A MODEL PREDICTIVE CONTROL APPROACH TO ROLL STABILITY OF A SCALED CRASH AVOIDANCE VEHICLE

A Thesis

presented to

The Faculty of California Polytechnic State University

San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Mechanical Engineering

by

Nikola John Linn Noxon

May 2012

©2012
Nikola John Linn Noxon
ALL RIGHTS RESERVED

COMMITTEE MEMBERSHIP

TITLE:	A Model Predictive Control Approach to Roll Stability of a Scaled Crash Avoidance Vehicle
AUTHOR:	Nikola John Linn Noxon
DATE SUBMITTED:	June 2012
COMMITTEE CHAIR:	Dr. Charles B. Birdsong
COMMITTEE MEMBER:	Dr. William R. Murray
COMMITTEE MEMBER:	Dr. Xiao-Hua Yu

ABSTRACT

A Model Predictive Control Approach to Roll Stability of a Scaled Crash Avoidance Vehicle

Nikola John Linn Noxon

In this paper, a roll stability controller (RSC) is presented based on an eight degree of freedom dynamic vehicle model. The controller is designed for and tested on a scaled vehicle performing obstacle avoidance maneuvers on a populated test track. A rapidly-exploring random tree (RRT) algorithm is used for the vehicle to execute a trajectory around an obstacle, and examines the geographic, non-homonymic, and dynamic constraints to maneuver around the obstacle. A model predictive controller (MPC) uses information about the vehicle state and, based on a weighted performance measure, generates an optimal trajectory around the obstacle. The RSC uses the standard vehicle state sensors: four wheel mounted encoders, a steering angle sensor, and a six degree of freedom inertial measurement unit (IMU). An emphasis is placed on the mitigation of rollover and spin-out, however if a safe maneuver is not found and a collision is inevitable, the program will run a brake command to reduce the vehicle speed before impact. The trajectory is updated at a rate of 20 Hz, providing improved stability and maneuverability for speeds up to 10 ft/s and turn angles of up to 20°.

ACKNOWLEDGEMENTS

Special thanks to Dr. Charles Birdsong, Dr. Saeed Niku, Dr. John Fabijanic, Dr. William Murray, and Dr. Helen Yu for their guidance in this project. Thank you to Sean Hirtle for laying the groundwork of the dynamic vehicle model. And finally, thank you to Brian Ujiie, Josh Miller, and Gordon Woods for their work on the crash avoidance vehicle and the rapidly-exploring random tree algorithm.

TABLE OF CONTENTS

Page

LIST OF FIGURES viii

NOMENCLATURE xii

Vehicle Model Nomenclature xii

Model Predictive Controller nomenclature xiii

I. INTRODUCTION 1

Current Technologies 1

II. PROJECT DESCRIPTION 3

Hardware Layout 3

Software Layout 6

III. RAPIDLY-EXPLORING RANDOM TREE PATH PLANNER 7

RRT Navigation Procedure 8

Pure Pursuit Path Follower 9

IV. DEVELOPMENT OF DYNAMIC VEHICLE MODEL 11

Existing Vehicle Models 11

Coordinate System Definition 14

Linear Equations of Motion 16

Angular Equations of Motion 22

Dynamic Weight Transfer Forces 26

Tire Models 27

DVM Validation 30

Summary and Comparison to NAVDyn Model 32

V. SYSTEM IDENTIFICATION OF THE TEST VEHICLE 34

Sprung Center of Mass and the Static Roll Center 35

Vehicle Dimensions and Inertial Properties 37

Spring Stiffness and Shock Damping 39

Cornering Coefficient 39

VI. MODEL PREDICTIVE CONTROL 41

State Space Representation of the Dynamic Vehicle Model 42

Linearization of the Dynamic Vehicle Model 46

Discretization of the Dynamic Vehicle Model 49

The MPC Cost Function 50

Constraints and Constraint Softening........51
MPC Optimization Formulation........52
Quadratic Programming (QP) Solver........52
Stability Considerations due to Decoupling........53
VII. SIMULATION RESULTS........55
Comments on the RRT and MPC Integration........58
Single-Lane Change Maneuver........60
Simple Steer Maneuver........61
RRT Path Planning Maneuver........63
Comparison to PD Control........65
VIII. EXPERIMENTAL RESULTS........67
Single-Lane Change Maneuver........67
Simple Turn Maneuver........68
RRT Path Planning Maneuver........70
IX. CONCLUSIONS........73
X. RECOMMENDATIONS........74
Hardware Limitations and Recommendations........74
Software Limitations and Recommendations........76
BIBLIOGRAPHY........78
APPENDIX A: ONBOARD COMMUNICATIONS........81
APPENDIX B: DEVELOPMENT BOARD SCHEMATIC........90
APPENDIX C: RAPIDLY-EXPLORING RANDOM TREE – MATLAB/SIMULINK STRUCTURE........92
APPENDIX D: TEST VEHICLE PHYSICAL PARAMETERS FOR DVM........104
APPENDIX E: TEST VEHICLE MATERIAL PROPERTIES........106

LIST OF FIGURES

Figure 1: Rollover related fatalities over the last 30 years have been steadily increasing, highlighting a need for improvements in rollover mitigation technology 1

Figure 2: Experimental Setup 4

Figure 3: R/C Vehicle used for experimental validation 5

Figure 4: System flow diagram for the stability control of an autonomous crash avoidance vehicle 6

Figure 5: RRT nodes and branches mapping acceptable paths (green) and marking failed paths (red) in the driving plane. 7

Figure 6: Pure Pursuit path follower 9

Figure 7: Bicycle Model 11

Figure 8: CarSim running a simulation. 12

Figure 9: Global and chassis coordinate frames 16

Figure 10: Vehicle frame with pertinent vehicle parameters, forces, and velocities. 18

Figure 11: Symmetric and asymmetric planes of the vehicle 23

Figure 12: Longitudinal weight transfer force. 26

Figure 13: The wheel slip angle is a function of the longitudinal and cornering speeds of the wheel 28

Figure 14: Example cornering force curve of the Magic Formula (B=0.5, C=2.5, D=1, E=1) 29

Figure 15: Sprung mass lateral acceleration and yaw rate vs. time for a 42 degree steering wheel step turn at 40 kph. 30

Figure 16: Sprung mass lateral acceleration and yaw rate vs. time for a 142 degree steering wheel step turn at 40 kph. 31

Figure 17: Roll angle and roll rate vs. time for a 142 degree steering wheel step at 40 kph. 31

Figure 18: SolidWorks model of the test vehicle. 34

Figure 19: Location of the test vehicle sprung CM in the X-Z plane 35

Figure 20: Roll center determination of the rear suspension of the test vehicle. 36

Figure 21: Roll center of the rear suspension of the test vehicle 36

Figure 22: Vehicle dimensions used for the DVM 37

Figure 23: Sprung mass model used to determine the moments and products of inertia. 38

Figure 24: Comparison of the dynamic and static lateral front left tire forces 44

Figure 25: Comparison of the dynamic and static lateral front right tire forces 44

Figure 26: Comparison of the dynamic and static lateral rear left tire forces 45

Figure 27: Comparison of the dynamic and static lateral rear right tire forces 45

Figure 28: DVM vs LTI roll angle response to a step turn of 20° 47

Figure 29: DVM vs LTI roll rate response to a step turn of 20° 48

Figure 30: DVM vs LTI roll angle response to a step turn of 20° 48

Figure 31: Torque about the roll axis due to roll 59

Figure 32: Steer signal and MPC signal for the double-lane change manuever 60

Figure 33: X-Y path for the single-lane change maneuver...61

Figure 35: Steer signal and MPC signal for the simple-steer manuever...62

Figure 36: X-Y path for the simple turn maneuver 62

Figure 37: Roll angle response to MPC control for the double-lane change manuever 63

Figure 38: Steer signal and MPC signal for the RRT manuever 63

Figure 39: X-Y path of the vehicle in the RRT maneuver 64

Figure 40: Roll angle response to MPC control for the RRT manuever 64

Figure 41: Steer signal and PD signal for the RRT manuever..65

Figure 42: Roll angle response to PD control for the RRT manuever 66

Figure 43: Comparison of MPC and PD control efforts..66

Figure 44: Steer angle command in a single-lane change maneuver..67

Figure 45: X-Y Trajectory of the test vehicle for the single-lane change maneuver 68

Figure 46: Roll angle response in a double-lane change manuever..68

Figure 47: Steer angle command in a simple turn maneuver 69

Figure 48: X-Y trajectory of the test vehicle for the simple-turn maneuver...69

Figure 49: Roll angle response in a simple turn manuever..70

Figure 51: X-Y trajectory of the test vehicle in an RRT maneuver..71

Figure 52: Roll angle response in an RRT path planning manuever 71

Figure 53: Yaw rate response in an RSC RRT path planning manuever...72

Figure 54: Example of an experimental apparatus to determine the damping coefficient.............................76

Figure 55: MCU1 schematic 90

Figure 56: MCU2 schematic 91

NOMENCLATURE

Vehicle Model Nomenclature

$\dot{x}, \dot{y}$	Longitudinal and lateral velocity in the vehicle frame
$\ddot{x}, \ddot{y}$	Longitudinal and lateral acceleration in the vehicle frame
ϕ, ψ	Roll and yaw angle in the vehicle frame
$\dot{\phi}, \dot{\psi}$	Roll and yaw rate in the vehicle frame
$\ddot{\phi}, \ddot{\psi}$	Roll and yaw acceleration in the vehicle frame
a_{sy}	Lateral acceleration of sprung mass about CG
B_ϕ	Roll damping coefficient
C_α	Tire cornering stiffness
$C_{c\alpha}$	Tire cornering coefficient
t_f, t_r	Front and rear track width
F_x, F_y	Forces in vehicle frame
$F_{x_{\star,\circ}}, F_{y_{\star,\circ}}, F_{z_{\star,\circ}}$	Forces acting on tires in x,y, and z-directions respectively
$F_{l_{\star,\circ}}, F_{c_{\star,\circ}}$	Longitudinal and cornering tire forces
h_s	Height from roll axis to sprung mass CG
h_{uf}, h_{ur}	Height from roll axis to unsprung front/rear mass CG
I_{zzo}	Total vehicle moment of inertia about the z-axis
I_{xzs}	Sprung mass product of inertia about the x-z plane
$I_{xxs}, I_{yys}, I_{zzs}$	Sprung mass moment of inertia about the x, y, and z-axis respectively
K_ϕ	Roll stiffness rate
l_{cgs}	Distance from vehicle CG to sprung mass CG
l_f, l_r	Longitudinal distance between the center of front/rear tire and vehicle center of mass
M	Total vehicle mass
M_s	Sprung vehicle mass

M_{uf}, M_{ur}	Unsprung front/rear vehicle masses
T_{xs}, T_z	Moments in vehicle frame
$T_{\phi f}, T_{\phi r}$	Suspension torque about the roll axis
$v_{l_{\star,\circ}}, v_{c_{\star,\circ}}$	Longitudinal and cornering velocities in the tire frame
$v_{x_{\star,\circ}}, v_{y_{\star,\circ}}$	Longitudinal and lateral velocities in the vehicle frame
α	Tire slip angle
δ	Tire steering angle

Model Predictive Controller nomenclature

H_c, H_p	Control horizon interval, prediction horizon interval
$J(\xi[k], \Delta u[k])$	Cost function
$\boldsymbol{M}$	Mass matrix
$\boldsymbol{Q}, \boldsymbol{R_u}, \boldsymbol{R_{\triangle u}}$	Output, input, and input-rate weight matrices
$u, \triangle u$	Plant input, input rate
$V_{min}^{u}, V_{max}^{u}, V_{min}^{\Delta u}, V_{max}^{\Delta u}, V_{min}^{\xi}, V_{max}^{\xi}$	Equal Concern for Relaxation (ECR) vectors
ε	Slack variable
$\xi, \dot{\xi}$	Vehicle state, state trajectory
ρ	Slack weight
τ	Control sampling rate
$\omega_{ij}^{\xi}, \omega^{u}, \omega^{\Delta u}$	Output, input, and input-rate weights

I. INTRODUCTION

According to the National Highway Traffic Safety Administration (NHTSA), rollover occurrence accounted for over 35% of all fatalities in passenger vehicles and light trucks in 2009. This percentage has steadily risen over 7.5% since 1982 (see Figure 1), even though vehicle fatalities are at an all-time low in the U. (National Highway Traffic Safety Administration, 2009) S.. Advancements in safety features have played a contributing factor to the overall decline in automotive related deaths, but the rise in rollover as a percentage of fatalities reflects a potential area of improvement for car safety. The mitigation of high speed rollover and skid-out during crash avoidance is a primary concern for this paper.

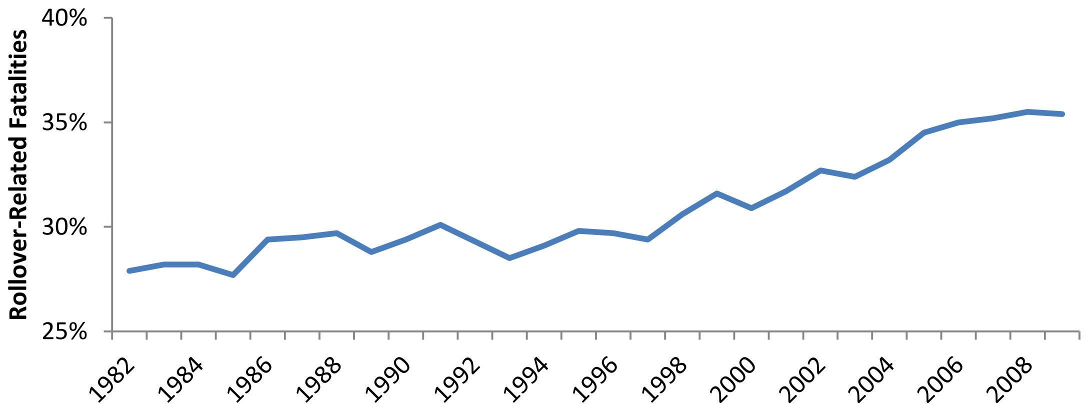


Figure 1: Rollover related fatalities over the last 30 years have been steadily increasing, highlighting a need for improvements in rollover mitigation technology

Current Technologies

Leaders in the automotive industry have devoted a tremendous amount of research and development to car safety systems over the years. Seat belts are estimated by the NHTSA to have saved over 267,000 lives *alone* in the US, while frontal air bags have prevented over 30,00 fatalities since 1975. Automatic braking systems have been another step in that direction, and the 1990s saw the development of electronic stability control. The Insurance Institute for Highway Safety (IIHS) reported in 2010 that electronic stability control (ESC) has reduced the risk of fatalities by 33% over the past 10 years (Farmer, Effects of Electronic

Stability Control on a Fatal Crash Risk, 2010). In fact, the success of ESC brought the NHTSA to require its implementation on all new vehicles sold in the U.S. after September 1, 2011 (National Highway Traffic Safety Administration, 2007).

Other recent existing features include forward collision mitigation, emergency braking systems, lane departure warning systems, and adaptive headlights. A 2008 report released by the IIHS determined that collision mitigation technology could potentially prevent 2.3 million crashes per year (Farmer, Crash Avoidance Potential of Five Vehicle Technologies, 2008). The report further describes collision avoidance as having the "greatest potential" for crash avoidance compared to other new car safety features. And recently automakers have been implementing various forms of crash mitigation. Using a combination of radar, laser, and millimeter wave cameras to detect collisions, these systems apply braking to avoid or minimize impact. Some examples are Audi's Pre-Sense system, Honda's Collision Mitigation Braking System, Mercedes-Benz' Pre-Safe with Brake Support, Infiniti’s Blind Spot Monitoring System, and Toyota's Vehicle Dynamics Integrated Management (Matsubayashi et al., 2006) (Motor Trend, 2009) (Toyota, 2006).

Collision warning/mitigation systems are being explored by various automakers, however Toyota is leads the way in terms of full collision avoidance. Collision avoidance is a relatively new area of research in automotive safety. In the event of an oncoming collision with an object or pedestrian, the controller is designed to apply corrective steering and braking to avoid a crash. In 2011 Toyota demonstrated the Pre-Crash technology on full scale vehicles during a media demo. The Pre-Crash System is on its way to production, although Toyota has yet to declare just how soon that may be (Melanson, 2011).

An important aspect of collision avoidance is the roll stability of the vehicle during high speed maneuvers. If the vehicle must make a maneuver that would cause skidding or rollover, then the safety of passengers as well as pedestrians would be compromised. The need for high speed collision avoidance is most prevalent on highways and freeways where the speeds involved are more than enough to flip a car when the wheel is rapidly turned. The decrease for rollover and skidding propensity during these collision maneuvers is the motivation of this paper.

II. PROJECT DESCRIPTION

The purpose of this thesis is to design, simulate, and test a roll stability controller (RSC) on a scaled vehicle performing autonomous crash avoidance maneuvers An emphasis is placed on the mitigation of rollover and spin-out, however if a safe maneuver is not found and a collision is inevitable, the program will run a brake command to reduce the vehicle speed before impact. If no safe maneuver is found, the vehicle performs emergency braking such that the impact of collision may be minimized. The crash avoidance is performed by a rapidly-exploring random tree (RRT) algorithm which adjusts the steer angle and throttle of the vehicle at a rate of 20 Hz. During each iteration of the program, the vehicle state and RRT outputs are fed into a model predictive controller (MPC) for processing. The MPC uses the sensor information and, based on a weighted performance measure, generates an optimal trajectory around the obstacle.

Mechatronics projects such as this are, by their nature, multidisciplinary ones in which elements of mechanical, electrical, and controls engineering are applied. As such, a proper breakdown of this project would include both a description of the hardware layout and the software flow diagram. Both are important but separate aspects which are presented here from a top level.

Hardware Layout

The experimental setup for the RSC includes a computer, a Xiphos board, an ultrasonic array, the crash avoidance vehicle, and the obstacle. The length of the test track is variable, but the width is limited to 6” to provide an accurate scale representation of a normal road.

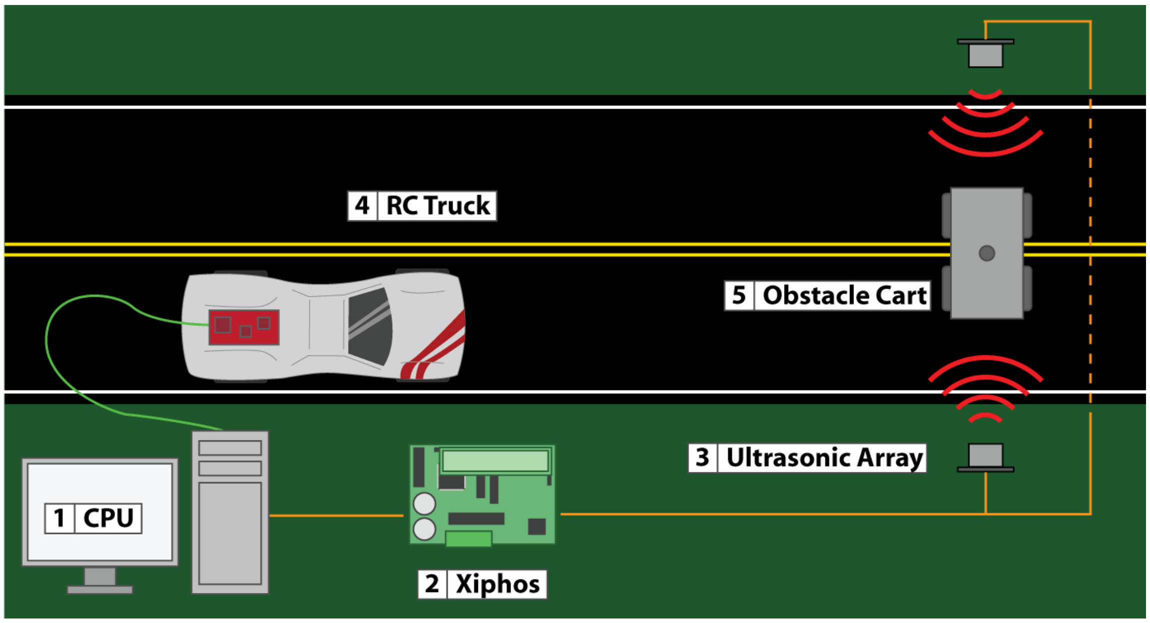


Figure 2: Experimental Setup

1. CPU – The main central processing unit runs MATLAB that contains the RRT algorithm, path planner, and MPC. The algorithms run in real-time as the obstacle position is updated and convert the selected path to vehicle steering and throttle commands. The MPC analyses the vehicle state and RRT steer command to generate a stable trajectory.
2. Xiphos – The Xiphos development board is responsible for collecting the position of the obstacle along the track and the obstacle's width. By using a dedicated microcontroller to track the obstacle, additional sensors may be added without stealing valuable processor power from the main algorithms.
3. Ultrasonic Array – The ultrasonic array tracks the obstacle along the width of the track. The ultrasonic array consists of two sensors across from one another on each along the width of the track. Each sensors waits for the other to receive their signal before triggering to avoid cross contamination. By running the Xiphos board at 16Mhz, accurate position and sizes are determined using the ultrasonic sensors.
4. R/C Truck – The R/C truck is a 1/10th scale electric vehicle. The truck uses a hobby servo for steering control and a Titan 12T550 DC brushless motor for power. It has a simple rear differential

and front wheel steering. The particular truck chosen was the Traxxas Slash 2WD truck because of its realistic suspension and high center of gravity.

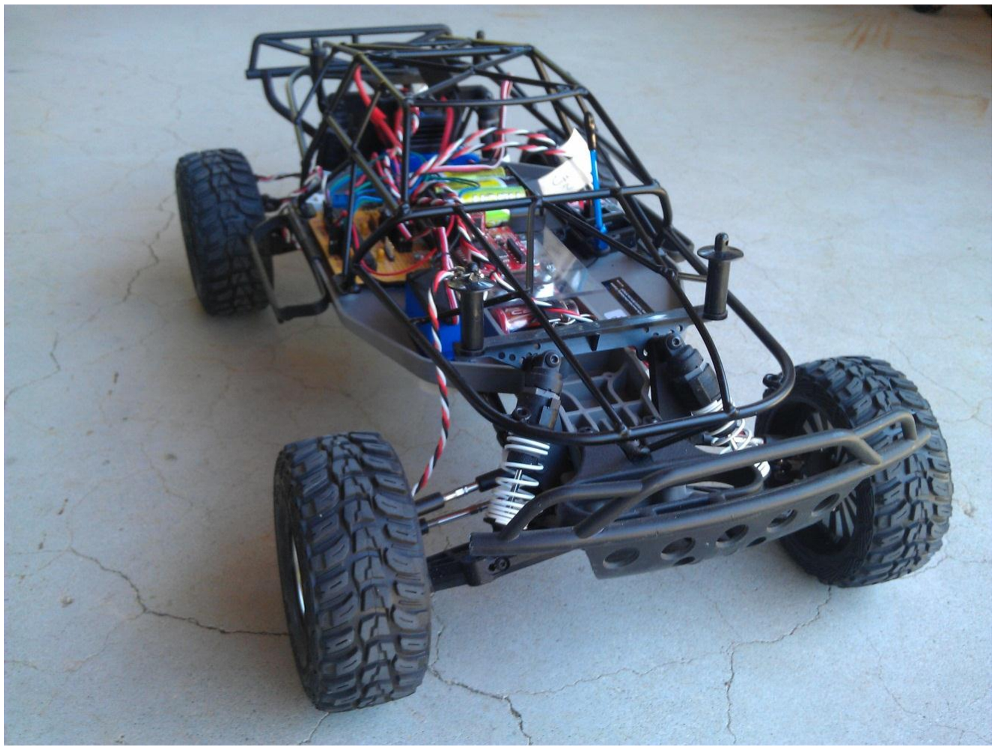


Figure 3: R/C Vehicle used for experimental validation

The driving motor and steering servo are controlled with an onboard microcontroller that reads in the commands from the main CPU. It is equipped with four wheel encoders, a six-degree-of-freedom inertial measurement unit (6DOF IMU), and two dsPIC microcontrollers for sensor processing and output control. The wheel encoders determine individual wheel speed while the IMU detects linear accelerations and roll rates in all three directions.

5. Obstacle Cart – The obstacle cart is a moving platform as a base for interchangeable objects. The obstacle cart moves manually on a linear path perpendicular to the test track.

Software Layout

As an aid to visualize the processes the crash avoidance system will progress, a flow diagram was created to organize and plan a framework for the software

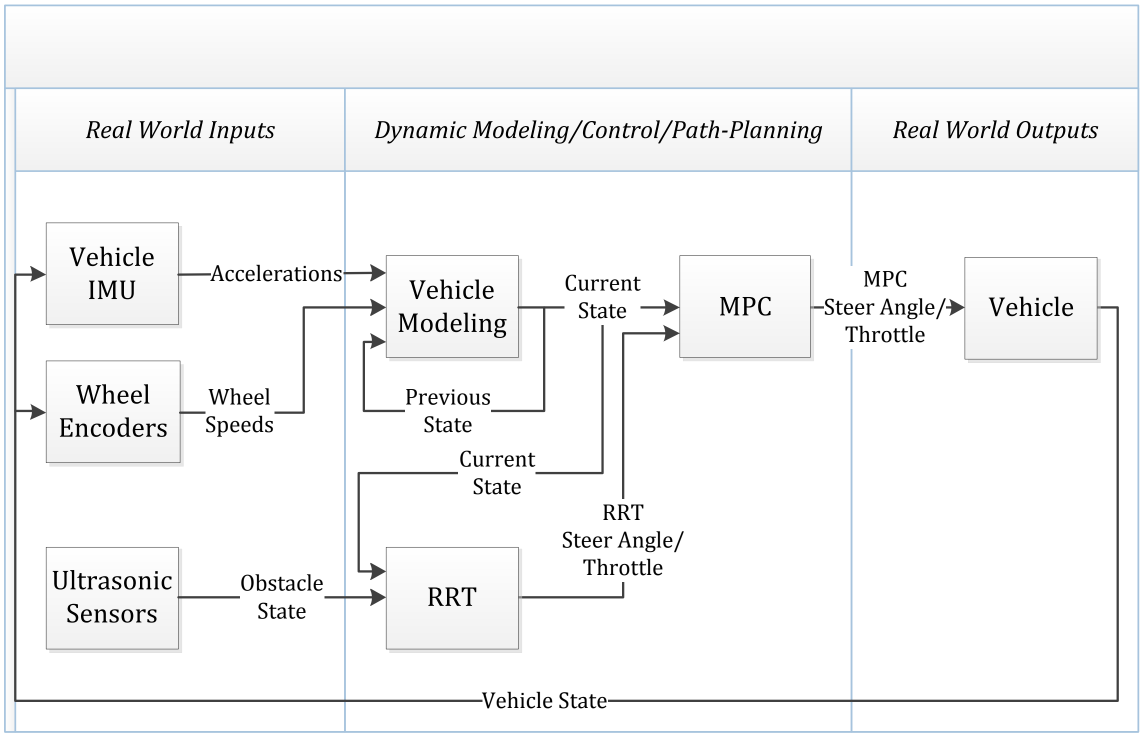


Figure 4: System flow diagram for the stability control of an autonomous crash avoidance vehicle

The software model is broken down into three main process categories. The IMU and wheel encoders send their information to be processed in the vehicle modeling subsystem. The vehicle model transmits the current state to the Rapidly-exploring Random Tree (RRT) path planner, which uses the vehicle state and information about the obstacle position from the ultrasonic sensors to generate feasible paths. The paths are then filtered according to the boundary conditions to remove potentially dangerous paths. The RRT feeds the finalized trajectory to the Model Predictive Controller (MPC), which adjusts the trajectory according to stability considerations. The finalized steer trajectory is fed to the vehicle. The details of the communication protocol between the test vehicle and MATLAB can be found in APPENDIX A: ONBOARD COMMUNICATIONS

III. RAPIDLY-EXPLORING RANDOM TREE PATH PLANNER

The Defense Advanced Research Projects Agency (DARPA) has produced prominent technologies through their Urban and Grand Challenges. In these challenges, full-scale vehicles are required to navigate autonomously through a course of waypoints and obstacles. Of the competitors, the Massachusetts Institute of Technology (MIT) team was noted for their use of the Rapidly-exploring Random Tree (RRT) path planning method (Kuwata, et al., Real-Time Motion Planning With Applications to Autonomous Urban Driving, Sept. 2009). The RRT method begins by generating random branches from a start node. Each branch is compared with boundary conditions defined by obstacles in the driving plane. If a branch does not meet the boundary conditions, the branch is terminated. Branches that meet the boundary conditions become intermediate nodes. From these intermediate nodes, new random branches are produced and compared with the boundary conditions. The branching process continues until a viable path is found that reaches the target node. In addition, MIT continued to monitor for collisions as their vehicle executed the feasible path. If interference occurred and all feasible paths were terminated, an emergency braking system was applied (Kuwata, et al., Motion Planning for Urban Driving using RRT, Sept. 2008).

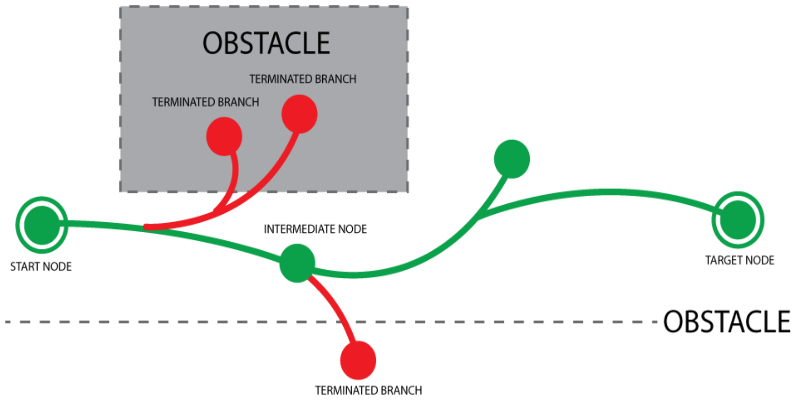


Figure 5: RRT nodes and branches mapping acceptable paths (green) and marking failed paths (red) in the driving plane.

RRT Navigation Procedure

The RRT algorithm is summarized in pseudo-code as follows:

```
1)  Add vehicle and goal nodes to the configuration space
2)  Add node $n_{nu}$ at distance x from closest node in the vehicle tree in the direction of the closest node
    in the goal tree
3)  If $n_{nu}$ is feasible, add $n_{nu}$ to the vehicle tree
4)  Else select a random point in between the vehicle and the goal, $n_{rnd}$
5)  Add node $n_{nu}$ at distance x from closest node in the vehicle tree in the direction of $n_{rnd}$
6)  Repeat from 3 until time $t$ and if no path is found by time $t$ execute emergency stop
7)  Add node $n_{nu}$ at distance x from closest node in the goal tree in the direction of the closest node in
    the vehicle tree
8)  If $n_{nu}$ is feasible, add $n_{nu}$ to the goal tree
9)  Otherwise select a random point in between the vehicle and the goal
10) Add node $n_{nu}$ at distance $x$ from closest node in the goal tree in the direction of
11) Repeat from 8 until time $t$ and if no path is found by time $t$ execute emergency stop
12) Repeat from 2 until both vehicle and goal trees meet
```

Advantages to the RRT method include generalization in path planning and minimal configuration. The RRT algorithm creates random branches non-reliant on system stimulus. Therefore, the system generates potential paths regardless of its surroundings. The filtering of these paths is done with sensor-acquired boundary conditions. In addition, the RRT method does not require pre-configuration with respect to its host (vehicle dimensions, vehicle speed, etc.). Vehicle dynamics and environment data are tracked independently of path prediction.

Disadvantages of the RRT method include potential lag time in determining complex paths and need for a high volume of memory. Complex paths through multiple obstacles will increase path-planning time using the RRT method. More random branches must be generated and assessed to determine a feasible path. In addition, the RRT method lacks an environmental memory system. Planning time could be reduced if vehicle dynamics and environmental conditions matched a previous occurrence and path plan. However, without a memory system, the system must re-calculate a feasible path, thus reducing reaction time.

To determine the feasibility of the added nodes it must be considered whether it intersects the obstacle and if the drivable path to the node is dynamically safe. By referencing its location against a virtual map of the area it is determined if it intersects with the obstacle by. Once the bare structure of a path is established by the RRT method, the path needs to be smoothed out by another algorithm; the Pure Pursuit Method (PPM) (Kuwata, et al., Motion Planning in Complex Environments using Closed-Loop Prediction, Aug. 2008).

Pure Pursuit Path Follower

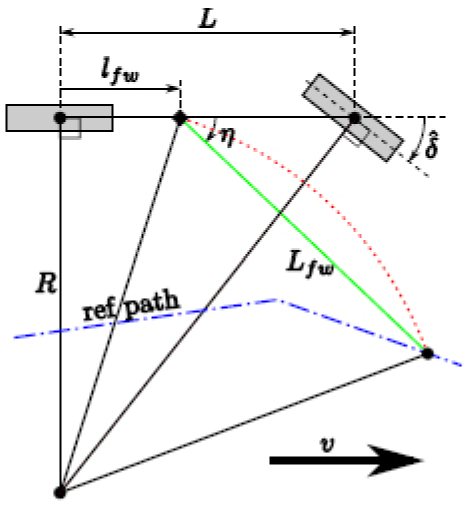


Figure 6: Pure Pursuit path follower

The Pure Pursuit Method determines what wheel angles are necessary to follow the path. If no feasible paths are constructed from a node after a set number of attempts then that node is removed from the tree. After the node has been removed, the area around that node is flagged as a non-realizably trajectory so that the algorithm will not attempt to add a node there again. There is also time limit set on how long the algorithm will attempt to create a path. If no possible crash avoidance is found within the time limit, then the vehicle applies the brakes and steers away from the obstacle. If a realizable trajectory is found, then the trajectory is fed into the Model Predictive Controller. The entire process is repeated at a rate of 20 Hz.

