank you for being great guys to work with.

There are too many people throughout my life that have played a part in getting me to where I am today. Everybody who helped with my undergraduate education at Clemson University, thank you. Special thanks to David Bell and Michael Sprunk, who are also in the GAVLAB, you both have been good friends and I appreciate all the help and support. To all the members of GAVLAB, thank you for making my graduate school experience what it was. Thank you to the professors at Auburn University and Clemson University for the knowledge that has allowed me to succeeded in mechanical engineering.

Lastly, I would like to thank Dr. Bevly for giving me the opportunity to work in his lab as a research assistant. After visiting other graduate schools, Dr. Belvy was the only professor that made me feel comfortable and confident in my decision to purse my masters degree in mechanical engineering. Thank you for taking a chance on a Clemson Graduate, and helping me to live up to my full potential. I feel very privileged to have worked in the lab at Auburn University with such knowledgeable people.

Table of Contents

List of Figures

List of Tables

# Chapter 1
# Introduction and Background

Vehicle automation is quickly expanding as technology continues to advance and push the boundaries of what is possible. There are six levels of automation as defined by SAE International, an automotive standardization body. The levels range from No Automation, to Partial Automation, and eventually to Full Automation [1]. No Automation is a level defined by the fact the driver performs all the driving tasks. Partial Automation is a state of automation where the vehicle has automated functions such as acceleration and steering, but the driver must remain engaged with the driving task and monitor the environment at all times. Full Automation involves a vehicle capable of performing all functions under all conditions, where the driver may have the option to control the vehicle. The algorithm proposed in this thesis was developed for integration within a complete control system for autonomous ground vehicles at the level of Partial Automation to Full Automation. Unmanned ground vehicles (UGVs) are widely used in the military at present for dangerous missions where a human driver would not be practical [2]. UGVs are the land-based counterpart to unmanned aerial vehicles, on which a large amount of the research presented in this thesis was based. The goal of this thesis is to develop a combined Proportional Navigation and Nonlinear Model Predictive Control algorithm that is able to effectively avoid moving obstacles in the UGV's path of travel.

Many scenarios require a UGV to avoid a moving obstacle as well stationary objects. Stationary object avoidance has been well researched with successful solutions, however this thesis focuses on moving obstacles. Collision avoidance has been commonly defined as a series of practices designed to prevent vehicles such as cars, trains, ships, and airplanes from colliding with obstacles. Longitudinal collision avoidance systems have already been implemented in newer cars on the road today. These systems use RADAR (Radio Detection and Ranging), LIDAR (Light Detection and Ranging), and image recognition in order to detect a possible collision scenario [3]. Collision avoidance is typically accomplished by braking the car once an obstacle is detected in the vehicle's path. At low vehicle speeds (below 20 mph) this method works well, however at higher speeds active steering is a more appropriate and successful collision avoidance technique.

An additional application of the Pro-Nav and NMPC used together is a leader/follower scenario. Truck platooning is a heavily pursued research area. Truck platooning is defined as a group of semi-trucks that are digitally connected in order to save fuel by driving closely with one vehicle following the other, thus reducing drag on the follower vehicle. In order to accomplish this, all the trucks must stay in line laterally as well as follow at a close distance in order for the draft effect to be utilized for fuel savings [4]-[5].

To handle the above mentioned scenarios, a novel approach is proposed which combines Proportional Navigation to provide a navigation command, and Nonlinear Model Predictive Control to implement the command and control the lateral position of the vehicle. Applications to avoid moving obstacles and for leader/follower truck platooning scenarios have been completed and analyzed.

## 1.1 Overview of Model Predictive Control

Model Predictive Control (MPC) is a leading method of process control, created in the 1980s, that was originally developed for use in chemical plants and oil refineries [6]. At the time of development, it was computationally expensive to run MPC on high dynamic systems. Thus, uses of MPC were not very expansive. Today, computers have become more powerful, allowing MPC to run in real time and handle complex systems, hence making it a popular control technique.

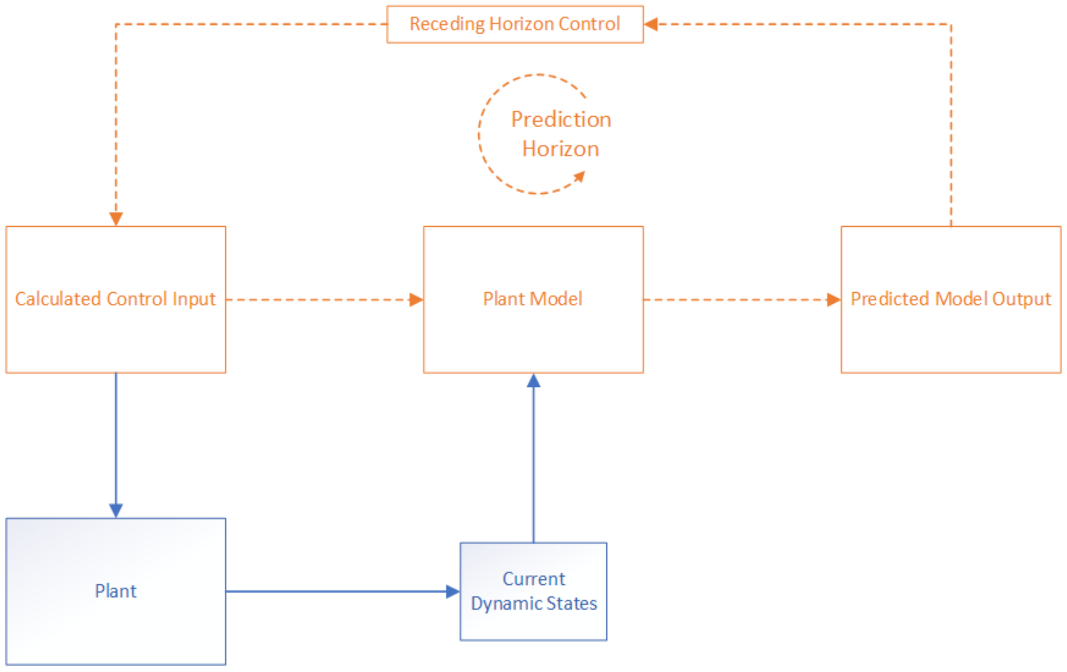

Figure 1.1: Basic MPC Diagram

Figure 1.1 shows a block diagram of the MPC control structure. MPC uses a dynamic model to produce a set of control inputs over a finite time horizon. The controller only applies the first control input, then optimizes again and the process continues to repeat.

Due to the finite horizon aspect of MPC, stability is not guaranteed, and tuning weights and horizons are the only way to achieve stability especially for the nonlinear MPC variant used in this thesis. Additionally, the disadvantages of MPC include the difficulty of obtaining a high-accuracy model, along with the computational expense of solving the objective function [7]. Advantages of the MPC include the ability to include hard and soft constraints on both the inputs and output states, as well as handling the nonlinear plant dynamics. In general, MPC has lower rise time, settling time, and overshoots when compared to Proportional, Integral, and Derivative (PID) control, thus providing smoother operation.

### 1.1.1 Model Predictive Control for Autonomous Ground Vehicles

MPC has been applied to both longitudinal and lateral dynamics for vehicle control. MPC in general has many advantages over other controllers, such as the ability to incorporate prediction of future events and to implement soft and hard constraints on the system. In [8] and [9], a vehicle bicycle model is used in conjunction with a nonlinear Pacejka tire model for the development of an autonomous steering system based on Nonlinear Model Predictive Control (NMPC). The nonlinearities of a real tire are captured well with the Pacejka model and better estimate the performance when they become saturated at high levels of lateral acceleration, as is discussed in the derivation of the Vehicle Model [10]. However, due to the nonlinear tire model, NMPC was used instead of linear MPC. NMPC has the advantage of being a controller that inherently captures nonlinearities within a dynamic system. NMPC allows for the stability boundary to be increased when compared with similar linear controllers.

Multiple papers were analyzed involving control of steering for autonomous vehicles. There are several methods that use either a MPC or NMPC algorithm based approach utilizing a vehicle model, these methods helped to develop the NMPC used in this thesis. The main goal of the NMPC based steering system in this thesis is to control a vehicle using reference lateral velocities, while considering physical constraints on the system.

## 1.2 Overview of Proportional Navigation

Proportional Navigation (Pro-Nav) is a guidance law that has shown to be successful in surface-to-air and air-to-air missile tracking systems [11]. The guidance law was developed by the United States during World War II. The law has an advantage of being relatively simplistic when compared to other guidance schemes [12]. It is important to note that Proportional Navigation is intended to ensure collision between two objects. In order to properly understand the guidance law, Figure 1.2 illustrates an object-target engagement scenario.

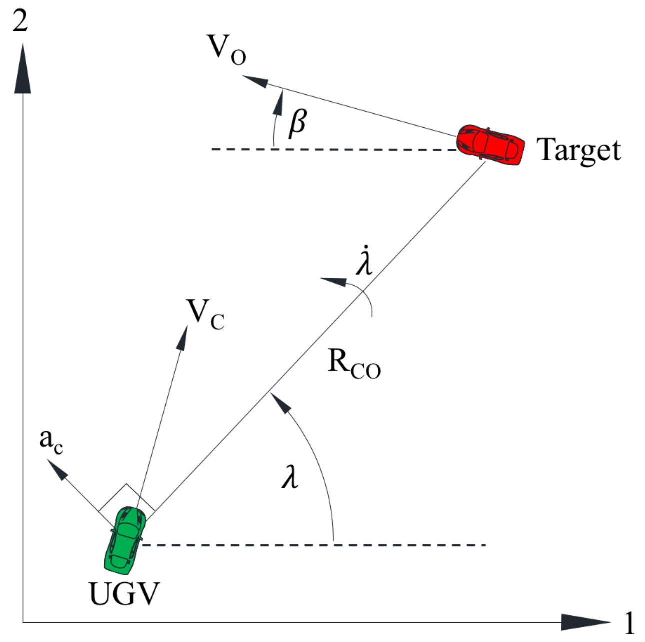

Figure 1.2: Pro-Nav Engagement Geometry

In Figure 1.2, $\lambda$ and $\dot{\lambda}$ represent the Line of Sight (LOS) and LOS rate, $a_c$ is the command acceleration, $V_C$ and $V_O$ are the velocities of the user and obstacle, and $R_{CO}$ is the range between the user car and target. The Line of Sight is an imaginary line between user and the target. The guidance law is based on the Constant Bearing Decreasing Range (CBDR) principle. CBDR states that when two objects are headed in constant directions with no change in the LOS angle, the objects will collide [13]. Pro-Nav dictates the user velocity vector must rotate at a rate proportional to that of the LOS, and in the same direction.

For Pro-Nav to be effective, full navigation states of the user and obstacle/target must be known. Many challenges are associated with a Pro-Nav based guidance system as it is classically implemented on a missile homing guidance system. In order for the system to provide the desired performance; the root-mean-square final distance from all noise sources need to be minimized, the guidance system must be able to maintain stability, and the system nonlinearities need to be minimized. For the application of Pro-Nav within the scope of this thesis, nonlinearities are accounted for in the controller and the stability of the Pro-Nav is assumed.

### 1.2.1 Proportional Navigation Types and Applications

There are various types of Proportional Navigation algorithms that have been developed [14]. Pure Proportional Navigation (PPN) follows the most basic definition of the Proportional Navigation law. The basic law is implemented using the following equation

$$a_c = NV_{UGV}\dot{\theta} \tag{1.1}$$

where, $a_c$ is the command acceleration, $N$ is the unitless navigation constant, $V_{UGV}$ is the user vehicle velocity, and $\dot{\theta}$ is LOS rate of change.

True Proportional Navigation (TPN) improves upon PPN by using the closing velocity instead of the user velocity. This is done because it is actually the closing velocity that drives the LOS rate of change to zero. The improved law is shown below,

$$a_c = N'V_c\dot{\theta} \tag{1.2}$$

where $N'$ is the effective navigation ratio, and $V_c$ is the closing velocity between the user and target. The main difference between the two laws is that the TPN law will be applied perpendicular to the LOS and not perpendicular to the velocity of the user.

Generalized TPN allows the lateral acceleration angle to be deviated by some angle in relation to the LOS. The last major development of basic Proportional Navigation (Pro-Nav) is Ideal Proportional Navigation(IPN). IPN implements the commanded lateral acceleration perpendicular to the relative velocity between the user and target. This specific law is what was used in the Leader/Follower example and Obstacle Avoidance variant of the Pro-Nav law described later in this thesis.

## 1.3 Prior Obstacle Avoidance Research

Obstacle avoidance for vehicles is used in a variety of situations and applications, where a possible threat (obstacle) could move into the planned path of a vehicle. Various techniques have been used for obstacle avoidance, most utilized is dynamic path planning. Within path planning there are several categories including grid-based, potential field, and optimization through MPC [15]-[16]. Most of the research within obstacle avoidance has been focused on land based mobile robots, however there is some research on UAVs and passenger vehicles. Most of these obstacle avoidance algorithms are required to run continually. In the case of the algorithm developed in this thesis, the algorithm would only take over if there is a detected obstacle. Brief descriptions of the previous work done is presented below.

### 1.3.1 Grid-Based Algorithms

The grid-based method to path planning overlays a grid over the environment the user is navigating. When the cell is located over an obstacle, the cell is set to a “False” value to represent the presence of an obstacle and area where the user vehicle can’t go. Figure 1.3 shows the grid-based path planning algorithm [17].

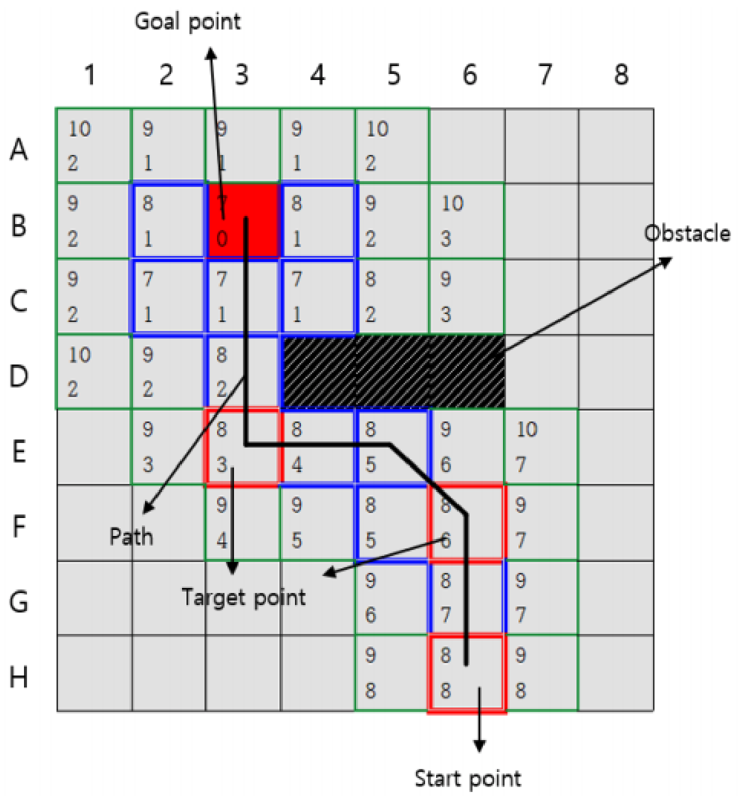

Figure 1.3: Grid-Based Schematic

Optimal search algorithms are used to find the global path, one that is free of obstacles, to connect the user to the goal position [18]. In general, grid-based approaches are much more suited to land-based mobile robots because of their good performance in low speed applications. They also have not been adapted well to moving obstacle scenarios, especially obstacles moving at higher speeds.

### 1.3.2 Potential Field Algorithm

The potential field algorithm views obstacle avoidance as a sub-task of a path planner within mobile robotics. The algorithm assumes that a ground vehicle is navigated by virtual forces that attract it towards a goal or repel it away from potential obstacles [19]. The path is then determined as a result of the virtual forces. The Virtual Force Field (VFF) method creates a virtual grid (similar to grid-based), then each occupied grid in the cell exerts a repulsive force driving the robot away from the obstacle as shown in Eq. (1.3),

$$\mathbf{F}(i,j) = \frac{F_{cr}C(i,j))}{d^2(i,j)}\left[\frac{x_i - x_0}{d(i,j)}\widehat{x} + \frac{y_j - y_0}{d(i,j)}\widehat{y}\right] \tag{1.3}$$

where $F_{cr}$ is the force constant, $d(i,j)$ is the distance between the cell and robot, $C(i,j)$ is the certainty level of the cell, $x_o, y_o$ are the robot coordinates, and $x_i, y_j$ are the coordinates of the cell [20]. Though the algorithm works reasonably well in certain situations, it struggles with confined land-based constraints (i.e tight spaces), has issues with real time implementation, and it inherently treats dynamic obstacles as static objects. Another disadvantage of the potential field method is the risk of getting stuck in a local minima, meaning an appropriate path may not be found.

### 1.3.3 Model Predictive Control Obstacle Avoidance

Model Predictive Control algorithms have typically fallen under the category of discrete optimization. MPC has also been used as an obstacle avoidance technique [21]. In these cases, MPC is used to calculate a trajectory around the obstacles, while a lower level controller is used to track the trajectory in both longitudinal and lateral vehicle directions. In the case of Park [22], longitudinal control was accomplished using a Proportion Integral Derivative controller (PID), while the lateral dynamics were controlled using a Linear Quadratic Regulator (LQR).

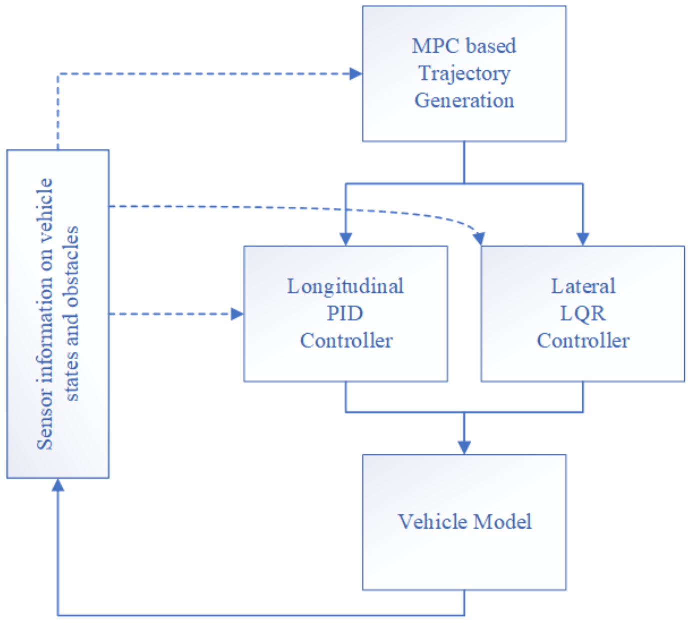
MPC based Trajectory Generation
Sensor information on vehicle states and obstacles
Longitudinal PID Controller
Lateral LQR Controller
Vehicle Model

Figure 1.4: Typical MPC Obstacle Control Structure

The above diagram Figure 1.4 shows the layout of a common MPC approach to obstacle avoidance. The approach works to avoid obstacles by implementing a cost term to reflect the obstacle threat. This MPC approach has a couple of disadvantages. The MPC must always be running and recalculating multiple trajectories before selecting the correct one, and it is not straightforward to account for moving obstacles within this algorithm, meaning there is no way to take current velocity and direction of an obstacle to better select a trajectory.

## 1.4 Prior Proportion Navigation and Model Predictive Control Combination

In [23], the combination of the Pro-Nav law and MPC was used on an Autonomous Underwater Vehicle (AUV) to track underwater cables and pipelines. It was shown that a PN law can be used to track a target objective and the MPC can be used to generate control commands to keep the vehicle following the reference trajectory.

