NON-LINEAR MODEL PREDICTIVE CONTROL FOR AUTONOMOUS VEHICLES.

by

MUHAMMAD AWAIS ABBAS

A Thesis Submitted in Partial Fulfillment
of the Requirements for the Degree of

MASTER OF APPLIED SCIENCE

IN ELECTRICAL AND COMPUTER ENGINEERING

in

The Faculty of Engineering and Applied Science

November, 2011

University of Ontario Institute Of Technology

© MUHAMMAD AWAIS ABBAS, 2011

CERTIFICATE OF APPROVAL

Submitted by **Muhammad Awais Abbas**

In partial fulfillment of the requirements for the degree of

Master of Applied Science in **Electrical and Computer Engineering**

Date of Defence: **December 1, 2011**

Thesis title: NON-LINEAR MODEL PREDICTIVE CONTROL FOR AUTONOMOUS VEHICLES

The undersigned certify that the student has presented **[his/her]** thesis, that the thesis is acceptable in form and content and that a satisfactory knowledge of the field covered by the thesis was demonstrated by the candidate through an oral examination. They recommend this thesis to the Office of Graduate Studies for acceptance.

Examining Committee

[illegible]	Carolyn McGregor Chair of Examining Committee
[illegible]	Andrew Hogue External Examiner
[illegible]	Mikael Eklund Research Co-Supervisor
[illegible]	Ruth Milman Research Co-Supervisor
	Lixuan Lu Examining Committee Member

As research supervisor for the above student, I certify that I have read and approved changes required by the final examiners and recommend the thesis for acceptance:

[illegible]	Mikael Eklund Research Co-Supervisor
[illegible]	Ruth Milman Research Co-Supervisor

Abstract

With the advent of faster computer processors and better optimization algorithms, Model Predictive Control (MPC) systems are more readily used for real-time applications. This research focuses on the application of MPC to trajectory generation of autonomous vehicles in an online manner. The operating environment is assumed to be unknown with various different types of obstacles. Models of simplified 2-D dynamics of the vehicle are developed, discretized and validated against a nonlinear CarSim vehicle model. The developed model is then used to predict future states of the vehicle. The relationship of the weight transfer to the tire slip angle is investigated. The optimal trajectory tracking problem is formulated in terms of a cost function minimization with constraints. Initially, a gradient descent method is used to minimize the cost function. A MATLAB based MPC controller is developed and interfaced with CarSim in order to test the controller on a vehicle operating in a realistic environment. The effects of varying MPC look-ahead horizon lengths on the computation time, simulation cost and the tracking performance are also investigated. Simulation results show that the new MPC controller provides satisfactory online obstacle avoidance and tracking performance. Also, a trajectory tracking criterion with goal point information is found to be superior to traditional trajectory tracking methods since they avoid causing the vehicle to retreat once a large obstacle is detected on the desired path. It is further demonstrated that at a controller frequency of $20Hz$, the implementation is real-time implementable only at shorter horizon lengths.

Keywords: Model Predictive Control, CarSim, MATLAB, Simulink, Trajectory Tracking, Autonomous Vehicles, Real-Time

Contents

List of Figures

List of Tables

Acknowledgements

My first and foremost acknowledgment goes to my supervisor, Dr. Mikael Eklund, for giving me the opportunity to work under his supervision. Without his help and encouragement I would not have achieved the results for my MASc. thesis. Under his supervision I have learnt alot, it was he who provided me with the direction and guidance for my graduate research work.

I am indeed thankful to my co-supervisor, Dr. Ruth Milman for her continuous support and valuable suggestions during the course of this research work.

I also thank many researchers and professors around the world who have helped me during the critical phases of this research. I especially thank Dr. Stavros Vougioukas at the Aristotle University of Thessaloniki, Dr. Farbod Fahimi at the University of Alberta, Dr. Robert Bitmead at the University of California San Diego and Jaemann Park at the Seoul National University of Korea.

I am grateful to the UOIT Mobile Computing for providing me access to the required licensed softwares, and the Library Services for providing me online access to e-Journals and other academic material.

Next, a special thanks to my sister and her family for their support at every stage during my stay in Canada.

Finally, I am grateful to my parents for their prayers, support and love. This thesis is dedicated to them.

Chapter 1

Introduction and Overview

The worldwide interest on autonomous vehicles (airborne, space borne, ground and submersible) has been increasing rapidly. Autonomous ground vehicles have found wide range of applications ranging form mine clearance to saving human lives in hostile environments. Alot of researchers are focusing on active steering methods to reduce the number of motor vehicle accidents and make driving experience safer. Autonomous vehicle research is in international domain with notable projects form Germany, Italy and European Union. But most notable is the *DARPA Grand Challenge*, an autonomous vehicle competition funded by the United State's Defense Advanced Research Projects Agency. With hefty spending in research, U.S. Department of Defense aims to make one thirds of its ground forces autonomous by year 2015. Another famous research project focusing on autonomous vehicles is the BErkeley AeRobot (BEAR) project [1]. The BErkeley AeRobot Team uses Model Predictive Control for real-time trajectory tracking of autonomous vehicles. The vision statement form Berkeley Aerobot Team's website is:

"We believe the autonomous systems are much more than a simple automaton, which performs the given task in a same routine ad nauseum. A simple waypoint navigation of an UAV is such an example: we believe an UAV should be a UAAV: unmanned, aerial, and autonomous vehicle. We have showcased such a capability for an UAV to sense the sur-

*rounding, compute the most optimal trajectory that avoids nearby obstacles while flying towards the original destination using a real-time trajectory generation layer using **model predictive control scheme**"* [1].

Other projects of interest from the BErkeley AeRobot Team are the *Vision-based Landing* project [1] which focuses on autonomous landing of aerial vehicles by using active sensors, another project is the *Swarms: SmartBAT Development* project which stresses on the possibility of deployment of a large number of autonomously functioning vehicles in the form of a swarm (swarming behavior in biology) to carry out a prescribed mission and to respond as a group to high-level management commands, and the *Formation Flight* project which targets to enhancing the techniques used for coordinating flights of helicopter teams [1].

This thesis presents such an application of Model Predictive Control (MPC) (Section 2.2) applied to the online trajectory generation of a ground vehicle. First, an understanding MPC is developed in a systematic way, starting form the definition of the optimal control to the evolution of the MPC from Linear Quadratic Regulator (LQR). A detailed discussion is presented on the various concepts required for the optimal control systems design such as cost function, mathematical model and constraints. Then a survey on the application of MPC to the trajectory generation of autonomous robots are presented. A detailed chapter on the methods used for this research are also presented.

A nonlinear mathematical model for a nonholonomic vehicle is developed (Section 3.2). For validity of the developed model, it is tested against a fully nonlinear *CarSim* [2] vehicle model. Tests are performed to study the effect of weight transfer on the tire slip angles. Gradient Descent Algorithm (Section 3.5) is used to find the optimal solution of the trajectory tracking problem and optimization of the cost function is subjected to vehicle dynamics. Additionally, steering angle constraint is directly incorporated into the cost function.

The simulations are performed in MATLAB's Model Based Design Environment, Simulink. CarSim vehicle model is embedded as a Simulink S-function and NMPC (Nonlinear Model Predictive Control) controller is coded in MATLAB script file and imported to Simulink by

using MATLAB Function Block. The operating frequency of predictive controller is set to 20Hz and no low level control is used. Various types of scenarios are simulated to validate the controller's performance against a fully nonlinear vehicle model. The operating environment is assumed to be unknown with various types of stationary obstacles.
An important parameter in MPC controller design is the *prediction horizon* or the *look-ahead horizon*. The effect of varying the horizon length on the overall simulation cost and the anticipated turning is explored. Then a real-time analysis is presented based on the simulation results. The effect of cold-start and warm-start initialization methods on the controller's computation time and iteration count is also investigated. Robustness testing is performed by varying the vehicle parameters and keeping the controller parameters unchanged. Two methods of vehicle's trajectory tracking criteria (Section 3.4.3) are presented and each of the method is simulated to find the best of the two methods.

1.1 Thesis Problem Statement

In this research, we explore the application of Model Predictive Control to perform trajectory tracking of a nonholonomic vehicle. We focus on the application of Nonlinear Model Predictive Control (MPC) for trajectory tracking of the vehicle because it allows for a trade off between the tracking performance and the computation cost. Additionally input and state constraints can be easily included in the control formulation which makes the method attractive for implementation in practical situations.

Some aspects which will be focused in this thesis are:

- Controller should be able to perform efficiently on a fully nonlinear vehicle model.
- Control and states constraints should be considered within the control formulation.
- Test the limits of the controller and its robustness to change in the vehicle parameters.
- Explore real-time aspects of the developed predictive controller.

- Vehicle should be able to re-plan trajectory once an obstacle is detected on the pre-planned path.

1.2 Organization of the Thesis

This thesis is divided into five chapters. ***Chapter 1*** provides a brief introduction to the research objective and thesis contents. ***Chapter 2*** develops an understanding of Optimal Control and how it lead to the development of the Model Predictive Control systems. Then a comprehensive review of the MPC implementation in industry and for trajectory tracking of autonomous vehicles is presented. ***Chapter 3*** gives a detailed account of all the methods used in the development of the MPC controller. First, development of a vehicle dynamic model is presented, then this model is validated against a fully nonlinear CarSim vehicle model. Then a trajectory tracking framework for MPC is described and two different methods of the reference trajectory following are explained. A short description of the gradient descent method is provided together with the properties of the algorithm. Concept of weight transfer and discretization is also explained in this chapter.

Chapter 4 presents simulations results of the MPC tracking controller performed on a fully nonlinear CarSim vehicle model. Different scenarios are considered for validation of the controller in different types of environments with obstacles. Controller robustness is verified by changing the vehicle's mass, track width, center of gravity and the yaw inertia. Based on these results a real-time analysis on the controller's performance is presented. ***Chapter 5*** concludes the thesis and provides guideline about the possible future work.

Chapter 2

Background and Literature Review

Control systems have penetrated every aspect of our lives. We rely on them in our daily lives in one form or another. Aerospace engineers have used them to control the position of spacecrafts [3], Civil engineers have used them to control active mass dampers for seismic protection [4], Mechanical engineers have used them to control complex milling machines [5] and Electrical engineers have used them to control the precise motion of a motor [6]. Mechanization of goal-oriented policies has grown into a hierarchy of goal-oriented control system strategies [7]. There are at least six different distinct control strategies i.e., $1)$ Adaptive control, $2)$ Hierarchical control, $3)$ Intelligent control, $4)$ Stochastic control, $5)$ Robust control, $6)$ Optimal control. Each of these strategies is explained in the following:

Adaptive control systems are the most popular in the aerospace industry. Adaptive controllers employ plant model identification methods to tune the controlled process parameters on-the-fly, which makes them adaptive to the systems whose parameters vary with time. An adaptive controller can automatically upgrade its structure or parameters to match the changes of dynamics of the controlled plant model [8].

Hierarchical control systems are comprised of control devices which are arranged in the form of a hierarchy and are connected together like a network. The control of large systems is always organized in a distributed hierarchy. This hierarchy takes the form of a tree in which each device performs its task independently. Each control element is a linked node in a hierarchical tree. Control tasks flow down the tree from superior nodes to subordinate nodes, whereas measured states flow up the tree from subordinate nodes to superior nodes. Each higher layer of nodes operates at slower operating frequency then its lower layer. Such a control technique is popular in the field of unmanned vehicles where each higher layer performs more intelligent tasks as compared to its immediate lower layer [9].

Intelligent control systems use artificial intelligence (AI) approaches such as neural networks, fuzzy logic and genetic algorithms to find the solution to a control problem [10]. According to Saridis [11], "An intelligent controller would replace a human mind in making decisions, planning control strategies, and learning new functions by training and performing other intelligence functions whenever the environment doesnt allow or doesnt justify the presence of a human operator". Intelligent control systems have found large application areas ranging from exploratory robots to mineral processing plants [12].

Stochastic control deals with the control of model based systems in which there exists some deviations in the data obtained from the plant. Such deviations occur when random noise and disturbance processes are present in a control system, so that the system does not follow its prescribed course but deviates from the latter by a randomly varying amount [13]. The exact disturbance function is unknown to the system designer but only some of its average properties are known before hand. A random signal may be generated by nature's processes like wind.

Robust control systems deal directly with uncertainty in control design. Its function is similar to Adaptive Control but there is an important difference. Robust Controllers are designed to be insensitive to the uncertainty of the controlled plant, whereas Adaptive Controllers are sensitive to plant uncertainty but they can adapt to changes on-the-fly. It should be noted that it is impossible to design a robust controller which can work for all times of uncertainties. A robust controller can be designed which is insensitive to plant uncertainties only in a bounded region [8]. So it is necessary to provide some information regarding the environment to the control designer [14] such as process model, model uncertainty bounds, types of inputs (setpoints and disturbances) and performance criteria. Such systems are popular in the areas where the environmental conditions can change from the assumed nominal operating conditions during the design process. Robust control theory measures the performance changes of a control system with changing system parameters.

Optimal control systems is a control technique in which a certain *performance index* is minimized. It is popular in applications where constrains (input or state) need to be considered in control formulation. This chapter focuses on the study of optimal control systems, more specifically Model Predictive Control systems. Then a survey is presented on the existing approaches for trajectory planning of autonomous vehicles.

2.1 Optimal Control Systems

Conventional control theory has allowed man to control and automate his environment for decades. More recently, optimal control techniques have allowed the engineers to optimize the control systems they build for cost and performance. Donald Kirk [15] in his book defines classical and optimal control techniques as:

"Classical control system design is generally a trial-and-error process in which various methods of analysis are used iteratively to determine the design parameters...The objective of optimal control theory is to determine the control signals that will cause a process to satisfy the physical constraints and at the same time minimize (or maximize) some performance criterion."

Classical control theory can be traced back to as early as 1930s with the works from Nyquist [16], Nicholas [17] and Bode [18]. They introduced the frequency response analysis methods for single-input single output (SISO) systems. In 1949, Weiner [19] introduced the first optimal control approach focusing on minimization of variance of signals. It was designed to reduce the amount of noise present in a signal by comparing it with an estimated desired noiseless signal. This filter was known as the “Weiner Filter”, however, there was a limitation to his designed filter i.e., it was applicable to the continuous-time domain only. Later, discrete-time equivalent of the Weiner Filter was derived by Kolmogorov. The combined theory of Kolmogorov and Weiner was named Weiner-Kolmogorov filtering theory. Weiner-Kolmogorov filter theory was the first statistically designed filter theory, and it lead to the development of advanced filters such as the Kalman filters.

Optimal control and estimation research gained lots of attention during the Space Race (a part of the larger Cold War between the United States and Soviet Russia for supremacy in space exploration) which was ignited by Soviet launch of *Sputnik 1* on 4th October 1957. The Space Race resulted in enormous increase in research spending. It was during this time when two ground-breaking papers by *Rudolf E. Kálmán* [20, 21] appeared in 1960. First of these two papers, deals with linear-quadratic feedback control which later came to be known as Linear Quadratic Regulator (LQR) (see next Section for further details). His second paper introduced the *Kálmán Filter* (it was formulated only for linear systems and assumed zero-mean gaussian noise). The combination of the first and the second paper formed the basis for Linear Quadratic Gaussian (LQG) control theory. Major contributions of these papers were the introduction of the concept of *state space representation* (including controllability and observability) and *filtering theory* for optimal control. The Kálmán filter was designed to produce estimates of the true values of the measurements (provided the plant is linear and continuous-time), estimating the uncertainty of the predicted value, and computing a weighted average of the predicted value and the measured value. Initially, Kálmán faced wide skepticism on his ideas. But later his developed LQ problem was included in famous textbooks [22, 23, 24, 25]. Kálmán filters were also used in the Apollo program, NASA Space Shuttle, Navy Submarines and the Cruise Missiles.

Basic Kálmán filter was limited to linear plant models, however complex systems can have nonlinear models. So Kálmán’s filter has been extended to the nonlinear plants, and

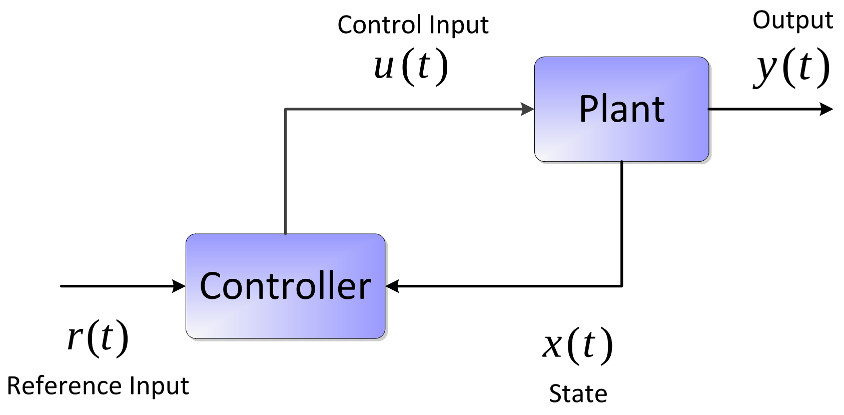


Figure 2.1: Configuration of a typical optimal control system

the extended version came to be known as the Extended Kálmán Filter (EKF). Another widely used variant of Kálmán filter is Unscented Kalman Filter (UKF) which captures a more accurate mean value of the signal by picking a minimal set of the points around the mean.

Referring to Figure 2.1, the main objective of optimal control is to determine the optimal plant (process) control input $u(t)^*$ (superscript * denotes optimal condition) which drives the system from some initial state to reference signal $r(t)$ value while minimizing (or maximizing) a performance index J. The formulation of an optimal control system at least requires the following:

- A mathematical model of the plant to be controlled
- Performance index (cost function)
- Constraints (optional)

2.1.1 Plant Description

A *plant* is described by a set of linear differential equations or nonlinear differential equations. For example a linear plant equation is written as:

$$\dot{x}(t) = Ax(t) + Bu(t) \tag{2.1}$$

$$y(t) = Cx(t) + Du(t) \tag{2.2}$$

where $dot(\dot{})$ on $x(t)$ in Equation 2.1 represents the differential of $x(t)$ with respect to time t. $x(t)$, $y(t)$ and $u(t)$ are state (n dimension), output (m dimension) and control (r dimension) vectors respectively. A is a $n \times n$ matrix, B is a $n \times r$ matrix, C is a $m \times n$ matrix and D is a $m \times r$ matrix.

2.1.2 Cost Function Description

A *cost function* is a performance index which needs to be minimized (or maximized). The most basic form of a cost function is the ***Time Optimal*** cost function in which we want the system to move from some initial state $x(t_0)$ to final state $x(t_f)$ in minimum amount of time [26]. Such a cost function is written as:

$$J = \int_{t_0}^{t_f} dt = t_f - t_0 = t^* \tag{2.3}$$

A second type of cost function is the ***Minimum-Effort*** cost function which minimizes the actuator effort $|u(t)|$ to minimize the actual cost of a process [26]. Such a cost function is formulated as:

$$J = \int_{t_0}^{t_f} R|u(t)|dt \tag{2.4}$$

where R is a weight factor. For example, in a spacecraft the control input amount $u(t)$ is equivalent to the amount of fuel spent. Minimizing the cost function in Equation 2.4 ensures minimum fuel consumption for a control move. A third and most common type of cost function is the ***General Optimal Control*** cost function or LQR cost function [26]. Mathematically it is written as:

$$J = x'(t_f)Fx(t_f) + \int_{t_0}^{t_f} [x'(t)Qx(t) + u'(t)Ru(t)]dt \tag{2.5}$$

where R, Q and F are the tuning matrices and determine the weight of each element in the cost function. The cost function 2.5 is called a quadratic cost function because terms of states and control are squared. Superscript prime ($'$) denotes transpose of a term. So the term $x'(t)Qx(t)$ is the matrix form of $qx(t)^2$. In Linear Quadratic Regulator, *linear* refers to the plant being linear and the term *quadratic* refer to the cost function with squared or quadratic factors. The term $x'(t_f)Fx(t_f)$ is called *terminal cost* and the integral term $\int_{t_0}^{t_f}[x'(t)Qx(t) + u'(t)Ru(t)]dt$ is the *running cost.*

The structure of the quadratic cost function signifies the type of the problem faced [27]. An optimal solution depends on the values of states, values of control variable and the form of the cost function [28]. If the cost function contains only terminal cost, the problem is called a ***Mayer problem***. If the cost function contains only running cost, such a problem is called a ***Lagrange problem***. The problem which contains both running and terminal cost in cost function is called a ***Bolza problem*** (this type of cost function has been used in this research work). For a detailed discussion on the history of these problem types and their solutions, the reader is refered to Peskir [29]

2.1.3 Constraints Description

One of the main problem with the LQR was its inability to deal with constraints. In real life situations, the states $x(t)$ and the control inputs $u(t)$ can have constraints on them. Simplest constraints are linear and have the following form:

$$U_{min} \leq u(t) \leq U_{max} \tag{2.6}$$

$$X_{min} \leq x(t) \leq X_{max} \tag{2.7}$$

where U_{max} and U_{min}, X_{max} and X_{min} are the upper and the lower limits on control input and states respectively. According to Qin [30] the maximum yield of process industries occurs when the plant is operated closed to the constraints boundary.

2.2 Model Predictive Control

In the late 1960s, LQR/LQG became immensely popular approach to solve the general control problems especially those related to the aerospace field. According to Goodwin [31] there are thousands of practical applications of LQG and around 400 registered patents. But LQR/LQG failed to make an impact on process industries. Some of the reasons of failure include [30]:

- inability to handle constraints
- inability to handle process nonlinearities
- lack of robustness
- unique cost function
- cultural reasons

A successful industrial controller must consider the constraints limitation during its operation [32]. As in process industries maximum yield is achieved when a process is operated near the constraints boundary [30]. No constraints were considered in LQR formulation. Industrial processes are nonlinear whereas LQG only handles linear processes. Also industrial processes change with time, and their exact model is hard to obtain as compared to the aerospace systems which have a well defined accurate mathematical model. The quadric cost associated with the LQR was not modifiable which limited the LQR applications. Processes need some extra elements in cost function to optimize production. But most important reason cited for LQR's failure is that the control engineers and technicians had limited knowledge of optimal control systems, so the process control community remained unaware of LQG benefits [30].

During the early 1970s the first few industrial applications of MPC were reported. Two of the most famous papers were by Richalet [33] and Cutler [34]. Richalet called his implementation as *Model Predictive Heuristic Control (MPHC)* and Cutler called his technology as *Dynamic Matrix Control (DMC)*. Early MPC implementations solved constrained optimal control problem over finite and receding horizon (known as look-ahead horizon).

$$LQR \, + \, constraints \, + \, finite\, horizon \, = \, simple\, linear\, MPC$$

Since dynamics of the industrial processes are slow, it was possible to solve the resulting optimization problem in an online manner even with limited computational power. It was possible to use any cost function formulation and the plant dynamic model could take any mathematical form. Constraints were included directly in the cost function to optimize performance. With the passage of time, MPC became enormously popular in the process control industry [30].

2.2.1 MPC Theory

A very general architecture of a Model Predictive Controller is given in Figure 2.2. MPC controller contains three basic functional blocks. The *Optimizer* finds the optimal control input $u^*(t)$ which when applied to the plant, gives the minimum value of the cost J. Of course, this optimization must be done in the presence of the *constraints* and the *cost function*. The *State estimator* is used to predict unmeasured states $\hat{x}(t)$ form the plant.

Model Predictive Control optimizes the output of a plant over a finite horizon in an iterative manner (Refer to Figure 2.3). Suppose the step size of the controller is T. At time step k the current plant state is sampled and the optimizer computes a cost minimizing control strategy $u^*(t)$ for finite time steps in future $k = t + 0T, t + 1T, ..., t + pT$ where p is the number of look ahead prediction horizon steps. In practical situations, the whole optimal control sequence cannot be applied to the process. This is due to the inaccurate process model and added disturbances in the process which can cause error between the predicted output and the actual process output. If only a mathematical model was used for prediction and state calculation, prediction errors could accumulate. Thus only the first step

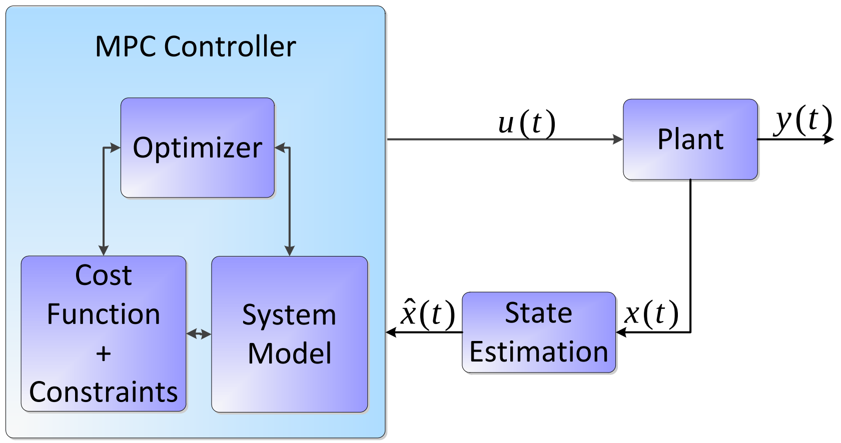


Figure 2.2: Model Predictive Controller block diagram

of the control strategy is applied to the plant and the plant state is measured again to be used as the initial state for the next time step. This *feedback* of the measurement information to the optimizer adds *robustness* to the control [35]. The plant state is sampled again and the whole process is repeated again with the newly acquired states. The prediction time window $[t + 0T, t + 1T, ..., t + pT]$ shifts forward at every time step (reason why MPC is also known as ***Receding Horizon Control*** [36]). Other names like ***Rolling-Horizon Planning***, ***Dynamic Matrix Control (DMC)*** [37] and ***Generalised Predictive Control (GPC)*** [38] have also been used for MPC.

MPC is a growing field and attracts many researchers around the world. Figure 2.4 depicts the number of publications available in IEEE database whose abstract or title contains the term “Model Predictive Control”. It is evident from the Figure 2.4 the number of research publications have been increased during this period, from only 2 publications in 1985 to 654 publications in 2010. During the period $1985 - 2000$, we see a linear rise in research publications, but after 2000 the increase is exponential. This is due to the rapid development of faster computers and advanced algorithms. It should be noted that these are only the publications with the name “Model Predictive Control”. There are several publi-

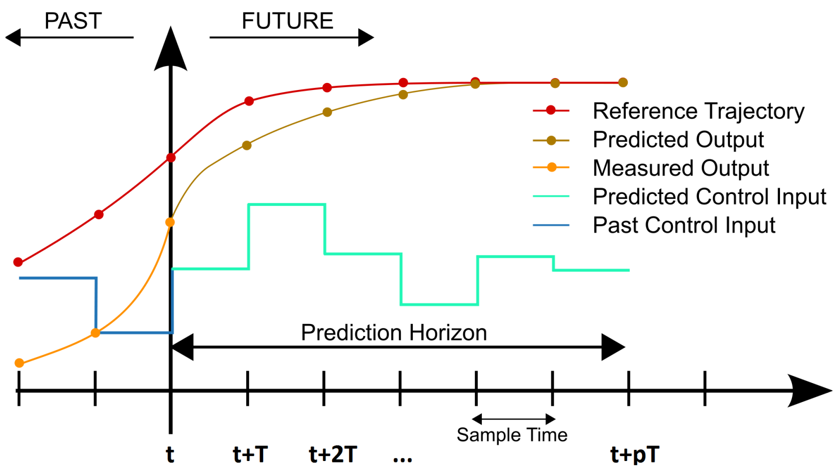


Figure 2.3: Model Predictive Control scheme. Modified form [39]

cations with various other names of the MPC. Also there are many publications other than IEEE. Looking at the fitted curve, one can predict that the publications will reach above 1000/year mark in 2015.

2.2.2 Modern Industrial MPC

MPC is extensively used in the oil refinery industry where it is implemented in the form of a hierarchy (Figure 2.5). The control structure is a four layered structure. Top layer is the ***Plant Wide Optimizer*** which finds the optimal settings for each individual unit in an industrial plant. The ***Local Economic Optimizer*** [40] calculates the steady-state operating points for a particular unit. These steady state points are sent to Dynamic constraint control layer where the intelligent MPC exists. The ***Dynamic Constraint Control*** (MPC) must move the unit from one set point to another in minimum amount of time without violating constraints [40]. MPC commands are then sent to the ***Basic Dynamic Control*** which contains low-level control systems like PID controller to command individual actuators. For further information on industrial predictive control practices, the reader is referred to an excellent survey by Qin [30].

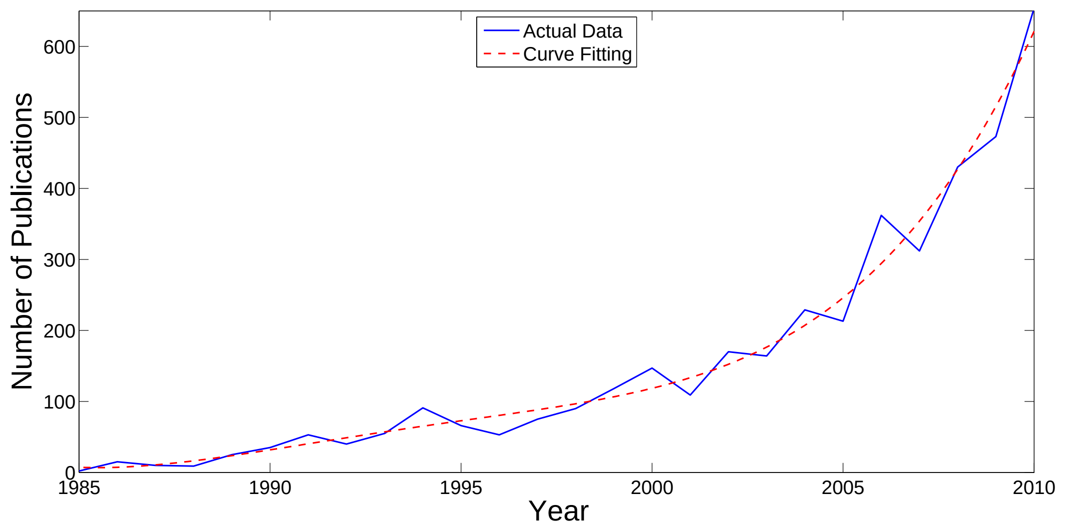


Figure 2.4: Model Predictive Control research trend based on number of publications of the IEEE over the period $1985 - 2010$

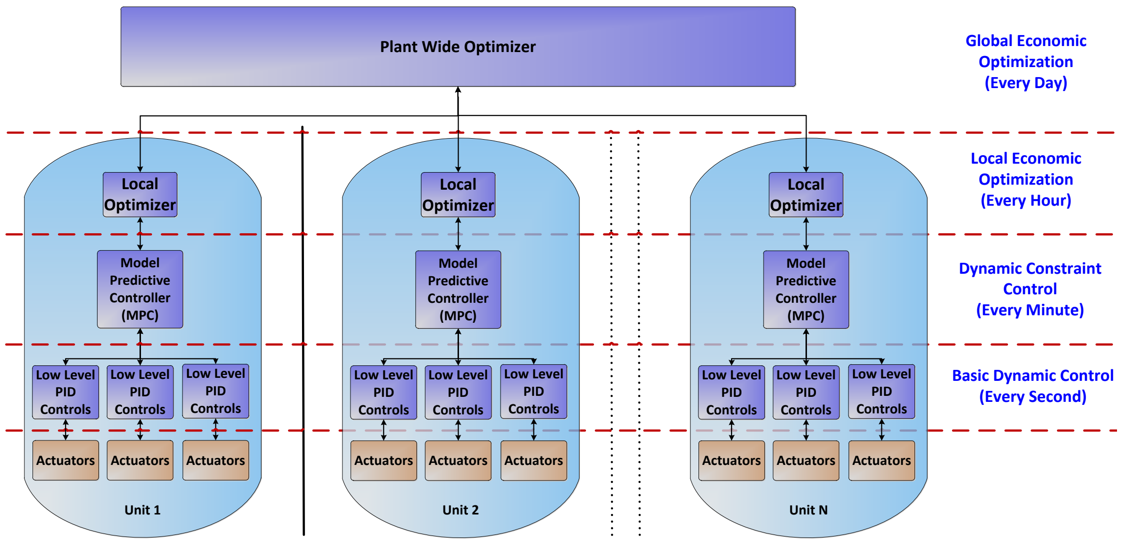

Plant Wide Optimizer
Global Economic Optimization (Every Day)
Local Optimizer
Local Economic Optimization (Every Hour)
Model Predictive Controller (MPC)
Dynamic Constraint Control (Every Minute)
Low Level PID Controls
Basic Dynamic Control (Every Second)
Actuators
Unit 1
Unit 2
Unit N

Figure 2.5: Hierarchy of control system functions in refinery industry.

