The Dynamics and Control of Independent Wheel Drive Electric Vehicles

Ryan Scott

A thesis
submitted in partial fulfillment of the
requirements for the degree of

Master of Science in Mechanical Engineering

University of Washington

2014

Reading Committee:

Brian Fabien, Chair

Santosh Devasia

Ashley Emery

Program Authorized to Offer Degree:
UW Mechanical Engineering

University of Washington

Abstract

The Dynamics and Control of
Independent Wheel Drive Electric Vehicles

Ryan Scott

Chair of the Supervisory Committee:
Professor Brian Fabien
Mechanical Engineering

This thesis is a preliminary investigation of the dynamics and control principles as applied to 2 Independent Wheel Drive (2IWD) vehicles. It will discuss the derivation of the dynamic models for a vehicle with independent drive wheels and variable front tire angles as well as the experimental validation of a fixed tire angle model. In addition, a controller will be developed to control the vehicles velocity, and orientation with integrated traction control. Controllers will be rated on their ability to meet performance specifications while minimizing power consumption. Finally, it will be shown how these techniques may be applied to more general cases with fluctuating ground conditions.

TABLE OF CONTENTS

Page

List of Figures iv

List of Tables v

Glossary vi

Chapter 1: Introduction 1
1.1 Motivation 1
1.2 Methodology 2
1.3 Outline of Work 3

Chapter 2: Experimental Design 5
2.1 Introduction 5
2.2 Mechanical Design 5
2.3 Electrical Design 7
2.4 Data Acquisition 10
2.5 Sensor Noise and Measurement Uncertainty 10
2.6 Summary 14

Chapter 3: Dynamic Modeling 16
3.1 Introduction 16
3.2 Nonlinear Formulation 16
3.3 Near Linear Model 27
3.4 Linear Model 30
3.5 Model Validation 35
3.6 Summary 43

Chapter 4: Controller Design and Implementation 44
4.1 Introduction 44
4.2 Choice of Performance Index 45
4.3 Output Feedback 46
4.4 Traction Control 50
4.5 Parametric Controller Weighting 51
4.6 Controller Validation 54
4.7 Summary 56

Chapter 5: Simulation and Results 58
5.1 Introduction 58
5.2 Software Packages Employed 58
5.3 Additional Approximations 59
5.4 Model Verification 59
5.5 Traction Control Implementation 62
5.6 Fluctuating Traction Conditions 64
5.7 Summary 65

Chapter 6: Conclusions 67
6.1 Summary of Work 67
6.2 Future Work 67
6.3 Final Statements 68

Bibliography 69

Appendix A: Part Documents 70
A.1 Accelerometer Electrical Diagram 71
A.2 Micro-Controller Electrical Diagram 72
A.3 H-Bridge Spec Sheet 73

Appendix B: Complete Electrical Diagram 74

Appendix C: Arduino Code 76

Appendix D: Simulation Code 94
D.1 Vehicle Characterization 95
D.2 Experimental Controller Analysis 101
D.3 Linear Simulink Model 107
D.4 Nonlinear Simulink Model 130
D.5 Continuous Traction Control and Slip Instants 158
D.6 Discrete Traction Control and Slip Instants 162

LIST OF FIGURES

Figure Number		Page
2.1	Original Motor and Drive Train, First, and Second Revisions	6
2.2	Dual Drive Shaft Model	6
2.3	Custom Suspension with Wheel Mounted Motors	7
2.4	Final Experimental Design	8
2.5	Possible Motor Driving Circuitry	9
2.6	Multi-rate Sampling Logic	14
3.1	FBD of Bulk Dynamics	18
3.2	FBD of Wheel	20
3.3	Coefficient of Friction vs Slip Ratio	21
3.4	Wheel Velocity Estimation	24
3.5	Motor Characterization Experiment	37
3.6	Motor Characterization Results and Simulation	39
3.7	Trajectory of Vehicle to Open Loop Step Input	40
3.8	Vehicle Characterization Results	41
3.9	Simulation of Traction Effects on Wheel Speeds	42
4.1	Output Feedback Block Diagram	45
4.2	Slip Dependent Controller Weighting	53
4.3	Experimental Controller Implementation	54
4.4	Experimental Results Using PI Control (Part 1)	56
4.5	Experimental Results Using PI Control (Part 2)	57
5.1	Simulation of Open Loop Response	60
5.2	Open Loop Response	61
5.3	Nonlinear Simulations of Perturbed System	62
5.4	Discrete Traction Response to a Sudden Slip	65
5.5	Continuous Traction Response to a Sudden Slip	66

LIST OF TABLES

Table Number Page

3.1 Definition of Parameters 29
3.2 Motor Characterization Constants 39
3.3 Vehicle Characterization Constants 43
4.1 Experimental Results Using PI Control 55
5.1 Final Position and Orientation in Perturbed Dynamics 62

GLOSSARY

CLOSED LOOP: describes a dynamic system that employs a feedback loop to compare the system behavior against the desired behavior and adjust accordingly.

DRIFT: small deviations from the desired course. This will normally occur slowly over time.

FREE SPIN: a condition where a wheel is free to rotate with out resistance from ground interactions

OPEN LOOP: Describes a dynamic system that receives an input but does not employ a feedback loop.

PULSE WIDTH MODULATION: the process by which an approximate analog signal is generated by creating a digital pulse train of variable duration and frequency.

RELAXED: describes a state that is in equilibrium with a value of zero.

RISE TIME: the time that is required for a system to move from 10% to 90% of its steady state response.

ROTARY ENCODER: a device that measures the angular position of a shaft from the orientation of offset components (mechanical, magnetic, etc.).

SLIP RATIO: a dimensionless measure of the amount that the wheel will roll versus slide.

STEADY STATE: a state experienced after a sufficiently long period of time, such that the system has come to an equilibrium behavior.

TRAJECTORY: the path traveled by the vehicle through space and time.

TRANSIENT RESPONSE: the response of a system to a new or changing input as it moves to a new equilibrium.

YAW: the angle measured from a datum axis to the forward direction of the vehicle.

ACKNOWLEDGMENTS

The author would like to thank all those that contributed to this project. Most, notably, Professor Fabien, for his guidance and insight into the planning and execution of this project. Sylvie Troxel, for her efforts in the fabrication of the experimental apparatus. Andrew Davis and Jonathan Realmuto for their discussions of model simulation and filter implementation.

Chapter 1

INTRODUCTION

1.1 Motivation

In recent decades the automotive industry has begun to shift its efforts from vehicles powered entirely by internal combustion technology to a purely electric or a hybrid-electric power source. As battery and motor technology continues to improve, these alternative approaches are becoming more commercially viable. In addition, the use of electric motors leads to a number of benefits related to systems and controls. They respond quickly, motor torques can be accurately determined and controlled in real time (leading to improved traction and braking), and finally, they can be easily scaled up or down. These advantages lead to a number of opportunities to implement new vehicle concepts. One such concept is the Independent Wheel Drive (IWD). This utilizes a network of electric motors to drive each wheel independently (in a 2IWD or 4IWD configuration). IWD has four main objectives:

1. The reduction of vehicle weight by removing traditionally required mechanical components such as the drive train and differential. When each wheel is driven independently the control system will replace these components by varying the power applied to each motor.

2. The reduction of vehicle cost: The idea being that by requiring a network of smaller electric motors as opposed to a single large motor (and the infrastructure that it requires), it may be possible to lower the production cost, therefore, making a more commercially viable product.

3. The reduction of power consumption: When operating a network of smaller motors, each of which is subjected to a lower load, it may be possible to reduce steady state power consumption. In other cases, such as turning, it may be possible to reduce the power to the motors on the inside of the turn. This could reduce the power consumed during various dynamic situations.

4. Improve vehicle performance: This includes a broad spectrum of issues, including traction control, reduced stopping distances through dynamic braking, improved acceleration profiles, yaw rate control (resistance to putting the vehicle into a spin), and dynamic drift control (accounting for road conditions, tire wear, and motor imbalances to prevent drifting).

1.2 Methodology

As mentioned above, there are many different facets to IWD vehicles. This work will focus primarily on the 4^{th} objective outlined above. This will be done by designing a controller with three objectives; maintaining a driving trajectory (drift rejection), achieving a desired velocity profile, and maintaining traction (or recover it in the event of excessive slip). These three objectives are interrelated. The rate at which power is applied to each wheel will determine possible slip scenarios as well as generate yaw moments (in the event of non-symmetric vehicle dynamics).

The foremost objective of the controller is to maintain vehicle stability. This includes the subjects of traction and drift control. Drift control has been previously accounted for by mechanical components (differential) as well as the driver's steering input. It is important for the controller to correct for a variety of scenarios where drift will be introduced, and in the event that it is introduced, correct the trajectory with minimal overshoot. It should be clear that an over-aggressive controller could potentially make the system unstable by repeatedly over correcting for drift. The desire is then to generate a drift controller that is

almost critically damped so that the vehicle returns to the desired trajectory with minimal oscillations around the final value. Traction control is a more straight forward topic. Often times this is accomplished by a controller that is left in standby until a slip ratio exceeds a maximum specified in the control law, at which point it will move to regain traction. However, there are more active control methods which continually monitor and adjust motor behavior to keep the slip ratio within a desired band.

The secondary objective is the speed moderation. The controller will be designed to bring the vehicle from an initial velocity profile to a final profile while not violating the goals described above. The performance of this portion can be improved by allowing for larger drifts and slips until the vehicle is brought close to the operating point, where the drift and slip are then brought back into acceptable limits. Or alternatively, this objective can be implemented in conjunction with the former so that all parameters are adjusted together. This performance will be dependent on the weighting of each control input, based off of the user's specified priorities.

1.3 Outline of Work

This work will investigate one specific case of 2IWD with the following goals.

1. The development of suitable modeling techniques. This will begin with a full nonlinear model, followed by the linear approximation.

2. Determine the degree to which the model must be reflected in the on board computation. This will determine whether the controller should be given the ability to simulate and thus predict its own dynamics based on sensor information.

3. Design of a controller capable of meeting the required performance criteria of traction

control, drift rejection, and velocity control.

4. Design and fabricate a test bed for the validation of parts 1-3.

Chapter 2
EXPERIMENTAL DESIGN

2.1 Introduction

One of the primary goals of this work was the design and construction of a small scale 2IWD vehicle. This will serve as a test bed for future control concepts and provide a proof of concept for the methods described in this paper. This was accomplished by retro-fitting a remote control car with a new drive train and power supply in such a way that the torque delivered to the drive wheels may be controlled independently. This introduces three main challenges. The first of which is the mechanical design. The chassis and suspension of the vehicle must be adapted so that an additional motor may be attached, in as close to a symmetric manner as possible. Next, the electrical system must be augmented. The original system was only capable of operating a single motor and did not have the capability of implementing a custom control algorithm. Finally, a controller and sensor network must be established, such that, a sufficient amount of data is available to make a well informed decision and execute a variety of control algorithms.

2.2 Mechanical Design

The car originally housed a single electric motor with a differential, drive train, and suspension system. Prior to the beginning of this project, another group of students had removed said motor and attached a dual motor drive [8] using belt connections to the original suspension system (Figure 2.1). It was decided that this design left room for unacceptable amounts of delay between the motor dynamics and the wheel/ground dynamics. Multiple approaches were attempted using the original suspension with a symmetric drive shaft. However, the

tolerances required for functionality were not achieved in the time allotted. Overall this design was exceptionally sensitive to bends in the main drive shaft. Even small deflections from straight were sufficient to bind the drive train.

Figure 2.1: Original Motor and Drive Train, First, and Second Revisions

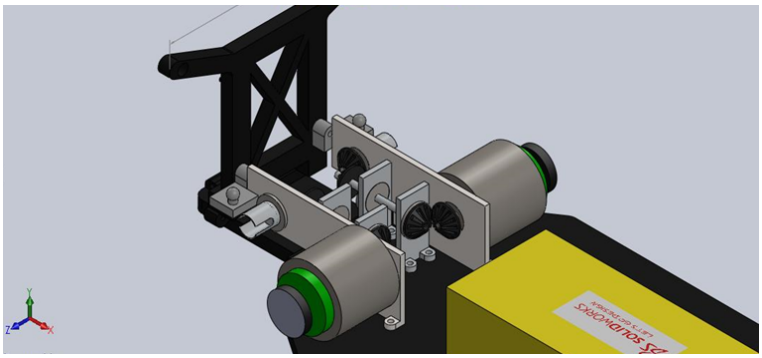


Figure 2.2: Dual Drive Shaft Model

A new approach was taken where the suspension was redesigned to house a motor directly attached to the wheel. While this design provided a smoother motor/wheel interface and maintained the vehicles suspension, it was eventually abandoned due to its high sensitivity to motor diameter and the location of the drive shaft on the motor. Finally a rigid body

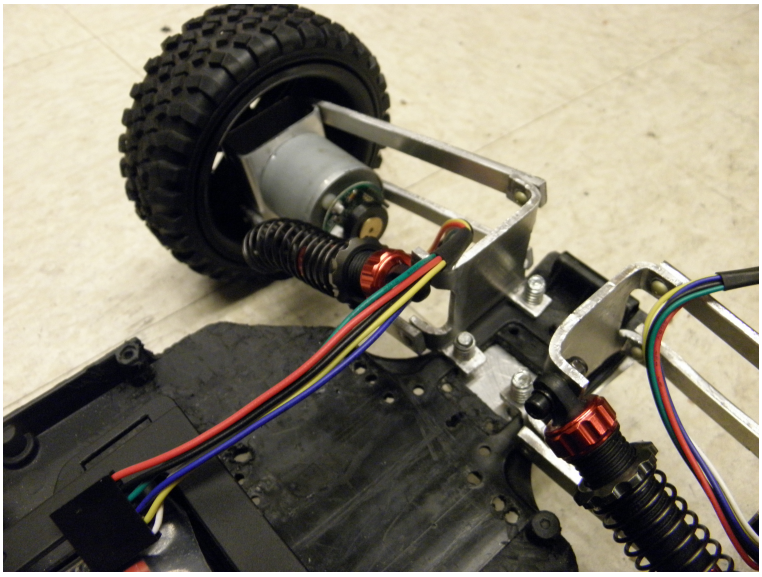


Figure 2.3: Custom Suspension with Wheel Mounted Motors

design was adopted in the interest of time. This final iteration houses the motor on the main vehicle chassis while maintaining the direct motor/wheel characteristics. A portion of the suspension dynamics are preserved through the softening of the front suspension, allowing for small chassis twisting angles in response to terrain. Now all necessary deflections and torques can be addressed in the front suspension with minimal effects to the overall vehicle performance. This is acceptable only for the scale being addressed in this work; for a full scale vehicle the rear suspension should be introduced. However, literature demonstrates the promise of hub mounted motors [5].

2.3 Electrical Design

The original electrical system derived its power from an on board battery pack and was controlled by a hardwired Electronic Speed Control (ECS) module. While this did provide sufficient power for a single motor, it did not allow for the addition of a second. In addition

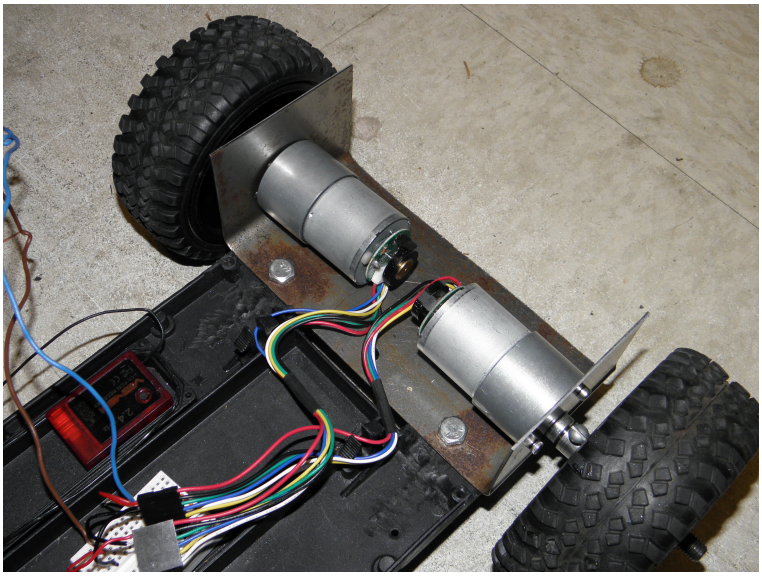
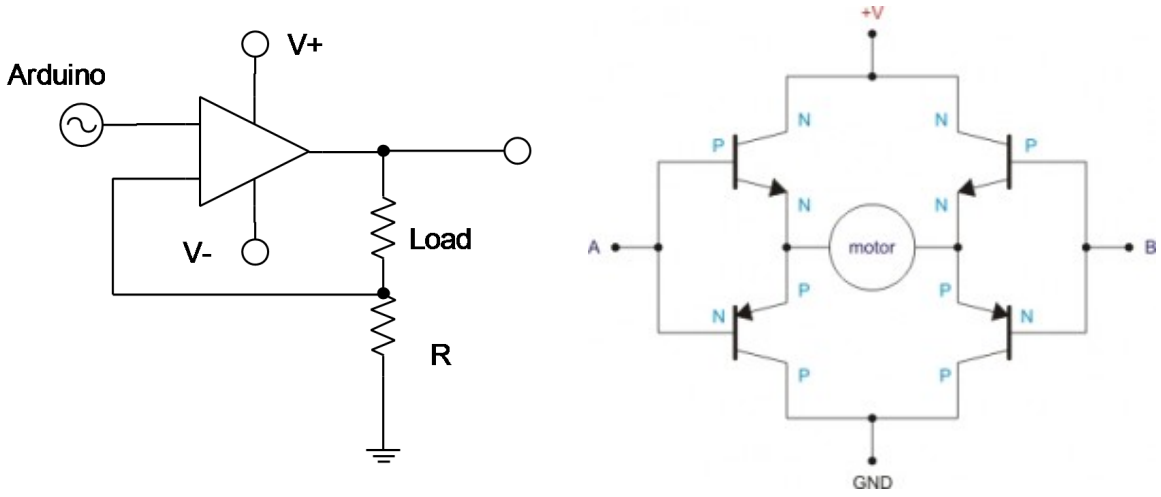


Figure 2.4: Final Experimental Design

the ECS was not capable of controlling two independent motors. Therefore, it was decided that these components should be replaced. This would require a power supply, a microcontroller, and a means of driving the motors.

As a starting point an Arduino Mega 2650 [1] board was chosen for the microcontroller. This will allow the incorporation of a number of different inputs (data from sensors, etc.) as well as generate the required control signals for the motors. Finally, this will allow for the software controlling the Arduino to be changed with ease and data to be extracted for modeling purposes. While the controller can produce the desired signal, it does not have the capability to supply sufficient current to run the motors. Therefore, the signal must be augmented by a power source. This leads into the next part of the design.

There are a number of ways to use a control signal to manipulate the flow of a greater power source. Most notably are Operational Amplifiers (op-amps) and Bipolar Junction



(a) Op-Amp with Negative Feedback

(b) Example of H-Bridge Driving a Motor [1]

Figure 2.5: Possible Motor Driving Circuitry

Transistors (BJTs). An op-amp amplifies the difference between two signals, or as shown below, magnifies a single signal by a specified gain through the sizing of the resistors. A BJT on the other hand can be used as a gate to moderate the flow of current through it. In this case a PWM signal will be used to control the BJT, yielding a voltage close to that supplied, operating in the same manor as the control signal. Based on the components immediately available it was decided to use a network of BJTs (an H-Bridge.) This allows for the control of the motor speed and direction. In the interest of ensuring sufficient power (regardless of motor requirements)an external DC power supply was chosen with a tether to the body of the vehicle. A full system diagram may be found in Appendix B.

[1]Figure used from: `http://www.dannyvanmaanen.nl/?p=149&lang=en`

2.4 Data Acquisition

In order for the controller to make meaningful decisions it must be provided with sufficient information to determine its state of affairs. Given that the vehicle is allowed to move in 5 Degrees of Freedom (DOFs), namely translational velocities in the X and Y directions, rotation about the Z axis, and the rotational velocity of each of the rear wheels (Figure 3.1). If these quantities cannot be determined directly, data that is related should be acquired so that these parameters may be estimated. Knowing what sort of information is required structures the sensor choice. The bulk properties of the vehicle may be determined from an accelerometer. This work utilized the PS-MPU-6000A for this task (Appendix A). This module was used to measure the bulk planar acceleration of the vehicle as well as the rotational velocity. By use of an integrator and filter the bulk velocity and angular orientation can be calculated in real time.

A key component to the choice of the motors for this experiment were the built in encoders on the main shaft. These allow the angular position of the shaft to be determined at specific points in time, and therefore, the wheel velocity to be calculated. This information, coupled with that of the accelerometer, is sufficient to estimate the slip ratio of each wheel and therefore determine where, when, and to what degree traction is lost. One possible control method use this information for Optimal Slip Control [3] in order to improve driving performance.

2.5 Sensor Noise and Measurement Uncertainty

All measurements taken in reality are inherently subject to some degree of uncertainty and random noise. Because of these issues a single measurement is not sufficient to estimate the true value. However, this may be overcome, either through the implementation of data smoothing hardware (Op Amp filters, capacitor power isolation) or through software based

approaches. This work will utilize both, but will primarily focus on the software methods designed to minimize the risk of using inaccurate data.

The accelerometer mentioned above incorporates a system of capacitors to isolate the power supply to minimize fluctuations in the circuitry (effecting the reliability of the measurements). There are a large number of software methods that can assist this. There are digital filters that compute a likely data point depending on the current measurement and a certain number of previous measurements (depending on the order and type of filter). There are also averaging methods, assuming the noise is truly random and evenly distributed around the real value. This work makes use of both, filters and averaging.

2.5.1 Filter Design

The purpose of a filter is to limit the effects of noise on the general behavior of a set of data points. This work makes use of a first order low pass filter which effects low frequency signals minimally, while significantly damping the effects of high frequency signals. It is common for noise to operate in high frequency ranges (in the case of the bulk vehicle dynamics, significantly higher). The digital low pass filter is designed around a step response function. Essentially, if each discrete data point is considered as an input function, held constant between the sampling instants, and then undergoes a step to the next point, the filter would be the first order response to this input. The output of the filter will be a smooth contour, continuous in time, approximately fitting the data set. What this implies is that for a single (or small number of) data point(s) that deviate significantly from the remainder of the data set, the filter response will move in that direction, however, the effects of the following points will bring it back into reasonable ranges.

When designing a filter, the cutoff frequency must be chosen. This frequency determines how rapidly the filter will respond to each successive data point. In short, a high cutoff

frequency will respond quickly to rapidly fluctuating signals, while a low cutoff frequency will move sluggishly toward the new input. This can be seen mathematically. Below is the differential equation that describes a low pass filter with a cutoff frequency of a (in Hz), where f is the filtered value, and u is the data input.

$$\dot{f} + a\, f = a\, u \tag{2.1}$$

This equation can now be discritized, using the backward Euler's approximation of the derivative, so it may be directly implemented in the digital realm.

$$\frac{f(k) - f(k-1)}{\Delta t} + a\, f(k) = a\, u(k) \tag{2.2}$$

Now solve for $f(k)$ in terms of $u(k)$ and $f(k-1)$.

$$f(k)\left(\frac{1}{\Delta t} + a\right) = a\, u(k) + \frac{1}{\Delta t} f(k-1)$$
$$f(k)\left(\frac{1 + a\Delta t}{\Delta t}\right) = a\, u(k) + \frac{1}{\Delta t} f(k-1)$$

$$f(k) = \left(\frac{a\Delta t}{1 + a\Delta t}\right) u(k) + \left(\frac{1}{1 + a\Delta t}\right) f(k-1) \tag{2.3}$$

Consider the effects of the cutoff frequency again. In the above equation it becomes clear that if a is allowed to become small the first coefficient will tend to zero while the second coefficient will tend to one. Meaning that the current filter value will be approximately the previous filter value with small influences from the current data point. Should a become large the first coefficient will tend to one, while the second tends to zero. Therefore, the choice of a effectively determines the relative influence of each term. In most cases Δt will be near constant, meaning that these coefficients may be computed before hand saving on processor time.

2.5.2 Data Averaging

In this system it was found that a simple filter, of the type described above, is insufficient to produce meaningful data. A stationary average is employed to generate a more stable data point prior to filtering. This is accomplished by taking a number of data points in rapid succession and then averaging these results. The idea being that if the points are taken rapidly enough compared to the sampling period, the average value may be used to approximate a single sampling instant with the average value. The number of points required to determine a good estimate of the true value is dependent on the amplitude of the noise and the frequency at which it occurs.

2.5.3 Multi-rate Sampling

In light of the discussion in the previous section, it is clear that it could become too computationally intensive to average and then filter all results. In an effort to simplify the computations required in the acquisition of data (and thus shorten the sampling period) the sensor noise in each measurement should be approximated, and the sensors that consistently yield more stable results can be sampled at a slower rate, skipping the averaging process entirely.

Consider the mechanics of how each measurement is taken. The encoders, for example, count the number of rotations that the motor shaft makes. This number divided by the time spent counting produces the angular velocity of the shaft. In order to get stable measurements it is necessary to count rotations over a span of no less than 20 miliseconds. While this produces an accurate and precise measurement, it is also the slowest measurement process. Therefore, the controller may save time by only measuring the motor steeds once per loop and then filtering, while the accelerometer results are sampled at a rate approximately 30 times faster. This multi-rate sampling has been shown to be an effective means of minimizing computational time while maintaining measurement stability. Figure 2.6 illustrates this, where j is the current iteration number, n is the number of samples to average, T_{s_1} is the sampling period associated with the accelerometer, and T_{s_2} is the sampling period associated with the encoders.

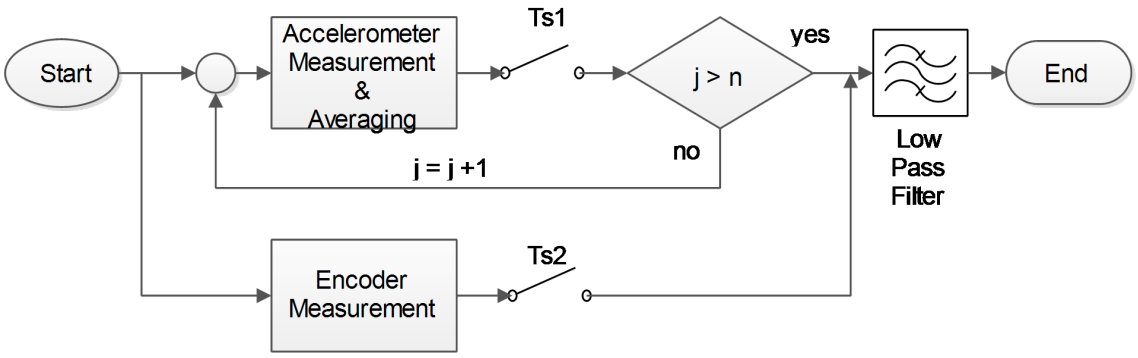


Figure 2.6: Multi-rate Sampling Logic

2.6 Summary

This chapter discussed the means by which the test bed was designed and fabricated. It demonstrated the successful implementation of one of the numerous methods to address this

problem; using rigid chassis mounted motors controlled by H-bridge circuitry. A future incarnation of this project should revisit the design of a suspension system as well as a portable power supply with more robust circuitry.

Chapter 3
DYNAMIC MODELING

3.1 Introduction

A preliminary step to designing a controller is to understand the basic dynamics of the system. Without this information there is no guarantee that a controller will not cause the system to become unstable. It is the objective of this chapter to quantify the dynamics of the vehicle described previously. The behavior of this system can be completely described with just seven states. The bulk linear velocity of the vehicle ($\dot{Y}$), the bulk sliding (lateral) velocity ($\dot{X}$), the yaw angle (ϕ), the wheel velocities ($\omega_{1,2}$) , and the slip ratios ($\lambda_{1,2}$). If the tire angle is allowed to vary (α) this must also be known, or applied as an input. The following sections will outline the derivation and significance of the dynamics of this system along with the assumptions made.

3.2 Nonlinear Formulation

The modeling of this system can be broken into two interrelated systems, namely the bulk vehicle dynamics and the wheel/motor dynamics. First, look at the bulk dynamics. These may be considered to be a rigid body acting in a relative coordinate frame where the Y-axis is oriented forward and the X-axis is oriented to the vehicle's right, under the influence of the traction forces generated by the rear wheels. The wheel dynamics are represented as an electro-mechanical system, where it is assumed that the voltage of the motor may be controlled directly, as opposed to representing the second order electrical dynamics. The combinations of these subsystems will produce a system of seven nonlinear, Differential Algebraic Equations (DAEs).

3.2.1 Bulk Vehicle Dynamics

The moving coordinate frame in this diagram is centered at the centroid of the vehicle. There are 3 degrees of freedom (DOF) in this situation, displacements in the X and Y directions as well as rotation about the Z axis (pointing out of the diagram). Figure 3.1 shows a free body diagram (FBD) of the vehicle behavior. Where, J_c is the moment of inertia of the rigid body model of the vehicle, l_i is the i^{th} distance from the center of gravity to the traction point of the tire, D is the distance between the rear tires, m is the vehicle mass, F_{T_i} is the i^{th} traction force evolved by the drive wheels, and α is the tire angle. Note that the inertial coordinate frame (X',Y') is considered static, while the vehicle coordinate frame may rotate. The majority of the computations will be performed in the vehicle's coordinate frame, however for the purpose of position tracking and drift rejection the inertial coordinates must also be known. These may be computed using a simple trigonometric transformation shown below.

$$X'(t) = X(t)\,cos\,(\phi(t)) - Y(t)\,sin\,(\phi(t)) \tag{3.1}$$

$$Y'(t) = X(t)\,sin\,(\phi(t)) + Y(t)\,cos\,(\phi(t)) \tag{3.2}$$

First apply the principles of Newtonian mechanics to the translation of the car in the Y direction (forward/backwards). The FBD shows that the net force acting on the vehicle in this direction is the difference between the force evolved by the drive wheels and the resistance of friction. The significance of the frictional forces will be discussed in following sections.