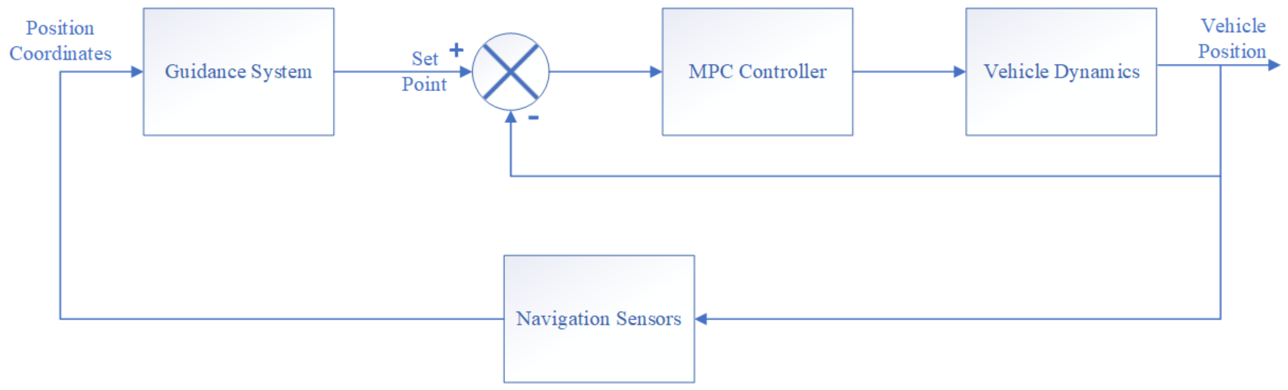

Figure 1.5: Navigation, Guidance, and Control Structure

Figure 1.5, shows the design structure used in the AUV paper. The same design structure was used as general basis for this thesis. Another implementation of a Line of Sight, not to be confused with Pro-Nav, and MPC-based approach was used in [24]. In [24], a decision-making model using a collision cone algorithm, similar to the type that will be described later in this thesis, and MPC for lane changing is shown to be successful. Thus, the main scientific hypothesis for the algorithm presented in this thesis was shown to be true. However, the algorithm from [24] differs from the one developed in this thesis in that Pro-Nav was not used, the algorithm was only applied to passing scenarios, and Nonlinear MPC was not used.

## 1.5 Contributions

This thesis presents an obstacle avoidance Pro-Nav and NMPC algorithm capable of safely navigating a UGV around a moving obstacle in various user vehicle and obstacle engagement scenarios. The obstacle avoidance algorithm calculates a desired command acceleration to avoid the obstacle if the obstacle is considered a threat. This command acceleration gets transformed into a command velocity in the correct reference frame through rotation and integration. The command velocity is then used as a reference for the NMPC controller. Advantages over other obstacle avoidance algorithms include the implementation of a completely independent obstacle avoidance algorithm and the ability for the NMPC to run when the vehicle is making high dynamic maneuvers. This allows the algorithm to take control of the vehicle from a driver or from a full automation control system. The proposed algorithm uses basic conditional statements and is simple in comparison to other current obstacle avoidance methods. Simulations of the Pro-Nav and NMPC algorithm to be used for obstacle avoidance as well as a leader/follower application are provided. Simulation of the performance are analyzed to validate the usefulness of the proposed architecture. In summary, the thesis provides the following contributions to the field:

- Development of an obstacle avoidance algorithm for a UGV to avoid a moving obstacle
- NMPC algorithm featuring constraints on vehicle steer angle as well as the option to place hard constraints on virtual lane boundaries
- Pro-Nav and NMPC algorithm applied to a Leader/Follower example that is applicable to convoying research
- Simulation results that validate the obstacle avoidance algorithm

# Chapter 2

# Vehicle Model

This chapter presents the vehicle model used for the control of the system, as well as MATLAB simulation. The most simplified model used in vehicle dynamics is a two degree of freedom bicycle model, representing the lateral and yaw motions. In this work, the Nonlinear Model Predictive Control (NMPC) algorithm uses the lateral dynamic vehicle model with a nonlinear tire as the predication model.

## 2.1 Vehicle Dynamic Model

The vehicle bicycle model is one of the most commonly used models for the control algorithms of land vehicles. The two common variations of the bicycle model are the kinematic model and the lateral dynamic model [25]. The later model is used in this work, however the kinematic model could be used at lower speeds.

### 2.1.1 Kinematic Bicycle Model

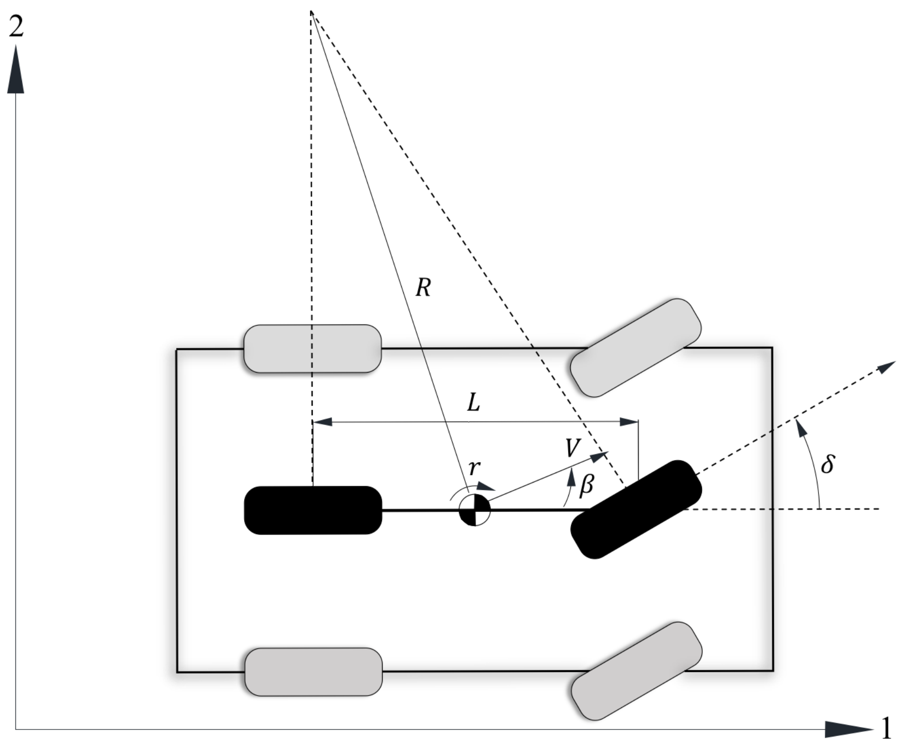

Figure 2.1: Kinematic Bicycle Model

The governing equations in the global frame for the kinematic bicycle model shown in Figure 2.1 are given below,

$$\dot{X} = V \ \cos(\psi + \beta) \tag{2.1}$$

$$\dot{Y} = V \ \sin(\psi + \beta) \tag{2.2}$$

$$\dot{\psi} = \frac{V}{l_r} \sin(\beta) \tag{2.3}$$

$$\dot{\beta} = \tan^{-1}\left(\frac{l_r}{l_f + l_r} \tan(\delta)\right) \tag{2.4}$$

where $X$ and $Y$ are the coordinates of the center of mass in the global frame $(X, Y)$. Also, $\psi$ is the heading angle, $V$ is the speed of the vehicle, $l_f$ and $l_r$ is the distance from the center of mass to the front and rear tires, and $\beta$ is the angle of the current velocity direction with respect to the longitudinal axis of the car, also known as the slide slip angle. This model assumes that slip angles at both wheels are zero. This assumption is good at relatively low speeds (less than 5 m/s), however since the desired user car speed in this thesis will be faster than 5 m/s, a model more suited to higher velocities needed to be developed.

### 2.1.2 Lateral Bicycle Model

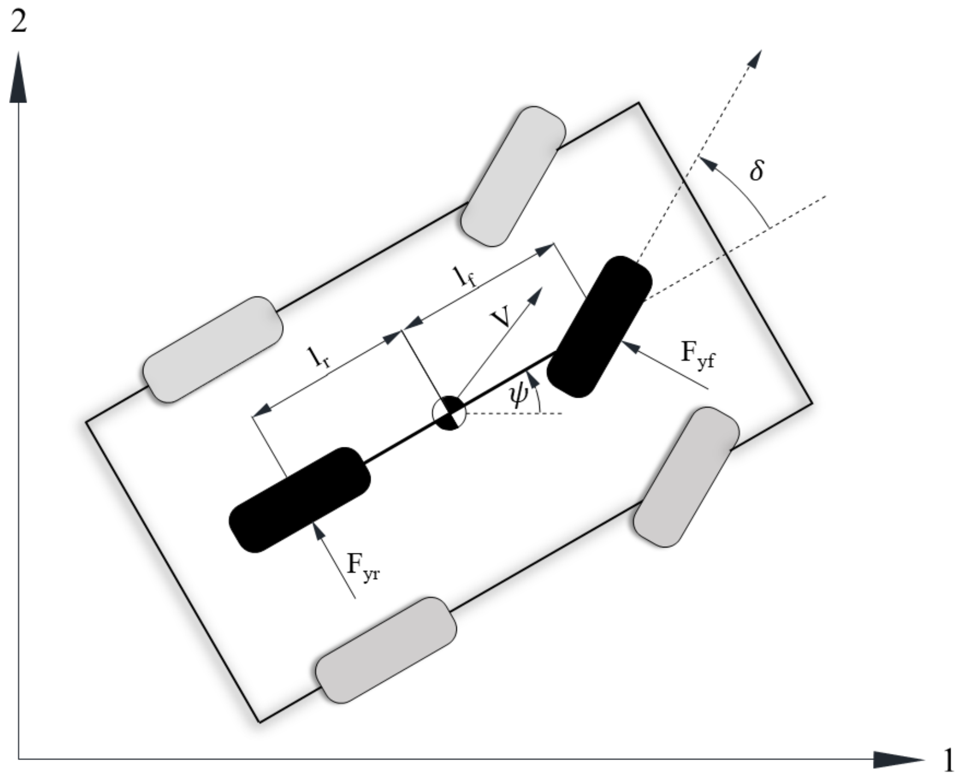

Figure 2.2: Lateral Bicycle Model

The model used in this thesis is shown in Figure 2.2. The model has two degrees of freedom, lateral position and heading angle. The dynamic model is derived using Newton's laws of motion. The dynamic model is given as:

$$m\ddot{y} = -m\dot{\psi}V_x + 2F_{yf} + 2F_{yr} \tag{2.5}$$

$$I_z\ddot{\psi} = 2l_fF_{yf} - 2l_rF_{yr} \tag{2.6}$$

where $F_{yf}$ and $F_{yr}$ are the front and rear tire force, $V_x$ is the longitudinal vehicle speed, $\ddot{y}$ and $\ddot{\psi}$ are the body frame acceleration in the lateral direction and the yaw angular acceleration about the center of mass of the vehicle respectively. Vehicle parameters $m$, $I_z$, $l_f$, and $l_r$ represent mass, yaw inertia and distance for the front axle and rear axle to the center of mass. The above model assumes that longitudinal velocity is constant and that there is no rolling or pitching [26]. Global $X$ and $Y$ positions must also be calculated for use in the model and for validation of the simulation. These calculations are shown in the following equations,

$$\dot{X} = V_x \cos(\psi) - \dot{y} \sin(\psi) \tag{2.7}$$

$$\dot{Y} = V_x \sin(\psi) + \dot{y} \cos(\psi) \tag{2.8}$$

where $\dot{X}$ and $\dot{Y}$ are global velocities in the $X$ and $Y$ directions. These equations are integrated using Euler's method in order to find $X$ and $Y$ position of the vehicle. Euler's method was used for simplicity, however future work could include using a Runge-Kutta method for better accuracy. The input to the resulting model is the front steering wheel angle, and the outputs through integration of the dynamic model are lateral velocity $\dot{y}$, lateral position y, yaw angle $\dot{\psi}$ and heading angle $\psi$.

## 2.2 Tire Model

In vehicle handling analysis, there are many different tire models that can be used to model the tire and road interaction. The Pacejka model is chosen because of its relative accuracy and general use in vehicle dynamic modeling. Though a linear tire model may be more computationally efficient, the loss of accuracy at high lateral acceleration or near tire saturation is considered to be too costly. The Pacejka model is used for representing the front and rear tire forces, and is based on a semi-empirical model [27]. The lateral force is described below:

$$F_{yf} = D\sin\{C\tan^{-1}[B(1-E)\alpha_f + E\tan^{-1}(B\alpha_f)]\} \tag{2.9}$$

$$F_{yr} = D\sin\{C\tan^{-1}[B(1-E)\alpha_r + E\tan^{-1}(B\alpha_r)]\} \tag{2.10}$$

where,

$$D = a_1 F_z^2 + a_2 F_z \tag{2.11}$$

$$C = 1.3 \tag{2.12}$$

$$B = \frac{a_3 \sin[a_4 \tan^{-1}(a_5 F_z)]}{CD} \tag{2.13}$$

$$E = a_6 F_z^2 + a_7 F_z + a_8 \tag{2.14}$$

and,

$$\alpha_f = \delta - \frac{\dot{y} + l_f \dot{\psi}}{V_x} \tag{2.15}$$

$$\alpha_r = -\frac{\dot{y} + l_r \dot{\psi}}{V_x} \tag{2.16}$$

Eq. (2.11 - 2.14) are functions of $F_{zf}$ and $F_{zr}$ which are the vertical forces on the front and rear tires. $F_{zf}$ and $F_{zr}$ are calculated by determining the weight split on the front and rear tires due to the location of the center of gravity. Eq. (2.11 - 2.14) also are functions of $a_1 - a_8$, these are tire parameters that themselves are dependent on many other factors including specific tire, road, and weather conditions [27]. It is also important to note that in the model used in the Nonlinear Model Predictive Controller and in the MATLAB simulations, weight transfer was not considered, therefore just static front and rear load are assumed. In the tire slip angles calculated in Eq. (2.15) and Eq. (2.16), $\delta$ is the front steering wheel angle.

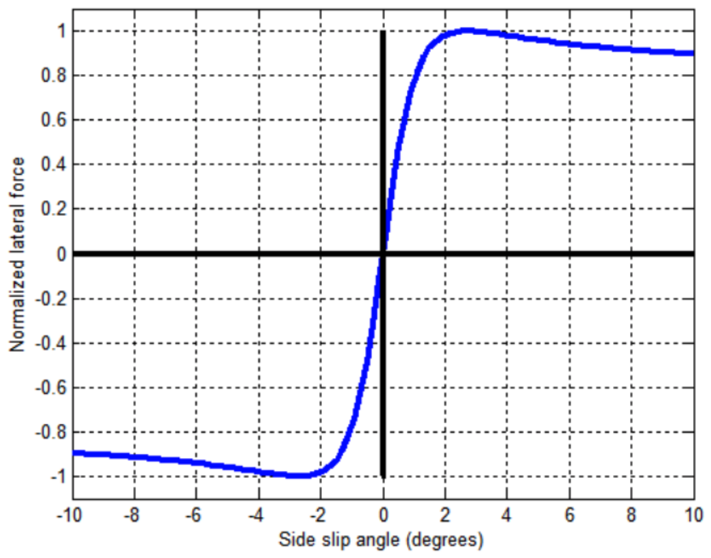

Figure 2.3: Pacejka Tire Model

Figure 2.3 shows an example of the Pacejka tire curve. The lateral tire forces are calculated using both slip angle $\alpha$ and the normal force $F_z$. This plot shows a relatively linear region at small slip angles and then as the slip angle increases the curve becomes nonlinear, this affect is commonly referred to as tire saturation.

## 2.3 Vehicle Dynamics Blockset Vehicle Model

The dynamic lateral vehicle model along with the nonlinear tire model is effective and works well with the NMPC that provides future control inputs based on the approximation of the model. Also, for initial testing the model is a good approach for studying the vehicle's characteristics close to steady state conditions. Results from these MATLAB simulations will be shown later in this thesis. In order to test the algorithm on a higher fidelity model, the Simulink Vehicle Dynamics Blockset (SVDB) [28] provides a two track vehicle model that is more realistic.

The two track or dual track model is a Six Degree of Freedom (6DOF) vehicle, where the movement is defined by the bicycle model along with vertical vehicle displacement, roll, and pitch. This means that a suspension model needed to be incorporated into the dynamics, as well as first-order dynamics applied to load transfer of the vehicle [29]. The model also takes advantage of the Pacjeka tire model for the lateral tire dynamics as well as longitudinal dynamics. Below Figure 2.4 shows a free body diagram used to develop the model.

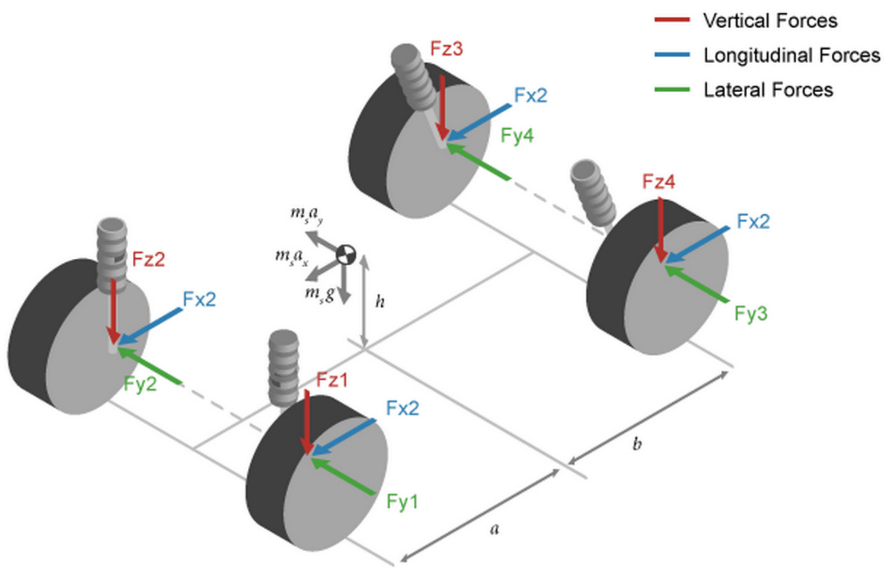

Figure 2.4: Two Track Model Vehicle Dynamics Blockset

The equations required to develop the two track model used by the vehicle dynamic blockset, is out of the scope of this thesis because they were not directly used in calculations. For implementation of this higher fidelity model within simulation, Simulink was used. The only computed input to the model was steering angle, and a constant longitudinal velocity. The model was modified with vehicle parameters from an Infinity G-35 as well as realistic tire parameters, this implementation is discussed in Chapter 5.

# Chapter 3

# Proportional Navigation Collision Avoidance Algorithm

## 3.1 Considerations from Proportional Navigation

Proportional Navigation is inherently designed to intercept and collide with a target. In theory, if the algorithm was designed to collide with a target, it should be possible to adapt it for avoidance of an obstacle. Development of this idea began with developing a "ghost" target. The "ghost" target would be placed at a specific distance relative to the obstacle. If the user collides with the "ghost" target, than obstacle avoidance is ensured. This idea was extended in a way that allowed the "ghost" target to dynamically change location depending on the geometry of the user/obstacle engagement scenario.

## 3.2 Obstacle Avoidance Algorithm Development

The collision avoidance algorithm is designed to maintain a safe distance between the car and the obstacle. This algorithm has been named the Proportional Navigation Collision Avoidance Algorithm (PNCAG). Figure 3.1 shows a configuration for the collision avoidance scenario. The obstacle (O) in this scenario is represented by a vehicle in a two-dimensional ground plane. The obstacle cone region is formed by the points A,B, and the car coordinates (C). For collision avoidance, a collision avoidance vector is defined as $\overrightarrow{CB}$ or $\overrightarrow{CA}$. The algorithm may chose to use either the $\overrightarrow{CB}$ or the $\overrightarrow{CA}$ vector depending on the geometry of the engagement.