2.2.3 Explicit MPC

Online execution of the MPC requires substantial computation effort. Such an implementation is easy for slow processes like oil refineries, but may be hard for faster dynamical systems like automotive systems. There is a variant of MPC where optimization is computed offline and results are stored for each possible value of the states $x(t)$ in the form of a look-up table. A piece-wise linear feedback control law $f(x)$ is determined which generates the optimal control input $u(k) = f(x(k))$ such as shown in Figure 2.7. So online effort is reduced to finding the appropriate control solution from a lookup table [41]. Such solutions exists for linear systems with quadratic cost function which marks a limitation for this approach. The optimization problem is formulated as a multi-parametric program by treating the current state $x(t)$ as a parameter. Solving this multi-parametric program yields an explicit solution, i.e. an optimal look-up table, for the MPC problem. This is a suboptimal optimal solution for the considered problem and each of the region in the resulting solution can be treated as a separate LQR. Thus time-consuming computation is done offline [42]. The benefits of Explicit MPC can be summarized as [43]:

- Explicit MPC can be implemented at higher sampling rates (due to less computation effort required)
- Explicit MPC can be implemented on inexpensive hardware with fixed-point arithmetic
- Reliability of the Explicit MPC can be verified since we have solutions for all possible states and inputs (within a bounded region). Thus it can be used for safety-critical applications.

Numbers of the explit affine regions and consequently the complexity of the explicit solution increases exponentially with an increase in the number of the states and the control inputs. Other trade-off for explicit solution are the offline pre-calculation overheads can be

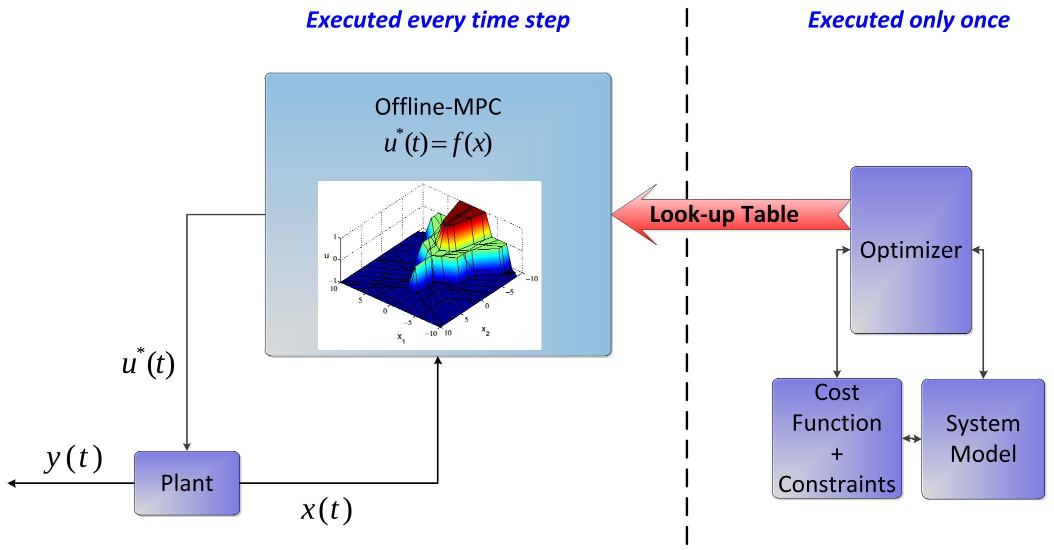
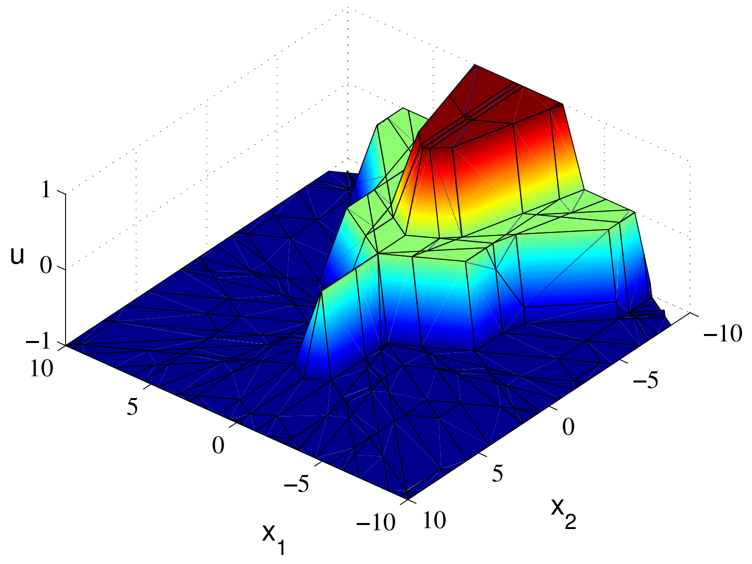


Figure 2.6: MPC-Explicit offline solution.

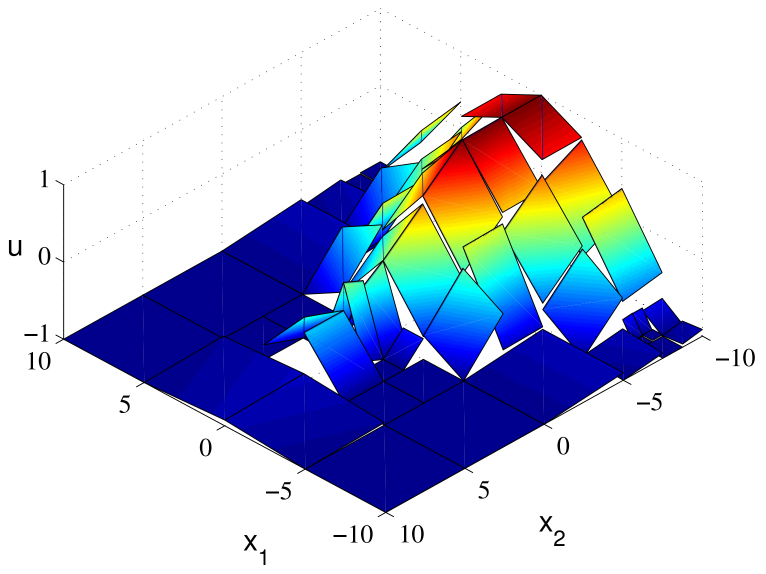
significant, and this method can only be applied for regulation only to specified setpoint where partition has been calculated in advance [44]. Also explicit MPC cannot be applied to plants with time-varying dynamics/cost function/constraints [45]. Researchers are working to reduce the complexity of explicit solutions [46, 47, 47, 48]. A user friendly and free Multi-Parametric Toolbox (MPT) for MATLAB [49] is also available which is helpful in design, analysis and deployment of optimal control systems and more specifically the explicit MPC systems.

2.3 MPC for Trajectory Tracking

The purpose of trajectory tracking is to minimize the deviation between a vehicle's traveled path and the desired path. PID controllers have been widely used for trajectory tracking of industrial robotic systems. Many other approaches for trajectory tracking of ground, underwater, water surface, aerial and space borne vehicles have been proposed. Divelbiss [51] used time-varying LQR approach for trajectory tracking of a car-trailer system. Wen [52]



(a)



(b)

Figure 2.7: (a) 277 region explicit MPC control law obtained for a an example in [50] by using MATLAB based MPT toolbox. (b) Simplified 55 region explicit MPC control law for the same application.

used iterative model predictive control for a nonholonomic vehicle's trajectory tracking. He used the gradient based method without constraints and proved that the online computation load is reduced significantly if no constraints are present.

Tsiotras [53] proposed a linear controller for a satellite's trajectory tracking by utilizing a non-quadratic Lyapunov function. Stability of the controller was proved by using a quadratic plus a logarithmic term.

Gordon [54] considered the problem involving a constrained submarine in order to explore the practicalities of NMPC implementation. An online gradient optimizer followed by an Extended Kalman Filter state estimator was used to control a continuous time submarine system while the controller embedded a discretized nonlinear model. Gordon also explored the effect of tuning matrices on optimization process.

Kim [55] investigated the feasibility of Nonlinear Model Predictive Tracking Control (NMPTC) for autonomous helicopters. NMPTC algorithm was formulated for planning paths under input and state constraints, and implemented online using a gradient-descent method. No obstacle avoidance was considered in this case.

Eklund [56] proposed a supervisory controller for pursuit and evasion of two fixed-wing autonomous aircrafts. NMPTC was used for real-time trajectory tracking of an evader as well as a pursuer aircraft. The NMPTC controller was then integrated in a UAV which participated in the Pursuit Evasion Games (PEGs) against a US Air Force Pilot operated F-15 aircraft. NMPTC controller was used as a high level trajectory generation layer and the low level control was performed by the aircraft's autopilot interface. Simulation/field test results showed that encoding of the PEGs into the cost function proved successful and the behavior of the NMPTC controlled aircraft was similar to what a pilot is taught to do in a flight school for that situation.

Fahimi [57] designed a Nonlinear Model Predictive Control law for controlling *multiple* autonomous surface vessels in arbitrary formations and in environment containing obstacles. A decentralized leader-follower approach was used for the vessel tracking and the effectiveness of the developed controller was tested in the presence of model uncertainties and external disturbances. Vougioukas [58, 59] used NMPC for precision guidance of agricultural tractors to assist farmers in their day to day activities. He further demonstrated through simulations the effect of different horizon lengths on the tracking performance of autonomous tractors.

Yoon [60] used model predictive approach for obstacle avoidance of car like unmanned ground vehicle (UGV). Two kind of potential functions were applied for obstacle avoidance, distance based and Modified Parallax. Simulation results proved that both methods worked equally well in simple environments but in complex environments the vehicle equipped with MP method produces better results in terms of tracking performance and computation time.

Chapter 3

Methods

This chapter focuses on the development of the controller and model formulations which are used in the rest of this thesis.

3.1 Work Methodology

In order to apply control to vehicle modeling in the CarSim environment, several stages of development were required. First vehicle dynamics were analyzed in order to develop an accurate vehicle model. This developed model was validated in the Simulink environment against a realistic CarSim model. Once this was accomplished, NMPC controller was developed in Matlab and tested with CarSim model.

3.2 Vehicle Modeling

The main requirement in any model predictive control (MPC) application is the plant model. MPC optimizer uses this model to predict plant future behavior and plan an optimized trajectory. At each control interval an MPC algorithm attempts to optimize future plant behavior by computing a sequence of future manipulated variable adjustments. Conventional MPC schemes use linear models to predict dynamic behavior of plant. Linear models can be inadequate in modeling highly nonlinear plants, such as a nonholonomic vehicle [61].

When dynamics can not be accurately modeled by linear models, nonlinear models may be essential to capture the full dynamics of the plant to be controlled. This section provides an overview of nonlinear and nonholonomic systems. With this framework, the mathematical modeling of the vehicle model is validated against a fully nonlinear CarSim vehicle model.

A *dynamical system* is one in which the states are undergoing changes as the time progresses. In this case the states X evolve and then Y can be described by differential equations (DE) or partial differential equations (PDE). A *linear* syatem is one in which input-output relationship is a linear map. For example, if input $x_1(t)$ produces response $y_1(t)$, and input $x_2(t)$ produces response $y_2(t)$, then the input $a_1x_1(t) + a_2x_2(t)$ (summed and scaled) produces the scaled and summed response $a_1y_1(t) + a_2y_2(t)$. Now if we apply input $x(t)$ to a system at time t, the output is $y(t)$. *Time invariance* means an input applied T seconds form now ($x(t-T)$) will produce identical output except with a time delay of T seconds ($y(t-T)$). The standard linear time invariant (LTI) system is given by the form:

$$\dot{x} = Ax + Bu$$

where states (x) and controls (u) form a linear combination but this is a very stringent requirement and many real life systems can not be accurately described by such models.

3.2.1 Nonlinear Systems

A nonlinear system is a system where the variables to be solved for cannot be written as a linear combination of independent components. The most general nonlinear system is described by the DE:

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

where the RHS is a function of both states, controls and time. These systems can become complicated and can not always be explicitly solved. This thesis deals with such a nonlinear system whose dynamics can be represented in the *state space* form. The state space representation models dynamic of the plant as a set of differential equations in a set of variables

known as *state variables*. *State variables* or in short *states* are a set of variables that fully describe the system's response to any given set of inputs. According to Derek [62]
"A mathematical description of the system in terms of a minimum set of variables $x_i(t), i = 1, ..., n$, together with knowledge of those variables at an initial time t_0 and the systems inputs for time $t \geq t_0$, are sufficient to predict the future system state and outputs for all time $t \geq t_0$."

State Equations

When multiple states and controls are present in a system, the general nonlinear equations of dynamical system are expressed in the form of n coupled differential equations known as *state equations*. Suppose we have n state variables $x_1(t), ..., x_n(t)$ and r system inputs $u_1(t), ..., u_r(t)$. In this case we should have n state equations:

$$\begin{aligned} \dot{x_1} &= f_1(x, u, t) \\ \dot{x_2} &= f_2(x, u, t) \\ \vdots &= \vdots \\ \dot{x_n} &= f_n(x, u, t) \end{aligned} \tag{3.1}$$

where $\dot{x_i} = dx_i/dt$ and $f_i(x, u, t)$ with $(i = 1, ..., n)$ is nonlinear function of states, inputs and time. These state equations are written in vector form:

$$\begin{aligned} \dot{\mathbf{x}} &= f(\mathbf{x}, \mathbf{u}, t) \\ \mathbf{y} &= h(\mathbf{x}, \mathbf{u}, t) \end{aligned} \tag{3.2}$$

where $\mathbf{x} = [x_1(t), x_2(t), ..., x_n]^T$, $\mathbf{u} = [u_1(t), u_2(t), ..., u_n]^T$ and $f(\mathbf{x}, \mathbf{u}, t)$ is a vector of $f_i(x, u, t)$ with $(i = 1, ..., n)$. The output vector is $\mathbf{y} = [y_1(t), y_2(t), ..., y_m(t)]^T$. Often, the outputs can be described as linear combination of the states. In this case $\mathbf{y} = C\mathbf{x}$ where C is a $m \times n$ filter matrix which limits output to states of interest only (Figure 3.1).

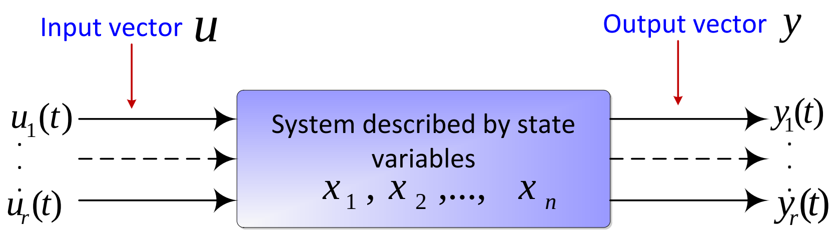


Figure 3.1: Block diagram of a state space system

The system of Eq. 3.2 models most physical systems encountered in engineering practice including linear systems of the form:

$$\begin{aligned}\dot{\mathbf{x}} &= A\mathbf{x} + B\mathbf{u} \\ \mathbf{y} &= C\mathbf{x} + D\mathbf{u}\end{aligned} \tag{3.3}$$

which are obtained from Equation 3.2 when $f(\mathbf{x}, \mathbf{u}, t)$ is a linear combination of $\mathbf{x}$ and $\mathbf{u}$. Linear control theory is a well developed field with many analysis methods of linear control systems [63]. However, most systems are nonlinear and we have to make following assumptions to apply linear control theory:

- the system is assumed linear around a specific operation point, $\hat{x}$, and
- the system will remain in a small region around that operation point during operation.

Both of these conditions must be satisfied, otherwise a nonlinear control strategy is needed. In this thesis we limit our attention to *time invariant* systems.

3.2.2 Nonholonomic Systems

"Nonholonomic systems are, roughly speaking, mechanical systems with constraints on their velocity that are not derivable from position constraints. They arise, for instance, in mechanical systems that have rolling contact (for example, the rolling of wheels without slipping) or certain kinds of sliding contact (such as the sliding of skates)." [64]

In a nonholonomic the directions of motion are limited so that although all of the spaces are reachable, maneuvers are required to reach many states and these become path dependent. States of a nonholonimc depend on the path taken to achieve it. A car is classic example of nonholonomic system. The vehicle has three degrees of freedom i.e., x,y position and its orientation. But it has only two controllable degrees of freedom i.e., acceleration and front tire steering angle, hence an under actuated system. A car must travel in the same direction as its heading and cannot move in any other direction (unless there is some slipping). Every path cannot be achieved by a nonholonomic vehicle. This nonholonimity of a vehicle makes turning and parallel parking a challenging maneuver. Since a nonholonomic system cannot move in arbitrary directions in its configuration space [65].

Parking a van is a great example of the maneuvers required by a nonholonomic system. This is illustrated in Figure 3.2. A van driver wants to parallel park between two cars and a limited space is available for parking. The driver cannot move into the parking space perpendicularly because the van can only roll in the direction of its heading (or $180\,\mathrm{deg}$ if he is reversing). This makes it a challenging maneuver and the driver has to move the van back and forth with different steering angles in order to optimally parallel park the van between the two cars.

3.2.3 Mathematical Modeling

Lateral vehicle models describe the dynamics along the width of the vehicle which can be used to track coordinates of the vehicle in an inertial frame. Longitudinal vehicle models describe dynamics along the length of the vehicle which is used to track the velocity of the vehicle. Lateral vehicle models were studied as early as 1950s by Segel [66] and Kasselmann [67]. Model Predictive Control requires an analytical and accurate dynamic mathematical model of the system to be controlled. Our current implementation uses a nonlinear model of vehicle dynamics to predict the future trajectory of system states. A lateral model for vehicle motion can be developed by using some assumptions. Wang [68] used

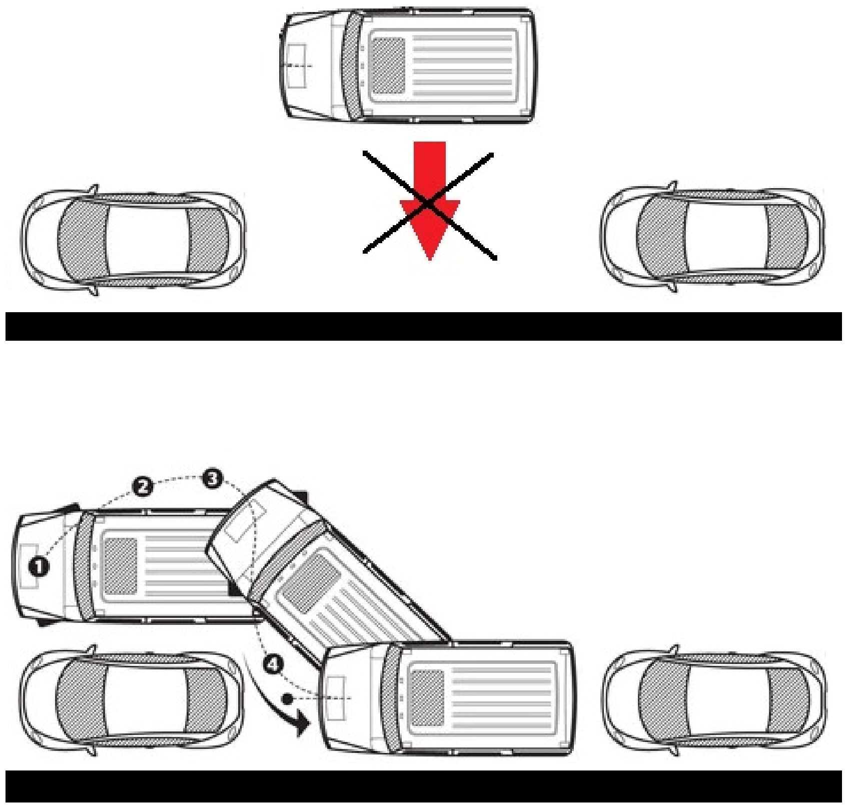


Figure 3.2: Rolling without slipping nonholonomic constraint of a vehicle. Top: Vehicle cannot parallel park directly due to nonholonomic constraint. Bottom: Driver has to plan a trajectory in order to parallel park the vehicle in the limited space.

a bicycle model to represent a four wheel vehicle. In a bicycle model, the front left and front right wheels are lumped into one wheel (Point A in Figure 3.4). Therefore the steering angle of front left and front right wheel become δ_f. Similarly the rear left and rear right wheels are lumped into one wheel (Point B in Figure 3.4). The rear wheel is considered fixed to the vehicle frame and does not steer. Point C is the center of gravity of the vehicle, the distance $\overline{CA}$ is l_f and distance $\overline{BC}$ is l_r, the vehicle wheel base is $l = l_f + l_r$. The vehicle is moving in a two dimensional plane and (X, Y) are the coordinates of the vehicle in this frame. Vehicle orientation is encoded in terms of roll, pitch and yaw angles, each of which is measured against three principal axes of the vehicle X, Y and Z respectively. An illustration of the three angles is given in Figure. 3.3. It is assumed that small friction

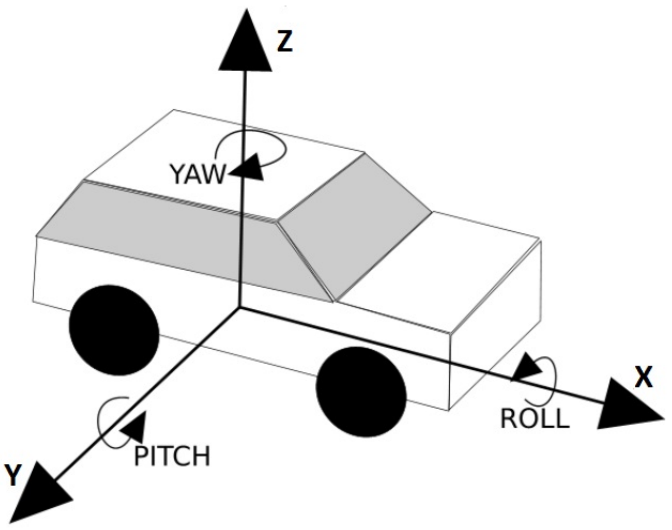


Figure 3.3: Body Frame Axis System: x, y and z axis of the vehicles body frame, with rotational degrees of freedom, pitch, roll and yaw. Modified from Melonee [69]

coefficients exist between the tire and the road, and weight transfer between front and rear end of the vehicle is negligible (low CG) which means that body roll and pitch behavior can be neglected. The yaw angle ψ is orientation of vehicle in X-Y frame and is measured with respect to global X axis. The velocity of the vehicle v makes an angle β (vehicle slip angle) with the longitudinal axis of the vehicle. v_x is the longitudinal velocity and $\dot{\psi}$ is the yaw rate of the vehicle. F_{yf} and F_{yr} are the lateral forces on front and rear tires respectively. The overall force at point A and B is $2F_{yf}$ and $2F_{yr}$ because each of these point represents two tire forces. It is the most important model used for lateral vehicle dynamics studies. In principle, any vehicle model can be described as a bicycle model [70] [71]. The lateral vehicle dynamics can be modeled by applying Newton's second law of motion along the Y-axis.

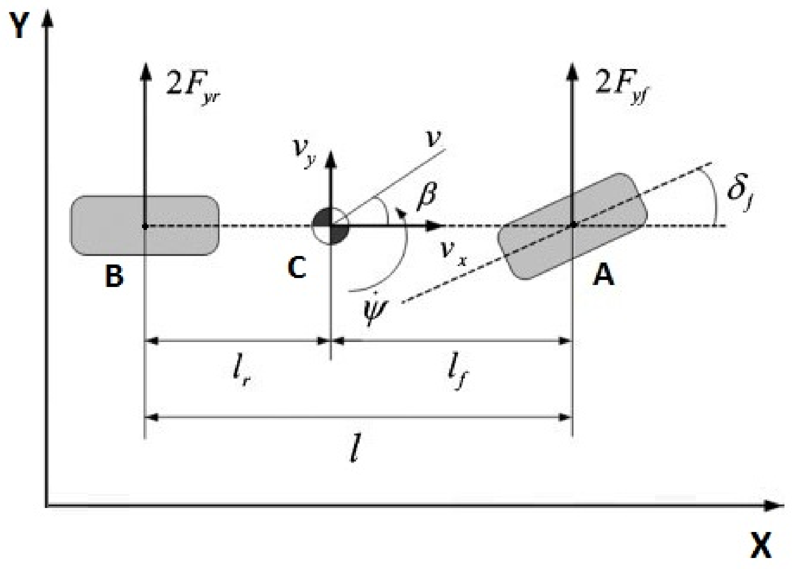


Figure 3.4: Simplified Bicycle Model [72]

$$\sum(Forces\ acting\ along\ Y\ axis) = ma_y$$

$$2F_{yf} + 2F_{yr} + F_{bank} = ma_y \tag{3.4}$$

where F_{bank} is the force due to the banking angle of the road, m is the mass of the vehicle and $a_y = (\frac{d^2y}{dt^2})$ is the acceleration of center of gravity of the vehicle along Y-axis. The force due to banking angle of the road is assumed zero in this case. A detailed model considering banking angles of the road can be found in [73]. The lateral acceleration a_y is composed of two terms which are: the acceleration due to vehicle motion along y-axis $\ddot{y}$ and the centripetal acceleration $v_x\dot{\psi}$.

$$a_y = \ddot{y} + v_x\dot{\psi}$$

So Eq. 3.4 becomes

$$2F_{yf} + 2F_{yr} = m(\ddot{y} + v_x\dot{\psi}) \tag{3.5}$$

The vehicle side slip angle is written as $\beta = \frac{\dot{y}}{v_x}$. Eq. 3.5 now becomes

$$2F_{yf} + 2F_{yr} = mv_x(\dot{\beta} + \dot{\psi}) \tag{3.6}$$

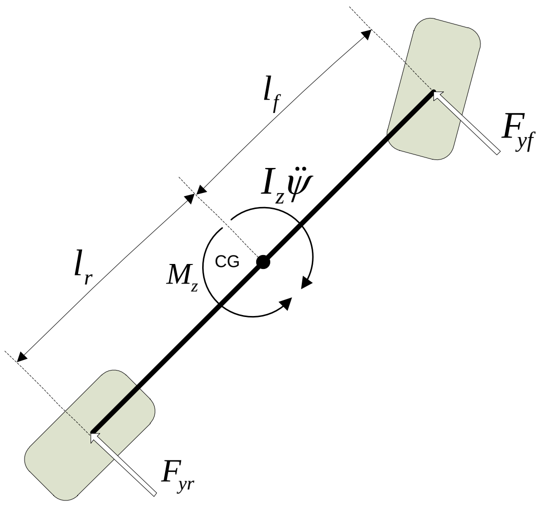


Figure 3.5: Vehicle momentum

Now consider the Figure3.5 where it is assumed that a moment M_z acts on the vehicle. This momentum is created by wind or road roughness. Also an inertial torque $I_z\ddot{\psi}$ acts on the vehicle. The lateral forces on front and rear tires also produce moment ($l_f F_{yf}$ and $l_r F_{yr}$). Total moment at center of gravity (CG) yields:

$$\sum M_{cg} = I_z\ddot{\psi} - M_z - 2F_{yf}l_f + 2F_{yr}l_r = 0 \tag{3.7}$$

where I_z is yaw moment of inertia. Assuming the torques due to wind M_z is zero:

$$I_z\ddot{\psi} = 2F_y l_f + 2F_y l_r \tag{3.8}$$

A further discussion on modeling of vehicle momentum forces is give by Andrzejewski [74]

Tire Model

Vehicle lateral dynamics and state estimations are highly dependent on the tire characteristics. The primary forces during lateral maneuvering, acceleration, and braking are generated by tires as a function of the driver input. The linear analysis of a tire model commonly considers constant tire side force coefficients at small force output range. The linear tire model doesn't consider longitudinal tire forces due to the complex interactions between lateral and longitudinal tire forces. Thus, linear tire model is suitable for analyzing a stable vehicle behavior under the assumption of small steering and acceleration [75]. Accurate tire modeling is quite complex, therefore numerous approximate methods have been developed for tire modeling. Svedenius [76] has provided a thorough survey on type of tire models used. A relatively simple but widely used tire model known as Pacejka [77] tire model is used by Borrelli [78] and Falcone [79], [80] for trajectory generation of a vehicle using MPC. The Pacejka tire model is a semi-empirical static model which involves the longitudinal and lateral forces acting on a tire.

$$\begin{aligned} F_c &= f_c(\alpha, s, \mu, F_z) \\ F_l &= f_l(\alpha, s, \mu, F_z) \end{aligned} \tag{3.9}$$

where F_c is lateral or *cornering* tire force and F_l is longitudinal tire force. Each of the force is a complex function (f_c, f_l) of tire slip angle α, longitudinal slip ratio s, friction μ and normal force F_z. Slip angle α is the angle between a rolling wheel's actual direction of travel and the direction towards which it is pointing (Figure 3.6). Slip ratio s is the ratio of rotational velocity of a tire to its translational longitudinal velocity. Mathematically:

$$s = \begin{cases} \frac{r_w \omega}{v_l}, & \text{if } v_l > r_w \omega \\ 1 - \frac{v_l}{r_w \omega}, & \text{if } v_l < r_w \omega \end{cases} \tag{3.10}$$

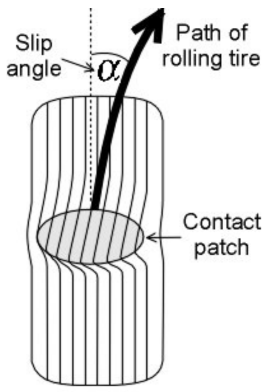


Figure 3.6: Tire slip angle [81]

where v_l is the tire longitudinal velocity, r_w is the wheel radius and ω is the radial velocity of the tire.