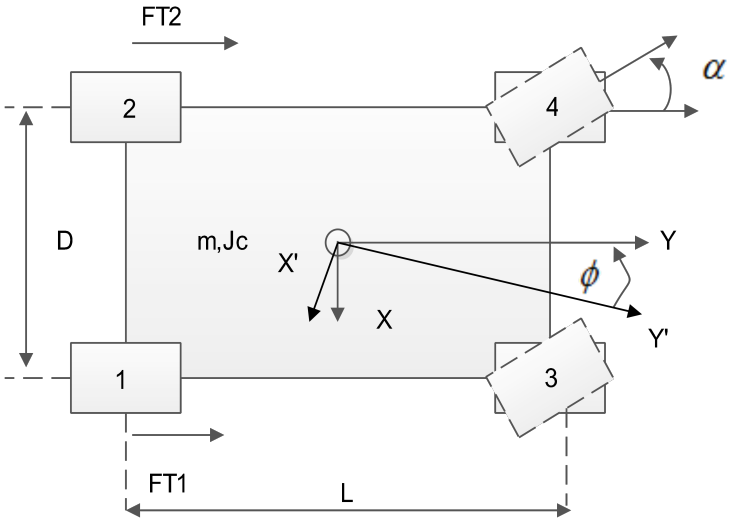


Figure 3.1: FBD of Bulk Dynamics

$$m\ddot{Y} = F_{T_1} + F_{T_2} - f_y(\dot{Y}) \tag{3.3}$$

Next evaluate the dynamics in the X direction. In this direction there are no inputs acting directly. However, the centripetal force associated with the rotation of the vehicle in the X-Y plane will cause some acceleration. Again this is counteracted by the friction of the tires, but in this case the motion is perpendicular to the direction of tire rotation, yielding a different method for compute the sliding friction forces.

$$m\ddot{X} = m\dot{\phi}\dot{Y} - f_x(\dot{X}) \tag{3.4}$$

Finally, consider the rotation of the chassis of the vehicle. When the traction forces are unequal they will generate a torque which will be counteracted by the sliding friction of the wheels (similar to equation 3.2). The effects of these frictional forces will be weighted based on their distance from the center of gravity (l_i) and the load being supported at that point (N_i).

$$J_c\ddot{\phi} = \frac{D}{2}\left[F_{T_1} - F_{T_2}\right] - f_\phi(\dot{\phi}) \tag{3.5}$$

3.2.2 Wheel Dynamics

Now a model will be formed to predict the behavior of the drive wheels as thy act under the influence of the DC motors, subject to ground influences. In the FBD, it is evident that the net torque on the wheel is found by the difference between the torque applied by the motor and the combined effects of the viscous friction of the motor bearings and the traction force. Where, J_i is the wheel/gearbox's effective moment of inertia, R is the wheel radius, ω_i is the rotational velocity of the wheel, N_i is the normal force acting on the wheel in reaction to the vehicle weight, V_i is the wheel's translational velocity, γ_i is the voltage gain of the DC motor, u_i is the applied voltage, and F_{T_i} is the traction force,

$$J_i\dot{\omega}_i = \gamma_i u_i - b_i\omega_i - RF_{T_i} \tag{3.6}$$

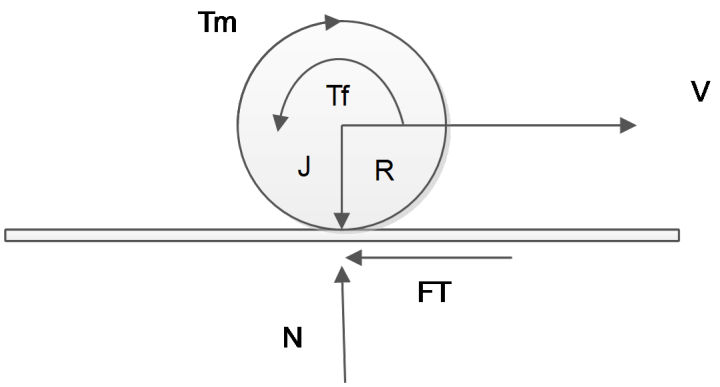


Figure 3.2: FBD of Wheel

F_{T_i} can be written as a the product of the normal force (N_i) and the coefficient of friction (μ_i).

$$F_{T_i} = N_i \mu_i \tag{3.7}$$

3.2.3 Tire Friction Model

It is important to recognize that all locomotion in the standard automobile is dependent on the traction force, which is in turn dependent on the friction between the ground and tire. This friction is a complicated relationship between the loading of the vehicle, the size and shape of the tires, the road conditions, and the rate at which the tires slip [6]. This work adopts a simplified friction model dependent only on the road condition and the slip ratio.

$$\mu = \frac{2\mu_p \lambda_p \lambda}{\lambda_p^2 + \lambda^2} \tag{3.8}$$

Where μ_p is the peak coefficient of friction for a set of road conditions, λ_p is the slip ratio at which it occurs, and λ is the current slip ratio [7]. It is worth noting that this allows for positive and negative coefficients of friction (and thus positive and negative traction forces) depending on the sign of the slip ratio. Below is the plot of λ vs μ for the road conditions used. For the purpose of this work, values for a new tire on wet cement were chosen ($\mu_p = 0.8$, $\lambda_p = 0.2$) [7] to imitate the polished nature of the floors where the experiments were performed.

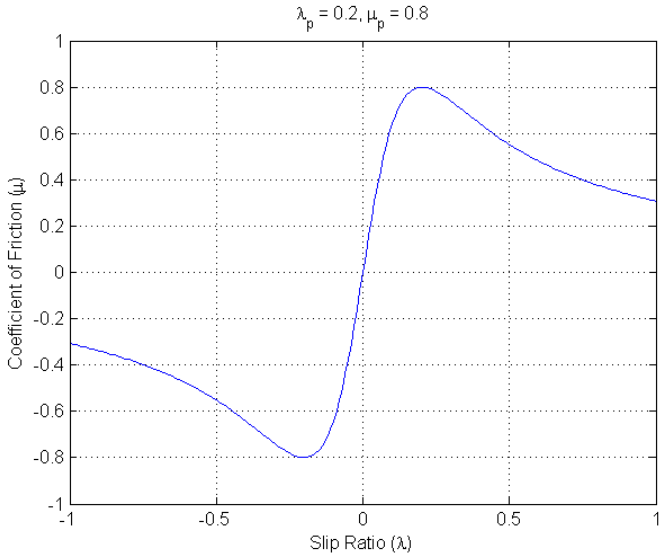


Figure 3.3: Coefficient of Friction vs Slip Ratio

A byproduct of this model is that the slip must be computed in real time for accurate

predictions. The slip model shown below is one of many possible representations. This model was chosen because of its relative ease of computation and its prediction of positive and negative slips.

$$\lambda_y = \frac{\omega R - V\,cos\,(\alpha)}{V_{max}}, \qquad \lambda_x = sin\,(\alpha)\,, \qquad V_{max} = max\,(\omega R, V) \neq 0 \tag{3.9}$$

where λ_y is the slip in the longitudinal direction, λ_x is the slip ratio in the lateral direction, and α is the tire angle [4].

This equation has two important features. The first is the appearance of both positive and negative slips. Should the car be accelerating, the wheel will be turning at a rate greater than the vehicle actually moves corresponding to a positive slip ratio ($\omega R > V$). Should the vehicle be braking, the wheels will be slowing faster than the vehicle ($\omega R < V$), corresponding to a negative slip ratio. Intuitively, this seems reasonable, however it can be confirmed by comparing equations (3.1) and (3.3). It is clear that the inertial effects of the vehicle are greater than that of the wheel and motor (corresponding to a smaller possible acceleration for a given input).

The second feature is the saturation at $\lambda = \pm 1$. This implies that if the wheel is turning in the opposite direction of the motion of the vehicle, the predicted behavior will be the same for a stationary wheel sliding ($\lambda = -1$), or conversely a rotating wheel in a stationary vehicle ($\lambda = 1$). This is significant in that the controller should not make sudden changes to the desired trajectories (suddenly reversing the direction of the drive wheels at high speeds, e.g.) to avoid slip saturation.

Now that the fundamentals of friction have been discussed, these may be applied to the

equations of motion developed in the previous sections. In equation (3.1) all of the frictional effects are combined into a single function $f_y(\dot{Y})$. This function will be composed of two parts, a viscous component, generated by the rotating mechanical parts (bearings, gears, etc.), and a sliding component, generated by the wheel angles. This sliding friction will instantaneously activate under the influence of a nonzero velocity. Therefore, the final friction force for the forward motion can be considered as,

$$f_y(\dot{Y}) = b_y\dot{Y} + N\frac{2\mu_p\lambda_p sin(\alpha)}{\lambda_p^2 + sin^2(\alpha)}sgn(\dot{y}) \tag{3.10}$$

This approach to sliding friction may be adapted to consider the lateral sliding and the rotation of the vehicle. In these cases there will be no viscous friction and there will be two components of sliding friction. One due to the sliding of the rear wheels ($\alpha = 0 \rightarrow \lambda = 1$) and the other due to the sliding of the front wheels, which will have components in the lateral and longitudinal direction (due to the tire angle). Using this information the sliding friction in the lateral and rotational cases can be expressed as,

$$f_x = \left[(N_1 + N_2)\frac{2\mu_p\lambda_p}{\lambda_p^2 + 1^2} + (N_3 + N_4)\frac{2\mu_p\lambda_p cos\,(\alpha)}{\lambda_p^2 + cos^2\,(\alpha)}\right] sgn(\dot{X}) \tag{3.11}$$

$$f_\phi = \left([l_1N_1 + l_2N_2]\,\frac{2\mu_p\lambda_p}{\lambda_p^2 + 1^2} - [l_3N_3 + l_4N_4]\,\frac{2\mu_p\lambda_p cos\,(\alpha)}{\lambda_p^2 + cos^2\,(\alpha)}\right) sgn(\dot{\phi}) \tag{3.12}$$

All of these friction models are discontinuous at zero velocity, this leads to challenges when performing numerical simulation as the system becomes very stiff at velocities close

to, or crossing zero. These difficulties are discussed more in the Simulation chapter.

Finally, these friction models consider the linear velocity of the wheel to determine slip. In the case of nonzero yaw rate, these will not be equal to the vehicle linear velocity. Using

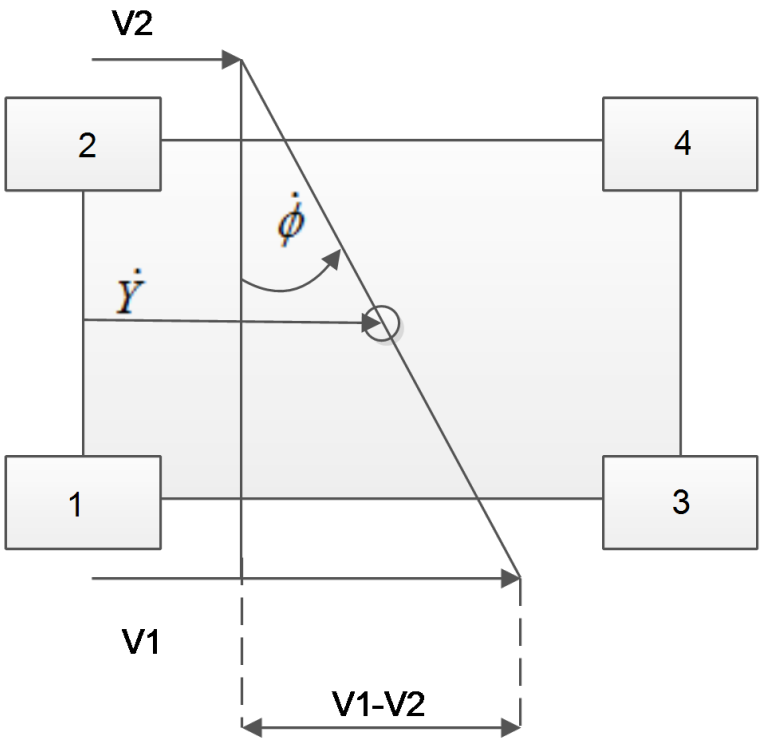


Figure 3.4: Wheel Velocity Estimation

the triangle drawn above, the difference of the wheel velocities is related to the bulk vehicle velocity by the sine of the yaw rate. Assume that the bulk velocity is the average of the wheel velocities. The following derivation outlines how the wheel velocities may be computed.

$$\dot{Y} = \frac{1}{2}\left[V_1 + V_2\right] \rightarrow V_1 + V_2 = \frac{1}{2}\dot{Y} \tag{3.13}$$

$$V_1 - V_2 = \dot{Y} sin\left(\dot{\phi}\right) \tag{3.14}$$

By the addition of (3.13) and (3.14),

$$2V_1 = \dot{Y}\left[2 + sin\left(\dot{\phi}\right)\right] \rightarrow V_1 = \dot{Y}\left[1 + \frac{1}{2}sin\left(\dot{\phi}\right)\right] \tag{3.15}$$

Through the application of the average condition (3.13), V_2 must be,

$$V_2 = \dot{Y}\left[1 - \frac{1}{2}sin\left(\dot{\phi}\right)\right] \tag{3.16}$$

3.2.4 Nonlinear State Space Representation

The final step to the formulation of the nonlinear model is to transform the algebraic equations into differential equations. After this the system will be described completely by ordinary differential equations (ODEs). This is accomplished by creating two additional states (λ_1, λ_2) as a function of the vehicle's velocity, yaw rate, tire angle, and the wheel's rotational velocity (equations 3.9, 3.15, and 3.16). This model will focus on the special case where the tire angle is zero and the vehicle is accelerating. Therefore, the slip ratios may be written

as,

$$\lambda_1 = \frac{\omega_1 R - \dot{Y}\left[1 + \frac{1}{2}sin\left(\dot{\phi}\right)\right]}{\omega_1 R} \tag{3.17}$$

$$\lambda_2 = \frac{\omega_2 R - \dot{Y}\left[1 - \frac{1}{2}sin\left(\dot{\phi}\right)\right]}{\omega_2 R} \tag{3.18}$$

Begin, by differentiating (3.17).

$$\dot{\lambda_1} = \frac{\dot{\omega_1} R - \ddot{Y}\left[1 + \frac{1}{2}sin\left(\dot{\phi}\right)\right] - \dot{Y}\ddot{\phi}cos\left(\dot{\phi}\right)}{\omega_1 R} - \frac{\omega_1 R - \dot{Y}\left[1 + \frac{1}{2}sin\left(\dot{\phi}\right)\right]}{\left(\omega_1 R\right)^2}\dot{\omega_1}$$

$$\dot{\lambda_1} = \frac{\dot{\omega_1} R - \ddot{Y}\left[1 + \frac{1}{2}sin\left(\dot{\phi}\right)\right] - \dot{Y}\ddot{\phi}cos\left(\dot{\phi}\right)}{\omega_1 R} - \frac{\lambda_1 \dot{\omega_1}}{\omega_1 R}$$

$$\dot{\lambda_1} = \frac{\left(R - \lambda_1\right)\dot{\omega_1} - \ddot{Y}\left[1 + \frac{1}{2}sin\left(\dot{\phi}\right)\right] - \dot{Y}\ddot{\phi}cos\left(\dot{\phi}\right)}{\omega_1 R} \tag{3.19}$$

This same process may be applied to (3.18).

$$\dot{\lambda_2} = \frac{\left(R - \lambda_2\right)\dot{\omega_2} - \ddot{Y}\left[1 - \frac{1}{2}sin\left(\dot{\phi}\right)\right] + \dot{Y}\ddot{\phi}cos\left(\dot{\phi}\right)}{\omega_2 R} \tag{3.20}$$

Now rewrite the traction forces in terms of the slip ratios and substitute the functions equal to $\ddot{Y}$, $\dot{\omega}_{1,2}$ and $\ddot{\phi}$ to achieve the full state space model.

$$\begin{bmatrix} \dot{\omega}_1 = f_1 \\ \dot{\omega}_2 = f_2 \\ \ddot{X} = f_3 \\ \ddot{Y} = f_4 \\ \ddot{\phi} = f_5 \\ \dot{\lambda}_1 = f_6 \\ \dot{\lambda}_2 = f_7 \end{bmatrix} = \begin{bmatrix} -\frac{b_1}{J_1}\omega_1 - \frac{R}{J_1}\frac{2\mu_p\lambda_p N_1\lambda_1}{\lambda_p^2+\lambda_1^2} + \frac{\gamma_1}{J_1}u_1 \\ -\frac{b_2}{J_2}\omega_2 - \frac{R}{J_2}\frac{2\mu_p\lambda_p N_2\lambda_2}{\lambda_p^2+\lambda_2^2}\frac{\gamma_2}{J_2}u_2 \\ \dot{Y}\dot{\phi} - b_x(\dot{X}) \\ \frac{1}{m}\left[(2\mu_p\lambda_p)\left(\frac{N_1\lambda_1}{\lambda_p^2+\lambda_1^2} + \frac{N_2\lambda_2}{\lambda_p^2+\lambda_2^2}\right) - b_y(\dot{Y})\right] \\ \frac{1}{J_c}\left[(D\mu_p\lambda_p)\left(\frac{N_1\lambda_1}{\lambda_p^2+\lambda_1^2} - \frac{N_2\lambda_2}{\lambda_p^2+\lambda_2^2}\right) - b_\phi(\phi)\right] \\ \frac{1}{\omega_1 R}\left[(R-\lambda_1)f_1 - \left(1+\frac{1}{2}sin(\dot{\phi})\right)f_4 - \dot{Y}cos(\dot{\phi})f_5\right] \\ \frac{1}{\omega_2 R}\left[(R-\lambda_2)f_2 - \left(1-\frac{1}{2}sin(\dot{\phi})\right)f_4 + \dot{Y}cos(\dot{\phi})f_5\right] \end{bmatrix} \tag{3.21}$$

3.3 Near Linear Model

Looking at (3.21), it is clear that the main nonlinearities of these dynamics are found in the modeling of the friction and the traction forces. For low slip speeds and yaw rates it is possible to approximate the sliding friction as viscous (proportional to appropriate velocity). However, a significant amount of information is lost when making a similar assumption for the traction forces. It is very common to have rapidly fluctuating slip ratios, and therefore, coefficients of friction and traction forces. It was considered acceptable to linearize the sliding friction around a small yaw rate and leave the traction forces nonlinear.

3.3.1 Small Yaw Rate Approximation

There is a pivotal caveat to make when assuming that the yaw rate is small. Namely, that the yaw acceleration ($\ddot{\phi}$) may not be small. In fact, due to the behavior of friction, it is possibly for $\ddot{\phi}$ to be highly sporadic. This behavior will average out to small changes in $\dot{\phi}$ and a generally smooth ϕ trajectory.

In addition, the centripetal acceleration caused by this rotation ($\dot{\phi} \times \dot{Y}$) will be insignificant when compared to the frictional effects, unless the vehicle is being operated at high

velocities. Therefore, it may be assumed that the $\dot{\phi} \times \dot{Y} \to 0$, implying that $\ddot{X} \to 0$. This yields that equation (3.4) may be neglected.

3.3.2 Viscous Sliding Friction

At this time all that is required to simplify the friction is to impose a viscous sliding condition, such that, $\mu_i N_i \to b_i v_i$. Where b_i represents the coefficient of viscous friction at a certain location and in a certain direction, and v_i is the corresponding velocity ($\dot{Y}$, $\dot{\phi}$, etc.). These coefficients and their contribution to the dynamics may be solved for experimentally (as discussed in section 1.3).

3.3.3 Linearized System of Equations

After imposing the assumptions made in the previous section, transform the system of equations into the following set of mostly linear (with the exception of the traction forces) differential equations.

$$m\ddot{Y} = F_{T_1} + F_{T_2} - b_y\dot{Y} \tag{3.22}$$

$$J_c\ddot{\phi} = \frac{D}{2}\left[F_{T_1} - F_{T_2}\right] - b_\phi\dot{\phi} \tag{3.23}$$

$$J_i\dot{\omega} = \gamma_i u_i - b_{\omega,i}\omega_i - RF_{T_i} \tag{3.24}$$

It will be useful later to transform these equations into the Laplace domain under the assumption of relaxed initial conditions, in such a way to solve for the response of the translational velocity ($\dot{Y}$), wheel velocities ($\omega_{1,2}$) , and yaw angle (ϕ). This yield the following

system of equations and definitions.

$$\dot{Y}(s) = \frac{K_y}{\tau_y s + 1} \left[F_{T_1} + F_{T_2} \right] \tag{3.25}$$

$$\phi(s) = \frac{K_\phi}{s \left(\tau_\phi s + 1 \right)} \left[F_{T_1} - F_{T_2} \right] \tag{3.26}$$

$$\omega_i(s) = \frac{1}{\tau_{m_i} s + 1} \left[K_{m_i} U_i(s) - K_{T_i} F_{T_i}(s) \right] \tag{3.27}$$

K_{T_1}	Wheel Traction Gain for the right drive wheel
K_{T_2}	Wheel Traction Gain for the left drive wheel
K_{m_1}	Voltage/Torque Gain for the right drive wheel
τ_{m_1}	Time Constant of the right motor
K_{m_2}	Voltage/Torque Gain for the left drive wheel
τ_{m_2}	Time Constant of the left motor
K_y	Velocity Gain
τ_y	Time Constant of Velocity
K_ϕ	Yaw Gain
τ_ϕ	Time constant of yaw

Table 3.1: Definition of Parameters

Should this formulation be accurate the dynamics of this system may be determined through the knowledge of these 10 constants.

3.4 Linear Model

Finally, a completely linear model will be derived. This will include the linearization of slip ratios, wheel velocity estimation, and the traction forces. When linearizing these parameters several interesting things become evident. First, the translational velocity at each wheel becomes indistinguishable from the bulk vehicle velocity. Second, all states become an explicit function of the rotational velocity of the wheels and the bulk velocity. Finally, the operational point chosen for the linearization acts as an inverse gain on most all terms.

3.4.1 Linearized Velocity and Slip Estimation

When looking at the equation for the estimated velocity at each wheel, it is clear that it is a nonlinear function of the bulk velocity ($\dot{Y}$) and the yaw rate ($\dot{\phi}$). Linearize this equation by performing a 1^{st} order Taylor expansion around the small yaw rate condition ($\dot{\phi}^* = 0$).

$$V_i = \dot{Y}\left[1 + (-1)^{i+1}\frac{1}{2}sin\left(\dot{\phi}\right)\right]$$
$$V_i \approx V_i(\dot{\phi}^*) + \left.\frac{\partial V_i}{\partial \dot{\phi}}\right|_{\dot{\phi}^*}\left(\dot{\phi} - \dot{\phi}^*\right)$$
$$V_i \approx \dot{Y} + \dot{Y}\,(-1)^{i+1}\frac{1}{2}cos(0)\left(\dot{\phi}\right)$$

This yields a final equation that is once again a nonlinear equation. Using the assumption that $\dot{\phi}$ is very small, it is safe to assume that when compared to $\dot{Y}$, the product of $\dot{Y}$ and $\dot{\phi}$ will be negligible. Finally, this leaves only one term.

$$V_i = \dot{Y}, \qquad for\, i = 1, 2 \tag{3.28}$$

Now that there is a linear relationship for the wheel velocity it is possible to linearize the slip ratio, once again, using a 1^{st} order Taylor expansion. This time the equation is linearized around the operating point (ω^*, V^*), where $\omega^* R = V^*$ (equivalent to $\lambda^* = 0$). First consider the case where the vehicle is accelerating while the tire angle is zero.

$$\lambda = \frac{\omega R - V}{\omega R}$$

$$\lambda \approx \lambda(\omega^*, V^*) + \left.\frac{\partial \lambda}{\partial \omega}\right|_{(\omega^*, V^*)} (\omega - \omega^*) + \left.\frac{\partial \lambda}{\partial V}\right|_{(\omega^*, V^*)} (V - V^*)$$

$$\approx \left[\frac{R}{\omega^* R} + \frac{\omega * R - V^*}{(\omega^* R)^2}\right] (\omega - \omega^*) + \left[-\frac{1}{\omega^* R}\right] (V - V^*)$$

$$\lambda \approx \frac{(\omega R - V) - (\omega^* R - V^*)}{\omega^* R}$$

$$\lambda \approx \frac{\omega R - V}{\omega^* R} \tag{3.29}$$

This same process can be applied to the case where the vehicle is decelerating with the following result.

$$\lambda \approx \frac{\omega R - V}{V^*} \tag{3.30}$$

It should be clear that when the system is linearized around no slip, V^* will be equivalent to $\omega^* R$, yielding the same equation regardless of the vehicles behavior. When equation (3.28) is combined with equation (3.30) the following is the final slip approximation.

$$\lambda \approx \frac{\omega R - \dot{Y}}{\dot{Y}^*} \tag{3.31}$$

3.4.2 Linearized Traction Forces

At this point it is possible to perform one final linearization. This last step will combine the results above to determine a linear traction estimation for small slip ratios ($\lambda^* = 0$). Referring to equations (3.5) and (3.6),

$$\begin{aligned} F_T &= N \frac{2\mu_p \lambda_p \lambda}{\lambda_p^2 + \lambda^2} \\ F_T &\approx F_T(\lambda^*) + \left.\frac{\partial F_T}{\partial \lambda}\right|_{\lambda^*} (\lambda - \lambda^*) \\ &\approx 0 + N \left[\frac{2\mu_p \lambda_p}{\lambda_p^2 + (\lambda^*)^2} - \frac{2\mu_p \lambda_p \lambda^*}{\left(\lambda_p^2 + (\lambda^*)^2\right)^2} (2\lambda^*) \right] (\lambda) \end{aligned}$$

$$F_{T_i} \approx N_i \frac{2\mu_p \lambda}{\lambda_p} \tag{3.32}$$

3.4.3 State Space Model

Combining all of the results in this and the previous section, a state space model may be implemented. Note that it is a function of the operational velocity that the equations are

linearized around. A nonzero operating point must be chosen in order to maintain finite dynamics. Begin by defining the states of the system and their derivatives.

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} = \begin{bmatrix} \omega_1 \\ \omega_2 \\ \dot{Y} \\ \phi \\ \dot{\phi} \\ \lambda_1 \\ \lambda_2 \end{bmatrix} \quad \rightarrow \quad \begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \\ \dot{x_4} \\ \dot{x_5} \\ \dot{x_6} \\ \dot{x_7} \end{bmatrix} = \begin{bmatrix} \dot{\omega}_1 \\ \dot{\omega}_2 \\ \ddot{Y} \\ \dot{\phi} = x_5 \\ \ddot{\phi} \\ \dot{\lambda}_1 \\ \dot{\lambda}_2 \end{bmatrix} \tag{3.33}$$

Use these definitions to rewrite the equations of motion in terms of these states as a matrix equation where the inputs are the motor voltages and the outputs are the vehicle velocity and yaw angle.

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \\ \dot{x_4} \\ \dot{x_5} \\ \dot{x_6} \\ \dot{x_7} \end{bmatrix} = \begin{bmatrix} -\frac{1}{\tau_{m_1}} & 0 & 0 & 0 & 0 & \frac{-2K_{T_1}N_1\mu_p}{\lambda_p} & 0 \\ 0 & -\frac{1}{\tau_{m_2}} & 0 & 0 & 0 & 0 & \frac{-2K_{T_1}N_1\mu_p}{\lambda_p} \\ 0 & 0 & -\frac{1}{\tau_y} & 0 & 0 & \frac{2\mu_p N_1}{m\lambda_p} & \frac{2\mu_p N_2}{m\lambda_p} \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -\frac{1}{\tau_\phi} & \frac{K_\phi \mu_p N_1}{\lambda_p} & -\frac{K_\phi \mu_p N_2}{\lambda_p} \\ -\frac{K_{m_1}u_1^*}{(x_1^*)^2} & 0 & 0 & 0 & -\frac{1}{\tau_\phi} & \frac{\left(-K_{T_1}-\frac{1}{m}-K_\phi x_3^*\right)2\mu_p N_1 - K_{m_1}u_1^*\lambda_p}{x_1^* R\lambda_p} & \frac{\left(-\frac{1}{m}-K_\phi x_3^*\right)2\mu_p N_2}{x_1^* R\lambda_p} \\ 0 & -\frac{K_{m_2}u_2^*}{(x_2^*)^2} & 0 & 0 & -\frac{1}{\tau_\phi} & \frac{\left(-\frac{1}{m}-K_\phi x_3^*\right)2\mu_p N_1}{x_2^* R\lambda_p} & \frac{\left(-K_{T_2}-\frac{1}{m}-K_\phi x_3^*\right)2\mu_p N_2 - K_{m_2}u_2^*\lambda_p}{x_2^* R\lambda_p} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix}$$

$$+ \begin{bmatrix} K_{m_1} & 0 \\ 0 & K_{m_2} \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ \frac{K_{m_1}}{x_1^*} & 0 \\ 0 & \frac{K_{m_2}}{x_2^*} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \tag{3.34}$$

$$y = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} \tag{3.35}$$

Upon investigation it can be seen that this model is sensitive to the wheel speed. Should this system be linearized around small wheel speeds the slip dynamics become unstable. However at high wheel the slip dynamics become insignificant. In conclusion this model is only useful in the approximation of steady state behavior. When the system is allowed to transition between various conditions, like variations in traction for example, this model will no longer be valuable. Therefore, the remainder of this work will focus on the nonlinear and near linear models.

3.5 Model Validation

3.5.1 Motor and Traction Characterization

Begin by considering the motor and wheel subsystem. The initial stages of all locomotion in this system originate in this subsystem. As discussed previously, this is being simplified as a 1^{st} order linear equation.

$$\omega_i(s) = \frac{1}{\tau_{m_i} s + 1} \left[K_{m_i} U_i(s) - K_{T_i} F_{T_i} \right] \tag{3.36}$$

This equation shows that the output (the angular velocity of the wheel) is a function of two inputs (the voltage applied to the motor and the traction force). This makes the calculation of the system coefficients more complicated as the traction forces fluctuate over the course of the test. In order to minimize this difficultly it is most effective to first separate the free spin dynamics from the ground interaction and solve for the coefficients separately.

3.5.2 Free Spin Motor Dynamics

Consider the case where the traction forces are equal to zero for all time. This simplifies the equation to a Single Input Single Output (SISO) system with two unknown parameters relating only to the motor and gearbox.

$$\omega_i(s) = \frac{K_{m_i}}{\tau_{m_i} s + 1} U_i(s) \tag{3.37}$$

These coefficients may be estimated from the manufacture's specifications, however, this will not account for the variation in the individual motors or in the way that they are implemented. Therefore, it is beneficial to experimentally determine these values. The mathematical model of free spinning may be realized by placing the vehicle on a stand so that the rear wheels are held off the ground. For a known input signal and a measured output it is then possible to calculate the gain and time constant, where the gain is a function of the steady state magnitudes of the input and output (angular velocity and voltage), while the time constant is only a function of the systems transient response . In the interest of simplicity it is best to choose a simple input signal, in this case, a step input of approximately 9 Volts. The equation above has a well-defined analytic solution to a step input which may be found by taking the inverse Laplace transform and applying initial conditions. This experiment is

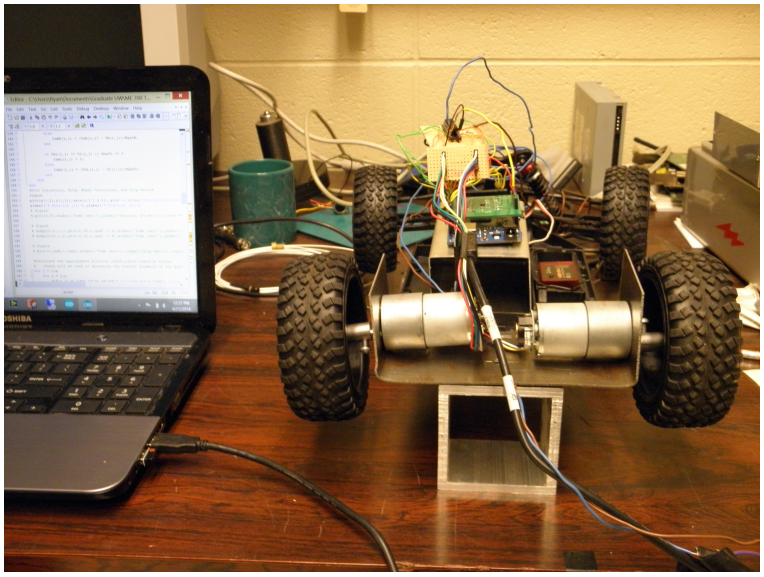


Figure 3.5: Motor Characterization Experiment

based on relaxed stationary initial states, yielding the following time domain representation.

$$\omega_i(t) = K_{m,i} u_i \left[1 - e^{\frac{-t}{\tau_{m,i}}}\right] \tag{3.38}$$

Begin by computing the motor gain (Km). This may be done by considering the steady state. If the limit of (3.38) is taken as t tends to infinity, it is clear that with a known steady state velocity (measured experimentally) and a known input (9 Volts) the gain may be found to be,

$$K_{m_i} = \frac{\omega_i(t \to \infty)}{u_i} \tag{3.39}$$

Now, consider the time constant. The following derivation demonstrates how the rise time may be used to compute the time constant.

$$\omega(t) = K_m u \left[1 - e^{\frac{-t}{\tau_m}}\right]$$

$$t_1 = 10\%(\omega_{ss}), \qquad t_2 = 90\%(\omega_{ss})$$

$$\frac{\omega(t)}{K_m u} = 1 - e^{\frac{-t}{\tau_m}}$$

$$@t = t_1 \rightarrow \qquad 0.1 = 1 - e^{\frac{-t_1}{\tau_m}}$$

$$0.9 = e^{\frac{-t_1}{\tau_m}} \rightarrow \boxed{t_1 = -\tau_m ln(0.9)}$$

$$@t = t_2 \rightarrow \qquad 0.9 = 1 - e^{\frac{-t_2}{\tau_m}}$$

$$0.1 = e^{\frac{-t_2}{\tau_m}} \rightarrow \boxed{t_2 = -\tau_m ln(0.1)}$$

$$T_R = t_2 - t_1 = \tau_m ln(9) \tag{3.40}$$

After performing this test over multiple trials, averaging the results, estimating the parameters (time constants and gains for each motor, results tabulated below). A simulation was performed to verify these results. (See Appendix D for simulation code.)