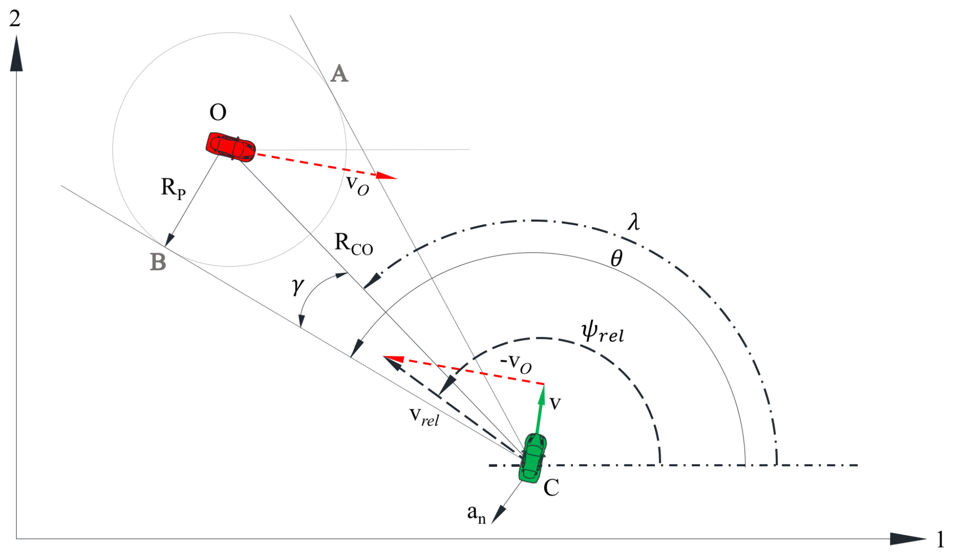

Figure 3.1: Obstacle PN Engagement Geometry

In Figure 3.1; $\lambda$ is the (LOS) angle, $\theta$ represents the direction of the collision avoidance vector, $\gamma$ is the collision avoidance vector, $\psi_{rel}$ is the direction of the relative velocity vector, $R_P$ is the set safety distance, C is the user vehicle, and O is the obstacle vehicle. This algorithm applies the "ghost" target to either point A or B, making either point the "ghost" target for the user vehicle to hit. If the user hits A or B then the obstacle will be avoided. Implementation of the PNCAG algorithm within a navigation system is shown in Table. 3.1, in pseudo code. Navigation mode represents normal autonomous vehicle operation where the vehicle is acting based on various possible navigation methods [30]. In collision avoidance mode, the car should perform collision avoidance maneuvers along its path to the goal. This division is done in order to show how the algorithm can be implemented along with a separate navigation mode.

Table 3.1: Pseudo Code for Obstacle Avoidance

```
Do while (Car does not reach the goal)
        Calculate
          $v_{rel} = v - v_O$
          $\gamma = \sin^{-1}\left(\frac{R_P}{R_{CO}}\right)$
        if (Relative velocity vector is out of the obstacle cone)
          Navigation mode initiated
        Else
          Collision avoidance initiated
        End
End
```

The algorithm first calculates the relative velocity vector ($v_{rel}$) between the car and the obstacle. If the vector is within the collision cone, collision avoidance mode is initiated. Within collision avoidance mode, the PNCAG algorithm steers the vehicle towards the collision avoidance vector $\overrightarrow{CB}$ (for the specific scenario pictured). The avoidance law is:

$$a_n = N_c v_{rel} \dot{\theta} \tag{3.1}$$

where $a_n$ is command acceleration, $v_{rel} = v - v_o$ is relative velocity, $\theta$ represents the direction of the collision avoidance vector, $\dot{\theta}$ is the angular velocity of the collision avoidance vector and $N_c$ is the proportional navigation constant. In order to obtain angular velocity of the collision avoidance vector, equations need to be established that define both the LOS angle ($\lambda$) and the angle between the LOS and the collision avoidance vector ($\gamma$).

$$\lambda = \tan^{-1}\left(\frac{R_{CO2}}{R_{CO1}}\right) \tag{3.2}$$

$$\gamma = \sin^{-1}\left(\frac{R_P}{R_{CO}}\right) \tag{3.3}$$

With the above angles now defined, the rate of change of each with respect to time can be calculated by direct differentiation of the expressions. Thus the avoidance law is equivalent to the ideal proportional navigation law, such that $\dot{\theta}$ is:

$$\begin{aligned}\dot{\theta} &= \dot{\lambda} + \dot{\gamma} \\ &= \frac{R_{CO1}V_{CO2} - R_{CO2}V_{CO1}}{R_{CO}^2} - \frac{R_P\dot{R}_{CO}}{R_{CO}^2\sqrt{1 - \frac{R_P^2}{R_{CO}^2}}}\end{aligned} \tag{3.4}$$

where $R_{CO}$ and $V_{CO}$ is the distance and velocity between the UGV (Unmanned Ground Vehicle) and the obstacle car. $R_P$ represents a safety distance, which can be chosen for the algorithm. Along with the safety distance, a detection radius ($R_D$) was also implemented. The detection radius would not only represent the possible scan range of a sensor, but also ensures that the avoidance algorithm is not activated until the obstacle is considered a threat. These equations are calculated by direct differentiation of the expressions for $\lambda$ and $\gamma$. Eq. (3.4) shows the PN guidance command can be generated from position and velocity information of both the car and the obstacle. For the specific scenarios and simulations completed in this thesis, obstacle and user velocity is set as a constant, with the obstacle heading being constant and user heading controlled with steering.

The obstacle collision avoidance algorithm developed in thesis uses a safety distance circle around the obstacle for the development of the avoidance geometry. A circle was used because the tangents from the safety distance circle to the user created the collision cone area as well as the collision avoidance vectors. Also, the algorithm states that if the relative velocity vector is not within the collision cone, then the obstacle is not a threat and navigation mode continues to run. Thus, there could be a situation where the obstacle vehicle could not be considered a threat, but could still pass in front of the obstacle vehicle at a closer distance than desired. This is not ideal, therefore a different shape safety area could help account for these cases. An ellipse shaped safety distance could be used to help avoid the car passing in the path of the obstacle vehicle. The parts of an ellipse include two foci, a center, and minor and major axes. The obstacle car could be placed on the focus of the ellipse so that the majority of the major axis is in front and in line with the path of the obstacle vehicle. The ellipse major axis would be rotated to be in line with the heading of the obstacle vehicle. The collision code could still be derived by taking the tangents of the ellipse to the user vehicle. The ellipse would effectively widen the collision code, and in turn trigger collision avoidance mode in situations where the safety distance circle would not have. Overall, this could improve the algorithm as to ensure the user doesn't pass in front of the obstacle at a distance that could make the user uncomfortable.

## 3.3 Sufficient Condition for Collision Avoidance

Conditions must be defined in order to convert from collision avoidance mode to navigation mode when collision avoidance is accomplished. The sufficient conditions for collision avoidance are listed as follows:
Condition 1: Obstacle is outside of the detection range. Therefore the obstacle can not be tracked and considered a threat.

$$R_{CO} > R_D$$

Condition 2: The range between user and obstacle is greater than the safety distance. Therefore the user will not hit the obstacle.

$$R_{CO} \geq R_P$$

Condition 3: The relative velocity vector is outside of the obstacle avoidance cone. The obstacle is located behind or ahead of the direction of the relative velocity vector. Therefore the user is in no danger of colliding with the obstacle.

$$\psi_{rel} \geq \frac{\pi}{2} + \tan^{-1}\left(\frac{y_O - y}{x_O - x}\right)$$

or

$$\psi_{rel} \leq \frac{\pi}{2} + \tan^{-1}\left(\frac{y_O - y}{x_O - x}\right)$$

## 3.4 NMPC Vehicle Steering and Pro-Nav Obstacle Integration

Now with the command acceleration calculated, the command acceleration is transformed into the body frame of the vehicle. First, the current yaw angle (heading) of the user vehicle in the global frame is obtained from the vehicle states. Then the angle $\psi_{rel}$ is calculated using the following equation:

$$\psi_{rel} = \cos^{-1}\left(\frac{(\vec{b} \cdot \vec{v_{rel}})}{\left\|\vec{b}\right\| \cdot \|\vec{v_{rel}}\|}\right) \tag{3.5}$$

where $\vec{b}$ is the base vector [1,0] from which all of the global angles are measured. Since the command acceleration is perpendicular to the relative velocity, the angle between the command acceleration and the body frame lateral vector is easily determined. Thus, the command acceleration can now be moved into the body frame of the vehicle. Then, in order to implement and use the PNCAG output, the command acceleration must be integrated and implemented as a reference for the NMPC to track. Integration is needed because lateral acceleration is not a typical vehicle state, however lateral velocity is. The implementation is shown in Eq. (3.6)

$$\dot{y}_{ref} = \dot{y} + a_c \Delta t \tag{3.6}$$

where, $\dot{y}_{ref}$ is a reference lateral velocity, $\dot{y}$ is the current lateral of the vehicle, $a_c$ is the command acceleration and $\Delta t$ is the discrete time step of the simulation. This method was found to be most effective in both MATLAB and Simulink Vehicle Dynamics Blockset simulations.

# Chapter 4
# Nonlinear Model Predictive Control

In general, a NMPC controller can be used on a controlled process with the state $x(n)$ that can be measured at discrete time instances. The lateral vehicle model in this thesis is a controlled process; “controlled” meaning that the steering input $u(n)$ is used to change the future behavior of the state of the vehicle. For the lateral vehicle model, the NMPC controller performs the tracking control, in other words, the NMPC will determine the control inputs $u(n)$ such that $x(n)$ follows the reference $x^{ref}(n)$ that is supplied by the Proportional Navigation Collision Avoidance Algorithm (PNCAG) algorithm. This chapter outlines the development of the Nonlinear Model Predictive Control (NMPC) algorithm using the nonlinear vehicle model presented previously in Grune [31]. The cost and constraint functions for the NMPC are shown in the following section in detail.

## 4.1 Cost/Constraint Functions

The optimization algorithm is used to search for the optimal control signal, minimizing the predefined cost function of the form:

$$J_N = \sum_{k=1}^{N} l(y(t+k|t), x(t+k|t), u(t+k|t)) \tag{4.1}$$

where $y(t+k|t)$, $x(t+k|t)$ and $u(t+k|t)$ represent the future outputs, future states, and control signal. Also $k$ is the current control interval and is given by $k = 1, 2...N$, where N is the prediction horizon. Specifically in this thesis, the following cost function is used:

$$J_N = \sum_{m=1}^{N} \Delta x(k_i+m|k_i)^T Q \Delta x(k_i+m|k_i) + \sum_{m=0}^{N} \Delta u(k_i+m)^T R \Delta u(k_i+m) \tag{4.2}$$

where $Q$ and $R$ are weight matrices used for changing characteristics of the controller. The $Q$ matrix penalizes state deviations from the references. The $R$ matrix penalizes the rate of change of the input variable. Additionally the following equality and inequality constraints on the system states $x$, output $y$, and control input $u$, can be considered:

$$c_i(y(t+k|t), x(t+k|t), u(t+k|t)) = 0 \tag{4.3}$$

$$c_j(y(t+k|t), x(t+k|t), u(t+k|t)) \geq 0 \tag{4.4}$$

where $c_i$ and $c_j$ can be linear or nonlinear functions. These functions can be used to place constraints on the output states as well as the input states.

Since the cost function includes the difference between the future reference and the predicted output, the optimization algorithm will generate the optimal control signal $u$. This in turn, will reduce the error between the output and the reference trajectory $w$ over the prediction horizon $N$. This concept is represented in Eq. (4.5).

$$min_u J_N(x, u) \tag{4.5}$$

Subject to the constraint equations, x and y are calculated over the prediction horizon using the discretized dynamic model and output model.

## 4.2 Sequential Quadratic Programming

For implementation of the NMPC algorithm, the non-convex optimization problems must be solved. This complex task is made more challenging by the fact that the NMPC needs to run in real time. The minimization of the cost function can be accomplished using numerical algorithms. In this thesis, a sequential quadratic programming (SQP) algorithm was implemented. SQP is one of the newer methods in the area of nonlinear programming [32]. SQP is an established class of methods to solve nonlinear optimization via a sequence of

Quadratic Programs (QP) [33]. It has been shown to out perform other methods in accuracy, efficiency, and the number of successful solutions over various different problems [34].

The goal of the SQP algorithm is to be able to solve the general constrained nonlinear programming problem, shown below:

$$\begin{aligned} &\min f(x) \\ x \in \mathbb{R}^n : \; &g_j(x) = 0, j = 1, ..., m_e, \\ &g_j(x) \geq 0, j = m_e + 1, ..., m, \end{aligned} \tag{4.6}$$

where $f$ and $g_j, j = 1, ..., m$ are continuously differentiable. An accurate solution to the problem depends on the number of constraints and design variables along with the characteristics of the objective function and constraints. Given the general problem above, the formulation of the QP subproblem is based on the Lagrangian function.

$$L(x, \lambda) = f(x) + \sum_{i=1}^{m} \lambda_i \cdot g_i(x) \tag{4.7}$$

In the above equation, the matrix $H_k$ is a positive definite approximation of the Hessian matrix of the Lagrangian function. $H_k$ can be updated using any quasi-Newton method, though specifically the BFGS method [35] was used in this thesis. The simplification made in Eq. (4.7) can be executed based on the assumption that bound constraints are expressed as inequality constraints. Then the QP subproblem can be obtained by linearizing the nonlinear constraints, shown below.

$$\begin{aligned} &\min_{d \in \mathbb{R}^n} \frac{1}{2} d^T H_k d + \bigtriangledown f(x_k)^T d \\ &\bigtriangledown g_i(x_k)^T d + g_i(x_k) = 0, i = 1, ..., m_e \\ &\bigtriangledown g_i(x_k)^T d + g_i(x_k) \leq 0, i = m_e + 1, ..., m \end{aligned} \tag{4.8}$$

The above subproblem can them be solved using a Quadratic Programming algorithm at each major iteration of the SQP method. The solution from the next iterate is as follows:

$$x_{k+1} = x_k + \alpha_k d_k \tag{4.9}$$

where the step length parameter $\alpha_k$ is determined by a line search procedure in order to decrease a merit function.

## 4.3 NMPC Implementation

The NMPC was treated as a completely separate subsystem within the Proportional Navigation (Pro-Nav) and NMPC guidance algorithm. This implementation allowed the same NMPC to be used for both navigation mode or collision avoidance mode within the guidance system. In navigation mode the reference lateral velocity will direct the user to the target and in Collision Avoidance mode the reference lateral velocity will direct the user to avoid the obstacle.

The inputs of the NMPC used in this thesis include the reference lateral velocity, all the current states of the user vehicle, the discrete time step used by the simulation, and boolean value that tells the controller the current guidance mode from the PNCAG algorithm. In the controller, the prediction horizon is set, and the current reference velocity is set as a constant over the time horizon. It is important to note that in these simulations, the control and prediction horizon are the same value. Also, constraints were placed on the steering angle of the output steering angle from the NMPC.

In order to implement weights and constraints, separate objective and constraint functions were developed to be used by the NMPC controller. The objective function takes current states of the user vehicle, the discrete time step, the prediction horizon, the boolean value that represents guidance mode, and the reference lateral velocity as well as the lateral location of the center lane. One advantage of the NMPC is that it can handle multiple inputs, thus location of the center lane is used in the NMPC to return the user vehicle to the desired lane of travel during navigation mode. The constraint function takes the current state of the user vehicle, the discrete time step, and the prediction horizon.

In the objective function, the Q matrix is chosen based on the current guidance mode selected by the PNCAG algorithm. If in navigation mode, the Q matrix includes a weight on the lateral position of the vehicle. If in Collision avoidance mode, the weight is set to 0 in order to ignore the lateral position that would drive the user vehicle towards the center lane. With the Q value chosen, the function now loops through the prediction horizon using the vehicle model and cost function, both of which were described early in the thesis. The output of the objective function is the cumulative cost from the state deviation and input rate of change. In the constraint function, constraints can be placed on the states of the user vehicle rather than the inputs. In the simulations in this thesis, no constraints were placed on the state variables. However, this implementation leaves room for constraints on any variables that output states of the user vehicle.

With the functions defined, they can now be used within the SQP algorithm. The algorithm takes the objective function, constraint function, and the constraints on the steering input. SQP determines the optimal set of control inputs over the control horizon using the provided functions. Then only the first control input is output (before recomputing) by the NMPC controller to be used on either the dynamic bicycle model in MATLAB, or with the dual track model found in the Simulink.

## 4.4 NMPC Weight Determination

The Q and R matrices of the cost functions described earlier in this chapter are the typical areas in Model Predictive Control (MPC) design that usually require tuning. In general, the plant of each individual MPC design requires different weight setups depending on the number of output variables versus input variables. For adjusting Q (output weights), $n_u$ is the number of plant inputs, $n_y$ is the number of plant outputs, and $n_{yc}$ is the number of desired controlled outputs. If an output is not desired to be held to a reference value (controlled), the weight is set to a value of 0. If $n_u = n_{yc}$, then it is possible to achieve zero tracking error to the reference at steady state. If $n_u > n_{yc}$, this is known as having excess degrees of freedom, therefore the input values could drift, even if the outputs are near their reference values. One way to prevent the drift is to set target values for the excess inputs. If $n_u < n_{yc}$, there are enough degrees of freedom to keep all desired outputs near reference values. Therefore, prioritizing reference tracking must be done by adjusting the weights. Table 4.1 gives rough guidelines for Q weight values.

Table 4.1: Q Weight Guidelines

| Weight Value | Description |
|---|---|
| 0.05 | Low Priority |
| 0.2 | Below-Average Priority |
| 1 | Average Priority |
| 5 | Above Average Priority |
| 20 | High Priority |

For R (input weights) setting, there is less structure to the process. In general, if it is desired to have more aggressive changes to the inputs in order to accomplish reference tracking, the value of the inputs are set below 1. If nominal input changes are desired, then a value of 1 is set for the inputs. If less aggressive changes to the inputs are required, a value of greater than 1 for the inputs should be used.

For the specific implementation within this thesis, the number of desired control outputs depends on whether navigation mode or collision avoidance mode is active. In collision avoidance mode, the only desired controlled output is lateral velocity. In navigation Mode, both lateral velocity and global lateral position are desired controlled outputs. So for collision avoidance mode, a weight of 1 was placed on lateral velocity. For navigation mode, a weight of 0.05 was placed on the global lateral position, and 1 was placed on lateral velocity. These weights were chosen based on the Table 4.1, even during navigation mode priority was still given to lateral velocity in order to reach the goal.

# Chapter 5

# Simulation Environments

## 5.1 MATLAB Simulation

Simulations were first conducted in MATLAB, based on vehicle parameters collected on the 2003 Infinity G35 Four Door Sedan, see Figure 5.1. Nominal tire parameters were taken from a paper written by Pacejka [27]. The Infinity G35 vehicle model was chosen because, for future work, the real vehicle can be used to test the algorithms presented in this thesis. Parameters used in the simulation are shown in Table 5.1.

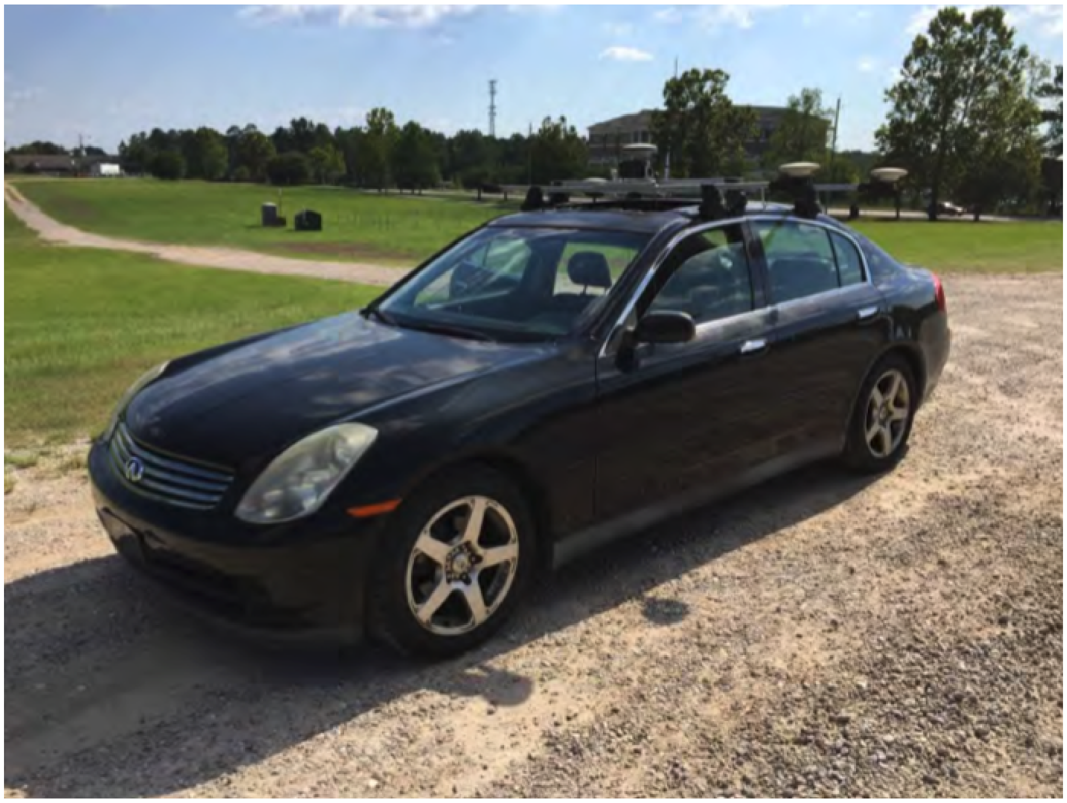

Figure 5.1: Infiniti G35 Test Vehicle

Table 5.1: Infinity G35/Tire Parameters

| Vehicle Parameters | | Tire Parameters | |
|---|---|---|---|
| $m$ | 1528 [kg] | $B$ | 0.22 $rad^{-1}$ |
| $a$ | 1.38 [m] | $C$ | 1.3 $rad^{-1}$ |
| $b$ | 1.48 [m] | $D$ | 5422 [N] |
| $I_{zz}$ | 2400 [kg m$^2$] | $E$ | -0.95 |

The MATLAB simulation was designed first in order to quickly test the algorithm to allow for tuning and verification. It is important to note that vehicle and tire parameters were also used in the NMPC controller and held constant throughout all simulations. Vehicle parameters from the MATLAB simulation were then transferred into the Simulink Vehicle Dynamics Blockset. Longitudinal velocity of the vehicle was set as a constant in all the simulation runs. User, obstacle, and target locations are given by $(X, Y)$ coordinates in a global reference frame, and the user vehicle starts at (0 m, 0 m) for all simulations. Figure 5.2 shows a visual representation of the MATLAB simulation structure used in this thesis.