Our main interest is to model the lateral tire forces F_{yf} and F_{yr}. In some literature, lateral force is considered linearly proportional to the slip angle of tire. This is due to the experimental data obtained from the vehicle tires as shown in Figure 3.7. It can be seen that at low slip angles (which is found to be $5\,\mathrm{deg}$ or less by Gillespie [82] and Stone [83]), the lateral forces F_y vary proportionally to the slip angle α. The transitional region is the region where maximum tire forces exist (usually $4\,\mathrm{deg}-6\,\mathrm{deg}$). Typical Anti-Lock Braking Systems (ABS) try to keep slip angle with-in this maximum force region to obtain maximum possible braking force. In the frictional region, the tire starts skidding and it can not provide as much lateral force. *Cornering Stiffness* $C_{f,r}$ is defined as the ratio between F_y and α. Subscript f and r denote front and rear tire respectively. SAE defines cornering stiffness as modulus of the slope which means it can never be negative [84]. Cornering stiffness depends upon the manufacturing features of the tire e.g., radius, width, tread, inflation pressure etc. In this research, this linear model of tire lateral dynamics will be used since it is very simple (hence suitable for online application) as compared to complex tire models and at the same time sufficiently accurate. Specifically:

$$F_{yf,yr} = C_{f,r}\alpha_{f,r} \tag{3.11}$$

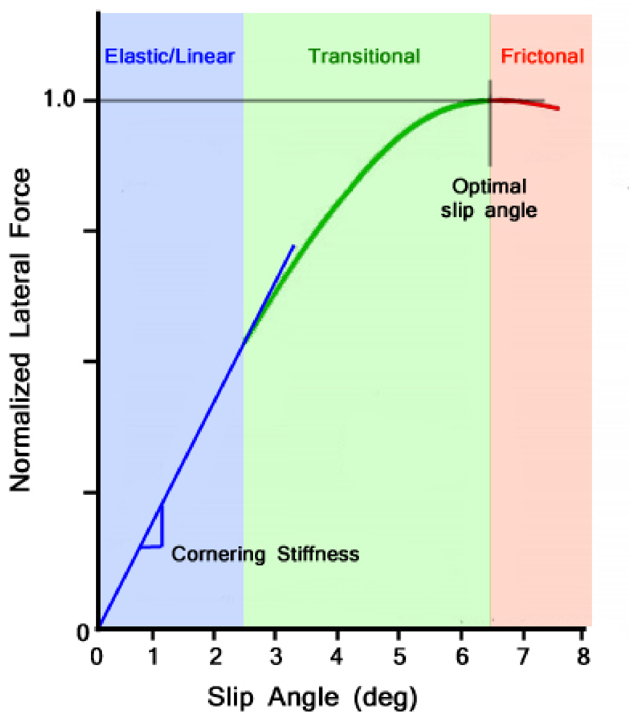


Figure 3.7: Typical slip angle vs. normal load curve obtained from experimental data [85]

From Figure 3.8, the front tire slip angle can be written as:

$$\alpha_f = \delta_f - \theta_{Vf} \tag{3.12}$$

where θ_{Vf} is the angle of velocity vector with longitudinal axis of the vehicle. δ_f is front tire steering angle. Similarly rear tire slip angle can be given as

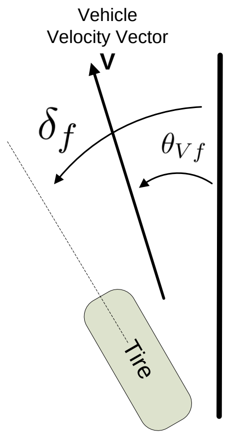


Figure 3.8: Vehicle velocity vector

$$\alpha_r = -\theta_{Vr} \tag{3.13}$$

Using Equations 3.11,3.12 and 3.13 the front and rear lateral tire forces now become:

$$\begin{aligned} F_{yf} &= C_f(\delta_f - \theta_{Vf}) \\ F_{yr} &= C_r(-\theta_{Vr}) \end{aligned} \tag{3.14}$$

The velocity angles of front and rear tires can be written by the following relation:

$$\begin{aligned} \theta_{Vf} &= \beta + \frac{l_f \dot{\psi}}{v_x} \\ \theta_{Vr} &= \beta - \frac{l_f \dot{\psi}}{v_x} \end{aligned} \tag{3.15}$$

Substituting above set of Equations in Equation 3.14 gives us the final relationship of tire longitudinal forces:

$$\begin{aligned} F_{yf} &= C_f(\delta_f - (\beta + \frac{l_f \dot{\psi}}{v_x})) \\ F_{yr} &= C_r(-(\beta - \frac{l_r \dot{\psi}}{v_x})) \end{aligned} \tag{3.16}$$

Next kinematic model of the vehicle motion is developed which tracks motion of the vehicle in earth fixed coordinate system without considering the forces causing the motion. In order to do so, geometric transformation equations in 2-D must be considered.

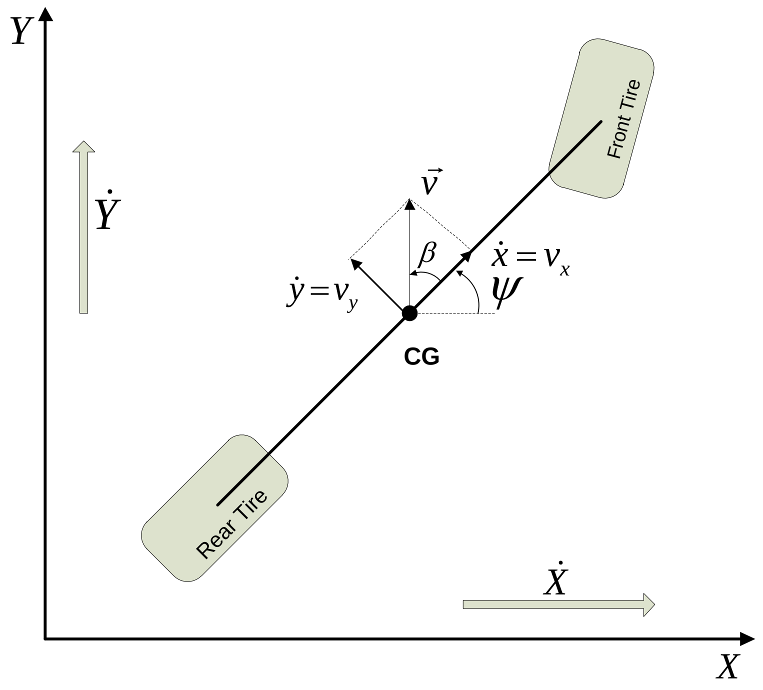


Figure 3.9: Kinematic motion of the vehicle

In Figure 3.9 a vehicle moves in the earth-fixed X-Y inertial frame of reference. As the vehicle turns, the velocity vector v makes an angle β (side slip angle) with the vehicle's longitudinal axis. Orientation of the vehicle in X-Y frame is ψ i.e., its yaw. It is assumed that the slip angle of both the tires is zero for simplicity. The equations of motion can be described by the following geometric expressions:

$$\begin{aligned} \dot{X} &= v\cos(\psi+\beta) \\ \dot{Y} &= v\sin(\psi+\beta) \end{aligned} \tag{3.17}$$

where $(\psi+\beta)$ is the angle velocity vector v makes with global X-axis. Expanding the above set of equations by angle sum identities gives:

$$\begin{aligned}\dot{X} &= v\cos(\beta)\cos(\psi) - v\sin(\beta)\sin(\psi)\\ \dot{Y} &= v\cos(\beta)\sin(\psi) + v\sin(\beta)\cos(\psi)\end{aligned} \tag{3.18}$$

$\dot{X}$ and $\dot{Y}$ are the X and Y components of vehicle's velocity in earth-fixed frame. Such an equation with derivative is easy to implement since it can easily be written in state-space form 3.2. Now the vehicle's velocity vector can be written in the terms of vehicle's longitudinal velocity i.e., $v_x = v\cos\beta$. Equation 3.18 becomes:

$$\begin{aligned}\dot{X} &= v_x\cos(\psi) - v_x\tan(\beta)\sin(\psi)\\ \dot{Y} &= v_x\sin(\psi) + v_x\tan(\beta)\cos(\psi)\end{aligned} \tag{3.19}$$

State vector can be written as $\xi = [\beta\ \psi\ \dot{\psi}\ X\ Y]$, where

- $\beta =$ Vehicle side slip angle
- $\psi =$ Vehicle yaw angle
- $\dot{\psi} =$ Vehicle yaw rate
- $X =$ x-coordinate of vehicle's CG in inertial frame
- $Y =$ y-coordinate of vehicle's CG in inertial frame

and $u = [\delta_f]$ where δ_f is front tire steering angle.

Combining Equations 3.6, 3.8 and 3.19 gives the following vector form of dynamic model:

$$\begin{aligned}\dot{\xi} &= \mathbf{f_{cont}}(\xi(\mathbf{t}), \mathbf{u}(\mathbf{t}))\\ \eta &= \mathbf{C}\xi\end{aligned} \tag{3.20}$$

C is filter matrix which eliminates states not required for the tracking. In this case, the output η is set to track X and Y coordinates of the vehicle in inertial frame. C matrix becomes:

$$C = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

with this choice of C matrix the output states become

$$\eta = [X \;\; Y]^T$$

Detailed model equations can be found in Appendix B.

3.3 Model Validation

"Model verification and validation are essential parts of the model development process if models to be accepted and used to support decision making" [86].

One of the most important question to ask after model development is that Is this model valid? Especially when some assumptions have been made while developing the model, it becomes necessary to validate the model against a fully non-linear vehicle model. For the purpose of this research, *CarSim* (Figure 3.10) a commercial software package developed by Mechanical Simulations Corporation has been used [2]. It can simulate fully nonlinear and close to reality vehicle models in response to driver controls such as steering angle.

3.3.1 Validation Setup

Developed model (Equation 3.20) can be validated by applying sine, sawtooth or step input at the steering input. However it should be noted that the steering angle is being applied only to tires on the road and not to the steering wheel. At lower-level a steering system model can be used to model the relationship between steering wheel turning angle and actual wheel steer angle [87]. For validation, a setup similar to Figure 3.11 has been used. A test input steering angle δ_f is applied to both developed mathematical model and the CarSim model at

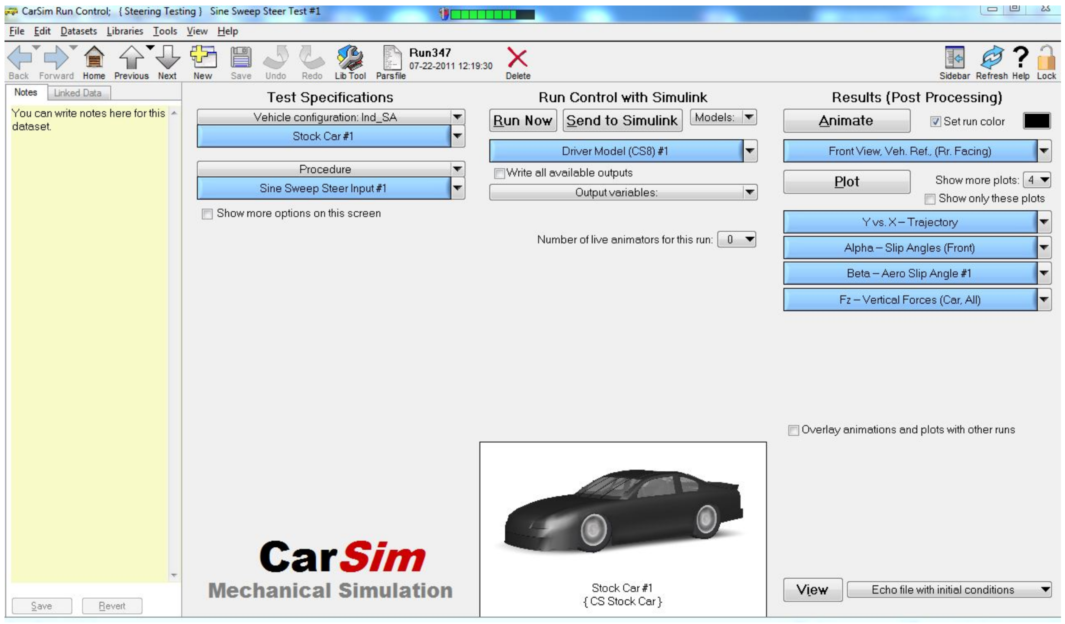

CarSim Run Control; { Steering Testing } Sine Sweep Steer Test #1
File Edit Datasets Libraries Tools View Help
Back Forward Home Previous Next New Save Undo Redo Lib Tool Parsfile
Run347
07-22-2011 12:19:30
Delete
Sidebar Refresh Help Lock
Notes Linked Data
You can write notes here for this dataset.
Save Revert
Test Specifications
Vehicle configuration: Ind_SA
Stock Car #1
Procedure
Sine Sweep Steer Input #1
Show more options on this screen
Run Control with Simulink
Run Now
Send to Simulink
Models:
Driver Model (CS8) #1
Write all available outputs
Output variables:
Number of live animators for this run: 0
Results (Post Processing)
Animate
Set run color
Front View, Veh. Ref., (Rr. Facing)
Plot
Show more plots: 4
Show only these plots
Y vs. X – Trajectory
Alpha – Slip Angles (Front)
Beta – Aero Slip Angle #1
Fz – Vertical Forces (Car, All)
Overlay animations and plots with other runs
CarSim
Mechanical Simulation
Stock Car #1
{ CS Stock Car }
View
Echo file with initial conditions

Figure 3.10: CarSim user interface

the same time. Output states from both the models are saves and plotted for further analysis and comparison. It should be noted that this is an open-loop test and there is no feedback of any sort to the input steering angle.

For model validation a test similar to Park[72] has been performed with a growing sinusoidal steering signal. An arbitrary car with specifications given in Table3.1 has been used.

3.3.2 Validation Results

The model was validated for different speeds and steering angles. Higher the speed, lower is the steering angle. First validation test was performed at the speed of 20 km/h and the results are presented in Figure 3.12. The bicycle model and the CarSim model side slip angle and yaw rate angle agree with each other at smaller steering angles ($< 20\ degrees$). However, the front tire slip angles are different between the two models. The bicycle model

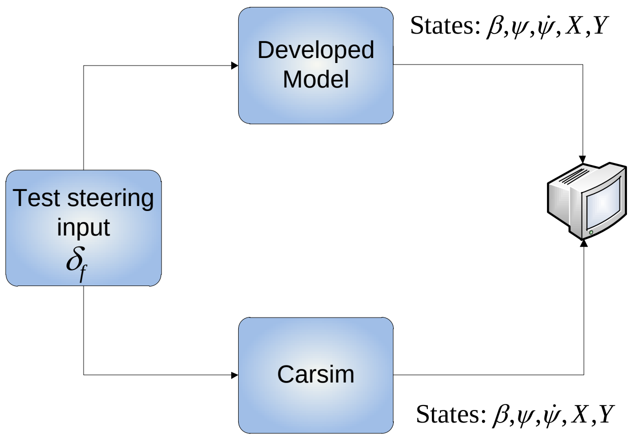

Developed Model
States: $\beta,\psi,\dot{\psi},X,Y$
Test steering input δ_f
Carsim
States: $\beta,\psi,\dot{\psi},X,Y$

Figure 3.11: Model validation structure

Table 3.1: Parameters of vehicle used for model validation

Parameter	Value
Vehicle Mass	$m = 1723\ kg$
Inertia	$I_z = 4175\ kg.m^2$
Vehicle Dimensions	$L = 4m, W = 1.988m$
Axle Length	$l_f = 1.232m, l_r = 1.468m$
Cornering stiffness	$C_f = 66900N/rad, C_r = 62700N/rad$

tire slip angle output is very smooth because it is produced from a linearized mathematical tire model. CarSim model left and right tire slip angles disagree from each other (Figure. 3.12d) demonstrating the turning effect in real cars. This fact is further investigated in the next subsection.

Second test was performed at $40\ km/h$ speed with a relatively small steering angle input (Figure 3.13a) as compared to previous test. Reason for decreasing steering angle is that the tire slip angle should remain small in order to operate closer to linear region. It can be seen from Figure 3.13d that the CarSim tire slip angles operate very close to that obtained from the bicycle model. Also CarSim left and right tire slip angles closely match

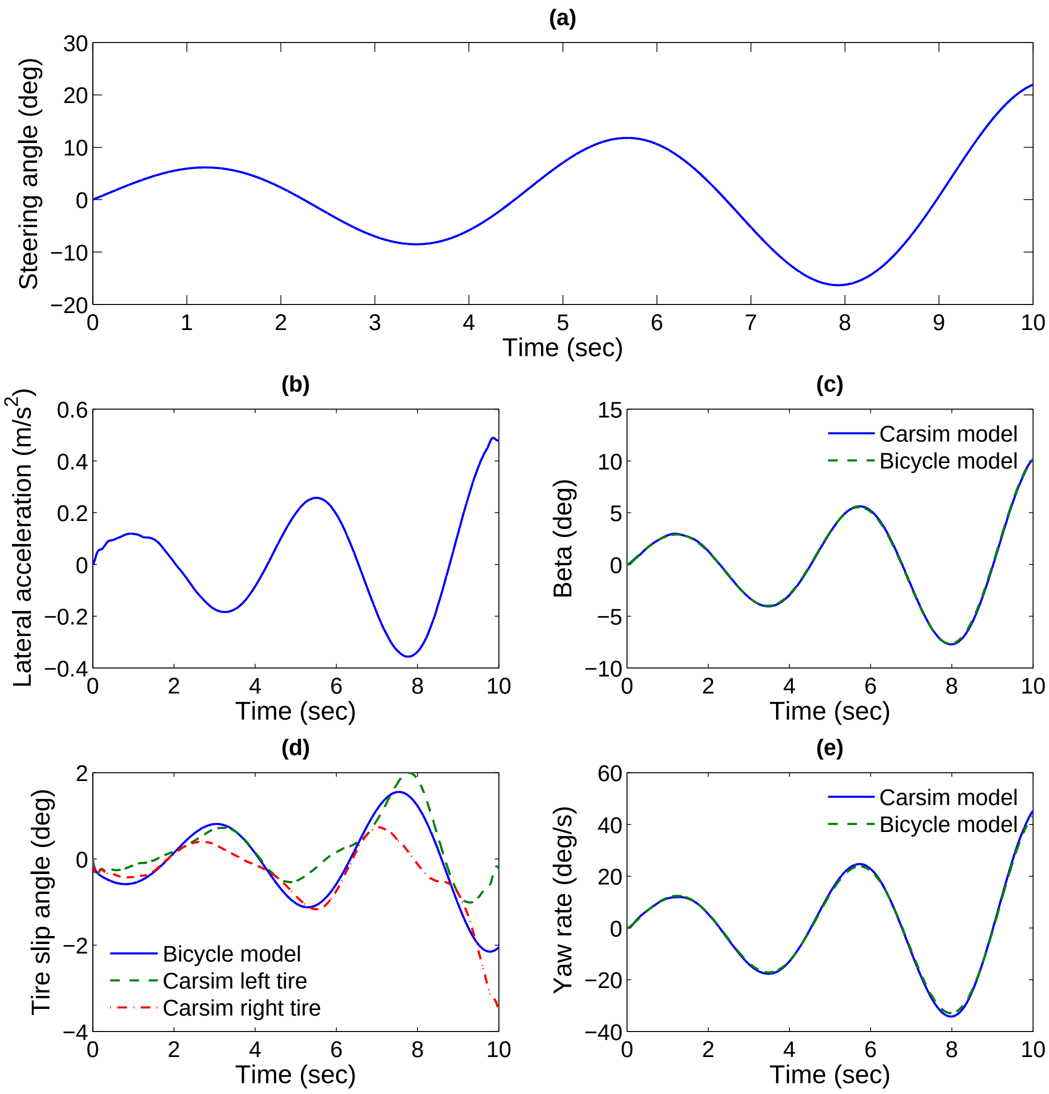


Figure 3.12: Model validation results at $v_x = 20km/h$ (a) Steering angle input applied to both bicycle model and CarSim model (b) Lateral acceleration CarSim model, (c) Vehicle side slip angles, (d) Front tire slip angles, (e) Vehicle yaw rate

each other since the steering angle and consequently load transfer is small. There is small variation in vehicle side slip angle (Figure 3.13c) and vehicle yaw rate (Figure 3.13e) due to speed increase.

Third test was performed at constant speed of 70 km/h and decreased steering angle (Figure 3.14a) input as compared to previous test. It was observed from these three tests that as the speed increased, CarSim model vehicle yaw rate and side slip angle disagreed more from the bicycle model but the CarSim tire slip angles agreed closely with the bicycle model tire slip angle (Refer to next section for an explanation of why this happens). Overall the model is considered usable since tire slip angles are not directly used in MPC formulation. At higher speeds ($> 40km/h$) and higher steering angles both model outputs disagree with each other so steering angle needs to be more limited.

Realistic Vehicle Model

The CarSim tire model is a nonlinear model that takes into consideration the interaction between the longitudinal and the cornering forces. For the purpose of this research, 225/60 R18 tire model has been used which is a realistic tire model available in the CarSim libraries. This model is defined in terms of look-up table (slip angle vs lateral tire force) and is plotted for different values of load in Figure 3.15. It can be seen from the plot that lower vertical load produces less lateral force. A higher value of vertical load moves the tire in higher curve of the plot. When the vertical load is not equal to one of the values in the lookup table, the lateral force values are calculated by interpolation of the two nearest plots. Further, it can be seen that the tire forces are linear in $0 - 1\,\mathrm{deg}$ slip angle region. At larger slip angles, the tire forces become increasingly non-linear functions of the slip angle.

From the validation results, it is seen that the vehicle tires show asymmetric behavior during turning. When the vehicle is at rest, there is a constant weight on each tire. Once the vehicle starts moving, more specifically turns, the weight of the vehicle on each of the individual tires changes. Figure 3.16 shows the tire lateral and longitudinal forces when the

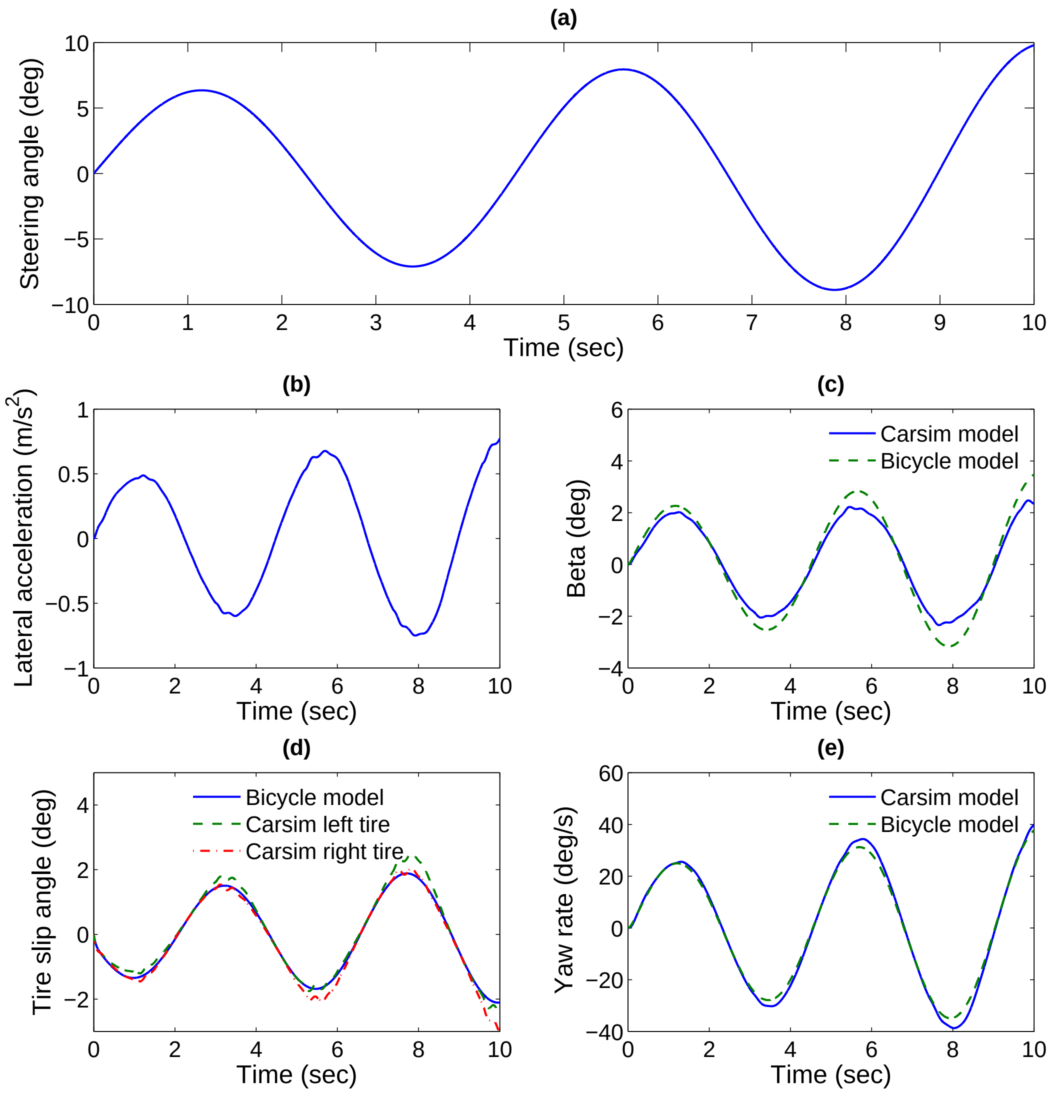


Figure 3.13: Model validation results at $v_x = 40km/h$ (a) Steering angle input applied to both bicycle model and CarSim model (b) Lateral acceleration CarSim model, (c) Vehicle side slip angles, (d) Front tire slip angles, (e) Vehicle yaw rate

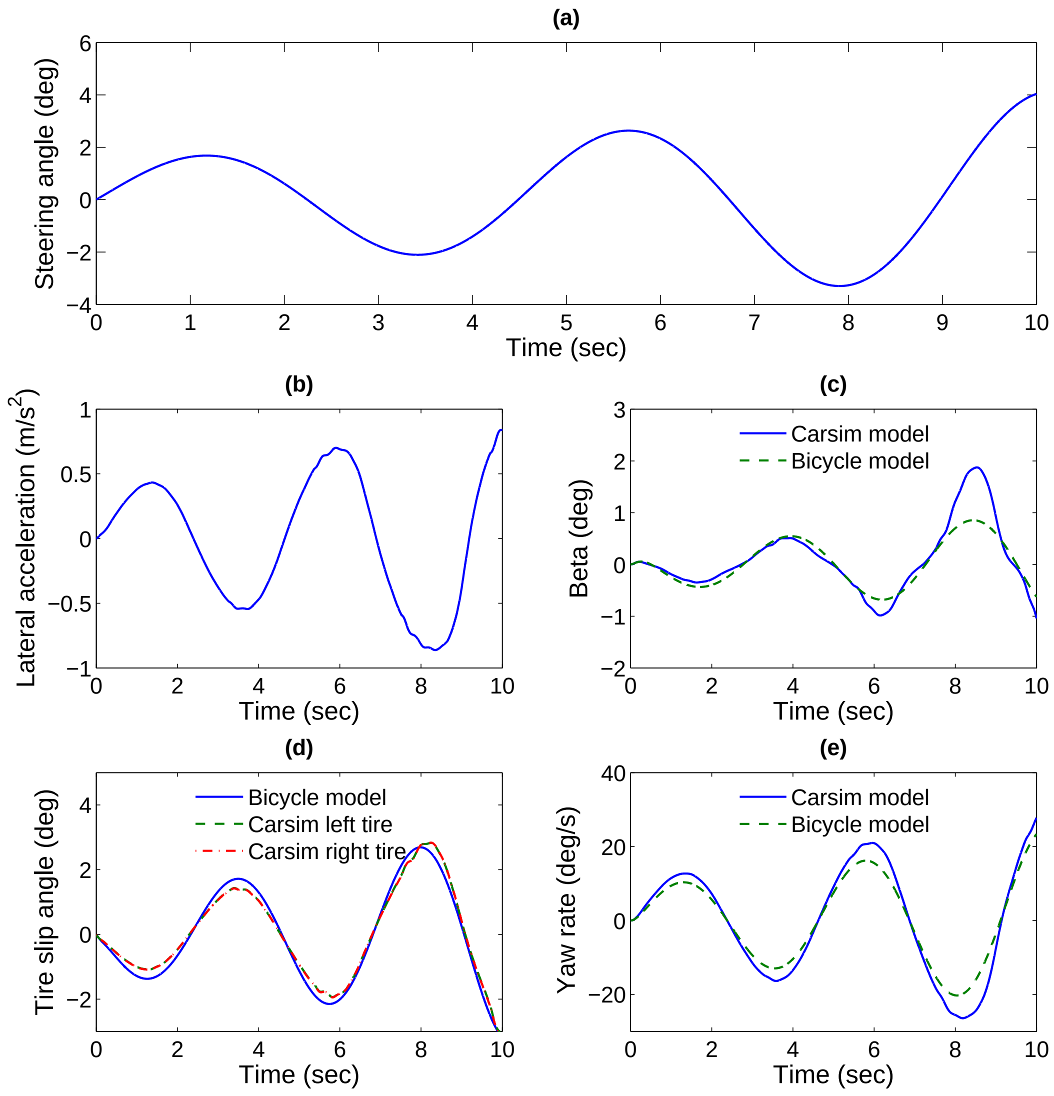


Figure 3.14: Model validation results at $v_x = 70km/h$ (a) Steering angle input applied to both bicycle model and CarSim model (b) Lateral acceleration CarSim model, (c) Vehicle side slip angles, (d) Front tire slip angles, (e) Vehicle yaw rate

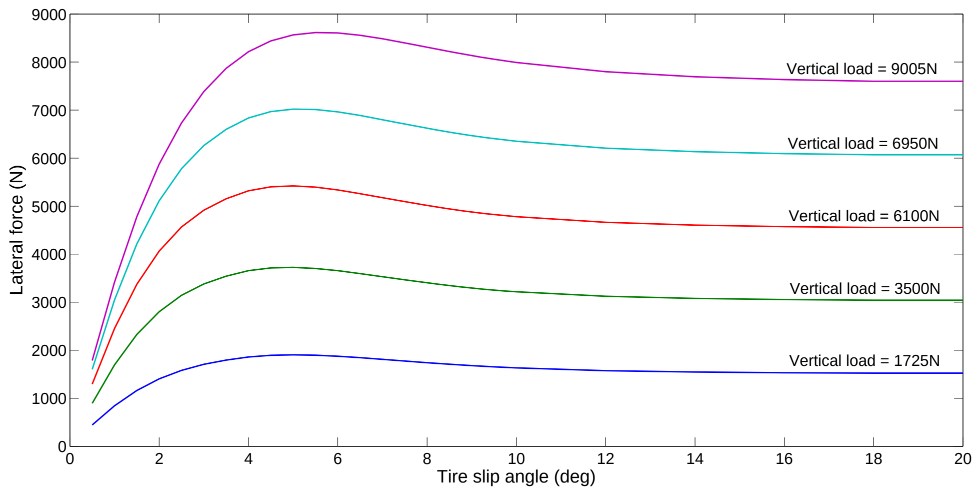
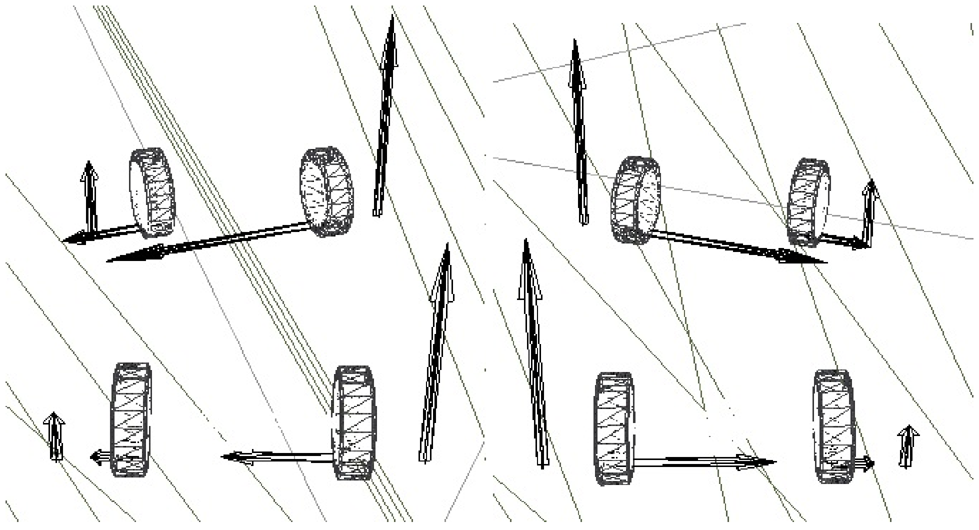


Figure 3.15: CarSim 225/60 R18 tire model



(a) Vehicle turning left (b) Vehicle turning right

Figure 3.16: Notice that while turning left, right tire lateral force is much larger than the left tire lateral force and vice versa

vehicle is turning. Turning to the left causes more right tire forces and turning to the right causes more left tire forces. Reason for this asymmetric tire lateral force in the CarSim model is the *Weight Transfer* (explained in next paragraph). It is worth noting here that the weight of the car does not change but transfers from one tire to another. According to Newton's first law:
"Every body persists in its state of being at rest or of moving uniformly straight forward, except insofar as it is compelled to change its state by force impressed" [88]
Tendency of a vehicle to move in a straight line is the inertia of the vehicle, it is due to this property of the vehicle's body that it tries to maintain its linear motion. While turning, road exerts a sideway force on the vehicle to divert it from its linear path and turn. The force that is exerted on the vehicle is centripetal force or lateral G force (as in Figure 3.17) it is always directed orthogonal to the velocity of the body, toward the instantaneous center of curvature of the path [89].