$K_{m_1}\left(\frac{rad}{Volt\,sec}\right)$	$K_{m_2}\left(\frac{rad}{Volt\,sec}\right)$	$\tau_{m_1}\left(\frac{1}{sec}\right)$	$\tau_{m_2}\left(\frac{1}{sec}\right)$
0.53	0.63	0.39	0.43

Table 3.2: Motor Characterization Constants

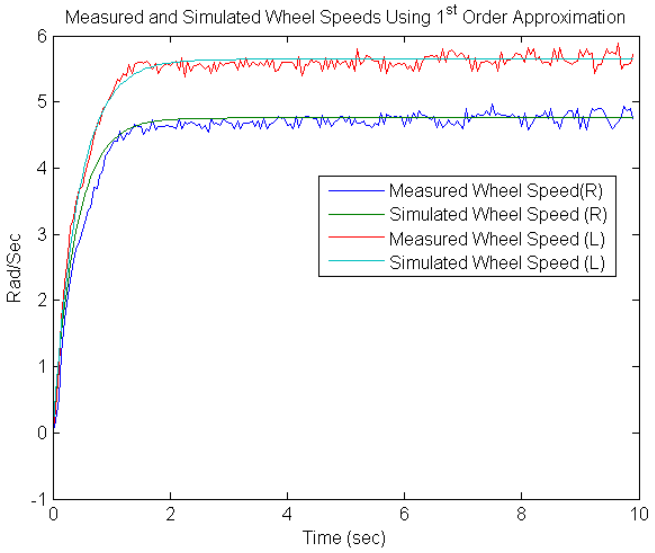


Figure 3.6: Motor Characterization Results and Simulation

3.5.3 Constrained Wheel and Traction Dynamics

Should the vehicle be removed from the cradle (nonzero traction forces), the previously determined motor characteristics may be used to help determine the remaining parameters. Under these new conditions the traction forces will be changing and must be calculated in order to determine the inputs to (3.36). As discussed in section (3.5.1), the traction force is a function of the slip ratio, and thus, a function of the wheel's rotational and linear velocities. The rotational velocity may be measured using the same method as above, however, the linear velocity must be estimated based on the yaw rate and velocity of the vehicle. Using

the accelerometer, the yaw rate and linear acceleration of the vehicle may be measured, and from this the rest may be computed.

With these equations and the information provided by the sensors, it will be possible to compute the slip ratios and traction forces. These will supplement the electrical input $(K_{m_i}\, u_i \, - \, K_{T_i}\, F_{T_i})$. Now that the inputs are known and the output measured, it is possible to determine the traction gains through an approximation of the steady state behavior.

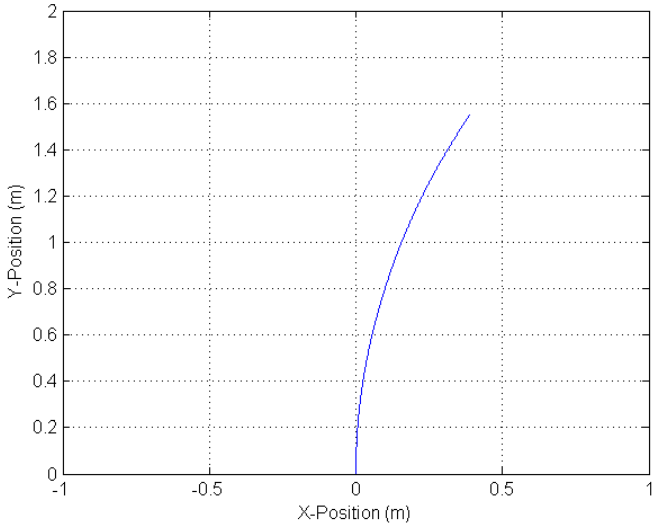


Figure 3.7: Trajectory of Vehicle to Open Loop Step Input

Using the same process derived previously the time constant may be estimated. Looking at the plot of the traction forces, it is clear that a quasi-steady state is reached. The average over this time domain $(2 \leq t \leq 5)$ will approximate the steady state force. After computing the steady state wheel velocity, the traction gains may be calculated by solving the equation below.

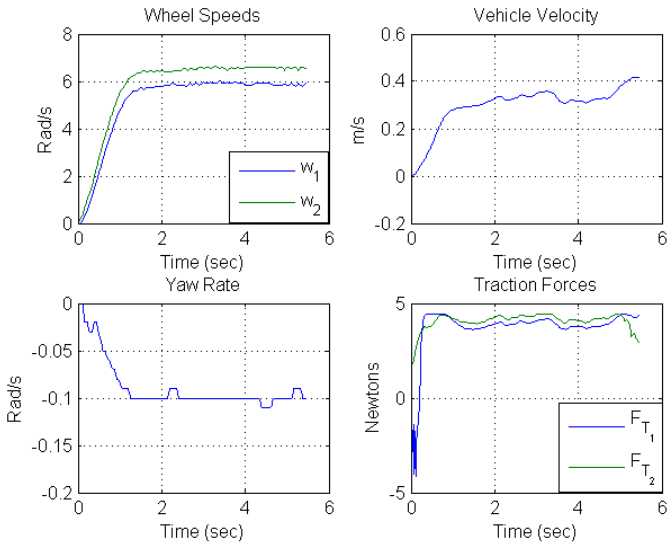


Figure 3.8: Vehicle Characterization Results

$$\omega_{ss_i} \approx K_{m_i} u_i - K_{T_i} F_{T_i}$$

Once these coefficients were calculated ($K_{T_1} = 0.03 \frac{rad}{Newton\, sec}$, $K_{T_2} = 0.10 \frac{rad}{Newton\, sec}$) a new set of simulations were conducted to verify the results.

3.5.4 Translation and Rotation Characterization

Now that the effects of the motors and traction forces on the wheel speed are known, it is possible to approximate the relationship between these traction forces and translation of the vehicle. The dynamics of the near linear model give a very straight forward relationship (equations 3.22 and 3.25). In this case the translation gain may be estimated based on the

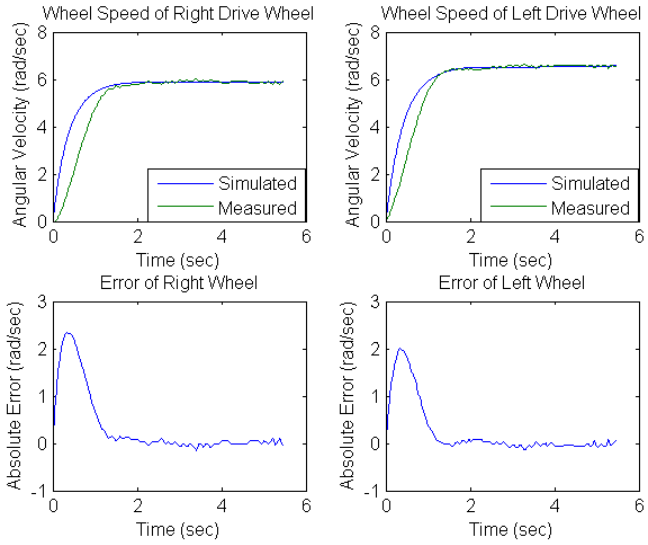


Figure 3.9: Simulation of Traction Effects on Wheel Speeds

mass and the time constant, as approximated from the rise time.

$$K_y = \frac{\tau_y}{m} \tag{3.41}$$

By similar reasoning (equations 3.23 and 3.26) the yaw gain and time constants may be computed from the rise time, moment of inertia and wheel base.

$$K_\phi = \frac{\tau_\phi D}{2J_c} \tag{3.42}$$

These relationships and the experiments described above give rise to the following coefficients.

$K_y \left(\frac{m}{Newton\, sec}\right)$	$\tau_y \left(\frac{1}{sec}\right)$	$K_\phi \left(\frac{rad}{Newton\, meter\, sec}\right)$	$\tau_\phi \left(\frac{1}{sec}\right)$
0.44	0.50	0.19	0.03

Table 3.3: Vehicle Characterization Constants

3.6 Summary

This chapter has investigated the development of three different models with varying levels of complexity: the nonlinear model, which accounts for all of the significant dynamics; the near linear model, which simplifies the bulk vehicle dynamics but retains the nonlinear behavior of the traction, slip, and velocity estimation; and finally, a pure linear model, which treats all dynamics as small perturbations from the operation conditions (in this case small slip, small yaw rate, uniform wheel velocities, and zero tire angle). This model was determined to be suitable only for steady state analysis as its dependence on the point of linearization was significant. And finally, the necessary coefficients for the near linear model were determined experimentally and will be used extensively in the simulation chapter.

Chapter 4

CONTROLLER DESIGN AND IMPLEMENTATION

4.1 Introduction

As discussed in the introductory chapter this work is utilizing control systems to replace previously mechanical components, therefore it is of great importance that the controller implemented is capable of maintaining vehicle stability while meeting performance criteria. This chapter will discuss the design and experimental validation of a control system capable of balancing motor torques with the objectives of eliminating drift, moving to a desired velocity and maintaining it, while monitoring for and correcting excessive slip. There are a large number of possible controller design techniques and methodologies capable of achieving these goals. This work will investigate only a few controllers and how they may be implemented.

- Output Feedback
- Traction Control
- Parametric Control Weighting

Each set of control laws will be implemented using multiple controller gains in the real system and the performance measured over a number of Performance Indicies (PIs). In addition it will discuss the measurement and observation of the outputs and states from the given sensor data.

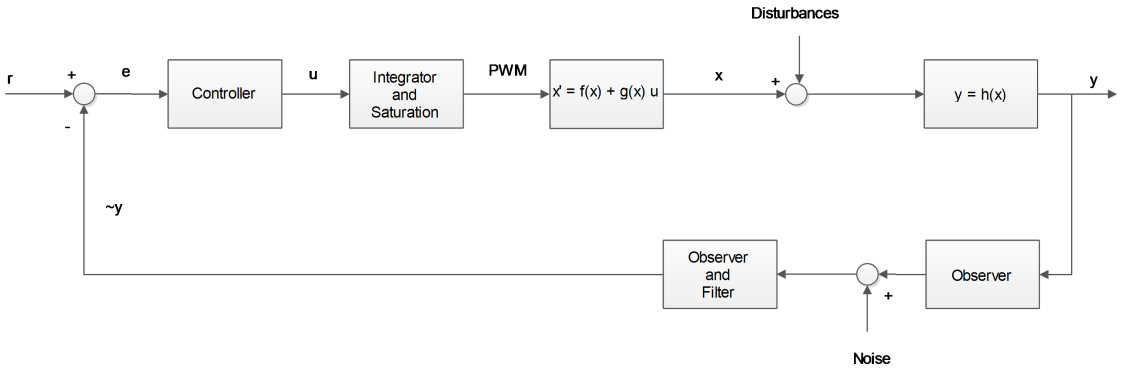


Figure 4.1: Output Feedback Block Diagram

4.2 Choice of Performance Index

When attempting to classify a controller as preferable to another, the standards by which they are measured must be considered. This work looks at several different indexes and how they are related to the choice of controller gains. Of the numerous possible choices of PIs the rise time, total energy input, maximum yaw angle, and maximum deviation are considered. The rise time gives an approximation of the aggressiveness of the controller. It should be small, correlating to a fast change of steady state position, however there is a trade off. A minimal rise time corresponds to large inputs and large overshoots. While overshoot is manageable in small quantities, large inputs are a more important concern. One of the primary goals in developing IWD vehicles is energy efficiency, yielding smaller inputs operating at higher mechanical inefficiencies. In addition, if the reference trajectory necessitates that the inputs are placed in saturation, there will be no room for reaction in the event of a disturbance. Therefore, it is important to choose a rise time that does not necessitate either of these conditions.

The total energy input is a scalar quantity generated by integrating the absolute value (to account for positive and negative motor directions) of the inputs and scaling it by the

amount of time that the experiment was run, to account for the variability in run times. This will give the an average energy cost of operation for a control method.

The maximum yaw angle and maximum deviation measure the greatest amount of angular and lateral deviation that is tolerated by the control method. All controllers discussed will seek to minimize this. However, this will require a more aggressive controller, yielding higher energy costs and overshoots.

4.3 Output Feedback

Output feedback is based on the principle that the behavior of the system (the output) is measured and compared to a desired output (the reference). The resulting error is then used as an input to the control system. This is preferable to state feedback in that it will require a significantly smaller amount of data to be processed in each iteration. In the event that more computational power is available (for instance, in a full size vehicle) it may be preferable to use state feedback to generate a more complete picture of the system's behavior in real time.

4.3.1 Output Observer

The dynamics of this system produce outputs of longitudinal velocity ($\dot{Y}$) and a yaw angle (ϕ). However, the accelerometer is only capably of measuring the linear acceleration and the yaw rate. Mathematically this is a simple problem to solve through integration. However, numerical integration is inherently unstable, meaning that a small nonzero acceleration offset, or even noise with a nonzero average can be integrated to a significantly different value than that of the actual velocity. This holds true for the yaw rate as well. The outputs of these computations are the basis for the entire control algorithm; therefore, computing them with consistent precision is crucial.

The observer is designed to rectify this by focusing on precision and robustness over speed. It takes a series of ten measurements in rapid succession, averaging them, and then applies a low pass filter with a cutoff at the sampling frequency. The output of this observer is seen to be is more consistent; however, computational time requires that the controller sample slower than it otherwise could.

4.3.2 Controller Methodology

This output controller is based on Proportional, Integral, and Derivative Control (PID). The controller will use a linear combination of the current error as well as the derivative and integral of the error.

$$u(t) = K_p e(t) + K_D \frac{de}{dt} + K_I \int_0^t e(\tau)\, d\tau \tag{4.1}$$

$$U(s) = \frac{s^2 K_D + s K_P + K_I}{s} E(s) \tag{4.2}$$

Where, u is the controller's input to the system, e is the error measured, and K_P, K_D, and K_I are the proportional, derivative, and integral gains respectively. Equation (4.1) expresses the input as a continuous function of time, where (4.2) is the corresponding transfer function. However, these control laws will be implemented in the digital domain (discrete time). Therefore, this equation may be rewritten with the follow discrete time transfer function.

$$U(z) = \left[K_P + K_D \frac{1}{T_s} \frac{z-1}{z} + K_I \frac{T_s}{2} \frac{z+1}{z-1} \right] E(z) \tag{4.3}$$

where T_s is the sampling period. The integral is approximated by the Trapezoidal method and the derivative is approximated by the backward Euler method. Note that due to the nature of sampling a continuous time signal the dynamics of the controller will change therefore, to get equivalent results the coefficients chosen in (4.2) will most likely not be equal to the coefficients in (4.3). This transfer function may be transformed into a sampled time equation that may be directly applied to the micro controller.

$$u(k) = u(k-1) + \left[K_P + \frac{K_D}{T_s} + \frac{K_I T_s}{2}\right] e(k) + \left[\frac{K_I T_s}{2} - \frac{2K_D}{T_s} - K_p\right] e(k-1) + \left[\frac{K_D}{T_s}\right] e(k-2) \tag{4.4}$$

where k is an integer referring to the number of the particular sample being considered. Therefore, the time at which the k^{th} sample is taken is kT_s. Note that (4.4) is a function of the previous input as well as the current error and the error at the two previous samples. This requires that for each PID law in use, 4 pieces of information must be stored for each computation.

Now that the structure of the control law has been established it is time to consider the mechanics of implementing it. There are two important features. First, consider the case where the vehicle moves to a steady state (velocity and yaw angle are constant) at the reference value (error is zero). With zero error the proportional control is removed. In addition, in steady state, the derivative control will also be zero. This leaves only the integral control. As the vehicle moves to steady state the integral will attenuate input until it reaches a constant value at zero error. This will ensure that the vehicle reaches the reference and remains there in the absence of disturbances. Second, consider the derivative control. Previously the noise in the measurement signal was discussed and the trouble that it leads to. This noise coupled with the numerical errors of differentiation is sufficient to generate a sporadic signal

that may easily exceed input saturation. In order to avoid this the derivative gain is set to zero. This yields the final form.

$$u(k) = u(k-1) + \left[K_p + \frac{K_I T_s}{2}\right] e(k) + \left[\frac{K_I T_s}{2} - K_P\right] e(k-1) \tag{4.5}$$

One last step is required to produce the structure of the output feedback controller. Equation (4.5) produces a single voltage signal from a single error signal, whereas this system will have two inputs and two errors. Therefore, an additional controller (of the same form) should be implemented in parallel. The first controller will accept a velocity error signal while the second accepts a yaw error signal where the total input to the system will be the summation of the two. While each motor will react the same to errors in velocity, it is required that they respond inversely to yaw errors. Consider a positive yaw error (the vehicle has rotated to the right). This should correspond to a positive control signal to the right wheel (increasing power) while the left wheel should receive a negative signal (reducing power) in order to correct from both sides. This rational produces the following control signals.

$$u_1 = u_{\dot{Y}_1} + u_{\phi_1} \tag{4.6}$$

$$u_2 = u_{\dot{Y}_2} - u_{\phi_2} \tag{4.7}$$

Equations (4.6) and (4.7) show a generalized case where the right and left motor are allowed to react differently to the same stimuli. Unless the vehicle is very poorly balanced

or the motors have significantly different characteristics, it is not necessary to imbalance the controllers. This work will consider the case where the controllers are balanced ($u_{\dot{Y}_1} = u_{\dot{Y}_2} = u_{\dot{Y}}$, and $u_{\phi_1} = u_{\phi_2} = u_{\phi}$).

4.4 Traction Control

Now that the drift and velocity control have been structured, it is time to consider the traction control. Below are a few of the possible methods to accomplish this.

1. Slip Rejection. The most simple traction control method. This moves to minimize all slips (For simplicity this is the method that this work will investigate).

2. State Slip Control. This method incorporates the slip ratio of each wheel under consideration and treats it as an additional state in the model. These may now be controlled to a desired value using standard control principles.

3. Optimal Slip Control [3]. Similar to #2 this method places the desired slip ratio at a desired value. However, it may also incorporate changing conditions and objectives to vary λ accordingly.

4. Gradient Slip Control. This method measures the rate of change of traction. Large positive or negative values indicate a sudden slip. While this method is less sensitive to sensor noise it only reacts to sudden changes, as opposed to the continuous fluctuation of traction.

The primary objective of this controller is to maintain slip ratios within some desired threshold. Recall from Figure 3.3 that the coefficient of friction is related to the amount of wheel slip. Therefore, when large slips are present, the corresponding traction forces will be small, reducing the control of the vehicle. It should be clear that this will generate potentially dangerous situations that should be prevented when possible and minimized when necessary.

Therefore, it is not uncommon for traction controllers to be highly aggressive. This work will consider the slip at each wheel independently. If only one wheel losses traction, that wheel will move to regain traction while its counter part moves to maintain bulk stability.

Consider the two cases under which large slip ratios will occur. First, if the wheel is spinning significantly faster than the vehicle is moving, this could be caused by a low coefficient of friction (ice or water on the road, etc.) or when power is applied suddenly and in large quantities (burning out, etc.). Regardless of how it is generated, this will correspond to a large positive slip ratio. And secondly, if a reverse or braking torque is suddenly applied while the vehicle is moving forward (emergency stop, etc.). This will cause a large negative slip ratio. Following this rational it can be shown that the direction the controller acts should be opposed to the slip ratio; meaning that for large positive slips the controller will decrease power to the affected wheel and, conversely, for large negative slips power should be increased. While there are many different ways to implement traction control, this work will utilize a simple proportional controller of the form:

$$u_\lambda = -K_\lambda \lambda \tag{4.8}$$

where K_λ must be chosen sufficiently large to generate a significant input, thereby reducing slip rapidly.

4.5 Parametric Controller Weighting

It is worth noting that not all slip is undesirable. In fact for λ close to λ_p maximum traction is achieved. This means that it is not helpful to apply a reducing input at all slips. Instead, it should only be applied at slips exceeding the threshold. This raises two questions. First,

what is a suitable threshold (θ)? And secondly, how should the control laws transition between drift and velocity focused to traction focused?

4.5.1 Slip Threshold

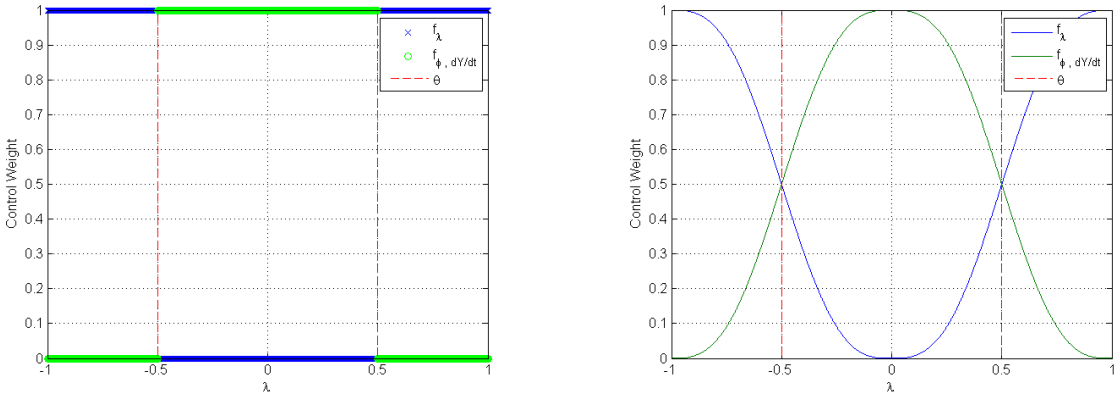
The choice of the slip threshold will determine to what extend the controller is allowed to ignore the wheel slip. This upper bound in magnitude of slip should be placed above the peak slip, but not so high as to allow vehicle control to degrade significantly. Once again referring to Figure 3.3 it can be seen that λ, should be greater than 0.2 (λ_p for these traction conditions). Looking at the rate at which μ drops off with λ, a threshold of 0.5 may be chosen.

4.5.2 Transition Methods

When discussing the transition between control methods, it is helpful to describe the overall control signal as the linear combination of the PI controller described previously and the traction controller.

$$u_{total} = f_{\lambda}(\lambda)u_{\lambda} + f_{\phi,\dot{Y}}(\lambda)u_{PI} \tag{4.9}$$

where f_{λ} and $f_{\phi,\dot{Y}}$ are the weighting of each controller. There are two distinct methods available to transition between a slip oriented control and a drift/velocity oriented control. First, there is a discrete transition. This is a binary decision where the weighting functions are set to 1 or 0 depending on the value of λ.



(a) Discrete

(b) Continuous

Figure 4.2: Slip Dependent Controller Weighting

$$f_\lambda = \begin{cases} 0 & \text{if} |\lambda| \leq \theta \\ 1 & \text{if} |\lambda| > \theta \end{cases} \tag{4.10}$$

$$f_{\phi,\dot{Y}} = \begin{cases} 1 & \text{if} |\lambda| \leq \theta \\ 0 & \text{if} |\lambda| > \theta \end{cases} \tag{4.11}$$

The second approach is to define f_λ and $f_{\phi,\dot{Y}}$ as continuous functions of λ that will gradually transition as the slip ratio increases. There are a large number of functions that fit this criteria and depending on the performance criteria and region of desired operation they may be tuned accordingly. This work will use a simple shifted harmonic to represent f_λ; this will allow $f_{\phi,\dot{Y}}$ to be defined in terms of f_λ.

$$f_\lambda = |\lambda| - \frac{1}{2\pi} sin\left(2\pi|\lambda|\right) \tag{4.12}$$

$$f_{\phi,\dot{Y}} = 1 - f_\lambda \tag{4.13}$$

4.6 Controller Validation

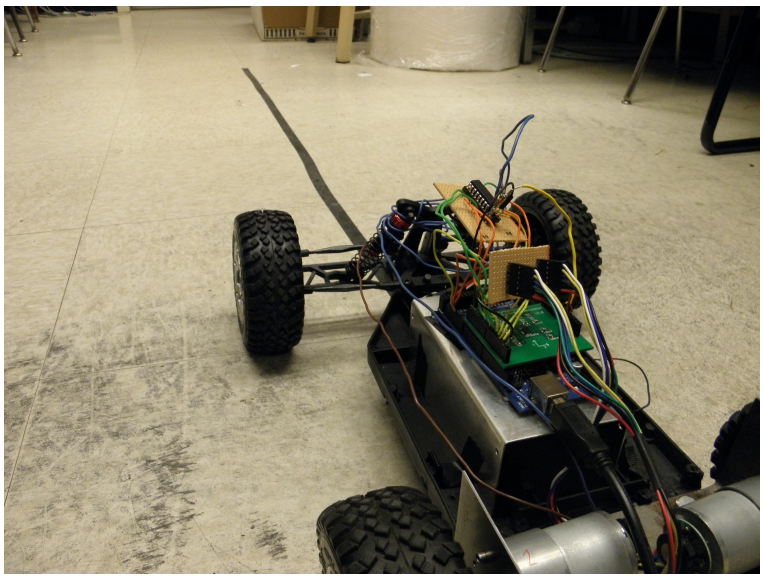


Figure 4.3: Experimental Controller Implementation

The proposed PI controller described above was tested using a set of 10 different controller gains, investigating connections between the proportional and integral control and how these effect the performance of the vehicle. Multiple variations were used where either the drift control or the velocity control was allowed to dominate and likewise the proportional and integral control. Given the desired gains, sampling methodology (as discussed previously), and the desired output, an Arduino program was written to implement the digital controller

(appendix ref). This program was run on board while exporting the measurement results and the calculated inputs to a computer via serial connection for post processing. As in Chapter 3 the vehicle was powered by a stationary DC power supply operating at 11 Volts with 1 Amp peaks. The vehicle was then placed on a test path, so drift and final position were more easily measured. When examining Table 4.1, [1] it is clear that when the controller acts aggressively the energy consumption is significantly higher. Also, in places where the velocity gains are significantly higher than the yaw gains, the maximum yaw angle increases dramatically, in addition to the energy consumption and maximum drift. However, in cases where the vehicle is allowed to approach the reference velocity slowly, the yaw angle is easier to control. However, the rise time is negativity affected.

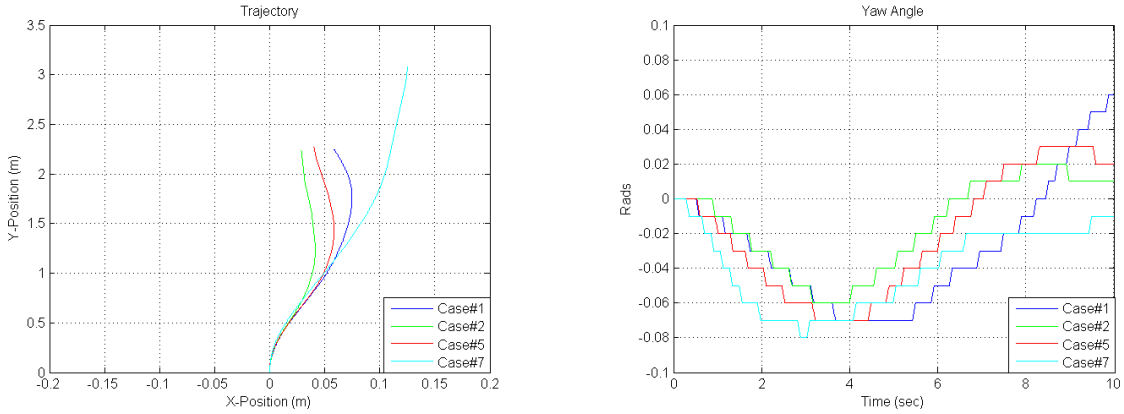
Case#	$K_{P_{\dot{Y}}}$	$K_{I_{\dot{Y}}}$	$K_{P_{\phi}}$	$K_{I_{\phi}}$	ϕ_{max} (rad)	X_{max} (m)	T_R (sec)	$\bar{U}$ (J) [2]
#1	10	10	10	10	0.07	0.07	9.45	13.41
#2	20	10	20	10	0.06	0.04	9.08	14.70
#3	50	10	20	10	0.06	0.06	8.35	16.03
#4	20	20	20	20	0.08	0.07	9.60	16.96
#5	30	10	20	10	0.07	0.06	9.41	14.50
#6	40	10	20	10	0.06	0.05	9.31	15.41
#7	30	30	20	5	0.08	0.12	9.65	16.62
#8	30	5	30	5	0.04	0.06	7.77	16.63
#9	20	15	20	15	0.05	0.05	8.24	16.25
#10	30	15	30	15	0.05	0.05	7.57	17.81

Table 4.1: Experimental Results Using PI Control

[1] Data analyzed using Matlab® file "FeedbackControllerAnalysis_rev2.m" Appendix D

[2] Where, X_{max} is the maximum lateral deviation, T_R is the recovery time, and $\bar{U}$ is the time normalized energy consumption

It is clear that a balance must be reached to achieve the best possible performance. Enhanced trajectories and plots are shown for cases #1, #2, #5, and #7, while a full scale representation of the trajectories are shown for all cases. It should be noted that case #7 deviates comparatively far away from the desired trajectory. This is due to the small value of K_{I_ϕ}. In contrast it was found that yaw integral gains larger than about 30 would produce excessive oscillations. While they would eventually die out, the recovery time was well outside of the design criteria.



(a) Enhanced Plot of Trajectories

(b) Yaw Angle Over Time

Figure 4.4: Experimental Results Using PI Control (Part 1)

4.7 Summary

This section discussed the methodology behind the control law design procedure and how it was experimentally tested. It has demonstrated the importance of the the interrelated nature of the performance indexes and how they must be balanced to achieve the desired behavior, specifically, the importance of stability and avoiding input saturation, while minimizing the

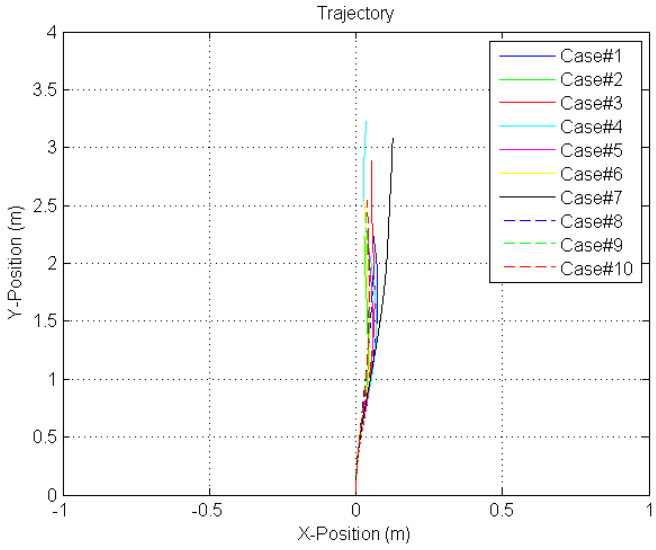


Figure 4.5: Experimental Results Using PI Control (Part 2)

yaw angle, the time that it is allowed to deviate and the energy consumed. It is possible that feedback alone may not be sufficient to accomplish these objectives. It is also possible that more sophisticated feedback criteria should be implemented, designed specifically for the optimization of the vehicle performance.

Chapter 5

SIMULATION AND RESULTS

5.1 Introduction

The three previous chapters have demonstrated the derivation of the system model and controller, as well as the experimental validation of these methods. This chapter will focus on the tuning of the model and the verification of its behavior in the open and closed loop. It will discuss how these models may be used to predict several situations that were not practical to verify experimentally at this time, namely the situation where traction is suddenly lost. Using both of the methods developed previously, the simulations will be run using the nonlinear model as it was found to be more accurate than the other models.

5.2 Software Packages Employed

The simulations discussed in this section were performed using Matlab Simulink (footnote with version: R2012a student edition) using two basic ordinary differential equation (ODE) packages, ode45, and ode23s. ode45 is based on the Dormand-Prince method for numerically solving explicit ODEs, while ode23s is based on the Modified Rosenbrock method for solving systems of stiff ODEs which may have multiple characteristic time scales. For the standard linear model ode45 is sufficient, however the nonlinear model runs better when using ode23s (see Appendix D for simulation code).