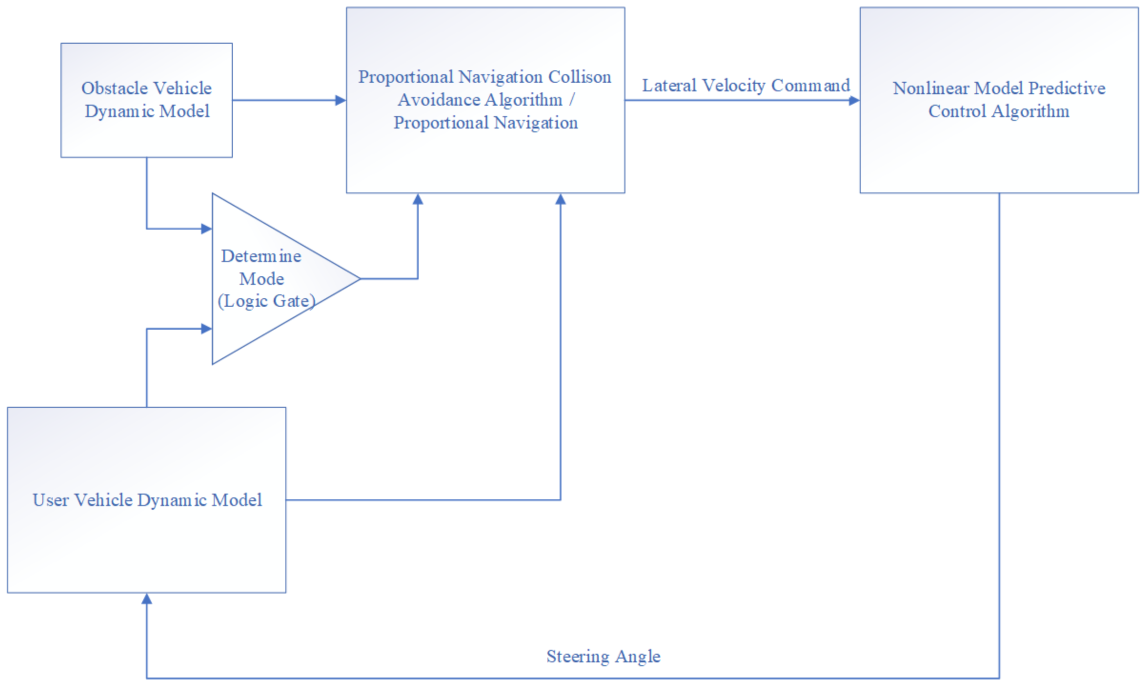
Obstacle Vehicle Dynamic Model
Proportional Navigation Collison Avoidance Algorithm / Proportional Navigation
Lateral Velocity Command
Nonlinear Model Predictive Control Algorithm
Determine Mode (Logic Gate)
User Vehicle Dynamic Model
Steering Angle

Figure 5.2: MATLAB Simulation Block Diagram

## 5.2 Simulink Vehicle Dynamics Blockset

Simulink Vehicle Dynamics Blockset (SVDB) is a new product released as a part of Matlab/Simulink 2018 for modeling and simulating vehicle dynamics. Simulink is a graphical programming environment for modeling, simulating, and analyzing dynamic systems. The interface includes a graphical block diagramming tool and block sets. The Vehicle Dynamics Blockset adds in high fidelity models for passenger cars, as well as libraries of propulsion, steering, suspension, vehicle body, brake, and tire components. For engineers familiar with CarSim, the Vehicle Dynamics Blockset has very similar features. Because the SVDB is integrated within Simulink it was chosen for this thesis. Figure 5.3 presents the Simulink model used for testing with the Vehicle Dynamics Blockset. The only input provided by the Pro-Nav NMPC algorithm to the user vehicle is the steering angle command. The desired longitudinal velocity of the vehicle was chosen and a classical PID controller was used to maintain the longitudinal velocity.

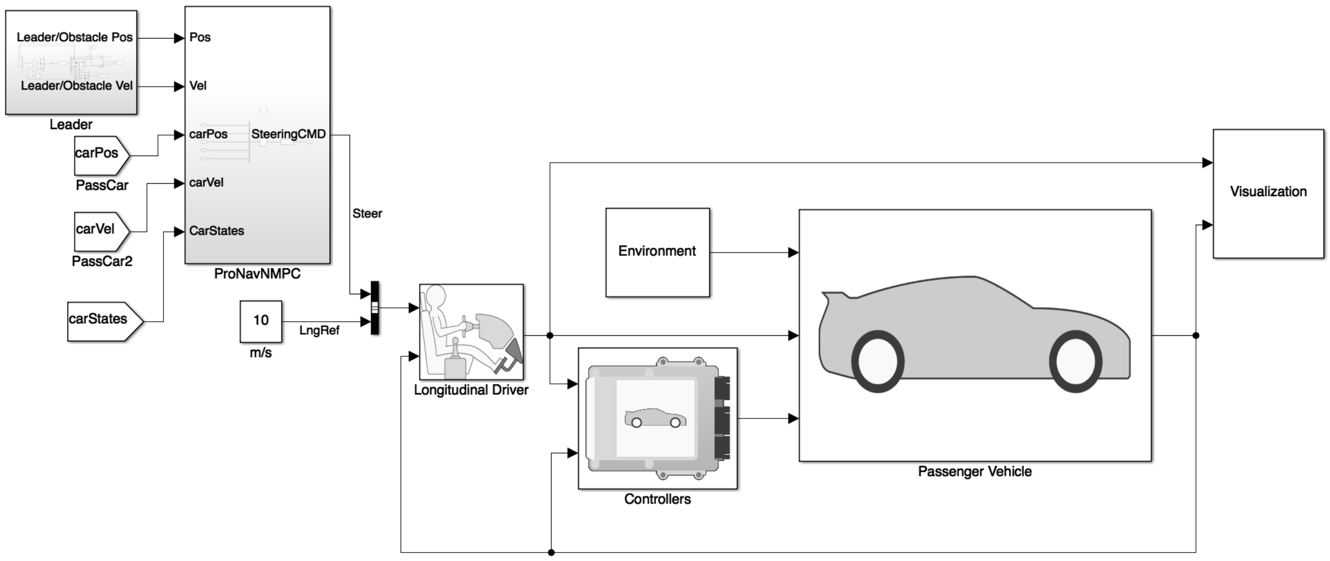

Figure 5.3: Vehicle Dynamics Blockset Simulink Model

## 5.3 NMPC Run Time

Often times when a controller is a candidate for real time implementation, run time of that controller is of interest. All the simulations in this thesis are completed with a prediction/control horizon of 10. Also, the time step of all the simulations is 0.01 seconds, thus the NMPC control horizon is 0.1 seconds. Computation time of the NMPC only is averaged over Case I of the MATLAB Simulation, Case I parameters are shown in Chapter 7. Figure 5.4 is a chart of the average time required to run the NMPC control algorithm at different control horizon lengths. This is done in order to observe the effect control horizon has on computation time.

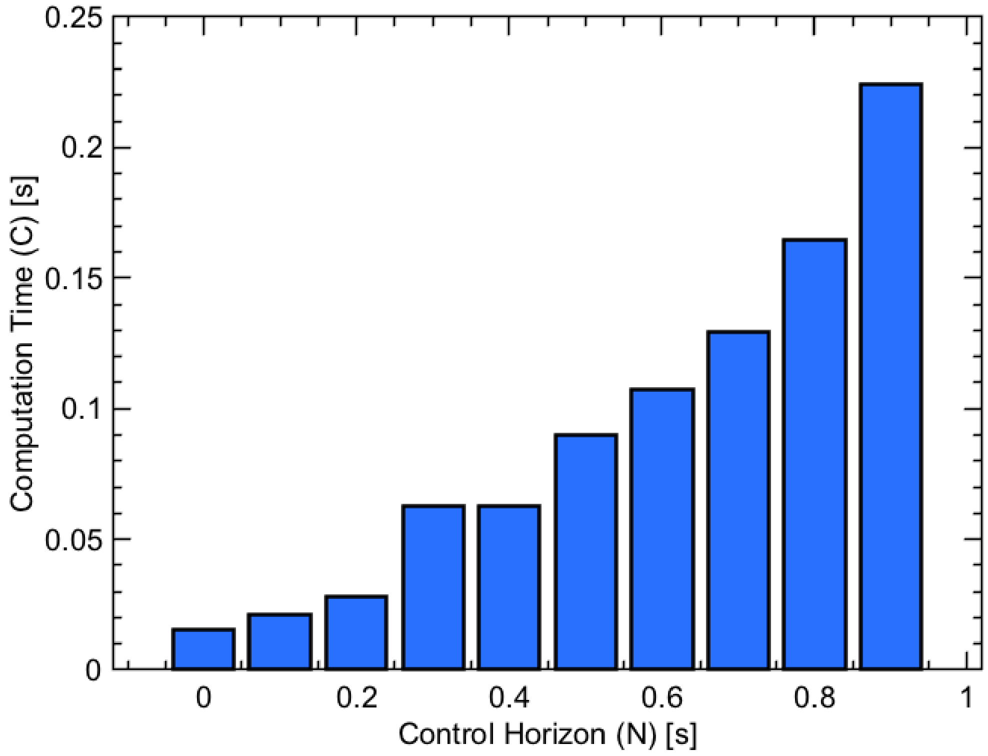

Figure 5.4: NMPC Run Times

By looking at the chart, the control horizon of 0.1 seconds being used in this thesis has a computation time of around 0.02 seconds. If only the NMPC component of the combined avoidance algorithm needed to be run, then the algorithm could be run in 50 hz system. This does not take into account time required to estimate user vehicle states or obstacle/target vehicle states in a real life scenario. However, for real time implementation, the entire algorithm can be coded into a faster programing language such as C++. This is due to the fact that MATLAB is a interpreted language, and C++ is a compiled language. C++ has been shown to greatly improve run times of MPC and NMPC in general [36].

# Chapter 6

# Vehicle Leader/Follower Example and Simulation Results

In order to demonstrate a possible use case for a Proportional Navigation (Pro-Nav) and Nonlinear Model Predictive Control (NMPC) algorithm, an example leader/follower scenario was setup. In the simulations within this chapter, the PNCAG algorithm was not used. Instead a classical proportional navigation law was implemented to track the lead car, then provide a reference lateral velocity to the NMPC. A modified Double Lane Change (DLC) maneuver was used in order to show whether the follower could successfully follow the leader vehicle with reasonable error. DLC maneuvers are used by the automotive industry to verify a vehicle dynamic path following ability. Figure 6.1 shows a DLC standard maneuver that the lead car was executing.

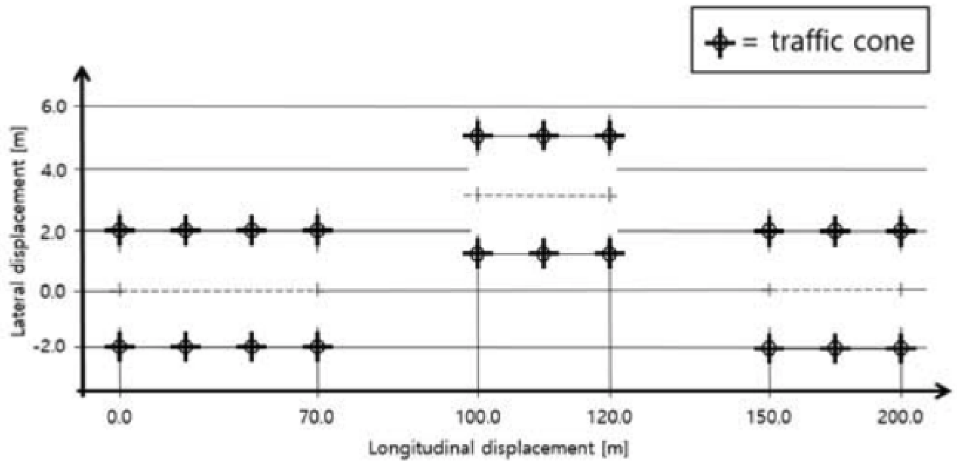

Figure 6.1: Default Path for Double Lane Change

These tests were completed in both the MATLAB environment and in the Simulink the Vehicle Dynamics Blockset (SVDB). The lead car was using the kinematic bicycle model in all simulations, while the follower was using the dynamic bicycle model in the MATLAB simulations, and the dual track model in the Simulink simulations, both of which were described in Chapter 2. The leader and follower vehicle are moving at a constant velocity of 10 m/s and performed the DLC maneuver. The leader vehicle starts 10 meters ahead of the follower vehicle in both simulations and this distance is maintained throughout. Both the MATLAB and Vehicle Dynamics Blockset simulations were setup with similar parameters, the weights for the NMPC controller were also kept the same for both simulations. A small change was also made to the NMPC algorithm. In the NMPC used with the PNCAG algorithm, a center lane lateral reference is provided during navigation mode in order to return the user car to the lane of travel. In the leader/follower version of NMPC algorithm, the lead lateral position was used as a reference instead of the center lane. The lead lateral position was used in conjunction with the lateral velocity reference to improve the performance of the follower vehicle. For both MATLAB and SVDB simulations, the parameters are as follows; prediction horizon $N$ =10, $R$ = 1, $Q = [0, 0, 1, 0, .05]$ and a time step dt = 0.01 seconds.

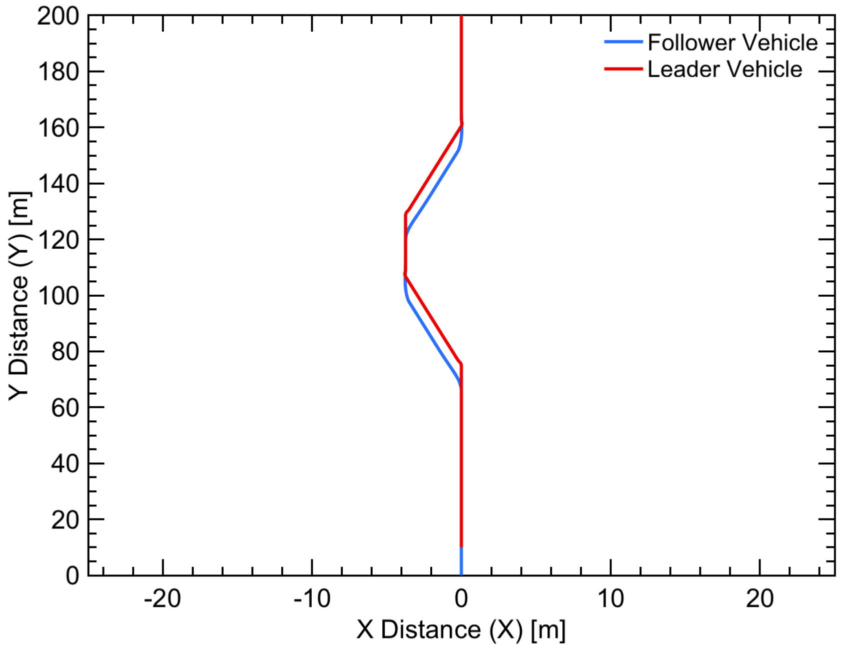

Figure 6.2: Leader/Follower MATLAB Simulation

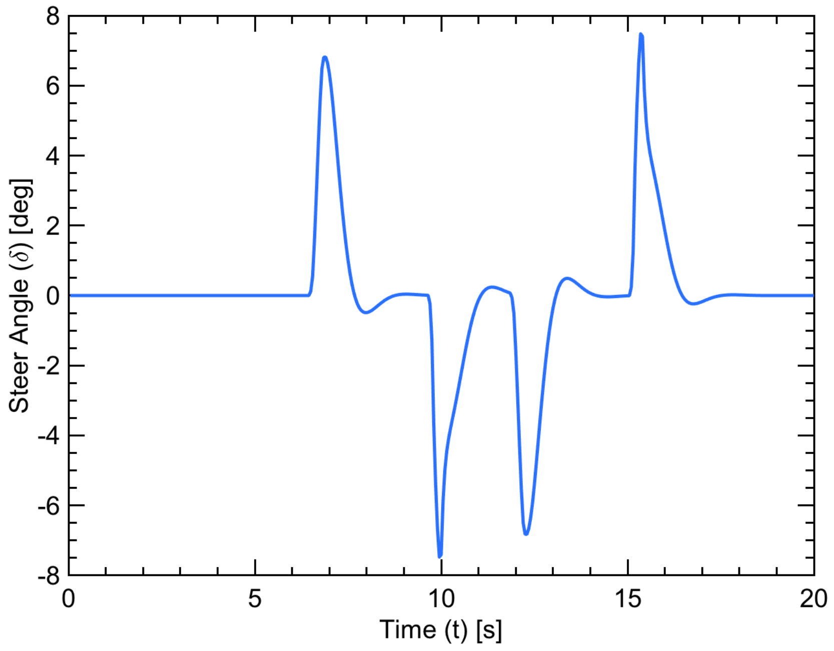

Figure 6.3: Steer Angle for MATLAB Follower

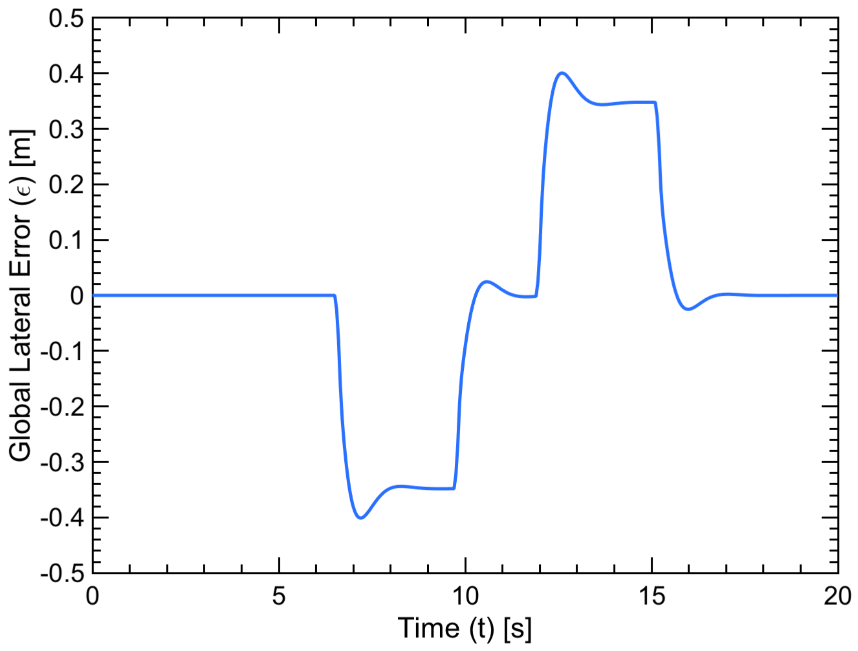

Figure 6.4: Lateral Error for MATLAB Leader/Follower

Figure 6.2 shows the DLC completed in MATLAB. Figure 6.3 shows the resulting steer angle for the follower vehicle in the DLC maneuver. As can be seen, the follower car is able to track the lead car effectively throughout with minimal lateral error. Lateral error is shown in Figure 6.4. From the plot it is observed that most of the error occurs during steering zones. This is to be expected, and the lateral error is relatively small which would still maintain the "draft" advantage gained from leader/follower.