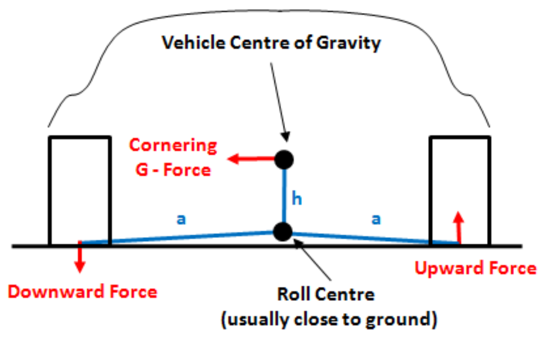

Vehicle Centre of Gravity
Cornering G - Force
h
a
a
Upward Force
Downward Force
Roll Centre (usually close to ground)

Figure 3.17: Sideways weight transfer [90]

In the Figure 3.17, cornering or lateral G-force acts on the center of gravity. h is the height of CG from roll center and a is the shortest distance between the roll center and the ire. The upward and downward weight transfer force in case of a right turn can be found as:

$$\begin{aligned} Left\ tire\ weight\ transfer\ downward\ force\ &= \\ G-Force\ \times\ Vehicle\ mass \times \frac{h}{a} && (3.21) \\ Right\ tire\ weight\ transfer\ upward\ force\ &= \\ G-Force\ \times\ Vehicle\ mass \times \frac{h}{a} && (3.22) \end{aligned}$$

The amount of weight transfer depends upon the vehicle's mass, speed, steering angle, center of gravity and the distance of center of gravity from the left and right tires (a in Figure 3.17) [91]. More mass results in more weight transfer so does a higher center of gravity (h in Figure 3.17). This is the reason why SUV's (vehicles having higher CG) are more prone to roll over. Increasing the track width (a+a) will decrease the weight transfer. That is why racing cars are wide and low. Reducing weight transfer provides more grip and sharp cornering. It is worth noting that the vehicle's springs, dampers and suspension have no effect on amount of steady state weight transfer but they effect the weight transition dynamics (overshoot, transition time, damping etc).

To elaborate why the CarSim left and right tire slip angles disagree at 20 km/h test but agree at 70 km test, a simple test was performed to approximate sideways weight transfer at these speeds. In the first test a step steering input of $22\deg$ magnitude (Figure 3.18a) was applied to the CarSim vehicle moving at 20 km/h and weight transfer was observed. Average weight transfer in this case was found to be around $30N$ (Refer to Figure 3.18d, 3.18e).

For second test a step steering input of $28\deg$ (3.19a) was applied to CarSim vehicle moving at a faster speed of 70 km/h. An average weight transfer of $450N$ was observed (Refer to Figure 3.19d, 3.19e). With these weight transfer results at different speeds Figure 3.12d and Figure 3.14d are reconsidered. In 20 km/h test (Figure 3.12) weight transfer was less but the left and right tire slip angles were found to be off whereas at 70 km/h test where weight transfer was large, the left and right tires slip angles agreed with each other. This interesting effect can be explained by looking at tire model (Figure 3.15). At

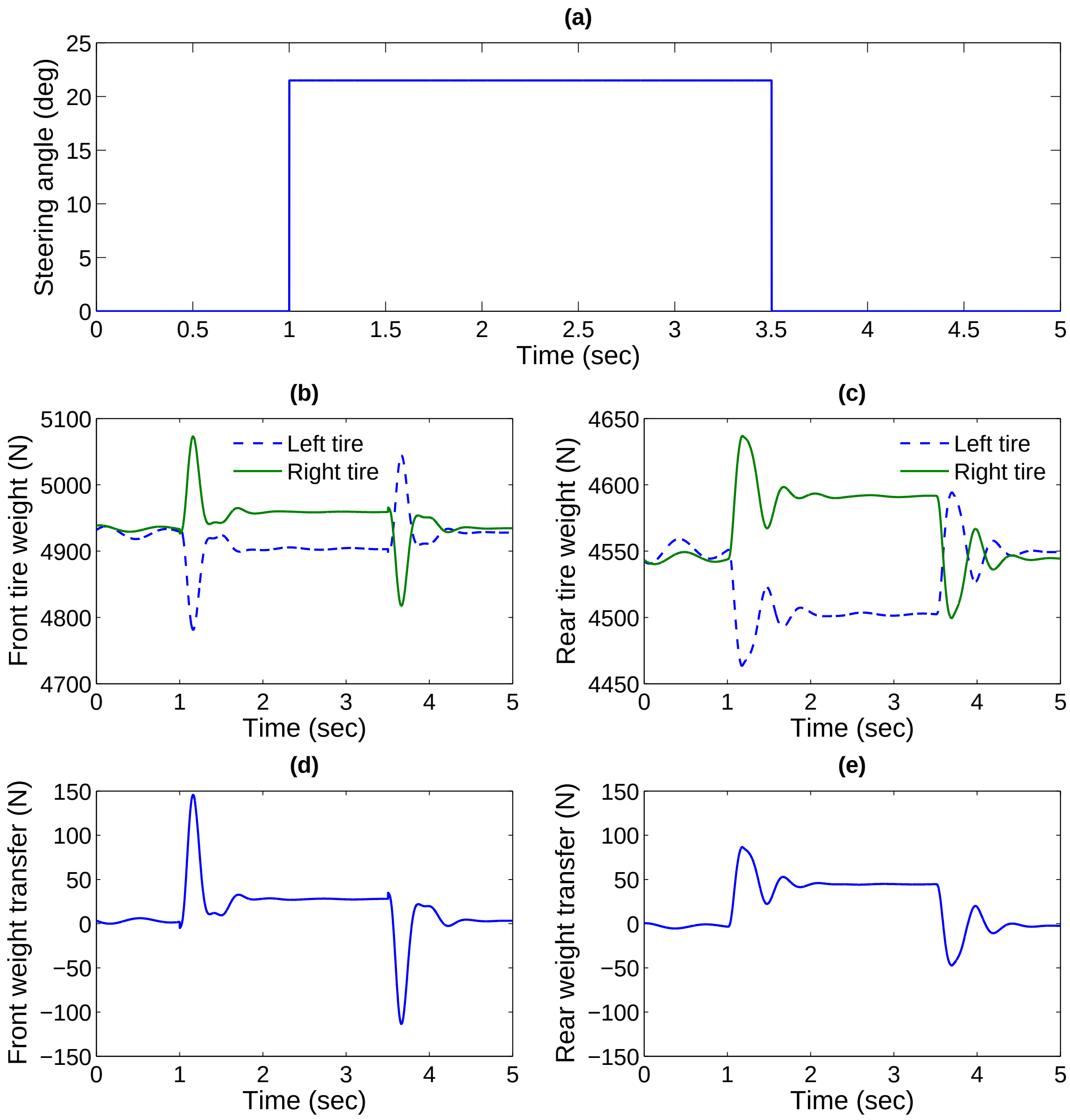


Figure 3.18: CarSim weight transfer test at 20 km/h: (a) Steering angle, (b) Front tires weight, (c) Rear tires weight, (d) Front weight transfer, (e) Rear wright transfer

20 km/h speed when the vehicle turns left, the right tire should provide more lateral force than right tire which can be provided either by increasing the weight (normal load) on tire or by increasing the slip angle. Since the weight transfer is small so the slip angle is increased to provide more lateral force than the left tire.

On the other hand during 70 km/h validation test right tire has a lot of added weight due to higher weight (resulting from large weight transfer) therefore it can provide required lateral force without increasing slip angle. i.e., the tire forces move in an upper region of the curve with higher lateral forces corresponding to high weight.

3.4 Trajectory Tracking using Model Predictive Control

The trajectory tracking architecture (Figure 3.20) used for control of a ground vehicle has long been used in the aerospace systems where it was known as Guidance and Navigation Control (GNC) system. Lu [92] and Smith [93] [94] used this architecture for simulations of Mars atmosphere entry and automatic landing of a spacecraft.

In Figure 3.20 top layer is the offline trajectory generation layer. The trajectory is generated in offline manner only once before the start of the mission. Usually this trajectory is planned by joining the start point and the end point (or way points) of desired travel with a straight line. However, other approaches can be used as well for offline trajectory generation. For example a driver wants to reach certain destination and enters the coordinates of the destination in his GPS navigation unit. The GPS navigation unit has many variables to customize such as: avoid toll or avoid congested areas which must be decided by the driver before travel so that the GPS navigation unit plans the trajectory accordingly.

Second layer in this architecture is the online trajectory generation layer which replans the trajectory during runtime. In case of a vehicle traveling on the road, it can be replanned based on happening of certain events such as: an obstacle is detected in path of the travel or a turn is missed by the driver. This is the layer where most intelligence and computation

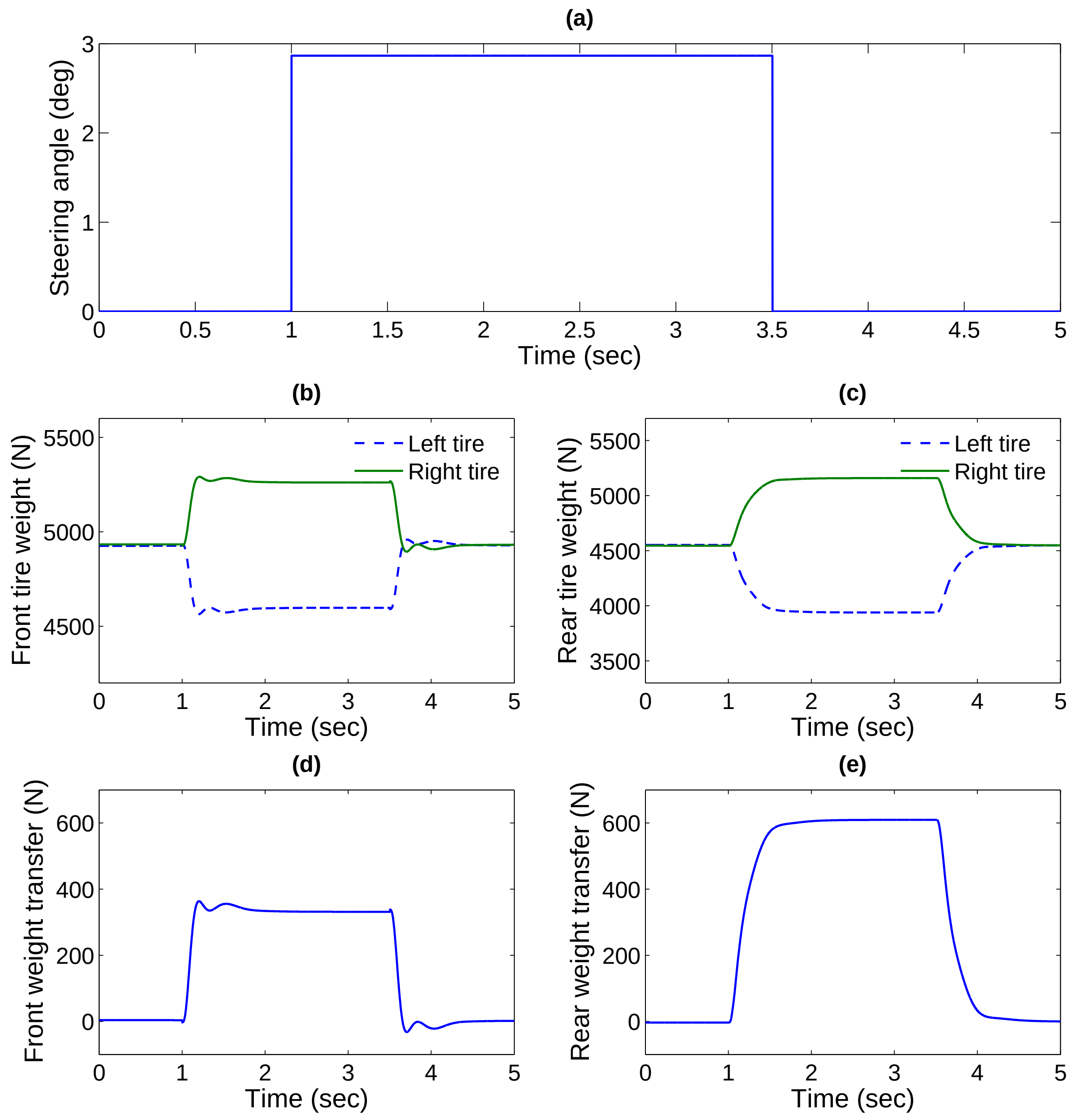


Figure 3.19: CarSim weight transfer test at 70 km/h: (a) Steering angle, (b) Front tires weight, (c) Rear tires weight, (d) Front weight transfer, (e) Rear wright transfer

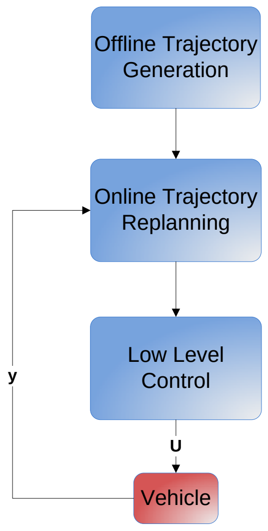


Figure 3.20: Vehicle Guidance System Architecture

is required in order to steer the vehicle efficiently through driving space. Model predictive controller trajectory controller works at this intelligent layer.

Third layer is the low level control layer which contains the basic control algorithms like PID and LQR to perform low level control on actuators (brakes and throttle etc) based on command from higher level control. Also the low level control uses different level of information from the high level control. High level controller can use the information available form satellites, cameras etc. Whereas the low level control uses the information from friction, throttle angle sensor etc.

The high level controller operates at much lesser frequency than the low level control. Typically the trajectory replanning module operates at in the range of 1Hz-50Hz and the low level controller operates in the range of 100Hz-1KHz depending upon the dynamics of the vehicle to be controlled.

The trajectory tracking problem is formulated as constrained optimization problem which minimizes a weighted sum of deviation from the desired trajectory, the state values and the control values. Consider the model developed in Equation 3.20, for the sake of MPC implementation this model need to be discretized first.

3.4.1 Discretization of Continuous-Time System

There are many methods for model discretization like Euler's forward differentiation method, Euler's backward differentiation method, Tustin's method, Bilinear transformation and Runge-Kutta method [95]. Higher order methods are much more accurate than the first order methods like Euler's method (Euler's method is explicit method). With Euler's method there is always an associated integration error which is proportional to the sampling interval T_s. However, a simple discretized model is key requirement for online-MPC implementation because for each controller step the model needs to be evaluated for planned trajectory tens of times. Higher order methods are more accurate but complicate the dynamics model unnecessarily [96]. So in this work the Euler's forward method is used for the simplicity of calculations. Continuous-time dynamics model is given by:

$$\dot{\xi}(t) = f_{cont}(\xi(t), u(t)) \tag{3.23}$$

where subscript $cont$ denotes that $\dot{\xi}$ is a continuous time function of the previous states. This function is discretized using the Euler's forward integration method for discretization interval T_s and time step $t = k$ or t_k. Replacing $\dot{\xi}(t)$ with its discrete-time equivalent $\dot{\xi}(t_k) \approx \frac{\xi(t_{k+1}) - \xi(t_k)}{T_s}$

$$\frac{\xi(t_{k+1}) - \xi(t_k)}{T_s} \approx f_{cont}(\xi(t), u(t))$$

$$\Downarrow$$

$$\xi(t_{k+1}) \approx \xi(t_k) + T_s f_{cont}(\xi_{t_k} k, u_{t_k})$$

$$\Downarrow$$

$$\xi(t_{k+1}) \approx f_{disc}(\xi_{t_k}, u_{t_k}) \tag{3.24}$$

for simplicity of notation t_k will be denoted by k. Detailed discretized model can be found in Appendix B. Value of the sampling interval T_s can be determined by the bandwidth of the vehicle to be used or by an experimental exercise. The latter strategy is used to determine the sampling interval in this case. A constant steer angle of 0.1 rad is applied to the steering input and results are plotted for the different sampling interval values and compared with the corresponding output from a continuous time model in Simulink A.

Figure 3.21a and Figure 3.21b are plots of continuous time model with 0.1 rad steering angle input. These plots serve as a reference while discretization of the model. A discretized model output should ideally be closer to the continuous time model plots.

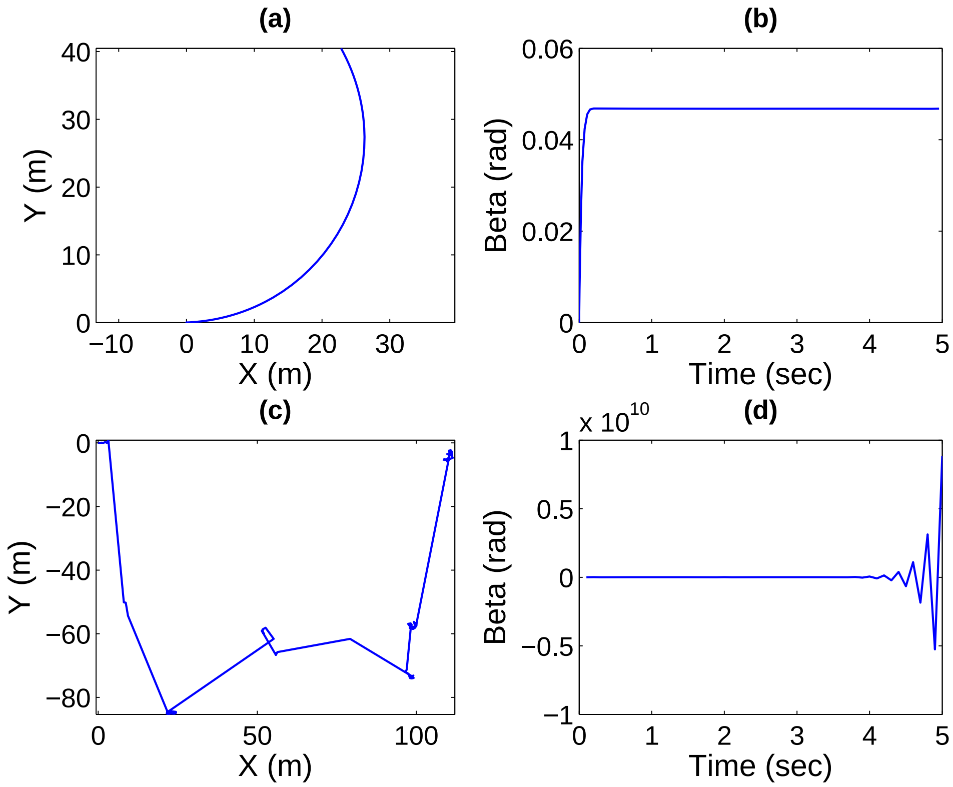


Figure 3.21: Stability Anaylsis: (a)Reference Trajectory obtained from *continuous time* model simulated in Simulink, (b) Beta (Vehicle side slip angle) obtained from *continuous time* model simulated in Simulink, (c) Vehicle trajectory at $T_s = 0.1s$, (d) Vehicle side slip angle $T_s = 0.1s$
, Note: Figures (c) and (d) demonstrate unstable results due to too slow sampling frequency.

Dynamic model is discretized by making some sensible choices of sampling frequency i.e., (1Hz -100Hz). We start from a 1Hz sampling frequency. Figures 3.21c and 3.21d show that for sampling frequency of 1Hz, the trajectory is randomly generated and the system is *unstable*. So the sampling frequency needs to be faster than 1Hz. Figures 3.22a and 3.22b are the model's outputs at 33.33Hz sampling frequency and the output is much close to the continuous time model frequency. Although there is some overshoot in the vehicle's side slip angle at $0.1s$. This overshoot is decreased when the model is simulated at 100Hz sampling frequency, as seen in Figures 3.22c, 3.22d, and the discrete-time model outputs closely matches the continuous time model output. Therefore the ceiling of safe operating range of model is slightly above 33Hz. The sampling frequency can be lowered but at the cost of more integration error.

3.4.2 Cost Function Formulation

For the trajectory generation with our required specifications we define the following generic cost function:

$$J = \phi(\tilde{\eta}_N) + \sum_{k=0}^{N-1} L(\xi_k, \tilde{\eta}_k, u_k) \tag{3.25}$$

where $\phi(\eta_N) = \frac{1}{2}\eta_N^T Q_0 \eta_N$ is the terminal cost function and penalizes the trajectory deviation at the last time step $(k = N)$ of look-ahead horizon and $\tilde{\eta}_k = \eta_{d,k} - \eta_k$. In this equation $\eta_{d,k}$ is the desired trajectory. $L(\xi_k, \eta_k, u_k)$ is the running cost and is defined as:

$$L(\xi_k, \tilde{\eta}_k, u_k) = \frac{1}{2}(\tilde{\eta}_k^T Q \tilde{\eta}_k + \xi_k^T S \xi_k + u_k^T R u_k) \tag{3.26}$$

The running cost penalizes the trajectory deviation at each time step $(k = (1, ..., N-1))$ during the entire horizon. The cost function (Equation 3.25) will be augmented later to suit our need. This is an advantage over traditional control techniques like LQR and PID which allows us to customize the controller cost according to our requirements. The matrices Q_0, Q, R and S are the penalty weighing matrices. The matrices $Q_0(2 \times 2)$ and $Q(2 \times 2)$

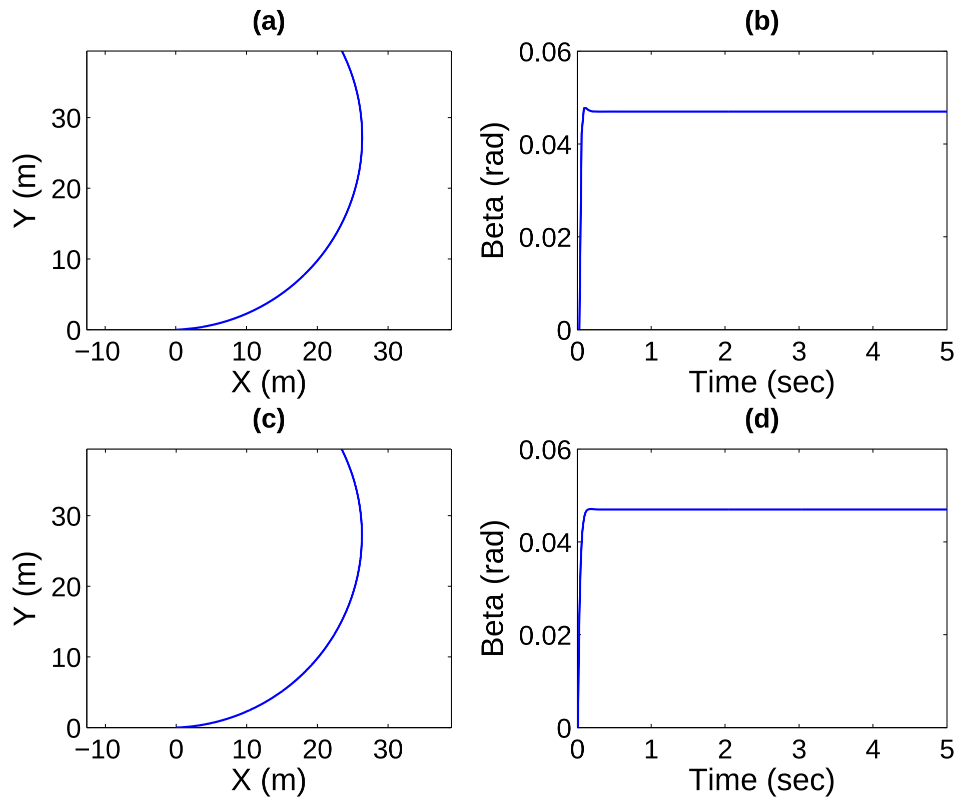


Figure 3.22: Stability Anaylsis: (a)Vehicle trajectory at $T_s = 0.03s$, (b) Vehicle side slip angle $T_s = 0.03s$, (c) Vehicle trajectory at $T_s = 0.01s$, (d) Vehicle side slip angle $T_s = 0.01s$

penalize the terminal trajectory deviation and the running trajectory deviation respectively. The matrices $S(5 \times 5)$ and $R(1 \times 1)$ penalize the large state values and the large input values. All the penalty weighing matrices are diagonal matrices.

For obstacle avoidance a point-wise repulsive potential function P_k is used as shown in Figure. 3.23

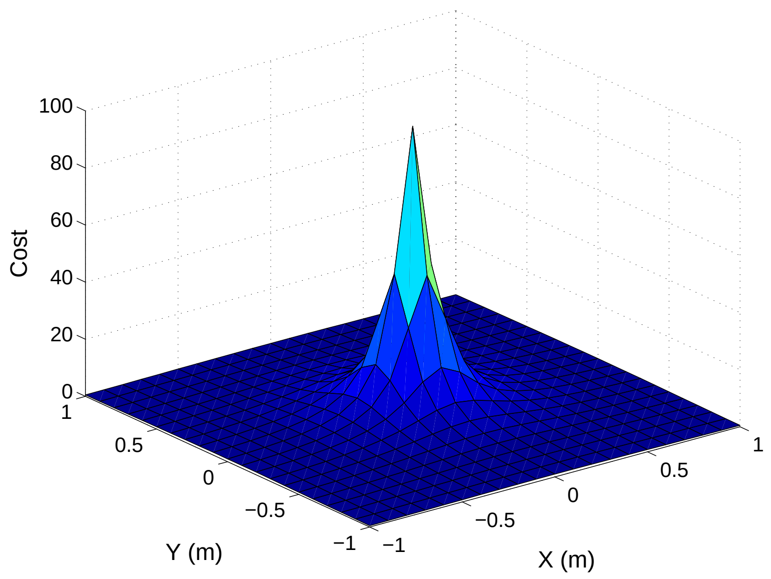


Figure 3.23: Point-wise repulsive function P_k

$$P_k = \frac{1}{(x - x_0)^2 + (y - y_0)^2 + \epsilon} \tag{3.27}$$

where (x, y) is the current location of the vehicle and (x_0, y_0) is the location of the obstacle to be avoided. ϵ is a small positive number for non singularity. Fahimi [57] and Xi [97] used such potential function for obstacle avoidance. The advantage with such simple function is that it is easily differentiable and does not lead to complex differential terms in the optimization part. It is notable here that the cost function in Figure. 3.23 can behave as a circular object function depending on where the trajectory tracking and object avoidance weight balance each other. For example the function of Equation 3.27 when cut from the base appears as in Figure 3.24. Radius of the circle in this figure can be increased by increasing the weight of potential function as compared to the trajectory tracking weights. With this choice of potential function, running cost becomes:

$$L(\xi_k, \tilde{\eta}_k, u_k) = \frac{1}{2}(\tilde{\eta}_k^T Q \tilde{\eta}_k + \xi_k^T S \xi_k + u_k^T R u_k) + P_k + constraint \tag{3.28}$$

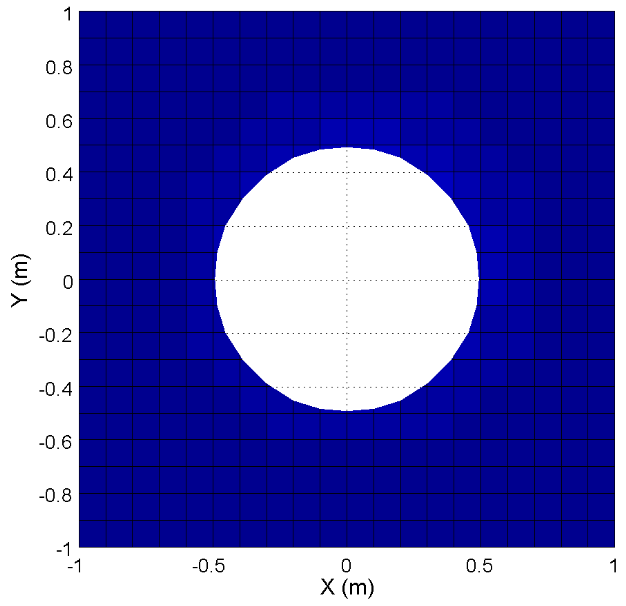
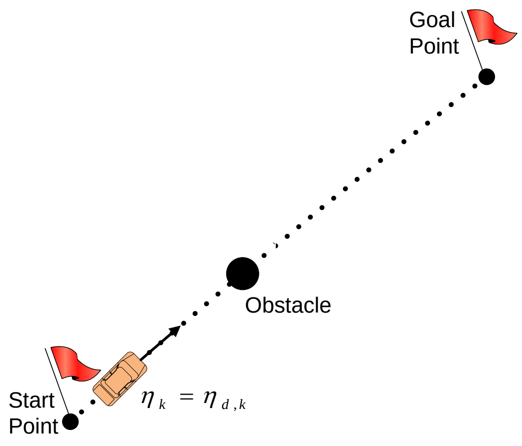
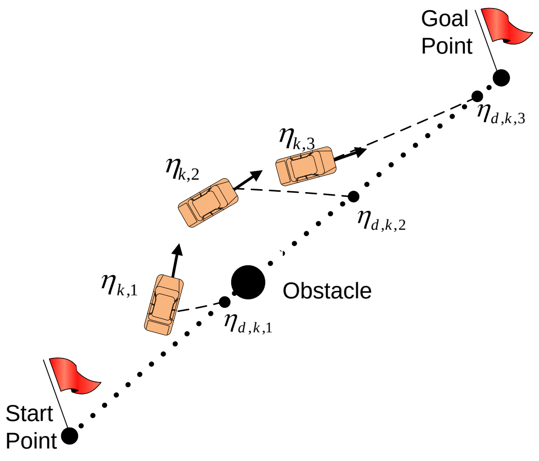


Figure 3.24: Top view of repulsive potential function P_k cut from bottom

The cost function 3.28 can be generalized to include as many obstacles as desired. Any obstacle shape can be generalized by this simple point obstacle method P_k if the nearest point on that obstacle is determinable. For example if a linear object is to be avoided, method in Section 3.4.3 can be used to find the nearest point on the obstacle's surface and then treat this point as P_k. Similar is the case with the circular obstacles. With multiple objects 3.28 becomes:



(a) No error accumulation



(b) Error accumulation after object detection

Figure 3.25: Time defined reference trajectory tracking. Error accumulates once the vehicle lags time step k

$$L(\xi_k, \tilde{\eta}_k, u_k) = \frac{1}{2}(\tilde{\eta}_k^T Q \tilde{\eta}_k + \xi_k^T S \xi_k + u_k^T R u_k) + \sum_{i=0}^{n} P_{ki} + constraint \tag{3.29}$$

where $i = 0, ..., n$ is the total number of the obstacles.

3.4.3 Reference Trajectory Criteria

Most important parameter of the cost function Equation 3.26 is the reference trajectory error $\tilde{\eta}_k$ which needs to be minimized. However, an interesting issue rises when the controller needs to know the desired trajectory point $\eta_{d,k}$ to calculate the error value $\tilde{\eta}_k = \eta_{d,k} - \eta_k$. Simplest solution is to define the desired trajectory as fixed function of the time.