5.3 Additional Approximations

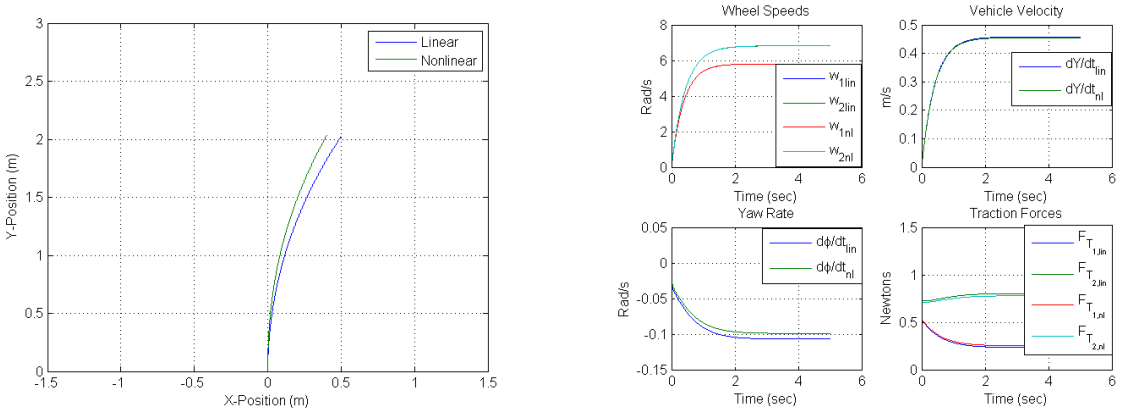
Referring to the nonlinear dynamic model, it can be seen that many of the frictional forces are developed instantaneously at nonzero velocities. This becomes an issue when considering any numerical method. Small numerical errors, especially in the vicinity of zero, may drastically change the behavior of the system. In an attempt to avoid these difficulties all sliding friction is approximated as viscous with saturation at the theoretical limits of sliding friction. This will retain the lateral dynamics as well as the nonlinear behavior in traction and velocity estimation.

5.4 Model Verification

First consider the experiments performed in Chapter 2. These provided data describing the vehicle trajectory as well as the individual wheel speed. This information will provide a sufficient comparison to determine the accuracy of the models.

5.4.1 Open Loop Comparison

Begin by comparing the open loop behavior of the same nature as in the experiments described above, namely a step input of 11 Volts to both motors. Through the comparison of figures 5.1 and 5.2 it is apparent that both models generally fit the data. The nonlinear model is slightly better partly due to the nonlinear slip and velocity estimation, and partly due to the inclusion of the lateral dynamics. Note that while the vehicle velocity is in the same range as the simulated, it fluctuates more extensively. This, coupled with fluctuations in the yaw rate, produces sporadic slip ratios, which leads to the large differences in the traction forces found by modeling, than those estimated from sensor data. However, because the vehicle trajectories are as similar as they are, it is a safe assumption that the nonlinear model is a reasonable predictor of bulk vehicle behavior.



(a) Linear and Nonlinear Trajectories

(b) State Behaviors

Figure 5.1: Simulation of Open Loop Response

5.4.2 Model Sensitivity

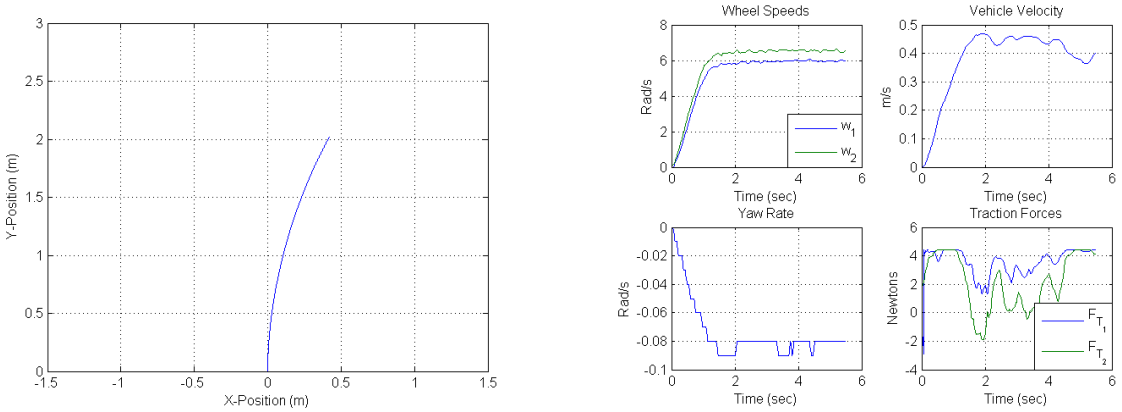
With the full dynamic model validated, an important consideration is robustness. If the simulation is very sensitive to small changes in some of the parameters (mass, moment of inertia, coefficient of friction, etc.) it becomes extremely important that the model is known exactly, which is not practical in reality. The following five perturbations will be considered, where each one acts independently of the rest:

1. Increase the vehicle mass by 10%.

2. Increase the vehicle moment of inertia by 10%.

3. Increase the viscous friction in the Y-direction (b_y) by 5%.

4. Decrease the traction gain for the left tire (K_{T_2}) by 10%.



(a) Experimental Trajectory

(b) State Behaviors

Figure 5.2: Open Loop Response

5. Decrease the viscous friction in the lateral direction (b_x) by 5%.

The trajectories generated by each perturbation is compared to the nominal behavior of the nonlinear model. Figure 5.3 demonstrates that any deviations between the nominal simulation and the 5 perturbations are very small. To better quantify these discrepancies the table below compares the final coordinates and yaw angle.

In table 5.1 it can be seen that no perturbation is more than 1cm away from the nominal simulation and no orientation is off by more than 7.5×10^{-4} rads. Note that just because the system behaves consistently under small variations in five of the parameters, does not guarantee that large changes can be absorbed well, especially in the slip dynamics which have the most significant nonlinear effects.

	Nominal	#1	#2	#3	#4	#5
X(m)	0.4023	0.3995	0.4020	0.3972	0.4100	0.4041
Y(m)	2.0332	2.0327	2.0333	2.0280	2.0383	2.0389
ϕ (rad)	-0.4496	-0.4469	-0.4495	-0.4450	-0.4569	-0.4537

Table 5.1: Final Position and Orientation in Perturbed Dynamics

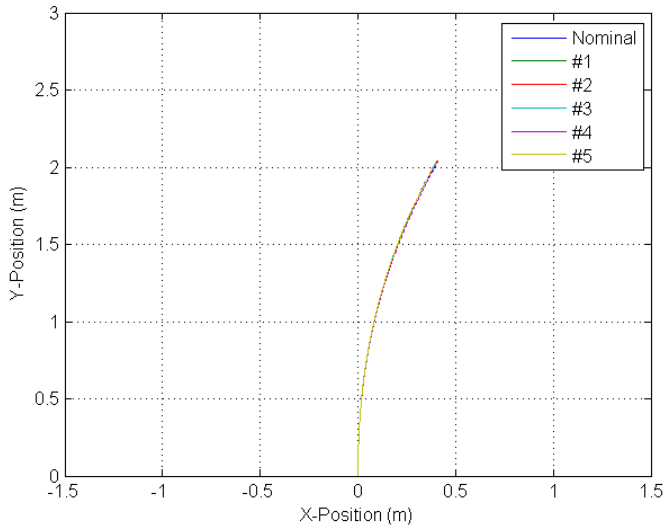


Figure 5.3: Nonlinear Simulations of Perturbed System

5.5 Traction Control Implementation

The final step in completing the vehicle's control system is the addition of traction control. As discussed previously, this will be used to ensure the stability of the vehicle through a variety of maneuvers. Using the structure outlined in equations (4.8) and (4.9), an augmenting controller may be placed in parallel with the existing feedback controller where the relative contribution of each controller (f_λ,$f_{\phi,\dot{Y}}$) is determined by the slip ratio. Both of the methods

for controller transition outlined in Chapter 4 will be implemented in this chapter and the performance measured.

5.5.1 Discrete Controller Weighting

Mathematically, the implementation of a discrete controller weighting function is straight forward. From a controls perspective it is slightly more involved. It requires the construction of a supervisory controller that monitors the slip ratio at each wheel and switches the controlling method based on the slip ratios detected. When one of the wheels exceeds the slip threshold, both wheels will react accordingly (both wheels act to avoid generating additional yaw). Because this controller will react the instant slip exceeds the threshold (θ), and is inactive the remainder of the time, the closed loop system becomes extremely stiff. If the control gain is not chosen sufficiently large, such that λ is moved well below the threshold, it is possible to produce chattering around the $\lambda = \theta$ point where the transition occurs. To avoid this, K_{λ_d} is chosen to be 2. Therefore, for each successive sampling instant where $\lambda \geq \theta$, the applied voltage to each motor will decrease by 2λ Volts (no smaller than an 18% reduction).

5.5.2 Continuous Controller Weighting

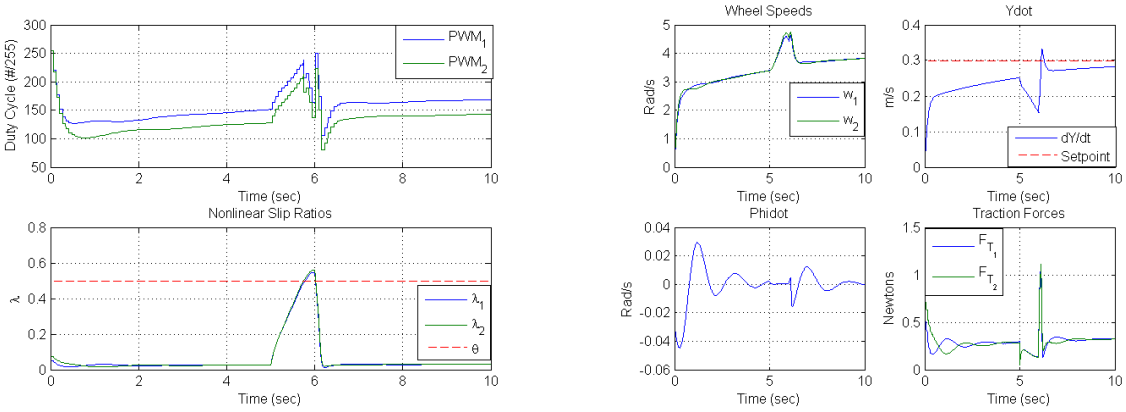
The technique implemented above makes use of a binary decision performed by the supervisory controller. This is sufficient for systems that only require stability. However, should a continuous weight function be used, the traction control could act at all times (to some extent) passively tuning the performance of the motors. This would provide smooth transitions between small and large slip situations without large spikes in inputs. A continuous weight function provides an easy stepping stone to implement a more sophisticated traction control that seeks to maintain a desired λ to optimize performance [3]. This will not be

addressed in this work.

5.6 Fluctuating Traction Conditions

Figure 5.4 illustrates the behavior of a binary traction controller as applied to the #6 PI feedback controller (recall table 4.1). In this case the system is given the same set points as before ($\dot{Y} = 0.3^{m}/_{s}$, $\phi = 0$), however, beginning at $t = 5$sec and ending at $t = 6$sec the peak coefficient of friction (μ_p) is reduced from 0.8 to 0.15. This number was chosen as it is the lower limit for rubber on wet cement, simulating driving through a patch of water on the verge of hydroplaning. Note that the traction controller does not activate until over half way through the slip, at which point λ is kept from growing and begins to gradually decrease. It is also important to note that from the beginning of the slip, the velocity of the vehicle decreases. However, there is a lag between this and the time that a slip is detected. Therefore, the controller ramps up the control signal to the motors until it is eventually cut off by the traction control law.

Using the weight functions defined in equations (4.12) and (4.13), along with the same feedback controller and slip situation, the following results are obtained. It can be seen that the shock to the input is drastically reduced. While the discrete method generated a PWM jump of approximately 150, this produced a jump only a third as large, all the while reducing the maximum slip (0.44 vs 0.56) and conserving more energy (12.3J vs 12.5J). In the systems response to shocks and large slips the continuous certainly seems to be superior, however, because the continuous slip control is constantly active the rise time of the system will be lengthened. Once again this down side may be overcome by treating the slip as a state or output and apply feedback to manipulate its behavior.



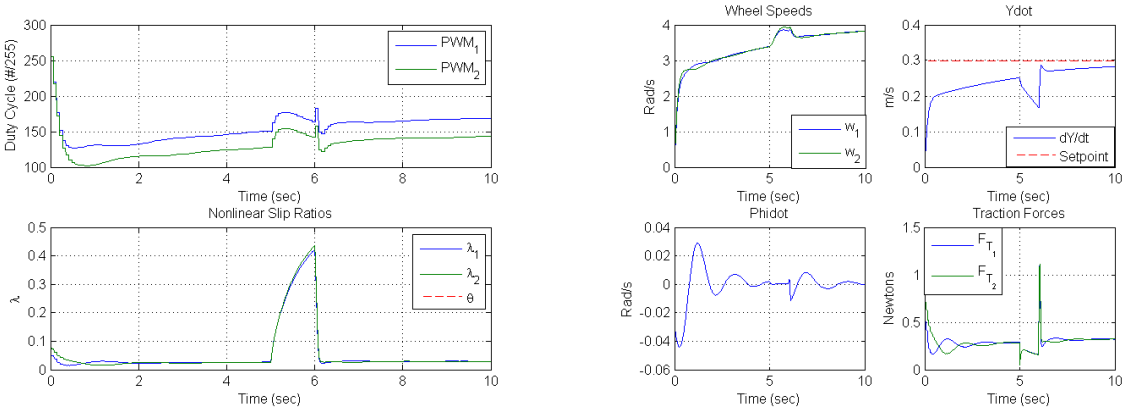
(a) Slip and Controller Input

(b) State Behaviors

Figure 5.4: Discrete Traction Response to a Sudden Slip

5.7 Summary

This chapter has shown the ability to predict the vehicle behavior in scenarios which are not yet feasible to verify experimentally. It has demonstrated the pros and cons of each traction control algorithm, namely that while the continuous traction controller provides better stability for a given set of circumstances, it reduces the effectiveness of the velocity control, increasing the rise time. Finally, though the discrete traction controller gives rise to larger inputs and sudden shocks to the system behavior, it is less computationally intensive. In the case where the processing power of the controller is extremely limited it would be an effective way to preserve stability with minimal impact on the sampling period.



(a) Slip and Controller Input

(b) State Behaviors

Figure 5.5: Continuous Traction Response to a Sudden Slip

Chapter 6

CONCLUSIONS

6.1 Summary of Work

This thesis has seen the development of several tools that could be of value in the next iteration of IWD vehicle control. It has completed a functional test bed capable of independent wheel and direction control. In addition, it has seen the completion of multiple dynamic models with varying levels of complexity, as well as an output feedback controller capable of velocity matching, drift rejection, and traction control. It has demonstrated that the quality of the control system will provide a new bound on the overall efficiency of the vehicle.

6.2 Future Work

6.2.1 Continuation of Current Work

As this project comes to an end, several questions are still in need of investigation and over the course of this work, new questions have come to light. As this work continues onward, there are opportunities to increase the complexity of this IWD. First, integrating the control of the front tire angle. Also, it may be possible to redesign this vehicle as a 4IWD, requiring a more complicated model and controller as well as an additional traction controller. Next, the rear suspension system could be reintroduced, including a more adaptable motor mounting system. And finally, it may be possible to implement a feedforward controller with the feedback to reduce the recovery time for the drift rejection. These improvements in effectiveness would ultimately be used to improve energy efficiency.

6.2.2 Applications to Full Scale Vehicles

If this work was to be applied to a full scale vehicle a number of things should be changed. First, a more intimate, and possibly higher order, model of the vehicle should be addressed. It seems reasonable that additional computation power would be available, therefore, online optimization should be used to choose desired profiles in real time to accomplish some overall objective. With this additional computer power, it would be feasible to also employ methods such as Kalman filtering to better estimate system behavior in real time. And finally, a more sophisticated slip control should be applied in an effort to maximize performance and estimate ground behavior to improve stability in real time.

6.3 Final Statements

Over the course of this project it has become clear that the field of IWD vehicles has a large number of new problems in the field of control systems. It represents a possibility to motivate a new wave of innovation in the automotive industry while attempting to maximize the potential of current electric vehicle technology.

BIBLIOGRAPHY

[1] Arduino. Mega 2560. http://arduino.cc/en/Main/arduinoBoardMega.

[2] Carlos Canudas de Wit and Panagiotis Tsiotras. Dynamic tire friction models for vehicle traction control. In *Proceedings of the 38th IEEE Conference on Decision and Control*.

[3] Shanbao Wang Gaugong Yin and Xianjian Jin. Optimal slip ratio based fuzy control of acceleration slip regulation for four-wheel independent driving electric vehicles. *Hindawi*, 2013.

[4] Tor A. Johansen Thor I. Fossen Jens C. Kalkkuhl Havard Grip, Lars Imsland and Avshalom Suissa. Nonlinear vehicle side-slip estimation with friction adaption. *automatica*, 2008.

[5] Yoichi Hori. Future vehicle driven by electricity and control–research on four-wheel-motored uot electric march ii. *IEEE*, 2004.

[6] Yizhai Zhang Jingliang Li and Jingang Yi. A hybrid physical-dynamic tire/road friction model. *ASME*, 2013.

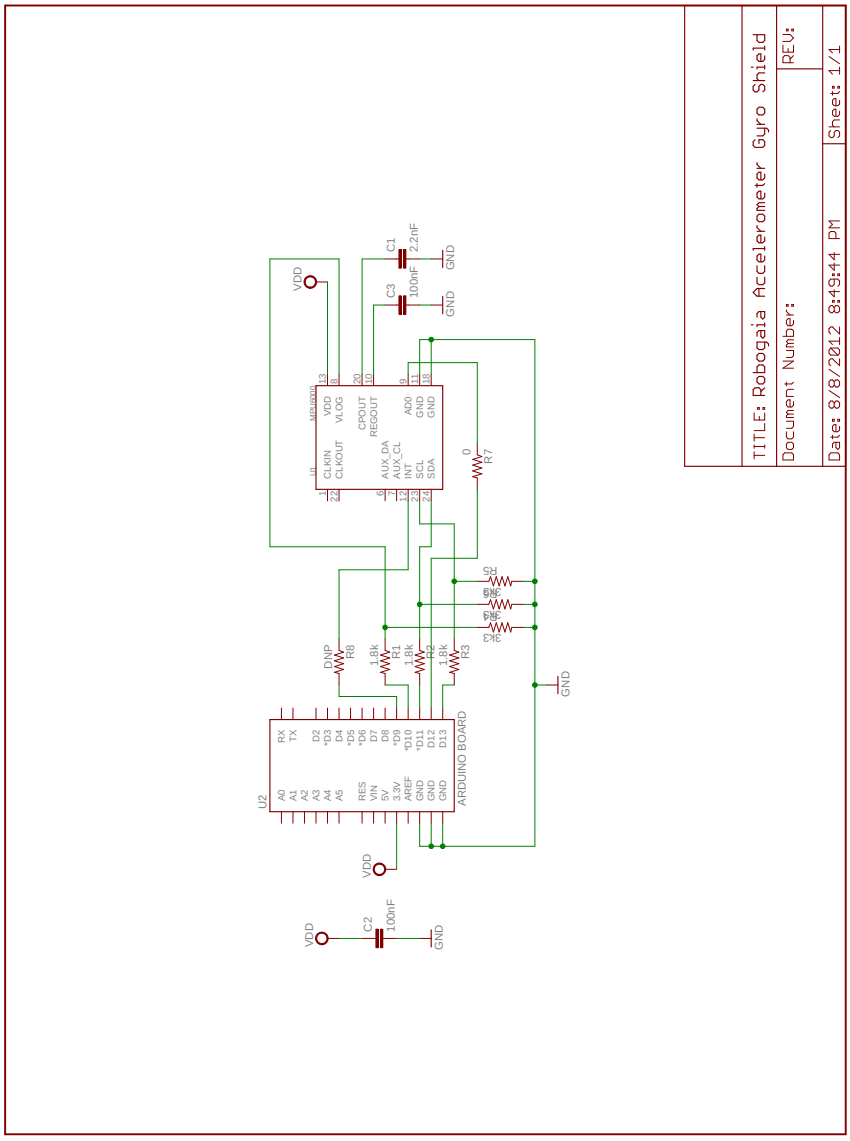
[7] Qian Ming. Sliding mode controller design for abs system. Master's thesis, Virginia Tech, 1997.

[8] Polou Robotics and Electronics. 131:1 metal gearmotor 37dx57l mm with 64 cpr encoder. http://www.pololu.com/product/1447.

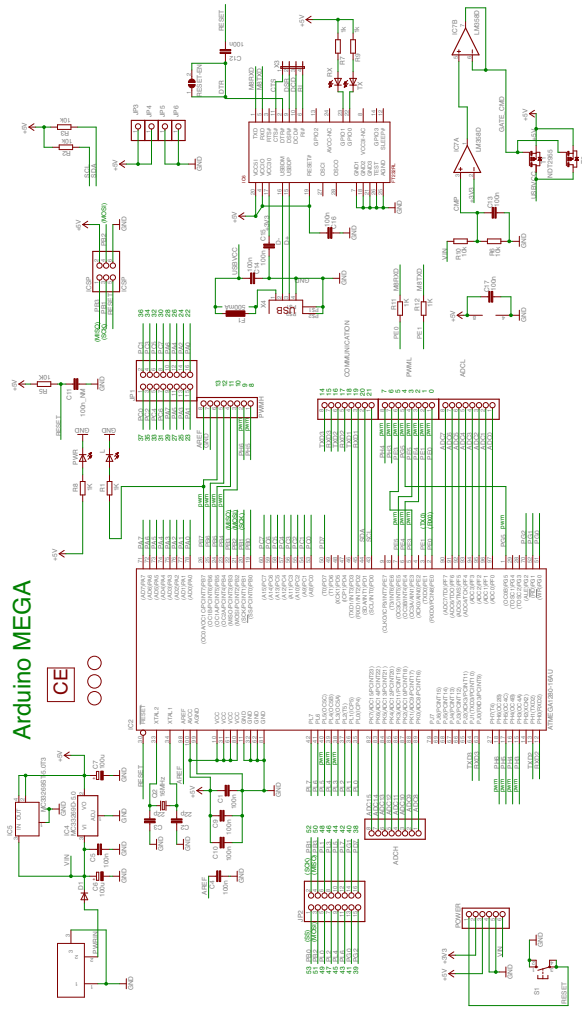
Appendix A

PART DOCUMENTS

A.1 Accelerometer Electrical Diagram



A.2 Micro-Controller Electrical Diagram



3/26/09 11:14 AM f=0.70 /Users/mellis/Documents/Physical Computing/eagle/mega_v2/Arduino_MEGA_02.sch (Sheet: 1/1)

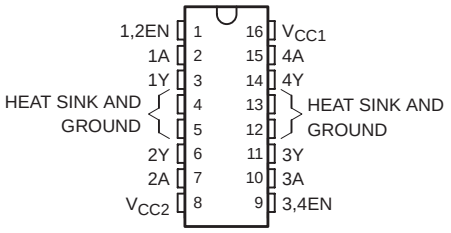
A.3 H-Bridge Spec Sheet

SN754410
QUADRUPLE HALF-H DRIVER

SLRS007B – NOVEMBER 1986 – REVISED NOVEMBER 1995

- **1-A Output-Current Capability Per Driver**
- **Applications Include Half-H and Full-H Solenoid Drivers and Motor Drivers**
- **Designed for Positive-Supply Applications**
- **Wide Supply-Voltage Range of 4.5 V to 36 V**
- **TTL- and CMOS-Compatible High-Impedance Diode-Clamped Inputs**
- **Separate Input-Logic Supply**
- **Thermal Shutdown**
- **Internal ESD Protection**
- **Input Hysteresis Improves Noise Immunity**
- **3-State Outputs**
- **Minimized Power Dissipation**
- **Sink/Source Interlock Circuitry Prevents Simultaneous Conduction**
- **No Output Glitch During Power Up or Power Down**
- **Improved Functional Replacement for the SGS L293**

NE PACKAGE
(TOP VIEW)



FUNCTION TABLE
(each driver)

INPUTS†		OUTPUT
A	EN	Y
H	H	H
L	H	L
X	L	Z

H = high-level, L = low-level
X = irrelevant
Z = high-impedance (off)
† In the thermal shutdown mode, the output is in a high-impedance state regardless of the input levels.

description

The SN754410 is a quadruple high-current half-H driver designed to provide bidirectional drive currents up to 1 A at voltages from 4.5 V to 36 V. The device is designed to drive inductive loads such as relays, solenoids, dc and bipolar stepping motors, as well as other high-current/high-voltage loads in positive-supply applications.

All inputs are compatible with TTL-and low-level CMOS logic. Each output (Y) is a complete totem-pole driver with a Darlington transistor sink and a pseudo-Darlington source. Drivers are enabled in pairs with drivers 1 and 2 enabled by 1,2EN and drivers 3 and 4 enabled by 3,4EN. When an enable input is high, the associated drivers are enabled and their outputs become active and in phase with their inputs. When the enable input is low, those drivers are disabled and their outputs are off and in a high-impedance state. With the proper data inputs, each pair of drivers form a full-H (or bridge) reversible drive suitable for solenoid or motor applications.

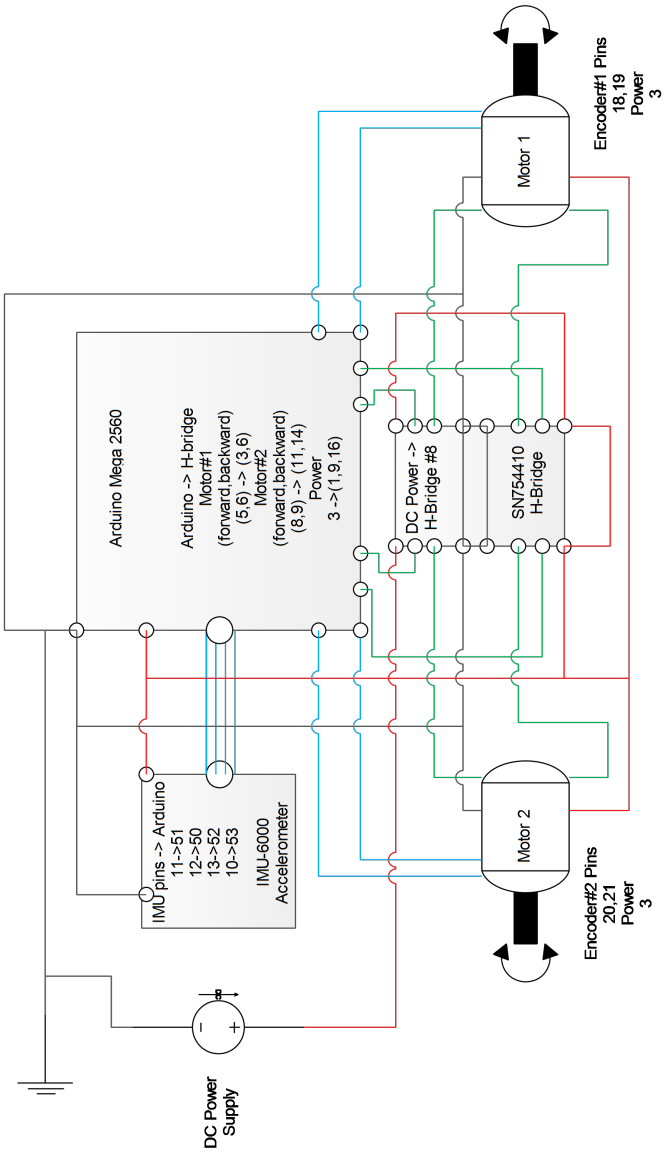
A separate supply voltage (V_{CC1}) is provided for the logic input circuits to minimize device power dissipation. Supply voltage V_{CC2} is used for the output circuits.

The SN754410 is designed for operation from −40°C to 85°C.