In order to set the goal node, we can pick a point past the obstacle. Since this point cannot be seen by the vehicle it is re-evaluated once the vehicle can see past the obstacle it is trying to avoid, ensuring that the goal node is safe.

$$\delta = -tan^{-1}\left(\frac{Lsin(\eta)}{\frac{L_{fw}}{2} + l_{fw}cos\eta}\right) \tag{3.1}$$

Using Equation 3.1 the turning angle is found based on the look-ahead distance L_{fw}, the time delay of the turning actuator l_{fw} and the wheel base L. The look-ahead distance is based on the speed of the vehicle. The angle v is found from the intersection of the look-ahead distance and the planned path as shown in Figure 6.

$$L_{fw} > \upsilon\tau - l_{fw} \tag{3.2}$$

The look-ahead distance must constrain to Equation 3.2 where v is the velocity of the vehicle, and τ is the time constant of the steering actuator.

The details of the RRT implementation may be found in APPENDIX C: RAPIDLY-EXPLORING RANDOM TREE – MATLAB/SIMULINK STRUCTURE.

.

IV. DEVELOPMENT OF DYNAMIC VEHICLE MODEL

Existing Vehicle Models

Almost every introductory vehicle dynamics textbook presents what is commonly referred to as the bicycle model as is shown in Figure 7 (Jazar, 2008). Developing the bicycle model requires several simplifying assumptions. First, the steering angles for the front wheels are assumed to be equal. It is also required that both the lateral and longitudinal forces induced by the tires are modeled linearly. The linearization is only valid for small steering angles at low acceleration rates. To simplify the model even further, the tire forces occurring at each axle are taken as an average neglecting any effects of dynamically induced weight transfer. By neglecting the body roll the roll-yaw coupling inherent in the general moment equations is lost. As a result the model is only valid for the mildest vehicle maneuvers. In terms of accident avoidance this is a serious problem because most emergency maneuvers require the use of extreme turning and braking and the nonlinearities will become extremely important in developing an accurate solution. It is obvious that a more complex model is required.

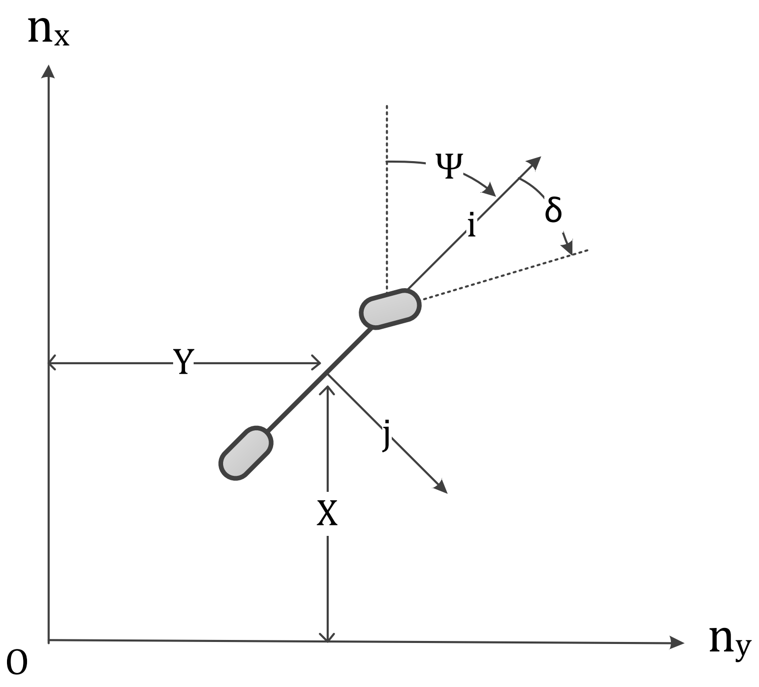


Figure 7: Bicycle Model

A far more accurate model would incorporate all of the available degrees of freedom in the vehicle, as well as any and all non-linear tire effects. Because there are thousands of moving parts on a typical vehicle, trying to solve for them all will quickly become a physically impossible. By lumping the masses as much as possible (sprung mass/unsprung masses) the model is still accurate but with a much smaller number of degrees of freedom. These include the three degrees of translational and rotational freedom given to the body, as well as the minimum number needed for an accurate suspension model. In Figure 2 a screen shot of CarSim, a professionally developed vehicle simulating software package, is displayed. The CarSim model would be an excellent example meeting the mentioned requirements. This 14 degree of freedom model includes all three translational and rotational degrees of freedom given to the vehicle body, as well as one rotational and one translational degree of freedom for each tire (Mechanical Simulation Corperation, 2000).

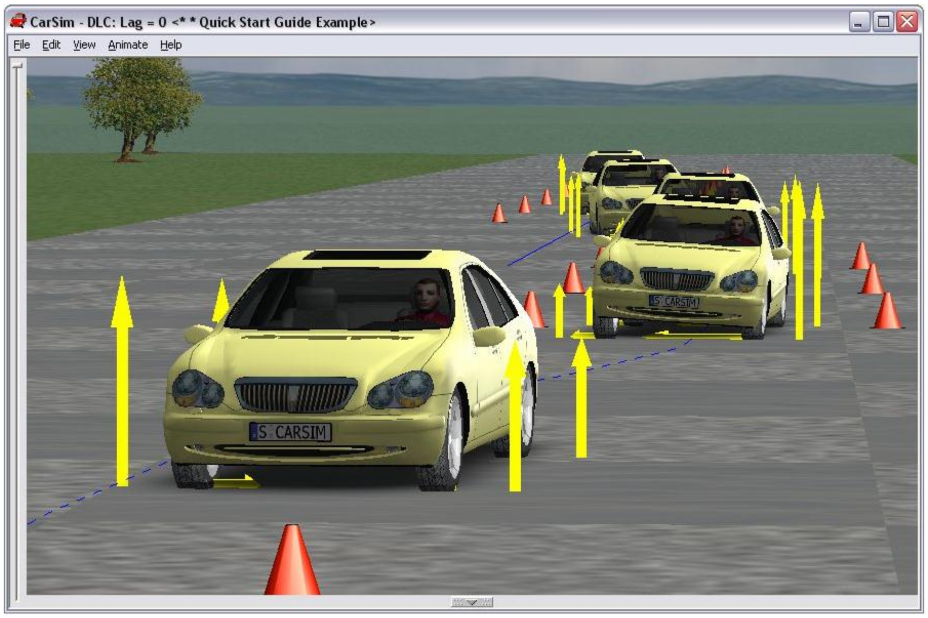

CarSim - DLC: Lag = 0 <* * Quick Start Guide Example>
File Edit View Animate Help
S CARSIM

Figure 8: CarSim running a simulation.

The challenge with these models is that they require an extensive amount of suspension design knowledge before the motions of the tires can be accurately solved. Because the tires are essentially modeled as non-linear dampers, any inaccuracies in the suspension model will result in incorrect tire velocities, and the tire forces will not be accurate. Use of these models requires the knowledge of suspension link mounting positions and the various degrees of translation/rotational freedom for each link. A model that is more accurate than the bicycle model, but simpler than CarSim would be desirable.

One way to improve the bicycle model is to no longer average the tire normal forces and resulting lateral and longitudinal forces. Pitch, roll, and body bounce are still neglected, but now there are four contact points supporting the vehicle mass and the dynamic weight transfer forces are taken into consideration (Casanova, 2000), commonly referred to as the track model. The track model with non-linear tires serves as a very solid improvement, but the pitch-roll-yaw coupling is still neglected. Realizing that almost every vehicle suspension carries left-right symmetry there is the possibility to include roll. The roll axis of any vehicle is the line about which the body rolls such that the tires do not have any induced lateral velocities. For some passenger sedans, such as the Ford Taurus, the roll axis is assumed horizontal (Demerly, 2000). It is then possible to keep track of the roll degree of freedom. It should be noted that most vehicles lack front-rear symmetries in their suspensions due to anti-pitch and anti-lift mechanisms, and thus determining the pitch becomes very difficult. To simplify the model pitch is neglected, and our equations of motion will only include the roll-yaw coupling. The vertical translational degree of freedom is also neglected, which means that our model is only valid for flat, smooth roads. The wheel translational degree of freedom is also neglected, but the rotational degrees of freedom are kept. This is done by treating the tire spring and the suspension spring as a pair of springs in series. Additionally, the cornering force F_c is what primarily affects the lateral force on the vehicle, and therefore is the main contributor to rolling and skid-out in a turn. The longitudinal force F_l meanwhile is a factor which plays a role during a braking maneuver. Since braking only occurs in the event of a failed crash avoidance maneuver, braking is neglected for the pure cornering case which this study considers. Therefore we have an eight degree of freedom model that covers longitudinal translation, lateral translation, yaw, roll, and the rotation of the four wheels.The list of assumptions is provided below as a summary.

- Longitudinal and lateral velocities
- Roll about x-axis

- Yaw about z-axis
- Uses smooth, flat road
- Lateral tire forces
- Inputs: current states, steering, sensor data
- Outputs: Linear/Angular positions, velocities, and rotations, slip angles, various forces

Aside from determining what degrees of freedom to model, the numerical solution scheme must also be considered. By choosing to use the MATLAB programming language, there are three possibilities. The first possibility is to write an entire solver from scratch using *.m files. Writing a solver would be a monumental task, but can be vastly sped up by making use of the numerical solvers already provided by MATLAB. The first sets of solvers available are the Runge-Kutta solvers. While these solvers have already been designed to maximize calculation efficiency, they lack the ability to return the highest order derivative obtained in the solution. For general vehicle motion through space this is typically not a problem, but for modeling collision avoidance it may be necessary to know the given accelerations at any time step. The accelerations could be an input for a particular accident avoidance solver. It is also necessary for computing the lateral acceleration gain. When validating the model we will also need the acceleration values as the data sets available all make acceleration comparisons. One option available is the Simulink environment in MATLAB. Simulink is commonly used when modeling controllers as it is an environment based on block diagrams. It also allows the accelerations calculated at every time step to be plotted in real time. This was the approach used in the NAVDyn Model (Demerly & Youcef-Toumi, 2000); all portions of the model existed as connected block diagrams. However, Simulink also supports the use of *.m files, and allows the combination of block diagrams and modular programming. Simulink is the method adopted here because it allows for maximum user flexibility without having to write a complex solver from scratch.

Coordinate System Definition

In order to accurately model the vehicle dynamics, three sets of coordinate axes are necessary. The method recommended in SAE J670e is adopted as has been done by various other authors (Casanova, 2000), (Demerly, 2000). First, the Earth-fixed coordinate axes *XYZ* are defined and the uppercase letters are used to denote this system. These axes are considered an inertial frame of reference; the coordinates are orthogonal and right handed. A point of origin must be defined as well as the direction of the *X* axis.

The vehicle-fixed, chassis coordinate system *xyz* and body coordinate system *x'y'z'* are located at the same point at specified on the vehicle at rest. Assuming the vehicle has lateral symmetry, which is reasonable for most passenger sedans, it is possible to determine the location of the roll axis. This is the axis that allows the body to rotate without any induced velocities in the tires. By projecting a line vertically downward through the vehicle center of gravity, the origin of each system is located at the intersection of this line and the roll axis. The advantage of selecting this location is that it provides the simplest means of developing the tire forces. The *x* and *x'* axes are in the longitudinal (forward) direction, the *y* and *y'* axes are in the right hand lateral direction, and the *z* and *z'* axes point vertically downward. The unit vectors for each frame are given as:

1) *XYZ* $\boldsymbol{n}_x, \boldsymbol{n}_y, \boldsymbol{n}_z$
2) *xyz* ***i, j, k***
3) *x'y'z'* ***i', j', k'***

where the unit vectors are listed in x-y-z order. The standard SAE definition of right-hand rotations, starting with *xyz* aligned with *XYZ*, are given by:

1) Yaw rotation ψ about the *z*-axis
2) Pitch rotation θ about the *y*-axis
3) Roll rotation φ about the *x*-axis

These are taken about the vehicle-fixed axes *xyz*. The *x'y'z'* axis system will roll about the *x*-axis and remain fixed to the body. The transform between *x'y'z'* and *oxyz* is used when developing the equations involving the body (see Figure 9).

To allow for compact notation $\star$ denotes the front and rear of the vehicle, while $\circ$ denotes the left and right side of the vehicle. This notation eliminates the need for redundant equations for the front left, front right, rear left, and rear right segments of the vehicle.

Linear Equations of Motion

To begin, the origin of the inertial reference frame is defined. The distance from this origin to the origin of the chassis reference frame is:

$$\boldsymbol{R}_o = X\boldsymbol{n}_x + Y\boldsymbol{n}_y \tag{4.1}$$

where *X* is the x-axis coordinate, *Y* is the y-axis coordinate, and $\boldsymbol{R}_o$ is the radius with respect to the inertial frame as shown in Figure 9.

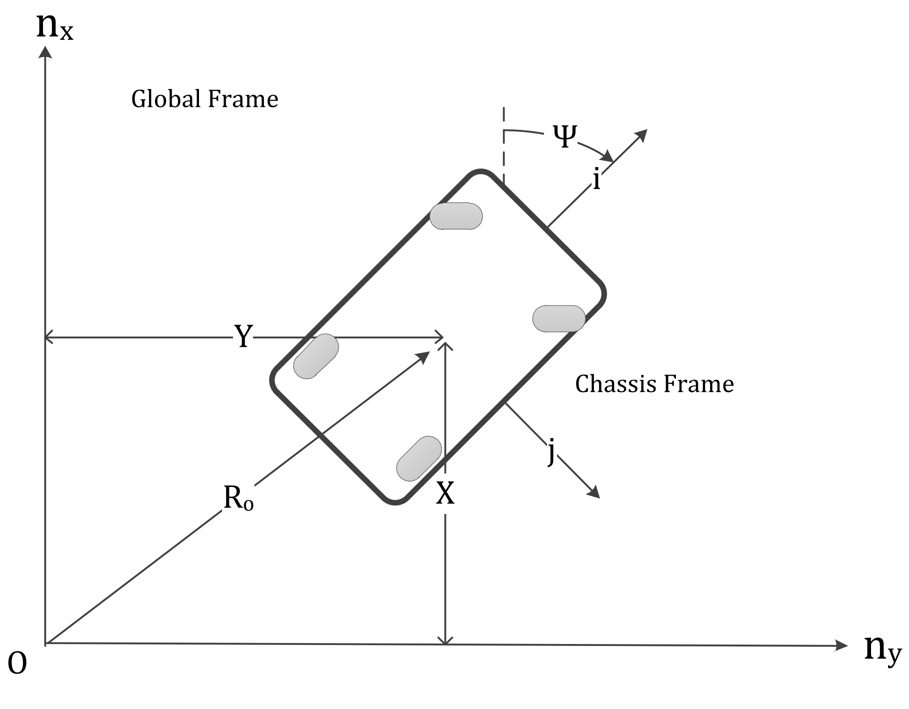


Figure 9: Global and chassis coordinate frames

Equation is transformed into the chassis frame by setting:

$$\boldsymbol{n}_x = \cos\psi\,\boldsymbol{i} - \sin\psi\,\boldsymbol{j} \tag{4.2}$$

$$\boldsymbol{n}_y = \sin\psi\,\boldsymbol{i} + \cos\psi\,\boldsymbol{j} \tag{4.3}$$

and by inserting Equation 4.2 and 4.3 into Equation 4.1 it is shown that the radius is equal to:

$$\boldsymbol{R}_o = (X\cos\psi + Y\sin\psi)\boldsymbol{i} + (-X\sin\psi + Y\cos\psi)\boldsymbol{j} \tag{4.4}$$

and this is given with respect to the chassis unit vectors. The velocity of the chassis origin is given by taking the time derivative of Equation 4.4 giving:

$$\boldsymbol{V}_o = \frac{d\boldsymbol{R}_o}{dt} = \dot{X}\boldsymbol{n}_x + \dot{Y}\boldsymbol{n}_y$$

$$\boldsymbol{V}_o = \left(\dot{X}\cos\psi + \dot{Y}\sin\psi\right)\boldsymbol{i} + \left(-\dot{X}\sin\psi + \dot{Y}\cos\psi\right)\boldsymbol{j} \tag{4.5}$$

and by defining

$$V_{ox} \overset{\text{def}}{=} \dot{X}\cos\psi + \dot{Y}\sin\psi \tag{4.6}$$

$$V_{oy} \overset{\text{def}}{=} -\dot{X}\sin\psi + \dot{Y}\cos\psi \tag{4.7}$$

the velocity Equation 4.5 is simplified into

$$\boldsymbol{V}_o = V_{ox}\boldsymbol{i} + V_{oy}\boldsymbol{j} \tag{4.8}$$

which is given in the chassis reference frame. Because the body and chassis systems share the same origin, Equation 4.8 defines the linear velocity of the body reference frame as well. To calculate the acceleration of the chassis/body origin the derivative with respect to time is applied again to Equation 4.8:

$$\boldsymbol{a}_o = \frac{d^2\boldsymbol{R}_o}{dt^2} = \frac{\partial\boldsymbol{V}_o}{\partial t} + \Omega_c \times \boldsymbol{V}_o \tag{4.9}$$

To define the angular velocities the fact that the body is free to roll with respect to the chassis must be taken into consideration. Therefore the angular velocity of each is given by:

$$\Omega_c = \dot{\psi}\boldsymbol{k} \tag{4.10}$$

$$\Omega_b = \dot{\phi}\boldsymbol{i} + \dot{\psi}\boldsymbol{k} \tag{4.11}$$

and by inserting Equations 4.10 and 4.11, the acceleration of the chassis origin is shown to be:

$$\boldsymbol{a}_o = (\dot{V}_{ox} - \dot{\psi}V_{oy})\boldsymbol{i} + (\dot{V}_{oy} + \dot{\psi}V_{ox})\boldsymbol{j} \tag{4.12}$$

where the symbol *Ω* represents the rotation vector and the subscripts *c* and *b* represent the chassis and body respectively.

With the equation of motion of the chassis coordinate system defined, the equations of motion of the sprung and unsprung masses are considered. The locations of the sprung and unsprung masses are displayed in Figure 10.

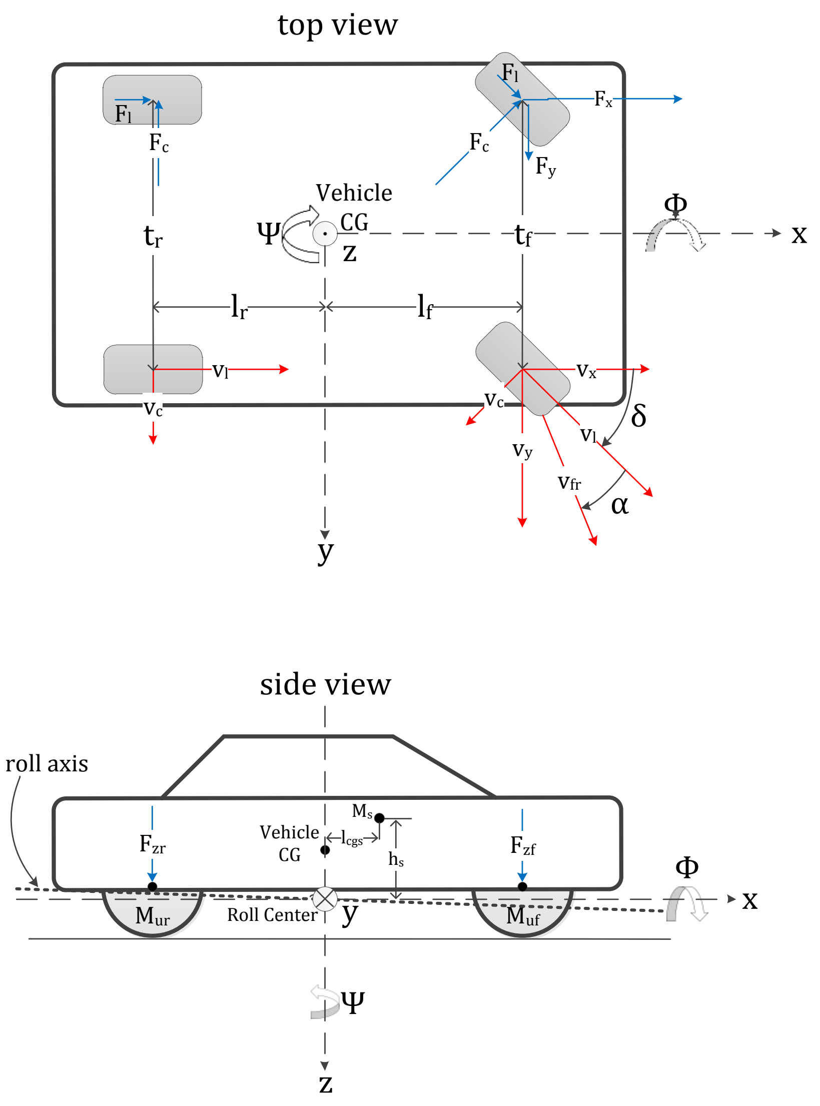


Figure 10: Vehicle frame with pertinent vehicle parameters, forces, and velocities.

The sprung mass is defined as all parts of the vehicle which are supported by the suspension. The unsprung mass, accordingly, includes the suspension linkages, shocks, wheels, bearings, and brakes. The position of the front unsprung mass is determined from:

$$\boldsymbol{R}_{uf} = \boldsymbol{R}_o + \boldsymbol{r}_{uf} \tag{4.13}$$

where $\boldsymbol{r}_{uf}$ is the radius in the chassis frame. From the position of the front unsprung mass in the chassis coordinate system is shown to be:

$$\boldsymbol{r}_{uf} = l_f\boldsymbol{i} - h_{uf}\boldsymbol{k} \tag{4.14}$$

where l_f is the longitudinal distance between the center of the front tire and the vehicle center of mass, and h_{uf} is the height, typically equal to the rolling radius. Now the time derivative of Equation 4.14 is taken to get the front unsprung mass velocity:

$$\boldsymbol{V}_{uf} = \frac{d\boldsymbol{R}_{uf}}{dt} = \boldsymbol{V}_o + \frac{\partial \boldsymbol{r}_{uf}}{\partial t} + \Omega_c \times \boldsymbol{r}_{uf} \tag{4.15}$$

and by inserting Equations 4.13 and 4.14 it is shown that:

$$\boldsymbol{V}_{uf} = V_{ox}\boldsymbol{i} + \left(V_{oy} + l_f\dot{\psi}\right)\boldsymbol{j} \tag{4.16}$$

in the chassis frame of reference. The acceleration of the front unsprung mass is derived by taking the time derivative of the equation above to show:

$$\boldsymbol{a}_{uf} = \frac{d\boldsymbol{V}_{uf}}{dt} = \boldsymbol{a}_o + \dot{\Omega}_c \times \boldsymbol{r}_{uf} + \Omega_c \times \left(\Omega_c \times \boldsymbol{r}_{uf}\right) \tag{4.17}$$

By inserting Equations 4.12 and 4.14 it is shown that:

$$\boldsymbol{a}_{uf} = \left(\dot{V}_{ox} - \dot{\psi}V_{oy} - l_f\dot{\psi}^2\right)\boldsymbol{i} + \left(\dot{V}_{oy} + \dot{\psi}V_{ox} + l_f\ddot{\psi}\right)\boldsymbol{j} \tag{4.18}$$

again in the chassis frame of reference. The exact same process is carried out for the rear unsprung mass. The only change is in the position vector of the rear unsprung mass in the chassis coordinate system. The sequence is presented below:

$$\boldsymbol{R}_{ur} = \boldsymbol{R}_o + \boldsymbol{r}_{ur} \tag{4.19}$$

$$\boldsymbol{r}_{ur} = -l_r\boldsymbol{i} - h_{ur}\boldsymbol{k} \tag{4.20}$$

$$\boldsymbol{V}_{ur} = \frac{d\boldsymbol{R}_{ur}}{dt} = \boldsymbol{V}_o + \frac{\partial \boldsymbol{r}_{ur}}{\partial t} + \Omega_c \times \boldsymbol{r}_{ur} \tag{4.21}$$

$$\boldsymbol{V}_{ur} = V_{ox}\boldsymbol{i} + \left(V_{oy} - l_r\dot{\psi}\right)\boldsymbol{j} \tag{4.22}$$

$$\boldsymbol{a}_{ur} = \frac{d\boldsymbol{V}_{ur}}{dt} = \boldsymbol{a}_o + \dot{\Omega}_c \times \boldsymbol{r}_{ur} + \Omega_c \times (\Omega_c \times \boldsymbol{r}_{ur}) \tag{4.23}$$

$$\boldsymbol{a}_{ur} = \left(\dot{V}_{ox} - \dot{\psi}V_{oy} + l_r\dot{\psi}^2\right)\boldsymbol{i} + \left(\dot{V}_{oy} + \dot{\psi}V_{ox} - l_r\ddot{\psi}\right)\boldsymbol{j} \tag{4.24}$$

It is now necessary to define the linear motion of the sprung mass. The process is similar to that carried out for the unsprung masses with the exception that the sprung mass is free to roll about the roll axis. Therefore it is necessary to project the sprung mass position vector from the body coordinate frame into the chassis coordinate frame. The vector locating the body is shown to be:

$$\boldsymbol{R}_s = \boldsymbol{R}_o + \boldsymbol{r}_s \tag{4.25}$$

where $\boldsymbol{r}_s$ is the vector from chassis origin to sprung mass, and $\boldsymbol{R}_s$ is the net position vector. The conversion from body reference frame to chassis reference frame is given by:

$$\boldsymbol{i}' = \boldsymbol{i}$$

$$\boldsymbol{j}' = \cos\phi\,\boldsymbol{j} + \boldsymbol{s}\text{in}\,\phi\,\boldsymbol{k}$$

$$\boldsymbol{k}' = -\,\boldsymbol{s}\text{in}\,\phi\,\boldsymbol{j} + \cos\phi\,\boldsymbol{k}$$

which are used to redefine the position vector of the sprung mass in the chassis coordinate frame. These equations are shown to give:

$$\begin{aligned} \boldsymbol{r}_s &= l_{cgs}\boldsymbol{i}' - h_s\boldsymbol{k}' \\ \boldsymbol{r}_s &= l_{cgs}\boldsymbol{i} + \mathrm{h_s}\boldsymbol{s}\text{in}\,\phi\,\boldsymbol{j} - \mathrm{h_s}\cos\phi\,\boldsymbol{k} \end{aligned} \tag{4.26}$$

where l_{cgs} is the longitudinal location of the body center of gravity and $\mathrm{h_s}$ is the height with the vehicle at rest. The velocity is determined the same way as before and is shown to be:

$$\boldsymbol{V}_s = \frac{d\boldsymbol{R}_s}{dt} = \boldsymbol{V}_o + \frac{\partial \boldsymbol{r}_s}{\partial t} + \Omega_c \times \boldsymbol{r}_s \tag{4.27}$$

and by again inserting Equations 4.8 and 4.12 it is shown that:

$$\boldsymbol{V}_s = \left(V_{ox} - h_s\dot{\psi}\,\mathbf{sin}\,\phi\right)\boldsymbol{i} + \left(V_{oy} + h_s\dot{\phi}\cos\phi + l_{cgs}\dot{\psi}\right)\boldsymbol{j} + h_s\dot{\phi}\sin\phi\,\boldsymbol{k} \tag{4.28}$$

Due to the relative motion of the body with respect to the chassis coordinate frame, the acceleration of the sprung mass is slightly different than the acceleration of the unsprung masses, and is given by:

$$\boldsymbol{a}_s = \frac{d\boldsymbol{V}_s}{dt} = \boldsymbol{a}_o + \dot{\Omega}_c \times \boldsymbol{r}_s + \Omega_c \times (\Omega_c \times \boldsymbol{r}_s) + 2\Omega_c \times \frac{\partial \boldsymbol{r}_s}{\partial t} + \frac{\partial^2 \boldsymbol{r}_s}{\partial t^2} \tag{4.29}$$

and by substituting in 4.12 and 4.14 it is shown that:

$$\begin{aligned}\boldsymbol{a}_s = &\left(\dot{V}_{ox} - \dot{\psi}V_{oy} - 2h_s\dot{\psi}\dot{\phi}\cos\phi - h_s\ddot{\psi}\sin\phi - l_{cgs}\dot{\psi}^2\right)\boldsymbol{i}\\ &+\left(\dot{V}_{ox} + \dot{\psi}V_{oy} + h_s\ddot{\phi}\cos\phi - h_s\dot{\phi}^2\sin\phi + l_{cgs}\ddot{\psi} - h_s\dot{\psi}^2\sin\phi\right)\boldsymbol{j}\\ &+\left(h_s\ddot{\phi}\sin\phi + h_s\dot{\phi}^2\cos\phi\right)\boldsymbol{k}\end{aligned} \tag{4.30}$$

Now by recalling from Newton that:

$$\sum F = \frac{d\boldsymbol{P}}{dt} = M\boldsymbol{a}_o \tag{4.31}$$

We can sum the accelerations of the three masses and rearrange the equations to obtain:

$$M\dot{V}_{ox} = \sum F_x + M_s\left(-2h_s\dot{\varphi}\dot{\psi}\cos\varphi - h_s\ddot{\psi}\sin\varphi\right) + M\dot{\psi}V_{oy} \tag{4.32}$$

$$M\dot{V}_{oy} = \sum F_y - M_s\left(-h_s\ddot{\varphi}\cos\varphi + h_s\dot{\varphi}^2\sin\varphi + h_s\dot{\psi}^2\sin\varphi\right) - M\dot{\psi}V_{ox} \tag{4.33}$$

These are the equations of motion in the longitudinal and lateral coordinates. Noting that the first term in each equation represent the accelerations derived in the *chassis coordinate frame*, while the second term represents the normal acceleration of the chassis frame as it rotates, it is possible to adjust the frame of reference used. Moving the second term from the right side to the left is the equivalent of adopting the rotating frame as the frame of reference. Since this is the chassis frame, and the driver is most familiar with the chassis frame, it is adopted for the model. The above are therefore modified to obtain:

$$M\dot{V}_{ox} = \sum F_x + M_s\left(-2h_s\dot{\varphi}\dot{\psi}\cos\varphi - h_s\ddot{\psi}\sin\varphi\right) \tag{4.34}$$

$$M\dot{V}_{oy} = \sum F_y - M_s\left(-h_s\ddot{\varphi}\cos\varphi + h_s\dot{\varphi}^2\sin\varphi + h_s\dot{\psi}^2\sin\varphi\right) \tag{4.35}$$

The sum of the forces in the *x* and y directions are given by:

$$\sum F_x = F_{xlf} + F_{xrf} + F_{xlr} + F_{xrr} \tag{4.36}$$

$$\sum F_y = F_{ylf} + F_{yrf} + F_{ylr} + F_{yrr} \tag{4.37}$$

and represent the resulting longitudinal and lateral forces provided by the tires. Additional forces, such as aerodynamic drag, can be added as desired.

Angular Equations of Motion

To define the angular motion of the vehicle the sprung mass angular momentum is defined first. Because the unsprung masses are only permitted to rotate about the *z*-axis, their equations of motion are much simpler and are added later. For the sprung mass the standard definition of the angular momentum is given by:

$$\boldsymbol{H_c} = I_c\Omega_{\boldsymbol{c}} \tag{4.38}$$

where *H* is the angular momentum and *I* is the inertia tensor. It should be recalled that the rotation vector is given in the chassis coordinate frame, but the inertia tensor is defined for the sprung body in the body coordinate frame. Therefore some details must be given for a proper definition. For the sprung mass, the inertia tensor is given as:

$$I_{\boldsymbol{sb}} = \begin{bmatrix} I_{xxs} & 0 & I_{xzs} \\ 0 & I_{yys} & 0 \\ I_{zxs} & 0 & I_{zzs} \end{bmatrix} + \begin{bmatrix} M_s h_s^2 & 0 & M_s h_s l_{cgs} \\ 0 & M_s\left(h_s^2 + l_{cgs}^2\right) & 0 \\ M_s h_s l_{cgs} & 0 & M_s l_{cgs}^2 \end{bmatrix} \tag{4.39}$$

where the subscript *s* indicates that it is the sprung body and the subscript *b* indicates that this is calculated in the body frame of reference. The first term represents the moment of inertia tensor as calculated about the body center of mass. The zero elements on the off diagonal terms arise from the assumed vehicular symmetry when viewed in the *x-y* and *y-z* planes. Because the vehicle lacks symmetry when viewed on the *x-z* plane, this off diagonal term must be included as shown in Figure 5. The second term represents the corrections from the parallel axis theorem and the fact that the actual body origin is not the body center of mass, but slightly behind and below as has been shown.