Consider a vehicle starts from start point and proceeds towards goal point. Initially the vehicle trajectory η_k and the reference trajectory $\eta_{d,k}$ is synchronized because the vehicle is moving in a straight line (Figure 3.25a). But once the object is detected, the vehicle steers to avoid the obstacle and the vehicle CG fixed trajectory coordinate η_k lags but the reference trajectory $\eta_{d,k}$ keeps on moving along the straight line thus accumulating the error (Figure 3.25b). This error accumulation can be avoided by using some alternate method of the error calculation. Two possible method are considered in this thesis and each explained as following:

Method 1

One way is to find the nearest point on the reference line by using some trigonometric identities and treat this point as the desired trajectory point for current controller step. Such a method has been used by Eklund [56] to repel an aircraft from closest point on the boundary in order to keep it inside a predefined fixed boundary. However during this research work, a problem was found with this method. It can work well to repel vehicle from a boundary but if used for trajectory following this method can cause the vehicle to retreat once a big obstacle is detected. This method does not incorporate the goal point information. When retreating, the vehicle will follow the same line but will go backwards towards starting point. This effect will be shown later in the simulation results.

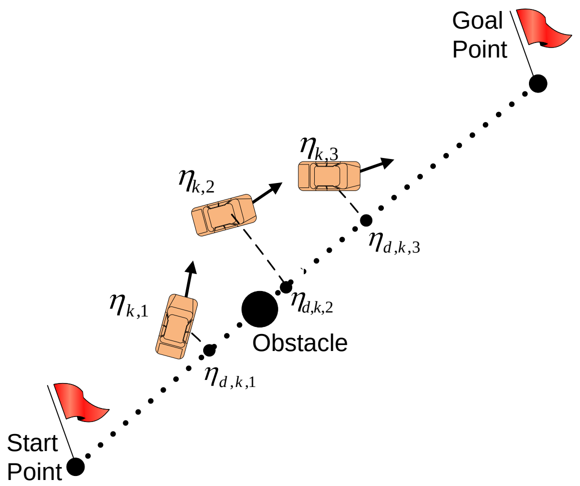


Figure 3.26: Method 1 for trajectory error calculation

Method 2

To overcome the problem with Method 1 another method developed by Yoon [98] has been used. At each time step $\eta_{d,k}$ is calculate as a point which lies on straight lines from vehicle CG to the reference line and selecting the one which is closer to the goal point. In Figure 3.27 when the vehicle is at point $\eta_{k,1}$, two points are found on the reference line corresponding to this point $\eta_{d,k,1,X}$ and $\eta_{d,k,1,Y}$ which are calculated by a horizontal and a vertical line from the vehicle CG to the reference line. But only point $\eta_{k,1,X}$ is considered to be the desired point $\eta_{d,k,1}$ because it is closer to the goal point. Thus the reference trajectory criterion has considered the approach to the goal point rather than just following the trajectory in any direction. Similarly when the vehicle is at point $\eta_{k,2}$, only $\eta_{k,2,Y}$ is considered the desired point $\eta_{d,k,2}$ due to its proximity to the goal point.

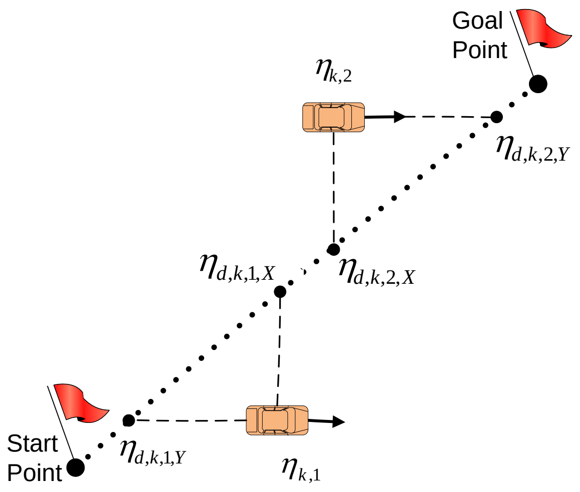


Figure 3.27: Method 2 for trajectory error calculation

3.5 Gradient Descent Algorithms

Gradient descent algorithm has played an important role in the progress in control science [99]. The actual gradient based optimization method dates back to as early as 1969 when they appear in Bryson's book [100]. Later Mehra [101] augmented the gradient based method with constraints. Gradient descent is used to minimize NMPC problem's Hamiltonian [15]. It is also known as *Steepest descend* and *Gradient* method. Plant input is the decision variable of this optimization problem. This method is valid only if the gradients can be calculated (i.e., J is differentiable with respect to every decision variable).

$$\frac{d}{du_k}J, \quad k = 1, ..., M \tag{3.30}$$

where M is the total number of the decision variables. To illustrate the concept further, consider a function $f(x)$ as shown in Figure 3.28. It is desired to find the minimum of this function. Optimization process is started by taking an initial guess on the function. Then slope of the function is calculated $\frac{\partial f(x)}{\partial x}$ and a step Δ in negative direction of the slope is taken and the value of $f(x)$ is calculated for each point. The process is repeated until the variation in $f(x)$ is zero, and it is said that local minimum of function is reached at point $x = m$.

Equation(3.13) is incorporated into the cost function (Equation(3.12) by introducing a sequence of Lagrange multiplier vectors $\lambda_k(k = 1, ..., N)$. The modified cost function becomes:

$$J = \phi(\tilde{\eta}_N) + \sum_{k=0}^{N-1} L(\xi_k, \tilde{\eta}_k, u_k) + \lambda_{k+1}^T[f_{disc}(\xi_k, u_k) - \xi_{k+1}] \tag{3.31}$$

Since $[u_k^*]_{k=0}^{N-1}$ needs to be determined such that the cost function J is minimized. Therefore dJ needs to be calculated as:

$$\begin{aligned} dJ = [\frac{\partial \phi}{\partial \xi_N} - \lambda_N^T]d\xi_N + \frac{\partial H_0}{\partial \xi_0}d\xi_0 + \frac{\partial H_k}{\partial \tilde{\eta}_0}d\tilde{\eta}_0 + \frac{\partial H_k}{\partial u_0}du_0 \\ + \sum_{k=0}^{N-1}[(\frac{\partial H_k}{\partial \xi} - \lambda_k^T)d\xi_k + \frac{\partial H_k}{\partial \tilde{\eta}_k}d\tilde{\eta}_k + \frac{\partial H_k}{\partial u_k}du_k] \end{aligned} \tag{3.32}$$

Since $\xi_{k+1} - f_{disc}(\xi_k, u_k) = 0$ the term $\lambda_{k+1}^T[f_{disc}(\xi_k, u_k) - \xi_{k+1}]$ is also zero. λ_{k+1}^T can be chosen arbitrarily since it is being multiplied by a zero term. For this research, a

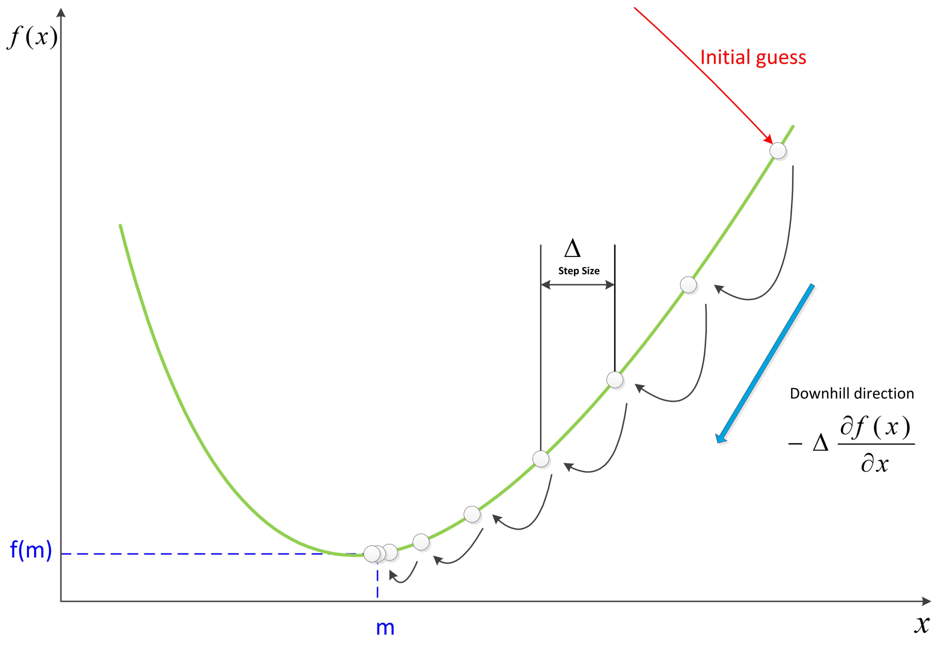


Figure 3.28: Gradient descent algorithm

costate equation similar to Fahimi [57] has been used but with corrections. Equation 23 in his paper was found to have error and the subtraction sign in second part of the equation should be replaced by equality. The author of the paper was contacted verified the error in publication. Costate values selected are:

$$\lambda_N^T = \frac{\partial \phi}{\partial \xi_N} = -\tilde{\xi}_N^T Q_0 C \tag{3.33}$$

$$\lambda_k^T = \frac{\partial H_k}{\partial \xi_k} + \frac{\partial H_k}{\partial \tilde{\eta}_k}\frac{\partial \tilde{\eta}_k}{\partial \xi_k} \tag{3.34}$$

$$= x_k^T S + \lambda_{k+1}^T + \lambda_{k+1}^T \frac{\partial f_{disc,k}}{\partial \xi_k} - \tilde{\eta}^T QC + \frac{\partial P_k}{\partial \xi_k} \tag{3.35}$$

With these choices Eq.(3.32) is simplified to following form:

$$dJ = \sum_{k=0}^{N-1} \frac{\partial H_k}{\partial u_k} du_k + \lambda_0^T d\xi_0 \tag{3.36}$$

where

$$\frac{\partial H_k}{\partial u_k} = u_k R + \lambda_{k+1}^T \frac{\partial f_{disc,k}}{\partial u_k} \tag{3.37}$$

Algorithm

The current state at which optimization is being solved, ξ_0 remains constant. So $d\xi_0 = 0$, Equation 3.36 becomes $dJ = \sum_{k=0}^{N-1} \frac{\partial H_k}{\partial u_k} du_k$. Therefore the partial derivative $\frac{\partial H_k}{\partial u_k}$ is essentially the gradient of cost function J with respect to u_k assuming that the state ξ_0 remains constant during this interval. At local minimum of J partial derivative $(\frac{\partial H_k}{\partial u_k}) \approx 0$ which serves as a condition for terminating the optimization loop.

Following optimization problem is solved at each sampling instant [55]:

- At initial state ξ_0, assume a candidate input $[u_k^*]_{k=0}^{N-1}(k = 0, ..., N-1)$.
- Calculate co-states backward in time $(k = N, ..., 1)$ using Eq.(3.35)
- Obtain gradients from Equation(3.37)
- Update candidate input $[u_k^*]_{k=0}^{N-1}$ by taking a small step downhill the cost function (in negative direction of gradient) i.e., $[u_k^*]_{k=0}^{N-1} = [u_k^*]_{k=0}^{N-1} - [\Delta \frac{\partial H}{\partial u}]_{k=0}^{N-1}$
- Repeat above four steps until $dJ \approxeq 0$.
- Apply first element of optimized input vector $[u_k^*]_{k=0}^{N-1}$ to the plant under control.
- Repeat the whole problem for next sampling instant.

Only the first element in optimized input vector $[u_k^*]_{k=0}^{N-1}$ is applied because of the difference in identified prediction model and system to be controlled. The actual closed-loop input and state trajectories differ from the predicted open-loop trajectories, even if no model plant mismatch and no disturbances are present. This prediction error can accumulate over time if only identified model is used for the predictions over the entire horizon N.

3.6 Properties of Gradient Descent Algorithms

Initial Guess For good initial guess of $[u_k^*]_{k=0}^{N-1}$, physical insight about the system should be utilized. One way is to use initial state of the system ξ_0 to find LQR control input vector and then initialize $[u_k^*]$ with this vector. In this research, another method has been used. At the first sampling instant $[u_k^*]_{k=0}^{N-1}$ is initialized to $0(1 \times N)$. Then at each subsequent sampling instant the input vector is initialized by $[u_k^*]_{k=0}^{N-1} = (u_2, ..., u_N, u_N)$. Doing so the iteration count is reduced significantly [55].

Memory Requirements At each iteration the candidate control sequence $[u_k^*]_{k=0}^{N-1}$, corresponding state vector $[\xi^*]_{k=1}^{N}$ and cost function J history must be stored. However if the storage space is limited, $[\xi^*]_{k=1}^{N}$ can be calculated again when required which increases the computation cost. But it can increase the accuracy of algorithm because we no longer need to assume that initial condition ξ_0 should remain constant [15].

Convergence This optimization is not novel but is computationally feasible. The algorithm can trap into local minima instead of the global minima thus giving a suboptimal solution. But a quick and sub-optimal solution is preferred for the real-time systems as a delayed solution is wrong solution no matter how accurate it is. Also, as a minimum is approached, the algorithm converges slowly because step size Δ decreases.

Computation Requirement At each iteration $S \times N$ state equations need to be solved to compute the planned trajectory. Where S is the number of states and N is the prediction horizon. But gradient calculation takes most of the computation time. For each input perturbation we need (N+1) simulation steps times prediction horizon nN.

Stopping Criteria The iterations can be stopped when the stopping criteria $||\frac{\partial H}{\partial u}|| \leq \epsilon_1$ or $|J^{i-1} - J^i| \leq \epsilon_2$ is satisfied. ϵ_1 and ϵ_2 are small positive constant values and i is the iteration number. In this research only second criterion is used.

Effect of tuning R Tuning of the input weighing matrix R has noticeable effect on fine tuning of the optimization. With smaller value of R, first elements of candidate control vector are found more accurately which is desirable since only the first element is applied to the system to be controlled. This issue has been further explored in detail by Sutton [102].

Constraints Handling A simple method of implementing constraints is by projecting the candidate input $[u_k^*]_{k=0}^{N-1}$ on the constraint set. If a particular value of $[u_k^*]_{k=0}^{N-1}$ violates the constraint, it becomes the value at which the constraint was violated. An accurate but slightly more computationally expensive way of constraint handling is by using the penalty functions. This method has been used for constraining the steering angle in this research.

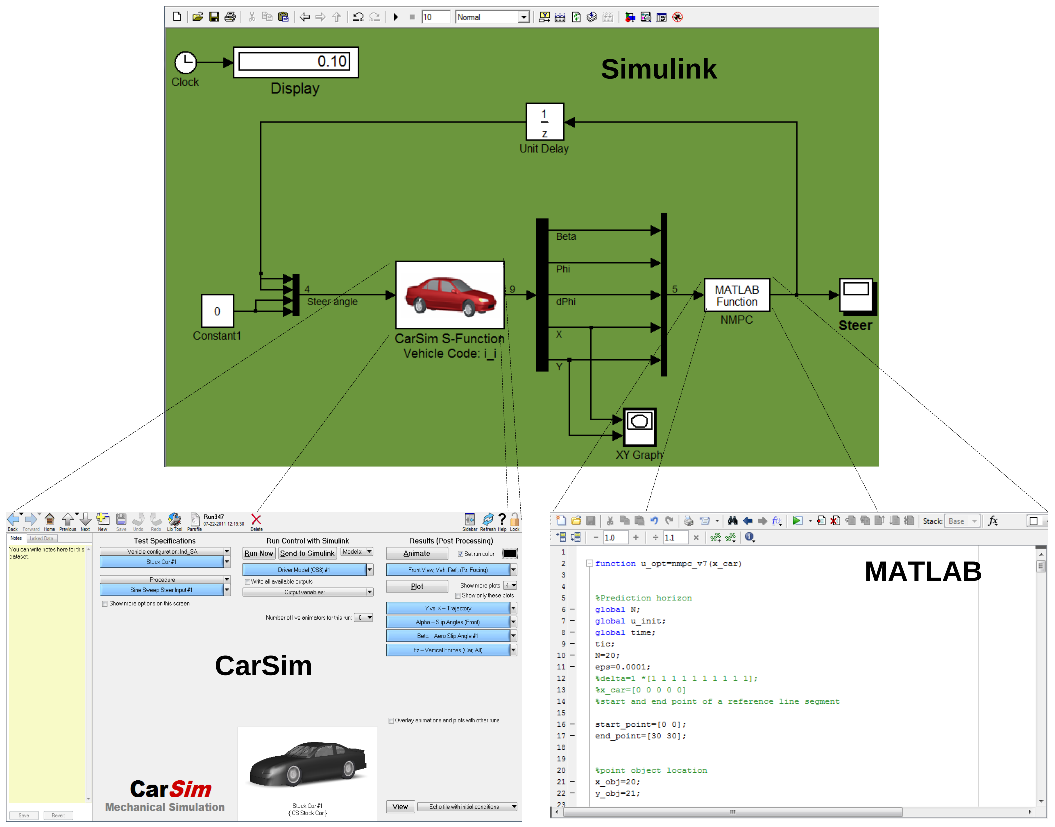
Chapter 4

Results

This chapter presents the results obtained from simulations performed on Carsim model. Pros and cons of multiple reference trajectory criteria (outlined in Sections 3.4.3, 3.4.3) are demonstrated with simulations performed on the fully nonlinear CarSim model. Then some light is shed on the real-time suitability of this implementation. Various types of objects (point, circular and linear) are simulated in different scenarios. Finally, tests are presented to demonstrate how far controller can tolerate the variations in vehicle's parameters without the need of being re-tuned.

4.1 Simulation Environment

MATLAB was selected for NMPC implementation because the matrix operations are easy to implement. Also, CarSim math model of the whole vehicle system (aerodynamics, driver, ground) can be extended to Simulink/MATLAB in order to add advanced controllers. CarSim vehicle model is exported as an S-function into the Simulink. NMPC controller is coded as MATLAB script file and imported to Simulink by using MATLAB Function block. Thus the Simulink acts as a bridge between the MATLAB and the CarSim. It is worth noting here that CarSim must be running during the simulations as the S-function embedded inside Simulink uses CarSim math model at the runtime and does not contain a math model in it-



```
function u_opt=nmpc_v7(x_car)


%Prediction horizon
global N;
global u_init;
global time;
tic;
N=20;
eps=0.0001;
%delta=1 *[1 1 1 1 1 1 1 1 1 1];
%x_car=[0 0 0 0 0]
%start and end point of a reference line segment

start_point=[0 0];
end_point=[30 30];


%point object location
x_obj=20;
y_obj=21;
```

Figure 4.1: Simulation environment

self. Sampling time of the NMPC controller can be set by changing the sampling time value in the Function Block parameter and the external ADC/DAC are not required. Simulation is initiated by the Start Simulation button in the Simulink control buttons which calls both the MATLAB and the CarSim S-Fuction block (Figure 4.1).

4.2 Comparison of Multiple Reference Trajectory Criteria

The multiple reference trajectory following criterion explained in the Section 3.4.3 are explored here with simulations performed using the point and linear obstacles. Parameter of the vehicle are same as those given in the Table 3.1. The vehicle starts from South West corner and goal point is located on the North East corner of the simulation area. Reference trajectory is a straight line joining the start point to the goal point.

4.2.1 Scenario I: Point Obstacle

In Scenario I, a point obstacle was placed on the planned path of the vehicle and controller was tuned according to the parameters in Table 4.1. First simulation was performed with Method 1 trajectory following criterion. Vehicle clearly avoided the obstacle by steering away from the obstacle and reached goal point successfully (Figure 4.2).

Table 4.1: Parameters used in Scenario I

Parameter	Value
Horizon Length	$N = 20steps$
Sampling Time	$T_s = 0.05s$
Steering Angle Limit	$-20\deg \leq u \leq 20\deg$
Speed	$v_x = 5.5m/s$
Simulation Time	$20s$
Q_0	$[0.1\ 0; 0\ 0.1]$
Q	$[0.1\ 0; 0\ 0.1]$
R	$[0.01]$
S	$0(5 \times 5)$

Second simulation was performed using the same controller tuning matrices but changing the trajectory following criterion to Method 2. Other simulation parameters were the same. Resulting simulation results are give in the Figure 4.3. Vehicle successfully avoided the obstacle by steering away from the planned path. There was a slight difference in how the vehicle approached the reference trajectory after avoiding the obstacle but the vehicle heading, side slip and the steering angle remained the same in both cases.

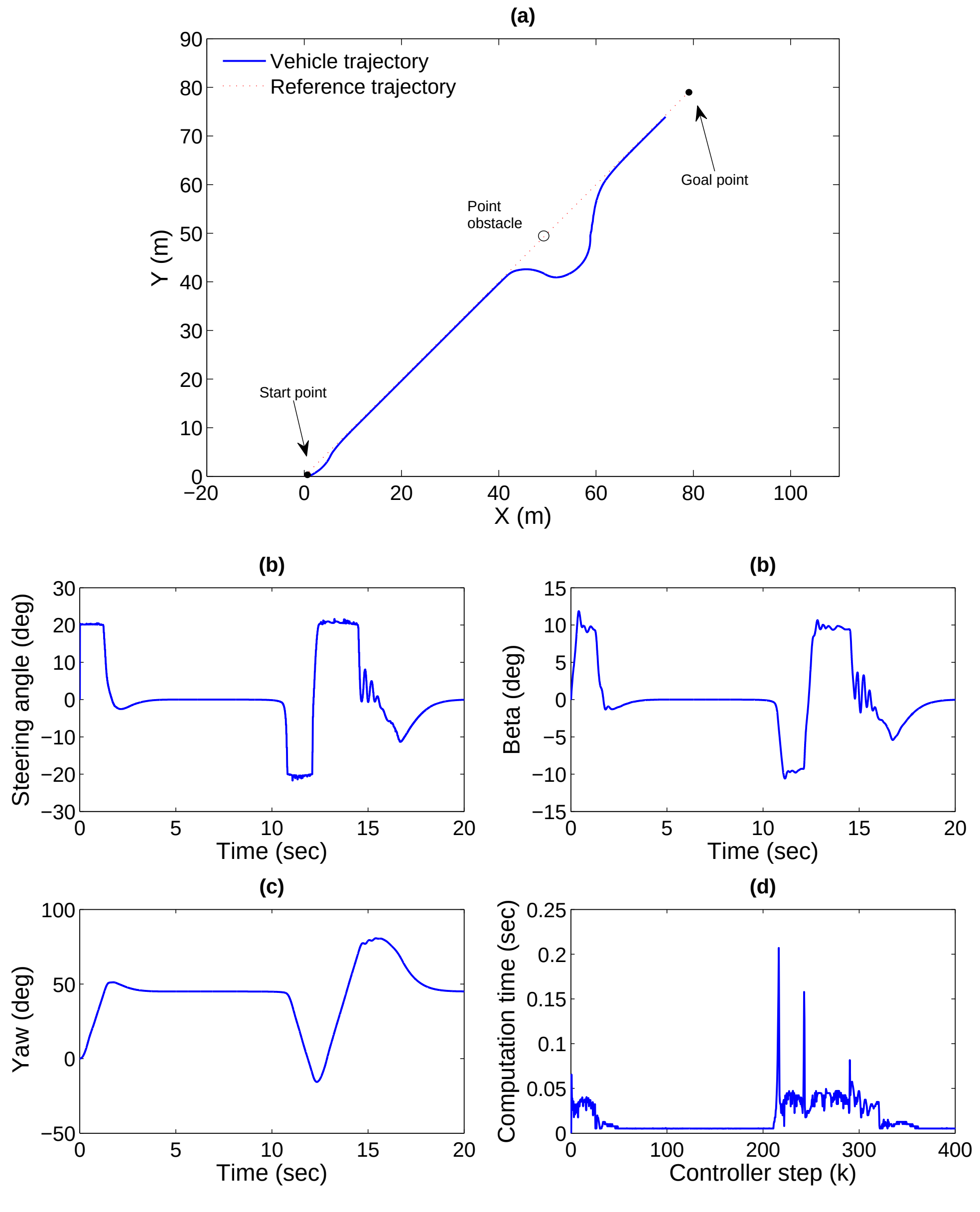


Figure 4.2: Scenario I: Point obstacle avoidance using Method 1 trajectory following criterion. (a) Trajectory (b) Steering input angle, (c) Vehicle side slip angle, (d) Vehicle heading, (e) Computation time per controller step

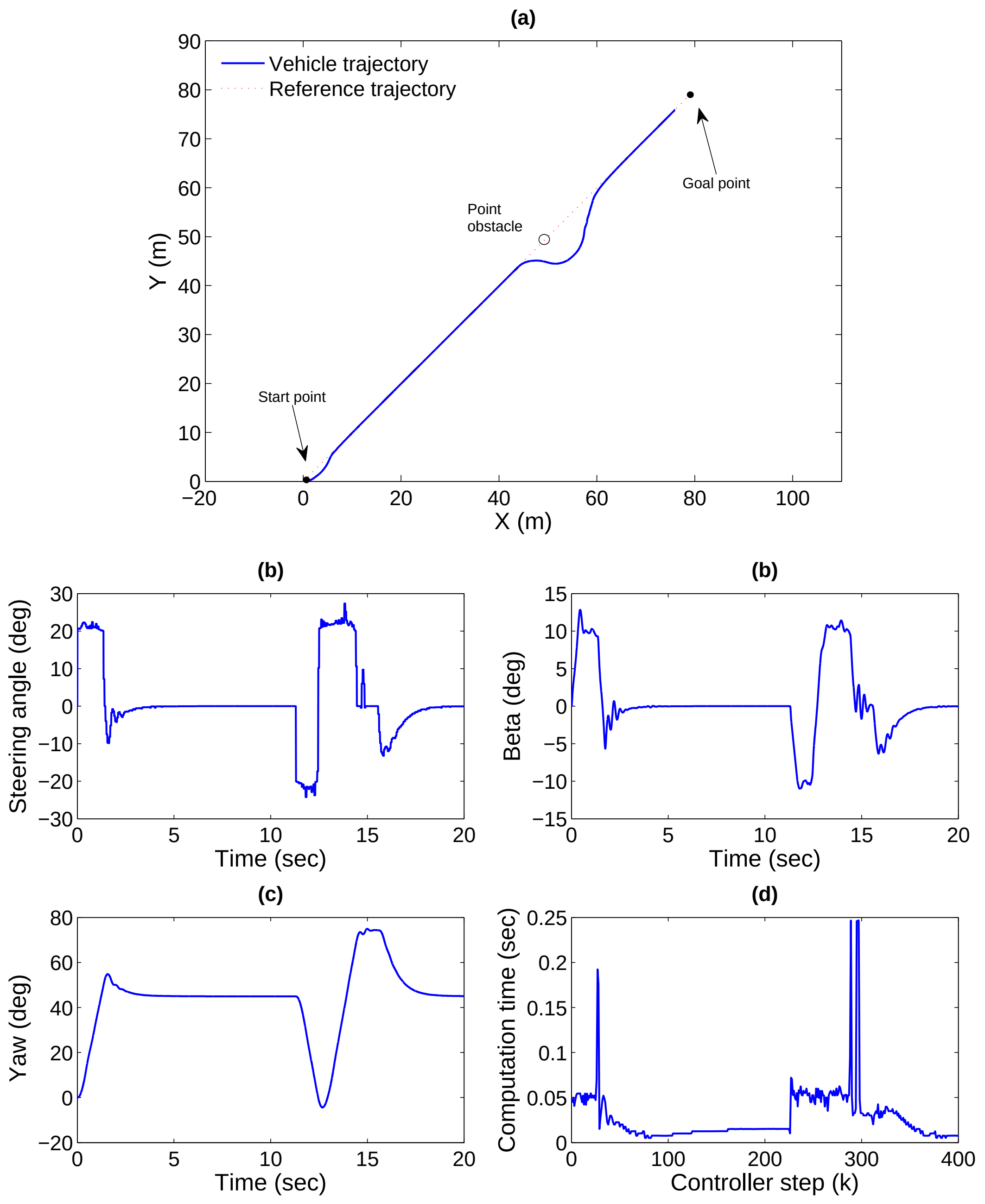


Figure 4.3: Scenario I: Point obstacle avoidance using Method 2 trajectory following criterion. (a) Trajectory, (b) Steering input angle, (c) Vehicle side slip angle, (d) Vehicle heading, (e) Computation time per controller step

4.2.2 Scenario II: Linear Obstacle

In Scenario II a linear obstacle was placed on path of the vehicle and the controller was tuned according to the parameters in Table 4.2. First simulation was performed with Method 1 trajectory following criterion. Purpose of placing a linear object was so that the vehicle sees a larger obstacle as compared to a point obstacle. Also, the steering angle constraint was relaxed to 10 degrees. From Figure 4.4 it is observed that vehicle retreats because the trajectory following criterion does not have information about the goal point. Objective of the vehicle was to follow the linear trajectory only and not to reach the goal point. The obstacle was so large that NMPC algorithm prefered to follow the trajectory in opposite direction.

Table 4.2: Parameters used in Scenario II

Parameter	Value
Horizon Length	$N = 20steps$
Sampling Time	$T_s = 0.05s$
Steering Angle Limit	$-20\,\deg \leq u \leq 20\,\deg$
Speed	$v_x = 5.5m/s$
Simulation Time	$20s$
Q_0	$[0.1\ 0; 0\ 0.1]$
Q	$[0.1\ 0; 0\ 0.1]$
R	$[0.01]$
S	$0(5 \times 5)$

Retreating situation was avoided by using Method 2. With the same controller tuning matrices, the car did not retreat but rather steered away from the object. Results for this simulation are shown in Figure 4.5

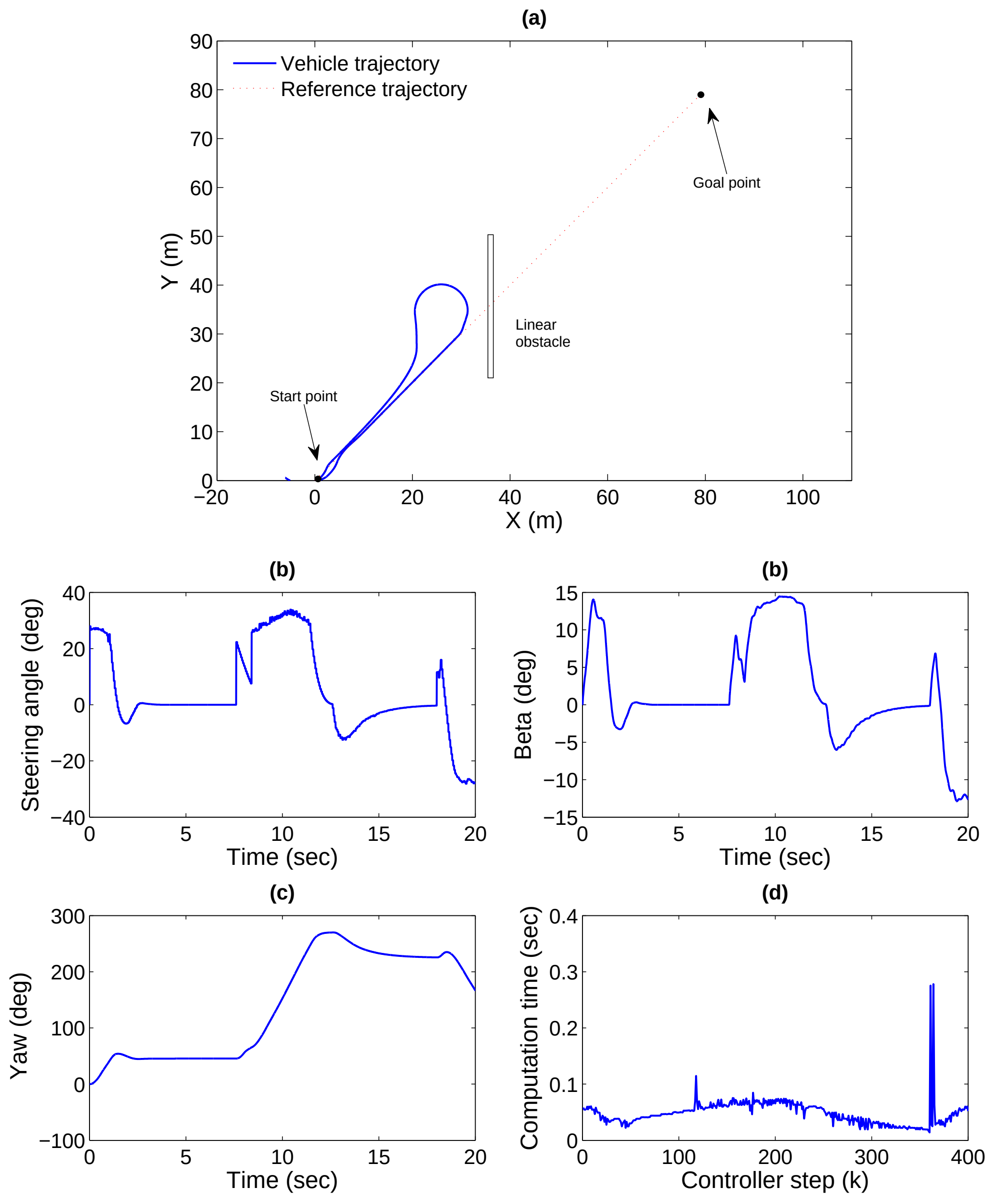


Figure 4.4: Scenario II: Linear obstacle avoidance using Method 1 trajectory following criterion and vehicle retreat (a) Trajectory, (b) Steering input angle, (c) Vehicle side slip angle, (d) Vehicle heading, (e) Computation time per controller step

4.2.3 Discussion

Both methods of the trajectory following work well for small obstacles but once a large obstacle is detected, vehicle can retreat in case of Method I. Method 2 works for every scenario regardless of the size of obstacle.