TEXAS INSTRUMENTS
POST OFFICE BOX 655303 • DALLAS, TEXAS 75265
POST OFFICE BOX 1443 • HOUSTON, TEXAS 77251–1443

Appendix B

COMPLETE ELECTRICAL DIAGRAM



Appendix C

ARDUINO CODE

[8]

```
#include <SPI.h>              //Include the serial communication library
#include <digitalWriteFast.h> //Include the fast reading and writing library    //
for accurate Encoder measurments

//*****************************************************
/* Ryan Scott
   4/21/14
   Velocity and Yaw Control

   The purpose of this program is to input a control algorithim as described below
     to control the velocity and yaw angle of the vehicle by adjusting motor power.

   Rev2: Uses Encoder Lib instead of custom reading programs
   Rev3: Use  DigitalWriteFast lib in custom velocity calcs (Undo rev2)
   Rev4: Changed Code in "getXAccelerometerCounts" subroutine.
           Changed the "|" in the return to a "+" to match the other
           "getAccelerometerCounts" routines.
   Rev5: Changed the yAccelerationOffset to -0.05 and the zGyroOffset to -1.3.
   Rev6: Added Calibration code to choose the appropriate offset at the  begining of each run.(Undid rev3)
   Rev7: Added a filter section just before the accelerometer values are returned.
           with a cutoff frequecy of 5Hz
   Rev8: Save computation time by combining the speed measurments into a
           single function call.

*/
//*****************************************************

//Begin definition of pins
//*****************************************************
#define pwr 2  //powers H-Bridge, and encoders

#define Encd2A 18  //Encoder Pins to read and calculate velocities
#define Encd2B 19
#define Encd1A 20
#define Encd1B 21

#define MRf 5    //Pins to control direction and speed of motors
#define MRb 6
#define MLf 8
#define MLb 9
//*****************************************************
```

```
//Initialize Parameters
//*****************************************************
 float t_start = 0;
 float t = 0;
 int countOld1 = 0;
 int countNew1 = 0;
 int countOld2 = 0;
 int countNew2 = 0;
 float rps[] = {0,0};
 float t_old = 0;
 float To = 0;
 float Tn = 0;
 int i = 0;
 int j = 0;

 //int Tsample = 50;  //Time between loop runs (milisecs)
 float r1 = 0.3;   //Desired Velocity (m/sec)
 float r2 = 0;    //Desired Yaw Angle(rad/sec)

 float yDot = 0.0;
 float Phi = 0.0;

 int pin1 = 0;  //Initialize pin and PWM values
 int pwm1 = 0;
 int pin2 = 0;
 int pwm2 = 0;

 float Vw1 = 0.0;  //Initialize wheel speed estimator values
 float Vw2 = 0.0;
 float V = 0.0;

 float Volt = 11.0;
 float u1 = 0;
 float u2 = 0;
 float R = 3*2.54*0.01;  //Radius of wheel in meters
 float two_pi = 6.2831;

 volatile int count1 = 0;    //For use in Encoder subroutines
 volatile int count2 = 0;
 volatile int dir1 = 0;
 volatile int dir2 = 0;
```

```
float alpha = 0.8;    //Filtering Coefficients (Coresponds to ~5HZ Lowpass Filter)
float beta = 1 - alpha;
float xAccelerationValueOld = 0;  //Initialize Storage for accelerometer data
float yAccelerationValueOld = 0;
float zGyroValueOld = 0;

float Av = 0;    //Velocity and Yaw weighting coefficients based on controller
float Bv = 0;    //  calculations
float Aphi = 0;
float Bphi = 0;



float e1 = 0;      //Velocity and yaw error
float e1_prev = 0;
float e2 = 0;
float e2_prev = 0;

float Kpv = 40;    //Proportional and integral controller gains for
float Kiv = 10;    //    velocity and yaw rate
float Kpphi = 20;
float Kiphi = 10;


//MPU 6000 register addresses
const int ACCEL_XOUT_H = 0x3B;    //59  R ACCEL_XOUT[15:8]
const int ACCEL_XOUT_L = 0x3C;    //60  R ACCEL_XOUT[7:0]
const int ACCEL_YOUT_H = 0x3D;    //61  R ACCEL_YOUT[15:8]
const int ACCEL_YOUT_L = 0x3E;    //62  R ACCEL_YOUT[7:0]
const int ACCEL_ZOUT_H = 0x3F;    //63  R ACCEL_ZOUT[15:8]
const int ACCEL_ZOUT_L = 0x40;    //64  R ACCEL_ZOUT[7:0]
const int TEMP_OUT_H   = 0x41;      //65  R TEMP_OUT[15:8]
const int TEMP_OUT_L   = 0x42;      //66  R TEMP_OUT[7:0]
const int GYRO_XOUT_H  = 0x43;     //67  R GYRO_XOUT[15:8]
const int GYRO_XOUT_L  = 0x44;     //68  R GYRO_XOUT[7:0]
const int GYRO_YOUT_H  = 0x45;     //69  R GYRO_YOUT[15:8]
const int GYRO_YOUT_L  = 0x46;     //70  R GYRO_YOUT[7:0]
const int GYRO_ZOUT_H  = 0x47;     //71  R GYRO_ZOUT[15:8]
const int GYRO_ZOUT_L  = 0x48;     //72  RGYRO_ZOUT[7:0]

const int READ_FLAG =0x80;   //128 has to be added to the address register
```

```
// MPU 6000 registers
#define MPUREG_WHOAMI 0x75 //
#define MPUREG_SMPLRT_DIV 0x19 //
#define MPUREG_CONFIG 0x1A //
#define MPUREG_GYRO_CONFIG 0x1B
#define MPUREG_ACCEL_CONFIG 0x1C
#define MPUREG_INT_PIN_CFG 0x37
#define MPUREG_INT_ENABLE 0x38
#define MPUREG_ACCEL_XOUT_H 0x3B //
#define MPUREG_ACCEL_XOUT_L 0x3C //
#define MPUREG_ACCEL_YOUT_H 0x3D //
#define MPUREG_ACCEL_YOUT_L 0x3E //
#define MPUREG_ACCEL_ZOUT_H 0x3F //
#define MPUREG_ACCEL_ZOUT_L 0x40 //
#define MPUREG_TEMP_OUT_H 0x41//
#define MPUREG_TEMP_OUT_L 0x42//
#define MPUREG_GYRO_XOUT_H 0x43 //
#define MPUREG_GYRO_XOUT_L 0x44 //
#define MPUREG_GYRO_YOUT_H 0x45 //
#define MPUREG_GYRO_YOUT_L 0x46 //
#define MPUREG_GYRO_ZOUT_H 0x47 //
#define MPUREG_GYRO_ZOUT_L 0x48 //
#define MPUREG_USER_CTRL 0x6A //
#define MPUREG_PWR_MGMT_1 0x6B //
#define MPUREG_PWR_MGMT_2 0x6C //


// Configuration bits  MPU 6000
#define BIT_SLEEP 0x40
#define BIT_H_RESET 0x80
#define BITS_CLKSEL 0x07
#define MPU_CLK_SEL_PLLGYROX 0x01
#define MPU_CLK_SEL_PLLGYROZ 0x03
#define MPU_EXT_SYNC_GYROX 0x02
#define BITS_FS_250DPS              0x00
#define BITS_FS_500DPS              0x08
#define BITS_FS_1000DPS             0x10
#define BITS_FS_2000DPS             0x18
#define BITS_FS_MASK                0x18
#define BITS_DLPF_CFG_256HZ_NOLPF2  0x00
#define BITS_DLPF_CFG_188HZ         0x01
```

```
#define BITS_DLPF_CFG_98HZ          0x02
#define BITS_DLPF_CFG_42HZ          0x03
#define BITS_DLPF_CFG_20HZ          0x04
#define BITS_DLPF_CFG_10HZ          0x05
#define BITS_DLPF_CFG_5HZ           0x06
#define BITS_DLPF_CFG_2100HZ_NOLPF  0x07
#define BITS_DLPF_CFG_MASK          0x07
#define BIT_INT_ANYRD_2CLEAR     0x10
#define BIT_RAW_RDY_EN     0x01
#define BIT_I2C_IF_DIS              0x10


// pins used for the connection with the sensor
// the other you need are controlled by the SPI library):
//const int dataReadyPin = 6;
const int chipSelectPin = 53;


//*****************************************************
//Calibration values
//Accelerations are calibrated to read in m/s2
//Gyro values calibrated to read in rad/s

 float xAccelerationGain=9.81/8192.0;
 float xAccelerationOffset=0;

 float yAccelerationGain=9.81/8192.0;
 float yAccelerationOffset=0;

 float zAccelerationGain=9.81/8192.0;
 float zAccelerationOffset=0;

 float xGyroGain=34.91/32768.0;
 float xGyroOffset=0;

 float yGyroGain=34.91/32768.0;
 float yGyroOffset=0;

 float zGyroGain=34.91/32768.0;
 float zGyroOffset=0;
//*****************************************************
//end calibration values
```

```
//*****************************************************


//Begin Pin Setup
void setup()  {
  Serial.begin(9600);

  attachInterrupt(5, Count2A, RISING);    //Attach interrupts for the encoders
  attachInterrupt(4, Direction2, RISING);
  attachInterrupt(3, Count1A, RISING);
  attachInterrupt(2, Direction1, RISING);

  pinMode(pwr, OUTPUT);    //Turn on H-Bridge and Encoders
  digitalWrite(pwr, HIGH);

  pinMode(MRf, OUTPUT);    //Set motor pins to output
  pinMode(MRb, OUTPUT);
  pinMode(MLf, OUTPUT);
  pinMode(MLb, OUTPUT);

  pinMode(Encd1A, INPUT);  //Set encoder pins to inputs
  pinMode(Encd1B, INPUT);
  pinMode(Encd2A, INPUT);
  pinMode(Encd2B, INPUT);

  pinMode(chipSelectPin, OUTPUT);    //Activate the accelerometer
  digitalWrite(chipSelectPin, HIGH);


  MPU6000_Init();  //Initialize the accelerometer
  delay(100);
  Calibration();  //Calibrate the accelerometer
  delay(1000);
  t_start = 0.001 * millis();  //Set the zero time
}

//*****************************************************
//BEGIN OF LOOP
//*****************************************************
void loop() {
        float sumX = 0;
        float sumY = 0;
```

```
        float sumZ = 0;

        float xAccelerationValue;
        float yAccelerationValue;
        float zGyroValue;

        float rps1;
        float rps2;

        MeasureSpeed();
        t = 0.001*millis() - t_start;


        Serial.print(t);
        Serial.print(",");
        Serial.print(rps[0]);
        Serial.print(",");
        Serial.print(rps[1]);
        Serial.print(",");

        //Take 10 measurements in rapid succession and average results
        for(i=0;i<=9;i++){
        xAccelerationValue = xAccelerationGain * (float)getXAccelerometerCounts()  +  xAccelerationOffset;
        sumX = sumX + xAccelerationValue;
        yAccelerationValue = yAccelerationGain * (float)getYAccelerometerCounts()  +  yAccelerationOffset;
        sumY = sumY + yAccelerationValue;
        zGyroValue = zGyroGain *(float) getZGyroCounts()  +  zGyroOffset;
        sumZ = sumZ + zGyroValue;
        }
        xAccelerationValue = sumX / 10;
        yAccelerationValue = sumY / 10;
        zGyroValue = sumZ / 10;

        //the values received from MPU 6000 are raw and have to be calibrated
        //we use a linear function y=mx +b to transform the raw counts to real
        //acceleration values  G   and rotations / sec rot gyros
        // in this example the gain (m) and offset (x) are setup to m=1 and x=0  //so there is no change in raw data


//Filter Results
        xAccelerationValue  =  alpha * xAccelerationValueOld + beta * xAccelerationValue;
        xAccelerationValueOld = xAccelerationValue;
```

```
      yAccelerationValue  =  alpha * yAccelerationValueOld + beta * yAccelerationValue;
      yAccelerationValueOld = yAccelerationValue;

      zGyroValue  =  alpha * zGyroValueOld + beta * zGyroValue;
      zGyroValueOld = zGyroValue;

      Serial.print(xAccelerationValue);
      Serial.print(",");
      Serial.print(yAccelerationValue);
      Serial.print(",");
      Serial.print(zGyroValue);
      Serial.print(",");

//*****************************************************
    //For vehicle characterization comment out code from here until marked section
//*****************************************************
     Vw1 = rps[0] * two_pi * R;
     Vw2 = rps[1] * two_pi * R;
     V = 0.5*(Vw1 + Vw2);

   //Integrate Accelerometer Data to determine bulk velocity and yaw angle
   yDot = yDot + (yAccelerationValue)*(t - t_old);
   Phi  = Phi  + (zGyroValue)*(t-t_old);

   Serial.print(yDot);
   Serial.print(",");
   Serial.print(Phi);
   Serial.print(",");
   Serial.print(V);
   Serial.print(",");

   //Adjust inputs according to control Laws
   e1 = r1 - V;    //Calculate Error
   e2 = r2 - Phi;

   Av = Kpv + 0.5*Kiv*(t - t_old);    //Calculate Velocity Weighting Coefficients
   Bv = 0.5*Kiv*(t - t_old) - Kpv;

   Aphi = Kpphi + 0.5*Kiphi*(t-t_old);  //Calculate Yaw Weighting Coefficients
   Bphi = 0.5*Kiphi*(t - t_old) - Kpphi;
```

```
u1 = u1 + Av*e1 + Aphi*e2 + Bv*e1_prev + Bphi*e2_prev;  //Calculate new inputs to motor1
u2 = u2 + Av*e1 - Aphi*e2 + Bv*e1_prev - Bphi*e2_prev;  //  and motor 2

e1_prev = e1;  //Reset previous errors
e2_prev = e2;

Serial.print(u1);
Serial.print(",");
Serial.print(u2);
Serial.print(",");


pwm1 = 255*(u1)/(Volt);  //Convert required voltage to PWM signal
pwm2 = 255*(u2)/(Volt);

//Allpy Saturation conditions to PWM signal
if (pwm1 >= 255){
  pwm1 = 255;}
else if(pwm1 <= -255){
  pwm1 = -255;}

if (pwm2 >= 255){
  pwm2 = 255;}
else if(pwm2 <= -255){
  pwm2 = -255;}

Serial.print(pwm1);
Serial.print(",");
Serial.println(pwm2);

//Determine the desired direction and select the corresponding pin
if( pwm1 >= 0){
  pin1 = MRf;
}
else{
  pin1 = MRb;
  pwm1 = -1*pwm1;
}

if( pwm2 >= 0){
  pin2 = MLf;
}
```

```
     else{
       pin2 = MLb;
       pwm2 = -1*pwm2;
     }
     analogWrite(pin1,pwm1);  //Send PWM signals to both motors
     analogWrite(pin2,pwm2);

     //*****************************************************
      //For vehicle characterization comment out code up to this point
      //  and uncomment seccode below
      //*****************************************************
     //digitalWrite(MRf,HIGH);
     //digitalWrite(MLf,HIGH);

     t_old = t;  //Reset time


}
//*****************************************************
//END OF LOOP
//*****************************************************



//Begin Subroutines
//*****************************************************
int getXAccelerometerCounts(void)
//*****************************************************
{
    int tempData_HI,tempData_LO;
     //Read the  data
    tempData_HI = readRegister(ACCEL_XOUT_H);
    tempData_LO = readRegister(ACCEL_XOUT_L );

    return ((tempData_HI << 8) + tempData_LO);

}//end func

//*****************************************************
int getYAccelerometerCounts(void)
//*****************************************************
{
```

```
    int tempData_HI,tempData_LO;

    tempData_HI = readRegister(ACCEL_YOUT_H);
    tempData_LO = readRegister(ACCEL_YOUT_L );

    return ((tempData_HI << 8) + tempData_LO);
}//end func

//*****************************************************
int getZAccelerometerCounts(void)
//*****************************************************
{
    int tempData_HI,tempData_LO;

    tempData_HI = readRegister(ACCEL_ZOUT_H );
    tempData_LO = readRegister(ACCEL_ZOUT_L );

    return ((tempData_HI << 8) + tempData_LO);

}//end func

//*****************************************************
int getXGyroCounts(void)
//*****************************************************
{
     int tempData_HI,tempData_LO;

     tempData_HI = readRegister(GYRO_XOUT_H );
     tempData_LO = readRegister(GYRO_XOUT_L );

     return ((tempData_HI << 8) + tempData_LO);
}//end func

//*****************************************************
int getYGyroCounts(void)
//*****************************************************
{
     int tempData_HI,tempData_LO;

     tempData_HI = readRegister(GYRO_YOUT_H);
     tempData_LO = readRegister(GYRO_YOUT_L );
```

```
     return ((tempData_HI << 8) + tempData_LO);
}//end func

//*****************************************************
int getZGyroCounts(void)
//*****************************************************
{
     int tempData_HI,tempData_LO;

     tempData_HI = readRegister(GYRO_ZOUT_H);
     tempData_LO = readRegister(GYRO_ZOUT_L);

     return ((tempData_HI << 8) + tempData_LO);
}//end func

//*****************************************************
unsigned int getTemperature(void)
//*****************************************************
{
     int tempData_HI,tempData_LO;

     tempData_HI = readRegister(TEMP_OUT_H + READ_FLAG);
     tempData_LO = readRegister(TEMP_OUT_L + READ_FLAG);

     return ((tempData_HI << 8) + tempData_LO);
}//end func


//Read from or write to register
//*****************************************************
unsigned int readRegister(byte thisRegister)
//*****************************************************
{

  unsigned int result = 0;   // result to return
  byte addr = thisRegister + 0x80;
 // Serial.print(thisRegister, BIN);
 // byte dataToSend = thisRegister ;
  //Serial.println(thisRegister, BIN);
  // take the chip select low to select the device:
  digitalWrite(chipSelectPin, LOW);
  // send the device the register you want to read:
```

```
  SPI.transfer(addr);
  // send a value of 0 to read the first byte returned:
  result = SPI.transfer(0x00);
  // take the chip select high to de-select:
  digitalWrite(chipSelectPin, HIGH);
  // return the result:
  return(result);
}


//Sends a write command
//*****************************************************
void writeRegister(byte thisRegister, byte thisValue)
//*****************************************************
{


  thisRegister = thisRegister ;
  // now combine the register address and the command into one byte:
  byte dataToSend = thisRegister;

  // take the chip select low to select the device:
  digitalWrite(chipSelectPin, LOW);

  SPI.transfer(dataToSend); //Send register location
  SPI.transfer(thisValue);  //Send value to record into register

  // take the chip select high to de-select:
  digitalWrite(chipSelectPin, HIGH);
}//


// MPU6000 Initialization and configuration
//*************************************************
void MPU6000_Init(void)
//*************************************************
{
    // MPU6000 chip select setup
   // pinMode(MPU6000_CHIP_SELECT_PIN, OUTPUT);
   // writeRegister(MPU6000_CHIP_SELECT_PIN, HIGH);

    // SPI initialization
```

```
    SPI.begin();
    SPI.setClockDivider(SPI_CLOCK_DIV16);      // SPI at 1Mhz (on 16Mhz clock)
    delay(10);

    // Chip reset
    writeRegister(MPUREG_PWR_MGMT_1, BIT_H_RESET);
    delay(100);
    // Wake up device and select GyroZ clock (better performance)
    writeRegister(MPUREG_PWR_MGMT_1, MPU_CLK_SEL_PLLGYROZ);
    delay(1);
    // Disable I2C bus (recommended on datasheet)
    writeRegister(MPUREG_USER_CTRL, BIT_I2C_IF_DIS);
    delay(1);
    // SAMPLE RATE
    //MPU6000_SPI_write(MPUREG_SMPLRT_DIV,0x04);     // Sample rate = 200Hz    Fsample= 1Khz/(4+1) = 200Hz
    writeRegister(MPUREG_SMPLRT_DIV,19);     // Sample rate = 50Hz    Fsample= 1Khz/(19+1) = 50Hz
    delay(1);
    // FS & DLPF   FS=2000/s, DLPF = 20Hz (low pass filter)
    writeRegister(MPUREG_CONFIG, BITS_DLPF_CFG_20HZ);
    delay(1);
    writeRegister(MPUREG_GYRO_CONFIG,BITS_FS_2000DPS);  // Gyro scale 2000/s
    delay(1);
    writeRegister(MPUREG_ACCEL_CONFIG,0x08);            // Accel scale 4g (4096LSB/g)
    delay(1);

    writeRegister(MPUREG_PWR_MGMT_1,MPU_CLK_SEL_PLLGYROZ);
    delay(1);
}

//Determine the desired offsets for the xAcceleration, yAcceleration, and zGyro
//*****************************************************
void Calibration(){
  //*****************************************************
  float xAccelerationValue;
  float yAccelerationValue;
  float zGyroValue;
  float xAcc[] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
  float yAcc[] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
  float zGyro[] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
  float sumXAcc = 0;
  float sumYAcc = 0;
  float sumZGyro = 0;
```

```
  int i = 0;

  for( i = 0; i <= 9; i++){
       xAccelerationValue = xAccelerationGain * (float)getXAccelerometerCounts()  +  xAccelerationOffset;
       xAcc[i] = xAccelerationValue;
       yAccelerationValue = yAccelerationGain * (float)getYAccelerometerCounts()  +  yAccelerationOffset;
       yAcc[i] = yAccelerationValue;
       zGyroValue = zGyroGain *(float) getZGyroCounts()  +  zGyroOffset;
       zGyro[i] = zGyroValue;
       delay(50);
  }
  //Now average results and set these to the offset values
  for (int i = 0; i <= 9; i++){
   sumXAcc = sumXAcc + xAcc[i];
   sumYAcc = sumYAcc + yAcc[i];
   sumZGyro = sumZGyro + zGyro[i];
  }
  xAccelerationOffset = - sumXAcc / 10;
  yAccelerationOffset = - sumYAcc / 10;
  zGyroOffset = - sumZGyro / 10;
}//End Function

//*****************************************************
void MeasureSpeed()  {
  //*****************************************************
  countOld1 = 0;
  countNew1 = 0;
  countOld2 = 0;
  countNew2 = 0;
  To = 0;
  Tn = 0;

  countOld1 = count1;
  countOld2 = count2;

  To = 0.001 * millis();
  delay(50);

  countNew1 = count1;
  countNew2 = count2;

  Tn = 0.001 * millis();
```

```
  rps[0] = dir1 * ( countNew1 - countOld1 )/( (Tn - To) * 16 * 131);
  rps[1] = dir2 * ( countNew2 - countOld2 )/( (Tn - To) * 16 * 131);
}//End Function


//Define Interput Service Routines
//*****************************************************
void Count1A(){
  //*****************************************************
  count1 ++;
}//End Function


//*****************************************************
void Count2A(){
  //*****************************************************
  count2 ++;
}//End Function


//*****************************************************
void Direction1(){
  //*****************************************************
  if( digitalReadFast(Encd1B) == digitalReadFast(Encd1A) ) {
      dir1 = 1;}
  else{
      dir1 = -1;
  }
}//End Function


//*****************************************************
void Direction2(){
  //*****************************************************
  if( digitalReadFast(Encd2B) == digitalReadFast(Encd2A) ){
      dir2 = -1;}
  else{
      dir2 = 1;
  }
}//End Function
```

```
//*****************************************************
//*****************************************************
//END OF PROGRAM
//*****************************************************
//*****************************************************
```

Appendix D

SIMULATION CODE

D.1 Vehicle Characterization

[8]

```
%Ryan Scott
%ME700
% 5/7/14

%The purpose of this m file is to perform analysis on the
%   open loop behavior of the IWD vehicle by importing data
%   from another file, plotting and determining the corresponding
%   traction gains

clc
close all
clear all

%Retrieve data

[Q,Notes] = OpenLoop050714(2);
N1 = min( find(Q(:,2)) ); %Find the first time the right wheel turns
N2 = min( find(Q(:,3)) ); %Find the first time the left wheel turns
N = min(N1,N2) - 1;%Take the time instance just before movement
n = length(Q(:,1));
Notes
%seperate out vectors of data
t = Q(N:n,1);
rpsR = Q(N:n,2) * 2*pi;
rpsL = Q(N:n,3) * 2*pi;
xAcc = Q(N:n,4);
yAcc = Q(N:n,5);
zGyro = Q(N:n,6);

T = t(1);
t = t - T;  %Reset time scale to start at the begining of the test


%interpolate results for even point spacing
time = [0:0.01:max(t)]';

rpsR = interp1(t,rpsR,time);
rpsL = interp1(t,rpsL,time);
xAcc = interp1(t,xAcc,time);
```

```
yAcc = interp1(t,yAcc,time);
zGyro = interp1(t,zGyro,time);

t = time;
n = length(t);  %Redefine length of vectors

%Define system parameters
R = 3*2.54*0.01;    %Wheel Radius in Meters
N = 5*4.448;      %Vehicle Weight in Newtons
g = 9.81;           %Acceleration due to gravity
m = N/g;            %Vehicle mass in kg
Volt = 11.5;          %Volts applied to motors
mu_p = 0.8;         %Peak coefficiant of friction
lam_p = 0.2;        %Slip ratio at which it occurs

%Results Determined in the Matlab program "MotorCharacterization_rev2.m"
KmR = 0.5287;    %Right Motor Free Spin Gain (rad/(sec*volt))
TmR = 0.3864;    %Right Motor Time Constant (1/sec)
KmL = 0.6274;    %Left Motor Free Spin Gain (rad/(sec*volt))
TmL = 0.4318;    %Left Motor Time Constant (1/sec)

%Plot Results of raw data (uncomment this code for plots)
% figure
% subplot(2,2,1),plot(t,rpsR,t,rpsL),title('Wheel Speeds'),xlabel('Time (sec)'),ylabel('rad/s')
% subplot(2,2,2),plot(t,xAcc),title('xAcc'),xlabel('Time (sec)'),ylabel('m/s^2')
% subplot(2,2,3),plot(t,yAcc),title('yAcc'),xlabel('Time (sec)'),ylabel('m/s^2')
% subplot(2,2,4),plot(t,zGyro),title('zGyro'),xlabel('Time (sec)'),ylabel('rad/sec')

%Initialize vectors for vehicle velocity, global velocities,
%   positions and orientations
Ydot = zeros(n,1);
Vy = zeros(n,1);
y = zeros(n,1);
Xdot = zeros(n,1);
Vx = zeros(n,1);
x = zeros(n,1);
Phi = zeros(n,1);

%Calculate the pure rolling wheel velocities
VwR = R*rpsR;
VwL = R*rpsL;
```

```
%Determine the longitudinal and lateral velocities  and yaw angle from
%   the trapizoidal approximate integral
for i = 2:n
    Ydot(i) = 0.5*(yAcc(i) + yAcc(i-1))*(t(i)-t(i-1)) + Ydot(i-1);
    Xdot(i) = 0.5*(xAcc(i) + xAcc(i-1))*(t(i)-t(i-1)) + Xdot(i-1);
    Phi(i) = 0.5*(zGyro(i) + zGyro(i-1))*(t(i)-t(i-1)) + Phi(i-1);

    %Calculate wheel velocities
    VR(i) = Ydot(i)*(1 + sin(zGyro(i)));
    VL(i) = Ydot(i)*(1 - sin(zGyro(i)));

    %Calculate Velocities in the global coordinate frame
    Vx(i) = - sin( Phi(i) )*Ydot(i);
    Vy(i) = cos( Phi(i) )*Ydot(i);

    %Calculate position in the global coordinate frame
    y(i) = (0.5 * (Vy(i) + Vy(i-1)) )*(t(i) - t(i-1)) + y(i-1);
    x(i) = (0.5 * (Vx(i) + Vx(i-1)) )*(t(i) - t(i-1)) + x(i-1);


    %Calculate Slip Ratios
    if abs(VwR(i)) >= abs(VR(i))
        MaxVR = VwR(i);
    else
        MaxVR = VR(i);
    end

    if abs(VwL(i)) >= abs(VL(i))
        MaxVL = VwL(i);
    else
        MaxVL = VL(i);
    end


    if VwR(i) == VR(i) || MaxVR == 0
        LamR(i) = 0;
    else
        LamR(i) = (VwR(i) - VR(i))/MaxVR;
    end

    if VwL(i) == VL(i) || MaxVL == 0
        LamL(i) = 0;
```

```
    else
        LamL(i) = (VwL(i) - VL(i))/MaxVL;
    end

    %Calculate coefiicients of friction
    muR(i) = (2*mu_p*lam_p*LamR(i))/(lam_p^2 + (LamR(i))^2);
    muL(i) = (2*mu_p*lam_p*LamL(i))/(lam_p^2 + (LamL(i))^2);

    %Calculate traction Forces
    FtR(i) = 0.25*N*muR(i);
    FtL(i) = 0.25*N*muL(i);
end

figure
subplot(2,2,1),plot(t,rpsR,t,rpsL),xlabel('Time (sec)'),ylabel('Rad/s')
    title('Wheel Speeds'),grid on,legend('w_1','w_2')
subplot(2,2,2),plot(t,Ydot),xlabel('Time (sec)'),ylabel('m/s')
    title('Vehicle Velocity'),grid on
subplot(2,2,3),plot(t,zGyro),xlabel('Time (sec)'),ylabel('Rad/s')
    title('Yaw Rate'),grid on
subplot(2,2,4),plot(t,FtR,t,FtL),xlabel('Time (sec)'),ylabel('Newtons')
    title('Traction Forces'),grid on,legend('F_{T_1}','F_{T_2}')

figure
plot(t,FtR,t,FtL),xlabel('Time (sec)'),ylabel('Newtons')
    title('Traction Forces'),grid on,legend('F_{T_1}','F_{T_2}')
figure
plot(x,y),xlabel('X-Position (m)'),ylabel('Y-Position (m)')
    grid on,axis([-1 1 0 2])

%Calculate positions and velocity in english units
xe = x*(100/(2.54*12));
ye = y*(100/(2.54*12));
Ydote = Ydot*(100/(2.54*12));

%Start with Simulating the w1(s) and w2(s)
    %          1
    %w1(s) = ------- [ Km1 V1(s) - Kt1 Ft1(s) ]
    %        Tm1*s+1
    %
    %          1
    %w2(s) = ------- [ Km2 V2(s) - Kt2 Ft2(s) ]
```

```
    %        Tm2*s+1
%Inputs will be of the form U = Km*Volts - Kt*Ft(s)
MotorR = tf([1],[TmR 1]);   %Build transfer functions for the motors
MotorL = tf([1],[TmL 1]);
dt = (max(t)-min(t))/(n-1);
T = [min(t):dt:max(t)]';    %Create time span of uniform spacing

%Determine SS response approximately
iR_2 = find(t == 2);
iL_2 = find(t == 2);
I_4 = find(t == 4);

wR_ss = mean(rpsR(iR_2:I_4));     %Approximate SS velocity
wL_ss = mean(rpsL(iL_2:I_4));

FtR_ss = mean(FtR(iR_2:I_4));     %Approximate SS Traction
FtL_ss = mean(FtL(iL_2:I_4));

KtR = (KmR*Volt - wR_ss ) / FtR_ss    %Approximate Gains
KtL = (KmL*Volt - wL_ss ) / FtL_ss

UR = KmR*Volt*ones(n,1) - KtR * FtR';  %Generate inputs for the right
wR = lsim(MotorR,UR,T);                 % motor and simulate
errorR = wR - rpsR;
MaxER = max(abs(errorR))/max(rpsR);



UL = KmL*Volt*ones(n,1) - KtL * FtL';  %Generate inputs for the left
wL = lsim(MotorL,UL,T);                 % motor and simulate
errorL = wL - rpsL;
MaxEL = max(abs(errorL))/max(rpsL);

figure                                  %Plot the results
subplot(2,2,2),plot(t(:,1),wL(:,1),t(:,1),rpsL(:,1)),xlabel('Time (sec)')
    ylabel('Angular Velocity (rad/sec)'),legend('Simulated','Measured')
    title('Wheel Speed of Left Drive Wheel')
subplot(2,2,4),plot(t(:,1),errorL(:,1)),xlabel('Time (sec)')
    ylabel('Absolute Error (rad/sec)'),title('Error of Left Wheel')
subplot(2,2,1),plot(t(:,1),wR(:,1),t(:,1),rpsR(:,1)),xlabel('Time (sec)')
    ylabel('Angular Velocity (rad/sec)'),legend('Simulated','Measured')
    title('Wheel Speed of Right Drive Wheel')
```

```
subplot(2,2,3),plot(t(:,1),errorR(:,1)),xlabel('Time (sec)')
    ylabel('Absolute Error (rad/sec)'),title('Error of Right Wheel')
```

D.2 Experimental Controller Analysis

[8]

```
%Ryan Scott
%ME 700
%Feedback control analysis and controller design

%The purpose of this program is analysis closed loop experimental data
    %using the PI controllers defined in the "Gains" matrix.  Data is
    %stored in a seperate file "ClosedLoop051714.m"

close all
clear all
clc



%Controller Gains used in experiment
Gains = [10 10 10 10;
         20 10 20 10;
         50 10 20 10;
         20 20 20 20;
         30 10 20 10;
         40 10 20 10;
         30 30 20 5;
         30 5  30 5;
         20 15 20 15;
         30 15 30 15
         ];

Volt = 11;  %operating Voltage
m1 = 1;     %First Data Set to test
m2 = 10;    %Final Data Set to test
MaxX = zeros(m2-m1+1,1);    %Initialize performance index vectors
MaxPhi = zeros(m2-m1+1,1);
Trv = zeros(m2-m1+1,1);
RecoveryTime = zeros(m2-m1+1,1);
Unorm = zeros(m2-m1+1,1);
MY = zeros(m2-m1+1,1);

I = [1 2 5 7];  %List of entries to use (if you wish to use only
                    %select entries replace i with I(i) in the data
                    %call below.
```

```
for i = m1:m2;
    [Q,Notes] = ClosedLoop051714(i);    %Retrieve Data

    t = Q(:,1);     %Select time vector
    n = length(t);

    N1 = find(Q(:,2),1,'first'); %Find the first time the right wheel turns
    N2 = find(Q(:,3),1,'first'); %Find the first time the left wheel turns
    N = min(N1,N2) - 1;%Take the time instance just before movement

    %seperate out vectors of data
    t = Q(N:n,1);
    t = t - t(1);  %Reset time scale to start at the begining of the test
    n = find(floor(t)==10);
    t = t(1:n);
    time = [0:0.01:max(t)]';    %Define a uniform time scale


    %Shift data to the new time and interpolate to get
    %   uniformly spaced entries
    Phi = Q(N:n+N-1,8);     Phi = interp1(t,Phi,time);
    V = Q(N:n+N-1,9);       V = interp1(t,V,time);
    U1 = Q(N:n+N-1,10);     U1 = interp1(t,U1,time);
    U2 = Q(N:n+N-1,11);     U2 = interp1(t,U2,time);
    pwm1 = Q(N:n+N-1,12);   pwm1 = interp1(t,pwm1,time);
    pwm2 = Q(N:n+N-1,13);   pwm2 = interp1(t,pwm2,time);

    %Redefine t as the shift and uniform time scale
    t = time;
    n = length(t);
    U = 0;


    u1 = (Volt/255)*pwm1;
    u2 = (Volt/255)*pwm2;
    Vx = - sin(Phi) .* V;
    Vy = cos(Phi) .* V;
    x = zeros(n,1);
    y = zeros(n,1);

    %Calculate the total energy used (U(i)) from the energy input to each
```

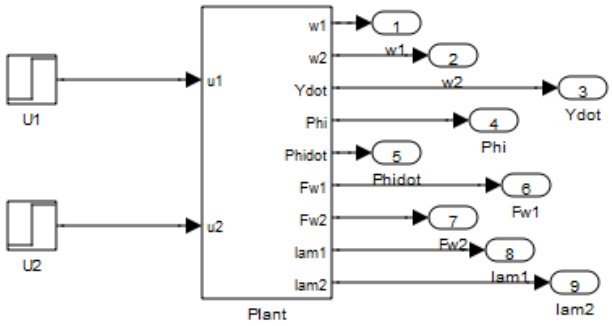
```
%motor over time.  Compute the trajectory
for j = 2:n
    U = U + 0.5*((u1(j) + u2(j)) + (u1(j-1) + u2(j-1)))*(t(j)-t(j-1));
    x(j) = x(j-1) + 0.5*(Vx(j) + Vx(j-1))*(t(j)-t(j-1));
    y(j) = y(j-1) + 0.5*(Vy(j) + Vy(j-1))*(t(j)-t(j-1));
end

%Define performance indexes
Trv(i) = t( find( abs(V-0.3) <=0.03,1,'first') ); %Time to Rise
Unorm(i) = (U)/max(t);
MaxX(i) = max(abs(x));
MaxPhi(i) = max(abs(Phi));
RecoveryTime(i) = t( find(Phi,1,'last') ) - t( find(Phi,1,'first'));

%Set the color coding for each iteration so that you can tell the
%   difference between each test
if i == 1
    K = 'b';
elseif i ==2
    K = 'g';
elseif i ==3
    K = 'r';
elseif i ==4
    K = 'c';
elseif i ==5
    K = 'm';
elseif i ==6
    K = 'y';
elseif i ==7
    K = 'k';
elseif i ==8
    K = 'b--';
elseif i ==9
    K = 'g--';
elseif i ==10
    K = 'r--';
end

%Plot trajectories
figure(1)
plot(x,y,K),title('Trajectory')
hold on
```

```
    %Plot trajectories (will be rescaled at the end of the loop)
    figure(2)
    plot(x,y,K),title('Trajectory')
    hold on
    %Plot yaw angle
    figure(3)
    plot(t,Phi,K),title('Yaw Angle')
    hold on

    MY(i) = max(y); %Determine the maximum distance traveled
end
MaxY = max(MY); %Determine which trial went the farthest

%Zoom in and label
figure(1)
legend('Case#1','Case#2','Case#3','Case#4',...
       'Case#5','Case#6','Case#7','Case#8','Case#9','Case#10')
% legend('Case#1','Case#2','Case#5','Case#7') %For use if selecting the I
% enties at the top
axis([-0.2 0.2 0 3.5])
xlabel('X-Position (m)'),ylabel('Y-Position (m)')
grid on

%Zoom out and label
figure(2)
legend('Case#1','Case#2','Case#3','Case#4',...
       'Case#5','Case#6','Case#7','Case#8','Case#9','Case#10')
% legend('Case#1','Case#2','Case#5','Case#7')
axis([-1,1,0,4])
xlabel('X-Position (m)'),ylabel('Y-Position (m)')
grid on

%Use suitible scale and label
figure(3)
legend('Case#1','Case#2','Case#3','Case#4',...
       'Case#5','Case#6','Case#7','Case#8','Case#9','Case#10')
% legend('Case#1','Case#2','Case#5','Case#7')
xlabel('Time (sec)'),ylabel('Rads')
axis([0 max(t) -0.1 0.1])
grid on

%Display matrix of Results
```

```
disp('Kpv -- Kiv -- Kpphi -- Kiphi -- MaxPhi(rads) -- MaxX(m) -- TR(sec) -- Unorm(J) -- Trise(sec)')
disp([Gains(m1:m2,:),MaxPhi,MaxX,RecoveryTime,Unorm,Trv])
```