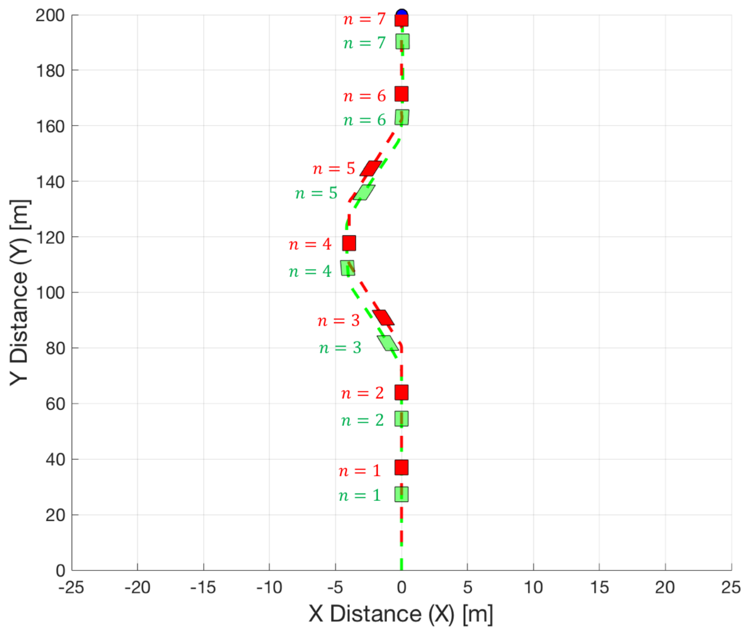

Figure 6.5: Leader/Follower SVDB Simulation

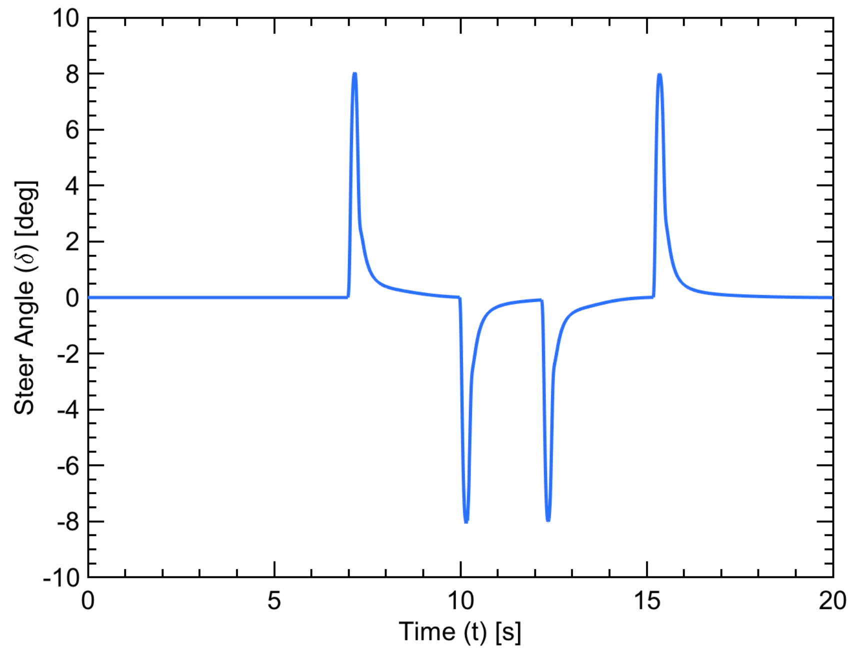

Figure 6.6: Steer Angle for SVDB Follower

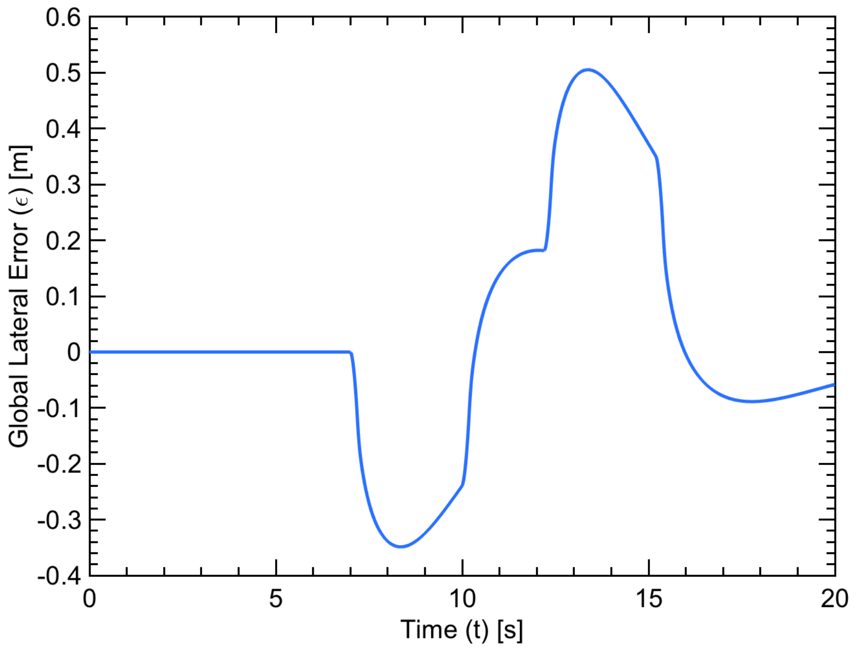

Figure 6.7: Lateral Error for SVDB Leader/Follower

Figure 6.5 shows the DLC completed in the SVDB. Figure 6.6 shows the resulting steer angle for the follower in the DLC maneuver. The follower car is again able to effectively follow the lead car through the maneuver. Lateral error again shown in Figure 6.4 shows that the error in the SVDB is increased when compared with MATLAB. This increase is most likely do to the lag in the steering system that is modeled by an actuator in the SVDB scenario. The error in the SVDB is still of similar magnitude as with MATLAB and occurs at the same points.

These results show that the follower successfully tracks the leader through a DLC scenario. Differences between the MATLAB and the SVDB simulations in vehicle path and steering input can be attributed to the differences with the fidelity of the user dynamic models. This shows the importance of matching the NMPC controller to actual dynamics of the user vehicle, within simulation and for future real world testing. More possible applications will be discussed in the conclusion and future work section of this thesis.

# Chapter 7

# Simulation and Results of Obstacle Avoidance

In order to analyze and demonstrate the performance of the proposed combined PNCAG and NMPC algorithm, simulation was done in MATLAB and with the Simulink Vehicle Dynamics Blockset (SVDB). The user car is assumed to be represented by a dynamic model and nonlinear tire model, calculated earlier in this thesis. For the simulation, five states were used; lateral position in the body frame ($y$), lateral velocity ($\dot{y}$), yaw angle($\psi$), yaw rate ($\dot{\psi}$), and lateral position in the global frame ($X$). The only input into the model is steering angle ($\delta$). The longitudinal velocity ($V_x$) is held constant or controlled by a PID in the SVDB case. There is a weight placed on the global lateral position ($X$). A reference for the lateral position of the user is set to $X = 0$ and is meant to represent the lane in which the car is driving. The weight is only used in navigation mode to keep or return the car to the desired lane. The obstacle is also simulated using a simple kinematic bicycle model as described previously in the thesis; however in the case studies shown, vehicle velocity and heading angle remain constant for the obstacle.

## 7.1 Simulation Conditions

Various simulation cases were developed in order to show the versatility of the PNCAG and NMPC algorithm. The simulation conditions for the case studies are shown in Tab. 7.1.

Table 7.1: Simulation Conditions

| | Case I | Case II | Case III | Case IV |
|---|---|---|---|---|
| Initial Vehicle Position [m] | (0,0) | (0,0) | (0,0) | (0,0) |
| Initial Obstacle Position [m] | (-100,100) | (-47,147) | (0,150) | (0,50) |
| Initial Vehicle Speed [m/s] | 10 | 15 | 20 | 20 |
| Initial Obstacle Speed [m/s] | 10 | 10 | 15 | 10 |
| Heading Angle Vehicle [rad] | $\pi/2$ | $\pi/2$ | $\pi/2$ | $\pi/2$ |
| Heading Angle Obstacle [rad] | 0 | $-\pi/4$ | $-\pi/2$ | $\pi/2$ |

Case I represents an obstacle avoidance scenario, perpendicular to the vehicle's path towards the goal. Case II represents an obstacle traveling diagonally and downward towards the vehicle. Case III represents an obstacle traveling head on towards the vehicle in the vehicle's path towards the goal. Case IV represents a passing scenario where the user car is moving in the same direction as the obstacle but a higher velocity. In all cases, for navigation mode, classical proportional navigation was used to solve for a lateral velocity in order to reach the goal in times when the obstacle is not considered a threat. The same NMPC was used for both modes of operation; navigation and collision avoidance.

These cases are meant to show possible real life scenarios. For example, Case I could be a intersection where the obstacle runs through a red light. Or Case III could represent a impending head on collision. These test cases were designed in order to ensure the PNCAG and NMPC will work as a obstacle avoidance solution in various engagement situations.

### 7.1.1 NMPC Parameters

NMPC Simulation parameters used for the MATLAB and SVDB simulations include a prediction horizon $N$ =10, $R = 1$, $Q = [0, 0, 1, 0, .05]$ in collision avoidance mode and $Q = [0, 0, 1, 0, 0]$ in navigation mode. It is important to note that Q is matrix of dimension $n \times n$ where $n$ is the number of output states. The values shown for Q are that of it's diagonal. The weight values are placed on lateral velocity ($\dot{y}$) and global position ($X$) for collision mode and just placed on lateral velocity ($\dot{y}$) for navigation mode . For the purpose of testing the algorithm no noise was added to the simulation. The time step for the simulation was 0.01 seconds.

Constraints on the steering deflection are placed at -.5 rad and .5 rad to represent not only the physical limitations of steering on the system, but also to prevent tire saturation that could cause vehicle instability. Ideally, placing constraints on lateral acceleration ($a_y$) or the front tire lateral force ($F_{yf}$) could directly prevent saturation of the tires. However, with the formulation of the NMPC, neither is a direct state, therefore direct constraints could not be applied. Placing constraints on steering indirectly will limit instability of the user vehicle.

### 7.1.2 PNCAG Parameters

In the PNCAG used in the MATLAB and SVDB simulations the obstacle has a safety distance of $R_p = 12$ m and a detection radius of $R_d = 40$ m. However, for Case IV a safety distance of $R_p = 7.5$ m, and a detection radius of $R_d = 25$ m, were changed to represent a realistic passing scenario. The inclusion of an detection radius in the simulation represents the distance in which a sensor placed on the car could get accurate measurements of the obstacle. The PNCAG also uses a Pro-Nav constant ($N_c$) of 4. A value between 3-5 is considered normal and can be tuned to improve performance.

## 7.2 MATLAB Simulations

In the MATLAB simulation engagement scenario plots, the blue line represents the user vehicle, while the red line represents the obstacle vehicle. The green cross represents the goal of the vehicle, where the navigation mode Pro-Nav algorithm is guiding the vehicle. For each scenario shown there are also plots of the input steering angle ($\delta$), the command acceleration of the body frame ($a_n$), and the distance from the user vehicle to the obstacle vehicle ($D$).

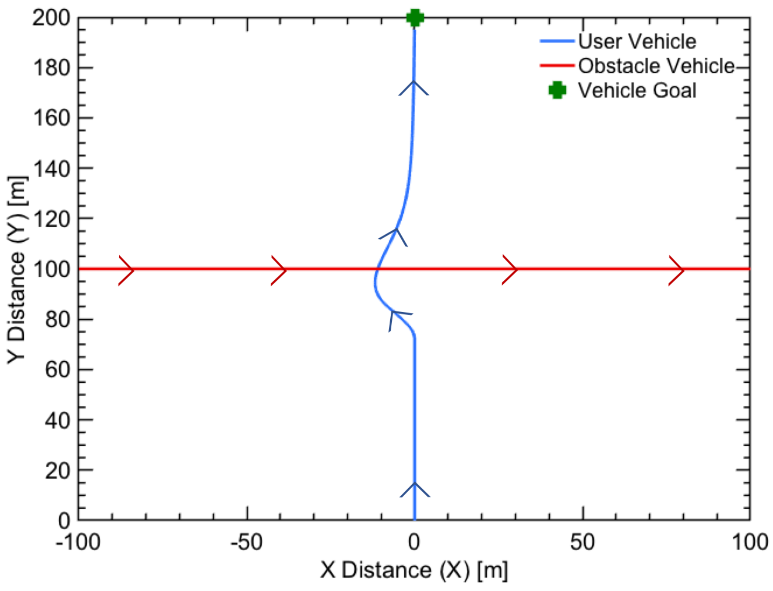

Figure 7.1: Vehicle and Obstacle Path MATLAB Case I

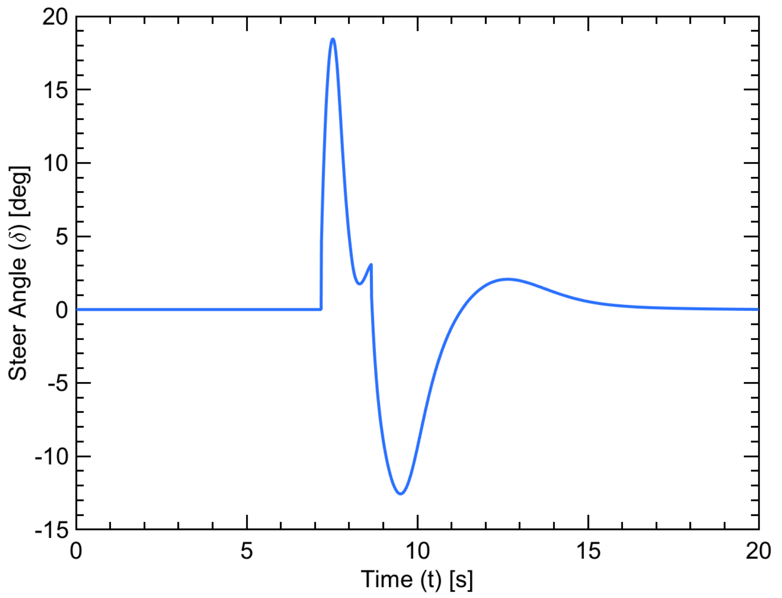

Figure 7.2: Steer Angle for Case I MATLAB

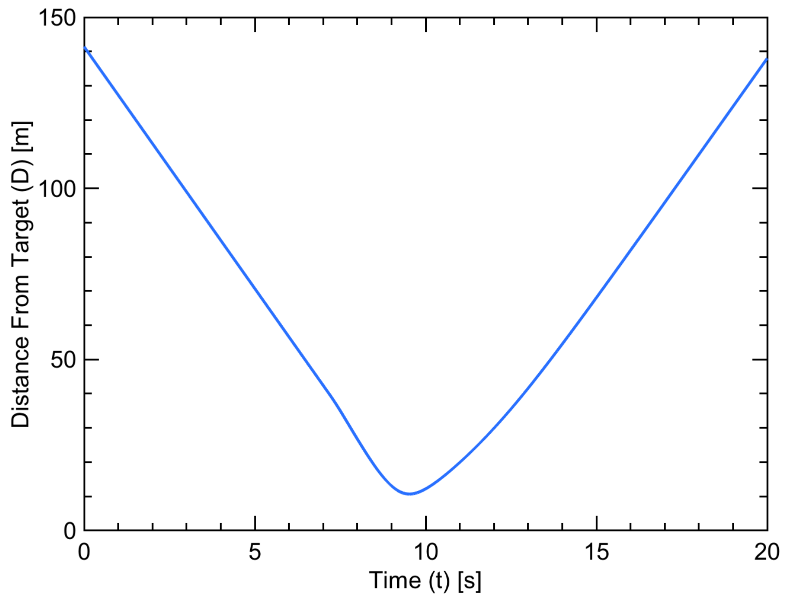

Figure 7.3: Distance from Obstacle for Case I MATLAB

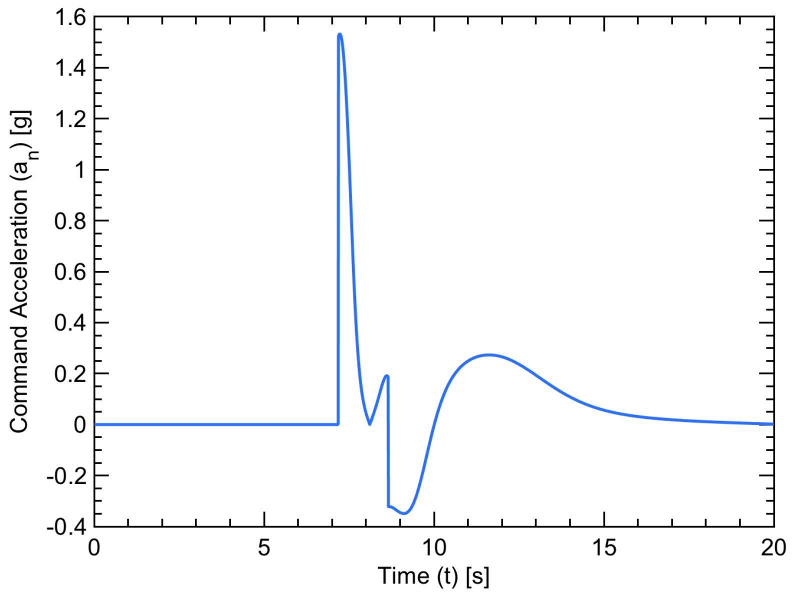

Figure 7.4: Command Acceleration (Body Frame) for Case I MATLAB

In Case I shown above, the obstacle car is detected and collision avoidance mode is initiated around the 7 second mark of the simulation. The user vehicle gets to within 12 m of the vehicle, which is the proscribed safety distance ($R_p$) but does not get any closer. Around the 9 second mark once the obstacle is no longer considered a threat, the vehicle switches into navigation mode and is navigated back towards the center of the lane and towards the goal. The user car successfully avoided the obstacle and stayed within the desired trajectory.