4.3 Obstacle Avoidance Simulations

Some advanced tests were conducted to demonstrate NMPC controller's ability to steer vehicle in *unknown environment* with obstacles. This section focuses on the obstacle avoidance simulations with combination of point and linear objects. Simulations results are presented considering the linear obstacle as the curb of the road and point object as an obstacle on the path. All simulations were performed on actual CarSim vehicle model.

4.3.1 Scenario I: Horizontal Road

In Scenario I, the vehicle was simulated on a realistic horizontal road with two lanes. Width of the road was set to $7.0m$ ($7.23m$ is the average width of two lane roads in North America). Length of the road was $250m$. It should be noted that as opposed to previous tests, the trajectory plot for this section are not proportional. y-coordinates of the vehicle trajectory are exaggerated as compared to x-coordinates of the vehicle trajectory. The tests were performed at three different speeds and two different horizon lengths. Tuning matrices and other parameters were same for all Scenario I simulations and are given in Table 4.3

The vehicle started at far West corner of the road and on the right lane. Two linear obstacle were placed at the boundaries of the road to represent the curb of the road. Target of the vehicle was to reach the East end of the road on the left lane while avoiding hitting the obstacles and the curb of the road. Lane change was planned to take place at $X = 150m$. Of course vehicle was allowed to change lane before $X = 150$ mark if there was any obstacle on right lane but had to return to original lane immediately after. Location of the obstacle

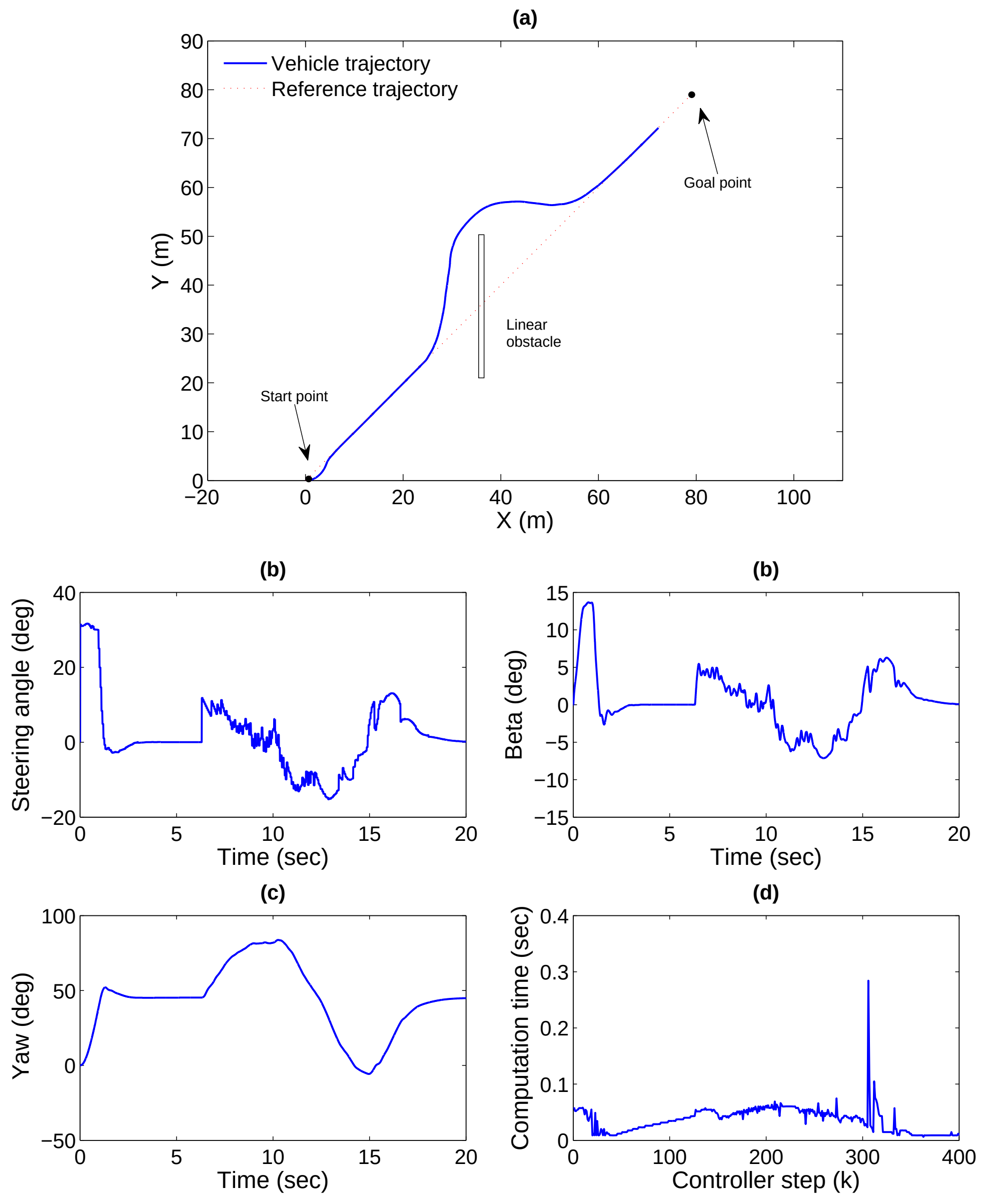


Figure 4.5: Scenario II: Linear obstacle avoidance using Method 2 trajectory following criterion. (a) Trajectory, (b) Steering input angle, (c) Vehicle side slip angle, (d) Vehicle heading, (e) Computation time per controller step

Table 4.3: Parameters used in Scenario I

Parameter	Value
Sampling Time	$T_s = 0.05s$
Q_0	$[0.1\ 0; 0\ 0.1]$
Q	$[0.1\ 0; 0\ 0.1]$
R	$[0.5]$
S	$0(5 \times 5)$

was $(0, 50)$ which was unknown to the vehicle unless it came within look-ahead horizon of the vehicle.

20km test

First test was performed at slower speed of $20km/h$. Length of simulation was set to $40s$ so that vehicle could cover $250m$ length of the road. For this speed and sampling time, look-ahead horizon distance at $N = 20$ and $N = 40$ was $5.5m$ and $11m$ respectively. Steering angle was constrained to $\pm 20\deg$ as the mathematical model has been demonstrated to work for this steering angle and speed in validation tests.

From Figure 4.6a it is evident that the vehicle successfully avoided the obstacle by steering to the left and then returned to its right lane. With longer lookahead horizon ($N = 40$), the vehicle did not turn to the left immediately but rather turned to the right first. This effect is called *anticipated move*. It might look like there is more error for this maneuver using longer horizon, but the net error is lesser as compared to shorter horizon test. This effect will be demonstrated further in Scenario II. Such effect was not seen with shorter horizon length of $N = 20$ because when the vehicle detected object, it did not have enough time to make anticipated move. Figure 4.6e shows how computation cost increases with larger look-ahead horizon. Thus look-ahead horizon (N) is a trade-off between tracking performance and computation effort. Larger the horizon, better the tracking performance

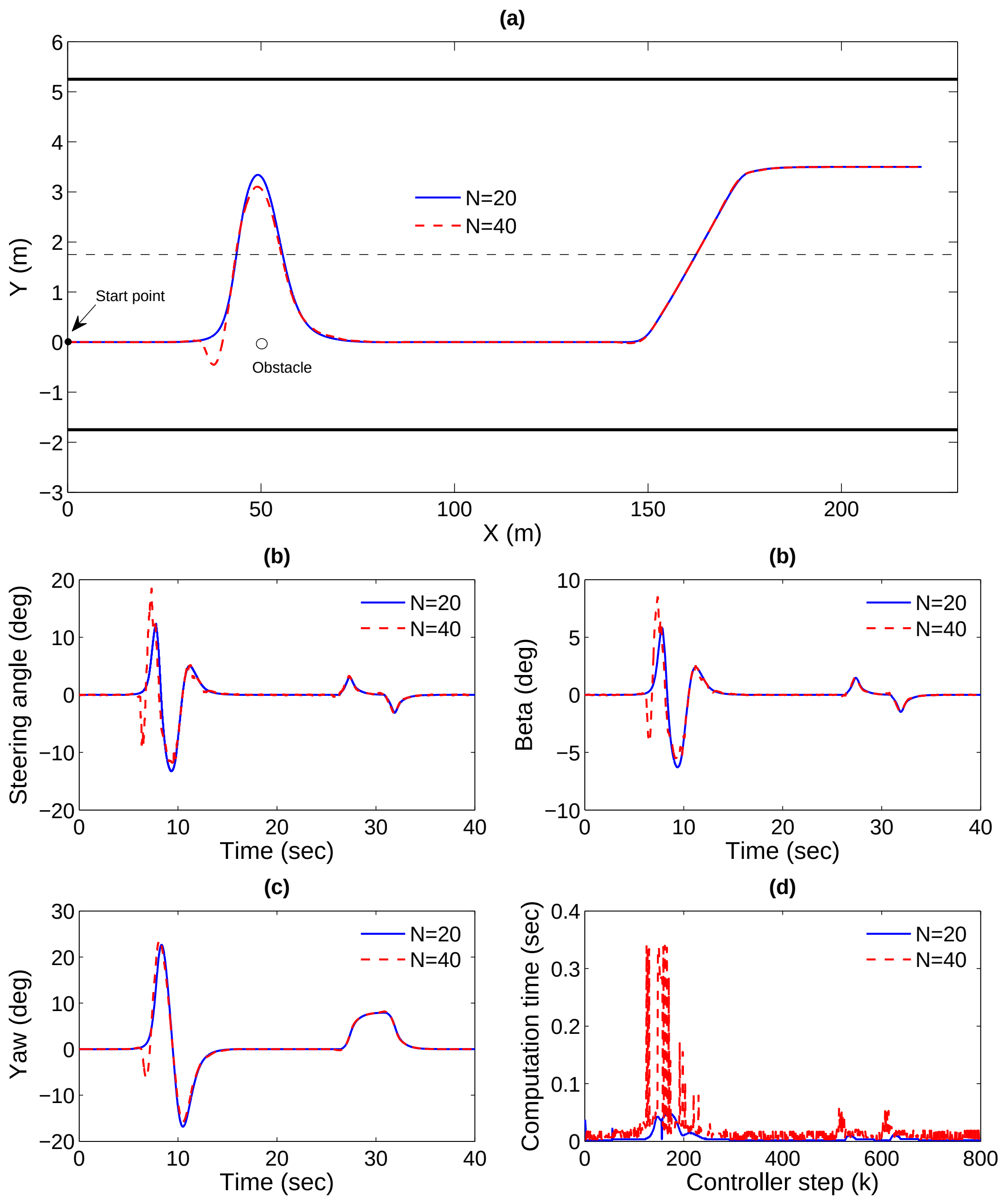


Figure 4.6: Scenario I: Horizontal Road test results, (a) Vehicle Trajectory, (b) Steering angle input, (c) Vehicle side slip angle, (d) Vehicle heading, (e) Computation time per controller step

and increased computation cost. Then the vehicle changed the lane successfully at $X = 150m$ and merged into the left lane. Tracking performance for both the horizon lengths was almost equal in this case.

40km test

Second test was performed at the speed of $40km/h$. Length of the simulation was cut shorter to $20s$ so that vehicle could cover entire $250m$ length of the road. For this speed and sampling time, look-ahead horizon distance at $N = 20$ and $N = 40$ was $11m$ and $22m$ respectively. Steering angle constraint was tightened to $\pm 10\deg$ for these simulations due to higher speed. Again vehicle successfully avoids obstacle and later turns to the left for lane change. More anticipated turning is seen in the case of $N = 40$ horizon length. Apart from computation time, longer horizon simulations show superior performance as compared to shorter horizon simulations.

60km test

Third test was performed at speed of $60km/h$. Length of simulation was $13s$ so that the vehicle could cover $250m$ length of the road. For this speed and sampling time, look-ahead horizon distance at $N = 20$ and $N = 40$ was $16.6m$ and $33.2m$ respectively. Steering angle constraint was further tightened to $\pm 5\deg$ for these simulations due to higher speed.

It was observed that at horizon length $N = 20$, the vehicle moved too close to the boundary with only $1m$ distance form curb of the road in order to avoid the obstacle. Where as at $N = 40$ the vehicle avoided the obstacle clearly while keeping a safe distance form curb. More anticipated turning was seen in this case due to higher speed and tighter steering constraints. The MPC controller took advantage of extra planning time available to further optimize the move and plan the path intelligently.

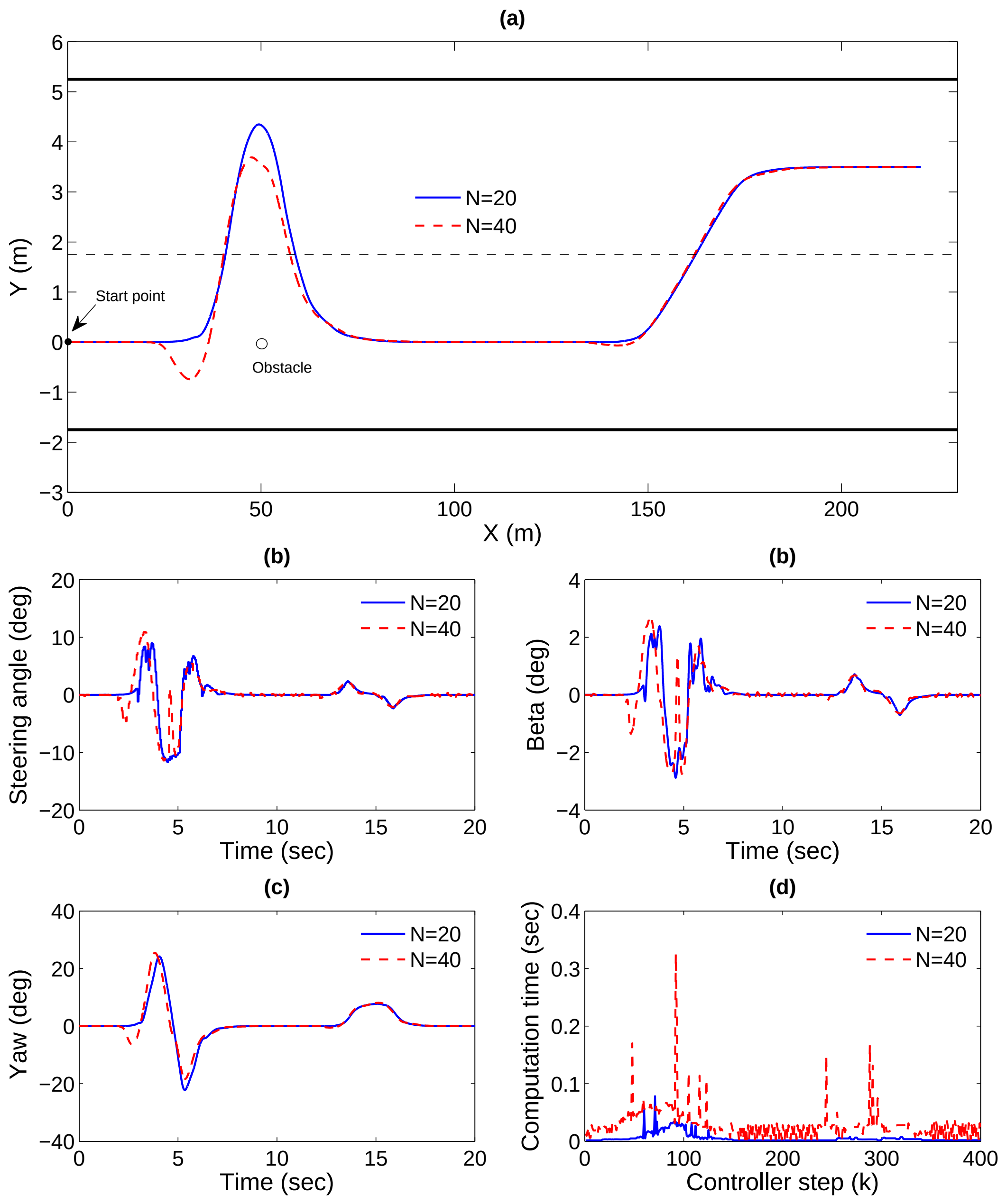


Figure 4.7: Scenario I: Horizontal Road test results, 40km test results, (a) Vehicle Trajectory, (b) Steering angle input, (c) Vehicle side slip angle, (d) Vehicle heading, (e) Computation time per controller step

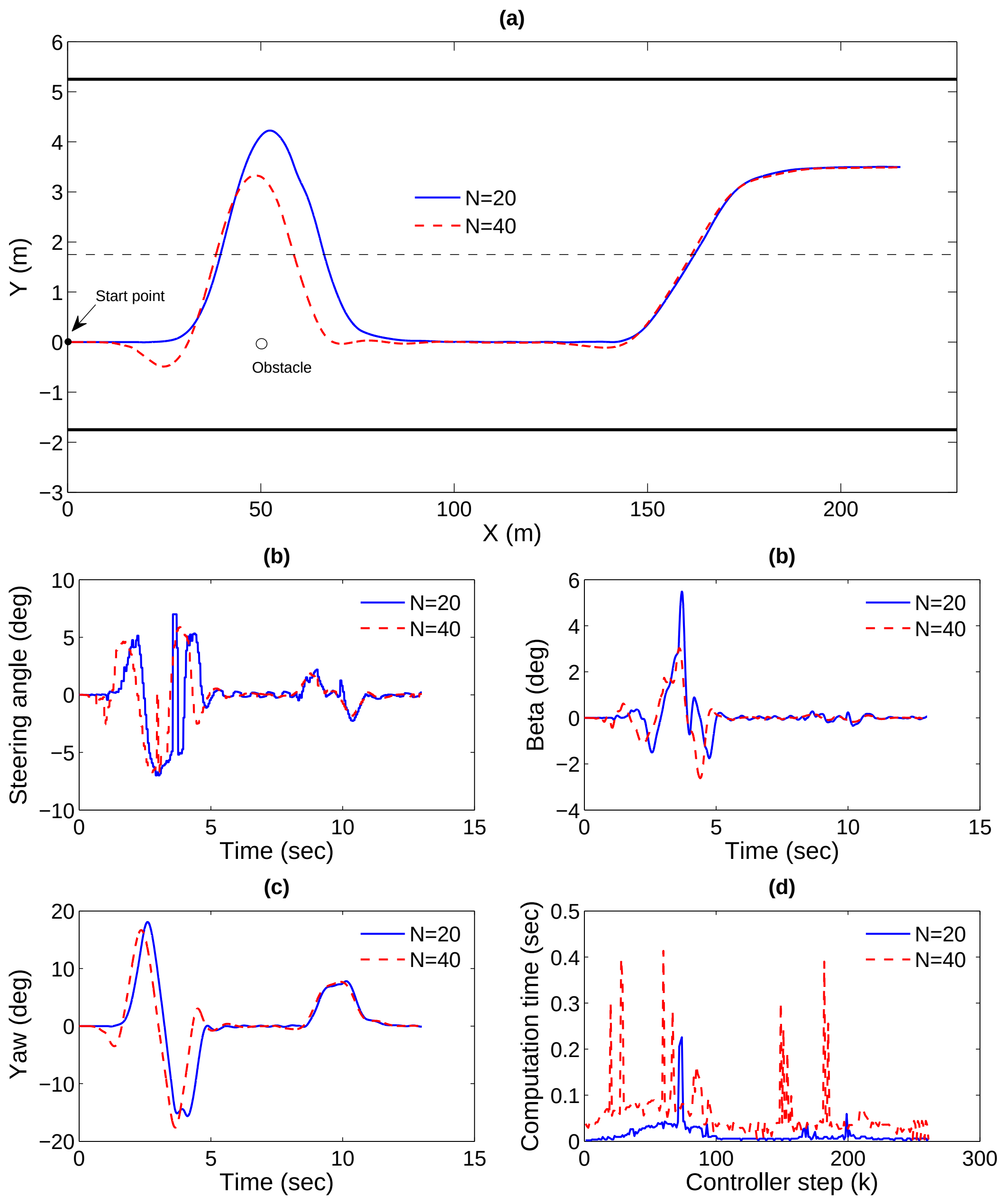


Figure 4.8: Scenario I: Horizontal Road test results, 60km test results, (a) Vehicle Trajectory, (b) Steering angle input, (c) Vehicle side slip angle, (d) Vehicle heading, (e) Computation time per controller step

4.3.2 Scenario II: Anticipated Move

To demonstrate the effect of anticipated move, some additional tests were performed. In Scenario II, the vehicle started at a speed of $40km/h$ headed towards the East, a $60\,\deg$ left turn was planned at $X = 60m$ and a $90\,\deg$ right turn at $X = 95m$. Different horizon lengths ranging from $N = 20$ to $N = 100$ were selected to observe their effect on the tracking performance of the controller. Steering constraint was $\pm 20\,\deg$ and other tuning parameters of the controller were the same as those in the Table 4.3. Figure 4.9a shows the planned and the actual trajectory for the Scenario II with all horizon lengths i.e., $N = 20, 40, 60, 80, 100$. $60\,\deg$ and $90\,\deg$ turns are zoomed in Figure 4.9b and Figure 4.9c respectively to clearly demonstrate the fine details in the turning maneuver.

Discussion

It is seen from Figure 4.9b and Figure 4.9c that the instantaneous tracking error before the turn is greater with longer horizon lengths than with turns at shorter horizon lengths. At $N = 20$ the vehicle did not make anticipative move and had to overshoot from the designed path. Whereas at horizon lengths greater than 20 vehicle made anticipated moves before the turn. This is because the MPC minimizes tracking error over the entire horizon resulting in overall lower cost for the test. A bar graph with the aggregate cost for whole simulation resulting form each horizon selection is shown in Figure 4.10. It was seen that as the look-ahead horizon increased from $N = 20$ to $N = 100$, simulation cost decreased from 291.80 to 67.21 (77 % decrease) and average computation time increased from $0.015s$ to $0.274s$ (1797 % increase). This test results prove how N is a trade-off between tracking performance and computation cost.

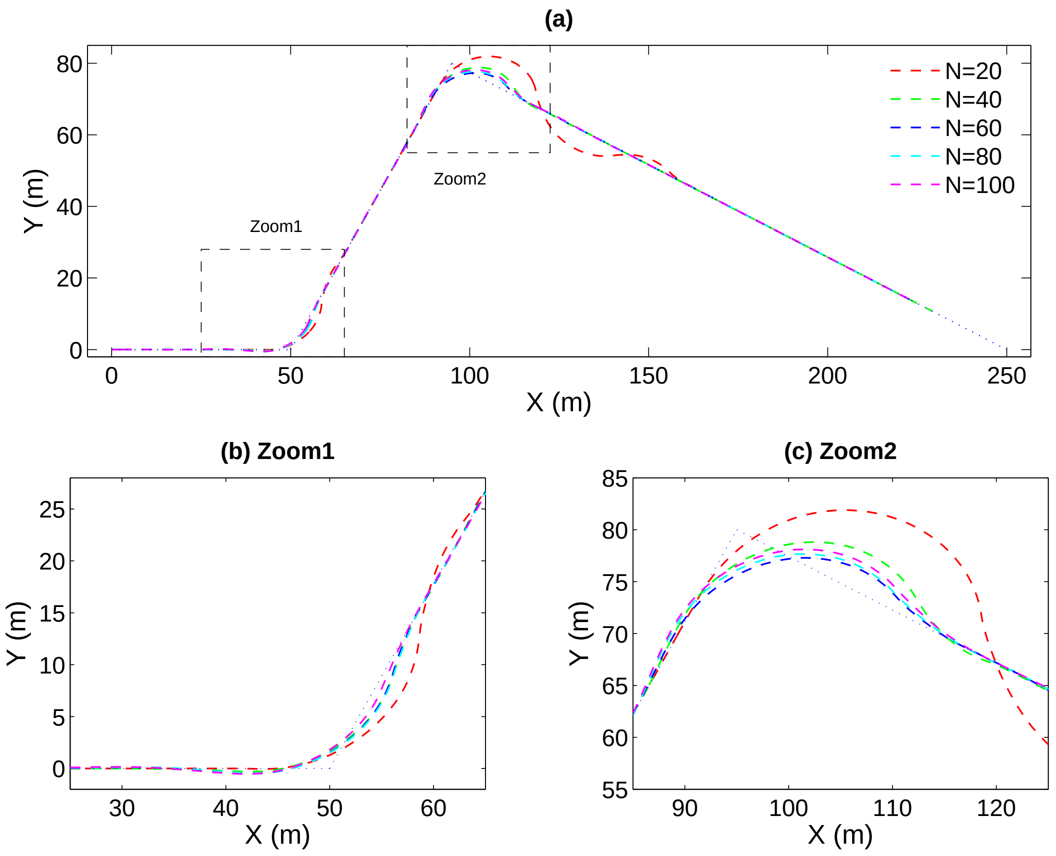
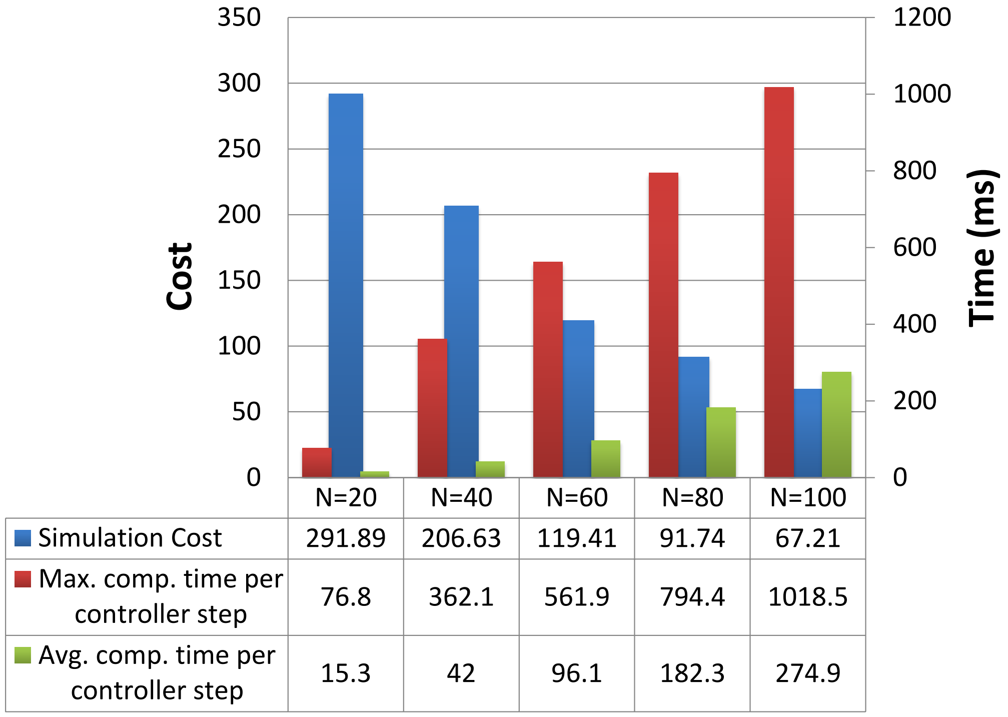


Figure 4.9: Scenario II: Anticipated Move, (a) Vehicle Trajectory, (b) Zoom1: Enlarged portion for $60\,\mathrm{deg}$ left turn, (c) Zoom2: Enlarged portion for $90\,\mathrm{deg}$ right turn



	N=20	N=40	N=60	N=80	N=100
🟦 Simulation Cost	291.89	206.63	119.41	91.74	67.21
🟥 Max. comp. time per controller step	76.8	362.1	561.9	794.4	1018.5
🟩 Avg. comp. time per controller step	15.3	42	96.1	182.3	274.9

Figure 4.10: Scenario II: Anticipated move, Bar graph comparing total cost for simulation, average computation time (in ms) per controller step and maximum computation time (in ms) per controller step

4.4 Controller Robustness Testing

It's good practice to test controller's sensitivity to prediction errors and its robustness to changes in operating conditions. Robust feasibility is a concern for MPC applications because it operates the system closer to the constraints boundary. While it is a fact that maximum performance is achieved from operation near the constraints boundary, small disturbances and prediction errors (due to model mismatch) could push the vehicle out of control. In this section some test results are presented which demonstrate the current controller's ability to withstand changes in vehicle's parameters without the need to be re-tuned. Simulations were performed with same controller tuning values but changing vehicle's mass, center of gravity, yaw inertia and track width. Vehicle was made to move on a track with varying width and two turns, two obstacles and linear obstacles as curb of the road.

4.4.1 Center of Gravity Variation

As explained in Section 3.3.2, weight transfer depends upon the height of the vehicle's center of gravity (CG). Higher the CG, more the weight transfer resulting in loss of traction. So some simulations were performed by changing the vehicle's CG value from $460mm$ to $260mm$,$660mm$ and $760mm$. It is seen from the Figure 4.11 that vehicle with $CG = 760$ rolled over at location $(50, 30)$. This is due to the fact that higher CG makes vehicle more prone to roll over due to higher weight transfer. Vehicles with all other CG heights reached endpoint safely while avoiding the objects and curb of the road. However, to deduce which test was better, the simulation cost was calculated and plotted as in Figure 4.12. It is seen that as the CG height increases, simulation cost increases. No cost exists at $CG = 760mm$ because the simulation stopped prematurely due to vehicle roll over. Figure 4.13 is a CarSim time lapse screen shot of the vehicle's roll over with higher CG height.