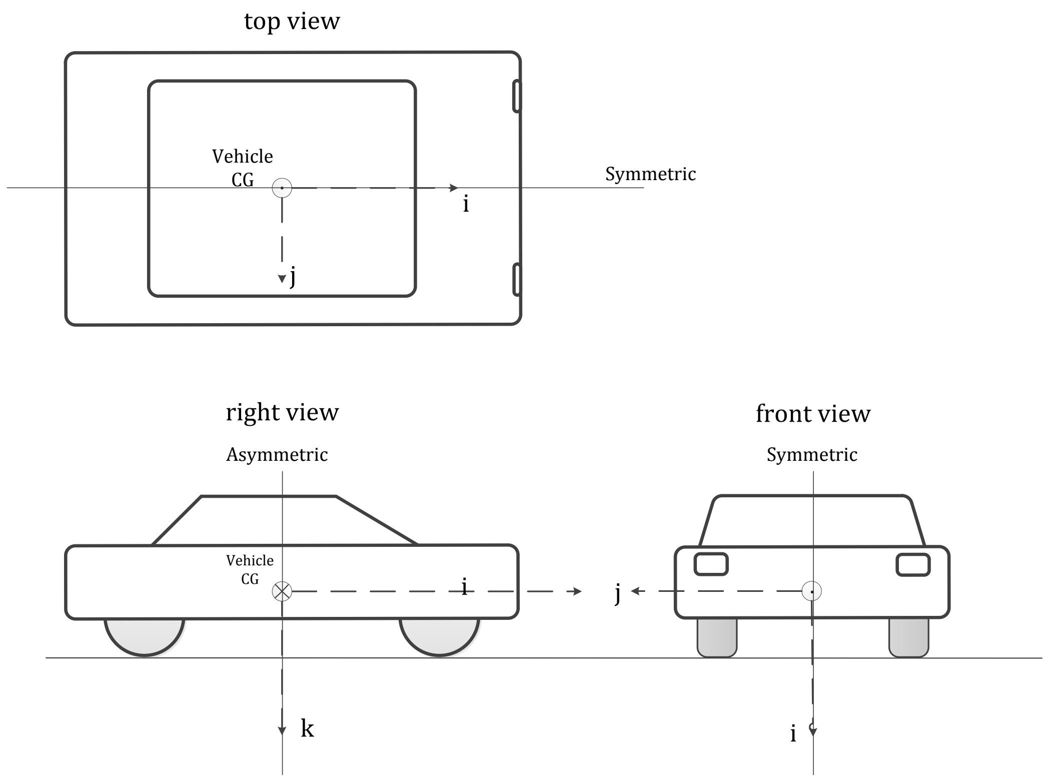


Figure 11: Symmetric and asymmetric planes of the vehicle

Now that the moment of inertia tensor has been defined in the body frame of reference, it is necessary that it be projected into the chassis frame of reference. The rotation matrix for the transformation is given by:

$$\boldsymbol{R} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix} \tag{4.40}$$

The transformation itself is shown to be:

$$I_{\boldsymbol{sc}} = RI_{sb}R^{T} \tag{4.41}$$

Finally, recalling that the unsprung masses only rotate about the *z*-axis, their moment of inertia terms must be added to the moment of inertia tensor. These additional terms are shown to be:

$$I_{\boldsymbol{usc}} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & I_{zzuf} + I_{zzur} + M_{uf} l_f^2 + M_{ur} l_r^2 \end{bmatrix} \tag{4.42}$$

where the *I* terms are the moments of inertia about the unsprung mass centers and the terms that follow represent the parallel axis corrections. Therefore the final moment of inertia tensor is shown to be:

$$I_{\boldsymbol{c}} = I_{sc} + I_{usc} \tag{4.43}$$

Taking the time derivative of Equation 4.38 and recalling that the pitch degree of freedom is neglected:

$$\frac{\partial \boldsymbol{H}_c}{\partial t} = \frac{d\boldsymbol{H}_c}{dt} + \Omega_{\boldsymbol{c}} \times \boldsymbol{H}_{\boldsymbol{c}} \tag{4.44}$$

$$\frac{\partial \boldsymbol{H}_c}{\partial t} = \left(\dot{H}_x - H_y \psi\right)\boldsymbol{i} + \left(\dot{H}_z + H_y \phi\right)\boldsymbol{k} \tag{4.45}$$

$$\frac{\partial \boldsymbol{H}_c}{\partial t} = \tag{4.46}$$

$$\begin{pmatrix} (I_{xxs} + M_s h_s^2)\ddot{\phi} + \left(I_{zxs} + M_s h_s l_{cgs}\right)\cos\phi\,\ddot{\psi} - \\ \left(I_{zxs} + M_s h_s l_{cgs}\right)\sin\phi\,\dot{\psi}\dot{\phi} - \left(I_{zzs} - I_{yys} - M_s h_s^2\right)\sin\phi\cos\phi\,\dot{\psi}^2 \end{pmatrix}\boldsymbol{i}$$
$$+ \begin{pmatrix} \left(I_{zxs} + M_s h_s l_{cgs}\right)\cos\phi\,\ddot{\phi} + \left[\left(I_{yys} + M_s\left(h_s^2 + l_{cgs}^2\right)\right)\sin^2\phi + \left(I_{zzs} + M_s l_{cgs}^2\right)\cos^2\phi\right]\ddot{\psi} \\ + \left(I_{zxs} + M_s h_s l_{cgs}\right)\sin\phi\,\dot{\phi}^2 + \left(I_{zzs} - I_{yys} - M_s h_s^2\right)\sin\phi\cos\phi\,\dot{\psi}\dot{\phi} \end{pmatrix}\boldsymbol{k}$$

Having defined the time derivative of the angular momentum, and recalling that:

$$\sum M = \frac{\partial \boldsymbol{H}_c}{\partial t} = \frac{d\boldsymbol{H}_c}{dt} + \Omega_{\boldsymbol{c}} \times \boldsymbol{H}_{\boldsymbol{c}} \tag{4.47}$$

the following equations of motion can be extracted. The first equation is the roll angular acceleration:

$$(I_{xxs} + M_s h_s^2)\ddot{\varphi} = \begin{pmatrix} \sum T_{xs} - \left(I_{xzs} - M_s h_s l_{cgs}\right)\cos\varphi\,\ddot{\psi} + \left(I_{xzs} - M_s h_s l_{cgs}\right)\sin\varphi\,\dot{\varphi}\dot{\psi} \\ + \left(I_{zzs} - I_{yys} - M_s h_s^2\right)\sin\varphi\cos\varphi\dot{\psi}^2 \end{pmatrix} \tag{4.48}$$

The second equation is the yaw angular acceleration:

$$I_{zzo}\ddot{\psi} = \begin{pmatrix} \sum T_z - \left(I_{xzs} - M_s h_s l_{cgs}\right)\cos\varphi\,\ddot{\varphi} - \left(I_{xzs} - M_s h_s l_{cgs}\right)\sin\varphi\,\dot{\varphi}^2 \\ +\left(I_{zzs} - I_{yys} - M_s h_s^2\right)\sin\varphi\cos\varphi\dot{\psi}^2 \\ -M_s h_s a_x \sin\varphi - \left(I_{zzs} - I_{yys} - M_s h_s^2\right)\sin\varphi\cos\varphi\dot{\varphi}\dot{\psi} \end{pmatrix} \tag{4.49}$$

The sum of the torques acting on the sprung mass about the *x* axis is shown to be:

$$\sum T_{xs} = T_{\varphi f} + T_{\varphi r} + M_s g h_s \sin\varphi \tag{4.50}$$

where:

$$T_{\varphi f} + T_{\varphi r} = -\left(K_{\varphi f} + K_{\varphi r}\right)\varphi - \left(B_{\varphi f} + B_{\varphi r}\right)\dot{\varphi} \tag{4.51}$$

Where $T_{\varphi f}$ and $T_{\varphi r}$ represent the net torque resulting from the suspension. The roll stiffness $K_{\varphi\star}$ and roll damping $B_{\varphi\star}$ are defined as:

$$K_{\varphi\star} = 0.766 K_{s\star}\frac{\mathrm{t}_f{}^2}{2} + K_{r\star} \tag{4.52}$$

$$B_{\varphi\star} = 0.827 B_{s\star}\frac{\mathrm{t}_f{}^2}{2} + B_{r\star} \tag{4.53}$$

Where $K_{s\star}$ is the suspension spring stiffness, $B_{s\star}$ is the shock damping coefficient, $K_{r\star}$ is the anti-roll bar stiffness, and $B_{r\star}$ is the anti-roll bar damping. The test vehicle is not mounted with anti-roll bars, so these values are zero. Recall that ⋆ denotes the front and rear of the vehicle.

The sum of torques about the *z* axis are given by:

$$\sum T_z = \begin{matrix} \left(F_{ylf} + F_{yrf}\right)l_f - \left(F_{ylr} + F_{yrr}\right)l_r + \left(F_{xlf} - F_{xrf}\right)\frac{t_f}{2} \\ +\left(F_{xlr} - F_{xrr}\right)\frac{t_r}{2} + M_{zlf} + M_{zrf} + M_{zlr} + M_{zrr} \end{matrix} \tag{4.54}$$

where the first four terms are the moments developed by the tire forces about the body and the last four terms are the tire self-aligning moments.

Dynamic Weight Transfer Forces

Having derived the equations of motion for both translation and rotation, it is obvious that the summation of forces and moments are necessary in order for each time step to be evaluated. These forces result from interactions of the tires with the ground, and are dependent upon the tire normal force. Because the chassis coordinate system is not an inertial coordinate system, a handful of correction accelerations must be supplied before the coordinate system is valid. The normal acceleration terms used to simplify the longitudinal and lateral equations of motion are one set of accelerations that will develop dynamic weight transfer in the vehicle. The other accelerations are the tangential accelerations provided by driving or braking the tires. These accelerations serve to cancel whatever acceleration is being experienced by the frame of reference and therefore make it an inertial system and valid for the Newtonian laws of physics.

For any longitudinal correcting acceleration the resulting normal force compensation is:

$$F_{\boldsymbol{zax}} = \frac{M_s h_s a_{sx} + M_{uf} h_{uf} a_{ufx} + M_{ur} h_{ur} a_{urx}}{2L} \tag{4.55}$$

and these terms are understood as the moment balance shown in Figure 12: Longitudinal weight transfer force..

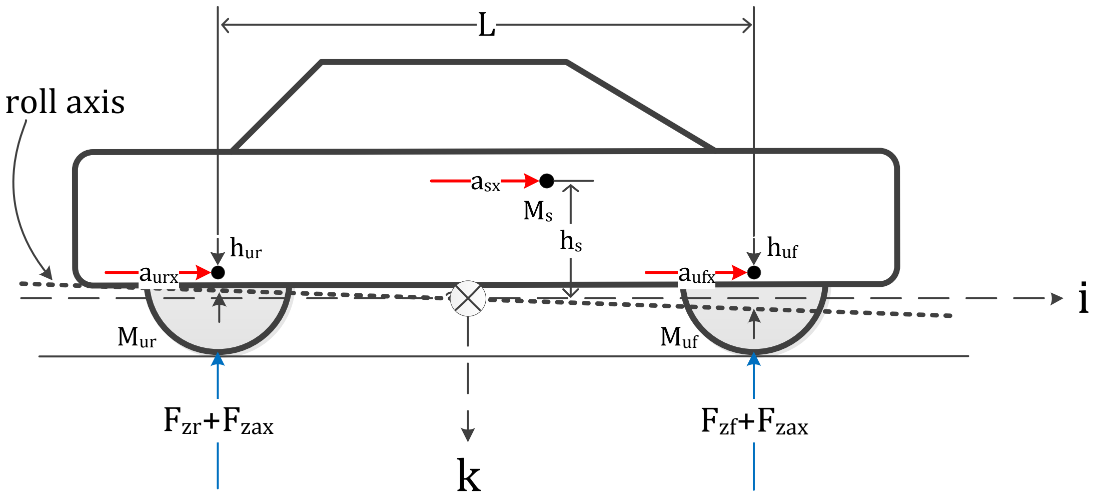


Figure 12: Longitudinal weight transfer force.

For lateral acceleration across the front of the vehicle the normal force compensation is shown to be:

$$F_{\boldsymbol{zayf}} = \frac{1}{t_f}\left(\frac{M_s l_r h_f}{2L} a_{sy} + M_{uf} h_{uf} a_{ufy}\right) \tag{4.56}$$

and the rear normal force compensation is shown to be:

$$F_{\boldsymbol{zayr}} = \frac{1}{t_r}\left(\frac{M_s l_f h_r}{2L} a_{sy} + M_{ur} h_{ur} a_{ury}\right) \tag{4.57}$$

To compensate for the normal force due to roll it is shown that the necessary force is equal to:

$$F_{\boldsymbol{z\phi f}} = -\frac{1}{t_f}\left(K_{\phi f}\phi + B_{\phi f}\dot{\phi}\right) \tag{4.58}$$

for the front half of the vehicle and:

$$F_{\boldsymbol{z\phi r}} = -\frac{1}{t_r}\left(K_{\phi r}\phi + B_{\phi r}\dot{\phi}\right) \tag{4.59}$$

for the rear half of the vehicle. The normal forces on each tire are then found by applying the above equations as follows:

$$F_{\boldsymbol{zlf}} = \frac{Mgl_r}{2L} - F_{zax} + F_{zayf} + F_{\boldsymbol{z\phi f}} \tag{4.60}$$

$$F_{\boldsymbol{zrf}} = \frac{Mgl_r}{2L} - F_{zax} - F_{zayf} - F_{\boldsymbol{z\phi f}} \tag{4.61}$$

$$F_{\boldsymbol{zlr}} = \frac{Mgl_f}{2L} + F_{zax} + F_{zayf} + F_{\boldsymbol{z\phi f}} \tag{4.62}$$

$$F_{\boldsymbol{zrr}} = \frac{Mgl_f}{2L} + F_{zax} - F_{zayf} - F_{\boldsymbol{z\phi f}} \tag{4.63}$$

Tire Models

With the normal force on each wheel determined, the longitudinal and lateral tire forces are derived. Several different methods exist that estimate the behavior of these forces under varying conditions. These models vary from simple linear approximations to more complicated nonlinear models. The slip angle of a tire is taken to be:

$$\alpha_{\star,\circ} = \tan^{-1}\left(\frac{v_{l_{\star,\circ}}}{v_{c_{\star,\circ}}}\right) \tag{4.64}$$

which measures the angle of the tires net velocity to the direction the tire is facing as shown in Figure 13.

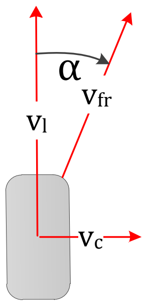


Figure 13: The wheel slip angle is a function of the longitudinal and cornering speeds of the wheel

Empirical tire functions do exist that can handle the observed nonlinearities. One of the most widely used is known as the *Pacejka Magic Formula* (Jazar, 2008), (Pacejka, 2002). The equation set used by the model for the lateral force is:

$$F_{y_{\star,\circ}} = D\sin\left(C\tan^{-1}\left(B\alpha_{\star,\circ} - E\left(B\alpha_{\star,\circ} - \tan^{-1}\left(B\alpha_{\star,\circ}\right)\right)\right)\right) \tag{4.65}$$

$$D = \mu F_{z_{\star,\circ}}$$

$$B = \frac{C_{\alpha_{\star,\circ}}}{CD}$$

$$C, E = shape\ factors$$

where C_α is the cornering stiffness of the tire, and the different constants allow for the development of accurate force approximations over a wide range of operating conditions.

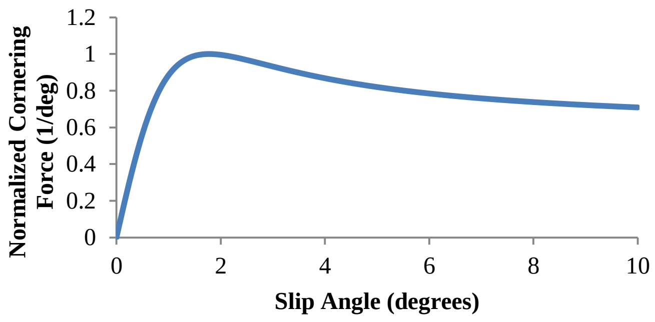


Figure 14: Example cornering force curve of the Magic Formula (B=0.5, C=2.5, D=1, E=1)

Notice that for small slip angles, the cornering force increases linearly with the slip angle. For small angles, the approximation $\sin\theta \simeq \theta$ and $\tan^{-1}\theta \simeq \theta$. Using this, the Magic Formula is reduced to:

$$F_c = DC\big(B\alpha - E(B\alpha - B\alpha)\big) = DCB\alpha = C_\alpha \alpha$$

This linear approximation is the simplest tire model. Under this model, the lateral and longitudinal tire forces are considered to vary linearly with slip angle and slip respectively.

$$F_{c_{\star,\circ}} = C_{\alpha_{\star,\circ}} \alpha_{\star,\circ} \tag{4.66}$$

Because of the strong dependence of the cornering stiffness on the normal force F_z, a factor known as the cornering coefficient is generally used:

$$C_{c\alpha_{\star,\circ}} = \frac{C_{\alpha_{\star,\circ}}}{F_{z_{\star,\circ}}} \tag{4.67}$$

which modifies Equation 2.66 to:

$$F_{c_{\star,\circ}} = C_{c\alpha_{\star,\circ}} F_{z_{\star,\circ}} \alpha_{\star,\circ} \tag{4.68}$$

Recalling that the longitudinal force is assumed to be negligible, the longitudinal force exerted on the tires by the road, in the wheel frame is thus:

$$F_{l_{\star,\circ}} = 0 \tag{4.69}$$

DVM Validation

In order to validate the model developed an outside data set is required. There are three different sets of data that this system is compared to. The first set is the results of a CarSim analysis carried out on a 1990's model Ford Taurus. The second set is the solutions to the NAVDyn block diagram simulator. NAVDyn was written entirely in Simulink using block diagrams. The third data set arises from actual accelerometer measurements taken off a real Ford Taurus (Demerly & Youcef-Toumi, 2000).

The first validation test used was the step steer test. Under the step steer test, the vehicle starts at a specific speed with the steering wheel at the zero angle position. At some point in time, the steering wheel 'steps' from zero to whatever value is desired. The first test performed was for a wheel turn of 42 degrees at a speed of 40 kph. The results are displayed in Figure 15. Solutions obtained from the model developed here appear on the left while the three sets used in this comparison are displayed on the right.

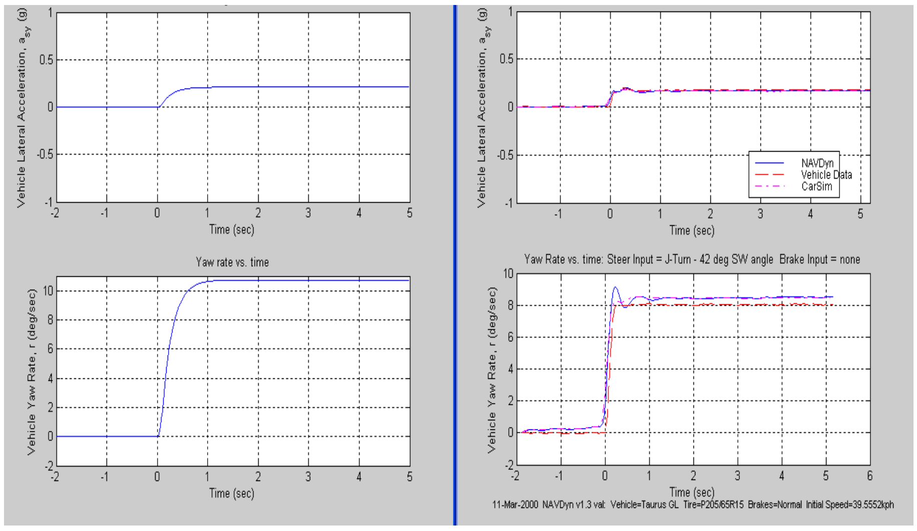


Figure 15: Sprung mass lateral acceleration and yaw rate vs. time for a 42 degree steering wheel step turn at 40 kph.

From the initial test it is shown that the dynamic solution obtained with the DVM generally matches the behavior observed in the other sets. The lateral acceleration of the sprung mass steps up to a value similar to the actual vehicle and in the same time frame, and the same can be said for the yaw rate. Noting the differences between various solutions, the NAVDyn yaw rate solution is much more oscillatory than any of the others. A similar set of observations comes from the 142 degree steering wheel turn at 40 kph as shown in Figure 16. The roll angle and roll rates for this test are also displayed in Figure 17.

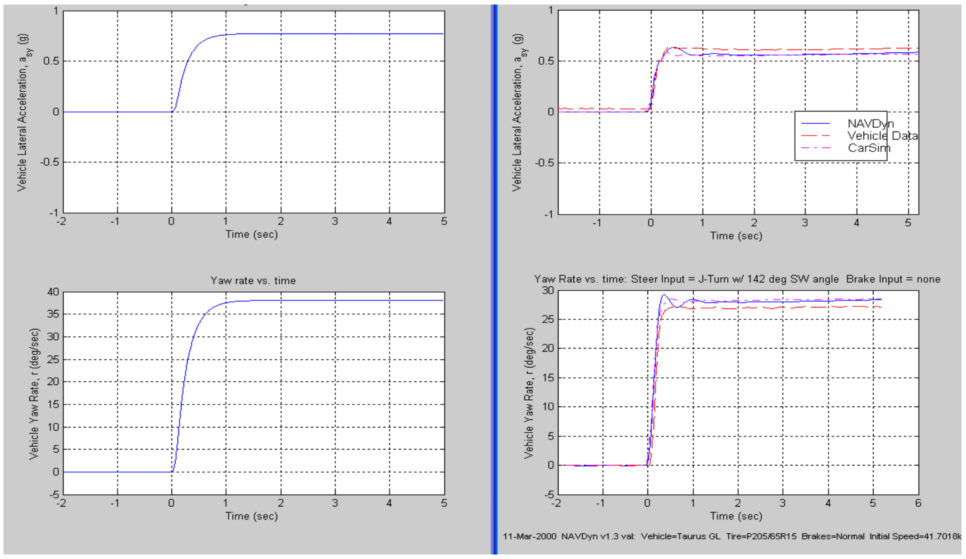


Figure 16: Sprung mass lateral acceleration and yaw rate vs. time for a 142 degree steering wheel step turn at 40 kph

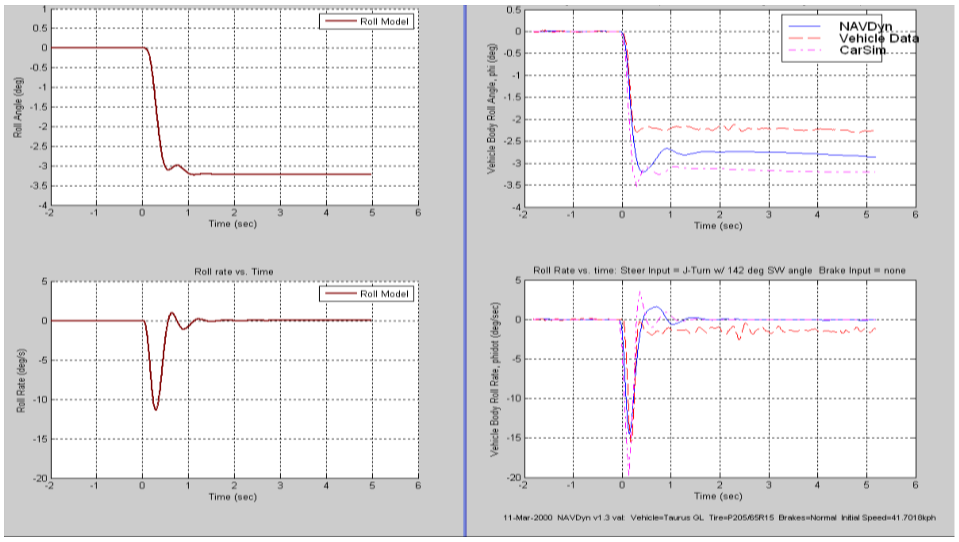


Figure 17: Roll angle and roll rate vs. time for a 142 degree steering wheel step at 40 kph.

Summary and Comparison to NAVDyn Model

The four equations of motion which describe the vehicle system are summarized as follows:

$$M\ddot{x} = \sum F_x + M_s\left(-2h_s\dot{\phi}\dot{\psi}\cos\phi - h_s\ddot{\psi}\sin\phi\right) \tag{4.70}$$

$$M\ddot{y} = \sum F_y - M_s\left(-h_s\ddot{\phi}\cos\varphi + h_s\dot{\phi}^2\sin\varphi + h_s\dot{\psi}^2\sin\phi\right) \tag{4.71}$$

$$\begin{aligned}(I_{xxs} + M_s h_s^2)\ddot{\phi} &= \sum T_{xs} - \left(I_{xzs} - M_s h_s l_{cgs}\right)\left(\ddot{\psi}\cos\phi + \dot{\phi}\dot{\psi}\sin\phi\right) + \\ &\quad \left(I_{zzs} - I_{yys} - M_s h_s^2\right)\dot{\psi}^2\sin\phi\cos\phi\end{aligned} \tag{4.72}$$

$$\begin{aligned}I_{zzo}\ddot{\psi} &= \sum T_z - \left(I_{xzs} - M_s h_s l_{cgs}\right)\left(\ddot{\phi}\cos\phi - \dot{\phi}^2\sin\phi\right) + \left(I_{zzs} - \right. \\ &\quad \left. I_{yys} - M_s h_s^2\right)\left(\dot{\psi}^2\sin\phi\cos\phi - \dot{\phi}\dot{\psi}\sin\phi\cos\phi\right) - M_s h_s\ddot{x}\sin\phi\end{aligned} \tag{4.73}$$

For comparison, the NAVDyn vehicle model is a similar nonlinear, 8 degree of freedom model (Demerly & Youcef-Toumi, Non-linear Analysis of Vehicle Dynamics (NAVDyn): A Reduced Order Model for Vehicle Handling Analysis, 2000). This design accounts for the primary nonlinearities in a vehicle while maintaining enough simplicity to be developed and run in real time. The model uses no more than forty vehicle and tire parameters along to generate an accurate prediction of vehicle behavior. The equations of motion for the NAVDym model are as follows:

$$M\dot{V}_{ox} = \sum F_x + M_s\left(2h_s\dot{\phi}\dot{\psi}\cos\phi + h_s\ddot{\psi}\sin\phi\right) + M\dot{V}_{oy}\dot{\psi} \tag{4.74}$$

$$M\dot{V}_{oy} = \sum F_y - M_s\left(h_s\ddot{\phi}\cos\varphi + h_s\dot{\phi}^2\sin\varphi + h_s\dot{\psi}^2\sin\phi\right) - M\dot{V}_{ox}\dot{\psi} \tag{4.75}$$

$$I_{xxso}\ddot{\phi} = \begin{array}{l}\sum T_{xs} - I_{xzso}\ddot{\psi}\cos\phi - M_s h_s a_{oy}\cos\phi \\ +\left(I_{zzs} - I_{yys} + M_s h_s^2\right)\dot{\psi}^2\sin\phi\cos\phi\end{array} \tag{4.76}$$

$$I_{zzo}\ddot{\psi} = \begin{array}{l}\sum T_z - I_{xzso}\ddot{\phi}\cos\phi + I_{xzso}\dot{\phi}^2\sin\phi + M_s h_s a_{ox}\sin\phi \\ -2\left(I_{yys} - I_{zzs} + M_s h_s^2\right)\dot{\phi}\dot{\psi}\sin\phi\cos\phi\end{array} \tag{4.77}$$

Equations 4.73 and 4.74 respectively describe the lateral and longitudinal accelerations of the vehicle from the chassis frame. The F_x and F_y terms represent the summation of tire forces in their respective directions and are given by:

$$\sum F_x = F_{xlf} + F_{xrf} + F_{xlr} + F_{xrr} \tag{4.78}$$

$$\sum F_y = F_{ylf} + F_{yrf} + F_{ylr} + F_{yrr} \tag{4.79}$$

Equations 4.75 and 4.76 similarly describe the roll and yaw accelerations of the vehicle. The T_{xs} term accounts for the moments caused by inertia and the anti-roll bar; while the T_z term accounts for the torque between the road and the tires. These terms are given by:

$$\sum T_{xs} = T_{\phi f} + T_{\phi r} + a_{sy} M_s h_s \sin\phi + g M_s h_s \sin\phi \tag{4.80}$$

$$\begin{gathered}\sum T_z = \left(F_{ylf} + F_{yrf}\right) l_f - \left(F_{ylr} + F_{yrr}\right) l_r + \left(F_{xlf} - F_{xrf}\right)\frac{t_f}{2} + \left(F_{xlr} - F_{xrr}\right)\frac{t_r}{2} \\ + M_{zlf} + M_{zrf} + M_{zlr} + M_{zrr}\end{gathered} \tag{4.81}$$

The NAVDyn model was used as a reference to compare against. In the process of deriving the equations of motion for this project, it was discovered that some discrepancies exist between the NAVDyn equations and the equations derived for this project. The NAVDyn report, however, does not include a full derivation of the equations of motion. Therefore, a complete decomposition and analysis of the discrepancy is not possible. In summary, the models are very similar, with only minor discrepancies between them. Additionally some terms of the roll and yaw equations were not included in the NAVDyn solution. Since the results from the NAVDyn report are very similar to actual vehicle data, it is concluded that these discrepancies only minimally affect the dynamics of the model.

V. SYSTEM IDENTIFICATION OF THE TEST VEHICLE

Thirty-six different parameters are needed by the DVM to define the geometric and material properties of the vehicle. These are broken down into the geometric dimensions, mass properties, inertial properties, suspension stiffness and damping, and tire slip characteristics. To facilitate the determination of these system parameters, a 3D model of the entire test vehicle was developed in SolidWorks.

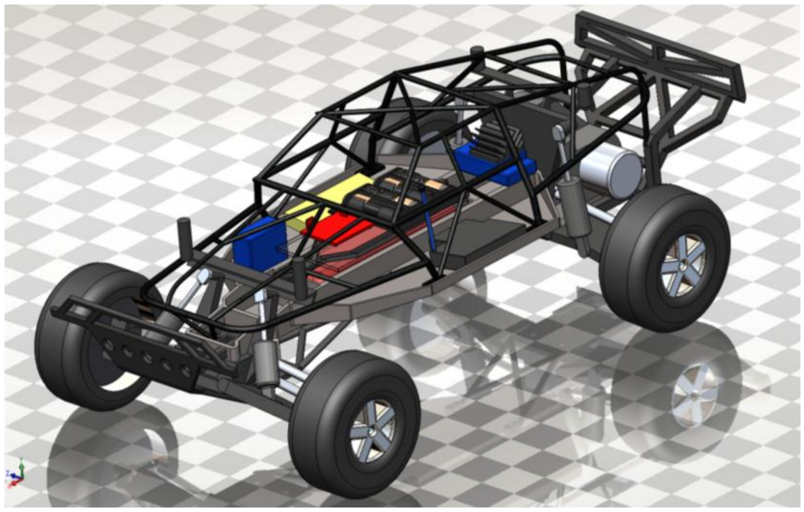


Figure 18: SolidWorks model of the test vehicle.

In order to develop this model, the vehicle was broken down into all its constituent components. Each part was carefully measured and weighed to an accuracy of ±0.5mm and ±0.05 grams, then reconstructed piece by piece using SolidWorks (see APPENDIX E: TEST VEHICLE MATERIAL PROPERTIES). From the individual components, an assembly of the vehicle was then designed. The total mass of the SolidWorks model was 3116.74g while the actual mass was 3139.33g, resulting in an error of only 0.72%. The dimensions and fitting of the vehicle assembly was adjusted manually to match the measurements on the test vehicle, resulting in a very accurately weighed and dimensioned model. Since SolidWorks cannot inherently and accurately position the sprung mass relative to the ground, this positioning was also determined by measuring the distance from the ground to the chassis on the test vehicle; the model was adjusted accordingly. It should be noted that if components are added or removed from the model, the not only will the mass and inertial properties change, but a redetermination of the sprung mass height must be performed by hand. This redetermination is needed because the sprung mass height affects the roll center and center of mass (CM), which are the reference points for the rest of the geometric dimensions.

Sprung Center of Mass and the Static Roll Center

The sprung CM and front/rear roll centers are the three reference points upon which all other geometric and inertial measurements are dependent. The center of mass was determined experimentally using a plum-bob test, (Colwell, 2011). The plum-bob test relies on the fact that if an object is suspended from a line, that line must run vertically through the center of mass of the object. By suspending the vehicle from multiple positions, the center of mass may be found by the intersection of these lines. The sprung CM was also found using the results of the *mass properties* tool in SolidWorks (Dassault Systèmes, 2012). By defining the mass of each part of the vehicle assembly, SolidWorks is able to provide an estimate of the CM.

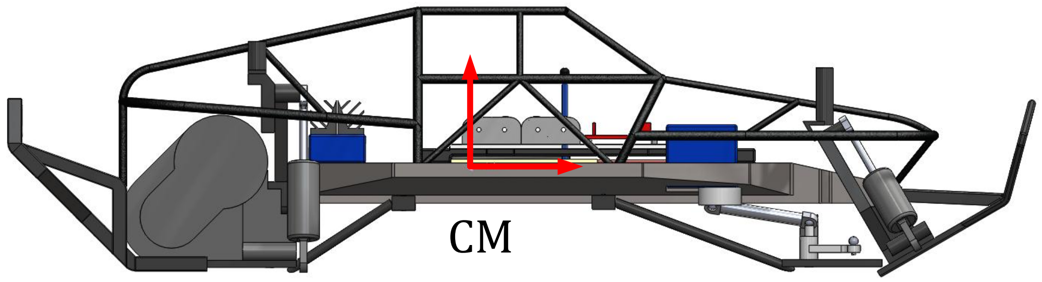


Figure 19: Location of the test vehicle sprung CM in the X-Z plane

The determination of the static roll center was done using 3D modeling of the unsprung assembly. 3D modeling was most practical because of the inherent difficult of measuring distances in a suspension as small as this one. The suspension for the test vehicle is an SLA suspension, which means that the wheel hub is connected to the chassis by two A-bar linkages (see Figure 20 for reference). By isolating the front and rear suspensions from the rest of the body, the roll center is found using the following procedure:

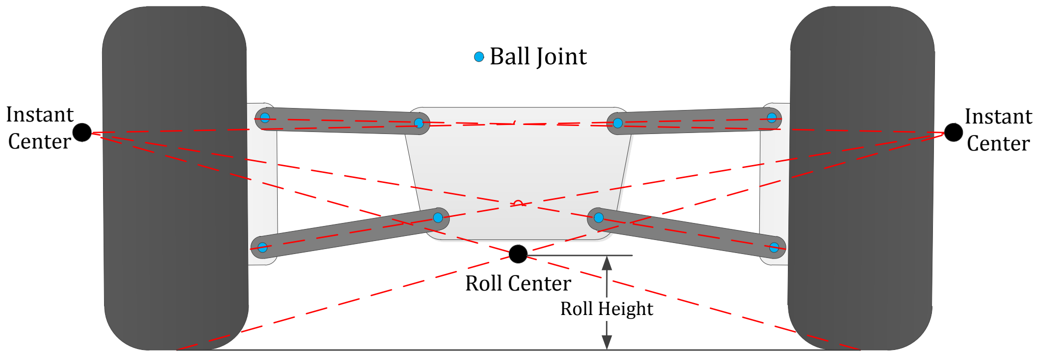


Figure 20: Roll center determination of the rear suspension of the test vehicle.