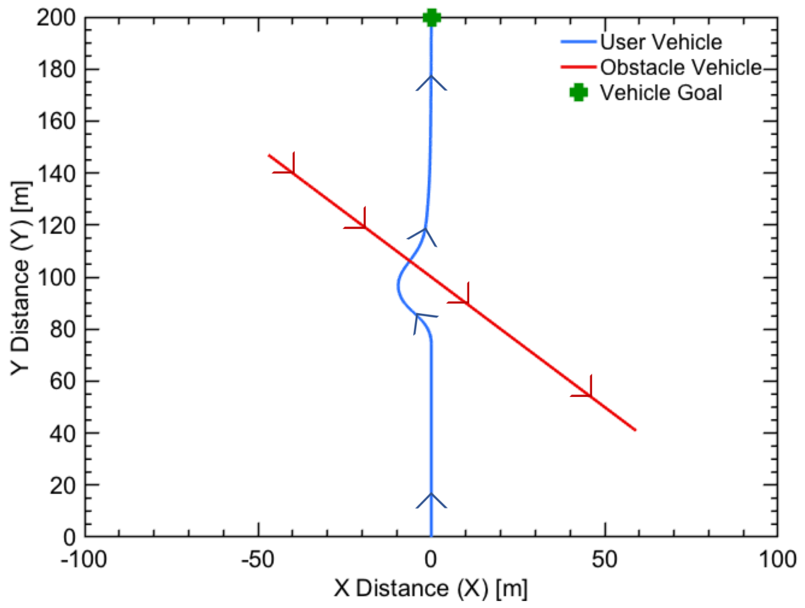

Figure 7.5: Vehicle and Obstacle Path MATLAB Case II

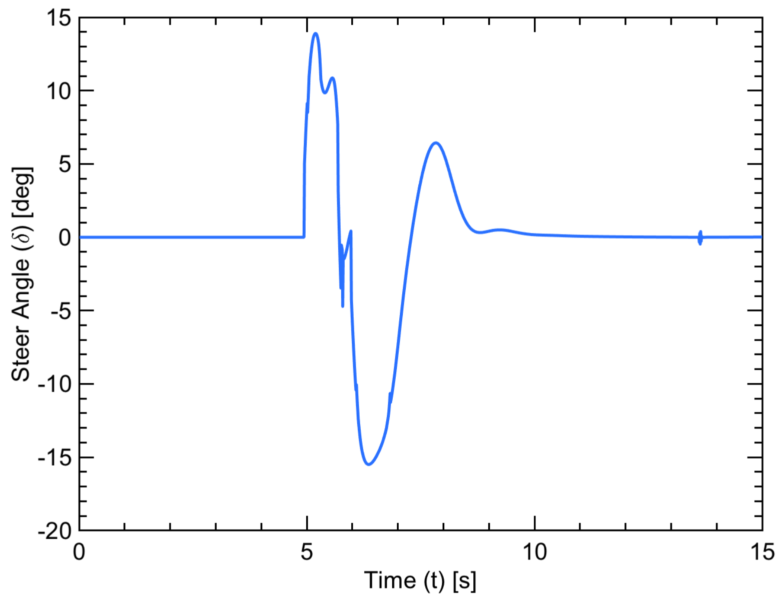

Figure 7.6: Steer Angle for Case II MATLAB

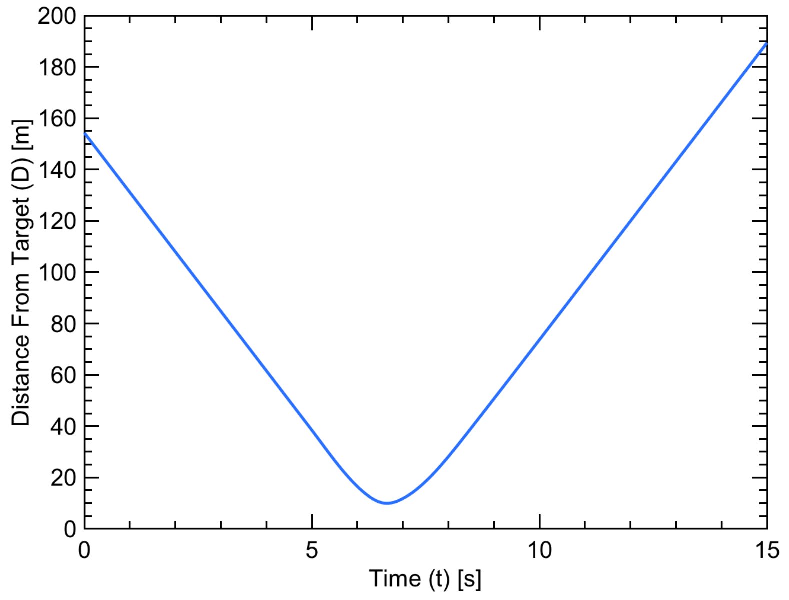

Figure 7.7: Distance from Obstacle for Case II MATLAB

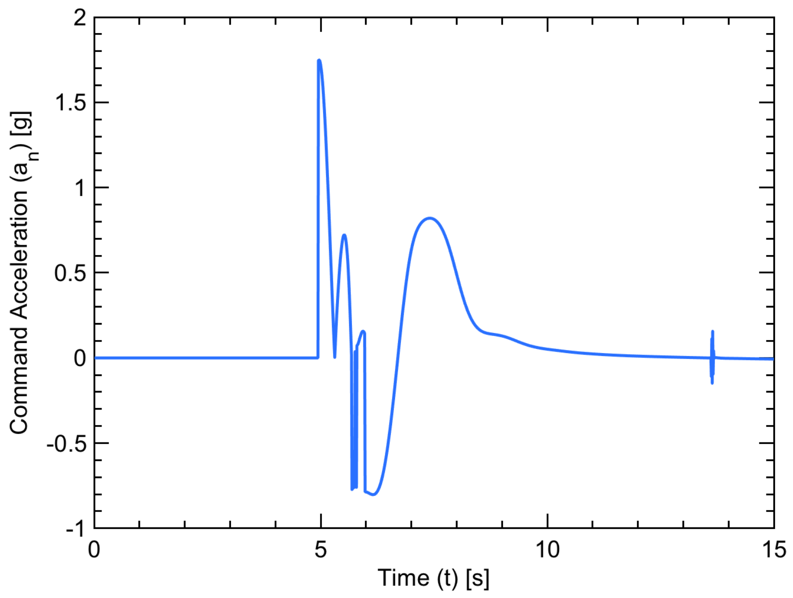

Figure 7.8: Command Acceleration (Body Frame) for Case II MATLAB

Case II shows the obstacle is again detected and collision avoidance mode is initiated around the 5 second mark of the simulation. The user maintains a distance of greater then 12 m. Once the obstacle is no longer considered a threat, the user vehicle's navigation mode is activated around the 6 second mark and the car returns to the lane, and heads towards the goal. The user vehicle again successfully avoids the obstacle to reach it's goal.

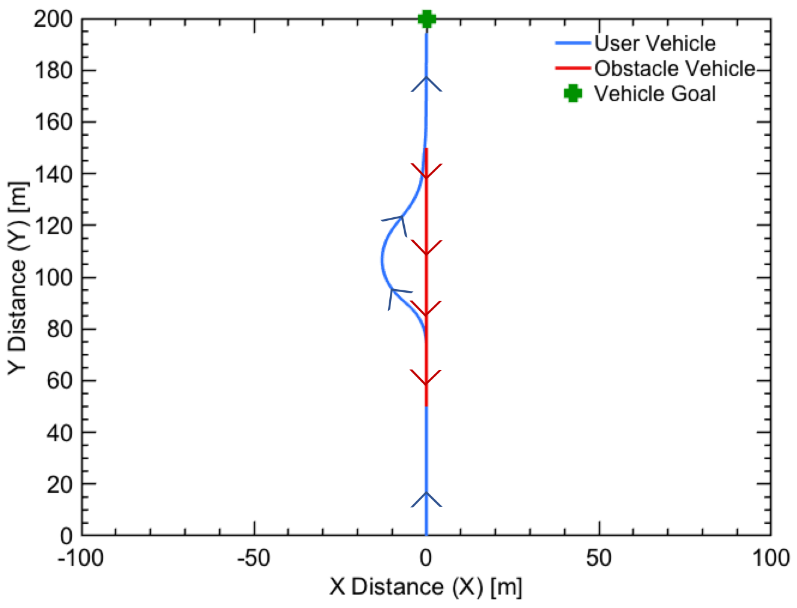

Figure 7.9: Vehicle and Obstacle Path MATLAB Case III

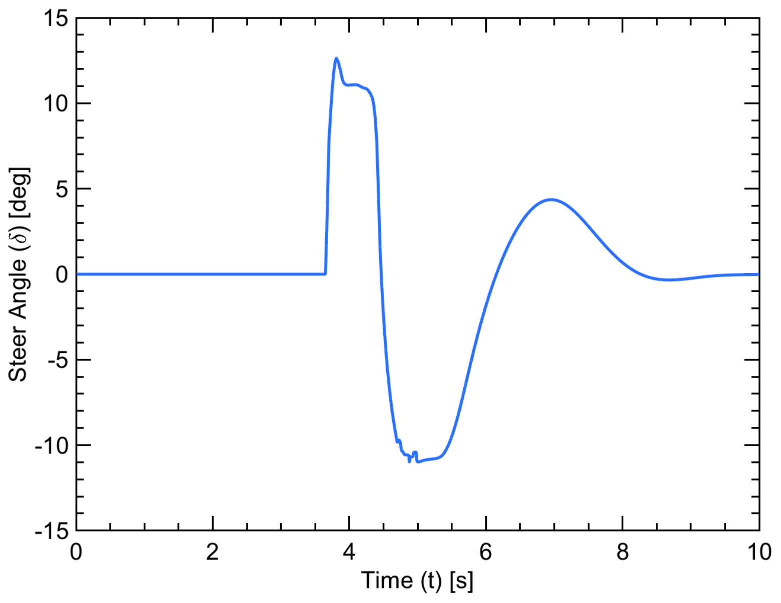

Figure 7.10: Steer Angle for Case III MATLAB

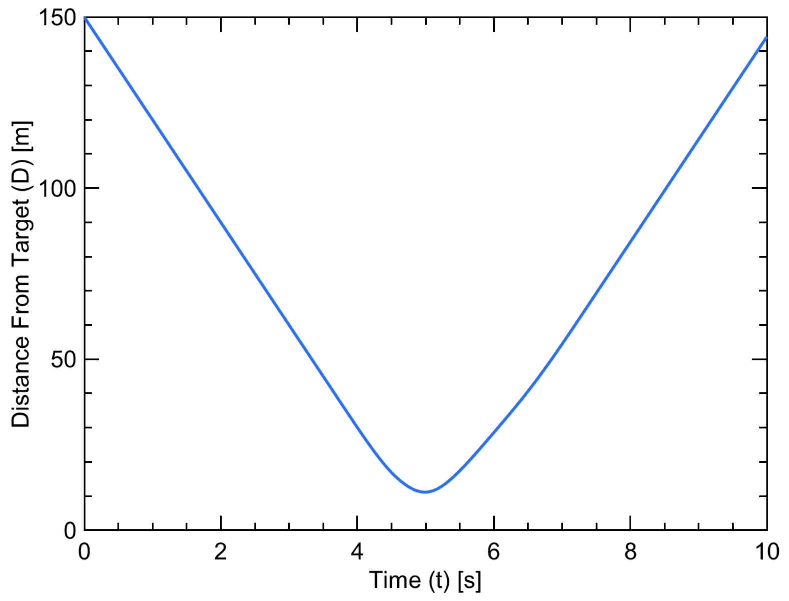

Figure 7.11: Distance from Obstacle for Case III MATLAB

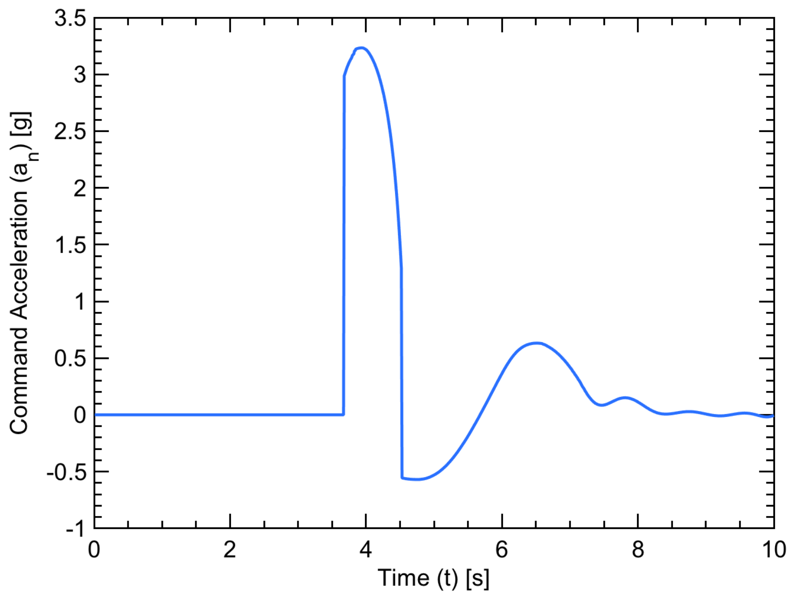

Figure 7.12: Command Acceleration (Body Frame) for Case III MATLAB

In Case III, the obstacle is detected and collision avoidance mode is initiated before the 4 second mark. The minimum safety distance of 12 m is again maintained. After the 4 second mark the user vehicle enters navigation mode and is guided back to the center lane and towards the goal. In this scenario, since the obstacle vehicle and user vehicle are both moving at an increased velocity, the NMPC controller weights that were kept constant in all the simulations were insufficient to keep the car steering from becoming slightly oscillatory.

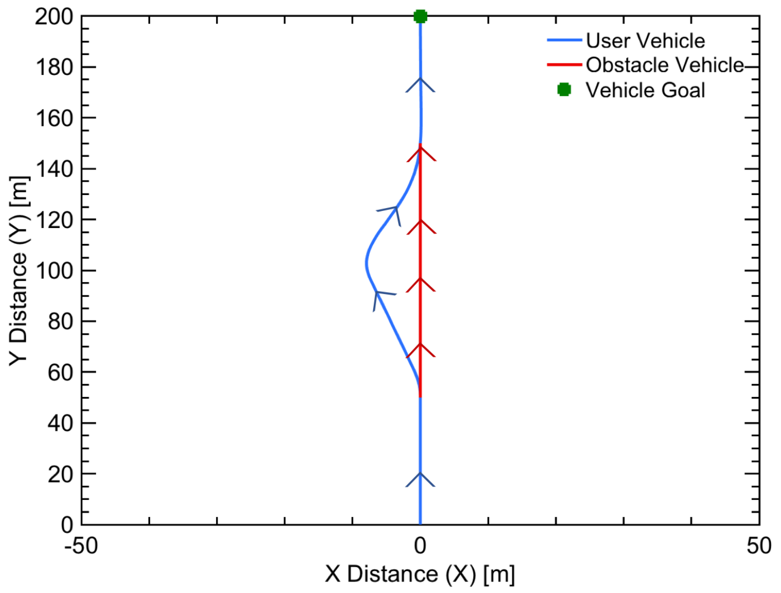

Figure 7.13: Vehicle and Obstacle Path MATLAB Case IV

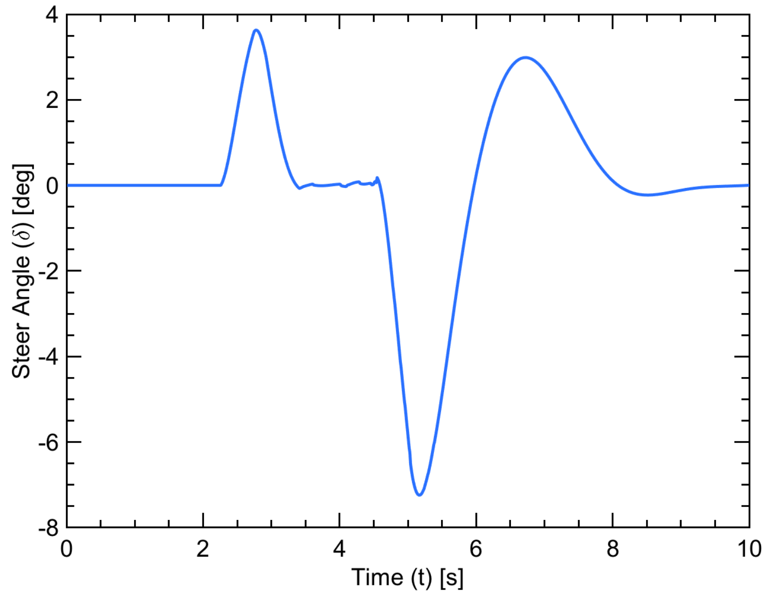

Figure 7.14: Steer Angle for Case IV MATLAB

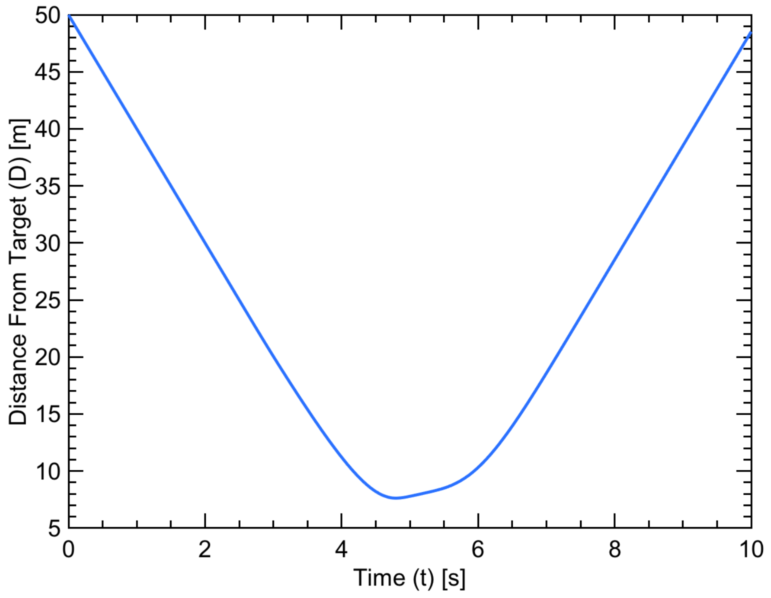

Figure 7.15: Distance from Obstacle for Case IV MATLAB

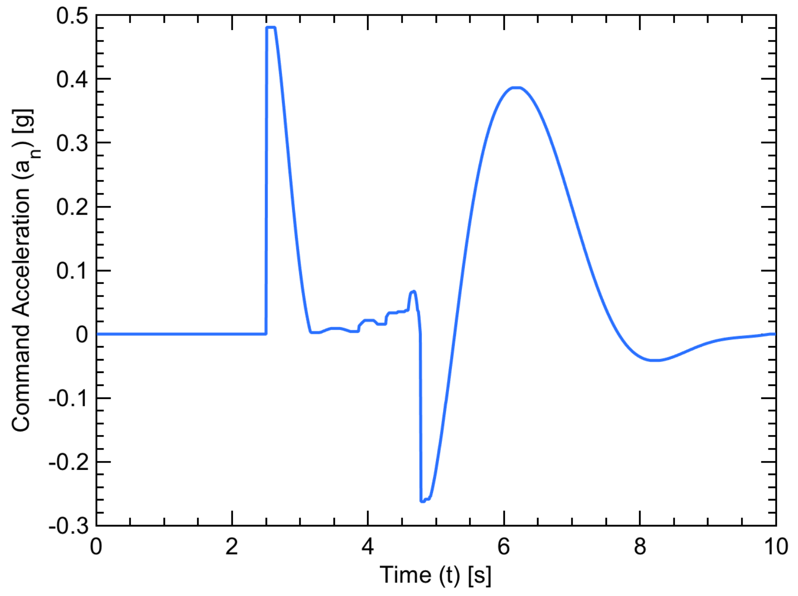

Figure 7.16: Command Acceleration (Body Frame) for Case IV MATLAB

In the last scenario, Case IV, the obstacle is detected and collision avoidance mode is initiated after the 2 second mark. The minimum safety distance of 7.5 m is maintained. At the 5 second mark the user vehicle enters navigation mode and is guided back to the center lane and towards the goal. In this scenario, since the obstacle vehicle and user vehicle are both moving at an increased velocity as with Case III, the NMPC controller weights that were kept constant in all the simulations were insufficient to keep the car steering from becoming slightly oscillatory. Future work could include dynamic weights that change with the speed of the vehicle and road conditions. However, the vehicle still successfully avoided the obstacle and reached the goal.

Simulation results from MATLAB show the UGV successfully avoids the moving obstacle in all three scenario cases. Due to the fact that the collision avoidance is steered towards the $\overrightarrow{CB}$ vector, the UGV always avoids the car in the negative $X$ direction. A conditional statement could be included within the algorithm that would steer the car towards the $\overrightarrow{CA}$. An example of the user vehicle steering toward the vector $\overrightarrow{CA}$ is shown in the next section.

## 7.3 Simulink Vehicle Dynamics Blockset

Next, the same scenarios were completed in the SVDB. All simulation parameters between MATLAB and SVDB remained the same. The vehicle path in this case is represented in a different way to better visually understand the engagement scenario. The red rectangle represents the obstacle, while the green rectangle represents the vehicle. The vehicle and obstacle are shown at various points throughout the simulation with the objective to better represent the path as the user vehicle is accomplishing avoidance. Also, the achieved lateral acceleration ($a_y$) of the vehicle was plotted for each case study. This is shown to ensure the user isn't performing any maneuvers that would be classified as "high g-force".