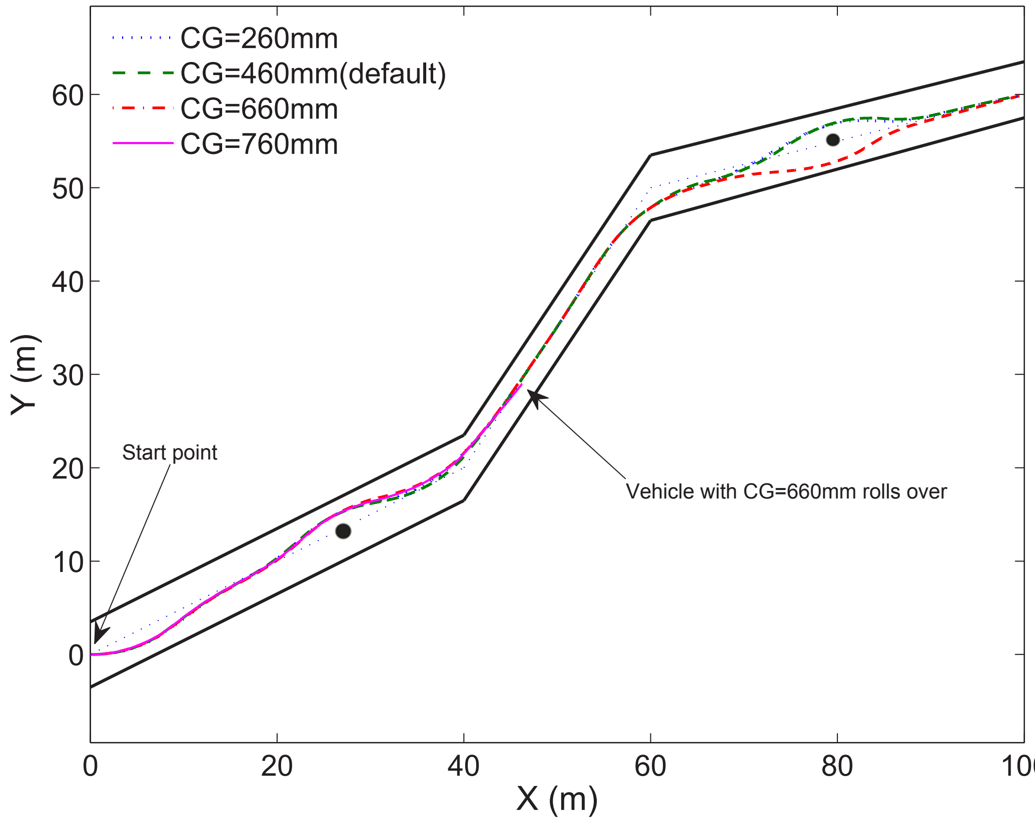


Figure 4.11: Vehicle trajectories with different center of gravity (CG) heights. Notice the vehicle roll over with $CG = 760mm$

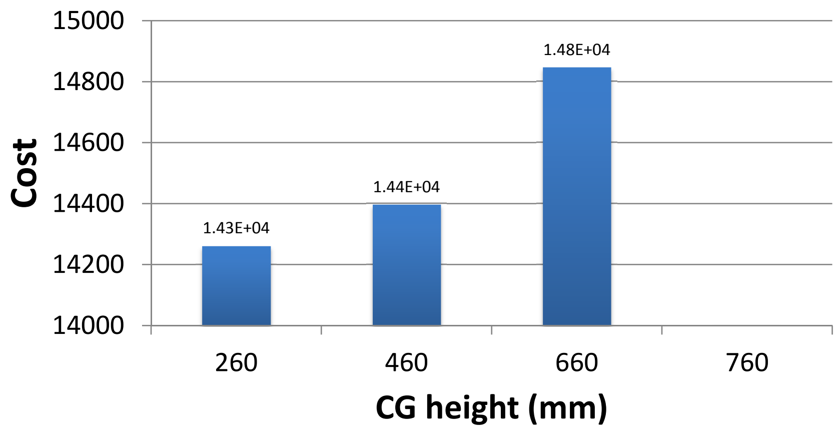


Figure 4.12: Cost of simulation with different CG heights. Notice the vehicle roll over with $CG = 760mm$ so no cost value calculated for this simulation

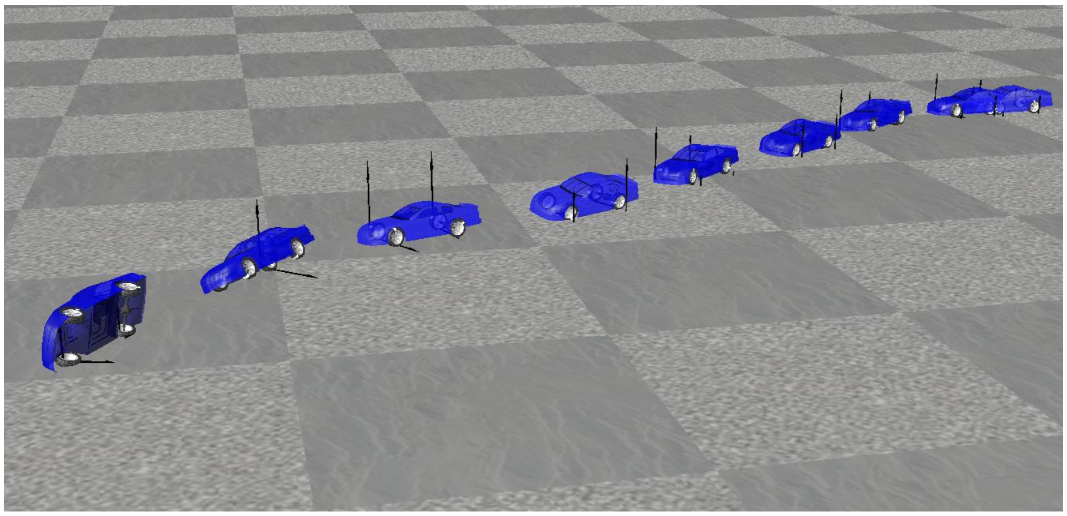


Figure 4.13: CarSim vehicle roll over scene with $CG = 760mm$

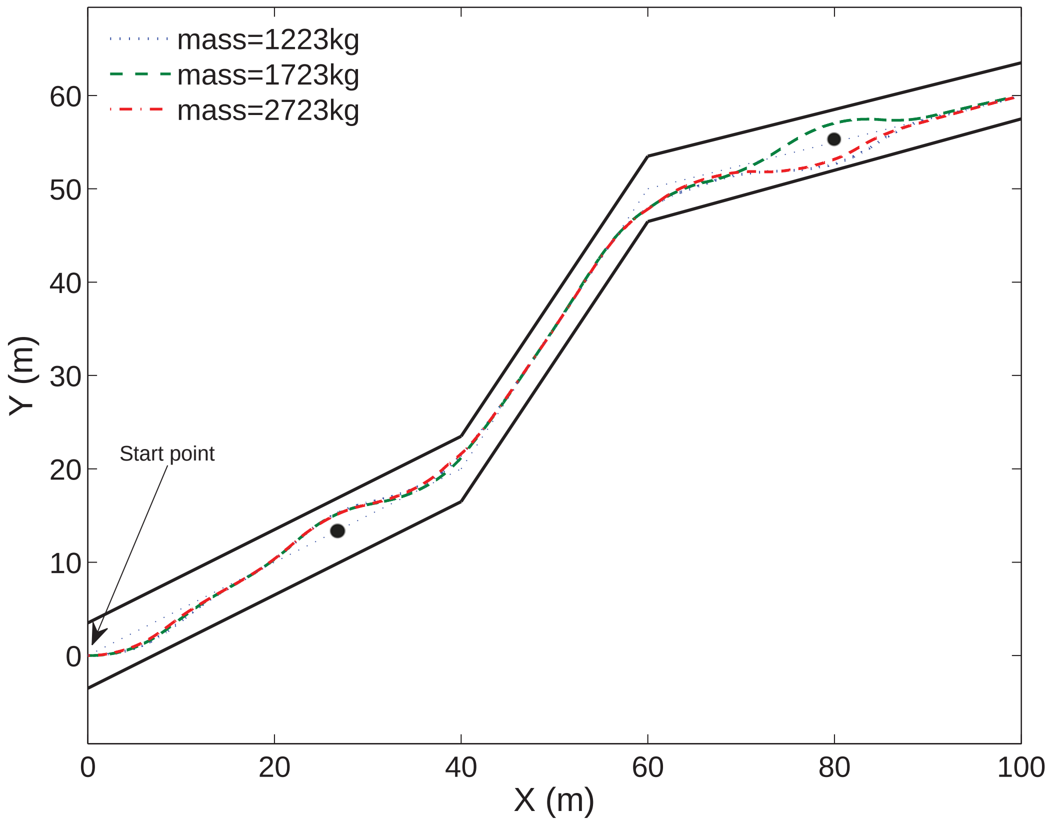


Figure 4.14: Vehicle trajectories with different mass values of the vehicle

4.4.2 Mass Variation

For this test, different values of vehicle mass were simulated to analyze controller's performance with vehicle's mass variations. Vehicle was able to steer away from obstacles and curb of the road with all the mass values used (as shown in Figure 4.14). But when vehicle was lighter ($m = 1223kg$) it narrowly escaped the curb of the road at $X = 80m$. Bar graph 4.15 shows that total cost of the simulation decreased with increasing weight. This is because tires produce more lateral force which is helpful for steering the vehicle quickly.

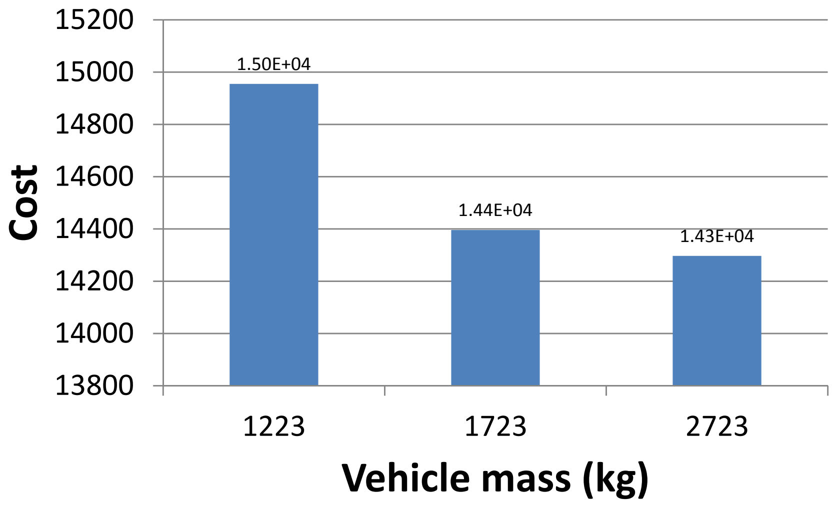


Figure 4.15: Cost of the simulation with different mass values of the vehicle

4.4.3 Track Width Variation

For third test, the vehicle track width (distance between left and right wheels) was varied and results are plotted in Figure 4.16. The vehicle with narrow track width $(1056mm)$ rolled over due to large weight transfer during cornering maneuver. Figure 4.17 shows the cost of simulation for different track widths.

4.4.4 Yaw Inertial Variation

Yaw inertia has a great effect on turning of the vehicle. Higher yaw inertia causes difficulty in sharp cornering maneuvers. Simulations were performed by changing the default yaw inertia value $4175kg-m^2$ to $2175kg-m^2$ and $6175kg-m^2$ as shown in Figure 4.18. It was observed that the cost of the simulation increased as the yaw inertia increased from its original value. Similarly, cost of the simulation decreased with decrease in yaw inertial value 4.19.

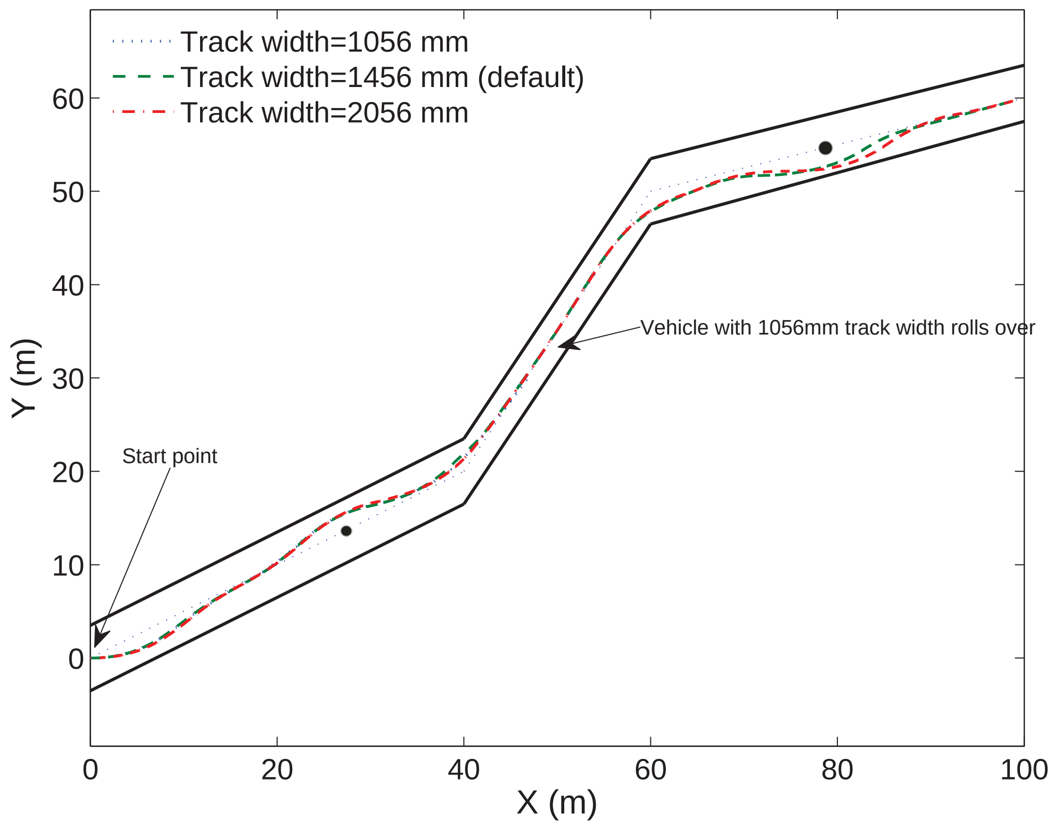


Figure 4.16: Vehicle trajectories with different track widths

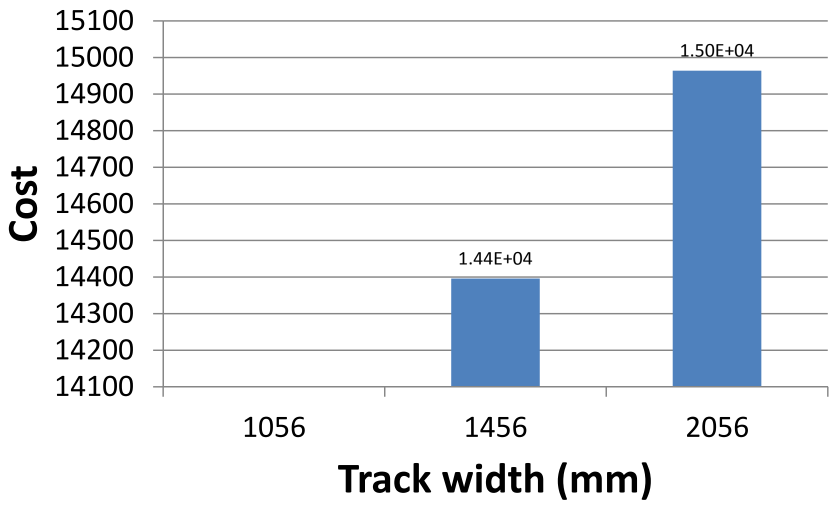


Figure 4.17: Cost of simulation with different track widths of vehicle. Notice no cost exists for track width $1056mm$ due to simulation termination because of roll over

4.5 Real-Time Analysis

For real-time implementation it is very important that the whole controller processing is completed within one controller time step. Delayed solution can destabilize the system so it is preferable to have a suboptimal solution rather that an optimal and delayed solution. Time per controller step (k) is an important performance metric which tells us weather if the NMPC algorithm is suitable for online and real-time application. It is evident from the previous simulation results that the computation time is variable and depends upon the simulation scenario. If there is an obstacle in range or the vehicle is away from the desired trajectory, computation time per controller step increases. Also it is worth mentioning that these time measurements are highly variable depending on the processor speed and presented here for reference purposes.

All simulations were performed on IBM ThinkPad Laptop Computer with Intel® Core$^{\text{TM}}$i7-2620M 2.27GHz dual core processor, 4GB RAM and running 32-bit Windows® 7 Operating System. First the effect of warm initialization (as explained in Section 3.6) on

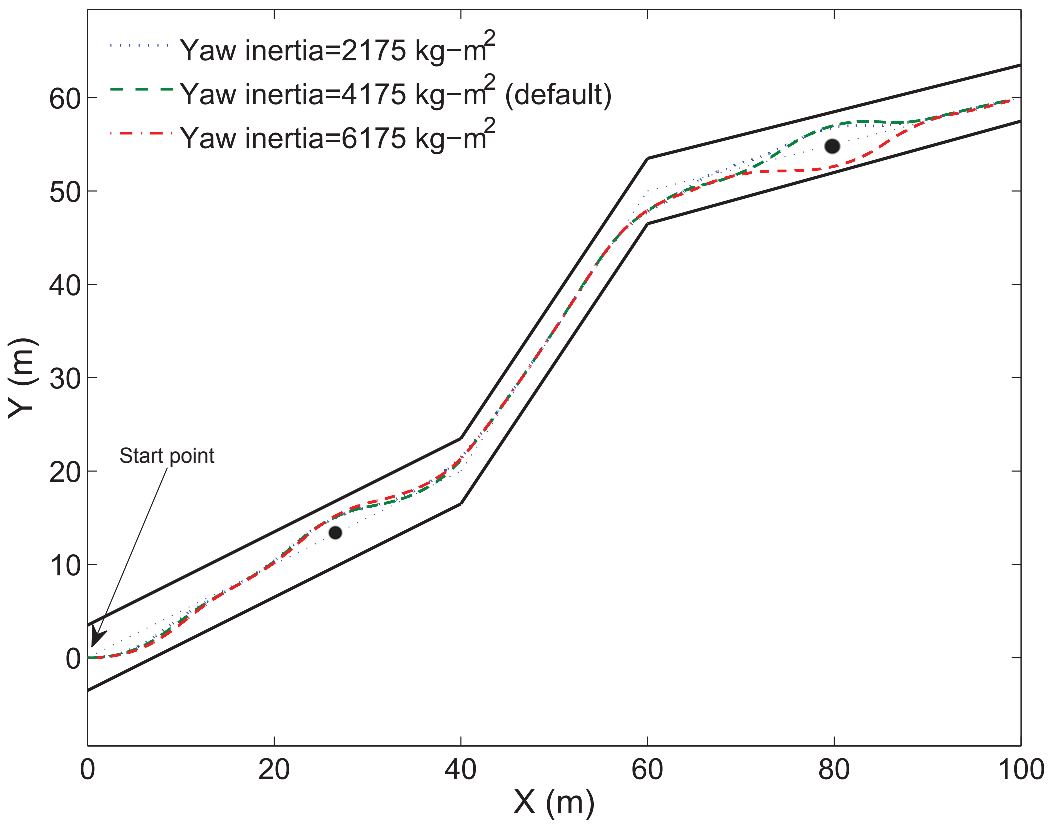


Figure 4.18: Vehicle trajectories with different yaw inertia values

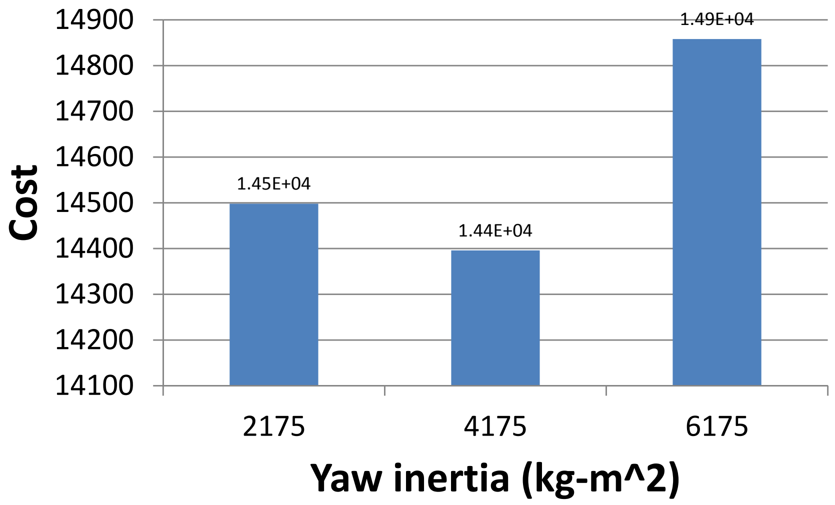


Figure 4.19: Cost of simulation with different yaw inertia values

the controller computation time is investigated. A test similar to the Figure 4.2 was performed and computation time per controller step and number of iterations per controller step were stored and plotted. Figure 4.20 is a plot of computation load for a typical simulation scenario without warm initialization of input vector u^*. It is seen that computation time exceeds the controller step size (0.05s) multiple times and average computation time is closer to the boundary.

When the same test was performed with the identical controller parameters and tuning matrices but initializing the input vector with previous step's input vector, the average computation time decreased visibly as compared to previous test.

4.5.1 Discussion

The same test was performed 10 times and average results were compared in Table 4.4. Its is evident that initializing optimization process with previous optimization results decreased iteration count significantly (32.5%) thus aiding the gradient algorithm for real-time application. Still there were occurances of the controller limit violations in warm start case.

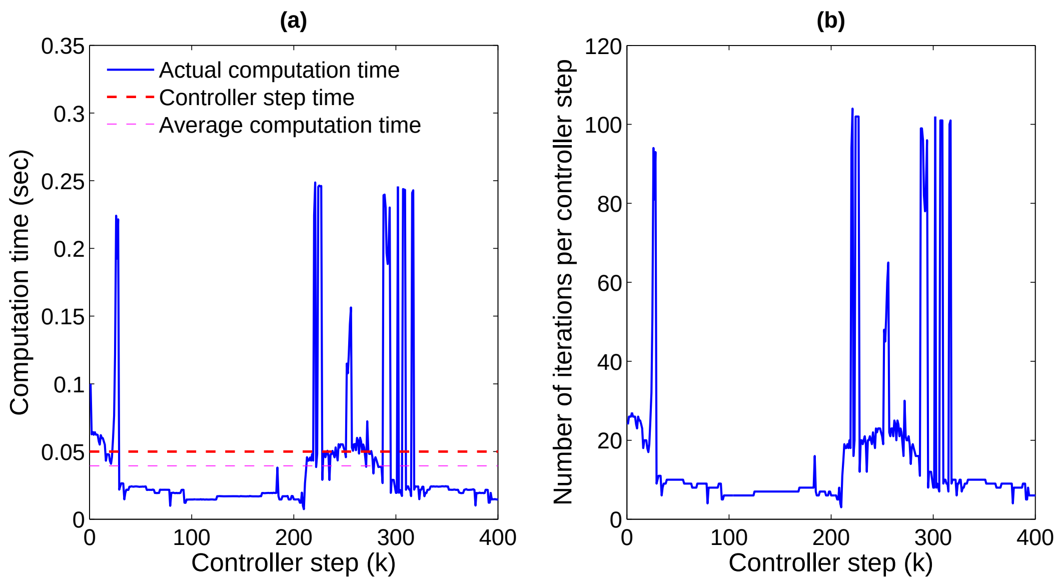


Figure 4.20: Cold start: (a) Computation time for each controller time step, (b) Number of iterations per controller step

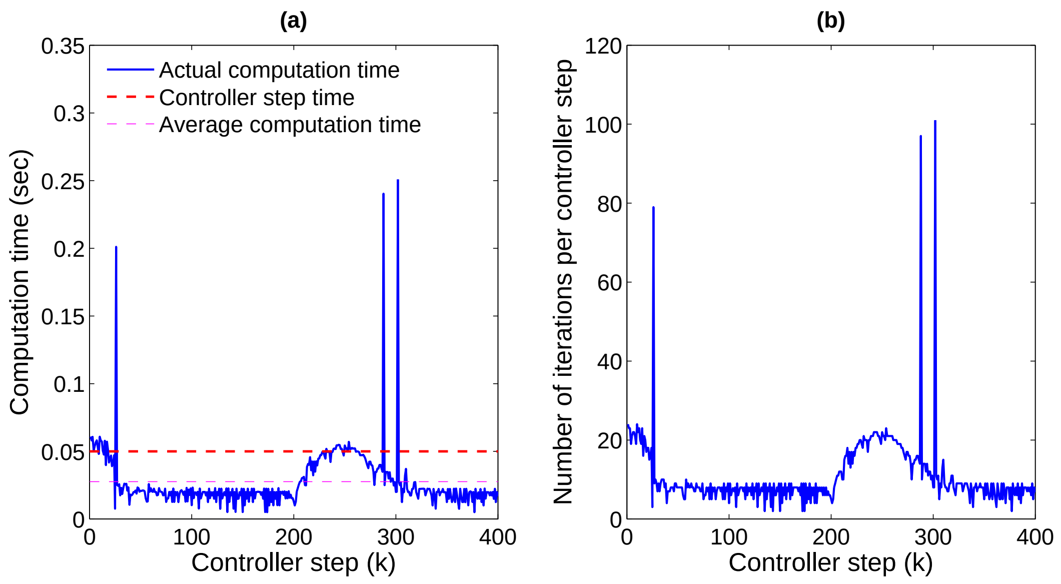


Figure 4.21: Warm start: (a) Computation time for each controller time step, (b) Number of iterations per controller step

Table 4.4: Comparison between cold start and warm start methods

Parameter	**Cold Start**	**Warm Start**	**Comparison**
Total optimizer iterations for a $20s$ run	6539	4410	32.5% decrease in iteration count
Average computation time per controller step	$0.0393s$	$0.0275s$	30% decrease in average computation time per controller step
Number of times $0.05s$ limit is violated	70	38	45% decrease in controller limit violations

Large compuation time steps occur at complicated points of the simulation such as when an object is encountered on the path or the vehicle is steering to approach the reference trajectory. These violations can be avoided by limiting the iteration count to a certain value so the optimization loop terminates as soon as that iteration number is reached. e.g., in these simulations the time for each algorithm iteration can be found by dividing the time taken for computation of each controller step by the number of iterations per controller step ($time \;\; per \;\; iteration = \frac{computation \;\; time}{iteration \;\; count}$). In this particular case the time per iteration count is found to be $0.0025s$. Which means the optimizer can iterate for a maximum of $\frac{0.05}{0.0025} = 25 \; times$ per controller time step without exceeding the $0.05s$ time ceiling. Thus if a strict real-time implementation is intended, a counter can be used which terminates the loop if iteration count increases 25 thus keeping the overall processing time within $0.05s$. However, doing so can cause stability problems the effect of which is not considered in this research and is left for future work. Also, this time calculation is true only for the current processor speed and should be recalculated for any other processor type.

Chapter 5

Conclusion and Future Work

5.1 Discussion

In this thesis, we have presented the application of Model Predictive Control to perform trajectory tracking of nonholonomic vehicles. Mathematical model of the vehicle was derived and then a nonlinear MPC framework was formulated using this model. The MPC controller was then interfaced with CarSim simulation software to perform simulations on a fully nonlinear vehicle model. Steering angle of the vehicle was constrained in the cost function formulation and the obstacle avoidance information was directly formulated inside the cost function. Operating environment of the vehicle was assumed to be unknown with various different types of the obstacles.

The controller was tested with CarSim vehicle model in different types of simulation scenarios at different speeds and varying horizon lengths. We show that proposed controller can successfully avoid the obstacles by steering away from the designed path and replan the trajectory online. Two methods of trajectory tracking i.e., the Method 1 (Section 3.4.3) and the Method 2 (Section 3.4.3) were tested in simulations.

The controller was operated at 20Hz frequency and the real-time applicability of this controller was tested in various scenarios with various horizon lengths. Discretization analysis was performed to find the suitable sampling frequency of the discrete time plant.

The robustness of the controller was tested against the variations in the vehicle parameters such as the mass, yaw inertia, track width and the center of gravity height. The effect of warm initialization on the controller's processing time was discussed. Also the effect of weight transfer on tires slip angles was explored.
Most of the relevant research papers discussed in the background research section use the same mathematical model for trajectory generation which has been used in the NMPC control formulation. This works well in the simulation environment, but for practical implementation, the controller needs to be implemented on a real vehicle (which is a nonlinear plant). So a realistic nonlinear vehicle model was used which had dynamics similar to that of a real vehicle, to validate the controller in real environment.

5.2 Conclusions

We conclude that the Method 2 (Section 3.4.3) for obstacle avoidance is superior to the Method 1 (Section 3.4.3) since it can keep the vehicle on the course even in the presence of large obstacles. We further demonstrated that the current algorithm (operated at 20Hz) is suitable for real-time application at lower speeds and shorter horizon lengths. As an increase in the horizon length causes the computation cost to increase considerably. Discrete model sampling rates around 100Hz produced exactly the same states output as the continuous time model output. However, slower sampling rates around 20Hz produced states with some small integration error. Model ouput was unstable at $1Hz$ sampling frequency.
We found that the controller is quite robust to the vehicle's parameters changes. With mistuned parameters, the vehicle was able to navigate in complex environment, but cost of the simulation increased. However, with greater parameters change, the vehicle rolled over which was due to the excessive side-weight transfer. Warm Starting the optimization resulted in 32.5% decrease in iteration count, 30% decrease in average compuattion time and a 45% decrease in the controller limit violations as compared to the Cold Start initialization

method. We conclude that the MPC controller can be systematically designed and implemented to control a nonlinear plant with constraints on the states and the control input.
Novel contribution of this thesis is the introduction of a NMPC controller to fully control a nonlinear realistic vehicle model.

5.3 Future Work

Abrupt changing of the steering angle can cause a stress on the components in a vehicle steering system. One possible future work can be the implementation of the rate of change constraints to limit the sudden variations in steering angle. Also implementing such a constraint will enhance the rider comfort level.
In this research, actuation was considered to be applied directly to the vehicle's wheels. A mathematical model of steering system can be implemented so that actuation is applied to the steering wheel itself. Doing so we can also limit the amount of torque transmitted to the wheel from the steering wheel by adjusting the steering ratio when the rate of change of the steering angle exceeds a maximum rate.
As is seen from the simulation results, the steering angle constraint has to be tightened with the increase in vehicle speed. This makes sense in practical situations as a driver tends to steer vehicle lesser at higher speeds. An automatic method can be implemented inside controller code which will tighten the steering constraint automatically as the speed changes.
During the offline trajectory generation process, a better method can be used to find efficient traversable path between two way-points by using A* algorithm which has proved to achieve better performance by the use of heuristics.
An important design metric for real-time implementation of the proposed controller is the computation time. According to MATLAB manual, MATLAB function block is slower than an S-Function block, so instead the code can be written in the form of an S-function.
Further increase in simulation speed can be achieved by writing the controller in C/C++ and

interfacing it directly with CarSim.

It has been observed that the controller computation time spikes suddenly once an obstacle is detected within the prediction horizon. These high spikes decide the worst case timing calculation for the controller. The effect of terminating the optimization loop prematurely once a large spike occurs on stability, needs further investigation.

Moving obstacles or other vehicles can be simulated to implement accident avoidance capability in MPC.

Instead of distance based method, Parallax based method [60] is known to produce better results in terms of tracking performance and computation cost. This method can be incorporated into the control formulation.

Practical hardware implementation of such an MPC controller is also a possible future work. It will be interesting to document the real-time effects of the controller when the code is executed on a dedicated hardware or a Digital Signal Processor. The controller can be used to automate the steering system of Go-Kart project at UOIT.

The controller can be tested on real-time CarSim simulator available on UOIT premises.

This thesis has contributed for autonomous vehicle research at UOIT and the developed code/methods/simulation results can be used by future students to enhance the work in this interesting field.

Bibliography

[1] B. B. A. Team, "Current Research Topics," *Accessed: November 2nd, 2011*. [Online]. Available: http://robotics.eecs.berkeley.edu/bear/current_research.html

[2] "Carsim Product Website," *Mechanical Simulations*. [Online]. Available: http://www.carsim.com/products/carsim/index.php

[3] S. N. Singh, "Model reference adaptive attitude control of spacecraft using reaction wheels," in *Decision and Control, 1986 25th IEEE Conference on*, vol. 25, dec. 1986, pp. 1514 –1519.

[4] ——, "Controlling civil infrastructures," in *Perspectives in Control Engineering Technologies, Applications, and New Directions*, 2001, pp. 417–441.