1) Project a cross section of the suspension onto a plane parallel to the Y-Z plane that passes though the center of the rear suspension
2) Draw a line through the center of the ball joints on either end of each suspension linkage, and project those lines across the Y-Z plane.
3) The line projection from the upper and lower linkages on the left side of the suspension will intersect; this intersection point is called the *instant center*. The other instant center is similarly defined from the intersection of the linkages on the right side suspension.
4) From each instant center, a line is projected across the body of the vehicle to the point where the opposing tire contacts the ground. Due to camber on the wheels, the contact point is on the outer edge of the tire.
5) The intersection point of these lines is the *rear roll center*.
6) The perpendicular distance from the roll center to the ground is the *rear roll height*.
7) Steps 1-6 are then repeated for the front suspension to obtain the *front roll height*.

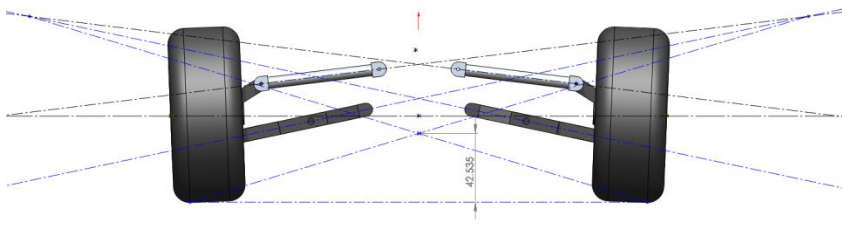


Figure 21: Roll center of the rear suspension of the test vehicle

The values for the static roll center can be found in APPENDIX D: TEST VEHICLE PHYSICAL PARAMETERS FOR DVM.

Vehicle Dimensions and Inertial Properties

The rest of the vehicle geometry is easily obtained using measurements with reference to the location of the CM and front/rear static roll centers. The values for the remaining vehicle lengths, heights, and widths can be found in APPENDIX D: TEST VEHICLE PHYSICAL PARAMETERS FOR DVM.

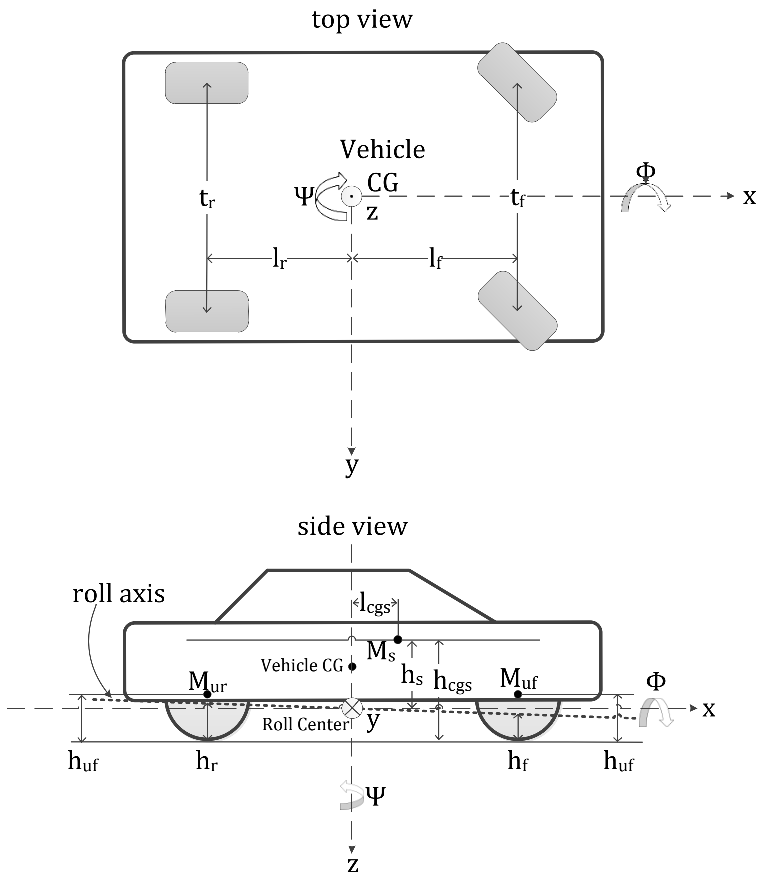


Figure 22: Vehicle dimensions used for the DVM

The requisite mass properties needed for the DVM are the sprung mass and the front/rear unsprung masses. As noted, each piece of the test vehicle was weighed and dimensioned for the SolidWorks model. The mass properties for each individual component can be found in APPENDIX E: TEST VEHICLE MATERIAL

PROPERTIES. The values for the sprung and unsprung masses can be found in APPENDIX D: TEST VEHICLE PHYSICAL PARAMETERS FOR DVM.

As is seen in Equation 4.39, the principal and off-axis values of the sprung mass inertial tensor are also required. These values affect the roll and yaw motion of the vehicle as it rotates. By assigning a mass value to each component in the SolidWorks assembly, SolidWorks is able to provide an estimate of the inertial tensor about the CG. Using the mass values found in APPENDIX E: TEST VEHICLE MATERIAL PROPERTIES, the *mass properties* tool in SolidWorks was then used to approximate the moments and products of inertia of the sprung mass (see APPENDIX D: TEST VEHICLE PHYSICAL PARAMETERS FOR DVM). An experimental method for determining the moments of inertia experimentally is explored in the *Conclusions and Recommendations* chapter of this report. The standard method of determining the products of inertia is done using a two-plane spin balance machine, which was unfortunately not available. A method for determining the moments of inertia on a scaled vehicle by means of a torsion machine has been developed and verified (Witaya, Parinya, & Krissada, 2009).

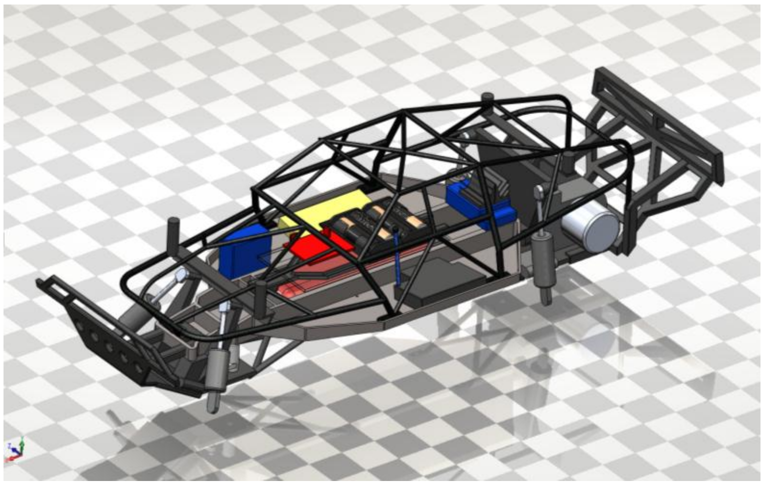


Figure 23: Sprung mass model used to determine the moments and products of inertia.

Spring Stiffness and Shock Damping

The test vehicle has four independent SLA suspensions with different shocks for the front and rear suspension. The front and rear spring stiffness values were determined using a static force-displacement test. Again, the results for the stiffness factors can be found in APPENDIX D: TEST VEHICLE PHYSICAL PARAMETERS FOR DVM.

The coefficient of damping for the front and rear shocks was estimated in simulation by testing different coefficients and comparing the dynamics to experimental results. Too low a value made the system far too stiff, and generated instability about the roll axis. Too high of a damping value over-damped the system and eliminated overshoot in the roll angle in response to step turns. A value of $B_{sr} = B_{sf} = 15$ *N-s/mm* correlated well and was used for experiment. Unfortunately, there are no data sheets available about the shocks which would provide the damping coefficients. Also, the available equipment at Cal Poly does not have the sensitivity to obtain a proper measurement of the damping coefficient. An experiment which would be able to empirically determine the damping coefficients is explored in the *Conclusions and Recommendations* chapter of this report.

Cornering Coefficient

As outlined in the Tire Models section of this report, several methods exist to model the tire cornering coefficients. The industry standard of nonlinear lateral force estimation is the Pacejka Magic Formula, which is a curve fit of empirically determined values (Pacejka, 2002). Another method for estimating the lateral force is to use a linear approximation, which directly relates the lateral force to the slip angle by the cornering coefficient. Recall the equation for the linear approximation is:

$$F_{y_{\star,\circ}} = C_{c\alpha_{\star,\circ}} F_{z_{\star,\circ}} \alpha_{\star,\circ}$$

$$C_{c\alpha_{\star,\circ}} = \frac{C_{\alpha_{\star,\circ}}}{F_{z_{\star,\circ}}}$$

For this project, a typical value for $C_{c\alpha} = 0.2\left(\mathrm{lb}_{\mathrm{Fy}}/\mathrm{lb}_{\mathrm{Fz}}/\mathrm{deg}\right)$ is used (Gillespie, 1992). The effort involved in designing and implementing a lateral force tire tester (LFTT) to directly measure the lateral force response is a massive undertaking, and is beyond the scope of this project. However, a method for directly relating the lateral force as a function of the slip angle is examined in the *Conclusions and Recommendations* chapter of this report.

VI. MODEL PREDICTIVE CONTROL

Model Predictive Control (MPC) is an optimal control technique that governs the output of a system. MPC differs in an important way from other optimization routines in that the optimization is not just solved once, but is instead solved again at each time step. It also uses feedforward compensation, which sets it apart from PID and state feedback control techniques. At every iteration the control signal (in this case, the steer signal from the RRT) is optimized with respect to the current state of the system, the predicted state, the desired state trajectory, the system constraints, and performance weighting. The future state of the model is evaluated from the current time step to the prediction horizon H_p based on a simulated series of optimized actions. These inputs are applied at each time step until the control horizon, H_c. The first of these inputs is applied to the plant, while the rest are discarded. The program then shifts to the next time step and the entire optimization routine is applied again.

MPC has been researched heavily in the past decade and its application towards vehicle stability has been experimentally demonstrated [9]. However, a limit in the effectiveness of MPC stems from its ability to handle nonlinear system models in real time. In many MPC applications, control of a nonlinear model has shown to work at low speeds and low sampling intervals. The computational complexity of the optimization routine limits the sampling rate to 20 Hz, and at speeds above 17 m/s (~40 mph) the MPC is unable to react in real-time[10]. In order to adapt the MPC for real-time processing in a high speed environment, a modification must be made. Instead of using a nonlinear plant model, a linear time-varying (LTV) model may be used. Autonomous tracking on full scale vehicles on icy roads at speeds up to 21 m/s has been shown to be effective with LTV MPC [9]. The responsiveness is improved further by assuming a constant, linearized model. In this way the controller does not need to update the linearization at each time step, but assumes that the plant model is constant. The linearized model has been tested under several driving maneuvers, and has demonstrated comparable dynamic response to the dynamic vehicle model (further details may be found in *Linearization of the Dynamic Vehicle Model* on page 46).

State Space Representation of the Dynamic Vehicle Model

For this project, an LTI MPC is used to control the steer angle of the vehicle. Braking is used only if the RRT cannot find a trajectory around the object, so the MPC needs only to focus on steering. The plant model is linearized around a stable, non-zero equilibrium point. The linearized plant model was chosen because of the inhibiting computational complexity of a LTV MPC. Further details regarding this are explored in the *Conclusions and Recommendations* section of this report. As previously discussed, Equations 4.69-4.72 are the four coupled, nonlinear equations of motion (EoMs). A state space representation of the model is needed to facilitate development of a controller. Assigning $\xi \overset{\text{def}}{=} \left(\dot{x}\ \dot{y}\ \dot{\phi}\ \phi\ \dot{\psi}\ \psi\right)^T$ and $u \overset{\text{def}}{=} \delta$, the *desired* form of the nonlinear equations of motion may be expressed compactly as:

$$\dot{\xi}(t) = f\big(\xi(t), u(t)\big) \tag{4.1}$$

To represent the model in state space form, all the acceleration terms from Equations 2.69-2.72 must be extracted from the right hand side of the equation. This extraction leaves a coupled, nonlinear equation of the form:

$$\boldsymbol{M}\dot{\xi}(t) = F\big(\xi(t), u(t)\big) \tag{4.2}$$

Where the mass matrix and right side of the equation are defined as:

$$M = \begin{bmatrix} M & 0 & 0 & 0 & M_s h_s \sin\phi & 0 \\ 0 & M & -M_s h_s \cos\phi & 0 & 0 & 0 \\ 0 & M_s h_s \cos\phi & I_{xxs} + M_s {h_s}^2(1+\cos^2\phi) & 0 & \left(I_{xzs} + 2M_s h_s l_{cgs}\right)\cos\phi & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ M_s h_s \sin\phi & 0 & \left(I_{xzs} + M_s h_s l_{cgs}\right)\cos\phi & 0 & I_{zzo} & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.3}$$

$$F\big(\xi(t),u(t)\big) = \begin{bmatrix} \sum F_x - 2M_s h_s\,\dot{\phi}\dot{\psi}\cos\phi \\ \sum F_y - M_s h_s \sin\phi\,(\dot{\phi}^2 + \dot{\psi}^2) \\ \begin{pmatrix} \sum T_{xs} + (I_{xzs} + M_s h_s l_{cgs})\dot{\phi}\dot{\psi}\sin\phi + (I_{zzs} - I_{yys})\dot{\psi}^2 \sin\phi\cos\phi \\ + M_s {h_s}^2 \dot{\phi}^2 \sin\phi\cos\phi - M_s h_s \dot{\psi}\dot{x}\sin\phi \end{pmatrix} \\ \dot{\phi} \\ \sum T_z - (I_{xzs} + M_s h_s l_{cgs})\,\dot{\phi}^2 \sin\phi - (I_{zzs} - I_{yys} - M_s h_s^2)\dot{\psi}\dot{\phi}\sin\phi\cos\phi \\ \dot{\psi} \end{bmatrix} \tag{4.4}$$

Moving the mass matrix M to the other side of the equation yields:

$$\dot{\xi}(t) = \boldsymbol{M}^{-1} F\big(\xi(t),u(t)\big) \tag{4.5}$$

$$\dot{\xi}(t) = f\big(\xi(t),u(t)\big) \tag{4.6}$$

The decoupling procedure assumes that the mass matrix is square and invertible, the justification for this can be found at the end of this chapter. The rest of the equations which define the parameters of equation 4.4 are presented again.

The sum of forces and torques exerted on the vehicle body is derived as:

$$\sum F_x = F_{xlf} + F_{xrf} + F_{xlr} + F_{xrr} \tag{4.7}$$

$$\sum F_y = F_{ylf} + F_{yrf} + F_{ylr} + F_{yrr} \tag{4.8}$$

$$\sum T_{xs} = T_{\phi f} + T_{\phi r} - M_s g h_s \sin\phi \tag{4.9}$$

$$\sum T_z = (F_{ylf} + F_{yrf})l_f - (F_{ylr} + F_{yrr})l_r + (F_{xlf} - F_{xrf})\frac{t_f}{2} + (F_{xlr} - F_{xrr})\frac{t_r}{2} \tag{4.10}$$

Recall that for compactness $\star$ denotes either the front or rear of the vehicle, while $\circ$ denotes either the left or right side of the vehicle. This notation eliminates the need to redundantly print equations which are identical for the front left, front right, rear left, and rear right. Therefore, the components of force exerted on the tires by the road, in the vehicle frame are:

$$F_{x_{\star,\circ}} = F_{l_{\star,\circ}} cos\delta_\star - F_{c_{\star,\circ}} sin\delta_\star \tag{4.11}$$

$$F_{y_{\star,\circ}} = F_{l_{\star,\circ}} sin\delta_\star + F_{c_{\star,\circ}} cos\delta_\star \tag{4.12}$$

The cornering and longitudinal force exerted on the tires by the road, in the wheel frame is:

$$F_{l_{\star,\circ}} = 0 \tag{4.13}$$

$$F_{c_{\star,\circ}} = -C_{c\alpha} F_{z_{\star,\circ}} \alpha \tag{4.14}$$

For the calculation of the normal force F_z, the dynamic weight transfer force (DWTF) equations add a great deal of computational complexity while minimally affecting the normal force. As the vehicle corners, the center of mass shifts. This shift changes the normal forces experiences by the tires (for example, the outside wheels of a vehicle in a turn will generate a larger normal force, while the inner wheels will have less force). To determine the effect of the DWTFs, a simulation was run with a step turn of 42° using the vehicle data from a Ford Taurus (Demerly & Youcef-Toumi, 2000). Based on the results seen in Figure 24-Figure 27 the difference in results was negligible. In fact, in Figure 24 and Figure 25, the data matches so much that the static and dynamic plots are nearly indistinguishable.

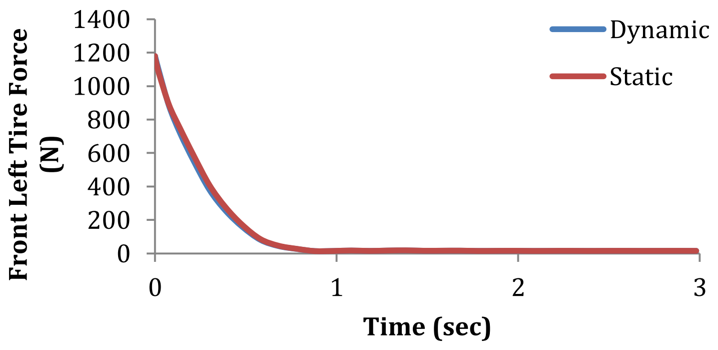


Figure 24: Comparison of the dynamic and static lateral front left tire forces

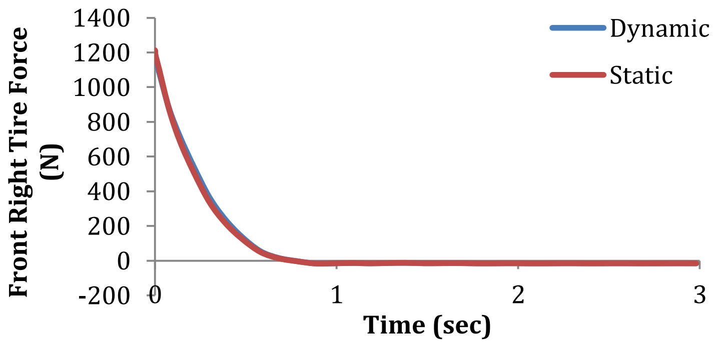


Figure 25: Comparison of the dynamic and static lateral front right tire forces

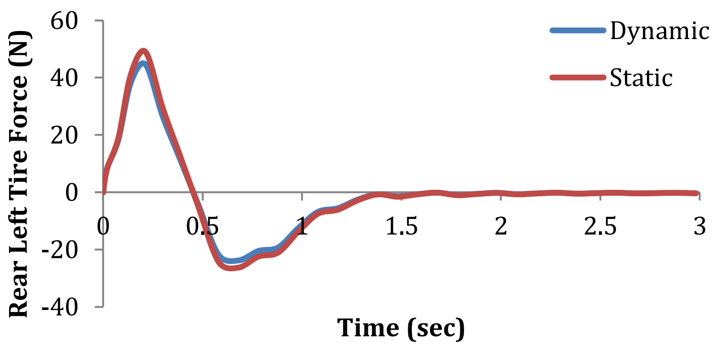


Figure 26: Comparison of the dynamic and static lateral rear left tire forces

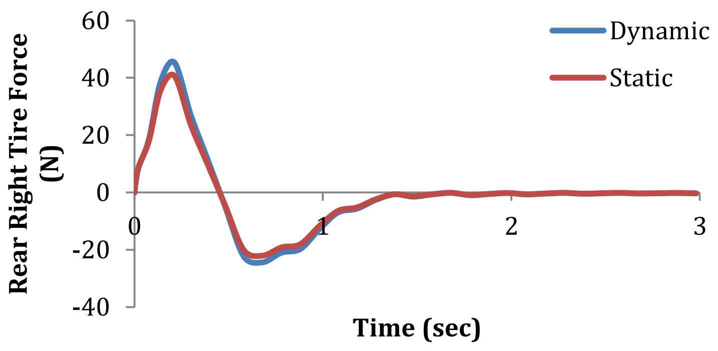


Figure 27: Comparison of the dynamic and static lateral rear right tire forces

The DWTFs depend greatly on the sprung mass lateral acceleration, a_{sy} (see Equations 4.56 and 4.57). However, even for values of $a_{sy} \geq a_y$, the effect of the DWTFs do not contribute much to the tire normal forces Based on these results, the following assumption was made:

Assumption 1: The dynamic weight transfer forces are negligible for small sprung accelerations

The normal force exerted on the tires by the vehicle body is therefore:

$$F_{z_{\star,\circ}} = \frac{Mgl_{\star}}{2L} \tag{4.15}$$

The slip angle of the tires is given as:

$$\alpha_{\star,\circ} = tan^{-1}\left(\frac{v_{c_{\star,\circ}}}{v_{l_{\star,\circ}}}\right) \tag{4.16}$$

The cornering and longitudinal velocities of the tires in the wheel frame:

$$v_{c_{\star,\circ}} = v_{x_{\star,\circ}} cos\delta_\star - v_{y_{\star,\circ}} sin\delta_\star \tag{4.17}$$

$$v_{l_{\star,\circ}} = v_{x_{\star,\circ}} sin\delta_\star + v_{y_{\star,\circ}} cos\delta_\star \tag{4.18}$$

The velocity components of the tires in the vehicle frame:

$$\begin{aligned} v_{x_{f,l}} &= \dot{x} + \frac{t_f}{2}\dot{\psi} & v_{y_{f,l}} &= \dot{y} + l_f\dot{\psi} \\ v_{x_{f,r}} &= \dot{x} - \frac{t_f}{2}\dot{\psi} & v_{y_{f,l}} &= \dot{y} + l_f\dot{\psi} \\ v_{x_{r,l}} &= \dot{x} + \frac{t_r}{2}\dot{\psi} & v_{y_{r,l}} &= \dot{y} - l_r\dot{\psi} \\ v_{x_{r,r}} &= \dot{x} - \frac{t_r}{2}\dot{\psi} & v_{y_{r,r}} &= \dot{y} - l_r\dot{\psi} \end{aligned} \tag{4.19}$$

Linearization of the Dynamic Vehicle Model

Consider the following continuous nonlinear system described in Equation (2.4):

$$\dot{\xi}(t) = f\big(\xi(t), u(t)\big) \tag{4.20}$$

The MPC requires that the system resemble the A, B, C, D form of a state space model. For the purposes of linearization it was assumed that the roll angle would remain small. This assumption is justified by comparison to the DVM at the end of this section.

Assumption 2: The roll angle is small ($sin\,\varphi = \varphi$; $cos\,\varphi = 1$)

For the purposes of this project, the initial state ξ_0 has a constant longitudinal velocity v_{x_0}, and zero for all other terms. This state represents a vehicle moving at constant speed in a straight line. Additionally since the vehicle begins the maneuver with no steer angle, u_0 is zero. Accounting for the assumption 3, the system is approximated through linearization around a state $\xi_0 = \big(v_{x_0}\ 0\ 0\ 0\ 0\ 0\big)$ and an input $u_0 = 0$ such that:

$$\delta\dot{\xi}(t) = A_0\delta\xi(t) + B_0\delta u(t) \tag{4.21}$$

$$A_0 = \left.\frac{\delta f}{\delta \xi}\right|_{\xi_0,u_0} \qquad B_0 = \left.\frac{\delta f}{\delta u}\right|_{\xi_0,u_0}$$

$$\delta\dot{\xi}(t) = \dot{\xi}(t) - \dot{\xi}_0 \qquad \delta\xi(t) = \xi(t) - \xi_0 \qquad \delta u(t) = u(t) - u_0$$

From Equation 4.6 it is found that for the initial state vector $\xi_0 = \left(v_{x_0}\ 0\ 0\ 0\ 0\ 0\right)$ and initial input $u_0 = 0$:

$$\dot{\xi}(\xi_0, u_0) = (0\ 0\ 0\ 0\ 0\ 0)^T$$
$$A_0\xi_0 = (0\ 0\ 0\ 0\ 0\ 0)^T$$
$$B_0u_0 = (0\ 0\ 0\ 0\ 0\ 0)^T$$

Thus Equation 4.21 reduces to:

$$\dot{\xi}(t) = A_0\xi(t) + B_0u(t) \tag{4.22}$$

The validity of the linearized model was tested in simulation for an applied step turn of 20° against the dynamic vehicle model. The results shown in Figure 28, Figure 29, and Figure 30 below demonstrate a negligible difference between the LTI model and the DVM.

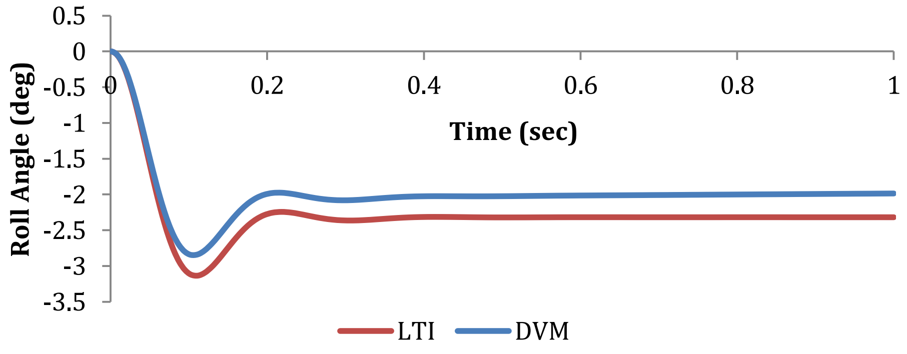


Figure 28: DVM vs LTI roll angle response to a step turn of 20°

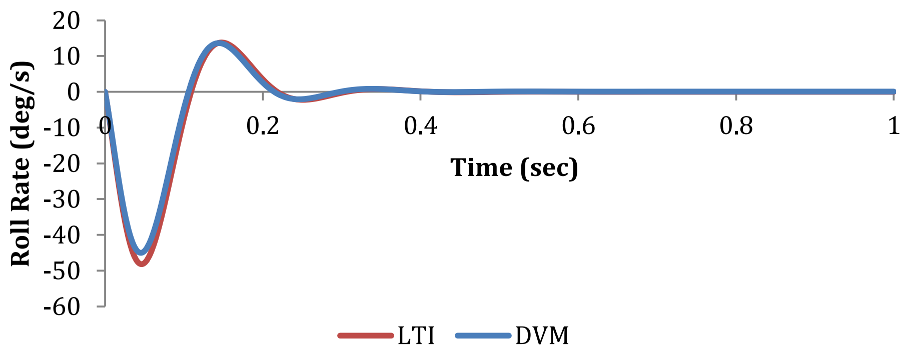


Figure 29: DVM vs LTI roll rate response to a step turn of 20°

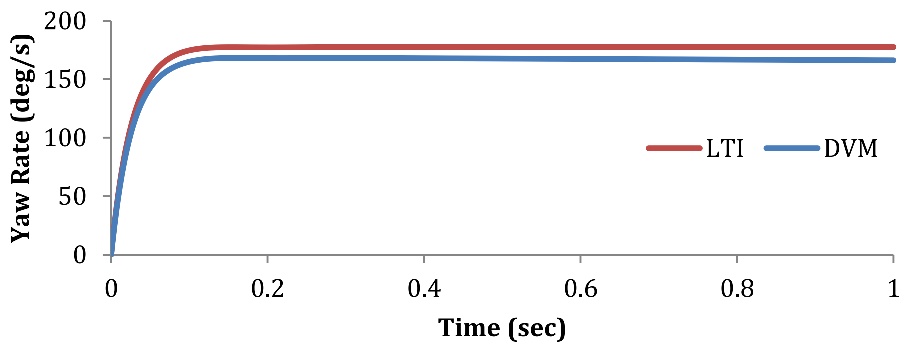


Figure 30: DVM vs LTI roll angle response to a step turn of 20°

The LTI appears to have a stiffer response, which results in a larger steady state roll angle. The LTI also has constant forward velocity, while in the DVM the forward velocity gradually decreases. The decrease in speed results in a smaller lateral acceleration, and accordingly decreases the roll angle. Notwithstanding, the LTI roll angle differs only by +0.33° from the DVM, the roll rate response is nearly identical, and the steady state yaw rate differs by only +6.8%. For the MPC, this is not a significant deviation. What is most important is that the general shapes of the response curves are correct – which they are. The MPC will assume from the linearized model that the roll and yaw response is slightly greater than it actually is, and this may be compensated for by proper tuning of the controllers input weights.

Discretization of the Dynamic Vehicle Model

The crash avoidance algorithm and MPC both operate at a constant sampling rate, so system must first be discretized. Through experimentation, it was found that 20 Hz was an ideal rate for the RRT to run. This rate allows the RRT to make enough adjustments to the trajectory without having a compromise in performance. The constant sampling rate means that the continuous system described in Equation 4.1 must be discretized. Using the Euler Method (Falcone, Borrelli, Tseng, Asgari, & Hrovat, 2008), the state trajectory may be approximated for small time steps by:

$$\dot{\xi}(t) \simeq \frac{\left(\xi(t + \triangle t) - \xi(t)\right)}{\triangle t} \tag{4.23}$$

Reorganizing terms and substituting $\dot{\xi}(t)$ for Equation (3.3):

$$\xi(t + \triangle t) \simeq \dot{\xi}(t) \triangle t + \xi(t) \tag{4.24}$$

$$\xi(t + \triangle t) \simeq \left(A_0\xi(t) + B_0u(t)\right) \triangle t + \xi(t) \tag{4.25}$$

Defining a constant sampling rate $\tau = \triangle t$ and series of time steps $k = 0 \rightarrow n$ such that:

$$\xi[k] \stackrel{\text{def}}{=} \xi(k\tau) \qquad\qquad u[k] \stackrel{\text{def}}{=} u(k\tau)$$

The previous equation is rewritten as:

$$\xi[k+1] = (A_0\xi[k] + B_0u[k])\tau + \xi[k] \tag{4.26}$$

Reorganizing terms:

$$\xi[k+1] = (I + \tau A_0)\xi[k] + \tau B_0u[k] \tag{4.27}$$

The following terms are grouped to obtain the standard state space form:

$$A_k \stackrel{\text{def}}{=} (I + \tau A_0) \qquad\qquad B_k \stackrel{\text{def}}{=} \tau B_0$$

The final form of the linearized, discretized system is therefore:

$$\xi[k+1] = A_k \xi[k] + B_k u[k] \tag{4.28}$$

The MPC Cost Function

The fundamental concept of the MPC is the minimization of the cost function. This minimization guarantees an *optimal* output for a given set of weights, constraints, and predicted states of the system. Consider the cost function $J(\xi[k], \Delta u[k])$ defined as follows:

$$\begin{aligned} J(\xi[k], \Delta u[k]) = \sum_{i=0}^{H_p - 1} \Big(&\left\| \xi[k+i+1] - \xi_{ref}[k+i+1] \right\|_Q^2 + \|\Delta u[k+i]\|_{R_{\Delta u}}^2 \\ &+ \left\| u[k+i] - u_{ref}[k+i] \right\|_{R_u}^2 + \rho \varepsilon^2 \Big) \end{aligned} \tag{4.29}$$

where the following notation applies: $\|\xi[k]\|_Q^2 = \xi[k]^T Q \xi[k]$. In this equation H_p is the prediction horizon. The predicted signal $\xi[k+1]$ is based on the current state $\xi[k]$ and input $u[k]$ according to Equation 4.28, while the reference signal $\xi_{ref}[k]$ represents the desired state trajectory. The predicted and desired states, ξ and ξ_{ref}, are vectors consisting of the output state variables $\xi = \left(\dot{x}\ \dot{y}\ \dot{\phi}\ \phi\ \dot{\psi}\ \psi\right)^T$. The input signal $u[k] = \delta_k$ is the steering angle at time-step k. The incremental input $\Delta u[k] = \Delta\delta_k$, where $\Delta\delta_k = \delta_k - \delta_{k-1}$. The incremental input is only considered from the current state to the control horizon, H_c. For $i \geq H_c$: $\Delta u[k] = 0$. The reference input $u_{ref}[k]$ is the desired input. u, Δu, and u_{ref} are all scalar values. The output weight matrix Q is an n_ξ– by– n_ξ matrix with nonnegative diagonal elements, ω_{ij}^{ξ}, corresponding to the weight of the state outputs. The input weight and input weight-rate matrices $R_{\Delta u}$and R_u are n_u-by-n_u matrices (in this case they are scalars, since the only input is the steer angle) consisting of the respective weight elements $\omega^{\Delta u}$ and ω^u. The slack variable ε is a constraint softening factor on the state output while the slack weight ρ determines the tolerance of constraint violation.