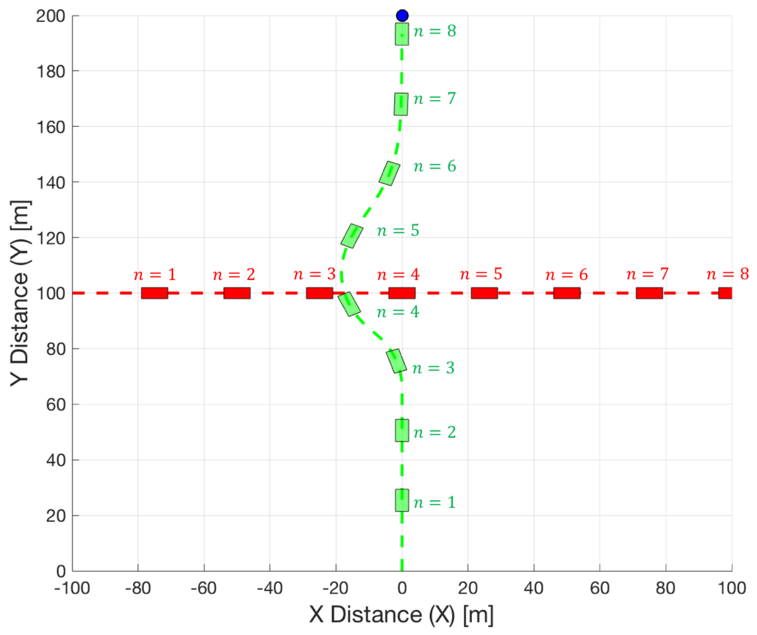

Figure 7.17: Vehicle and Obstacle Path SVDB Case I

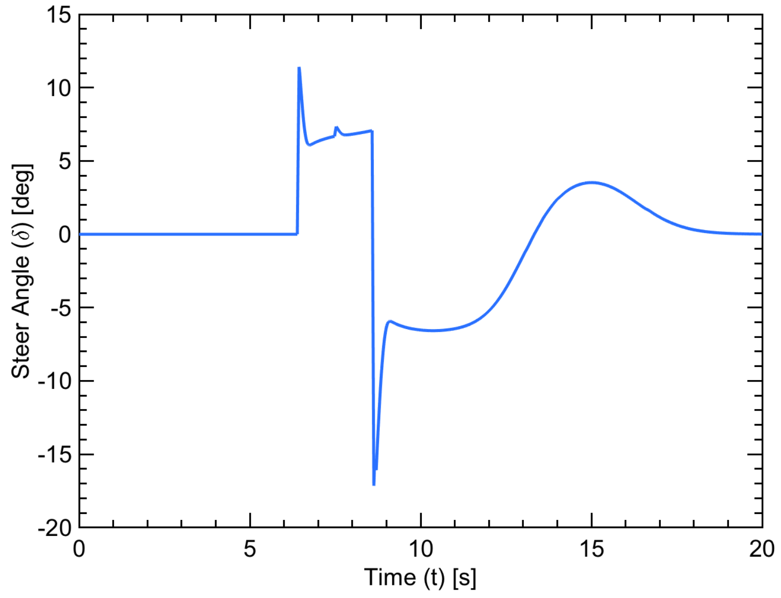

Figure 7.18: Steer Angle for Case I SVDB

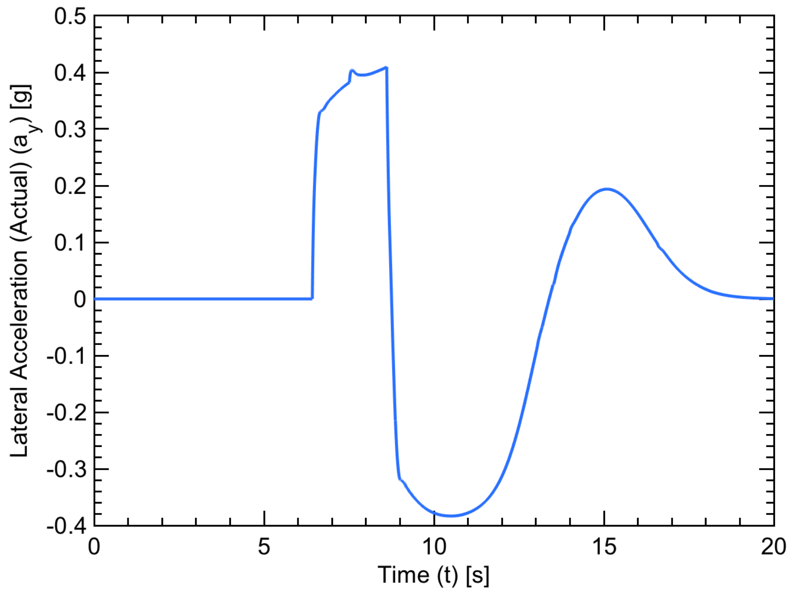

Figure 7.19: Lateral Acceleration for Case I SVDB

In the first scenario, Case I of the SVDB simulation, the obstacle car is detected and collision avoidance mode is activated around the 7 second mark. The user vehicle maintains the safety distance between the obstacle of 12 m. Navigation mode is then initiated around the 9 second mark. When compared with the MATLAB simulation, the steering input follows a very similar profile. The user vehicle successfully avoids the obstacle and reaches the goal. Figure 7.19 shows lateral acceleration ($a_y$) is within reasonable bounds and follows the same general shape as the steering plot.

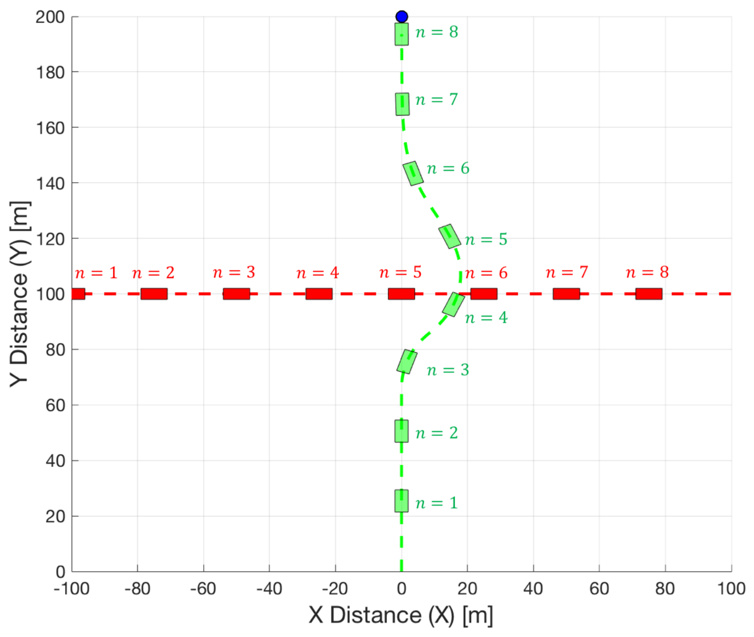

Figure 7.20: Vehicle and Obstacle Path SVDB Case I Alternate

Figure 7.20 shows the Case I scenario with the initial position of the obstacle placed at 100 m in the X Direction and with a heading of $\pi$ rad. This simulates the vehicle coming from the other direction. The PNCAG algorithm is modified to use the vector $\overrightarrow{CA}$ rather than vector $\overrightarrow{CB}$ for this scenario. As can be seen, the car avoids the car on the exact same way as with the original Case I.

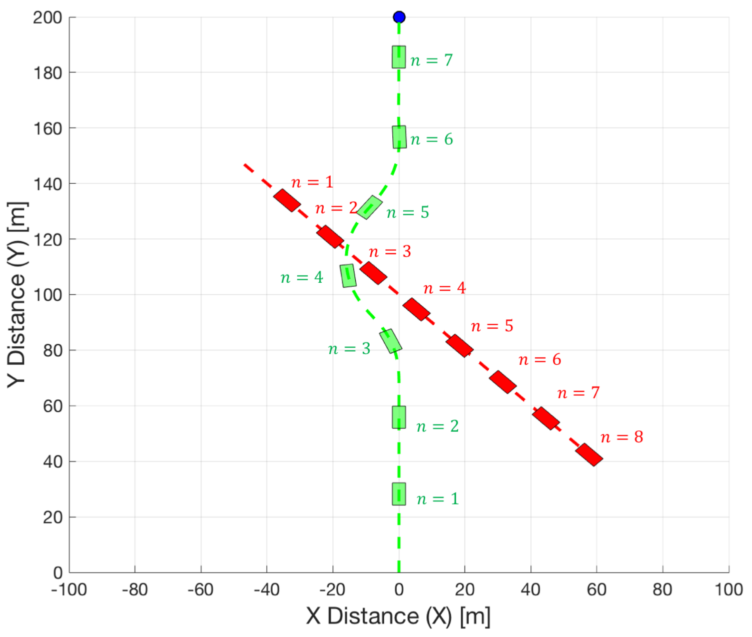

Figure 7.21: Vehicle and Obstacle Path SVDB Case II

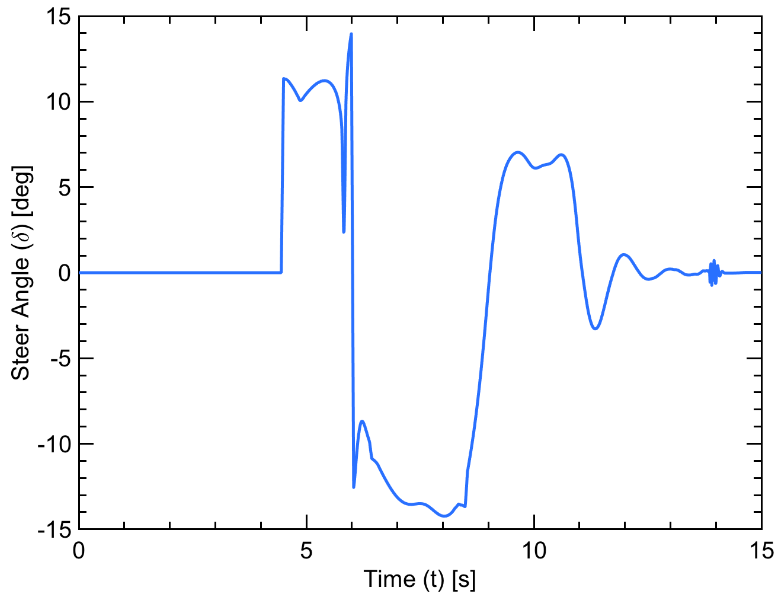

Figure 7.22: Steer Angle for Case II SVDB

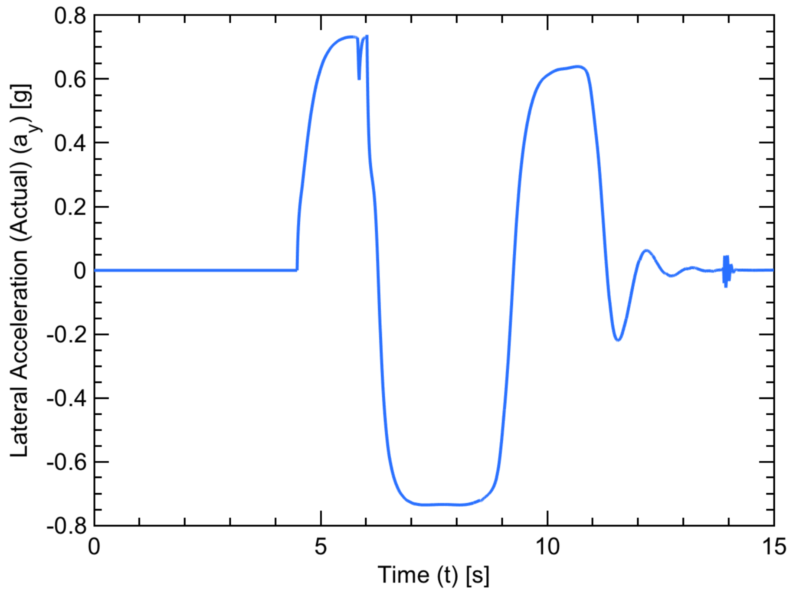

Figure 7.23: Lateral Acceleration for Case II SVDB

Case II shows the obstacle car detected and collision avoidance mode is activated around the 4 second mark. The user vehicle maintains the minimum safety distance of 12 m. At the 6 second mark, the vehicle enters navigation mode, and returns to the center lane and continues to the goal. Again, as with the previous scenario in the SVDB, the profile of the steering input matches well with the MATLAB simulation. Figure 7.23 shows lateral acceleration ($a_y$) is stable and follows the general shape as the steering plot as with Case I.

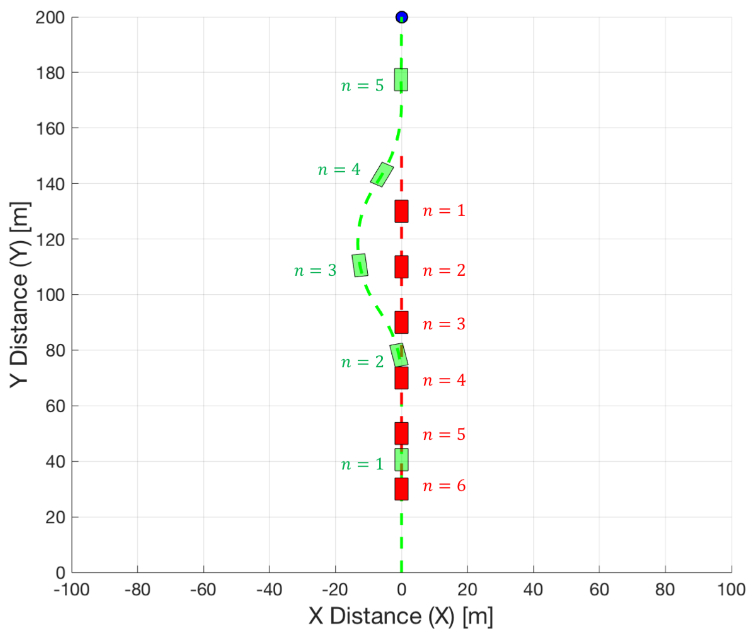

Figure 7.24: Vehicle and Obstacle Path SVDB Case III

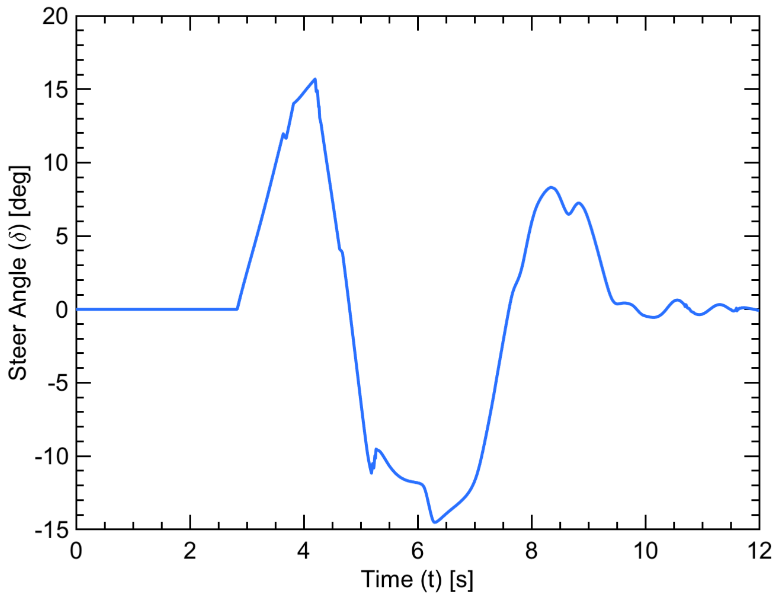

Figure 7.25: Steer Angle for Case III SVDB

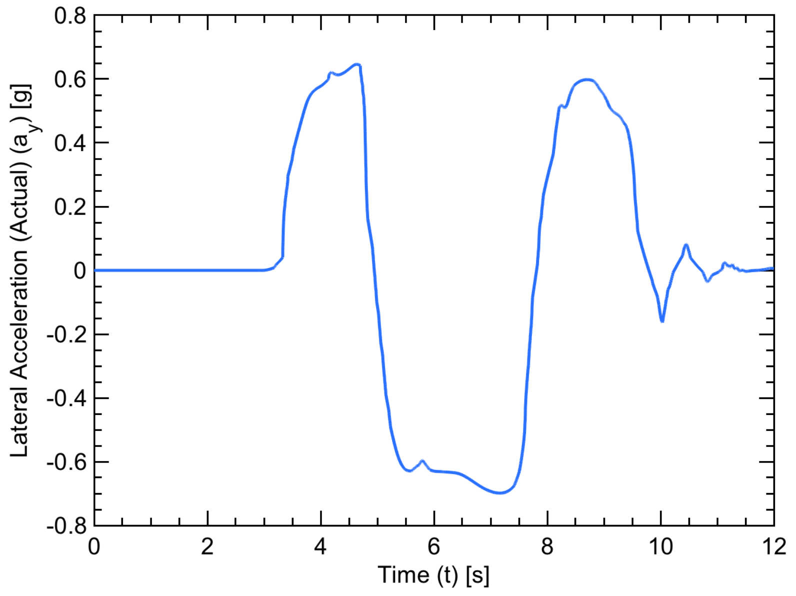

Figure 7.26: Lateral Acceleration for Case III SVDB

The Case III scenario shows the obstacle being detected and collision avoidance mode activating slightly before the 4 second mark. The user vehicle again maintains the desired 12 m of safety distance between itself and the obstacle vehicle. The user vehicle initiates navigation mode after the 4 second mark. The user vehicle then navigates back to the desired lane of the road eventually reaching the goal. As can be seen visually, the steering input has even more oscillatory effects than seen in the MATLAB simulation. This is believed to be caused by the higher speed and differences in the dynamic models of the two simulation environments, such as steering lag. Figure 7.26 also reflects the oscillatory behavior shown in the steering plot.

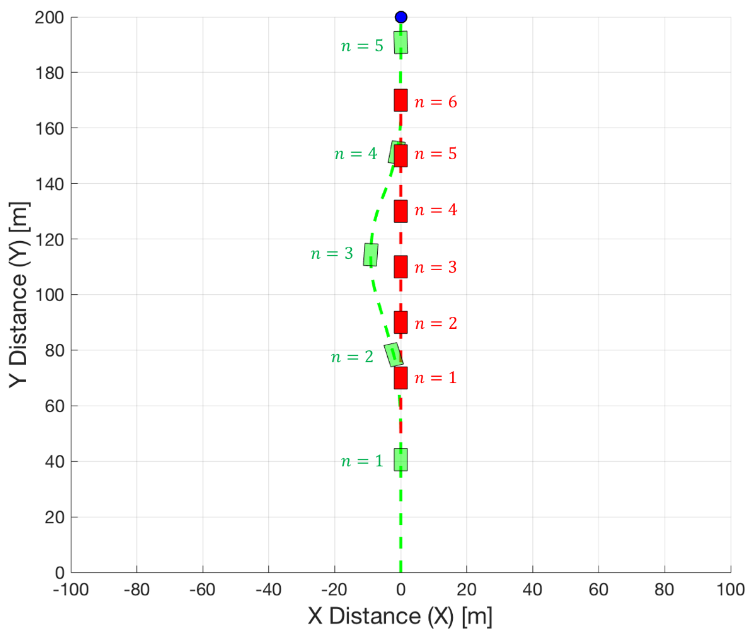

Figure 7.27: Vehicle and Obstacle Path SVDB Case IV

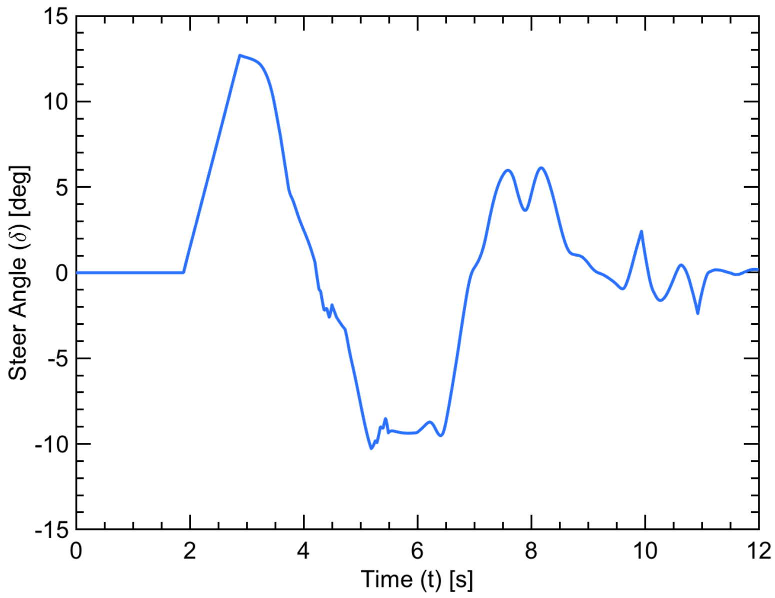

Figure 7.28: Steer Angle for Case IV SVDB

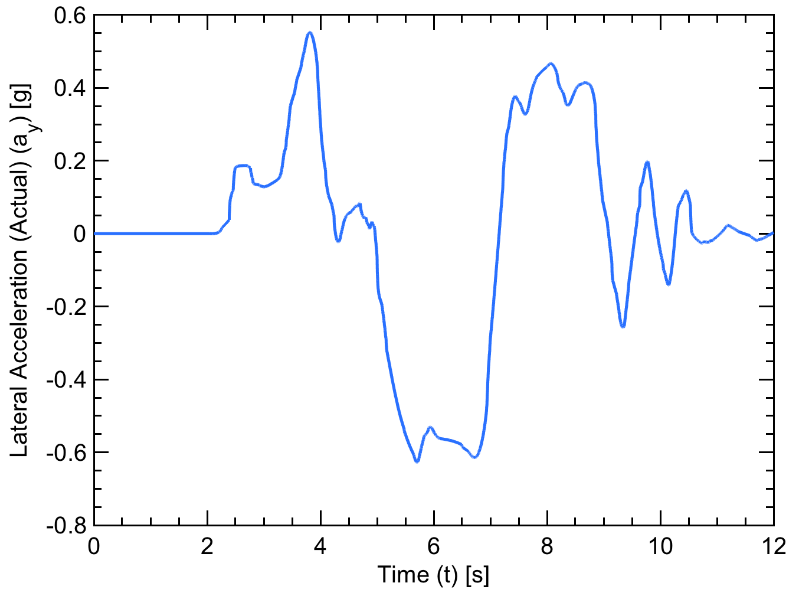

Figure 7.29: Lateral Acceleration for Case IV SVDB

The Case IV shows the obstacle being detected at a detection distance of 25 m and collision avoidance mode activating slightly after the 2 second mark. The user vehicle again maintains the safety distance of 7.5 m between itself and the obstacle vehicle. The user vehicle initiates navigation mode between the 4 and 5 second mark. The user vehicle then navigates back to the desired lane of the road eventually reaching the goal. As with Case III, the steering input has even more oscillatory response. Again, this is believed to be caused by the higher speed and differences in the dynamic models of the two simulation environments. Figure 7.29 also has a increased amount of oscillatory response.