[5] S.-J. Huang and M.-T. Yan, "The adaptive control for a retrofit traditional milling machine," *Industry Applications, IEEE Transactions on*, vol. 32, no. 4, pp. 802 –809, jul/aug 1996.

[6] A. Shi, M. Yan, J. Li, W. Xu, and Y. Shi, "The research of fuzzy pid control application in dc motor of automatic doors," in *Electrical and Control Engineering (ICECE), 2011 International Conference on*, sept. 2011, pp. 1354 –1358.

[7] R. Dorf and R. Bishop, "Textbook: Modern Control Systems, 11e," *Prentice Hall*, 2008.

[8] Y. Lu and Y. Z. Lu, "Industrial intelligent control: fundamentals and applications ," *John Willey and Sons limited*, 1996.

[9] D. H. Shim, H. J. Kim, and S. Sastry, "Hierarchical control system synthesis for rotorcraft-based unmanned aerial vehicle," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, aug. 2000, pp. 14–17.

[10] K. M. Hangos, R. Lakner, and M. Gerzson, "Intelligent control systems: an introduction with examples," *Prentice Hall*, 2002. [Online]. Available: http://www.springer.com/computer/ai/book/978-1-4020-0134-5

[11] G. Saridis, "Self-organizing Control of Stochastic Systems," *M. Dekker, NY*, 1977.

[12] P. Zhou, T. Chai, and H. Wang, "Intelligent optimal-setting control for grinding circuits of mineral processing process," *Automation Science and Engineering, IEEE Transactions on*, vol. 6, no. 4, pp. 730 –743, oct. 2009.

[13] "Stochastic control theory," *McGraw-Hill Science & Technology Encyclopedia*. [Online]. Available: http://www.answers.com/topic/stochastic-control-theory

[14] M. Morari and E. Zafiriou, "Robust process control," *Prentice Hall*, 1989.

[15] D. E. Kirk, "Optimal Control Theory: An Introduction," *Dover Publications*, 2004.

[16] H. Nyquist, "Regeneration Theory," *Bell System Tech.*, 1932. [Online]. Available: http://media.johnwiley.com.au/product_data/excerpt/14/07803602/0780360214.pdf

[17] N. Nicholas, " Backlash in a velocity lag servo mechanism," *Trans. ALE, 22(2), 173-179*, 1954.

[18] H. Bode, " Network Analysis and Feedback Amplifier Design," *D. Van Nostrand Company, Inc., Princeton, N.J*, 1945. [Online]. Available: http://www.amazon.com/Network-analysis-feedback-amplifier-design/dp/0882752421

[19] N. Weiner, "Extrapolation, Interpolation and Smoothing of Stationary Time Series, with Engineering Applications," *New York Technology Press and Wiley*, Jan. 1949. [Online]. Available: http://mitpress.mit.edu/catalog/item/default.asp?ttype=2&tid=7134

[20] R. E. Kalman, "Contributions to the theory of optimal control," *Bol. Soc. Mat. Mexicana*, vol. 5, pp. 102–119, 1960.

[21] R. Kalman, "A new approach to linear filtering and prediction problems," *Journal of Basic Engineering*, vol. 82D, pp. 34–35, 1960.

[22] M. Athans and P. Falb, "Optimal Control," *McGraw-Hill (New York)*, 1966.

[23] B. Anderson and J. Moore, "Linear Optimal Control," *Prentice Hall (Englewood Cliffs, NJ)*, 1971.

[24] R. Brockett, "Optimal Control," *Wiley (New York)*, 1970.

[25] H. Kwakernaak and R. Sivan, "Linear Optimal Control Systems," *Wiley (New York)*, 1972.

[26] D. S. Naidu, "Optimal Control Systems," *CRC Press*, 2003. [Online]. Available: http://books.google.com/books/about/Optimal_control_systems.html?id=hGxurdEZVtkC

[27] J. C. Hsu and A. U. Meyer, "Modern Control Principles and Applications," *McGraw Hill, New York*, 1968. [Online]. Available: http://www.abebooks.com/9780070306356/Modern-Control-Principles-Applications-Hsu-0070306354/plp

[28] B. Chachua, "Lectures 17-18: Problem Formulation," *Optimal Control Lectures, Department of Chemical Engineering, McMaster University*, 2009. [Online]. Available: http://lawww.epfl.ch/webdav/site/la/shared/import/migration/IC_32/Slides17-18.pdf

[29] A. N. S. Goran Peskir, Albert Shiryaev, "Optimal stopping and free-boundary problems, Volume 10," *Verlag Bassel*, 2006. [Online]. Available: http://books.google.ca/books?id=UinZbLqpUDEC

[30] S. Qin and T. A. Badgwell, "A survey of industrial model predictive control technology," *Control Engineering Practice*, vol. 11, no. 7, pp. 733 – 764, 2003. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0967066102001867

[31] G. C. Goodwin, S. F. Graebe, and M. E. Salgado, "Control system design," *Englewood Cliffs, NJ: Prentice Hall.*

[32] C. E. Garca, D. M. Prett, and M. Morari, "Model predictive control: Theory and practicea survey," *Automatica*, vol. 25, no. 3, pp. 335 – 348, 1989. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0005109889900022

[33] J. Richalet, A. Rault, J. Testud, and J. Papon, "Model predictive heuristic control: Applications to industrial processes," *Automatica*, vol. 14, no. 5, pp. 413 – 428, 1978. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0005109878900018

[34] C. Cutler and B. Ramaker, "Dynamic Matrix Control - A Computer Control Algorithm,," *Automatic Control Conference, San Francisco, CA*, 1980.

[35] F. Borrelli, A. Bemporad, and M. Morari, "Predictive Control for linear and hybrid systems," *Wikipedia: Accessed 28 October 2011*, Feb. 2011. [Online]. Available: http://control.ee.ethz.ch/~stdavid/BBMbook_Cambridge_newstyle.pdf

[36] S. M. Estil, "Real-time Receding Horizon Control: Application Programmer Interface Employing LSSOL," *Master's Thesis: (Texas A&M University)*, 2002. [Online]. Available: http://jagger.berkeley.edu/papers/ken_1.pdf

[37] Y. Wang and S. Boyd, “Fast Model Predictive Control Using Online Optimization,” *Proceedings of the 17th World Congress The International Federation of Automatic Control Seoul, Korea*, no. 1, pp. 6974–6979, Jul. 2008. [Online]. Available: http://www.stanford.edu/~boyd/papers/pdf/fast_mpc_ifac.pdf

[38] J. Maciejowski, “Lecture 1 - Introduction to Predictive Control,” *Predictive Control Lectures, The University of Cambridge, UK*. [Online]. Available: http://www-control.eng.cam.ac.uk/Homepage/officialweb.php?id=1

[39] M. Behrendt, “Model predictive control,” *Wikipedia: Accessed 28 October 2011*, 2011. [Online]. Available: http://en.wikipedia.org/wiki/Model_predictive_control

[40] P. Tatjewski, “Supervisory Predictive Control and On-Line Set-Point Optimization,” *International Journal of Applied MAthematics and Computation Science*, vol. 20, no. 3, pp. 483–495, Jan. 2010. [Online]. Available: http://matwbn.icm.edu.pl/ksiazki/amc/amc20/amc2035.pdf

[41] D. Sui, L. Feng, and M. Hovd, “Algorithms for Online Implementations of Explicit MPC Solutions,” *Proceedings of the 17th IFAC World Congress,*, 2008. [Online]. Available: http://www.ifac-papersonline.net/Detailed/36328.html

[42] A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos, “The explicit linear quadratic regulator for constrained systems,” *Automatica*, vol. 38, no. 1, pp. 3 – 20, 2002. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0005109801001741

[43] T. A. Johansen, “Fast Model Predictive Control,” *MPC and Optimization*. [Online]. Available: http://www.itk.ntnu.no/ansatte/Johansen_Tor.Arne/optimization.html

[44] R. Milman and E. Davison, “A fast mpc algorithm using nonfeasible active set methods,” *Journal of Optimization Theory and Applications*, vol. 139,

pp. 591–616, 2008, 10.1007/s10957-008-9413-3. [Online]. Available: http://dx.doi.org/10.1007/s10957-008-9413-3

[45] S. Fekri and F. Assadian, "Fast model predictive control and its application to energy management of hybrid electric vehicles," *Advanced Model Predictive Control, Tao Zhend (Ed.)*, 2011, 978-953-307-298-3. [Online]. Available: http://www.intechopen.com/articles/show/title/fast-model-predictive-control-and-its-application-to-energy-management-of-hybrid-electric-vehicles

[46] M. Kvasnica and M. Fikar, "Performance-lossless complexity reduction in explicit mpc," 2010. [Online]. Available: http://www.kirp.chtf.stuba.sk/publication_info.php?id_pub=894

[47] F. Borrelli, M. Baotic, J. Pekar, and G. Stewart, "On the Complexity of Explicit MPC Laws," *Proceedings of the 2009 European Control*, no. 1, pp. 2408–2413, 2009. [Online]. Available: http://www.mendeley.com/research/complexity-explicit-mpc-laws-2/

[48] K. Kouramas, N. Fasca, C. Panos, and E. Pistikopoulos, "Explicit/multi-parametric model predictive control (mpc) of linear discrete-time systems by dynamic and multi-parametric programming," *Automatica*, vol. 47, no. 8, pp. 1638 – 1645, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S000510981100255X

[49] M. Kvasnica, P. Grieder, and M. Baotić, "Multi-Parametric Toolbox (MPT)," 2004. [Online]. Available: http://control.ee.ethz.ch/~mpt/

[50] B. Genuit, W. Heemels, and L. Lu, "Approximation of Explicit MPC Using Regular PWA Functions: An ISS Approach," *Implementation of Feedback Controllers on*

Special-Purpose Hardware in IET Control Theory & Applications, 2011. [Online]. Available: http://www.dct.tue.nl/New/Heemels/GenHee_IET_12.pdf

[51] A. Divelbiss and J. Wen, "Trajectory tracking control of a car-trailer system," *IEEE Transactions on Control Systems Technology*, pp. 269–278, Aug. 2002.

[52] J. Wen and S. Jung, "Nonlinear model predictive control based on predicted state error convergence," *American Control Conference, 2004. Proceedings of the 2004*, pp. 2227–2231, Jul. 2005. [Online]. Available: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=1383793

[53] P. Tsiotras, "New control laws for the attitude stabilization of rigid bodies," in *in 13th IFAC Symposium on Automatic Control in Aerospace*, pp. 316–321.

[54] G. Sutton and R. Bitmead, "Performance and Computational Implementation of Nonlinear Model Predictive Control on a Submarine,," *Book:Nonlinear Model Predictive Control (F. Allgower and A. Zheng)*, pp. 461–472, 2000.

[55] H. Kim, D. Shim, and S. Sastry, "Nonlinear model predictive tracking control for rotorcraft-based unmanned aerial vehicles," in *American Control Conference, 2002. Proceedings of the 2002*, vol. 5, 2002, pp. 3576 – 3581 vol.5.

[56] J. M. Eklund, J. Sprinkle, and S. S. Sastry, "Switched and symmetric pursuit/evasion games using online model predictive control with application to autonomous aircraft," *Control Systems Technology, IEEE Transactions on*, vol. PP, no. 99, pp. 1 –17, 2011.

[57] F. Fahimi, "Non-linear model predictive formation control for groups of autonomous surface vehicles," *International Journal of Control*, no. 8, pp. 1248–1259, Aug. 2007.

[58] S. Vougioukas, N. Sigrimis, C. Arvanitis, and Y. Ampatzidis, "Nonlinear Model Predictive Path Tracking for Precision Guidance," *Proceedings of the XVI*

CIGR World Congress (International Commission of Agricultural Engineering), 2006. [Online]. Available: http://users.auth.gr/iampatzi/Vougioukas_Ampatzidis_CIGR2006.pdf

[59] S. Vougioukas, "Reactive Trajectory Tracking for Mobile Robots based on Non Linear Model Predictive Control," *2007 IEEE International Conference on Robotics and Automation*, pp. 3074–3079, May 2007. [Online]. Available: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=4209557

[60] Y. Yoon, T. Choe, Y. Park, and H. J. Kim, "Obstacle Avoidance for Wheeled Robots in Unknown Environments Using Model Predictive Control," *Proceedings of the 17th IFAC World Congress*, 2000. [Online]. Available: http://www.ifac-papersonline.net/Detailed/36853.html

[61] F. Allgower and A. Zheng, *Nonlinear model predictive control.* Basel, Switzerland: Birkhauser Basel, 2000. [Online]. Available: http://www.amazon.ca/Nonlinear-Model-Predictive-Control-Allgower/dp/3764362979

[62] D. Rowell, "State-Space Representation of LTI Systems," *Analysis and Design of Feedback systems*, Oct. 2002. [Online]. Available: http://web.mit.edu/2.14/www/Handouts/StateSpace.pdf

[63] B. Friedland, *Control System Design: An Introduction to State-Space Methods.* Newyork, USA: McGraw-Hill College, 1985. [Online]. Available: http://www.abebooks.com/9780070224414/Control-System-Design-Introduction-State-Space-0070224412/plp

[64] A. M. Bloch, J. E. Marsden, and D. V. Zenkov, "Nonholonomic dynamics," in *Control of nonholonomic systems on Riemannian manifolds. PROCEEDINGS OF NOLCOS '92*, 2005.

[65] G. Oriolo, "Control of Nonholonomic System," *Universita di Roma*, 2002. [Online]. Available: http://www.dis.uniroma1.it/~oriolo/cns/cns_slides.pdf

[66] M. Segel, "Theoretical prediction and experimental substantiation of the response of the automobile to steering control," *Proceedings of Automobile division of the institute of mechanical engineers*, vol. 7, pp. 310–330, 1950. [Online]. Available: http://archive.pepublishing.com/content/5r02783355x67525/

[67] J. Kasselmann and T. Keranen, "Adaptive steering," *Bendix Technical Journal*, vol. 2, pp. 26–35, 1969. [Online]. Available: http://archive.pepublishing.com/content/5r02783355x67525/

[68] D. Wang and F. Qi, "Trajectory Planning for a Four-Wheel-Steering Vehicle," *Proc. IEEE International Conference on Robotics and Automation*, May 2001. [Online]. Available: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=933130

[69] M. Wise and J. Hsu, "Application and analysis of a robust trajectory tracking controller for under-characterized autonomous vehicles," in *Control Applications, 2008. CCA 2008. IEEE International Conference on*, sept. 2008, pp. 274 –280.

[70] D. L. Margolis and J. Asgari, "Multipurpose Models of Vehicle Dynamics for Controller Design," *Passenger Car Conference & Exposition, Society of Automotive Engineers*, Sep. 1991. [Online]. Available: http://papers.sae.org/911927/

[71] P. Falcone, F. Borrelli, E. H. Tseng, and D. Hrovat, "On Low Complexity Predictive Approaches to Control of Autonomous Vehicles," *Lecture Notes in Control and Information Sciences*, vol. 402, pp. 195–210, 2010. [Online]. Available: http://www.springerlink.com/content/g15713711l540j34/

[72] 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://pid.sagepub.com/content/223/12/1499.short?patientinform-links=yes&legid=sppid;223/12/1499

[73] R. Rajamani, *Vehicle Dynamics and Control*. Birkhauser, 2006. [Online]. Available: http://www.amazon.com/Vehicle-Dynamics-Control-Mechanical-Engineering/dp/0387263969

[74] R. Andrzejewski and J. Awrejcewicz, *Nonlinear dynamics of a wheeled vehicle*. Newyork, USA: Springer, 2005, vol. 10. [Online]. Available: http://books.google.com/books?id=PjphGI1_eJYC&dq=Wheel+UK&ie=ISO-8859-1&source=gbs_gdata

[75] H. T. Zostak, W. R. Allen, and Rosenthal, "Analytical modeling of driver response in crash avoidance maneuvering volume ii: An interactive model for driver/vehicle simulation," in *U.S Department of Transportation Report NHTSA DOT HS-807-271*, apr. 1988. [Online]. Available: http://code.eng.buffalo.edu/dat/sites/tire/tire.html

[76] J. Svedeniusi, "Tire Modeling and Friction Estimation," *Department of Automatic Control, Lund University, Lund, Sweden*, Apr. 2007. [Online]. Available: http://www2.control.lth.se/index.php?mact=ReglerPublications,cntnt01,sendfull,0&cntnt01artkey=jsvenPDH&cntnt01year=2007&cntnt01returnid=60

[77] E. Bakker, L. Nyborg, and H. Pacejka, "Tyre modeling for use in vehicle dynamics studies," *Society of Automotive Engineers, paper No. 870421*, Jan. 1987. [Online]. Available: http://www.osti.gov/energycitations/product.biblio.jsp?osti_id=6514971

[78] F. Borrelli, P. Falcone, T. Keviczky, J. Asgari, and D. Hrovat, "MPC-based approach to active steering for autonomous vehicle systems," *International Journal of Vehicle*

Autonomous Systems, vol. 3, no. 2/3/4, pp. 265–291, 2005. [Online]. Available: http://www.inderscience.com/search/index.php?action=record&rec_id=8237

[79] P. Falcone, "Nonlinear Model Predictive Control for Autonomous Vehicles," *PhD thesis, Universita del Sannio, Dipartimento di Ingegneria, Piazza Roma 21, 82100, Benevento, Italy*, Jun. 2007. [Online]. Available: http://www.grace.ing.unisannio.it/thesisController.php?command=home&id=423

[80] E. Bakker, L. Nyborg, and H. Pacejka, "Predictive active steering control for autonomous vehicle systems," *IEEE Transactions on Control System Technology*, vol. 15, no. 1, pp. 566–580, May 2007. [Online]. Available: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=4162483

[81] Online, "Slip Angles and Handling," *Suspension Geometry/ Tracking*. [Online]. Available: http://www.mgf.ultimatemg.com/group2/suspension/chassis_and_handling/slip_angle.htm

[82] T. D. Gillespie, *Fundamentals of Vehicle Dynamics*. Michigan, USA: SAE International, Feb. 1992. [Online]. Available: http://books.sae.org/book-r-114

[83] R. Stone and J. K. Ball, *Automotive Engineering Fundamentals*. Warrendale, Pa: SAE International, 2004. [Online]. Available: http://books.sae.org/book-r-199

[84] "Vehicle Dynamics Terminology," *SAEJ670e Society of Automotive Engineers*, 1976. [Online]. Available: http://standards.sae.org/j670_200801/

[85] Online, "4d SimRacing Blog," *Slip curves*. [Online]. Available: http://f1elites.com/4d/?p=54

[86] C. M. Macal, "Model Verification and Validation," *Workshop on Threat Anticipation: Social Science Methods and Models*, 2005. [Online]. Available: http://jtac.uchicago.edu/conferences/05/resources/V&V_macal_pres.pdf

[87] D. Wu, Q. Zhang, J. Reid, H. Qiu, and E. Benson, “Model recognition and simulation of an e/h steering controller on off-road equipment,” in *Department of Agricultural Engineering University of Illinois at Urbana-Champaign*, 2008.

[88] I. Newton, “The Principia, A new translation by I.B. Cohen and A. Whitman,” *University of California press, Berkeley*, 1999.

[89] R. C. Hibbeler, *Engineering Mechanics: Dynamics (12 ed.)*, 12th ed. Prentice Hall, 2009.

[90] GT5mania.com, “Sideways Weight Transfer,” *Accessed: 06 October, 2011*. [Online]. Available: http://gt5mania.com/driving-guide/sideways-weight-transfer/

[91] “Weight Transfer,” *Articles about Handling*. [Online]. Available: http://www.turnfast.com/tech_handling/handling_weightxfr

[92] W.-M. Lu and D. S. Bayard, “Guidance and Control for Mars Atmospheric Entry: Adaptivity and Robustness,” *NASA Technical Documents*, 1997. [Online]. Available: http://www.archive.org/details/nasa_techdoc_20000060812

[93] R. Smith, “Aeromaneuvering entry in the martian atmosphere; an actuated pathfinder vehicle study,” Tech. Rep. CCEC-97-0528, 1997.

[94] R. Smith, D. Boussalis, and F. Y. Hadaegh, “Closed-Loop Aeromaneuvering for a Mars Precision Landing,” *NASA University Research Centers Technical Advances in Education, Aeronautics, Space, Autonomy, Earth and Environment, vol. 1, p. 942*, vol. 1, pp. 942–+, Feb. 1997.

[95] F. Haugen, “Discretization of simulator, filter, and PID controller,” *TechTeach technical publication*, May 2010. [Online]. Available: http://techteach.no/publications/articles/discretization/discretization.pdf

[96] D. Joosten, T. van den Boom, and M. Verhaegen, “Fault-tolerant control through a synthesis of model-predictive control and nonlinear inversion,” vol. 399, pp. 319–336, 2010. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-11690-2_11

[97] W. Xi and J. S. Baras, “Mpc based motion control of car-like vehicle swarms,,” *Mediterranean Conference on Control and Automation, Athens - Greece,*, 2007.

[98] Y. Yoon, J. Shin, H. J. Kim, Y. Park, and S. Sastry, “Model-predictive active steering and obstacle avoidance for autonomous ground vehicles,” *Control Engineering Practice*, vol. 17, no. 7, pp. 741 – 750, 2009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0967066108002025

[99] H. Su and T. Samad, “Chapter 10 - neuro-control design: Optimization aspects,” in *Neural Systems for Control*, O. Omidvar and D. L. Elliott, Eds. San Diego: Academic Press, 1997, pp. 259 – 288. [Online]. Available: http://www.sciencedirect.com/science/article/pii/B9780125264303500118

[100] A. E. Bryson and Y.-C. Ho, “Applied optimal control,” *Waltham, MA Blaisdell*, 1969. [Online]. Available: http://books.google.com/books/about/Applied_optimal_control.html?id=P4TKxn7qW5kC

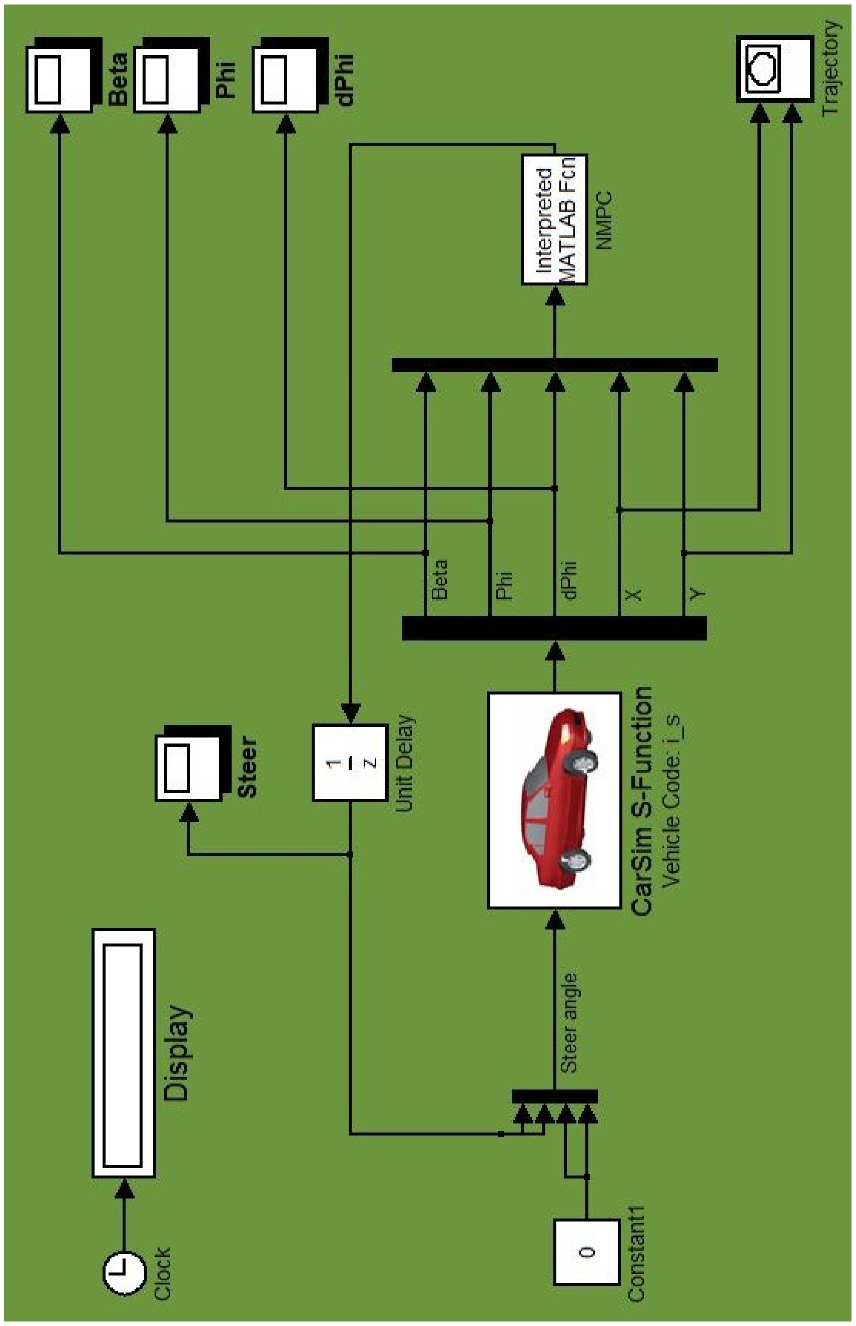
[101] R. Mehra and R. Davis, “A generalized gradient method for optimal control problems with inequality constraints and singular arcs,” *IEEE Transactions on Automatic Control*, no. 1, pp. 69–79, Feb. 1972. [Online]. Available: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=1099881

[102] G. J. Sutton and R. R. Bitmead, “Performance and computational implementation of nonlinear model predictive control on a submarine,” in *Nonlinear Model Predictive Control*, ser. Progress in Systems and Control Theory, F. Allgwer, A. Zheng,

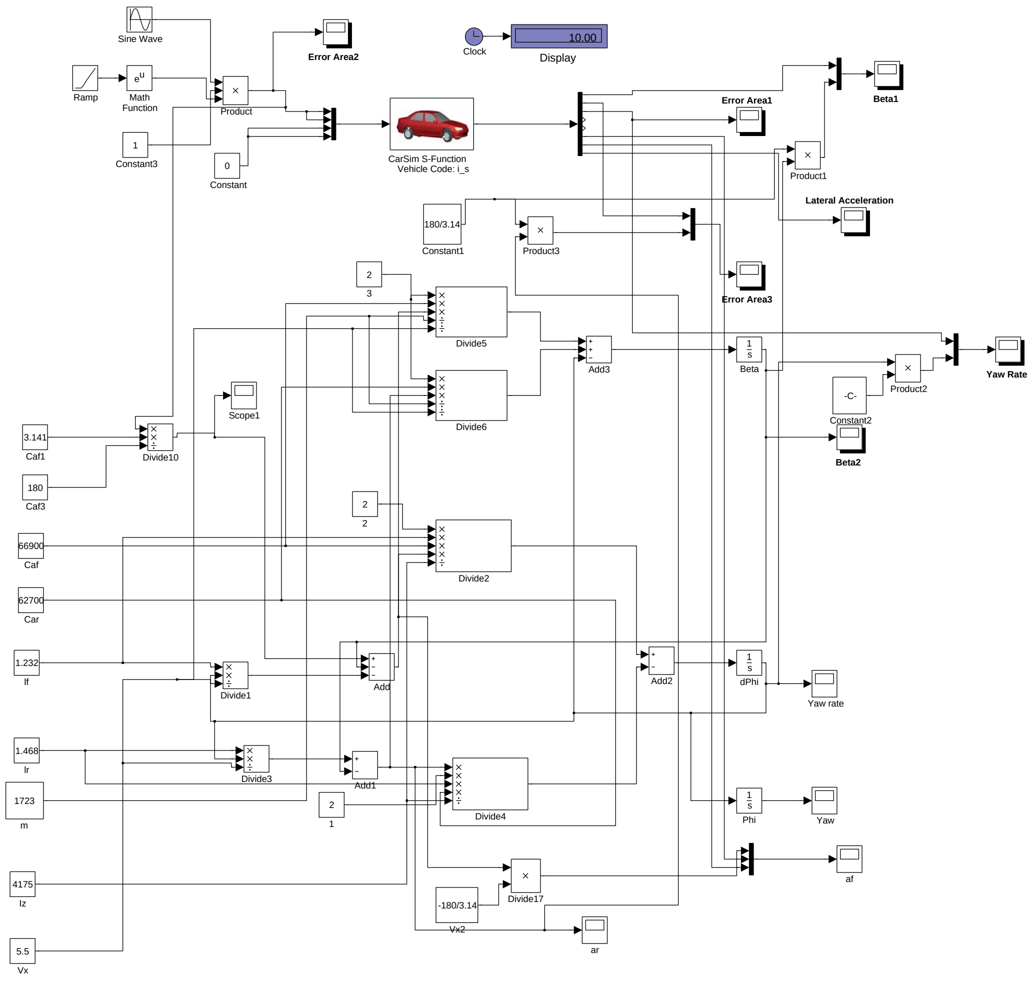
and C. I. Byrnes, Eds. Birkhuser Basel, 2000, vol. 26, pp. 461–472. [Online]. Available: http://dx.doi.org/10.1007/978-3-0348-8407-5_27

Appendix A

Simulink Model



Simulink model with CarSim S-function and MATLAB based NMPC controller



Mathematical model validation against CarSim vehicle model

Appendix B

Detailed Vehicle Model Equations

State vector = $[\beta \ \ \psi \ \ \dot{\psi} \ \ X \ \ Y]$

where

β= Vehicle side slip angle

ψ= Vehicle roll angle

$\dot{\psi}$= Vehicle roll rate

X= X-coordinate of vehicle's C.G.

Y= Y-coordinate of vehicle's C.G.

B.1 Continuous-time equations

$$\dot{\beta} = \frac{2C_f}{mv_x}[\delta_f - \beta - \frac{l_f\dot{\psi}}{v_x}] + \frac{2C_r}{mv_x}[-\beta + \frac{l_r\dot{\psi}}{v_x}] - \dot{\psi} \tag{B.1}$$

$$\dot{\psi} = \dot{\psi} \tag{B.2}$$

$$\ddot{\psi} = \frac{2l_fC_f}{I_z}[\delta_f - \beta - \frac{l_f\dot{\psi}}{v_x}] - \frac{2C_rl_r}{I_z}[-\beta + \frac{l_r\dot{\psi}}{v_x}] \tag{B.3}$$

$$\dot{X} = v_x \cos\psi - v_x \tan\beta \sin\psi \tag{B.4}$$

$$\dot{Y} = v_x \sin\psi - v_x \tan\beta \cos\psi \tag{B.5}$$

B.2 Discretized equations

$$\beta(k+1) = \beta(k) + T_s * [\frac{2C_f}{mv_x}[\delta_f(k) - \beta(k) - \frac{l_f\dot{\psi}(k)}{v_x}] + \frac{2C_r}{mv_x}[-\beta(k) + \frac{l_r\dot{\psi}(k)}{v_x}] - \dot{\psi}(k)] \tag{B.6}$$

$$\psi(k+1) = \psi(k) + T_s * \psi(k) \tag{B.7}$$

$$\dot{\psi}(k+1) = \psi(k) + T_s * [\frac{2l_fC_f}{I_z}[\delta_f(k) - \beta(k) - \frac{l_f\dot{\psi}(k)}{v_x}] - \frac{2C_rl_r}{I_z}[-\beta(k) + \frac{l_r\dot{\psi}(k)}{v_x}]] \tag{B.8}$$

$$X(k+1) = X(k) + T_s * [v_x \cos\psi(k) - v_x \tan\beta(k) \sin\psi(k)] \tag{B.9}$$

$$Y(k+1) = Y(k) + T_s * v_x \sin\psi - v_x \tan\beta(k) \cos\psi(k) \tag{B.10}$$