Constraints and Constraint Softening

Constraints are used in MPC to prevent the input signal and the system outputs from exceeding certain bounds. Here the constraints include the maximum and minimum steer angle, roll angle, roll rate, and yaw rate. The steer angle has a physical constraint because the steering has only a limited range of motion. The maxima and minima with respect to the state variables reflect desired performance criteria. However, in some circumstances these constraints may cause the controller to behave undesirably. Constraints may conflict with each other, resulting in a situation in which a solution is mathematically unrealizable. Alternatively, the controller may ignore input minimization to reach the required state constraints. Because of such cases, constraint softening is used to relax the tolerances and allow for violations of the constraints.

There are eight parameters which affect the degree to which the constraints are softened: The slack variable ε, the slack weight ρ, and the Equal Concern for Relaxation (ECR) vectors V_{min}^{u}, V_{max}^{u}, $V_{min}^{\Delta u}$, $V_{max}^{\Delta u}$, V_{min}^{y}, V_{max}^{y}. The slack variable ε is a normalized parameter which affects the weight of the constraint softening in both the cost function and the constraint bounds. The slack weight ρ appears in the cost function as a weight on ε, and acts to penalize the constraint softening. A larger value of ρ relative to the input and output weights prioritizes the minimization of constraint violations. The slack variable is a tunable, normalized parameter and the slack weight is a constant value where:

$$0 \leq \varepsilon \leq 1 \tag{4.30}$$

$$\rho \overset{\text{def}}{=} 10^5 * max\{\omega_{ij}^{y}, \omega^{\Delta u}, \omega^{u}\} \tag{4.31}$$

The soft constraints affect the bounds of the state outputs as follows:

$$u_{min} - \varepsilon V_{min}^{u} \leq u[k+i] \leq u_{max} + \varepsilon V_{max}^{u} \tag{4.32}$$

$$\Delta u_{min} - \varepsilon V_{min}^{\Delta u} \leq \Delta u[k+i] \leq \Delta u_{max} + \varepsilon V_{max}^{\Delta u} \tag{4.33}$$

$$\xi_{min} - \varepsilon V_{min}^{\xi} \leq \xi[k+i] \leq \xi_{max} + \varepsilon V_{max}^{\xi} \tag{4.34}$$

where the ECR vectors are tolerances that affect the constraint bounds. Larger ECR values soften the constraints while smaller ECR values harden the constraints. In this case, the values for the ECR vectors are:

$$V_{min}^{\xi} = V_{max}^{\xi} = 1;$$

$$V_{min}^{u} = V_{max}^{u} = V_{min}^{\Delta u} = V_{max}^{\Delta u} = 0;$$

Since the input to the system is a physical requirement, the steer angle does not make use of soft constraints.

As is seen in both the cost function and the constraint boundary relations, if $\varepsilon = 0$ then no priority is given in the cost function towards the minimization of constraint violations, and the constraints have no slack.

MPC Optimization Formulation

At each time step k, the following problem is considered:

$$\begin{array}{lc} minimize: & J(\xi[k+i], \Delta u[k+i]) \\ & for\ i = 0, \dots, H_p - 1 \\ subject\ to: & \xi[k+1] = A_k \xi[k] + B_k u[k] \\ & u_{min} - \varepsilon V_{min}^{u} \leq u[k+i] \leq u_{max} + \varepsilon V_{max}^{u} \\ & \Delta u_{min} - \varepsilon V_{min}^{\Delta u} \leq \Delta u[k+i] \leq \Delta u_{max} + \varepsilon V_{max}^{\Delta u} \\ & \xi_{min} - \varepsilon V_{min}^{\xi} \leq \xi[k+i] \leq \xi_{max} + \varepsilon V_{max}^{\xi} \\ & \Delta u[k+j] = 0 \\ & for\ j = H_c, \dots, H_p - 1 \end{array} \tag{4.35}$$

where the predicted input $u[k+i]$ is the value received from the RRT. The future state $\xi[k+i]$ is estimated by an internal Kalman filter (Bemporad, Morari, & N., 2012). The output of the optimization at each time step is the incremental change in input $\Delta u[k]$. Accordingly, the input to the system from the MPC is:

$$u[k] = u[k-1] + \Delta u[k] \tag{4.36}$$

Quadratic Programming (QP) Solver

An unconstrained optimal control problem is solved analytically. On the other hand, when constraints are introduced to the problem (such as this one), there is no analytical solution and a QP solver is needed. For these cases MATLAB uses QPKWIK, a robust QP solver, to handle the optimization routine. The details of how MATLAB handles the optimization problem, including the matrices and formulations used, may be found in (MPC users guide reference). The MATLAB user's guide summarizes the advantages and limitations of the solver as follows:

> *"The toolbox uses the KWIK algorithm [1] to solve the QP problem… In the very first control step, KWIK uses a cold start, in which the initial guess is the unconstrained solution described in Model Predictive Control Computation. If this x satisfies the constraints, it is the optimal QP solution, x^*, and the algorithm terminates. Otherwise this means that at least one of the linear inequality constraints must be satisfied as an equality. In this case, KWIK uses an efficient, numerically robust strategy to determine the active constraint set satisfying the standard optimality conditions. In the following control steps, KWIK uses a warm start. In this case, the active constraint set determined at the previous control step becomes the initial guess for the next.*
>
> *Although KWIK is robust, you should consider the following:*
>
> - *One or more linear constraints might be violated slightly due to numerical round-off errors. The toolbox employs a nonadjustable relative tolerance. This tolerance allows a constraint to be violated by 10^{-6} times the magnitude of each term. Such violations are considered normal and do not generate warning messages.*
> - *The toolbox also uses a nonadjustable tolerance when it tests a solution for optimality.*
> - *The search for the active constraint set is an iterative process. If the iterations reach a problem-dependent maximum, the algorithm terminates.*
> - *If your problem includes hard constraints, these constraints might be infeasible (impossible to satisfy). If the algorithm detects infeasibility, it terminates immediately.*
>
> *In the last two situations, with an abnormal outcome to the search, the controller will retain the last successful control output."*

Stability Considerations due to Decoupling

To make Equation 4.2 match the form of Equation 4.1, the mass matrix $\boldsymbol{M}$ must be moved to the right side of the equation. This manipulation can only be done if $\boldsymbol{M}$ is invertible, which means it must be square (which it is) and have a non-zero determinant. Solving for the determinant yields:

$$\begin{aligned} det(\boldsymbol{M}) = \quad & M^2\left(I_{xxs} + M_s {h_s}^2(1+\cos^2\phi)\right)I_{zzo} \\ & -M^2\left(\left(I_{xzs} + 2M_s h_s l_{cgs}\right)\cos\phi\right)\left(\left(I_{xzs} + M_s h_s l_{cgs}\right)\cos\phi\right) \\ & -M(-M_s h_s \cos\phi)(M_s h_s \cos\phi)I_{zzo} \end{aligned} \tag{1.1}$$

Except for the roll angle, all other terms are constant vehicle parameters. Setting the determinant equal to zero defines the condition for which the mass matrix is not invertible. Rearranging terms yields an equality which, if holds, means the decoupling procedure is not valid:

$$\cos^2\phi = \frac{-M^2 I_{zzo}\left(I_{xxs} + M_s {h_s}^2\right)}{\begin{pmatrix} M^2 M_s {h_s}^2 I_{zzo} + M{M_s}^2{h_s}^2 I_{zzo} - M^2 {I_{xzs}}^2 \\ -2M^2{M_s}^2{h_s}^2{l_{cgs}}^2 - 3M^2 M_s h_s l_{cgs} I_{xzs} \end{pmatrix}} \tag{1.2}$$

Using the values for the test vehicle in APPENDIX A: , Equation 4.6 reduces to:

$$\cos^2\phi = -0.5645$$

Since the square root of a negative number is complex – and the roll angle *cannot* be a complex number – this condition is a physical impossibility. Therefore the decoupling procedure is valid for this system.

VII. SIMULATION RESULTS

The parameters following parameters are defined in depth in the

MODEL PREDICTIVE CONTROL chapter. For review: τ is the sampling interval, and $1/_{\tau}$ is the rate at which the RRT and MPC run at. H_p is the prediction horizon – the number of sample intervals that the cost function (Equation 4.29) is optimized over. The control horizon, H_c, is the number of samples under which the input signal is optimized. The MPC assumes that for optimization between H_c and H_p the input u is held constant at the final value.

The constraints on the input, input rate, and state outputs are used to limit these values to within desired performance. ω_{ij}^{ξ}, corresponds to the weight of the state outputs and are the elements of the output weight matrix Q (see Equation 4.29). The weight elements $\omega^{\Delta u}$ and ω^{u} are the elements of the input weight and input weight-rate matrices $R_{\Delta u}$and R_u respectively. The larger relative weights are prioritized for minimization in the MPC.

The slack variable ε is a constraint softening factor on the state output while the slack weight ρ determines the tolerance of constraint violation. The ECR vectors V_{min}^{u}, V_{max}^{u}, $V_{min}^{\Delta u}$, $V_{max}^{\Delta u}$, V_{min}^{y}, V_{max}^{y} are tolerances that affect the constraint bounds. Larger ECR values soften the constraints while smaller ECR values harden the constraints.

These parameters were tuned to the following values to test the controller in simulation, designated as Controller A:

$$Sample\ Time: \qquad \tau = 0.05\ sec$$

$$Horizons: \qquad H_p = 10\tau \quad H_c = 2\tau$$

$$Constraints: \quad \begin{Bmatrix} -\infty\ m/s & \leq \dot{x} \leq & \infty\ m/s \\ -\infty\ m/s & \leq \dot{y} \leq & \infty\ m/s \\ -\infty\ r/s & \leq \dot{\phi} \leq & \infty\ r/s \\ -0.1\ rad & \leq \phi \leq & 0.1\ rad \\ -\infty\ r/s & \leq \dot{\psi} \leq & \infty\ r/s \\ -\infty\ rad & \leq \psi \leq & \infty\ rad \end{Bmatrix}$$

$$\begin{Bmatrix} -20\ deg & \leq \delta \leq & 20\ deg \\ \infty\ deg/s & \leq \Delta\delta \leq & \infty\ deg/s \end{Bmatrix}$$

$$Weights: \quad \begin{Bmatrix} \omega^{u} = 20 \\ \omega^{\Delta u} = 0.1 \\ \omega^{\dot{\phi}} = 20 \\ \omega^{\phi} = 50 \\ \omega^{\dot{\psi}} = 1 \\ \omega^{\dot{x}} = \omega^{\dot{y}} = \omega^{\dot{x}} \\ \omega_{ij}^{\xi} = 0\ for\ i \neq j \end{Bmatrix}$$

$$Slack: \qquad \varepsilon = 0.75 \quad \rho = 50(10)^5$$

$$ECR: \quad \begin{Bmatrix} V_{min}^{\xi} = V_{max}^{\xi} = 1 \\ V_{min}^{u} = V_{max}^{u} = V_{min}^{\Delta u} = V_{max}^{\Delta u} = 0 \end{Bmatrix}$$

A sample time of 0.5 seconds was used to match the sample rate of the RRT. A prediction horizon of 10 intervals is thus able to examine the future state up to 0.5 seconds in the future. A control horizon of 2 intervals is accordingly able to optimize the steer signal 0.1 seconds in advance. A minimal amount of constraints were used, since the added complexity provided no more meaningful control over the dynamics. The roll angle is constrained to ±0.1 radians (~5.7°), since that is the primary focus of the controller. This is not a hard constraint, as noted by the ECR vectors V_{min}^{ξ} and V_{max}^{ξ} which are equal to 1. This means that the constraint placed on the roll angle may be violated if it contributes optimally to minimize the other weighted variables – the roll rate and yaw rate. Like almost all ground vehicles, the front wheels of the test vehicle has a limited maximum and minimum steer angle. Accordingly, the steer angle is constrained to within its physical limits of ±20°. The constraint on the steer angle is hard since the steer angle cannot exceed its physical limitations. No constraint was placed on the steer angle rate change, however. This constraint would needlessly limit the responsiveness of the steering and compromise the ability of the RRT to navigate around an obstacle.

Comments on the RRT and MPC Integration

Because the RRT and MPC work more or less in opposition to each other, several methods were developed to balance their respective strengths and weaknesses. The path planning of the RRT was an important factor to consider, so a high weight was placed on the steer angle trajectory. This weight forced the MPC to track the steer signal as much as it could afford. A low weight was placed on the steer rate, meaning that the MPC would respond fluidly to changes in the steer angle. The highest weight was placed on the roll angle since the mitigation of rollover was another primary concern for this project.

The lateral distance between the roll axis and the center of mass is proportional to the roll angle of the vehicle, as can be seen in Figure 31. Due to the z-component of the forces acting on the vehicle, this distance adds to the torque about the roll axis. This roll moment generates further rotation about the roll axis which will eventually lead to rollover if left unchecked. These reductions in the peak roll angle directly translate to a reduction in the roll moment, which is a major contributing factor to rollover mitigation.

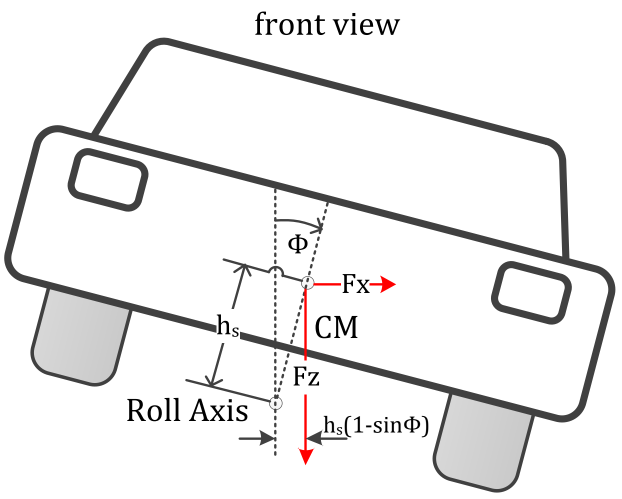


Figure 31: Torque about the roll axis due to roll

With regards to the RRT, a buffer is normally projected around the edges of the obstacle so that the RRT will not attempt to cut across the corner of the object. The RRT buffer was designed to be as tight as possible so that a minimum clearance was maintained. Additionally, it was found that the standard RRT maneuver does not necessarily induce rollover in all cases (see RRT Path Planning Maneuver in the EXPERIMENTAL RESULTS). As such, the buffer size of the object could be increased slightly without a loss in performance. The benefit of this being that the MPC would be able to cut back on the steer angle with less concern for crashing into the obstacle. This enabled the RSC to maneuver around an obstacle while still demonstrating the capabilities of the RSC.

Controller A was tested in simulation for three different steer commands: a single-lane change maneuver, a simple steer maneuver, and a sample RRT path planning maneuver. The single-lane change and simple steer tests are commonly used techniques to evaluate vehicle handling and performance in research and in industry (Falcone, Borrelli, Tseng, Asgari, & Hrovat, 2008) (Witaya, Parinya, & Krissada, 2009).

Single-Lane Change Maneuver

The single-lane change used for simulation is defined as:

1) Neutral steer for 1 second
2) A step turn to 5° for 1 second
3) A step turn to -7.5° for 1 second
4) Neutral steer

Figure 32 and Figure 34 demonstrate the result of the MPC control effort. The path followed by the vehicle is shown in Figure 33 below.

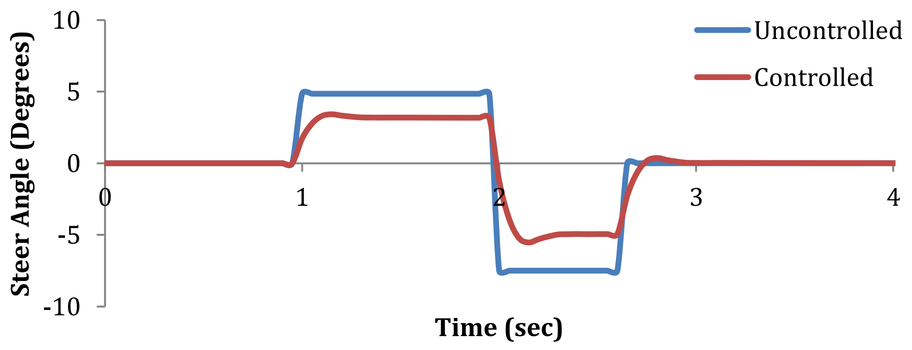


Figure 32: Steer signal and MPC signal for the double-lane change manuever

Figure 33 shows the path taken by the RSC (red) and uncontrolled (blue) single-lane change maneuver. The RSC takes a slightly less sharp turn, as shown in the previous figure. This steer trajectory results in a path that is not as wide as the uncontrolled case.

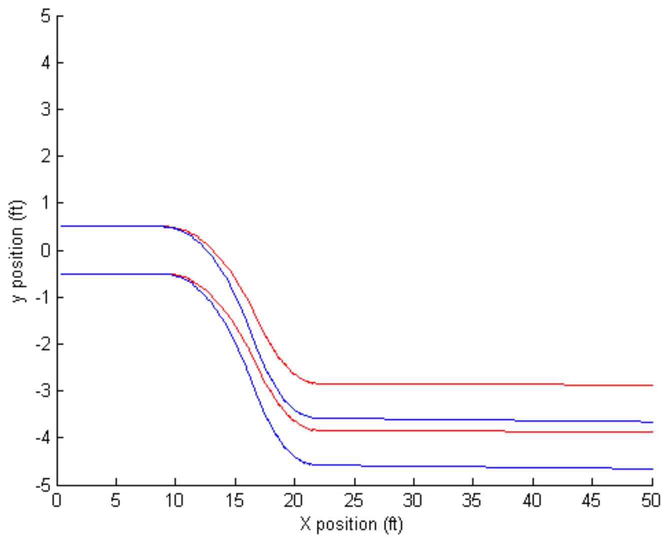


Figure 33: X-Y path for the single-lane change maneuver

As is seen in Figure 34, the roll angle overshoot is dramatically reduced in the controlled scenario. Additionally there is an average reduction in the roll angle. As explained previously,

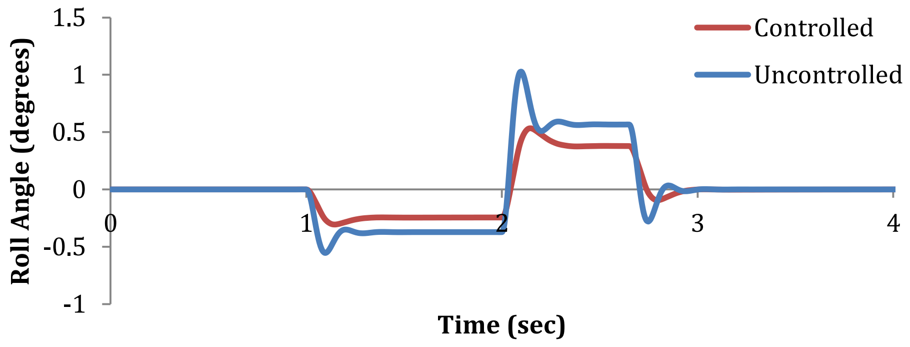


Figure 34: Roll angle response for the double-lane change manuever

Simple Steer Maneuver

The simple steer command used in simulation is defined as:

1) Neutral steer for 1 second
2) A step turn to 10° for 4 seconds

The results of the MPC control effort is seen in Figure 35 through Figure 37 below. Again it is noticed that the RRT gradually approaches the command steer angle, and that the maximum angle is throttled back slightly.

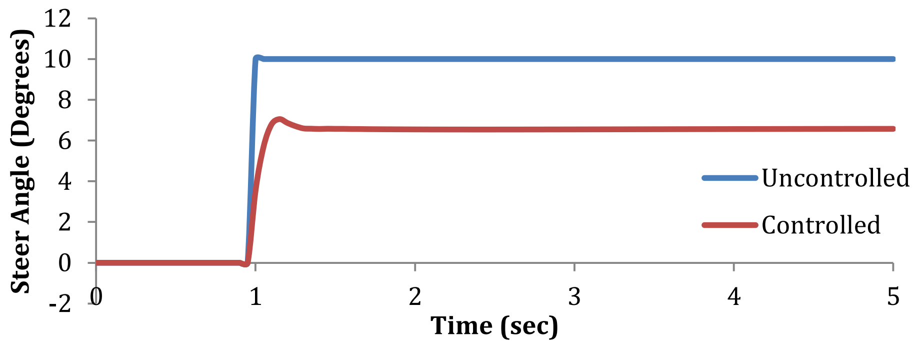


Figure 35: Steer signal and MPC signal for the simple-steer manuever

Due to the lower steady state turn angle, Figure 36 demonstrates the difference in paths taken by the controlled (red) and uncontrolled (blue) cases.

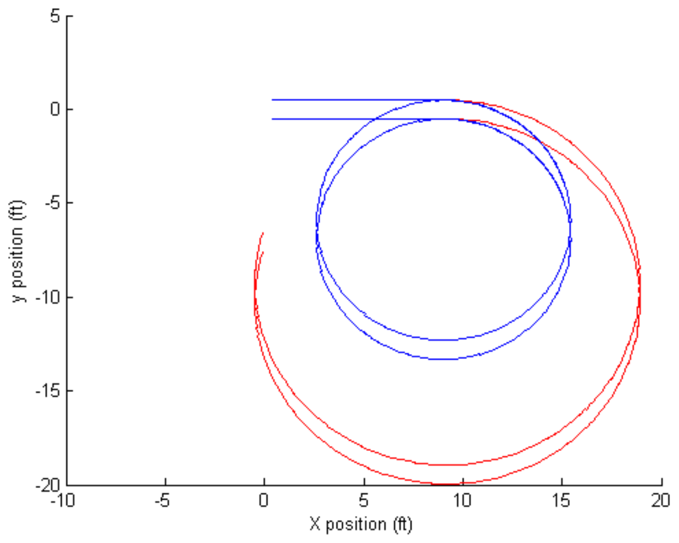


Figure 36: X-Y path for the simple turn maneuver

The overshoot of the roll angle is reduced from -1.12° to -0.62° – nearly 50%, which is a dramatic reduction. As demonstrated in the previous section, these reductions in peak roll angles contribute directly to a decrease in the roll moment, which is a major contributing factor in rollover.

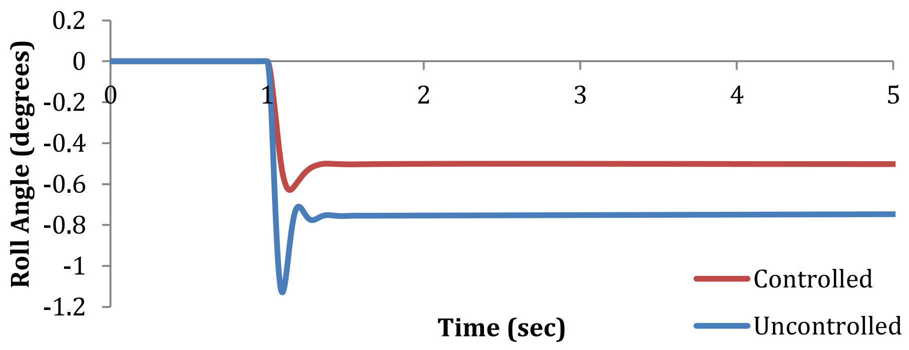


Figure 37: Roll angle response to MPC control for the double-lane change manuever

RRT Path Planning Maneuver

The final test used experimental data from an uncontrolled RRT crash avoidance maneuver. This test is unique to the project, but the results of which are directly pertinent to the effectiveness of the MPC. The RRT signal is much more dynamic than the other maneuvers, and is never the exact same command. However, it does display similar characteristics to a double-lane change since the RRT attempts to maneuver out of the way of the obstacle, and then back into its original lane.

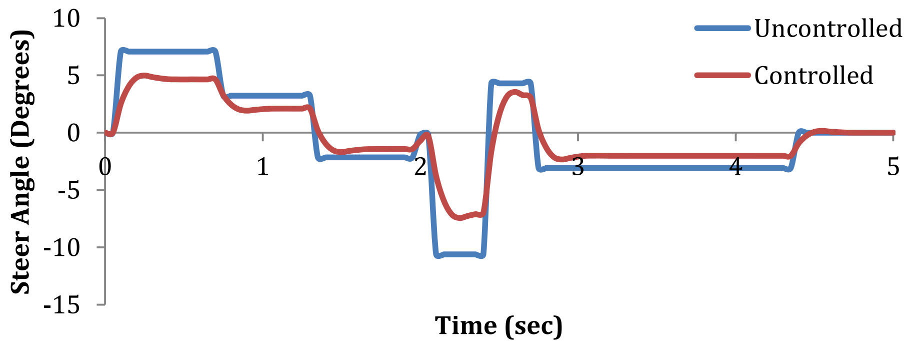


Figure 38: Steer signal and MPC signal for the RRT manuever

At each time step, the RRT calculates a turn angle that should steer the vehicle out of the way of the obstacle. The MPC in turn, minimized the turn angle with respect to the tuned model parameters. If the steer angle is too heavily constrained, the vehicle will not have the necessary trajectory to move out of the way of the obstacle. This collision would be a failure of the crash avoidance feature of the RSC. As is seen in Figure 39, the RRT only path (blue) and the MPC controlled path (red) both clear the obstacle without collision. The MPC controlled path uses less room to navigate the obstacle, which is a trade-off between stability and maneuverability. This result demonstrates that with proper tuning, a balance between these two performance criteria is achieved.

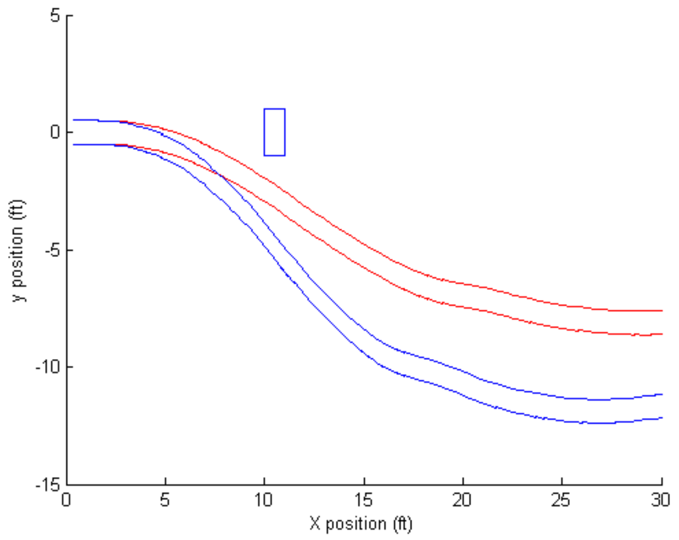


Figure 39: X-Y path of the vehicle in the RRT maneuver

Finally, the roll angle results are similar to those encountered in the previous two tests. The roll overshoot is dampened and the average roll angle magnitude is reduced.

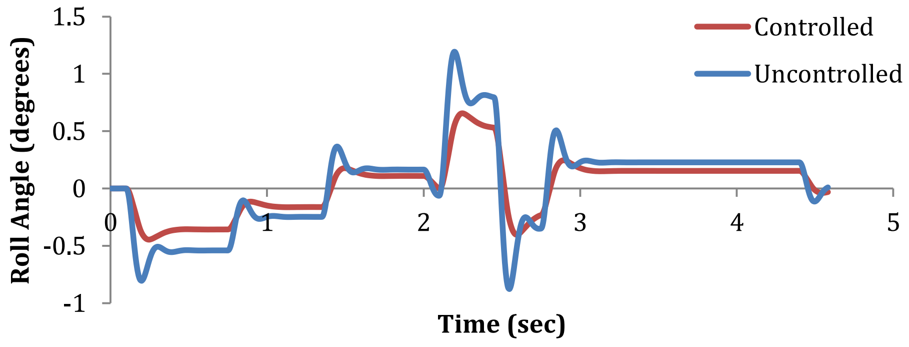


Figure 40: Roll angle response to MPC control for the RRT manuever

Comparison to PD Control

For comparison, a PD controller was designed using the roll angle for feedback. The controller, designated Controller B, has the following tunings:

$$K_p = 200$$
$$K_d = 20$$

These tuning were selected based on a criteria for roll reduction. The roll overshoot was to be reduced by at least 50%; and an average roll angle reduction of 10% was desired. These performance criteria were chosen to have a comparable roll reduction capability to the MPC.

The same experimental RRT steer signal used for the MPC test was used for the PD test. The results, shown in Figure 41 and Figure 42, demonstrate a comparable ability to minimize the roll angle. This minimization, however, comes at the expense of a much larger steer control effort. Most notably, the steer angle between 2 and 3 seconds experiences rapid fluctuations as the controller attempts to minimize the roll angle.

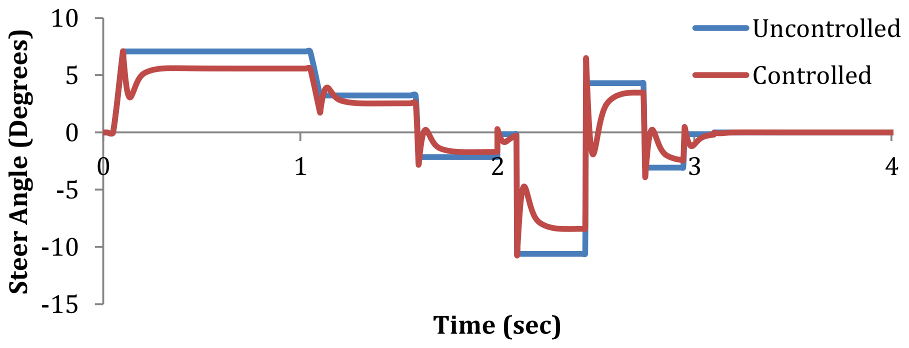


Figure 41: Steer signal and PD signal for the RRT manuever

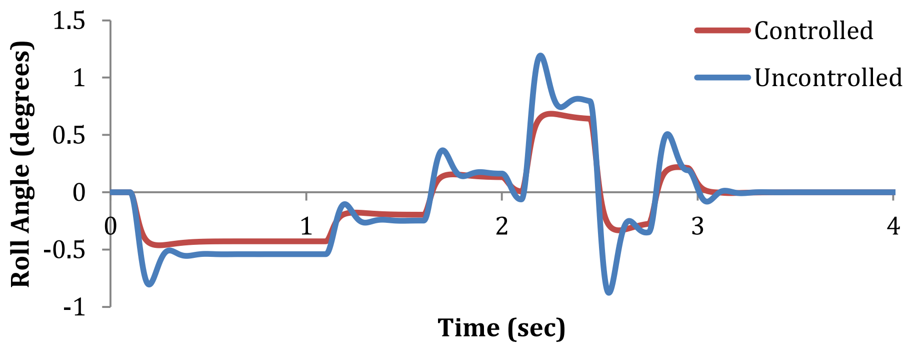


Figure 42: Roll angle response to PD control for the RRT manuever

Figure 43: Comparison of MPC and PD control efforts highlights the difference between the steer signal in Controller A (in blue) and Controller B (red). Controller A has a much smoother effort and negligible overshoot in comparison to Controller B. This limited performance of Controller B stems from the inherent limitations of PD feedback control. Unlike Controller A, Controller B has no way of predicting the future state or the future inputs. Controller A can predict what the steer signal is in future steps, and is able to approach that value *optimally*.

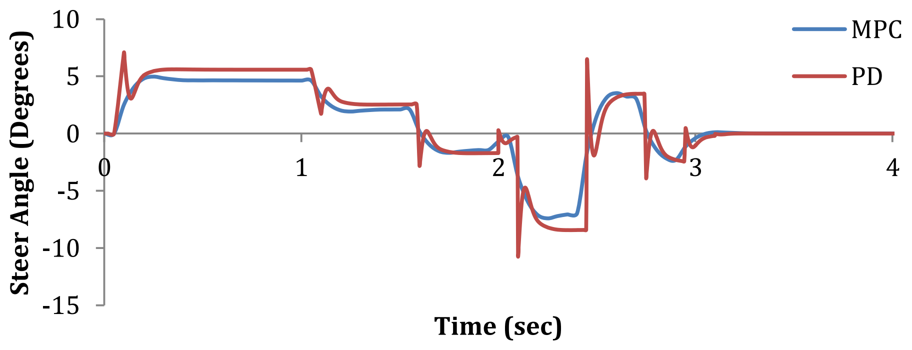


Figure 43: Comparison of MPC and PD control efforts

VIII. EXPERIMENTAL RESULTS

The roll stability controller (RSC) was tested under three different maneuvers: single-lane change (SLC), simple turn (ST), and the RRT path planner. For comparison, the vehicle was tested with the MPC turned both on and off for each maneuver.

Single-Lane Change Maneuver

The SLC maneuver is described by the following command signal:

1) Full throttle acceleration to top speed
2) Step steer angle of -15° for 0.45 seconds
3) Step steer angle of 20° for 0.95 seconds
4) Zero throttle and neutral (0°) steer angle

The uncontrolled SLC maneuver is plotted in blue in Figure 44 below. For comparison, the RSC controlled SLC is plotted in red. In the RSC case, the MPC attempts to track the blue path while maintaining vehicle stability.

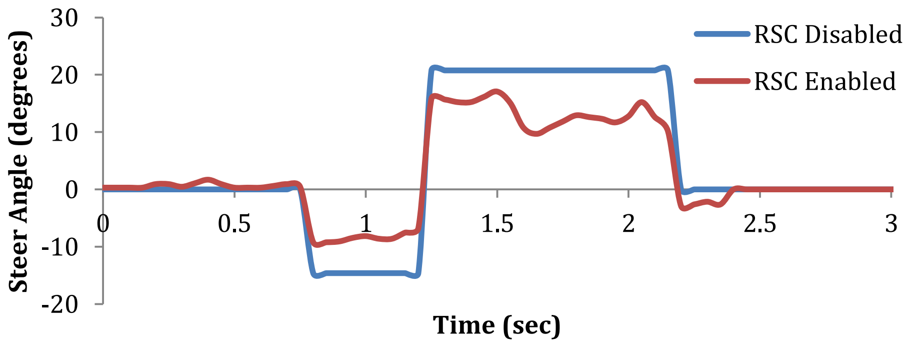


Figure 44: Steer angle command in a single-lane change maneuver

For the uncontrolled maneuver, rollover occurs just after Step 3: the shift from a full left turn to a full right turn at 1.25 seconds. The trajectories for the controlled and uncontrolled cases are plotted in Figure 45 below. The uncontrolled (blue) trajectory ends just as the vehicle makes its turn back to the right. The RSC trajectory (red) is able to complete the maneuver without issue.