Simulation results from the SVDB show the UGV successfully avoids the moving obstacle in all four scenario cases. As was seen with the Leader/Follower scenarios, the differences between the dynamic bicycle model used for the NMPC and the dual track dynamic model used for simulation, caused some delay in the steering and user car responsiveness. However, the having a different model in the SVDB simulation shows the controller can still accomplish the goal even when conditions are not ideal.

# Chapter 8

# Conclusions and Future Work

## 8.1 Conclusions

In this thesis, a Proportional Navigation Collision Avoidance Algorithm (PNCAG) and Nonlinear Model Predictive Control (NMPC) algorithm was presented for use on an unmanned ground vehicle (UGV). The ground vehicle dynamics were first discussed and the model was used in the MATLAB simulation with the NMPC controller. The NMPC uses a lateral dynamic bicycle model with a Pacejka tire model in order to capture the nonlinearities inherent within a vehicle specifically caused by the tires. This model was chosen due to its high accuracy and relatively low computational expense when compared to a model with a higher fidelity. Also, the use of a more complicated model leads to additional vehicle specific parameters, which can actually cause more errors and more parameters that would need to be calculated for each vehicle before the controller could be implemented. This error can be seen with the Simulink Vehicle Dynamic Blockset (SVDB) simulation set. In this set of simulations, the same NMPC was used as with MATLAB, however the model used by SVDB is of higher fidelity.

The PNCAG and NMPC presented in this thesis is a unique approach to not only obstacle avoidance, but Pro-Nav in general can also be applied to leader/follower applications. The Proportional Navigation Collision Avoidance Algorithm features an avoidance cone created by the safety circle around the obstacle and the user vehicle. This ensures that the PNCAG is steering the user away from the obstacle vehicle, as long as both velocity and position of the user and the obstacle are fully established. The PNCAG algorithm in general is geometrically based allowing it to be run at relatively low computational expense at high rate. However the PNCAG can not be used to directly determine steering angle commands, which is where a NMPC controller has been used for that purpose.

In order to integrate the PNCAG and NMPC algorithm, the best way to apply the command signals being output from the Pro-Nav filter had to first be determined. Proportional navigation is generally used for homing missile guidance applications, therefore some modification had to be applied for ground vehicle implementation. Classical Pro-Nav provides a command acceleration perpendicular to the line of sight from itself to the target. However, Ideal Pro-Nav commands the acceleration perpendicular to relative velocity between the user and the obstacle instead of the LOS. In order for this command acceleration to be used; the command needs to first be placed into the body frame of the vehicle, specifically in the lateral direction. Once accomplished, the rotated command acceleration needs to be output as a lateral velocity command to the vehicle. The reason this is done is to allow the velocity command to be given as an reference for the vehicle lateral controller. An integration equation was then developed and utilized in connecting the PNCAG and NMPC. The equation takes the current lateral velocity of the vehicle, and adds a term comprised of the command acceleration multiplied by the discrete time step of the controller. This process allows the command acceleration given by the PNCAG to be used by the NMPC as reference velocity.

A major challenge of using an NMPC controller is the time and computational effort required. However, given today's current technology, computational power is at a point where nonlinear model predictive control is a viable option for use in real time scenarios. Model predictive control operates by calculating a series of inputs over future time steps. Therefore at every time step, the controller is recomputing the series of inputs causing run time to be significantly impacted. The quickness of this NMPC is in part thanks to the SIMO (Single Input, Multiple Output) nature of the problem. Also, a relatively simple cost function consisting of weights on the output and inputs of the plant was used for the prospect of application to a real world system.

Simulations of the performance of PNCAG and NMPC algorithm were conducted using two different simulation environments: a vehicle model implemented in MATLAB and vehicle model provided by the Simulink Vehicle Dynamics Blockset. Both simulations use the same Pro-Nav and NMPC formulation, however the MATLAB simulation uses the same dynamic bicycle model/tire model used in the NMPC, while the Vehicle Dynamics Blockset uses a higher fidelity model. This was done so that the MATLAB simulation could test the NMPC without any variations in the model, thus eliminating a possible source of error and ensuring the obstacle avoidance algorithm could properly be isolated and tested. The MATLAB simulation verified that the algorithm performed successfully with both leader/follower and collision avoidance scenarios. Once the algorithm was verified in the MATLAB simulation it could then be placed in the SVDB simulation. All the available parameters from the 2003 Infinity G35 along with generic tire parameters were placed into the SVDB. It was determined that this would be a reasonable test for the PNCAG and NMPC to see how it could handle deviations in the model. The results from this thesis verified that the Pro-Nav and NMPC algorithm handled the leader/follower and collision avoidance with reasonable error. The algorithm was able to follow a leader car as it executed a double lane change. The algorithm was also able to avoid collision with a moving obstacle in four different scenarios. The work in this thesis lays the groundwork to provide a valuable tool for future development. This work can also be built upon in order to improve and augment a new or existing navigation system.

## 8.2 Future Work

There are several methods that could be improved upon within the PNCAG, NMPC, and the simulation environments, as well as examining the effects of more potential sources of error. Within the PNCAG, it can be seen in the case studies that the car always avoids the target in the negative X direction due to the algorithm steering the car towards the vector $\overrightarrow{CB}$. An example of the user being steered towards vector $\overrightarrow{CA}$ instead was show in

Figure 7.19. For future development it may be useful to allow the car to move in whichever direction would be best for avoidance. Therefore, a logical statement could be implemented allowing the car to steer to the vector $\overrightarrow{CA}$ or $\overrightarrow{CB}$ depending on the avoidance scenario. Another possible additional area of algorithm validation would be to add noise to the measurements of position and velocity of the user and obstacle car in order to see how the command acceleration may fluctuate with realistic sensor noise.

The NMPC algorithm could be improved upon within several key areas. Currently, Euler's method was used for integration because quickness of the NMPC controller was a concern. In order to gain more accuracy, ode23 or ode45 may be better choices. Other investigation could be done into different nonlinear program solvers, some of which may give better performance for the PNCAG and NMPC algorithm when compared with sequential quadratic programming (SQP). Once a real world car is chosen, the soft and hard constraints could be modified to represent the physical system. Better performance of the algorithm could be achieved if the control horizons could change throughout the run, dependent on vehicle speed and the distance to the obstacle.

The PNCAG and NMPC algorithm currently limits the steering deflection of the user vehicle through a constraint within the NMPC. The limits on steering can be modified to help insure indirectly that one or more the tires does not saturate. Saturation would cause a loss in vehicle controllability. As shown in [37], reorganizing a MPC in order to take front lateral tire forces as an input instead of the steer angle would be a solution to directly constrain the tire forces to avoid saturation on the front tires. Also in [37], rear tire saturation was shown to be preventable by implementing constraints on yaw rate and lateral velocity. Constraints to prevent rear tire saturation could be implemented easily with the current controller design.

Implementation of the algorithm could also be modified. Currently classical proportional navigation and NMPC is being used as the navigation mode. The usefulness of the algorithm presented in the thesis means the navigation mode could use something less computationally heavy than NMPC, especially for simple point to point navigation. Multiple moving obstacle avoidance could be another area of interest. The algorithm could be adapted for use with weights that would determine which threat is the more immediate, then place weights on individual obstacles in order to accomplish avoidance in a systematic process.

The algorithm currently has only been applied to obstacles moving at a constant speed and direction. In theory, the update rate of the PNCAG and NMPC algorithm should allow it to adapt to obstacles that change speed and direction of travel. Depending on the rate of change of the obstacle speed and direction, including a prediction of future obstacle states may improve the PNCAG and NMPC algorithm.

Currently if the obstacle is not within the detection distance in the PNCAG, then the obstacle is not considered a threat. As soon as the obstacle is with the detection distance, the PNCAG algorithm determines if the obstacle is a threat. If the obstacle is a threat, it immediately enters collision avoidance mode. Therefore, future work could involve determining whether avoidance is achievable. One solution could be to determine if the command velocity from the PNCAG algorithm is above a constraint on the lateral velocity. If it's above the constraint, then avoidance is not possible.

Finally, real experimental implementation of LIDAR, GPS (Global Positioning System), INS (Inertial Navigation System) on the user vehicle for state estimation of the user and the obstacle vehicle is the final step before implementing and testing the system on a real vehicle. The GPS/INS would provide full information on the user vehicle, where the LIDAR could provide relative position and velocities between the user and obstacle. Once the algorithm is verified to work with these sensors, real world testing on a “drive by wire” vehicle would be possible.

# Bibliography

[1] M. Lynberg, “Automated Vehicles for Safety,” Sep. 2017. [Online]. Available: https://www.nhtsa.gov/technology-innovation/automated-vehicles-safety

[2] M. Z. H. Noor, S. A. S. M. Zain, and L. Mazalan, “Design and development of remote-operated multi-direction Unmanned Ground Vehicle (UGV),” in *2013 IEEE 3rd International Conference on System Engineering and Technology*, Aug. 2013, pp. 188–192.

[3] Y. Wei, H. Meng, H. Zhang, and X. Wang, “Vehicle Frontal Collision Warning System based on Improved Target Tracking and Threat Assessment,” in *2007 IEEE Intelligent Transportation Systems Conference*, Sep. 2007, pp. 167–172.

[4] M. Saeednia and M. Menendez, “A Consensus-Based Algorithm for Truck Platooning,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 2, pp. 404–415, Feb. 2017.

[5] H. Fritz, “Longitudinal and lateral control of heavy duty trucks for automated vehicle following in mixed traffic: experimental results from the CHAUFFEUR project,” in *Proceedings of the 1999 IEEE International Conference on Control Applications (Cat. No.99CH36328)*, vol. 2, Aug. 1999, pp. 1348–1352 vol. 2.

[6] N. R. Ruchika, “Model predictive control: History and development,” *International Journal of Engineering Trends and Technology (IJETT)*, vol. 4, no. 6, pp. 2600–2602, 2013.

[7] L. Wang, *Model predictive control system design and implementation using MATLAB*, ser. Advances in industrial control. London: Springer, 2009, oCLC: ocn403385816.

[8] F. Borrelli, P. Falcone, T. Keviczky, J. Asgari, and D. H. , “MPC-based approach to active steering for autonomous vehicle systems,” *International Journal of Vehicle Autonomous Systems*, vol. 3, no. 2-4, pp. 265–291, Jan. 2005. [Online]. Available: https://www.inderscienceonline.com/doi/abs/10.1504/IJVAS.2005.008237

[9] R. C. Rafaila and G. Livint, “Nonlinear model predictive control of autonomous vehicle steering,” in *2015 19th International Conference on System Theory, Control and Computing (ICSTCC)*, Oct. 2015, pp. 466–471.

[10] M. Kamel, M. Burri, and R. Siegwart, “Linear vs Nonlinear MPC for Trajectory Tracking Applied to Rotary Wing Micro Aerial Vehicles,” *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 3463–3469, Jul. 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2405896317313083

[11] P. Zarchan, *Tactical and strategic missile guidance*, sixth edition ed., ser. Progress in astronautics and aeronautics. Reston, Va: American Institute of Aeronautics and Astronautics, 2012, no. v. 239.

[12] N. F. Palumbo, R. A. Blauwkamp, and J. M. Lloyd, "Basic principles of homing guidance," *Johns Hopkins APL Technical Digest*, vol. 29, no. 1, pp. 25–41, 2010.

[13] Y. M. Madany, E. A. El-Badawy, and A. M. Soliman, "Optimal Proportional Navigation Guidance Using Pseudo Sensor Enhancement Method (PSEM) for Flexible Interceptor Applications," in *2016 UKSim-AMSS 18th International Conference on Computer Modelling and Simulation (UKSim)*, Apr. 2016, pp. 372–377.

[14] G. M. Siouris, *Missile Guidance and Control Systems*. New York: Springer-Verlag, 2004. [Online]. Available: //www.springer.com/us/book/9780387007267

[15] WSEAS and C. A. Bulucea, Eds., *Recent advances in computational intelligence, man-machine systems and cybernetics: proceedings of the 8th WSEAS International Conference on computational intelligence, man-machine systems and cybernetics (CIMMACS '09), Puerto de la Cruz, Tenerife, Canary Islands, Spain, December 14-16, 2009*. S. l.: WSEAS Press, 2009, oCLC: 781333311.

[16] C. Lei, H. Chaofang, and W. Na, "Obstacle avoidance control of unmanned ground vehicle based on NMPC," in *2017 Chinese Automation Congress (CAC)*, Oct. 2017, pp. 6402–6406.

[17] C. K. Kim, S. W. Kim, H. H. Nguyen, D. H. Kim, H. K. Kim, and S. B. Kim, "Path Planning for Automatic Guided Vehicle with Multiple Target Points in Known Environment," in *AETA 2017 - Recent Advances in Electrical Engineering and Related Sciences: Theory and Application*, ser. Lecture Notes in Electrical Engineering, V. H. Duy, T. T. Dao, I. Zelinka, S. B. Kim, and T. T. Phuong, Eds. Springer International Publishing, 2018, pp. 726–735.

[18] X. Hu, L. Chen, B. Tang, D. Cao, and H. He, "Dynamic path planning for autonomous driving on various roads with avoidance of static and moving obstacles," *Mechanical Systems and Signal Processing*, vol. 100, pp. 482–500, Feb. 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0888327017303825

[19] Y. Lin, S. Saripalli, P. Scowen, G. Fainekos, J. Thangavelautham, C. Youngbull, and Arizona State University, "Moving Obstacle Avoidance for Unmanned Aerial Vehicles," in *ASU Electronic Theses and Dissertations*. Arizona State University, 2015. [Online]. Available: http://hdl.handle.net/2286/R.I.35967

[20] J. Borenstein and Y. Koren, "Real-time obstacle avoidance for fast mobile robots," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 19, no. 5, pp. 1179–1187, Sep. 1989.

[21] J. Liu, P. Jayakumar, J. L. Stein, and T. Ersal, "An MPC Algorithm With Combined Speed and Steering Control for Obstacle Avoidance in Autonomous

Ground Vehicles," Oct. 2015, p. V003T44A003. [Online]. Available: http://dx.doi.org/10.1115/DSCC2015-9747

[22] J.-M. Park, D.-W. Kim, Y.-S. Yoon, H. J. Kim, and K.-S. Yi, "Obstacle avoidance of autonomous vehicles based on model predictive control," *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, vol. 223, no. 12, pp. 1499–1516, Dec. 2009. [Online]. Available: http://journals.sagepub.com/doi/10.1243/09544070JAUTO1149

[23] W. Naeem, R. Sutton, and S. M. Ahmad, "Pure pursuit guidance and model predictive control of an autonomous underwater vehicle for cable/pipeline tracking," in *Proceedings-Institute of Marine Engineering Science and Technology Part C Journal of Marine Science and Environment.* THE INSTITUTE OF MARINE ENGINEERING, SCIENCE AND TECHNOLOGY, 2004, pp. 25–35.

[24] G. An and R. Langari, "Collision Cone Based Lane Changing Model for Collision Avoidance," Oct. 2017, p. V003T33A002. [Online]. Available: http://dx.doi.org/10.1115/DSCC2017-5045

[25] R. N. Jazar, *Vehicle Dynamics: Theory and Application*, 1st ed. New York, NY: Springer, Nov. 2009.

[26] R. Rajamani, "Lateral Vehicle Dynamics," in *Vehicle Dynamics and Control.* Boston, MA: Springer US, 2012, pp. 15–46. [Online]. Available: http://link.springer.com/10.1007/978-1-4614-1433-9_2

[27] E. Bakker, L. Nyborg, and H. Pacejka, "Tyre Modeling for Use in Vehicle Dynamics Studies," Jan. 1987.

[28] Mathworks, "Vehicle Dynamics Blockset." [Online]. Available: https://www.mathworks.com/products/vehicle-dynamics.html

[29] K. Berntorp, "Derivation of a Six Degrees-of-Freedom Ground-Vehicle Model for Automotive Applications," 2013.

[30] S.-C. Han, H. Bang, and C.-S. Yoo, "Proportional navigation-based collision avoidance for UAVs," *International Journal of Control, Automation and Systems*, vol. 7, no. 4, pp. 553–565, Aug. 2009. [Online]. Available: https://link.springer.com/article/10.1007/s12555-009-0407-1

[31] L. Grune and J. Pannek, *Nonlinear Model Predictive Control: Theory and Algorithms*, 2nd ed., ser. Communications and Control Engineering. Springer International Publishing, 2017. [Online]. Available: //www.springer.com/us/book/9783319460239

[32] P. T. Boggs and J. W. Tolle, "Sequential quadratic programming for large-scale nonlinear optimization," *Journal of Computational and Applied Mathematics*, vol. 124, no. 1, pp. 123–137, Dec. 2000. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0377042700004295

[33] G. Torrisi, S. Grammatico, R. S. Smith, and M. Morari, “A variant to Sequential Quadratic Programming for nonlinear Model Predictive Control,” in *2016 IEEE 55th Conference on Decision and Control (CDC)*, Dec. 2016, pp. 2814–2819.

[34] W. Hock and K. Schittkowski, “A comparative performance evaluation of 27 nonlinear programming codes,” *Computing*, vol. 30, no. 4, pp. 335–358, Dec. 1983. [Online]. Available: http://link.springer.com/10.1007/BF02242139

[35] J. Dennis and R. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, ser. Classics in Applied Mathematics. Society for Industrial and Applied Mathematics, Jan. 1996. [Online]. Available: https://epubs.siam.org/doi/book/10.1137/1.9781611971200

[36] T. Andrews, “Computation Time Comparison Between Matlab and C++ Using Launch Windows,” p. 6.

[37] C. M. Filho and D. F. Wolf, “Driver Assistance Controller for Tire Saturation Avoidance up to the Limits of Handling,” in *2015 12th Latin American Robotics Symposium and 2015 3rd Brazilian Symposium on Robotics (LARS-SBR)*, Oct. 2015, pp. 181–186.