D.3 Linear Simulink Model

LinearDynamics



Ryan

26-May-2014 22:55:27

Table of Contents

Model - LinearDynamics
System - LinearDynamics
System - LinearDynamics/Plant
System - LinearDynamics/Plant/Bulk Vehicle Dynamics
System - LinearDynamics/Plant/Bulk Vehicle Dynamics/Rotation Dynamics
System - LinearDynamics/Plant/Bulk Vehicle Dynamics/Translation Dynamics
System - LinearDynamics/Plant/Motor and Traction Dynamics
System - LinearDynamics/Plant/Motor and Traction Dynamics/Motor Dynamics
System - LinearDynamics/Plant/Motor and Traction Dynamics/Motor Dynamics/Motor/Wheel Dynamics1
System - LinearDynamics/Plant/Motor and Traction Dynamics/Motor Dynamics/Motor/Wheel Dynamics2
System - LinearDynamics/Plant/Motor and Traction Dynamics/Slip And Traction
System - LinearDynamics/Plant/Motor and Traction Dynamics/Slip And Traction/Linear Traction Estimation
System - LinearDynamics/Plant/Motor and Traction Dynamics/Slip And Traction/Linear Traction Estimation1
System - LinearDynamics/Plant/Motor and Traction Dynamics/Slip And Traction/Slip Estimation1
System - LinearDynamics/Plant/Motor and Traction Dynamics/Slip And Traction/Slip Estimation1/Determine Max Magnitude
System - LinearDynamics/Plant/Motor and Traction Dynamics/Slip And Traction/Slip Estimation2
System - LinearDynamics/Plant/Motor and Traction Dynamics/Slip And Traction/Slip Estimation2/Determine Max Magnitude
System - LinearDynamics/Plant/Motor and Traction Dynamics/Wheel Velocity Estimator
Appendix

List of Tables

1. Outport Block Properties
2. Step Block Properties

3. Inport Block Properties
4. Outport Block Properties
5. Inport Block Properties
6. Outport Block Properties
7. Gain Block Properties
8. Inport Block Properties
9. Integrator Block Properties
10. Outport Block Properties
11. Sum Block Properties
12. Gain Block Properties
13. Inport Block Properties
14. Integrator Block Properties
15. Outport Block Properties
16. Sum Block Properties
17. Inport Block Properties
18. Outport Block Properties
19. Inport Block Properties
20. Outport Block Properties
21. Inport Block Properties
22. Outport Block Properties
23. Sum Block Properties
24. TransferFcn Block Properties
25. Inport Block Properties
26. Outport Block Properties
27. Sum Block Properties
28. TransferFcn Block Properties
29. Inport Block Properties
30. Outport Block Properties
31. Gain Block Properties
32. Inport Block Properties
33. Outport Block Properties
34. Gain Block Properties
35. Inport Block Properties
36. Outport Block Properties
37. Constant Block Properties
38. Fcn Block Properties
39. Gain Block Properties
40. Inport Block Properties
41. Mux Block Properties
42. Outport Block Properties
43. Switch Block Properties
44. Abs Block Properties
45. Inport Block Properties
46. MinMax Block Properties
47. Outport Block Properties
48. Product Block Properties
49. Signum Block Properties
50. Constant Block Properties
51. Fcn Block Properties

52. Gain Block Properties
53. Inport Block Properties
54. Mux Block Properties
55. Outport Block Properties
56. Switch Block Properties
57. Abs Block Properties
58. Inport Block Properties
59. MinMax Block Properties
60. Outport Block Properties
61. Product Block Properties
62. Signum Block Properties
63. Fcn Block Properties
64. Inport Block Properties
65. Mux Block Properties
66. Outport Block Properties
67. Block Type Count

Model - LinearDynamics

Full Model Hierarchy

1. LinearDynamics
    1. Plant
        1. Bulk Vehicle Dynamics
            1. Rotation Dynamics
            2. Translation Dynamics
        2. Motor and Traction Dynamics
            1. Motor Dynamics
                1. Motor/Wheel Dynamics1
                2. Motor/Wheel Dynamics2
            2. Slip And Traction
                1. Linear Traction Estimation
                2. Linear Traction Estimation1
                3. Slip Estimation1
                    1. Determine Max Magnitude
                4. Slip Estimation2
                    1. Determine Max Magnitude
            3. Wheel Velocity Estimator

Simulation Parameter	Value
Solver	ode45
RelTol	1e-3
Refine	1
MaxOrder	5
ZeroCross	on

[more info]

System - LinearDynamics

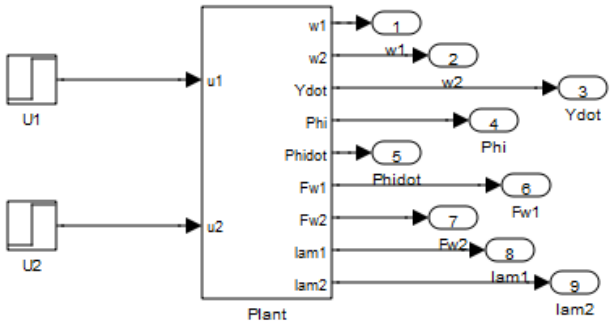


Table 1. Outport Block Properties

Name	Port	Icon Display	Lock Scale	Var Size Sig	Signal Type	Sampling Mode	Source Of Initial Output Value	Output When Disabled	Used By Blk
Fw1	6	Port number	off	Inherit	auto	auto	Dialog	held	LinearDynamics (model)
Fw2	7	Port number	off	Inherit	auto	auto	Dialog	held	LinearDynamics (model)
lam1	8	Port number	off	Inherit	auto	auto	Dialog	held	LinearDynamics (model)
lam2	9	Port number	off	Inherit	auto	auto	Dialog	held	LinearDynamics (model)
Phi	4	Port number	off	Inherit	auto	auto	Dialog	held	LinearDynamics (model)
Phidot	5	Port number	off	Inherit	auto	auto	Dialog	held	LinearDynamics (model)
w1	1	Port number	off	Inherit	auto	auto	Dialog	held	LinearDynamics (model)
w2	2	Port number	off	Inherit	auto	auto	Dialog	held	LinearDynamics (model)
Ydot	3	Port	off	Inherit	auto	auto	Dialog	held	LinearDynamics

		number							(model)

Table 2. Step Block Properties

Name	Time	Before	After	Sample Time	Zero Cross
U1	0	0	U1	0	on
U2	0	0	U2	0	on

System - LinearDynamics/Plant

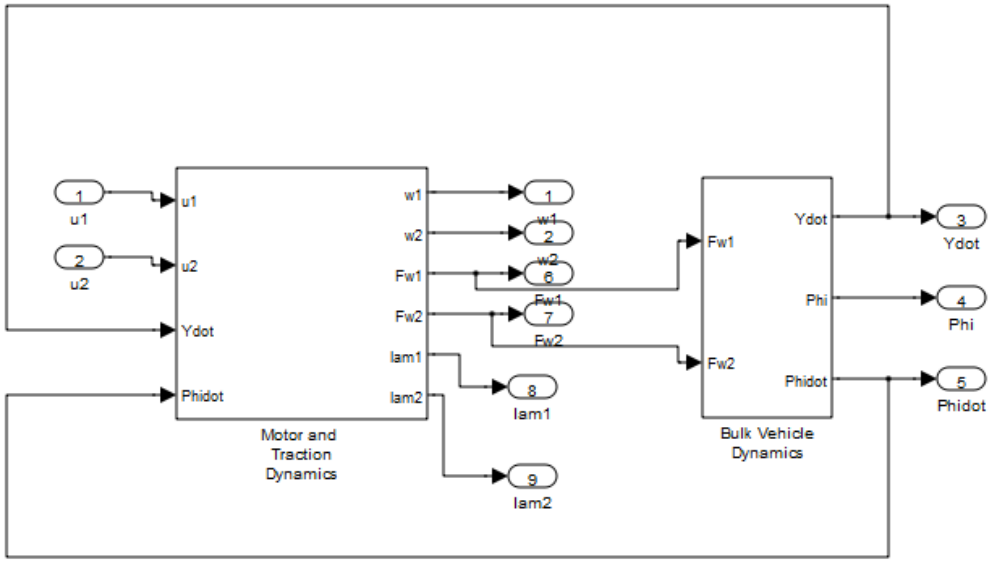


Table 3. Inport Block Properties

Name	Port	Defined In Blk
u1	1	U1
u2	2	U2

Table 4. Outport Block Properties

Name	Port	Icon Display	Lock Scale	Var Size Sig	Signal Type	Sampling Mode	Source Of Initial Output Value	Output When Disabled	Used By Blk
Fw1	6	Port number	off	Inherit	auto	auto	Dialog	held	Sum, Sum, LinearDynamics (model)
Fw2	7	Port number	off	Inherit	auto	auto	Dialog	held	Sum, Sum, LinearDynamics (model)

lam1	8	Port number	off	Inherit	auto	auto	Dialog	held	LinearDynamics (model)
lam2	9	Port number	off	Inherit	auto	auto	Dialog	held	LinearDynamics (model)
Phi	4	Port number	off	Inherit	auto	auto	Dialog	held	LinearDynamics (model)
Phidot	5	Port number	off	Inherit	auto	auto	Dialog	held	LinearDynamics (model), Fcn1, Fcn
w1	1	Port number	off	Inherit	auto	auto	Dialog	held	LinearDynamics (model)
w2	2	Port number	off	Inherit	auto	auto	Dialog	held	LinearDynamics (model)
Ydot	3	Port number	off	Inherit	auto	auto	Dialog	held	LinearDynamics (model), Fcn1, Fcn

System - LinearDynamics/Plant/Bulk Vehicle Dynamics

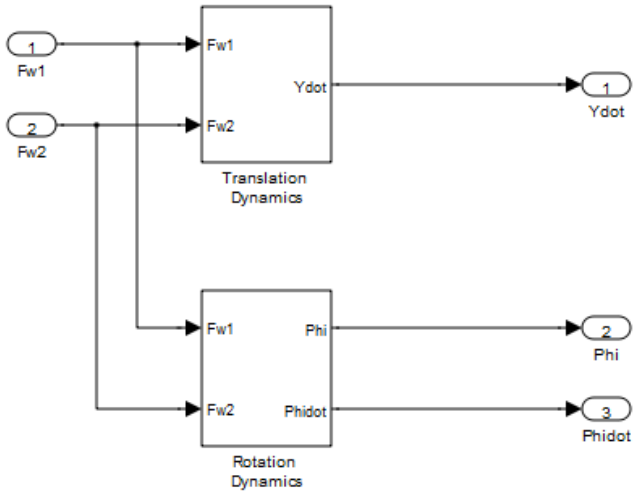


Table 5. Inport Block Properties

Name	Port	Defined In Blk
Fw1	1	Gain
Fw2	2	Gain

Table 6. Outport Block Properties

Name	Port	Icon Display	Lock Scale	Var Size	Signal Type	Sampling Mode	Source Of Initial Output Value	Output When Disabled	Used By Blk

				Sig					
Phi	2	Port number	off	Inherit	auto	auto	Dialog	held	LinearDynamics (model)
Phidot	3	Port number	off	Inherit	auto	auto	Dialog	held	LinearDynamics (model), Fcn1, Fcn
Ydot	1	Port number	off	Inherit	auto	auto	Dialog	held	LinearDynamics (model), Fcn1, Fcn

System - LinearDynamics/Plant/Bulk Vehicle Dynamics/Rotation Dynamics

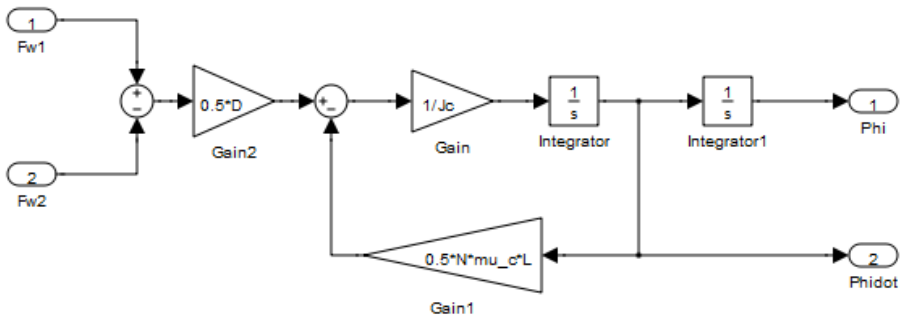


Table 7. Gain Block Properties

Name	Gain	Multiplication	Param Data Type Str	Out Data Type Str	Lock Scale	Rnd Meth	Saturate On Integer Overflow
Gain	1/Jc	Element-wise(K.*u)	Inherit: Inherit via internal rule	Inherit: Inherit via internal rule	off	Floor	off
Gain1	0.5*N*mu_c*L	Element-wise(K.*u)	Inherit: Inherit via internal rule	Inherit: Inherit via internal rule	off	Floor	off
Gain2	0.5*D	Element-wise(K.*u)	Inherit: Inherit via internal rule	Inherit: Inherit via internal rule	off	Floor	off

Table 8. Inport Block Properties

Name	Port	Defined In Blk
Fw1	1	Gain
Fw2	2	Gain

Table 9. Integrator Block Properties

Name	External Reset	Initial Condition Source	Absolute Tolerance	Zero Cross	Continuous State Attributes
Integrator	none	internal	auto	on	''
Integrator1	none	internal	auto	on	''

Table 10. Outport Block Properties

Name	Port	Icon Display	Lock Scale	Var Size Sig	Signal Type	Sampling Mode	Source Of Initial Output Value	Output When Disabled	Used By Blk
Phi	1	Port number	off	Inherit	auto	auto	Dialog	held	LinearDynamics (model)
Phidot	2	Port number	off	Inherit	auto	auto	Dialog	held	LinearDynamics (model), Fcn1, Fcn, Integrator1, Gain1

Table 11. Sum Block Properties

Name	Icon Shape	Inputs	Collapse Mode	Collapse Dim	Input Same DT	Accum Data Type Str	Out Data Type Str	Lock Scale	Rnd Meth	Saturate On Integer Overflow
Sum	round	+|-	All dimensions	1	off	Inherit: Inherit via internal rule	Inherit: Inherit via internal rule	off	Floor	off
Sum1	round	|+-	All dimensions	1	off	Inherit: Inherit via internal rule	Inherit: Inherit via internal rule	off	Floor	off

System - LinearDynamics/Plant/Bulk Vehicle Dynamics/Translation Dynamics

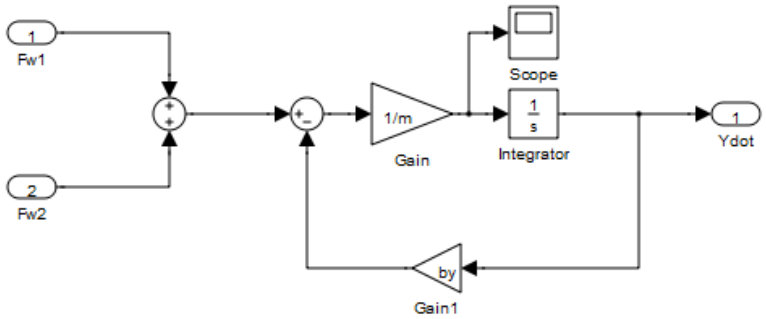


Table 12. Gain Block Properties

Name	Gain	Multiplication	Param Data Type Str	Out Data Type Str	Lock Scale	Rnd Meth	Saturate On Integer Overflow
Gain	1/m	Element-wise(K.*u)	Inherit: Inherit via internal rule	Inherit: Inherit via internal rule	off	Floor	off
Gain1	by	Element-wise(K.*u)	Inherit: Inherit via internal rule	Inherit: Inherit via internal rule	off	Floor	off

Table 13. Inport Block Properties

Name	Port	Defined In Blk
Fw1	1	Gain
Fw2	2	Gain

Table 14. Integrator Block Properties

Name	External Reset	Initial Condition Source	Absolute Tolerance	Zero Cross	Continuous State Attributes
Integrator	none	internal	auto	on	''

Table 15. Outport Block Properties

Name	Port	Icon Display	Lock Scale	Var Size Sig	Signal Type	Sampling Mode	Source Of Initial Output Value	Output When Disabled	Used By Blk
Ydot	1	Port number	off	Inherit	auto	auto	Dialog	held	LinearDynamics (model), Fcn1, Fcn, Gain1

Table 16. Sum Block Properties

Name	Icon Shape	Inputs	Collapse Mode	Collapse Dim	Input Same DT	Accum Data Type Str	Out Data Type Str	Lock Scale	Rnd Meth	Saturate On Integer Overflow
Sum	round	+|+	All dimensions	1	off	Inherit: Inherit via internal rule	Inherit: Inherit via internal rule	off	Floor	off
Sum1	round	|+-	All dimensions	1	off	Inherit: Inherit via internal rule	Inherit: Inherit via internal rule	off	Floor	off

System - LinearDynamics/Plant/Motor and Traction Dynamics

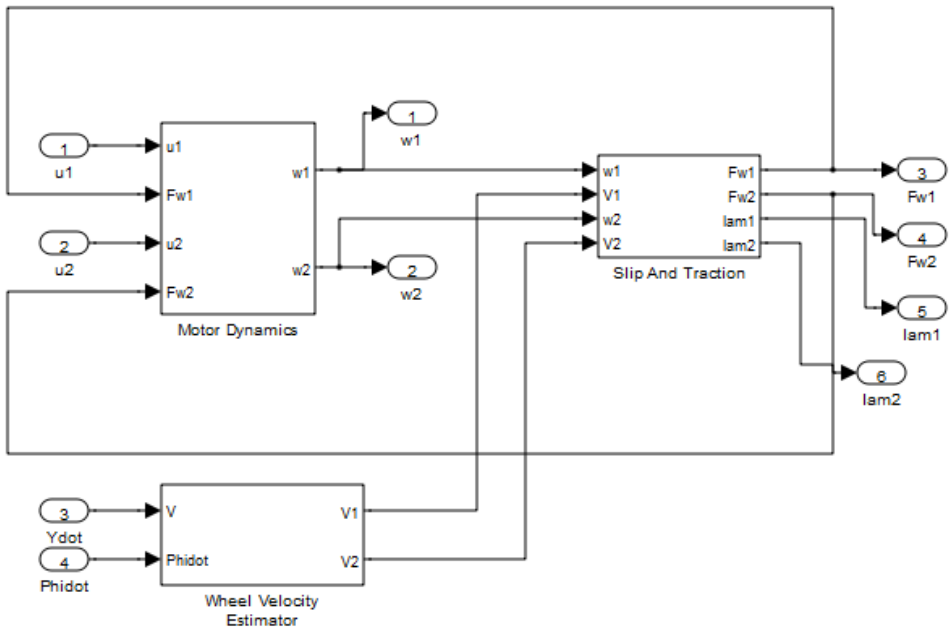


Table 17. Inport Block Properties

Name	Port	Defined In Blk
Phidot	4	Integrator
u1	1	U1
u2	2	U2
Ydot	3	Integrator

Table 18. Outport Block Properties

Name	Port	Icon Display	Lock Scale	Var Size Sig	Signal Type	Sampling Mode	Source Of Initial Output Value	Output When Disabled	Used By Blk
Fw1	3	Port number	off	Inherit	auto	auto	Dialog	held	Traction, Sum, Sum, LinearDynamics (model)
Fw2	4	Port number	off	Inherit	auto	auto	Dialog	held	Transfer Fcn1, Sum, Sum, LinearDynamics (model)
lam1	5	Port number	off	Inherit	auto	auto	Dialog	held	LinearDynamics (model)
lam2	6	Port number	off	Inherit	auto	auto	Dialog	held	LinearDynamics (model)
w1	1	Port number	off	Inherit	auto	auto	Dialog	held	Gain, LinearDynamics (model)
w2	2	Port	off	Inherit	auto	auto	Dialog	held	Gain, LinearDynamics (model)

		number							

System - LinearDynamics/Plant/Motor and Traction Dynamics/Motor Dynamics

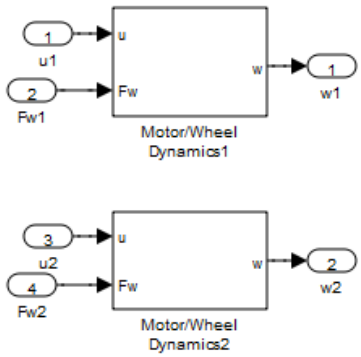


Table 19. Inport Block Properties

Name	Port	Defined In Blk
Fw1	2	Gain
Fw2	4	Gain
u1	1	U1
u2	3	U2

Table 20. Outport Block Properties

Name	Port	Icon Display	Lock Scale	Var Size Sig	Signal Type	Sampling Mode	Source Of Initial Output Value	Output When Disabled	Used By Blk
w1	1	Port number	off	Inherit	auto	auto	Dialog	held	Gain, LinearDynamics (model)
w2	2	Port number	off	Inherit	auto	auto	Dialog	held	Gain, LinearDynamics (model)

System - LinearDynamics/Plant/Motor and Traction Dynamics/Motor Dynamics/Motor/Wheel Dynamics1

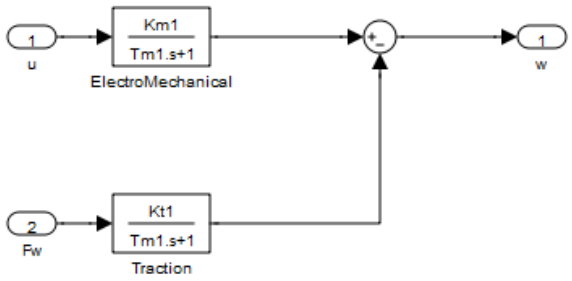


Table 21. Inport Block Properties

Name	Port	Defined In Blk
Fw	2	Gain
u	1	U1

Table 22. Outport Block Properties

Name	Port	Icon Display	Lock Scale	Var Size Sig	Signal Type	Sampling Mode	Source Of Initial Output Value	Output When Disabled	Used By Blk
w	1	Port number	off	Inherit	auto	auto	Dialog	held	Gain, LinearDynamics (model)

Table 23. Sum Block Properties

Name	Icon Shape	Inputs	Collapse Mode	Collapse Dim	Input Same DT	Accum Data Type Str	Out Data Type Str	Lock Scale	Rnd Meth	Saturate On Integer Overflow
Sum	round	|+-	All dimensions	1	off	Inherit: Inherit via internal rule	Inherit: Inherit via internal rule	off	Floor	off

Table 24. TransferFcn Block Properties

Name	Numerator	Denominator	Absolute Tolerance	Continuous State Attributes
ElectroMechanical	[Km1]	[Tm1 1]	auto	''
Traction	[Kt1]	[Tm1 1]	auto	''

System - LinearDynamics/Plant/Motor and Traction Dynamics/Motor Dynamics/Motor/Wheel Dynamics2

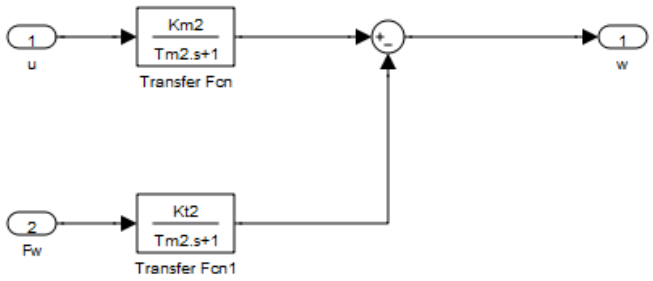


Table 25. Inport Block Properties

Name	Port	Defined In Blk
Fw	2	Gain
u	1	U2

Table 26. Outport Block Properties

Name	Port	Icon Display	Lock Scale	Var Size Sig	Signal Type	Sampling Mode	Source Of Initial Output Value	Output When Disabled	Used By Blk
w	1	Port number	off	Inherit	auto	auto	Dialog	held	Gain, LinearDynamics (model)

Table 27. Sum Block Properties

Name	Icon Shape	Inputs	Collapse Mode	Collapse Dim	Input Same DT	Accum Data Type Str	Out Data Type Str	Lock Scale	Rnd Meth	Saturate On Integer Overflow
Sum	round	|+-	All dimensions	1	off	Inherit: Inherit via internal rule	Inherit: Inherit via internal rule	off	Floor	off

Table 28. TransferFcn Block Properties

Name	Numerator	Denominator	Absolute Tolerance	Continuous State Attributes
Transfer Fcn	[Km2]	[Tm2 1]	auto	''
Transfer Fcn1	[Kt2]	[Tm2 1]	auto	''

System - LinearDynamics/Plant/Motor and Traction Dynamics/Slip And Traction

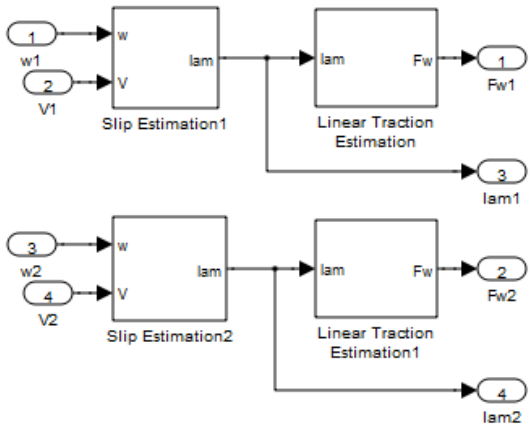


Table 29. Inport Block Properties

Name	Port	Defined In Blk
V1	2	Fcn
V2	4	Fcn1
w1	1	Sum
w2	3	Sum

Table 30. Outport Block Properties

Name	Port	Icon Display	Lock Scale	Var Size Sig	Signal Type	Sampling Mode	Source Of Initial Output Value	Output When Disabled	Used By Blk
Fw1	1	Port number	off	Inherit	auto	auto	Dialog	held	Traction, Sum, Sum, LinearDynamics (model)
Fw2	2	Port number	off	Inherit	auto	auto	Dialog	held	Transfer Fcn1, Sum, Sum, LinearDynamics (model)
lam1	3	Port number	off	Inherit	auto	auto	Dialog	held	LinearDynamics (model), Gain
lam2	4	Port number	off	Inherit	auto	auto	Dialog	held	LinearDynamics (model), Gain

System - LinearDynamics/Plant/Motor and Traction Dynamics/Slip And Traction/Linear Traction Estimation

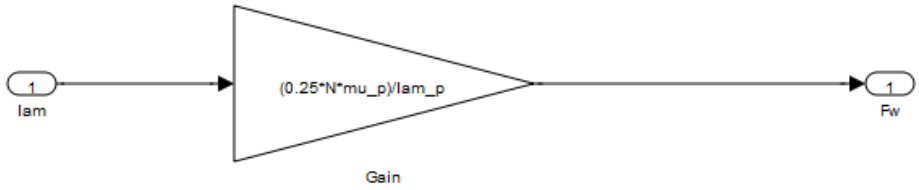


Table 31. Gain Block Properties

Name	Gain	Multiplication	Param Data Type Str	Out Data Type Str	Lock Scale	Rnd Meth	Saturate On Integer Overflow
Gain	(0.25*N*mu_p)/lam_p	Element-wise(K.*u)	Inherit: Inherit via internal rule	Inherit: Inherit via internal rule	off	Floor	off

Table 32. Inport Block Properties

Name	Port	Defined In Blk
lam	1	Switch1

Table 33. Outport Block Properties

Name	Port	Icon Display	Lock Scale	Var Size Sig	Signal Type	Sampling Mode	Source Of Initial Output Value	Output When Disabled	Used By Blk
Fw	1	Port number	off	Inherit	auto	auto	Dialog	held	Traction, Sum, Sum, LinearDynamics (model)

System - LinearDynamics/Plant/Motor and Traction Dynamics/Slip And Traction/Linear Traction Estimation1

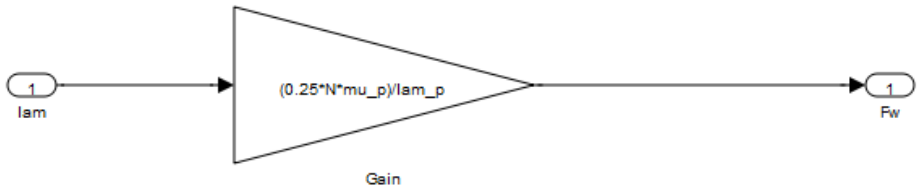


Table 34. Gain Block Properties

Name	Gain	Multiplication	Param Data Type Str	Out Data Type Str	Lock Scale	Rnd Meth	Saturate On Integer Overflow
Gain	(0.25*N*mu_p)/lam_p	Element-wise(K.*u)	Inherit: Inherit via internal rule	Inherit: Inherit via internal rule	off	Floor	off

Table 35. Inport Block Properties

Name	Port	Defined In Blk
lam	1	Switch1

Table 36. Outport Block Properties

Name	Port	Icon Display	Lock Scale	Var Size Sig	Signal Type	Sampling Mode	Source Of Initial Output Value	Output When Disabled	Used By Blk
Fw	1	Port number	off	Inherit	auto	auto	Dialog	held	Transfer Fcn1, Sum, Sum, LinearDynamics (model)

System - LinearDynamics/Plant/Motor and Traction Dynamics/Slip And Traction/Slip Estimation1

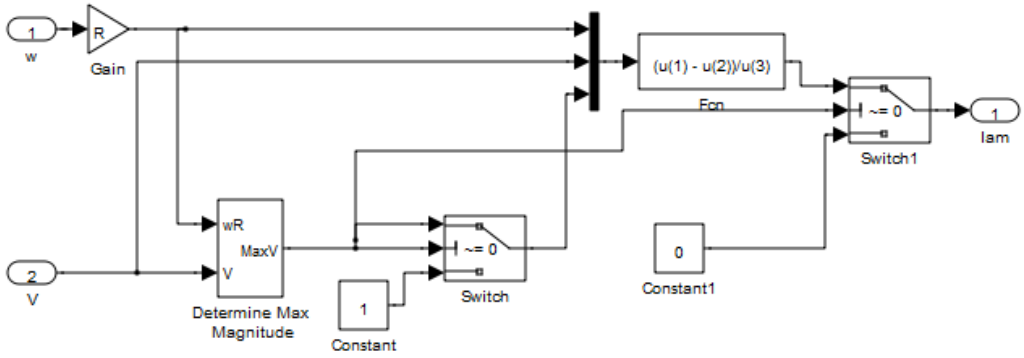


Table 37. Constant Block Properties

Name	Value	Sampling Mode	Out Data Type Str	Lock Scale	Sample Time	Frame Period
Constant	1	Sample based	Inherit: Inherit from 'Constant value'	off	inf	inf
Constant1	0	Sample based	Inherit: Inherit from 'Constant value'	off	inf	inf

Table 38. Fcn Block Properties

Name	Expr
Fcn	(u(1) - u(2))/u(3)

Table 39. Gain Block Properties

Name	Gain	Multiplication	Param Data Type Str	Out Data Type Str	Lock Scale	Rnd Meth	Saturate On Integer Overflow
		Element-	Inherit: Inherit via	Inherit: Inherit via			