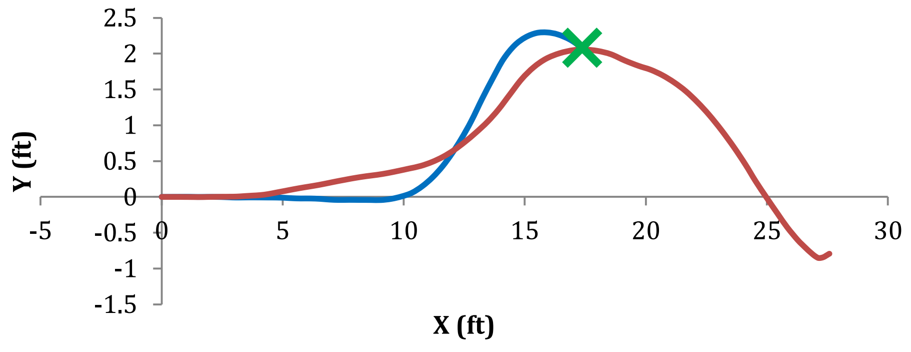


Figure 45: X-Y Trajectory of the test vehicle for the single-lane change maneuver

In Figure 46, the uncontrolled data (blue) is truncated after the roll angle reaches 90° to that the two data sets may be viewed at scale. A peak roll angle of 21° is reached before the vehicle makes a rapid turn to the right. As the vehicle enters the left turn, the weight is shifted to the right. When the vehicle makes the rapid turn to the right, the center of mass (CM) is thrown to the left by the lateral acceleration. The lateral tire forces react to the ground in the opposite direction of the CM, which generates a torque about the roll axis. It is this torque that causes massive roll acceleration, and thus leads to rollover.

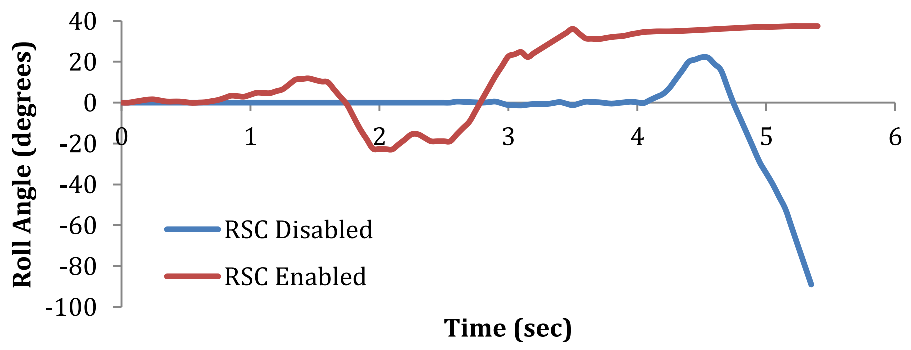


Figure 46: Roll angle response in a double-lane change manuever

Simple Turn Maneuver

The Simple Turn (ST) maneuver is described by the following command signal:

1) Full throttle acceleration to top speed

2) Step steer angle of 20° for 1.0 seconds

3) Zero throttle and neutral (0°) steer angle

The ST maneuver is plotted in Figure 47 with the uncontrolled signal (blue) and the RSC signal (red) for comparison.

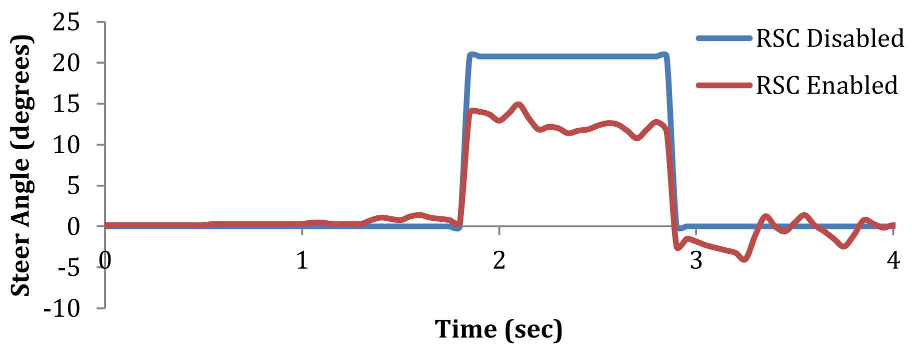


Figure 47: Steer angle command in a simple turn maneuver

As seen in Figure 48, rollover is seen to occur in the uncontrolled case (blue) before the vehicle even has a chance to enter the turn. Meanwhile the RSC maneuver is able to navigate the turn without incident.

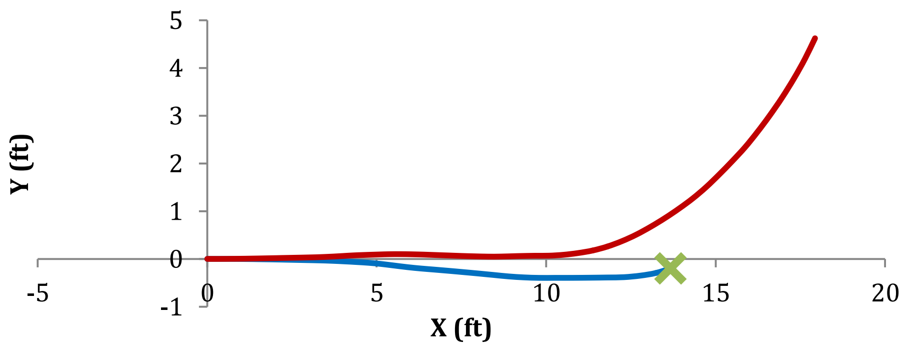


Figure 48: X-Y trajectory of the test vehicle for the simple-turn maneuver

The rollover can also be seen in Figure 49, marked by a sharp increase in the roll angle occurring at 2.75 seconds in the uncontrolled test (blue). Comparatively, the RSC maneuver pulls back on the steer angle just as the roll angle begins to grow.

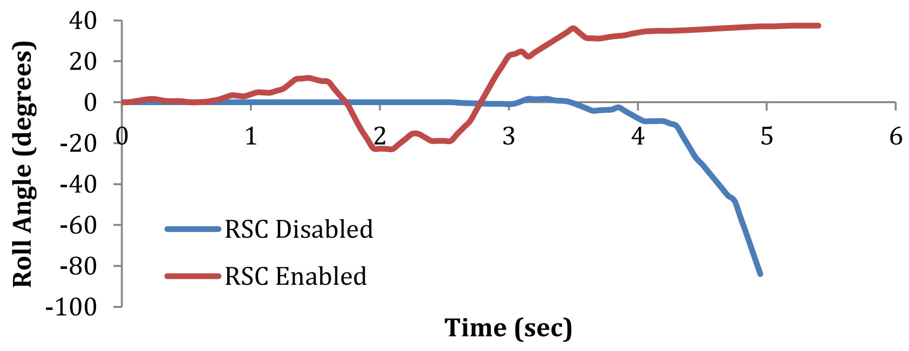


Figure 49: Roll angle response in a simple turn manuever

RRT Path Planning Maneuver

The final experimental test was a validation of the RSC on the RRT path planning algorithm. This steer command is plotted in blue in Figure 50 below. For the RSC enabled run, the same RRT maneuver was sent through the RSC, which then fed its updated signal (in red) to the vehicle. Since the RRT maneuver has a unique steer trajectory every time it runs, a successful RRT maneuver was recorded without the RSC enabled.

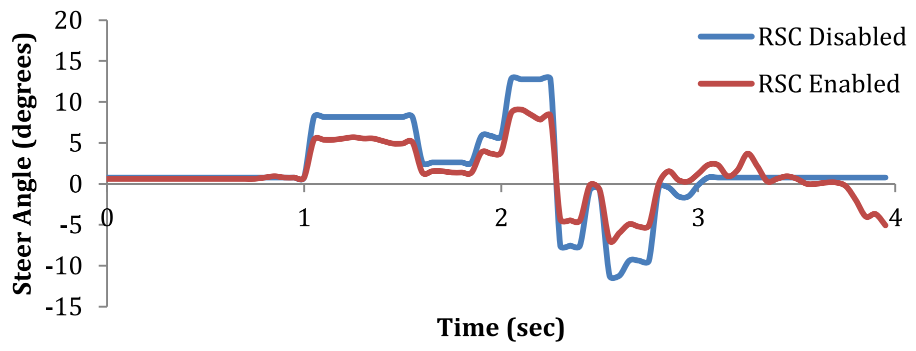


Figure 50: Steer angle command in an RRT path planning maneuver

Figure 51 demonstrates the different paths taken by the RSC maneuver (red) and the uncontrolled RRT maneuver (blue). The vehicle clears the track obstacle in both cases, which is a primary concern for the crash avoidance aspect of this project. The path taken by the RSC is a more constrained trajectory, which is a common result seen in both simulation and experiment for the RSC.

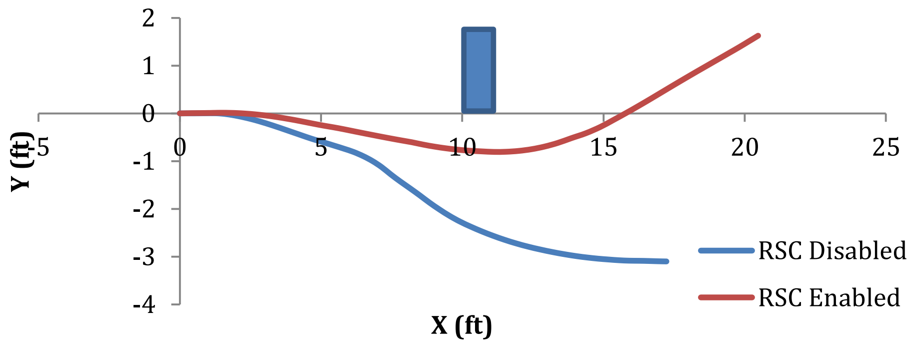


Figure 51: X-Y trajectory of the test vehicle in an RRT maneuver

Figure 52 compares the roll angle between the uncontrolled RRT and the RSC maneuver. Again, the average magnitude of the roll angle is smaller in the RSC, which helps to maintain vehicle stability.

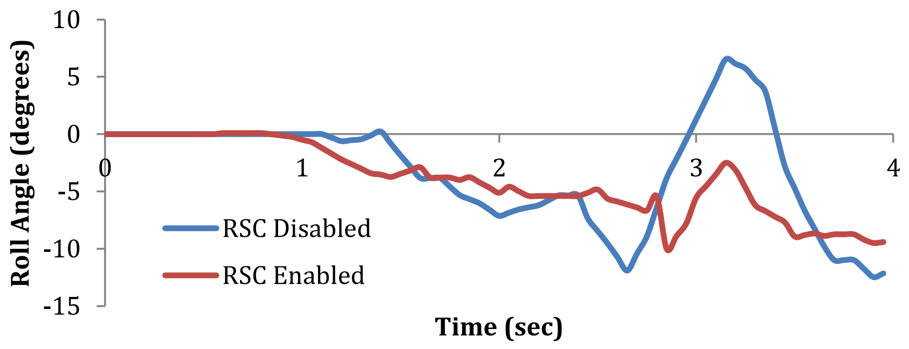


Figure 52: Roll angle response in an RRT path planning manuever

Notably between 1 and 1.5 seconds, the RSC maneuver has a slightly *higher* roll angle than the uncontrolled maneuver. This higher roll angle is a result of the weighting scheme used for the controller. Recall that the roll angle is not the only factor considered: roll rate and yaw rate are weighted as well. Since the roll angle is the integral of the roll rate, the response to its value lags behind the roll angle.

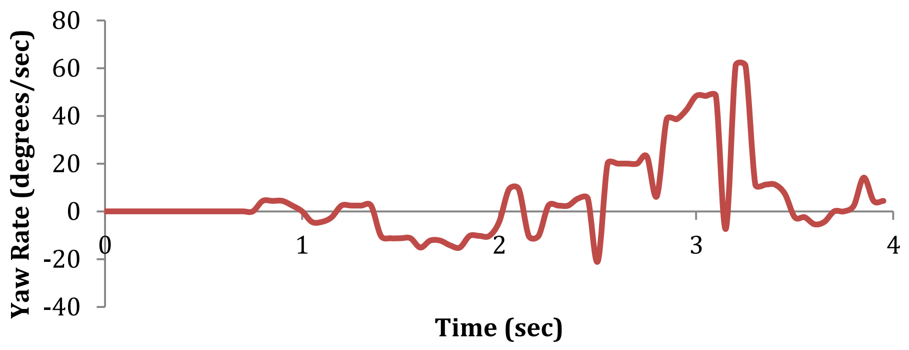


Figure 53: Yaw rate response in an RSC RRT path planning manuever

As is seen in Figure 52 above, there is a small yaw rate beginning at 0.75 seconds. The vehicle was traveling in straight line prior to 1 second; so no yaw rate should be present. This result is most likely due to IMU drift. Further details on IMU drift and how it was handled in this project are explored in the CONCLUSIONS chapter of this report.

IX. CONCLUSIONS

In this paper, a roll stability controller (RSC) was presented based on an eight degree of freedom dynamic vehicle model. The controller was designed for and tested on a scaled vehicle performing obstacle avoidance maneuvers on a populated test track. A rapidly-exploring random tree (RRT) algorithm was used for the vehicle to execute a trajectory around a static obstacle, and examines the geographic, non-homonymic, and dynamic constraints to maneuver around the obstacle. A model predictive controller (MPC) was used to generate an optimal trajectory around the obstacle though the mitigation of rollover and spin-out. Experimental results demonstrated a markedly improved stability and maneuverability for double-lane changes, simple turns, and RRT path planning at speeds up to 10 ft/s and turn angles of up to 20° on a 1/10th scale vehicle.

Many of the components of this project have been successfully tested in full scale experiments. MPC has already been verified in full scale lane change maneuvers (Falcone, Borrelli, Tseng, Asgari, & Hrovat, 2008). The RRT has also been used in the DARPA Grand Challenge, and the DVM has been verified against vehicle data from a Ford Taurus. These tests, along with the results of this paper, demonstrate the possibility of full scale testing. And as mentioned in the introduction, crash avoidance and roll stability have the potential to save thousands of lives, and prevent millions of vehicle collisions.

X. RECOMMENDATIONS

Hardware Limitations and Recommendations

In commercial electronic stability control (ESC), many production vehicles have independent braking for each wheel. Through individual adjustment of the brakes, a corrective yaw torque is generated on the vehicle. This torque is the primary yaw stabilizer for all commercial ESC. The vehicle used in this experiment has no independent braking, and only throttle and engine torque were available for control stabilization. The goal of this project was to develop a stability controller to work with the RRT despite these limitations. The inherent limitations of this setup make it impossible to employ state feedback control. Additionally, the lack of a steer-angle sensor meant that MPC had to use the applied steer angle from the RRT instead of having the true value of the steer angle from sensor measurements. This undoubtedly affected the performance due to the lag between the applied steer angle signal and the real-world steer angle. However, this work has laid the foundation for further experiments, including the possibility of RSC using four-wheel braking inputs.

It is recommended that for future experiments with the RTC, four-wheeled braking is installed on the vehicle. This could be accomplished by the attachment of small solenoids on the wheel hubs. These solenoids could be activated by the on-board MCU, and could be powered by the on-board battery. Ideally, these could provide a counter-torque to the yaw moment, which would help to provide stability. More importantly, it would provide state control without modifying the steer trajectory. This would help tremendously in terms of crash avoidance since the avoidance algorithm is requisite on the vehicle following the planned path.

The implementation of RSC using four-wheeled braking is not particularly difficult from a software standpoint. Each brake would induce a torque on their respective wheel, thereby adjusting the individual wheel velocities $v_{l_{\star,\circ}}$. This term appears in the function for the slip angle:

$$\alpha_{\star,\circ} = \tan^{-1}\left(\frac{v_{l_{\star,\circ}}}{v_{c_{\star,\circ}}}\right)$$

Currently, the plant model is linearized about the vehicle state and the steer input only. However, with four wheeled braking, the plant would then be linearized about the four lateral wheel velocities. This is the method successfully employed by P. Falcone and F. Borelli (Falcone, Borrelli, Tseng, Asgari, & Hrovat, Linear Time Varying Model Predictive Control and its Application to Active Steering Systems: Stability Analysis and Experimental Validation, 2008). The MATLAB MPC is multi-in/multi-out controller, so it would be able to handle a five-input system. It would be up to a future team to determine the relation between the braking force and the wheel velocity, however for small corrections to the wheel speeds, a constant proportionality should feasibly exist.

The inertial measurement unit (IMU) used was prone to drift and steady state offsets. The effect of this was minimized with calibration; however, the effects could not be completely mitigated through signal processing. This resulted in occasional errors in the yaw/roll rate which would compromise the performance of the RRT and MPC. For future work with the test vehicle, it is recommended that the IMU be replaced with a better performing model, and that it is used in conjunction with GPS to determine the vehicle state. This is a commonly used practice for vehicle tracking/positioning where a GPS provides accurate global positioning of the vehicle and the IMU provides relative accelerations (Farrell, 1998). The GPS can also be used to reset the IMU drift, so this combination would provide a more robust solution to vehicle positioning.

Two parameters needed for the vehicle model were unable to be determined empirically due to limitations in time, funding, and equipment. These included the front/rear shock damping coefficients and the cornering coefficient. A method to determine the damping coefficients could be accomplished using a shake table and a sensitive accelerometer.

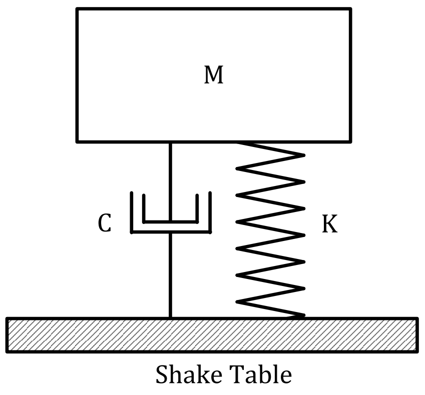


Figure 54: Example of an experimental apparatus to determine the damping coefficient.

By performing a sine-sweep test of a shock fixed to the shake table (with an appropriate mass to provide inertial resistance), the damping ratio could be determined. The coefficient of damping could then be found by using the known damping ratio, spring rate, and attached mass.

In order to use the Pacejka model, an experiment which directly relates the lateral force as a function of slip $F_y = f(\alpha)$ is needed. Such devices are generally large, expensive, have limited access, and are meant for full sized tires. However, a lateral force tire tester (LFTT) designed for scale models has been developed and verified (Witaya, Parinya, & Krissada, 2009). The experiment uses a large rotating drum to simulate the road conditions. A tire is mounted on top of and in contact with the drum. As the drum spins, the tire angle is adjusted to different positions, generating a slip angle. A torque motor/sensor keeps the tire in place, and also measures the torque required to do so. This torque is generated by the lateral force on the tire, so in this way the lateral force as a function of slip is obtained. A modified version which replaces the torque sensor with a spring was designed for this project. This design is a cost-effective and easy to manufacture, so future work with this project should include the design and implementation of the LFTT.

Software Limitations and Recommendations

The MATLAB/Simulink environment was a highly effective and versatile platform to implement the RSC. The Model Predictive Toolbox used was a highly tunable and helpful program that eliminated the need to write a solver for the controller. It is possible to apply multiple controllers which would each operate around particular set-points of the vehicle state. However, the QP solver is limited to only linear time invariant models. This means that pre-linearizations of the plant model must be done for each controller. However, a solver which could make use of linear time varying models would be able to update the plant at every iteration, thereby providing a more accurate representation of the vehicle, and accordingly more precise control.

Solvers such as NPSOL (Falcone, Borrelli, Tseng, Asgari, & Hrovat, MPC-based Approach to Active Steering for Autonomous Vehicle Systems, 2005) have been used successfully to handle the optimization routine at low speeds and larger control intervals. It must be stressed, however, that the current computational complexity of such solvers makes it infeasible to run at real time with enough control intervals to successfully maneuver at high speeds. Until either a faster solver is developed or until faster computers are available, an LTI or LTV approximation must be used.

BIBLIOGRAPHY

Bemporad, A., Morari, M., & N., R. (2012). *Model Predictive Control Toolbox™ Users Guide.* Natick, MA: The MathWorks, Inc. Retrieved from Mathworks web site.

Brogan, W. (1990). *Modern Control Theory, 3rd Ed.* Englewood Cliffs, NJ: Prentice-Hall, Inc.

Casanova, D. (2000). *On Minimum Time Vehicle Manoeuvring: The Theoretical Optimal Lap.*

Colwell, C. (2011, 11 17). *Resource Lesson: Center of Mass*. Retrieved from Physics Lab: http://dev.physicslab.org/Document.aspx?doctype=3&filename=RotaryMotion_CenterMass.xml

Craig, R., & Kurdila, A. (2006). *Fundamentals of Structural Dynamics, 2nd Ed.* Hoboken, NJ: John Wiley & Sons, Inc.

Dassault Systèmes. (2012, 9 14). *SolidWorks Mass Properties and Section Properties.* Retrieved from SolidWorks Help: http://help.solidworks.com/2012/English/SolidWorks/sldworks/HIDD_MASSPROPERTY_TEXT_DLG.htm

Demerly, J. (2000). *Emergency Braking Using Two Independent Steering Actuators While Maintaining Directional Control.* Massachusetts Institute of Technology.

Demerly, J., & Youcef-Toumi, K. (2000). Non-linear Analysis of Vehicle Dynamics (NAVDyn): A Reduced Order Model for Vehicle Handling Analysis. *Proceedings of the Automotive Dynamics & Stability Conference* (pp. 354-370). Troy, MI: Society of Automotive Engineers.

Dixon, J. C. (1996). *Tire, Suspension and Handling.* Warrendale: Society of Automotive Engineers.

Falcone, P., Borrelli, F., Tseng, H. E., Asgari, J., & Hrovat, D. (2005). MPC-based Approach to Active Steering for Autonomous Vehicle Systems. *International Journal Vehicle Autonomous Systems*, 265-291.

Falcone, P., Borrelli, F., Tseng, H. E., Asgari, J., & Hrovat, D. (2008). Linear Time Varying Model Predictive Control and its Application to Active Steering Systems: Stability Analysis and Experimental Validation. *International Journal of Robust and Nonlinear Control*, 862-875.

Farmer, C. (2008). *Crash Avoidance Potential of Five Vehicle Technologies.* Arlington, VA: Insurance Institute for Highway Safety.

Farmer, C. (2010). *Effects of Electronic Stability Control on a Fatal Crash Risk.* Arlington, VA: Insurance Institute for Highway Safety.

Farrell, J. (1998). *The Global Positioning System & Inertial Navigation.* New York, NY: McGraw-Hill Professional.

Gillespie, T. (1992). *Fundamentals of Vehicle Dynamics.* Warrendale, PA: Society of Automotive Engineers, Inc.

Highway Loss Data Institute. (2011, July 19). *High-tech system on Volvos is preventing crashes*. Retrieved May 10, 2012, from Insurance Institute for Highway Safety: http://www.iihs.org/news/rss/pr071911.html

Jazar, R. N. (2008). *Vehicle Dynamics: Theory and Application.* New York: Springer.

Kuwata, Y., Teo, J., Fiore, G., Karaman, S., Frazzoli, E., & How, J. (Aug. 2008). Motion Planning in Complex Environments using Closed-Loop Prediction. *Proceedings of the AIAA Guidance, Navigation, and Control Conference and Exhibit* (pp. AIAA-2008-7166). Honolulu, HI: AIAA.

Kuwata, Y., Teo, J., Fiore, G., Karaman, S., Frazzoli, E., & How, J. (Sept. 2008). Motion Planning for Urban Driving using RRT. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, (pp. 1681-1686). Nice, France.

Kuwata, Y., Teo, J., Fiore, G., Karaman, S., Frazzoli, E., & How, J. (Sept. 2009). Real-Time Motion Planning With Applications to Autonomous Urban Driving. *IEEE Transactions on Control Systems Technology, Vol. 17, no. 5*, 1105-1118.

Matsubayashi et al. (2006). *Development of Rear Pre-Crash Safety System For Rear-End Collisions.* Toyota Motor Corporation.

Mechanical Simulation Corperation. (2000). *CarSim Educational Users Manual v4.5.* Ann Arbor, MI: Mechanical Simulation Corperation.

Melanson, D. (2011, July 22). *Toyota's new crash-avoidance technology takes control of the wheel*. Retrieved July 22, 2011, from Engadget: http://www.engadget.com/2011/07/22/toyotas-new-crash-avoidance-technology-takes-control-of-the-whe/

Meriam, J. L., & Kraige, L. G. (1987). *Engineerig Mechanics: Dynamics.* New York: John Wiley and Sons.

Milliken, W., & Milliken, D. (1995). *Race Car Vehicle Dynamics.* Warrendale, PA: Soceity of Automotive Engineers, Inc.

Motor Trend. (2009, November). *First Drive: 2011 Infiniti M Prototype*. Retrieved May 7, 2010, from http://www.motortrend.com/roadtests/sedans/112_1001_2011_infiniti_m_prototype_drive/driving_impressions.html

National Highway Traffic Safety Administration. (2007). *Federal Motor Vehicle Safety Standards; Electronic Stability Control Systems; Controls and Displays - Final Ruling.* Washington, D.C.: U.S. Department of Transportation.

National Highway Traffic Safety Administration. (2009). *Traffic Safety Facts.* Washington, D.C.: U.S. Deptartment of Transportation.

National Highway Traffic Safety Administration. (2009). *Traffic Safety Facts: California.* Washington, D.C.: U.S. Department of Transportation.

Pacejka, H. (2002). *Tire and Vehicle Dynamics.* Warrendale, PA: Soceity of Automotive Engineers, Inc.

Reimpell, J., Stoll, H., & Betzler, J. W. (2001). *The Automotive Chassis: Engineering Principles.* Warrendale: Elsevier.

Schmidt, C., & L.T., B. (1994). Quadratic Programming Methods for Reduced Hessian SQP. *Computers & Chemical Engineering Vol. 18, Number 9*, 817-832.

Toyota. (2006, August 25). *Toyota Strengthens Efforts to Develop Safe Vehicles*. Retrieved May 6, 2010, from http://www.toyota.co.jp/en/news/06/0825.html

Wiener, K., & Boynton, R. (1992). Using the "Moment of Inertia Method" to Determine Product of Inertia. *51st Annual Conference of the Society of Allied Weight Engineers.* Hartfor, CT: S.A.W.E., Inc.

Witaya, W., Parinya, W., & Krissada, C. (2009). Scaled Vehicle for Interactive Dynamic Simulation. *IEEE International Conference on Robotics and Biometrics.* Bangkok, Thailand.

6.2.0 UART Communication

The communication method used between the main computer and both the Xiphos (ultrasonic array) and the R/C truck's onboard microcontroller was Universal Asynchronous Receiver/Transmitter (UART). Data transmitted through UART uses two separate lines to communicate, one for transmitting (Tx) data and one for receiving (Rx) data.

6.2.1 UART Basics

UART transmits one byte (8 bits or 1's and 0's) at a time. These bits get stored into a transmit or receive register until the register is full.

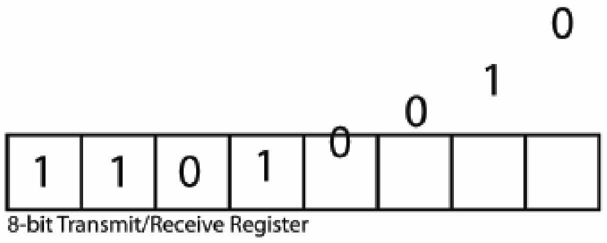


Figure 16. Data filling an 8-bit Transmit/Receive Register.

Once the register is full, a flag bit is set that alerts the main program that a full packet of data has been received or a full packet of data is ready to be sent. The purpose of waiting for a full register is to keep the data in discrete, recognizable chucks so it may be pieced back together predictably with little or no data corruption. In addition, because transmitting a full byte can take significant processing time, the flag ensure new data does not overwrite the register before the previous 8 bits are sent.

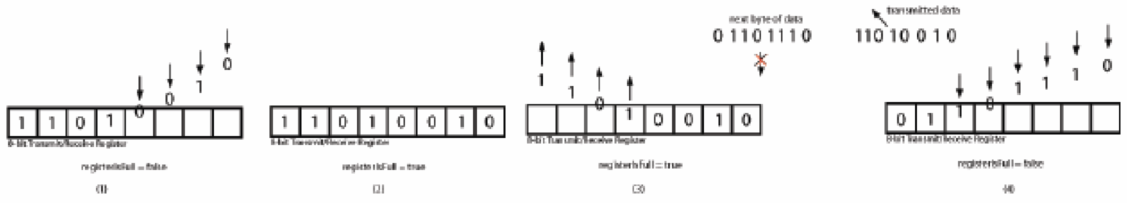


Figure 17. Data flow through UART transmit/receive registers.

Figure 17 presents the data basic data flow through a UART transmit/receive register. First a byte of information enters the register one bit at a time (Figure 17.1). The register is monitored to ensure data does not overflow past

28

the 8 bits. Once the register is full, a flag (e.g. registerIsFull = true) is set alerting the main routine that a complete packet of data is ready to be sent or received (Figure 17.2). As the data from the register is being sent or received, the next byte cannot enter the register until it has been cleared and the "registerIsFull" flag has been unset (Figure 17.3). Once the register has been cleared of all previous bits, the byte of information can be captured (Figure 17.4).

6.2.2 UART Framing

In addition to the data bytes, UART uses addition bites to frame the data into a package. There are occasions where data being sent consists of less than 8 bits. Because data can be less than 8 bits it is necessary to indicate when a set of bits begins and ends. UART uses a start and stop bit to wrap the data bits.

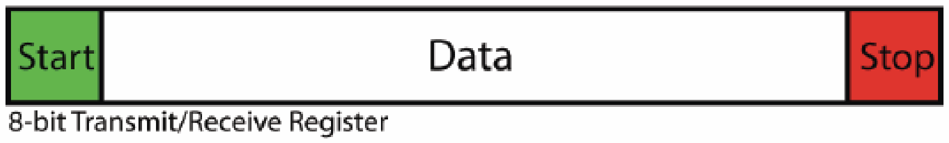


Figure 18. Data framed with a start and stop bit.

The start bit precedes the data bits and alerts the beginning of a new data package. Following the data is the stop bit that triggers the setting of the register is full flag.

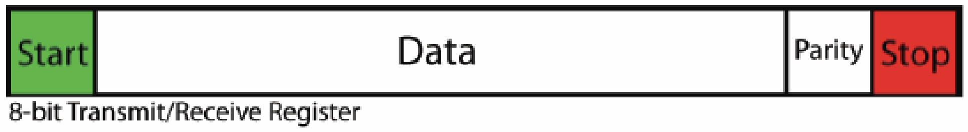
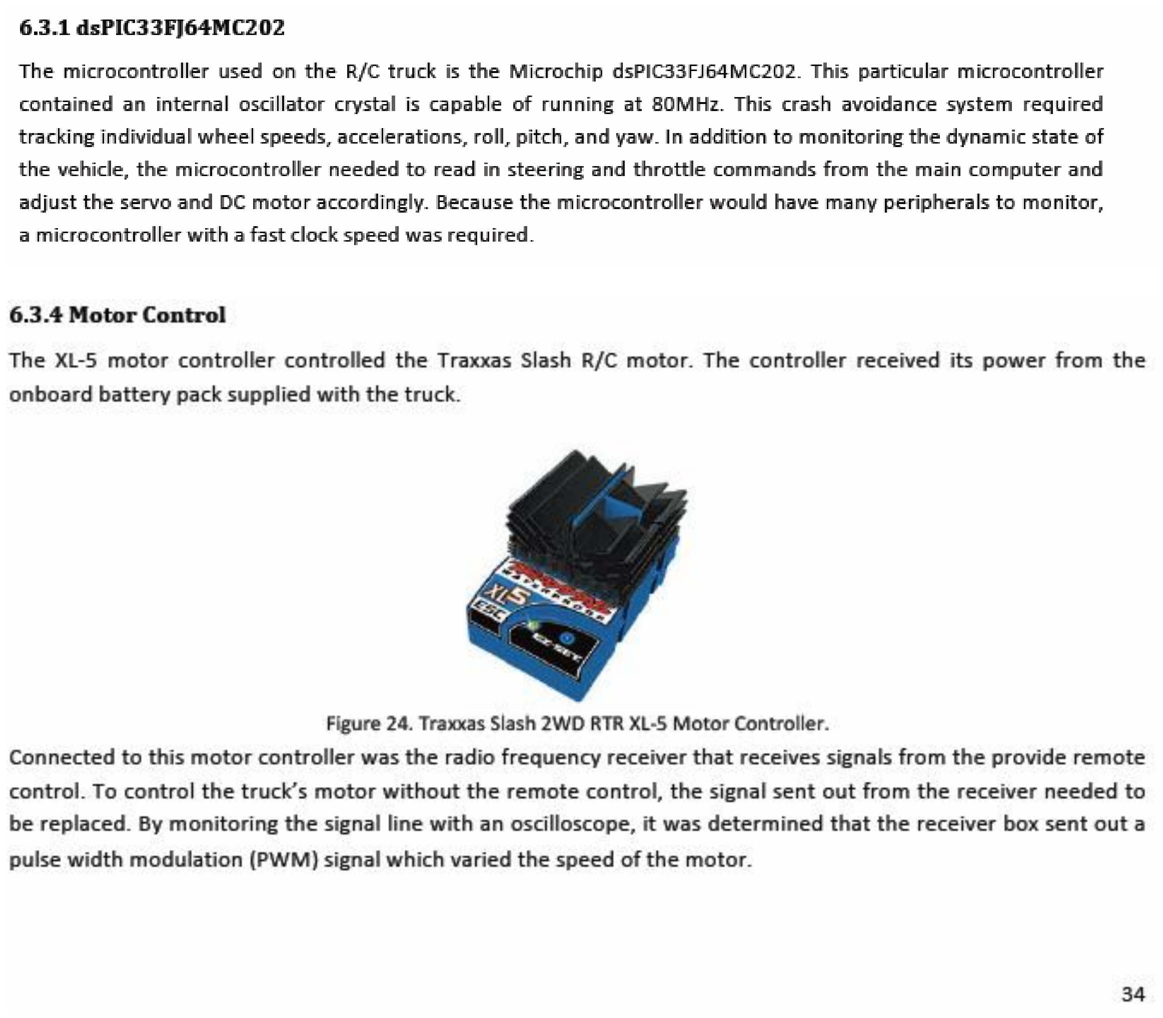


Figure 19. Parity bit set in a data package.