Gain	R	wise(K.*u)	internal rule	internal rule	off	Floor	off

Table 40. Inport Block Properties

Name	Port	Defined In Blk
V	2	Fcn
w	1	Sum

Table 41. Mux Block Properties

Name	Inputs	Display Option
Mux	3	bar

Table 42. Outport Block Properties

Name	Port	Icon Display	Lock Scale	Var Size Sig	Signal Type	Sampling Mode	Source Of Initial Output Value	Output When Disabled	Used By Blk
lam	1	Port number	off	Inherit	auto	auto	Dialog	held	LinearDynamics (model), Gain

Table 43. Switch Block Properties

Name	Criteria	Threshold	Input Same DT	Out Data Type Str	Lock Scale	Rnd Meth	Saturate On Integer Overflow	Zero Cross	Allow Diff Input Sizes
Switch	u2 ~= 0	0	off	Inherit: Inherit via internal rule	off	Floor	off	on	off
Switch1	u2 ~= 0	0	off	Inherit: Inherit via internal rule	off	Floor	off	on	off

System - LinearDynamics/Plant/Motor and Traction Dynamics/Slip And Traction/Slip Estimation1/Determine Max Magnitude

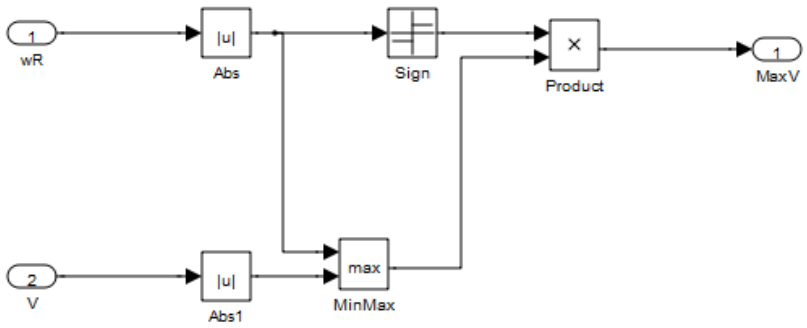


Table 44. Abs Block Properties

Name	Zero Cross	Out Data Type Str	Lock Scale	Rnd Meth	Saturate On Integer Overflow
Abs	on	Inherit: Same as input	off	Floor	off
Abs1	on	Inherit: Same as input	off	Floor	off

Table 45. Inport Block Properties

Name	Port	Defined In Blk
V	2	Fcn
wR	1	Gain

Table 46. MinMax Block Properties

Name	Function	Inputs	Input Same DT	Out Data Type Str	Lock Scale	Rnd Meth	Saturate On Integer Overflow	Zero Cross
MinMax	max	2	off	Inherit: Inherit via internal rule	off	Floor	off	on

Table 47. Outport Block Properties

Name	Port	Icon Display	Lock Scale	Var Size Sig	Signal Type	Sampling Mode	Source Of Initial Output Value	Output When Disabled	Used By Blk
MaxV	1	Port number	off	Inherit	auto	auto	Dialog	held	Switch, Switch, Switch1

Table 48. Product Block Properties

Name	Inputs	Multiplication	Collapse Mode	Collapse Dim	Input Same DT	Out Data Type Str	Lock Scale	Rnd Meth	Saturate On Integer Overflow
Product	2	Element-wise(.*)	All dimensions	1	off	Inherit: Inherit via internal rule	off	Zero	off

Table 49. Signum Block Properties

Name	Zero Cross
Sign	on

System - LinearDynamics/Plant/Motor and Traction Dynamics/Slip And Traction/Slip Estimation2

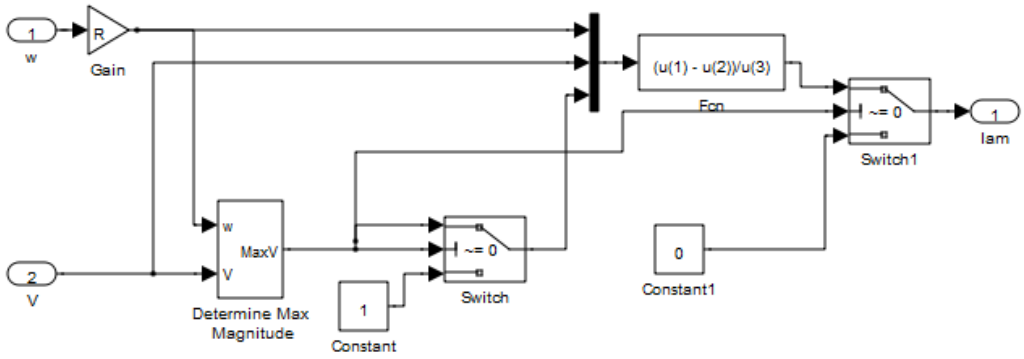


Table 50. Constant Block Properties

Name	Value	Sampling Mode	Out Data Type Str	Lock Scale	Sample Time	Frame Period
Constant	1	Sample based	Inherit: Inherit from 'Constant value'	off	inf	inf
Constant1	0	Sample based	Inherit: Inherit from 'Constant value'	off	inf	inf

Table 51. Fcn Block Properties

Name	Expr
Fcn	(u(1) - u(2))/u(3)

Table 52. Gain Block Properties

Name	Gain	Multiplication	Param Data Type Str	Out Data Type Str	Lock Scale	Rnd Meth	Saturate On Integer Overflow
Gain	R	Element-wise(K.*u)	Inherit: Inherit via internal rule	Inherit: Inherit via internal rule	off	Floor	off

Table 53. Inport Block Properties

Name	Port	Defined In Blk
V	2	Fcn1
w	1	Sum

Table 54. Mux Block Properties

Name	Inputs	Display Option
Mux	3	bar

Table 55. Outport Block Properties

Name	Port	Icon Display	Lock Scale	Var Size Sig	Signal Type	Sampling Mode	Source Of Initial Output Value	Output When Disabled	Used By Blk

lam	1	Port number	off	Inherit	auto	auto	Dialog	held	LinearDynamics (model), Gain

Table 56. Switch Block Properties

Name	Criteria	Threshold	Input Same DT	Out Data Type Str	Lock Scale	Rnd Meth	Saturate On Integer Overflow	Zero Cross	Allow Diff Input Sizes
Switch	u2 ~= 0	0	off	Inherit: Inherit via internal rule	off	Floor	off	on	off
Switch1	u2 ~= 0	0	off	Inherit: Inherit via internal rule	off	Floor	off	on	off

System - LinearDynamics/Plant/Motor and Traction Dynamics/Slip And Traction/Slip Estimation2/Determine Max Magnitude

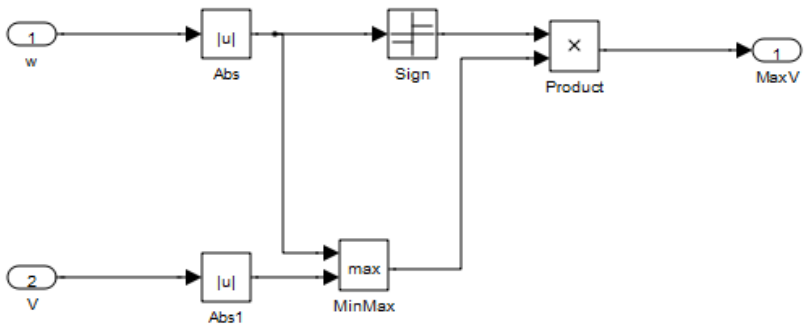


Table 57. Abs Block Properties

Name	Zero Cross	Out Data Type Str	Lock Scale	Rnd Meth	Saturate On Integer Overflow
Abs	on	Inherit: Same as input	off	Floor	off
Abs1	on	Inherit: Same as input	off	Floor	off

Table 58. Inport Block Properties

Name	Port	Defined In Blk
V	2	Fcn1
w	1	Gain

Table 59. MinMax Block Properties

Name	Function	Inputs	Input Same DT	Out Data Type Str	Lock Scale	Rnd Meth	Saturate On Integer Overflow	Zero Cross
MinMax	max	2	off	Inherit: Inherit via internal rule	off	Floor	off	on

Table 60. Outport Block Properties

Name	Port	Icon Display	Lock Scale	Var Size Sig	Signal Type	Sampling Mode	Source Of Initial Output Value	Output When Disabled	Used By Blk
MaxV	1	Port number	off	Inherit	auto	auto	Dialog	held	Switch, Switch, Switch1

Table 61. Product Block Properties

Name	Inputs	Multiplication	Collapse Mode	Collapse Dim	Input Same DT	Out Data Type Str	Lock Scale	Rnd Meth	Saturate On Integer Overflow
Product	2	Element-wise(.*)	All dimensions	1	off	Inherit: Inherit via internal rule	off	Zero	off

Table 62. Signum Block Properties

Name	Zero Cross
Sign	on

System - LinearDynamics/Plant/Motor and Traction Dynamics/Wheel Velocity Estimator

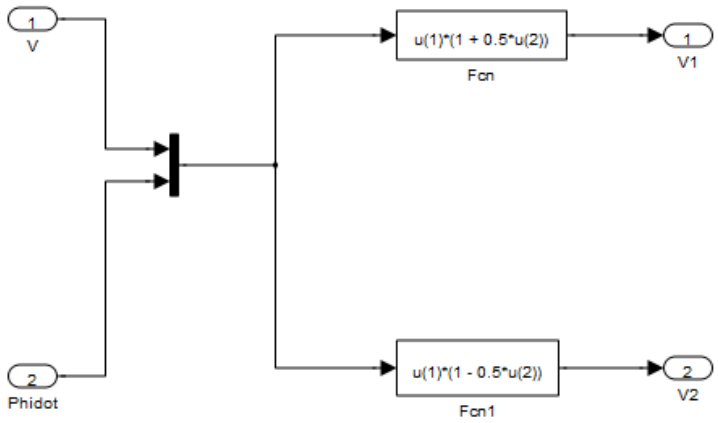


Table 63. Fcn Block Properties

Name	Expr
Fcn	u(1)*(1 + 0.5*u(2))
Fcn1	u(1)*(1 - 0.5*u(2))

Table 64. Inport Block Properties

Name	Port	Defined In Blk

Phidot	2	Integrator
V	1	Integrator

Table 65. Mux Block Properties

Name	Inputs	Display Option
Mux	2	bar

Table 66. Outport Block Properties

Name	Port	Icon Display	Lock Scale	Var Size Sig	Signal Type	Sampling Mode	Source Of Initial Output Value	Output When Disabled	Used By Blk
V1	1	Port number	off	Inherit	auto	auto	Dialog	held	Fcn, Abs1
V2	2	Port number	off	Inherit	auto	auto	Dialog	held	Fcn, Abs1

Appendix

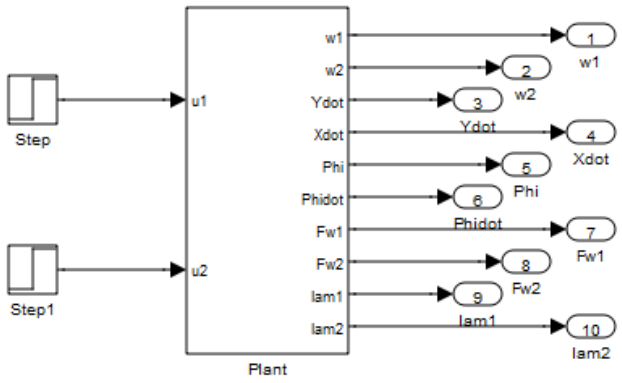
Table 67. Block Type Count

BlockType	Count	Block Names
Outport	46	Fw1, Fw2, Phi, Phidot, Phi, Phidot, Phi, Phidot, Ydot, Ydot, Fw1, Fw2, Fw1, Fw2, w, w, w1, w2, Fw1, Fw2, Fw, Fw, MaxV, lam, MaxV, lam, lam1, lam2, V1, V2, lam1, lam2, w1, w2, Phi, Phidot, Ydot, lam1, lam2, w1, w2, Ydot, lam1, lam2, w1, w2
Inport	36	Fw1, Fw2, Fw1, Fw2, Fw1, Fw2, Fw1, Fw2, Fw, u, Fw, u, u1, u2, Phidot, lam, lam, V, wR, V, w, V, w, V, w, V1, V2, w1, w2, Phidot, V, Ydot, u1, u2, u1, u2
SubSystem	16	Plant, Bulk Vehicle Dynamics, Rotation Dynamics, Translation Dynamics, Motor and Traction Dynamics, Motor Dynamics, Motor/Wheel Dynamics1, Motor/Wheel Dynamics2, Slip And Traction, Linear Traction Estimation, Linear Traction Estimation1, Slip Estimation1, Determine Max Magnitude, Slip Estimation2, Determine Max Magnitude, Wheel Velocity Estimator
Gain	9	Gain, Gain1, Gain2, Gain, Gain1, Gain, Gain, Gain, Gain
Sum	6	Sum, Sum1, Sum, Sum1, Sum, Sum
TransferFcn	4	ElectroMechanical, Traction, Transfer Fcn, Transfer Fcn1
Switch	4	Switch, Switch1, Switch, Switch1
Fcn	4	Fcn, Fcn, Fcn, Fcn1
Constant	4	Constant, Constant1, Constant, Constant1
Abs	4	Abs, Abs1, Abs, Abs1
Mux	3	Mux, Mux, Mux
Integrator	3	Integrator, Integrator1, Integrator
Step	2	U1, U2
Signum	2	Sign, Sign
Product	2	Product, Product
MinMax	2	MinMax, MinMax

Scope	1	Scope

D.4 Nonlinear Simulink Model

NonlinearDynamics



Ryan

26-May-2014 22:05:27

Table of Contents

Model - NonlinearDynamics
System - NonlinearDynamics
System - NonlinearDynamics/Plant
System - NonlinearDynamics/Plant/Bulk Vehicle Dynamics
System - NonlinearDynamics/Plant/Bulk Vehicle Dynamics/Lateral Slide Dynamics
System - NonlinearDynamics/Plant/Bulk Vehicle Dynamics/Rotation Dynamics
System - NonlinearDynamics/Plant/Bulk Vehicle Dynamics/Translation Dynamics
System - NonlinearDynamics/Plant/Motor and Traction Dynamics
System - NonlinearDynamics/Plant/Motor and Traction Dynamics/Motor Dynamics
System - NonlinearDynamics/Plant/Motor and Traction Dynamics/Motor Dynamics/Motor/Wheel Dynamics1
System - NonlinearDynamics/Plant/Motor and Traction Dynamics/Motor Dynamics/Motor/Wheel Dynamics2
System - NonlinearDynamics/Plant/Motor and Traction Dynamics/Slip And Traction
System - NonlinearDynamics/Plant/Motor and Traction Dynamics/Slip And Traction/Slip Estimation1
System - NonlinearDynamics/Plant/Motor and Traction Dynamics/Slip And Traction/Slip Estimation1/Determine Max Magnitude
System - NonlinearDynamics/Plant/Motor and Traction Dynamics/Slip And Traction/Slip Estimation2
System - NonlinearDynamics/Plant/Motor and Traction Dynamics/Slip And Traction/Slip Estimation2/Determine Max Magnitude
System - NonlinearDynamics/Plant/Motor and Traction Dynamics/Slip And Traction/Traction Estimation1
System - NonlinearDynamics/Plant/Motor and Traction Dynamics/Slip And Traction/Traction Estimation2

System - NonlinearDynamics/Plant/Motor and Traction Dynamics/Wheel Velocity Estimator
Appendix

List of Tables

1. Outport Block Properties
2. Step Block Properties
3. Inport Block Properties
4. Outport Block Properties
5. Inport Block Properties
6. Outport Block Properties
7. Gain Block Properties
8. Inport Block Properties
9. Integrator Block Properties
10. Outport Block Properties
11. Product Block Properties
12. Saturate Block Properties
13. Sum Block Properties
14. Gain Block Properties
15. Inport Block Properties
16. Integrator Block Properties
17. Outport Block Properties
18. Saturate Block Properties
19. Sum Block Properties
20. Gain Block Properties
21. Inport Block Properties
22. Integrator Block Properties
23. Outport Block Properties
24. Sum Block Properties
25. Inport Block Properties
26. Outport Block Properties
27. Inport Block Properties
28. Outport Block Properties
29. Inport Block Properties
30. Outport Block Properties
31. Sum Block Properties
32. TransferFcn Block Properties
33. Inport Block Properties
34. Outport Block Properties
35. Sum Block Properties
36. TransferFcn Block Properties
37. Inport Block Properties
38. Outport Block Properties
39. Constant Block Properties
40. Fcn Block Properties
41. Gain Block Properties

42. Inport Block Properties
43. Mux Block Properties
44. Outport Block Properties
45. Switch Block Properties
46. Abs Block Properties
47. Inport Block Properties
48. MinMax Block Properties
49. Outport Block Properties
50. Product Block Properties
51. Signum Block Properties
52. Constant Block Properties
53. Fcn Block Properties
54. Gain Block Properties
55. Inport Block Properties
56. Mux Block Properties
57. Outport Block Properties
58. Switch Block Properties
59. Abs Block Properties
60. Inport Block Properties
61. MinMax Block Properties
62. Outport Block Properties
63. Product Block Properties
64. Signum Block Properties
65. Fcn Block Properties
66. Inport Block Properties
67. Outport Block Properties
68. Fcn Block Properties
69. Inport Block Properties
70. Outport Block Properties
71. Fcn Block Properties
72. Inport Block Properties
73. Mux Block Properties
74. Outport Block Properties
75. Block Type Count

Model - NonlinearDynamics

Full Model Hierarchy

1. NonlinearDynamics
    1. Plant
        1. Bulk Vehicle Dynamics
            1. Lateral Slide Dynamics

2. Rotation Dynamics
3. Translation Dynamics

2. Motor and Traction Dynamics

1. Motor Dynamics

1. Motor/Wheel Dynamics1
2. Motor/Wheel Dynamics2

2. Slip And Traction

1. Slip Estimation1

1. Determine Max Magnitude

2. Slip Estimation2

1. Determine Max Magnitude
3. Traction Estimation1
4. Traction Estimation2

3. Wheel Velocity Estimator

Simulation Parameter	Value
Solver	ode45
RelTol	1e-3
Refine	1
MaxOrder	5
ZeroCross	on

[more info]

System - NonlinearDynamics

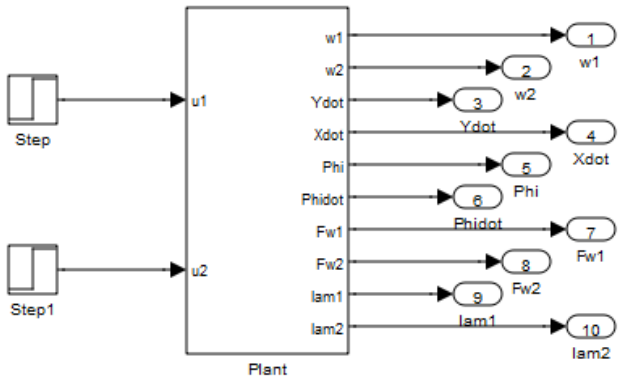


Table 1. Outport Block Properties

Name	Port	Icon Display	Lock Scale	Var Size Sig	Signal Type	Sampling Mode	Source Of Initial Output Value	Output When Disabled	Used By Blk
Fw1	7	Port number	off	Inherit	auto	auto	Dialog	held	NonlinearDynamics (model)
Fw2	8	Port number	off	Inherit	auto	auto	Dialog	held	NonlinearDynamics (model)
lam1	9	Port number	off	Inherit	auto	auto	Dialog	held	NonlinearDynamics (model)
lam2	10	Port number	off	Inherit	auto	auto	Dialog	held	NonlinearDynamics (model)
Phi	5	Port number	off	Inherit	auto	auto	Dialog	held	NonlinearDynamics (model)
Phidot	6	Port number	off	Inherit	auto	auto	Dialog	held	NonlinearDynamics (model)
w1	1	Port number	off	Inherit	auto	auto	Dialog	held	NonlinearDynamics (model)
w2	2	Port number	off	Inherit	auto	auto	Dialog	held	NonlinearDynamics (model)
Xdot	4	Port number	off	Inherit	auto	auto	Dialog	held	NonlinearDynamics (model)
Ydot	3	Port number	off	Inherit	auto	auto	Dialog	held	NonlinearDynamics (model)

Table 2. Step Block Properties

Name	Time	Before	After	Sample Time	Zero Cross
Step	0	0	U1	0	on
Step1	0	0	U2	0	on

System - NonlinearDynamics/Plant

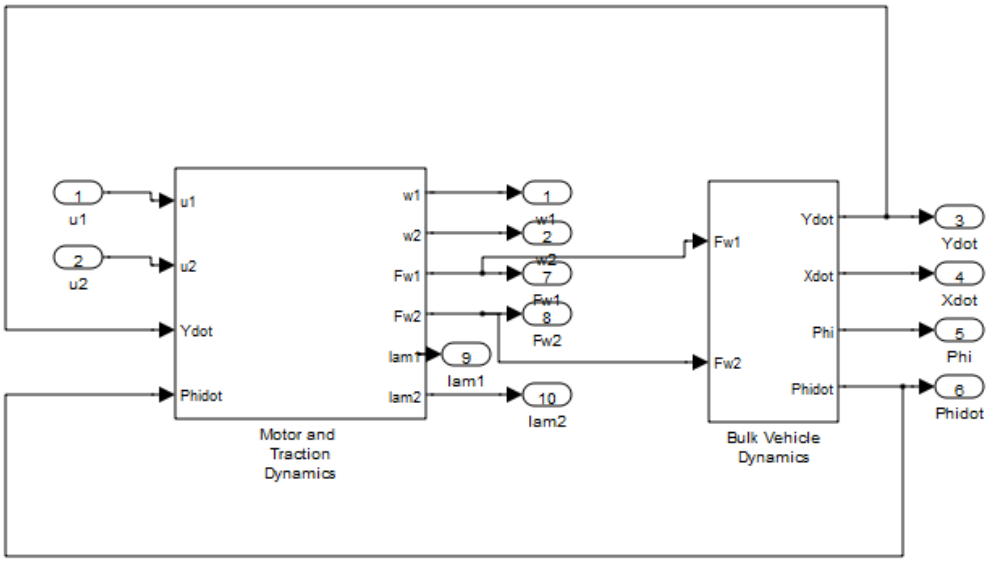


Table 3. Inport Block Properties

Name	Port	Defined In Blk
u1	1	Step
u2	2	Step1

Table 4. Outport Block Properties

Name	Port	Icon Display	Lock Scale	Var Size Sig	Signal Type	Sampling Mode	Source Of Initial Output Value	Output When Disabled	Used By Blk
Fw1	7	Port number	off	Inherit	auto	auto	Dialog	held	Sum, Sum, NonlinearDynamics (model)
Fw2	8	Port number	off	Inherit	auto	auto	Dialog	held	Sum, Sum, NonlinearDynamics (model)

lam1	9	Port number	off	Inherit	auto	auto	Dialog	held	NonlinearDynamics (model)
lam2	10	Port number	off	Inherit	auto	auto	Dialog	held	NonlinearDynamics (model)
Phi	5	Port number	off	Inherit	auto	auto	Dialog	held	NonlinearDynamics (model)
Phidot	6	Port number	off	Inherit	auto	auto	Dialog	held	NonlinearDynamics (model), Fcn1, Fcn
w1	1	Port number	off	Inherit	auto	auto	Dialog	held	NonlinearDynamics (model)
w2	2	Port number	off	Inherit	auto	auto	Dialog	held	NonlinearDynamics (model)
Xdot	4	Port number	off	Inherit	auto	auto	Dialog	held	NonlinearDynamics (model)
Ydot	3	Port number	off	Inherit	auto	auto	Dialog	held	NonlinearDynamics (model), Fcn1, Fcn

System - NonlinearDynamics/Plant/Bulk Vehicle Dynamics

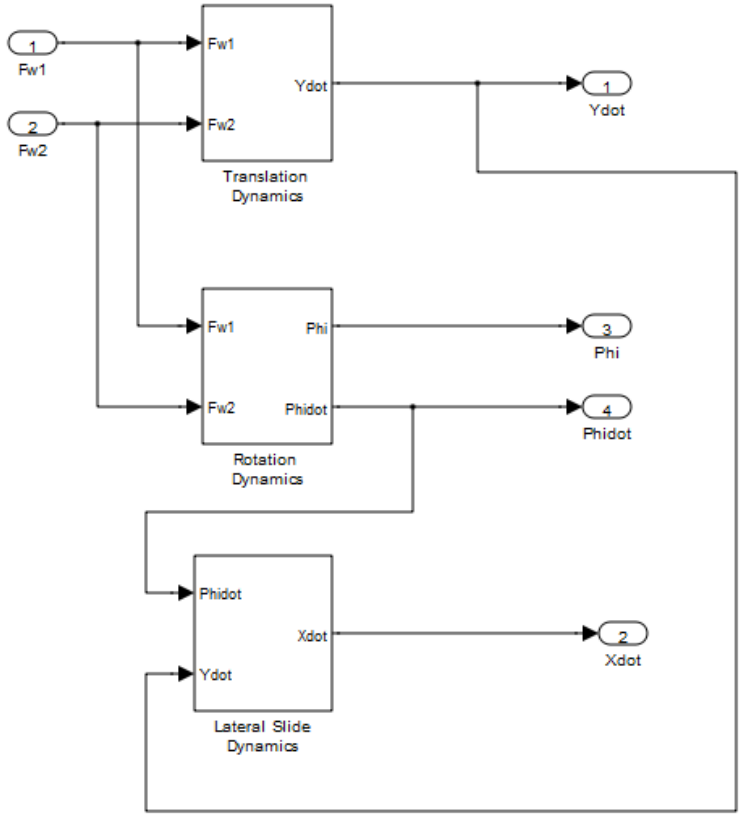


Table 5. Inport Block Properties

Name	Port	Defined In Blk
Fw1	1	Fcn
Fw2	2	Fcn

Table 6. Outport Block Properties

Name	Port	Icon Display	Lock Scale	Var Size Sig	Signal Type	Sampling Mode	Source Of Initial Output Value	Output When Disabled	Used By Blk
Phi	3	Port number	off	Inherit	auto	auto	Dialog	held	NonlinearDynamics (model)
Phidot	4	Port number	off	Inherit	auto	auto	Dialog	held	Product, NonlinearDynamics (model), Fcn1, Fcn

Xdot	2	Port number	off	Inherit	auto	auto	Dialog	held	NonlinearDynamics (model)
Ydot	1	Port number	off	Inherit	auto	auto	Dialog	held	Product, NonlinearDynamics (model), Fcn1, Fcn

System - NonlinearDynamics/Plant/Bulk Vehicle Dynamics/Lateral Slide Dynamics

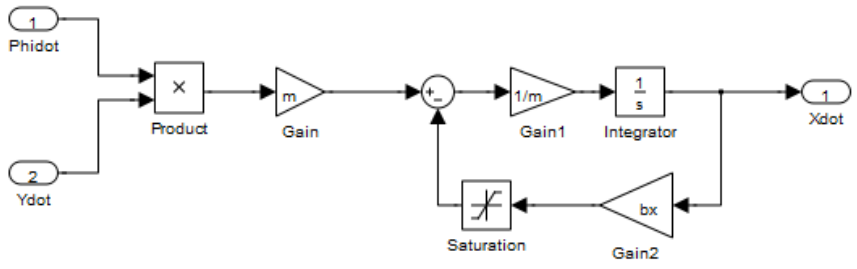


Table 7. Gain Block Properties

Name	Gain	Multiplication	Param Data Type Str	Out Data Type Str	Lock Scale	Rnd Meth	Saturate On Integer Overflow
Gain	m	Element-wise(K.*u)	Inherit: Inherit via internal rule	Inherit: Inherit via internal rule	off	Floor	off
Gain1	1/m	Element-wise(K.*u)	Inherit: Inherit via internal rule	Inherit: Inherit via internal rule	off	Floor	off
Gain2	bx	Element-wise(K.*u)	Inherit: Inherit via internal rule	Inherit: Inherit via internal rule	off	Floor	off

Table 8. Inport Block Properties

Name	Port	Defined In Blk
Phidot	1	Integrator
Ydot	2	Integrator

Table 9. Integrator Block Properties

Name	External Reset	Initial Condition Source	Absolute Tolerance	Zero Cross	Continuous State Attributes
Integrator	none	internal	auto	on	''

Table 10. Outport Block Properties

Name	Port	Icon Display	Lock Scale	Var Size Sig	Signal Type	Sampling Mode	Source Of Initial Output Value	Output When Disabled	Used By Blk
Xdot	1	Port number	off	Inherit	auto	auto	Dialog	held	NonlinearDynamics (model), Gain2

Table 11. Product Block Properties

Name	Inputs	Multiplication	Collapse Mode	Collapse Dim	Input Same DT	Out Data Type Str	Lock Scale	Rnd Meth	Saturate On Integer Overflow
Product	2	Element-wise(.*)	All dimensions	1	off	Inherit: Inherit via internal rule	off	Zero	off

Table 12. Saturate Block Properties

Name	Upper Limit Source	Upper Limit	Lower Limit Source	Lower Limit	Linearize As Gain	Zero Cross	Out Data Type Str	Lock Scale	Rnd Meth
Saturation	Dialog	bx	Dialog	-bx	on	on	Inherit: Same as input	off	Floor

Table 13. Sum Block Properties

Name	Icon Shape	Inputs	Collapse Mode	Collapse Dim	Input Same DT	Accum Data Type Str	Out Data Type Str	Lock Scale	Rnd Meth	Saturate On Integer Overflow
Sum	round	|+-	All dimensions	1	off	Inherit: Inherit via internal rule	Inherit: Inherit via internal rule	off	Floor	off

System - NonlinearDynamics/Plant/Bulk Vehicle Dynamics/Rotation Dynamics

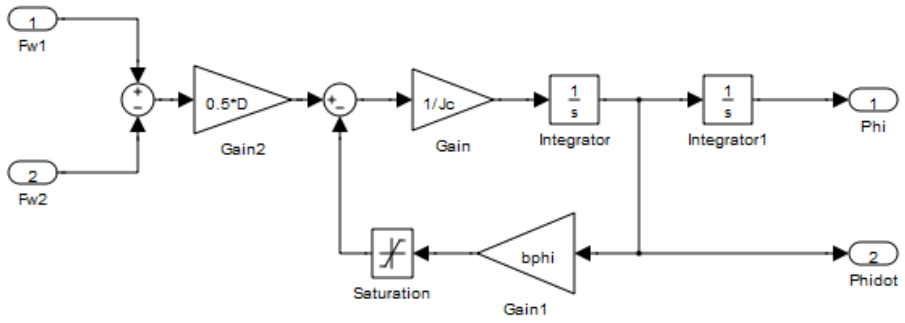


Table 14. Gain Block Properties

Name	Gain	Multiplication	Param Data Type Str	Out Data Type Str	Lock Scale	Rnd Meth	Saturate On Integer Overflow
Gain	1/Jc	Element-wise(K.*u)	Inherit: Inherit via internal rule	Inherit: Inherit via internal rule	off	Floor	off
Gain1	bphi	Element-wise(K.*u)	Inherit: Inherit via internal rule	Inherit: Inherit via internal rule	off	Floor	off
Gain2	0.5*D	Element-wise(K.*u)	Inherit: Inherit via internal rule	Inherit: Inherit via internal rule	off	Floor	off

Table 15. Inport Block Properties

Name	Port	Defined In Blk
Fw1	1	Fcn
Fw2	2	Fcn

Table 16. Integrator Block Properties

Name	External Reset	Initial Condition Source	Absolute Tolerance	Zero Cross	Continuous State Attributes
Integrator	none	internal	auto	on	''
Integrator1	none	internal	auto	on	''

Table 17. Outport Block Properties

Name	Port	Icon Display	Lock Scale	Var Size Sig	Signal Type	Sampling Mode	Source Of Initial Output Value	Output When Disabled	Used By Blk
Phi	1	Port number	off	Inherit	auto	auto	Dialog	held	NonlinearDynamics (model)

Phidot	2	Port number	off	Inherit	auto	auto	Dialog	held	Product, NonlinearDynamics (model), Fcn1, Fcn, Integrator1, Gain1

Table 18. Saturate Block Properties

Name	Upper Limit Source	Upper Limit	Lower Limit Source	Lower Limit	Linearize As Gain	Zero Cross	Out Data Type Str	Lock Scale	Rnd Meth
Saturation	Dialog	bphi	Dialog	-bphi	on	on	Inherit: Same as input	off	Floor