In addition to the start and stop bits is the parity bit. The parity bit acts as a data corruption check. Parity checks the number of bits that are set to one within the data. Even parity sets itself to one if the data contains an odd number of bits that are set to one resulting in an even number of ones in the package. If the data is then received and an odd number of ones are read, the data is considered corrupt. Similarly, odd parity keeps an odd number of ones in each data package and can be considered corrupt if once transmitted has an even number of ones.

6.3.0 R/C Truck Onboard Microcontroller

The R/C truck used an onboard microcontroller to actuate the steering servo and driving DC motor. The microcontroller read in steering and throttle control instructions sent by the central processing computer and adjusted the vehicle's state accordingly. The onboard microcontroller provided more control than the remote control packaged with the truck.



6.3.1 dsPIC33FJ64MC202

The microcontroller used on the R/C truck is the Microchip dsPIC33FJ64MC202. This particular microcontroller contained an internal oscillator crystal is capable of running at 80MHz. This crash avoidance system required tracking individual wheel speeds, accelerations, roll, pitch, and yaw. In addition to monitoring the dynamic state of the vehicle, the microcontroller needed to read in steering and throttle commands from the main computer and adjust the servo and DC motor accordingly. Because the microcontroller would have many peripherals to monitor, a microcontroller with a fast clock speed was required.

6.3.4 Motor Control

The XL-5 motor controller controlled the Traxxas Slash R/C motor. The controller received its power from the onboard battery pack supplied with the truck.

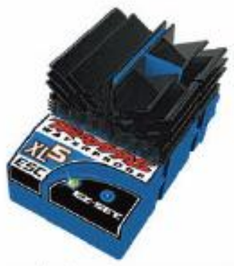


Figure 24. Traxxas Slash 2WD RTR XL-5 Motor Controller.

Connected to this motor controller was the radio frequency receiver that receives signals from the provide remote control. To control the truck's motor without the remote control, the signal sent out from the receiver needed to be replaced. By monitoring the signal line with an oscilloscope, it was determined that the receiver box sent out a pulse width modulation (PWM) signal which varied the speed of the motor.

34

Pulse width modulation is a common technique to vary the speed and torque of a motor without requiring an excess amount of current. The principle behind PWM is the idea that a signal or source can be pulsed on and off at different rates to simulate a range of voltages that cause the motor to rotate at different speeds. The pulse durations are based off a percentage of duty cycle where one hundred percent corresponds to the signal being high one hundred percent of the time. A signal that has a ten percent duty cycle with a frequency of one hundred hertz will be on for 0.001 seconds (10% of one cycle) and off for 0.009 seconds (90% of one cycle).

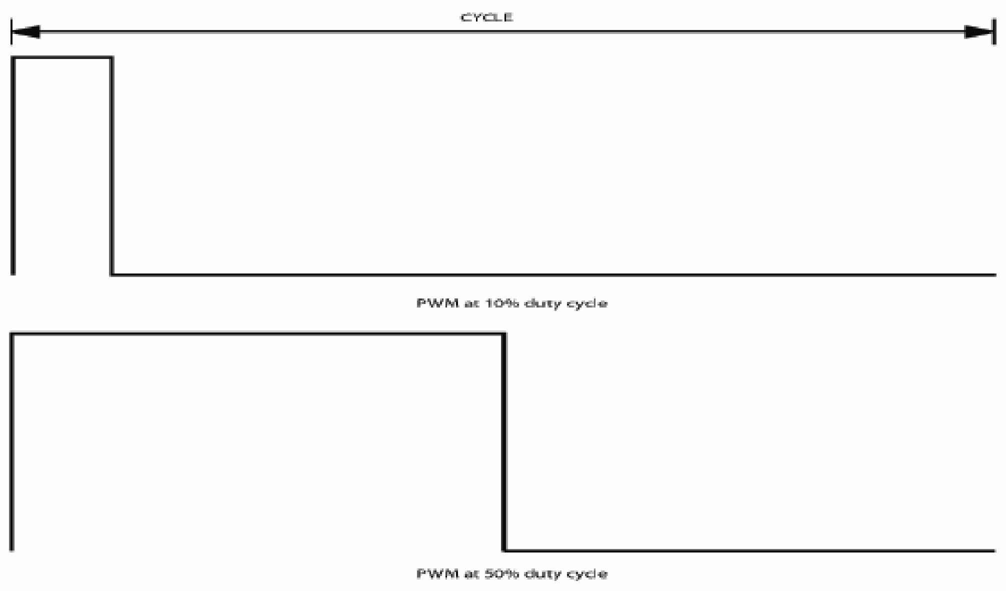


Figure 25. Pulse Width Modulation signals as a percentage of duty cycle.

For the Traxxas Slash R/C truck, the motor PWM ran at a frequency of 100Hz between 10% and 20% duty cycle. To achieve the 100Hz frequency of the PWM signal, the modules on the dsPIC33F needed to be configured. The PWM modules allow a user to define the period of the pulse width signal using the PTPER register 15 bit register which has a maximum value of 32767. Unfortunately it was not directly apparent how to associate this register number with the frequency of the signal. Therefore, the period number was determined experimentally using an oscilloscope. The period number that was entered in the register PTPER was 6249 (100Hz). In addition, the range of PWM duty cycle needed to be determined to correspond to full throttle, no throttle, and negative throttle. This duty cycle correlation was also tested experimentally using an oscilloscope. The R/C truck was hooked up to receive signals from the remote control. The oscilloscope was tapped into the R/C signal and monitored as the throttle was increased and decreased. The range of PWM duty cycles were 10% for maximum reversed speed, 15% for no throttle and 20% for full maximum forward speed.

A safety feature of the R/C truck is its neutral throttle protection. This neutral throttle protection ensures that the motor controller does not turn on and cause the vehicle to surge forward unless it is receiving the neutral (no throttle) signal. The PWM value needed to be specifically set to the neutral throttle protection value because calculations from throttle percentage to PWM were not accurate enough. The neutral throttle protection PWM value was determined to be 1885 which corresponded to 15.1% duty cycle. The protection is only required for the motor PWM because it was connected to the motor controller that supplies power to both the motor and the steering servo.

File: main.c

```
setMotorPWM(1885);
setServoPWM( 1775 );
```

File: motor.c

```
void configMotorPWM( void ) {
     PWM1CON1bits.PEN2H = 1;
     P1TCONbits.PTCKPS = 0b11;
     P1TPERbits.PTPER = 6249;
}

. . .

void setMotorPWM( unsigned int duty ) {
     P1DC2 = duty;
}
```

The setMotorPWM function set the duty cycle register direction that ensures the neutral throttle protection is set correctly.

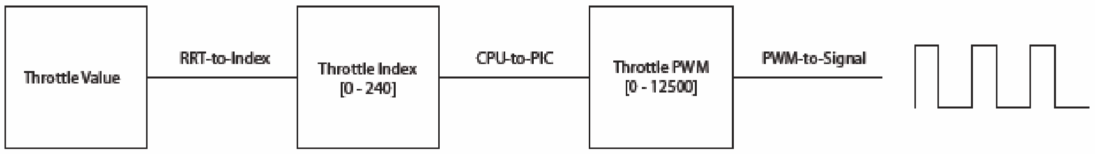


Figure 26. Progress of throttle control value from RRT to motor PWM.

To communication the throttle control between the onboard microcontroller and the main central processor, an index value was used. The path planner from the main computer produces a throttle value. Then, the throttle value is converted into an all-positive index from 0 to 240 and transmitted to the onboard microcontroller. The onboard microcontroller then converts the index into a PWM register value and sets the pulse accordingly.

6.3.5 Servo Control

The servo worked in much the same way as the motor control. However, the servo control did not require the PWM to be set to a neutral value. For practical purposes though, zeroing the steering servo to a zero degree angle was a good idea.

The range of PWM duty cycles was determined by using the same method as that of the motor. Hooking the signal up to an oscilloscope, it was determined to have a maximum steering angle left of 20 degrees at a duty cycle of 10% and a steering angle right of 20 degrees at a duty cycle of 18%. This left the center 0 degree angle at 14%. However, to get an accurate 0 angle, fine-tuning was required much like the neutral throttle protection value was determined. The 0 angle value was determined to be 1775.

File: servo.c

```
void setServoPWM( unsigned int duty ) {
     P1DC1 = duty;
}
```

The servo drivers have similar functions like the motor drivers. The servo PWM can be set direction with integer values within range of the 10 to 18% duty cycle. The setServoPWM function stores the value passed in directly to the servo PWM controller’s register.

In addition, the steering control followed the same data progression as the throttle control from the RRT to an index value to a PWM index value to a signal. It was important to note that sending a PWM signal with a duty cycle greater than or less than the max or min value of the required duty cycle the servo produced a high-pitched tone. Be aware that this high-pitched tone should be avoided by adding a buffer on each side of the duty cycle range to ensure safe servo operation.

Connecting the dsPIC33F microcontroller to the servo required determining which wire was power, ground, and signal.

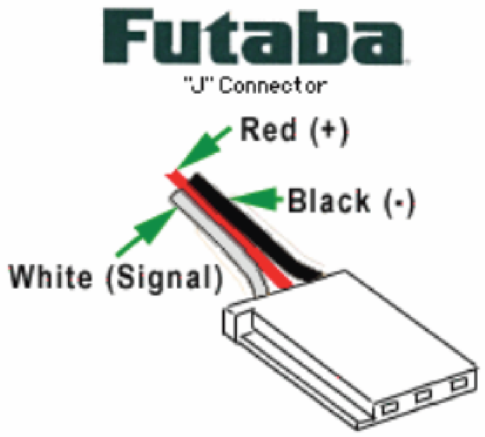


Figure 27. Generic hobby servo connector wire scheme.

The wiring color scheme for the steering servo and motor controller were the same as the Futaba “J” connector in figure 25. Based on the color coded wires the steering servo and motor controller were connected to the microcontroller’s PWM motor driver pins with a common ground to ensure a solid signal.

One of the problems that occurred was a noisy signal when the motor was running. For future iterations of the system, the common ground should be removed and a relay should be used with a separate signal source to keep the microcontroller and motor controller on separate power sources.

6.3.6 IMU

The IMU selected was a six degree of freedom Atomic IMU (XBee ready) with a dedicated onboard ATMega 168TM microcontroller to monitor the three accelerometers and three gyroscopes. Communication with the IMU can be done through a UART connection at a baudrate of 115200. The unit runs off the same 3.3V power source that the dsPIC33F uses. This reduces the amount of power supplies needed to run the R/C truck's data management system.

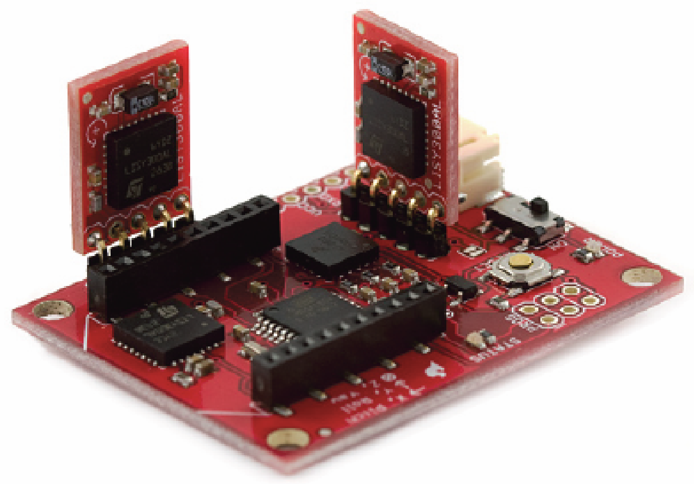


Figure 28. Atomic 6 DOF IMU XBee Ready.

The IMU came preinstalled with a bootloader and a command line user interface. The interface consisted of a menu navigation that allowed customization of the unit.

```
6DOF Atomic setup, version 1.0
================================
1) View/edit active channel list
2) Change output mode, currently binary
3) Set Auto run mode, currently off
4) Set accelerometer sensitivity, currently 1.5g
5) Set output frequency, currently 100
6) Save settings and run unit
```

Figure 29. IMU command line menu.

From the menu, the user could configure the output mode that toggles from binary to ASCII output. ASCII was used mainly for viewing data straight out of the terminal. For this application, binary was a more suitable output method as it did not require any conversion. In binary mode, the data gets sent in two 8bit pieces to form one 16 bit piece for each measurement. The IMU microcontroller contained a 10 bit analog to digital converter, which required at the least 16 bits to store—assuming data was sent in 8bit increments. In addition to the 16bits per measurement—accelerometers X, Y, and Z and gyroscopes for pitch, roll, and yaw—there were two framing bytes that got sent at the beginning and at the end of each package of data. These framing bytes allowed the main program to determine what each number represents. The only means of distinguishing one measurement from another was to keep track of how many bytes of data had been sent since the starting frame byte. The data was always sent in a particular order—Start byte, Count, Acceleration X, Acceleration Y, Acceleration Z, Pitch, Roll, Yaw, End Byte. With all the measurements the IMU sent, the total amount of data that got transmitted was 16 bytes.

IMU Data Package

'A'	Count	Accel X	Accel Y	Accel Z	Pitch	Roll	Yaw	'Z'
1 Byte	2 Bytes	2 Bytes	2 Bytes	2 Bytes	2 Bytes	2 Bytes	2 Bytes	1 Byte

Figure 30. IMU Data package format.

The start and stop bytes represented by ASCII characters in figure 29, but should not be confused with binary and ASCII output modes. All ASCII characters have the same binary representation whether the IMU outputs binary or ASCII. The output mode toggling is meant for the data being sent.

Decimal	Oct	Hex	Binary	Symbol	HTML No.	HTML Name	Description
48	060	30	00110000	0	&#48;	--	Zero
49	061	31	00110001	1	&#49;	--	One
50	062	32	00110010	2	&#50;	--	Two
51	063	33	00110011	3	&#51;	--	Three
52	064	34	00110100	4	&#52;	--	Four
53	065	35	00110101	5	&#53;	--	Five
54	066	36	00110110	6	&#54;	--	Six
55	067	37	00110111	7	&#55;	--	Seven
56	070	38	00111000	8	&#56;	--	Eight

Figure 31. ASCII Character table excerpt.

For example, if the acceleration about the X axis was recorded by the IMU to be 834 the ASCII output would be three successive numbers, 8 (00111000), 3 (00110011), and 4 (00110100) represented in ASCII by virtue of their binary representation. Note that the binary representation of the numbers is not mathematically equivalent. When the IMU value 834 is sent in ASCII mode the binary string 00111000-00110011-00110100 will be sent in its place. In binary mode, the number 834 will be represented by its mathematical equivalent in binary form. The number 834 will be sent as 0000001101000010. It was advantageous to work in binary output mode because the value received from the IMU was mathematically equivalent what the measurement reading with no conversion necessary.

Unfortunately, because UART communication supports only 8bits of data being sent at any given time, it was a challenge to transmit the 16bit data to the main computer for processing. The data from the IMU was stored using the little endian method of storing bits. The little endian method stores the least significant bit into the smallest address.

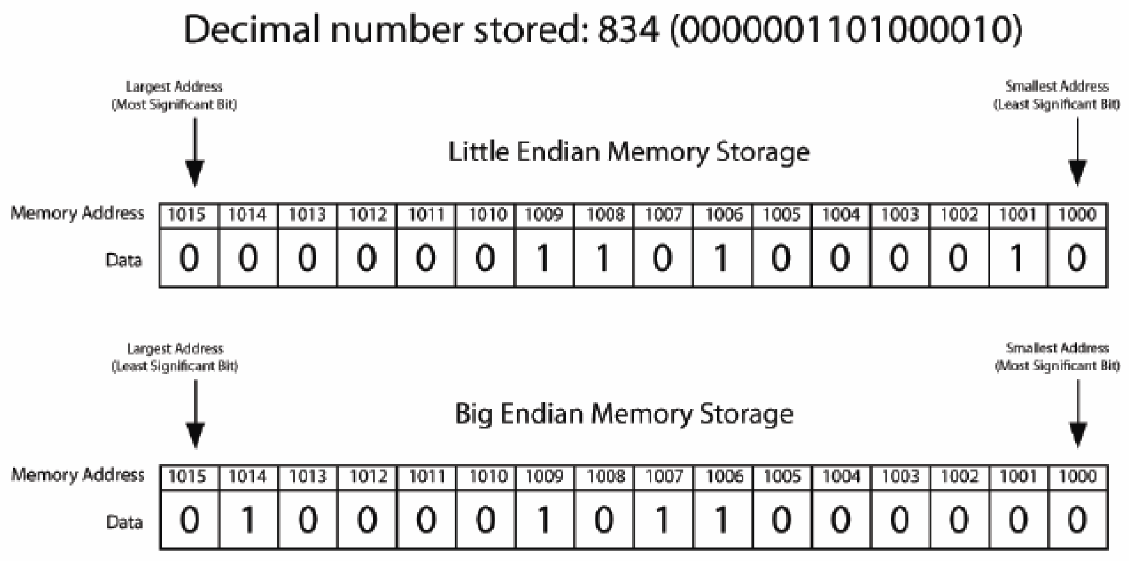


Figure 32. Little Endian and Big Endian memory storage.

The little endian method may be easier to understand because it follows the traditional way we read, from left to right. The use of little endian or big endian is based heavily on preference and application. In this application, the data was sent first byte, then second byte and pieced back together as such. This allowed the main computer to assume that the first byte received of each measurement pair was the beginning of the number and the second byte was the end of the number. The main CPU concatenated the two bytes into one 16bit number and continued to do so for the remaining measurement pairs.

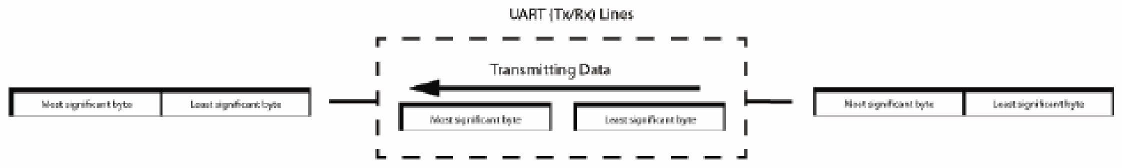


Figure 33. Little Endian data transmission through UART.

The data from the IMU enters and exits the UART communication line in the same order making it easier and less cumbersome to process the data.

Using the command line menu makes configuring the IMU easy. However, when connected to the dsPIC33F microcontroller, the luxury of a user interface is gone. Instead, the IMU recognizes certain characters to correspond with certain configuration commands. For example, to run the IMU in binary mode, the microcontroller would send an ASCII '#' (pound) symbol to configure the IMU. Similarly, other features such as data collection frequency and sensitivity used various ASCII characters for configuration purposes. The IMU was configured to run at a frequency of 50Hz with outputs in binary. A frequency of 50Hz was chosen because of the refresh rate of the microcontroller. The microcontroller data monitoring based on the main program runs at 1Hz, or one update per second. Therefore, the IMU did not need to be updated past one cycle per second but was not configurable any lower than 50Hz.

APPENDIX B: DEVELOPMENT BOARD SCHEMATIC

The on-board control is performed by two microcontrollers, MCU1 and MCU2. MCU1 collects data from the IMU and the other two encoders; then transmits this data to MCU 2. MCU2 communicates with the CPU, commands the steer servo and throttle motor, collects two of the encoder values, and receives data from the MCU1. This setup was done because one of the dsPICs alone cannot process all the data without having transmission errors. Not included in the board schematics is the power regulator, which steps the input voltage down to a stable 3.3 volts.

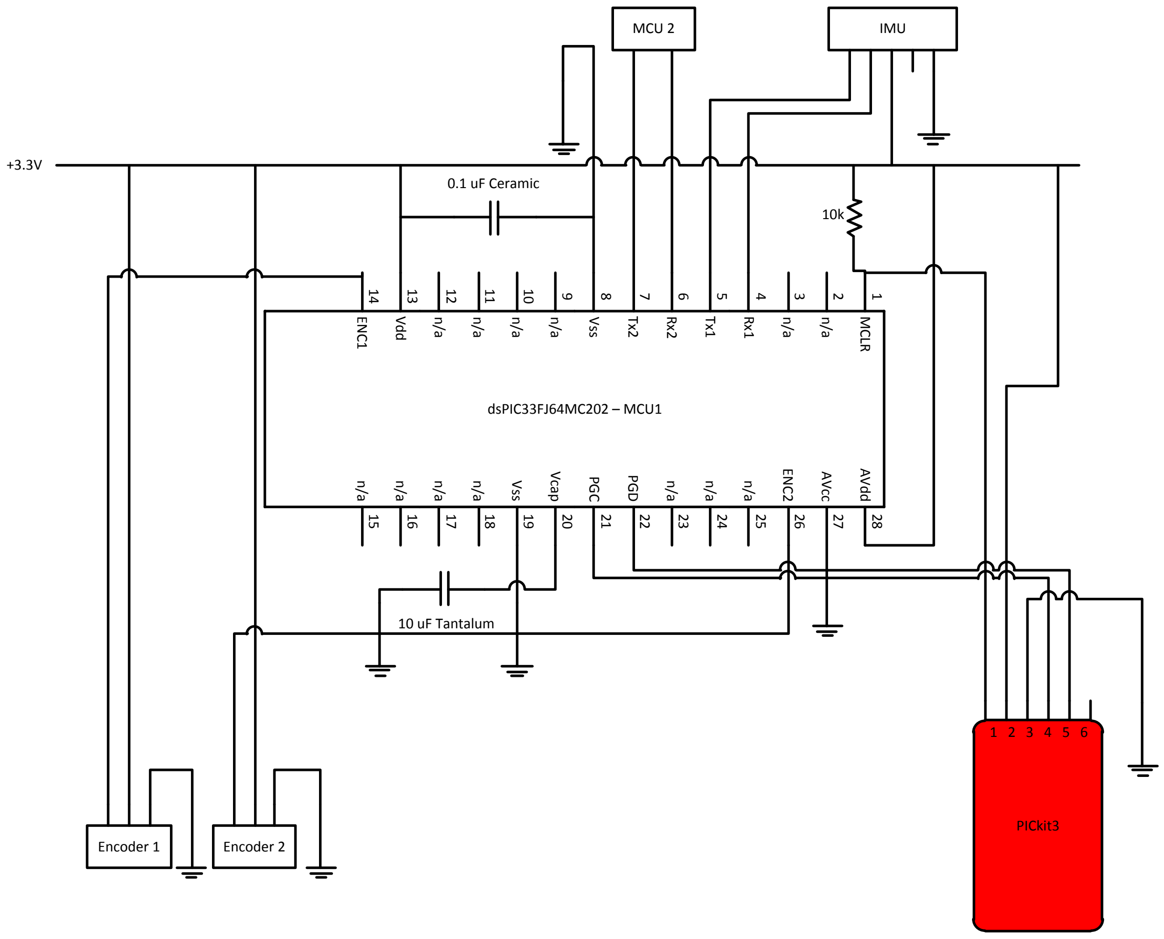


Figure 55: MCU1 schematic

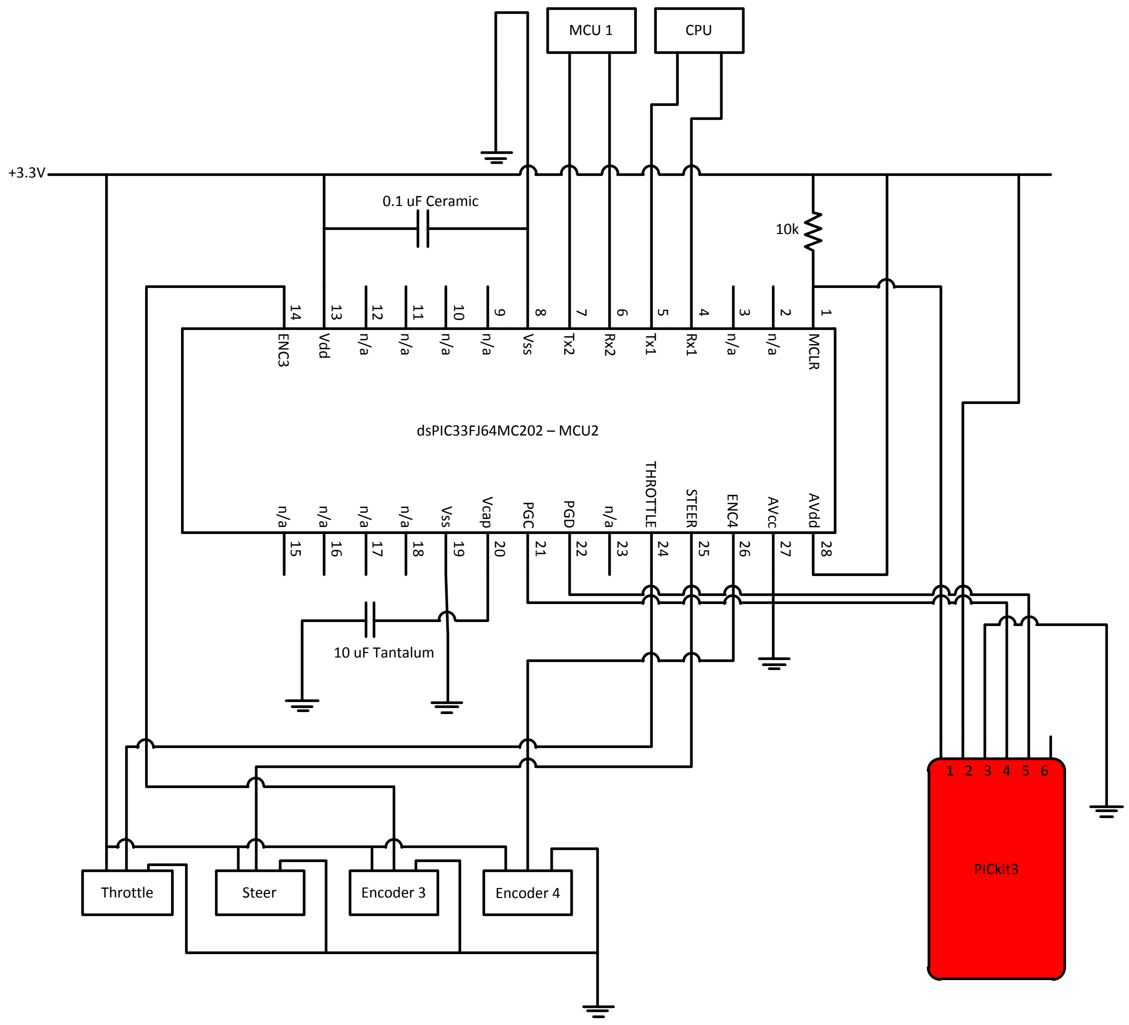


Figure 56: MCU2 schematic

6.4.0 MATLAB Rapidly-Exploring Random Tree

6.4.1 MAIN_BLOCK.m

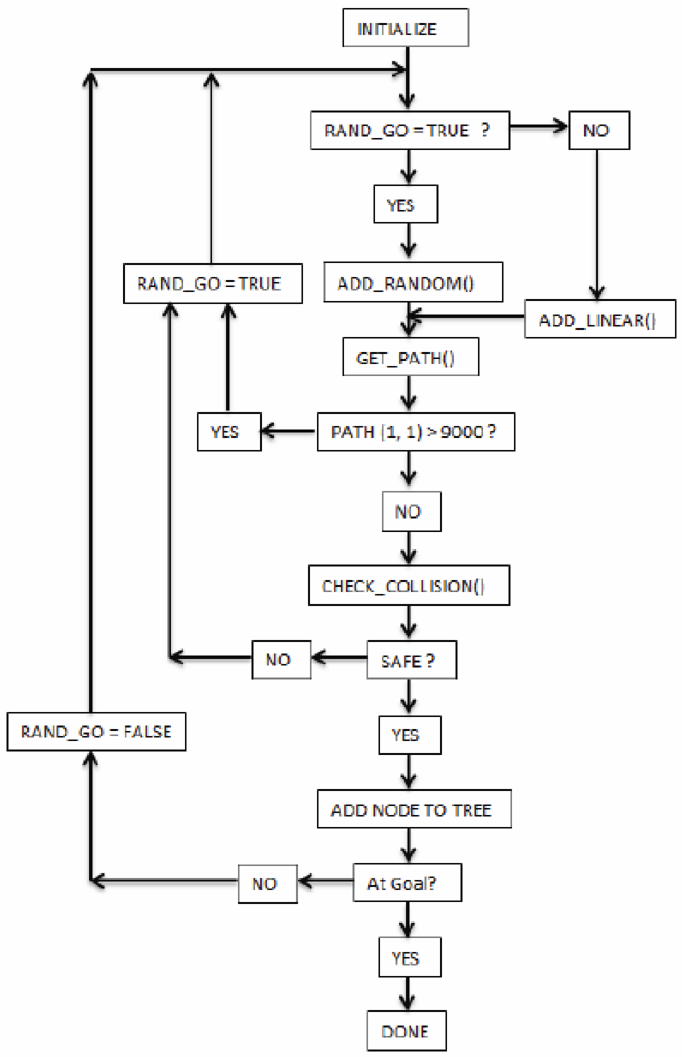


Figure 34. Rapidly-exploring Random Tree algorithm flow chart.

MATLAB was used to develop and run the rapidly-exploring random tree algorithm. The main file used for debugging and testing the software was MAIN_BLOCK.m. This file was used to prepare the code for Simulink and displays the final path with a video. At the start of MAIN_BLOCK the user can set the goal location of the path planner as well as the dimensions of the vehicle, obstacle, and road. The program then loops until the location of the vehicle is within a certain range of the goal. At the beginning of the loop the node and path trees are initialized so that points from the previous iteration do not interfere with the current iteration, and the flags are reset. Next a random space to search for nodes is set up so that all nodes used to generate the path are in front of the vehicle.

The program then enters another loop that will cycle until a path has been completed or it determines that no path can be found. Next the program checks to see if it should generate a node along a straight path to the goal or one in the random space (ADD_LINEAR or ADD_RANDOM). This node, TEMP_NODE, is then sent to GET_PATH in order to generate a feasible vehicle path to TEMP_NODE and stores that path in the PATH array. A flag is set if the first element is greater than 9000, a value outside the range of a valid path, which tells the main program that no path could be generated. For testing purposes all valid paths are plotted at this point. If the path is not valid then it will return to the top of the loop and try to add another random node. If the path is valid then the program will run CHECK_COLLISION. This function returns a 1 if the path is not safe and a 0 if it is safe. If the path is safe then TEMP_NODE will be added to the TREE array otherwise the program will try to add a new random node. Once a path has been completed to the goal then the program will exit the loop and plot the path.

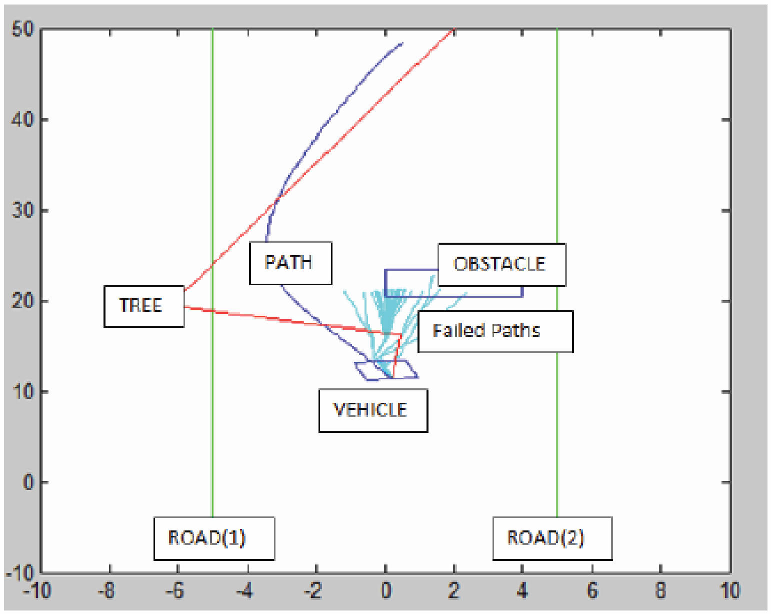


Figure 35. Single path generation instance of the RRT algorithm.

The tree generated by the RRT is shown in red, the path needed to follow the tree is in dark blue, and failed paths are in light blue. TREE is only the final tree branch used to generate the final path. The paths in light blue are working off different branches of the RRT network.

6.4.2 ADD_RANDOM.m

```
function TEMP_NODE= ADD_RANDOM(NODE_DIST, TREE, NODES, RAND_SPACE)
```

This function returns TEMP_NODE, a node to be tested before it can be added to TREE. It accepts TREE, NODE_DIST, NODES, and RAND_SPACE as parameters. TREE is an array of all the current nodes in the random tree and NODE_DIST is the distance between those nodes. NODES are the number of nodes in TREE, and RAND_SPACE is the search space for random nodes. ADD_RANDOM first selects a random point in the search space RANDOM_NODE. Next it searches through TREE for the closest node to the random node, CLOSE_NODE. Once CLOSE_NODE is found the function calculates the location of a temporary node, TEMP_NODE, at a predefined distance from the close node and in the direction of RANDOM_NODE. TEMP_NODE is then returned to the calling program

6.4.3 ADD_LINEAR.m

```
function TEMP_NODE= ADD_LINEAR(NODE_DIST, TREE, NODES, GOAL)
```

This function returns TEMP_NODE as well. It also accepts TREE, NODE_DIST, NODES, and GOAL as parameters. GOAL is the goal location of the path planner. CLOSE_NODE is set as the last node added to TREE. TEMP_NODE is then calculated to be at a certain distance from CLOSE_NODE in the direction of GOAL. TEMP_NODE is returned to the calling program.

6.4.4 GET_PATH.m

```
function [PATH, PATH_COUNT] = GET_PATH(TREE, TEMP_NODE, NODE_DIST, VEHICLE, VEHICLE_GEO)
```

This function is used to generate a path to the TEMP_NODE and returns a completed path the calling program, PATH, and the number of elements in PATH, PATH_COUNT. It accepts TREE, TEMP_NODE, NODE_DIST, VEHICLE, and VEHICLE_GEO as parameters. VEHICLE contains the current location of the vehicle, its speed, and orientation and VEHICLE_GEO contains the dimensions of the vehicle. PATH is an array that stores the position, orientation, steer angle, and time of the vehicle as it follows a path.

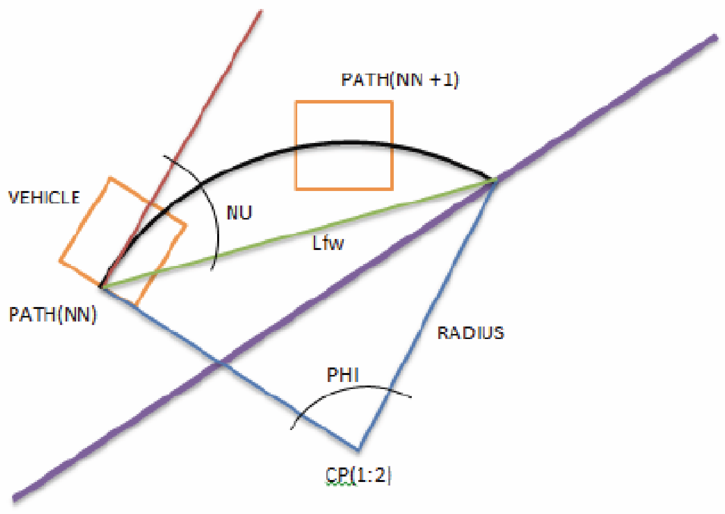


Figure 36. Path smoothing diagram based on vehicle geometry.

Figure **36** depicts the geometry and variables used to determine how the vehicle will follow the RRT network being propagated, shown in purple. NN is a counter variable used to track the vehicle at each point along the path so if the vehicle is currently at NN its next location will be NN + 1. The proceeding path location is not at the end of the turn. This is done in order to avoid oscillations about the branch path and allowing the vehicle to gradually approach the RRT path. Lfw is the linear distance between the start and end location, RADIUS is the radius of curvature for the turn that needs to be made, and CP is the x, y coordinate of the center of the arc.

First GET_PATH starts at the last element added to TREE and works back through the tree to get a single branch to be used to determine a path. This branch is stored in ROUTE_TREE. This tree is then used in conjunction with the pure pursuit path follower discussed in section 4.1.2. The function enters a loop that will cycle until the full path is generated. First Lfw is calculated to be the distance from the rear axle of the vehicle to a point along ROUTE_TREE. Next NU is calculated to be the angle between the heading of the vehicle and the direction of ROUTE_TREE. If NU is 0 then the car is going straight along the path and a point directly in front of the vehicle is added to PATH. If NU is not 0 then RADIUS is calculated to be the radius of curvature that the vehicle will need to take in order to stay on course with ROUTE_TREE. From RADIUS the steer angle, SIGMA, can be calculated.

```
%CALCULATE Lfw
Lfw=((PATH(NN,1)-ROUTE_TREE(CC-DIV_LOOK,1))^2+(PATH(NN,2)-ROUTE_TREE(CC-DIV_LOOK,2))^2)^.5;

%CALCULATE Nu
NU=atan((PATH(NN,1)-ROUTE_TREE(CC-DIV_LOOK,1))/(PATH(NN,2)-ROUTE_TREE(CC-DIV_LOOK,2)))-
PATH(NN,3);

%CALCULATE RADIUS
RADIUS=Lfw/(2*sin(NU/2));
```

PATH(NN, 1) is the current x position of the vehicle along the path and PATH(NN, 2) is the y position. PATH(NN,3) is the vehicles orientation at this point in radians. CC is a counter variable used the move down the path, and DIV_LOOK is how far ahead the path planner is looking.

SIGMA is then checked to against the maximum turning capabilities of the vehicle, ANGLE_LIMIT. If and steering command exceeds this limit then the path is no good. SIGMA is stored in PATH(NN, 6). The center of the turn is then calculated using the following equations. CENTER (1) is the x coordinate and CENTER(2) is the y coordinate.

```
%LOCATE THE CENTER OF THE TURN
CENTER(1)=PATH(NN,1)+RADIUS*cos(PATH(NN,3));   %X
CENTER(2)=PATH(NN,2)+RADIUS*sin(PATH(NN,3));   %Y
```

After the center of the turn is found the position of the vehicle after it makes the turn is added as the next element of PATH. PHI is the angle between the start and end of the turn and THETA is the orientation of the vehicle at the start of the turn.

```
%ADD NEXT NODE TO THE TREE [x,y,theta]
PHI=-2*NU;
PATH(NN,1)=CENTER(1)-RADIUS*cos(THETA+PHI/DIV_LOOK);
PATH(NN,2)=CENTER(2)-RADIUS*sin(THETA+PHI/DIV_LOOK);
PATH(NN,3)=PHI/DIV-THETA;
```

GET_PATH then determines the time it would take the vehicle to get from the start of the turn to the end, TIME_DIF. VEHICLE(4) contains the velocity of the vehicle in ft/s. TIME_TOT is the total time the car will take to get to the turn being looked at. The total time up to the current turn is stored in PATH(NN, 5).

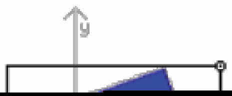
```
%CALCULATE THE TIME DIFFERENCE BETWEEN THE CURRENT AND LAST NODE
TIME_DIF=(((PATH(NN,1)-PATH(NN-1,1))^2+(PATH(NN,2)-PATH(NN-1,2))^2)^.5)/VEHICLE(4);

TIME_TOT=TIME_TOT+TIME_DIF;
PATH(NN,5)=TIME_TOT;
```

NN is incremented after each steering command is calculated so that PATH gets filled with the state of the vehicle as it should travel along the path. Once the path is completed the entire PATH array along with the number of points in PATH is returned to the calling program.

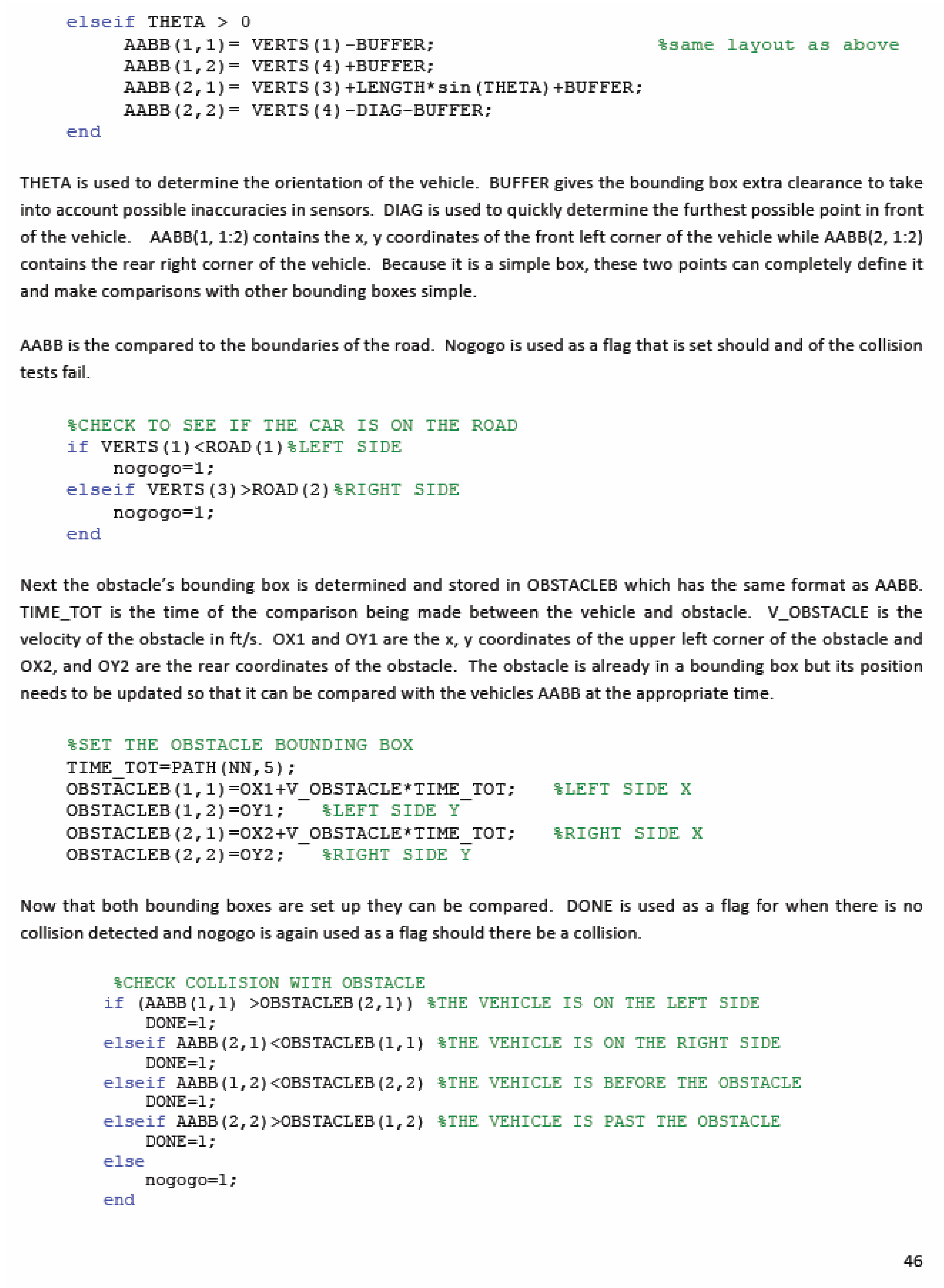
6.4.5 CHECK_COLLISION.m

```
function nogogo = CHECK_COLLISION(PATH, OBSTACLE, VEHICLE_GEO, ROAD, PRINT)
```

CHECK_COLLISION is a function that accepts PATH, OBSTACLE, VEHICLE_GEO, ROAD, and PRINT as parameters. It checks whether or not the vehicle can follow the proposed path without colliding with any obstacles or going off the road. OBSTACLE



[redacted]



```
    elseif THETA > 0
         AABB(1,1)= VERTS(1)-BUFFER;                    %same layout as above
         AABB(1,2)= VERTS(4)+BUFFER;
         AABB(2,1)= VERTS(3)+LENGTH*sin(THETA)+BUFFER;
         AABB(2,2)= VERTS(4)-DIAG-BUFFER;
    end
```

THETA is used to determine the orientation of the vehicle. BUFFER gives the bounding box extra clearance to take into account possible inaccuracies in sensors. DIAG is used to quickly determine the furthest possible point in front of the vehicle. AABB(1, 1:2) contains the x, y coordinates of the front left corner of the vehicle while AABB(2, 1:2) contains the rear right corner of the vehicle. Because it is a simple box, these two points can completely define it and make comparisons with other bounding boxes simple.

AABB is the compared to the boundaries of the road. Nogogo is used as a flag that is set should and of the collision tests fail.

```
    %CHECK TO SEE IF THE CAR IS ON THE ROAD
    if VERTS(1)<ROAD(1)%LEFT SIDE
        nogogo=1;
    elseif VERTS(3)>ROAD(2)%RIGHT SIDE
        nogogo=1;
    end
```

Next the obstacle's bounding box is determined and stored in OBSTACLEB which has the same format as AABB. TIME_TOT is the time of the comparison being made between the vehicle and obstacle. V_OBSTACLE is the velocity of the obstacle in ft/s. OX1 and OY1 are the x, y coordinates of the upper left corner of the obstacle and OX2, and OY2 are the rear coordinates of the obstacle. The obstacle is already in a bounding box but its position needs to be updated so that it can be compared with the vehicles AABB at the appropriate time.

```
    %SET THE OBSTACLE BOUNDING BOX
    TIME_TOT=PATH(NN,5);
    OBSTACLEB(1,1)=OX1+V_OBSTACLE*TIME_TOT;   %LEFT SIDE X
    OBSTACLEB(1,2)=OY1;   %LEFT SIDE Y
    OBSTACLEB(2,1)=OX2+V_OBSTACLE*TIME_TOT;   %RIGHT SIDE X
    OBSTACLEB(2,2)=OY2;   %RIGHT SIDE Y
```

Now that both bounding boxes are set up they can be compared. DONE is used as a flag for when there is no collision detected and nogogo is again used as a flag should there be a collision.

```
         %CHECK COLLISION WITH OBSTACLE
        if (AABB(1,1) >OBSTACLEB(2,1)) %THE VEHICLE IS ON THE LEFT SIDE
            DONE=1;
        elseif AABB(2,1)<OBSTACLEB(1,1) %THE VEHICLE IS ON THE RIGHT SIDE
            DONE=1;
        elseif AABB(1,2)<OBSTACLEB(2,2) %THE VEHICLE IS BEFORE THE OBSTACLE
            DONE=1;
        elseif AABB(2,2)>OBSTACLEB(1,2) %THE VEHICLE IS PAST THE OBSTACLE
            DONE=1;
        else
            nogogo=1;
        end
```

46

6.4.6 RUN_RRT.m

```
function FINAL_PATH = RUN_RRT(VEHICLE, OBSTACLE, ROAD, GOAL)
```

RUN_RRT.m is a function that accepts VEHICLE, OBSTACLE, ROAD, and GOAL as parameters and returns a possible path. VEHICLE contains the state of the vehicle and OBSTACLE contains the state of the obstacle. ROAD contains the dimensions of the road and goal is the goal of the path. It has the same functionality as MAIN_BLOCK.m but lacks the ability to plot. It is also a standalone function instead of an executable like MAIN_BLOCK is. This allows RUN_RRT to be passed parameters and return a path when used in the command line of MATLAB or Simulink.

6.5.0 Simulink Model

6.5.1 Overview

Simulink was chosen to run the path planning algorithm in real-time and also for communication with the two microcontrollers we employed. Multiple custom blocks were developed in order to use live data from sensors in conjunction with the RRT algorithm. The ROAD and GOAL blocks are hard coded values meant to reflect the testing scenario. GOAL is the goal location of the path planner and ROAD contains the distance of the shoulders from the centerline of the road. The Vehicle block contains the initial properties of the vehicle: x, y coordinate, orientation from the x-axis, and velocity.

6.5.2 COM 3, COM 5

Simulink has library blocks which can automatically listen to various communication ports on a computer, such as Ethernet or serial. Since this system communicates through serial wire, two blocks configure the properties of these ports so that they properly interpret the signals received from the dsPIC and Xiphos board, as well as send the correct outputs to the dsPIC. (explain specific configurations).

6.5.3 IMU_DATA

The model receives live data about the state of the vehicle from the dsPIC. The data is received one byte at time so the IMU_DATA block is used to concatenate the data into an array the RRT_BLOCK can use. The IMU data is in a 10 bit format and comes in high byte then low byte. To put the two numbers together the high byte is multiplied by 2^8, a mathematical bit shift, and then added to the low byte, this done for the all parameters measured by the IMU. This block was left incomplete because we were unable to send the IMU data through the PIC successfully. The IMU_DATA block would also be used to receive the encoder data from the dcPIC chip.

6.5.4 OBSTACLE

The OBSTACLE block is used to combine obstacle data from the Xiphos board with the preset obstacle data into a single array. The format is as follows: [x, y, L, W, V]. The coordinates of the obstacle are in x, y format. L corresponds to the length of the obstacle in the y direction, W is the width of the obstacle in the x direction, and V is the velocity of the obstacle in ft/s. OBSTACLE can also be swapped out for a constant block contain the same array of elements should it be desirable to hard code the properties of the obstacle.

6.5.5 RRT_BLOCK

The RRT_BLOCK is used to execute the RUN_RRT function discussed in section 6.4.6. It accepts vehicle, obstacle, road, goal, trigger, and clock data as parameters. GOAL and ROAD are preset characteristics. GOAL is where the vehicle is planning on going and ROAD is the dimensions of the road. The trigger is used so that RUN_RRT will on execute once when triggered instead of continually generating a new path with each iteration of Simulink. A step function is followed by a derivative block for the trigger signal. The derivative returns the slope of the incoming signal. The step function has a constant slope except for when it changes values; at this point its slope is very large. And the trigger port is held high. RRT_BLOCK will only execute RUN_RRT if the trigger port is greater than 0. The clock input is used to ensure the time stamp for the steering commands matches the time of Simulink. The output of the block is an array storing the executable path. The path is stored in a data store so that it remains constant after the next iteration of Simulink allowing other blocks to access it as the model runs.

6.5.6 TURN_OUT

The TURN_OUT block accepts a path array stored in memory, clock, and vehicle velocity as parameters. It compares the time stamps for the turning commands in the path array to the Simulink clock. If the two times correspond to each other the proper steering command will be sent out along with a throttle command. The output format is [250, steer, throttle]. This array is then sent to the dsPIC to control the vehicle. The array starts with 250 in order to signal to the microcontroller that steering and throttle commands will be sent next. Both the steering and throttle are scaled from 0 to 240 to ensure a large resolution since they are sent as 8 bit integers. Steering commands range from -120 to 120 where -120 is a maximum left turn and 120 is a maximum right turn. It is then shifted so that -120 is 0 and 120 is 240. This was done because the PIC uses unsigned numbers in order to set the duty cycles for the steering servo and drive motor.

6.5.7 Real- Time Pacer

The Real-Time Pacer block is a user created block from the MATLAB website that allows Simulink to be run in sync with the wall clock, basically in real time. Simulink needs to be run in real time for this project so that the turning commands will be executed properly. Ideally Simulink should be run in its own real time kernel, but because many custom blocks are being used developing a real time kernel is exceedingly complex. The Real-Time Pacer is a simple work around that continually adjusts the Simulink clock to match the computers system clock. Although it is not as exact as the real time kernel, very accurate results can be achieved.

6.6.0 Xiphos

6.6.1 Ultrasonic.c

The Xiphos board is used to read ultrasonic range finders, calibrate the readings, and send them to the computer. The ultrasonic sensors used are Maxbotix LV-EZ3. The program used on the board has two modes that can be run, calibration mode and auto mode. In calibration mode the user can set the calibration constant used for the sensors in order to determine an optimal calibration. Figure 37Figure **38** demonstrates how the ultrasonic sensors are calibrated. D is the true distance to the object and Z is an 8 inch dead zone if front of the sensor where a constant reading of 2 is seen. If R is the actual reading of the sensor and C is the calibration constant then equation 6.1 can be used to determine the true distance.

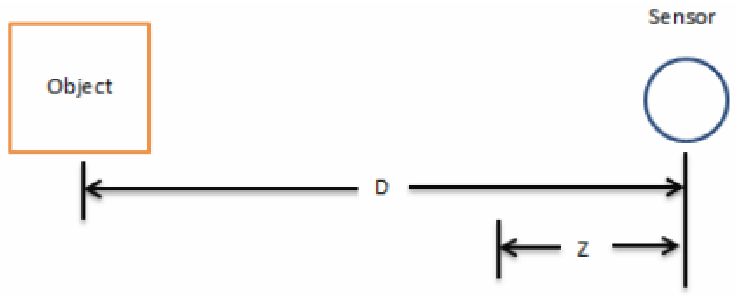


Figure 38. Obstacle measurement references.

D = C (R-2) + 8 **Equation 6.1**

When the user starts the program in calibration mode the following menu is seen

```
a: display ultrasonic reading on port 0
s: display ultrasonic reading on port 2
d: run both ultrasonic sensors in series
q: calibrate ultrasonic on port 0
w: calibrate ultrasonic on port 2
```

A and s simply show the selected sensors reading after calibration, d sets both sensors to run in series so that their pulses do not interfere with each other and q and w allow for the setting of the calibration constant with a number between 0 and 9. The following code implements equation 4 to calibrate the raw sensor readings.

```
inputed1= (10+cal1)*(reading1-2)/10+8;
```

The calibration constant, cal1, is initially scaled by a factor of 10 to achieve a higher resolution when using integer values. After some testing, cal1 was found to be optimal at 9 resulting in a calibration constant of 1.9 with an accuracy of ±1 inch.

In auto run mode the program returns the width, location, and velocity of the object and sends this data to a computer over a serial connection. The sensors are fired one after another so that the sonar pulses from the different sensors do not interfere with each other and the Maxbotix LV-EZ3 sensors make this very simple. The following lines of code demonstrate how this was implemented.

```
 //read ultrasonic on port 0 first
//pull the RX pin on sensor 1 high for 1 ms
 PORTA = PORTA | 0b10000000;
 delayMs(1);
 PORTA = 0b00000000;
 //Wait for a completed reading
 delayMs(50);
//store the reading
 inputed1= analog (0);
```

49

```
//read ultrasonic on port 2 second
PORTA = PORTA | 0b01000000;
delayMs(1);
PORTA = 0b00000000;
delayMs(50);
inputed2= analog (2);
```

The ultrasonic sensor documentation states that to command a reading the RX pin of the sensor must be pulled high for at least 0.02 milliseconds. When RX is set to low again then the sensor will stop emitting sonar pulses. There is then a 49 millisecond delay until a voltage is set on the sensors analog pin therefore after 50 milliseconds the program reads the corresponding analog port and stores the value.

In order to obtain the desired properties of the object the sensors need to be set up according to Figure **39**. The Variable X is the distance to the center of the object from sensor 1, W is the width of the object, and S is the separation between the two sensors. All units are in inches.

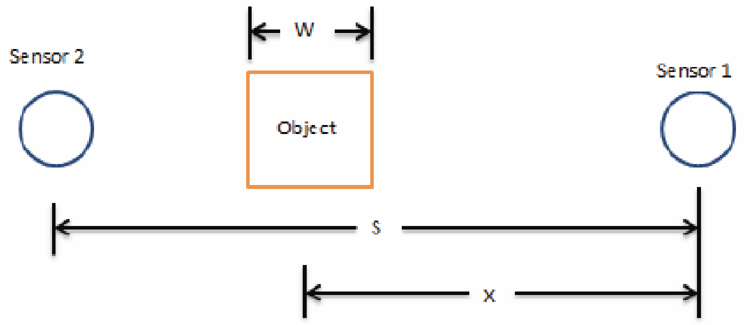


Figure 39. Two sensor array for object tracking.

Based on this geometry the following code is used to obtain the width, position, and velocity of the object.

```
//calculate the width of the object
width = separation-(inputed1+inputed2);

//calculate where the center of the object is
current_position= inputed1+width/2;

//calculate the velocity
// v=d_position/d_time
// d_time = .1s
// value is scaled by 10 for higher accuracy when dealing with integers
velocity = (current_position-previous_position);

//save the current position as the previous position
previous_position=current_position;
```

Inputed1 and inputed2 are the calibrated readings from both sensors 1 and 2 respectively. The velocity is determined by comparing the current position to the previous position and then multiplying the difference by the time between the two readings. Each ultrasonic sensor takes 50 milliseconds to complete a reading resulting in a total of 0.1s for both to be read. Since 0.1 s is much longer than any other process the difference in time between the current and previous position can be assumed to be 0.1s. The velocity is scaled by 10 in order to achieve a higher accuracy since only integers are being used.

6.6.2 Wiring configuration

Table 5. Xiphos microcontroller ultrasonic array configuration

		Xiphos
Sensor 1	Rx	Port A Pin 7
	AN	Analog 0
Sensor 2	Rx	Port A Pin 6
	An	Analog 2

6.7.0 R/C Truck Encoders

The 8 DOF dynamic model needed to know the speed of each wheel in order to function. These data were gathered by four encoders, which were comprised of infrared sensors and a patterned disk attached to each wheel. The sensors passed this data to the onboard microcontroller, which ultimately delivered it to the CPU.

6.7.1 QRE1113 Line Sensor Analog Breakout Board

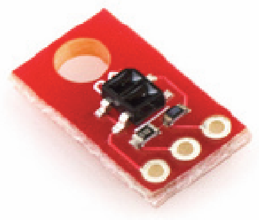


Figure 40. QRE1113 Line Sensor Analog Breakout Board.

The sensor utilized by the encoder is the QRE1113 Line Sensor Analog Breakout Board [Appendix O]. These sensors included two components: an infrared LED which emitted light outwards towards a surface, and an infrared sensor which detected incoming infrared light being reflected by the surface. This board also had a mounting hole designed to fit a #4 screw, which was used to attach the sensor to a mounting bracket attached to the R/C truck.

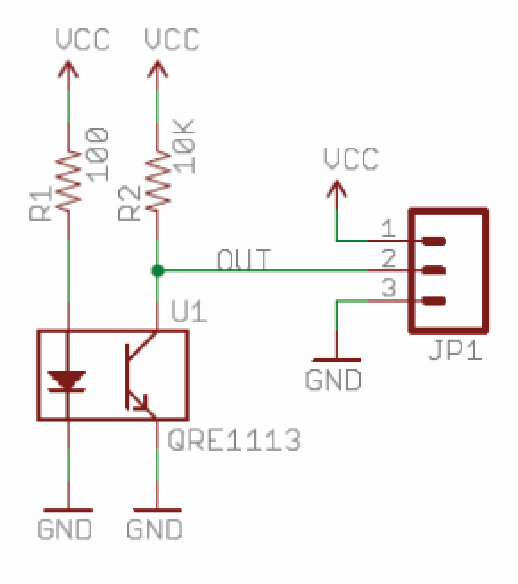


Figure 41. Circuit Diagram of QRE1113 IR Sensor

The circuit was arranged such that the output pin was high when there was little light being sensed, and was pulled lower as it received more light. As the surface changed in reflectivity, the voltage signal alternated between high and low voltage, which was analyzed by the microcontroller. These alternations are known as 'ticks'.

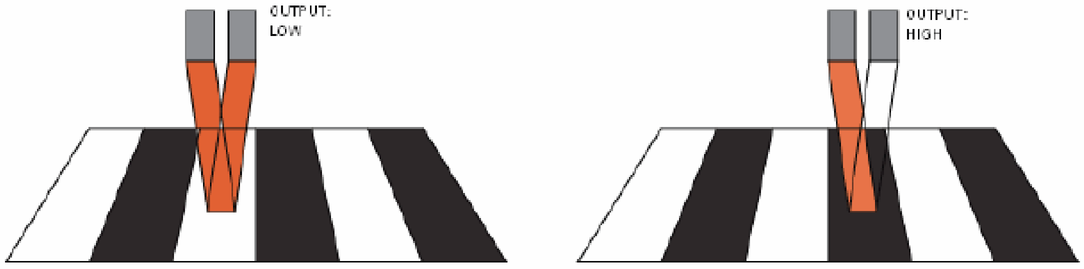


Figure 42. Infrared Sensor response to reflective surfaces

A digital version of the QRE1113 was originally purchased under the expectation that it would output a digital high/low signal that required no processing power, however it was discovered that instead it utilized a capacitor discharge system, in which a pulse-width digital signal is transmitted which is intended to gauge variable reflectivity rather than a binary pattern, and as such would require much more resources to process.

6.7.2 Encoder Disk

The IR sensor tracked the speed of the wheel by reading the ticks of a patterned disk as it rotated with the wheel and counted them. Since the disk's pattern was known, the number of ticks counted only had to be divided by the number of spokes on the wheel to yield rotation. This disk was cut out of acrylic by a laser cutter. It measured 2.5 inches in diameter so that it could be nested in the hub of the R/C truck wheel. It was cut such that it slid onto the

APPENDIX D: TEST VEHICLE PHYSICAL PARAMETERS FOR DVM

Longitudinal distance between the center of tire and vehicle center of mass

$l_f = 212.019\ mm$

$l_r = 126.490\ mm$

Front and rear track width

$t_f = 307.311\ mm$

$t_r = 307.552\ mm$

Distance between ground and static roll center

$h_f = 21.898\ mm$

$h_r = 42.535\ mm$

Height of sprung mass CG above ground

$h_{cgs} = 107.315\ mm$

Height of front/rear unsprung mass CG above ground

$h_{uf} = 51.18\ mm$

$h_{ur} = 51.12\ mm$

Total vehicle mass

$M = 3116.74\ g;$

Front/Rear unsprung mass

$M_{uf} = 342.34\ g$

$M_{ur} = 329.66\ g$

Moments of inertia of sprung mass about its CG

$I_{xxs} = 5768307.05\ \ g\text{-}mm^2$

$I_{yys} = 43789753.01\ \ g\text{-}mm^2$

$I_{zzs} = 45206669.73\ \ g\text{-}mm^2$

Products of inertia of sprung mass about its CG

$I_{xys} = 658566.19\ \ g\text{-}mm^2$

$I_{xzs} = 918367.67\ \ g\text{-}mm^2$

$I_{yzs} = 67637.22\ \ g\text{-}mm^2$

Moment of inertia of wheel about its spin axis

I_{tlf} = 203865.64 g-mm^2

I_{trf} = 203865.64 g-mm^2

I_{tlr} = 203865.64 g-mm^2

I_{trr} = 203865.64 g-mm^2

Front/rear suspension spring stiffness

K_{sf} = 491 N/mm

K_{sr} = 577.9 N/mm

Front/rear shock damping

B_{sf} = 15 N-s/mm

B_{sr} = 15 N-s/mm

Front/Rear anti-roll bar stiffness

K_{rf} = 0 Nm/rad *(no roll bars installed on test vehicle)*

K_{rr} = 0 Nm/rad *(no roll bars installed on test vehicle)*

Front/Rear anti-roll bar damping

B_{rf} = 0 Nm/rad *(no roll bars installed on test vehicle)*

B_{rr} = 0 Nm/rad *(no roll bars installed on test vehicle)*

Steering-to-wheel angle ratio

K_{sr} = 1

Tire Width/Radius

t_{wid} = 44.5 mm

t_{rad} = 54.0 mm

Cornering Coefficient, Longitudinal Coefficient, Torque Coefficient

$C_{c\alpha}$ = 0.2 deg^{-1}

$C_{\kappa\alpha}$ = 0 deg^{-1}

$C_{T\alpha}$ = 0 deg^{-1}

APPENDIX E: TEST VEHICLE MATERIAL PROPERTIES

Part	Volume (mm^3)	Mass (g)	Density (kg/m^3)
AA battery	8078.3758	29.375	3636.250742
AA battery case	5508.35	12.779	2319.932466
amp	33466.14	84.5	2524.940134
arm (front)	19724.29	16.7	846.671794
arm (rear)	20353.98	17.1	840.1305298
axle (rear)	4588.82	17.4	3791.824478
battery	131913.17	377.6	2862.489015
bellcrank (left)	3803.39	4.9	1288.324363
bellcrank (right)	3096.25	4.4	1421.07388
bumper (front)	49973.96	56.7	1134.590895
bumper (rear)	68964.28	72.2	1046.918782
caster block (front)	2712.27	2.9	1069.215086
chassis	184381.04	276.15	1497.713648
controller	45245.45	105.5	2331.726174
draglink	851.95	4.65	5458.066788
IMU and mount	39290.93	80.9	2058.999367
motor	62743.24	249.475	3976.125556
radio	42680.3	28.65	671.2698833
roll cage	62732.76	453.1	7222.701504
servo saver	4442.47	5.6	1260.560004
shock bottom (front)	8935.77	15.3	1712.21954
shock bottom (rear)	11354.79	20	1761.371192
shock top	1009.8	3	2970.885324
skid plate (front)	21052.29	30.3	1439.273352
skid plate (rear)	21537.37	30.998	1439.273352
steer block (front)	2953.31	13.9	4706.583461
steer motor	26534.62	43.15	1626.177424
suspension mount (front)	54743	34.7	633.8709972
suspension mount (rear)	43547.15	57.4	1318.111518
tie rod (front)	2874.97	6.25	2173.935728
tie rod (rear)	2990.43	6.25	2090.000435
transmission	202139.26	217.225	1074.630431
turnbuckle (long)	3057.58	7.5	2452.920283
turnbuckle (short)	2191.68	3.3	1505.694262
wheel 1	307502.85	116.8	379.833878
wheel 2	307502.85	125.5	408.1262987
wheel 3	307502.85	126.5	411.378301
wheel 4	307502.85	127.3	413.979903