Table 19. Sum Block Properties

Name	Icon Shape	Inputs	Collapse Mode	Collapse Dim	Input Same DT	Accum Data Type Str	Out Data Type Str	Lock Scale	Rnd Meth	Saturate On Integer Overflow
Sum	round	+|-	All dimensions	1	off	Inherit: Inherit via internal rule	Inherit: Inherit via internal rule	off	Floor	off
Sum1	round	|+-	All dimensions	1	off	Inherit: Inherit via internal rule	Inherit: Inherit via internal rule	off	Floor	off

System - NonlinearDynamics/Plant/Bulk Vehicle Dynamics/Translation Dynamics

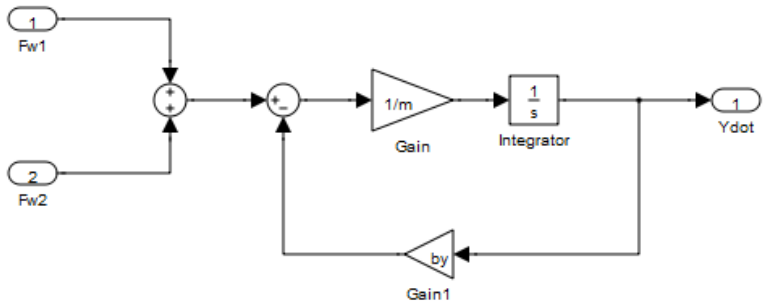


Table 20. Gain Block Properties

Name	Gain	Multiplication	Param Data Type Str	Out Data Type Str	Lock Scale	Rnd Meth	Saturate On Integer Overflow
		Element-	Inherit: Inherit via	Inherit: Inherit via			

Gain	1/m	wise(K.*u)	internal rule	internal rule	off	Floor	off
Gain1	by	Element-wise(K.*u)	Inherit: Inherit via internal rule	Inherit: Inherit via internal rule	off	Floor	off

Table 21. Inport Block Properties

Name	Port	Defined In Blk
Fw1	1	Fcn
Fw2	2	Fcn

Table 22. Integrator Block Properties

Name	External Reset	Initial Condition Source	Absolute Tolerance	Zero Cross	Continuous State Attributes
Integrator	none	internal	auto	on	''

Table 23. Outport Block Properties

Name	Port	Icon Display	Lock Scale	Var Size Sig	Signal Type	Sampling Mode	Source Of Initial Output Value	Output When Disabled	Used By Blk
Ydot	1	Port number	off	Inherit	auto	auto	Dialog	held	Product, NonlinearDynamics (model), Fcn1, Fcn, Gain1

Table 24. Sum Block Properties

Name	Icon Shape	Inputs	Collapse Mode	Collapse Dim	Input Same DT	Accum Data Type Str	Out Data Type Str	Lock Scale	Rnd Meth	Saturate On Integer Overflow
Sum	round	+|+	All dimensions	1	off	Inherit: Inherit via internal rule	Inherit: Inherit via internal rule	off	Floor	off
Sum1	round	|+-	All dimensions	1	off	Inherit: Inherit via internal rule	Inherit: Inherit via internal rule	off	Floor	off

System - NonlinearDynamics/Plant/Motor and Traction Dynamics

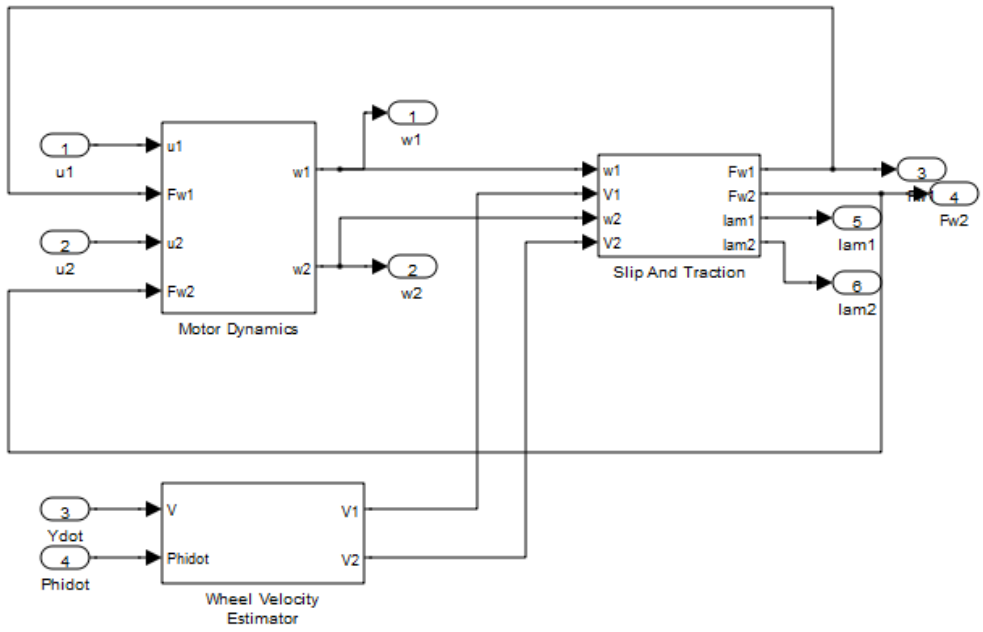


Table 25. Inport Block Properties

Name	Port	Defined In Blk
Phidot	4	Integrator
u1	1	Step
u2	2	Step1
Ydot	3	Integrator

Table 26. Outport Block Properties

Name	Port	Icon Display	Lock Scale	Var Size Sig	Signal Type	Sampling Mode	Source Of Initial Output Value	Output When Disabled	Used By Blk
Fw1	3	Port number	off	Inherit	auto	auto	Dialog	held	Traction, Sum, Sum, NonlinearDynamics (model)
Fw2	4	Port number	off	Inherit	auto	auto	Dialog	held	Transfer Fcn1, Sum, Sum, NonlinearDynamics (model)
lam1	5	Port number	off	Inherit	auto	auto	Dialog	held	NonlinearDynamics (model)

lam2	6	Port number	off	Inherit	auto	auto	Dialog	held	NonlinearDynamics (model)
w1	1	Port number	off	Inherit	auto	auto	Dialog	held	Gain, NonlinearDynamics (model)
w2	2	Port number	off	Inherit	auto	auto	Dialog	held	Gain, NonlinearDynamics (model)

System - NonlinearDynamics/Plant/Motor and Traction Dynamics/Motor Dynamics

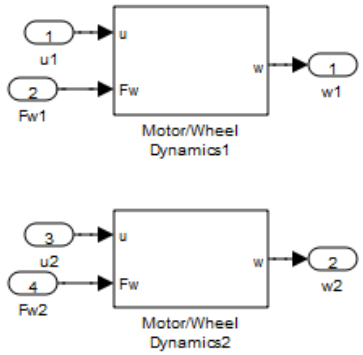


Table 27. Inport Block Properties

Name	Port	Defined In Blk
Fw1	2	Fcn
Fw2	4	Fcn
u1	1	Step
u2	3	Step1

Table 28. Outport Block Properties

Name	Port	Icon Display	Lock Scale	Var Size Sig	Signal Type	Sampling Mode	Source Of Initial Output Value	Output When Disabled	Used By Blk
w1	1	Port number	off	Inherit	auto	auto	Dialog	held	Gain, NonlinearDynamics (model)
w2	2	Port number	off	Inherit	auto	auto	Dialog	held	Gain, NonlinearDynamics (model)

System - NonlinearDynamics/Plant/Motor and Traction Dynamics/Motor Dynamics/Motor/Wheel Dynamics1

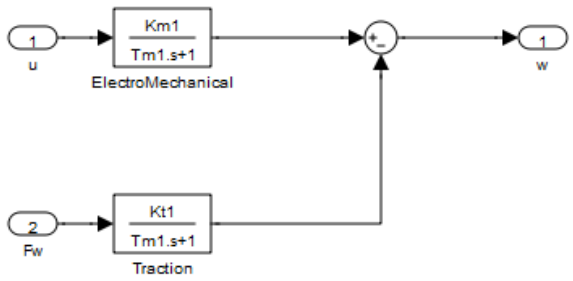


Table 29. Inport Block Properties

Name	Port	Defined In Blk
Fw	2	Fcn
u	1	Step

Table 30. Outport Block Properties

Name	Port	Icon Display	Lock Scale	Var Size Sig	Signal Type	Sampling Mode	Source Of Initial Output Value	Output When Disabled	Used By Blk
w	1	Port number	off	Inherit	auto	auto	Dialog	held	Gain, NonlinearDynamics (model)

Table 31. Sum Block Properties

Name	Icon Shape	Inputs	Collapse Mode	Collapse Dim	Input Same DT	Accum Data Type Str	Out Data Type Str	Lock Scale	Rnd Meth	Saturate On Integer Overflow
Sum	round	|+-	All dimensions	1	off	Inherit: Inherit via internal rule	Inherit: Inherit via internal rule	off	Floor	off

Table 32. TransferFcn Block Properties

Name	Numerator	Denominator	Absolute Tolerance	Continuous State Attributes
ElectroMechanical	[Km1]	[Tm1 1]	auto	''

Traction	[Kt1]	[Tm1 1]	auto	''

System - NonlinearDynamics/Plant/Motor and Traction Dynamics/Motor Dynamics/Motor/Wheel Dynamics2

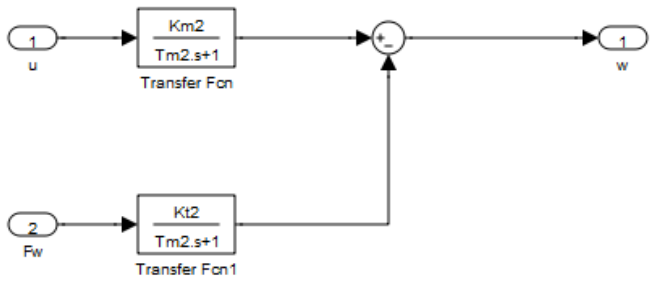


Table 33. Inport Block Properties

Name	Port	Defined In Blk
Fw	2	Fcn
u	1	Step1

Table 34. Outport Block Properties

Name	Port	Icon Display	Lock Scale	Var Size Sig	Signal Type	Sampling Mode	Source Of Initial Output Value	Output When Disabled	Used By Blk
w	1	Port number	off	Inherit	auto	auto	Dialog	held	Gain, NonlinearDynamics (model)

Table 35. Sum Block Properties

Name	Icon Shape	Inputs	Collapse Mode	Collapse Dim	Input Same DT	Accum Data Type Str	Out Data Type Str	Lock Scale	Rnd Meth	Saturate On Integer Overflow
Sum	round	|+-	All dimensions	1	off	Inherit: Inherit via internal rule	Inherit: Inherit via internal rule	off	Floor	off

Table 36. TransferFcn Block Properties

Name	Numerator	Denominator	Absolute Tolerance	Continuous State Attributes
Transfer Fcn	[Km2]	[Tm2 1]	auto	''

Transfer Fcn1	[Kt2]	[Tm2 1]	auto	''

System - NonlinearDynamics/Plant/Motor and Traction Dynamics/Slip And Traction

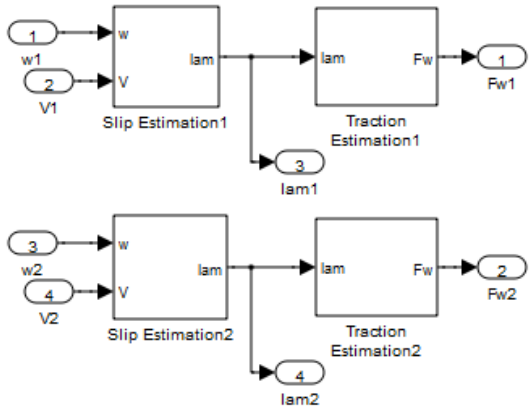


Table 37. Inport Block Properties

Name	Port	Defined In Blk
V1	2	Fcn
V2	4	Fcn1
w1	1	Sum
w2	3	Sum

Table 38. Outport Block Properties

Name	Port	Icon Display	Lock Scale	Var Size Sig	Signal Type	Sampling Mode	Source Of Initial Output Value	Output When Disabled	Used By Blk
Fw1	1	Port number	off	Inherit	auto	auto	Dialog	held	Traction, Sum, Sum, NonlinearDynamics (model)
Fw2	2	Port number	off	Inherit	auto	auto	Dialog	held	Transfer Fcn1, Sum, Sum, NonlinearDynamics (model)
lam1	3	Port number	off	Inherit	auto	auto	Dialog	held	NonlinearDynamics (model), Fcn

lam2	4	Port number	off	Inherit	auto	auto	Dialog	held	NonlinearDynamics (model), Fcn

System - NonlinearDynamics/Plant/Motor and Traction Dynamics/Slip And Traction/Slip Estimation1

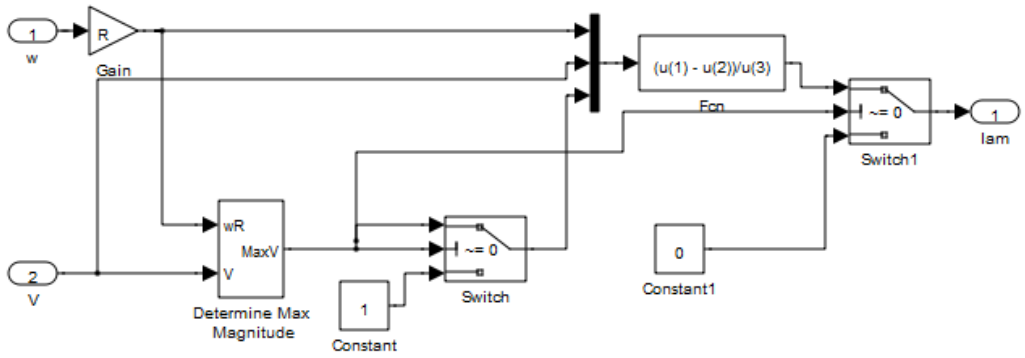


Table 39. Constant Block Properties

Name	Value	Sampling Mode	Out Data Type Str	Lock Scale	Sample Time	Frame Period
Constant	1	Sample based	Inherit: Inherit from 'Constant value'	off	inf	inf
Constant1	0	Sample based	Inherit: Inherit from 'Constant value'	off	inf	inf

Table 40. Fcn Block Properties

Name	Expr
Fcn	(u(1) - u(2))/u(3)

Table 41. Gain Block Properties

Name	Gain	Multiplication	Param Data Type Str	Out Data Type Str	Lock Scale	Rnd Meth	Saturate On Integer Overflow
Gain	R	Element-wise(K.*u)	Inherit: Inherit via internal rule	Inherit: Inherit via internal rule	off	Floor	off

Table 42. Inport Block Properties

Name	Port	Defined In Blk

V	2	Fcn
w	1	Sum

Table 43. Mux Block Properties

Name	Inputs	Display Option
Mux	3	bar

Table 44. Outport Block Properties

Name	Port	Icon Display	Lock Scale	Var Size Sig	Signal Type	Sampling Mode	Source Of Initial Output Value	Output When Disabled	Used By Blk
lam	1	Port number	off	Inherit	auto	auto	Dialog	held	NonlinearDynamics (model), Fcn

Table 45. Switch Block Properties

Name	Criteria	Threshold	Input Same DT	Out Data Type Str	Lock Scale	Rnd Meth	Saturate On Integer Overflow	Zero Cross	Allow Diff Input Sizes
Switch	u2 ~= 0	0	off	Inherit: Inherit via internal rule	off	Floor	off	on	off
Switch1	u2 ~= 0	0	off	Inherit: Inherit via internal rule	off	Floor	off	on	off

System - NonlinearDynamics/Plant/Motor and Traction Dynamics/Slip And Traction/Slip Estimation1/Determine Max Magnitude

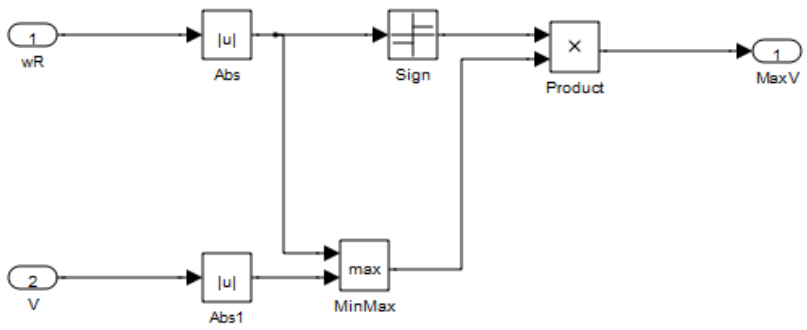


Table 46. Abs Block Properties

Name	Zero Cross	Out Data Type Str	Lock Scale	Rnd Meth	Saturate On Integer Overflow
Abs	on	Inherit: Same as input	off	Floor	off
Abs1	on	Inherit: Same as input	off	Floor	off

Table 47. Inport Block Properties

Name	Port	Defined In Blk
V	2	Fcn
wR	1	Gain

Table 48. MinMax Block Properties

Name	Function	Inputs	Input Same DT	Out Data Type Str	Lock Scale	Rnd Meth	Saturate On Integer Overflow	Zero Cross
MinMax	max	2	off	Inherit: Inherit via internal rule	off	Floor	off	on

Table 49. Outport Block Properties

Name	Port	Icon Display	Lock Scale	Var Size Sig	Signal Type	Sampling Mode	Source Of Initial Output Value	Output When Disabled	Used By Blk
MaxV	1	Port number	off	Inherit	auto	auto	Dialog	held	Switch, Switch, Switch1

Table 50. Product Block Properties

Name	Inputs	Multiplication	Collapse Mode	Collapse Dim	Input Same DT	Out Data Type Str	Lock Scale	Rnd Meth	Saturate On Integer Overflow
Product	2	Element-wise(.*)	All dimensions	1	off	Inherit: Inherit via internal rule	off	Zero	off

Table 51. Signum Block Properties

Name	Zero Cross
Sign	on

System - NonlinearDynamics/Plant/Motor and Traction Dynamics/Slip And Traction/Slip Estimation2

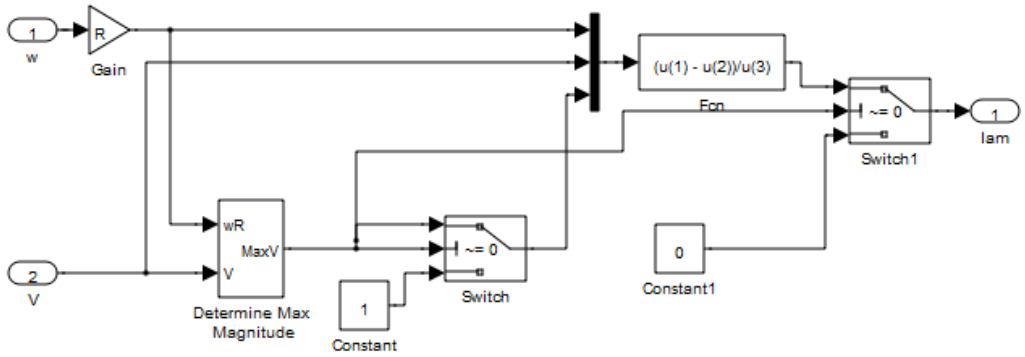


Table 52. Constant Block Properties

Name	Value	Sampling Mode	Out Data Type Str	Lock Scale	Sample Time	Frame Period
Constant	1	Sample based	Inherit: Inherit from 'Constant value'	off	inf	inf
Constant1	0	Sample based	Inherit: Inherit from 'Constant value'	off	inf	inf

Table 53. Fcn Block Properties

Name	Expr
Fcn	(u(1) - u(2))/u(3)

Table 54. Gain Block Properties

Name	Gain	Multiplication	Param Data Type Str	Out Data Type Str	Lock Scale	Rnd Meth	Saturate On Integer Overflow
Gain	R	Element-wise(K.*u)	Inherit: Inherit via internal rule	Inherit: Inherit via internal rule	off	Floor	off

Table 55. Inport Block Properties

Name	Port	Defined In Blk
V	2	Fcn1
w	1	Sum

Table 56. Mux Block Properties

Name	Inputs	Display Option

Mux	3	bar

Table 57. Outport Block Properties

Name	Port	Icon Display	Lock Scale	Var Size Sig	Signal Type	Sampling Mode	Source Of Initial Output Value	Output When Disabled	Used By Blk
lam	1	Port number	off	Inherit	auto	auto	Dialog	held	NonlinearDynamics (model), Fcn

Table 58. Switch Block Properties

Name	Criteria	Threshold	Input Same DT	Out Data Type Str	Lock Scale	Rnd Meth	Saturate On Integer Overflow	Zero Cross	Allow Diff Input Sizes
Switch	u2 ~= 0	0	off	Inherit: Inherit via internal rule	off	Floor	off	on	off
Switch1	u2 ~= 0	0	off	Inherit: Inherit via internal rule	off	Floor	off	on	off

System - NonlinearDynamics/Plant/Motor and Traction Dynamics/Slip And Traction/Slip Estimation2/Determine Max Magnitude

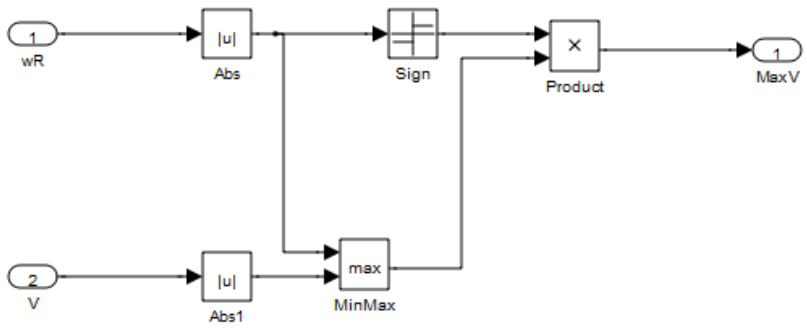


Table 59. Abs Block Properties

Name	Zero Cross	Out Data Type Str	Lock Scale	Rnd Meth	Saturate On Integer Overflow
Abs	on	Inherit: Same as input	off	Floor	off
Abs1	on	Inherit: Same as input	off	Floor	off

Table 60. Inport Block Properties

Name	Port	Defined In Blk
V	2	Fcn1
wR	1	Gain

Table 61. MinMax Block Properties

Name	Function	Inputs	Input Same DT	Out Data Type Str	Lock Scale	Rnd Meth	Saturate On Integer Overflow	Zero Cross
MinMax	max	2	off	Inherit: Inherit via internal rule	off	Floor	off	on

Table 62. Outport Block Properties

Name	Port	Icon Display	Lock Scale	Var Size Sig	Signal Type	Sampling Mode	Source Of Initial Output Value	Output When Disabled	Used By Blk
MaxV	1	Port number	off	Inherit	auto	auto	Dialog	held	Switch, Switch, Switch1

Table 63. Product Block Properties

Name	Inputs	Multiplication	Collapse Mode	Collapse Dim	Input Same DT	Out Data Type Str	Lock Scale	Rnd Meth	Saturate On Integer Overflow
Product	2	Element-wise(.*)	All dimensions	1	off	Inherit: Inherit via internal rule	off	Zero	off

Table 64. Signum Block Properties

Name	Zero Cross
Sign	on

System - NonlinearDynamics/Plant/Motor and Traction Dynamics/Slip And Traction/Traction Estimation1

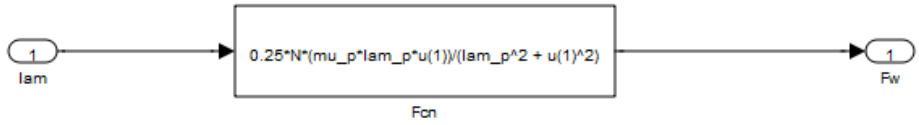


Table 65. Fcn Block Properties

Name	Expr
Fcn	0.25*N*(mu_p*lam_p*u(1))/(lam_p^2 + u(1)^2)

Table 66. Inport Block Properties

Name	Port	Defined In Blk
lam	1	Switch1

Table 67. Outport Block Properties

Name	Port	Icon Display	Lock Scale	Var Size Sig	Signal Type	Sampling Mode	Source Of Initial Output Value	Output When Disabled	Used By Blk
Fw	1	Port number	off	Inherit	auto	auto	Dialog	held	Traction, Sum, Sum, NonlinearDynamics (model)

System - NonlinearDynamics/Plant/Motor and Traction Dynamics/Slip And Traction/Traction Estimation2

Table 68. Fcn Block Properties

Name	Expr
Fcn	0.25*N*(mu_p*lam_p*u(1))/(lam_p^2 + u(1)^2)

Table 69. Inport Block Properties

Name	Port	Defined In Blk

lam	1	Switch1

Table 70. Outport Block Properties

Name	Port	Icon Display	Lock Scale	Var Size Sig	Signal Type	Sampling Mode	Source Of Initial Output Value	Output When Disabled	Used By Blk
Fw	1	Port number	off	Inherit	auto	auto	Dialog	held	Transfer Fcn1, Sum, Sum, NonlinearDynamics (model)

System - NonlinearDynamics/Plant/Motor and Traction Dynamics/Wheel Velocity Estimator

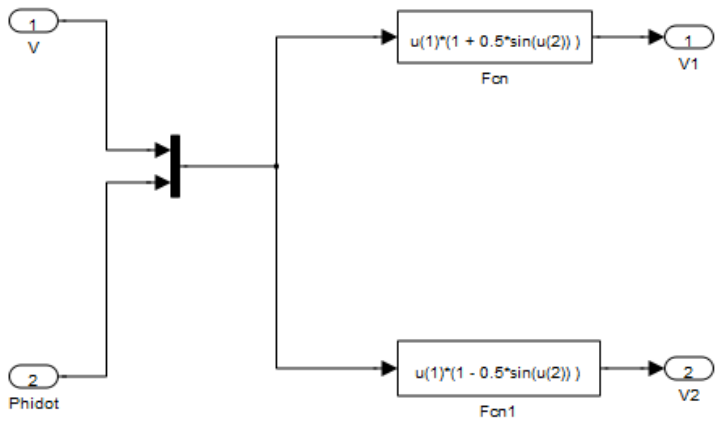


Table 71. Fcn Block Properties

Name	Expr
Fcn	u(1)*(1 + 0.5*sin(u(2)))
Fcn1	u(1)*(1 - 0.5*sin(u(2)))

Table 72. Inport Block Properties

Name	Port	Defined In Blk
Phidot	2	Integrator
V	1	Integrator

Table 73. Mux Block Properties

Name	Inputs	Display Option
Mux	2	bar

Table 74. Outport Block Properties

Name	Port	Icon Display	Lock Scale	Var Size Sig	Signal Type	Sampling Mode	Source Of Initial Output Value	Output When Disabled	Used By Blk
V1	1	Port number	off	Inherit	auto	auto	Dialog	held	Fcn, Abs1
V2	2	Port number	off	Inherit	auto	auto	Dialog	held	Fcn, Abs1

Appendix

Table 75. Block Type Count

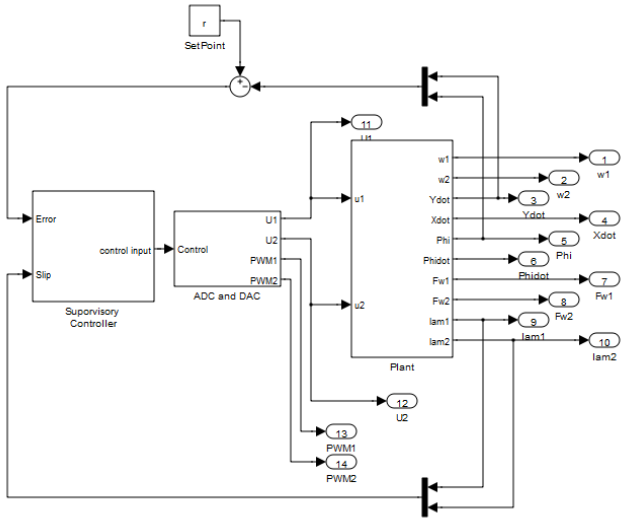
BlockType	Count	Block Names
Outport	50	Fw1, Fw2, Phi, Phidot, Xdot, Phi, Phidot, Phi, Phidot, Ydot, Xdot, Ydot, Fw1, Fw2, Fw1, Fw2, w, w, w1, w2, Fw1, Fw2, MaxV, lam, MaxV, lam, Fw, Fw, lam1, lam2, V1, V2, lam1, lam2, w1, w2, Phi, Phidot, Xdot, Ydot, lam1, lam2, w1, w2, Xdot, Ydot, lam1, lam2, w1, w2
Inport	38	Fw1, Fw2, Phidot, Ydot, Fw1, Fw2, Fw1, Fw2, Fw1, Fw2, Fw, u, Fw, u, u1, u2, Phidot, V, wR, V, w, V, wR, V, w, lam, lam, V1, V2, w1, w2, Phidot, V, Ydot, u1, u2, u1, u2
SubSystem	17	Plant, Bulk Vehicle Dynamics, Lateral Slide Dynamics, Rotation Dynamics, Translation Dynamics, Motor and Traction Dynamics, Motor Dynamics, Motor/Wheel Dynamics1, Motor/Wheel Dynamics2, Slip And Traction, Slip Estimation1, Determine Max Magnitude, Slip Estimation2, Determine Max Magnitude, Traction Estimation1, Traction Estimation2, Wheel Velocity Estimator
Gain	10	Gain, Gain1, Gain2, Gain, Gain1, Gain2, Gain, Gain1, Gain, Gain
Sum	7	Sum, Sum, Sum1, Sum, Sum1, Sum, Sum
Fcn	6	Fcn, Fcn, Fcn, Fcn, Fcn, Fcn1
TransferFcn	4	ElectroMechanical, Traction, Transfer Fcn, Transfer Fcn1
Switch	4	Switch, Switch1, Switch, Switch1
Integrator	4	Integrator, Integrator, Integrator1, Integrator
Constant	4	Constant, Constant1, Constant, Constant1
Abs	4	Abs, Abs1, Abs, Abs1
Product	3	Product, Product, Product
Mux	3	Mux, Mux, Mux
Step	2	Step, Step1
Signum	2	Sign, Sign
Saturate	2	Saturation, Saturation

MinMax	2	MinMax, MinMax

D.5 Continuous Traction Control and Slip Instants

NonlinearDynamics_ContinuousTraction_Slip

Details for NonlinearDynamics_ContinuousTraction_Slip/Suporvisory Controller and below



Ryan

27-May-2014 13:04:26

Table of Contents

Model - NonlinearDynamics_ContinuousTraction_Slip
System - NonlinearDynamics_ContinuousTraction_Slip/Suporvisory Controller
System - NonlinearDynamics_ContinuousTraction_Slip/Suporvisory Controller/Traction Control
System - NonlinearDynamics_ContinuousTraction_Slip/Suporvisory Controller/Velocity and Drift Control
Appendix

List of Tables

1. Constant Block Properties
2. Demux Block Properties
3. Fcn Block Properties
4. Inport Block Properties
5. Mux Block Properties
6. Outport Block Properties
7. Product Block Properties
8. Sum Block Properties
9. Demux Block Properties
10. Gain Block Properties
11. Inport Block Properties
12. Outport Block Properties
13. Demux Block Properties
14. Inport Block Properties
15. Outport Block Properties
16. PID 1dof Block Properties
17. Sum Block Properties
18. Block Type Count

Model - NonlinearDynamics_ContinuousTraction_Slip

Full Model Hierarchy

1. NonlinearDynamics_ContinuousTraction_Slip
    1. ADC and DAC
        1. ADC
        2. ADC1
        3. DAC

4. DAC1

2. Plant
    1. Bulk Vehicle Dynamics
        1. Lateral Slide Dynamics
        2. Rotation Dynamics
        3. Translation Dynamics
    2. Motor and Traction Dynamics
        1. Motor Dynamics
            1. Motor/Wheel Dynamics1
            2. Motor/Wheel Dynamics2
        2. Slip And Traction
            1. Slip Estimation1
                1. Determine Max Magnitude
            2. Slip Estimation2
                1. Determine Max Magnitude
            3. Traction Estimation1
            4. Traction Estimation2
        3. Wheel Velocity Estimator
3. Suporvisory Controller
    1. Traction Control
    2. Velocity and Drift Control

Simulation Parameter	Value
Solver	ode45
RelTol	1e-3
Refine	1
MaxOrder	5
ZeroCross	on

[more info]

System - NonlinearDynamics_ContinuousTraction_Slip/Suporvisory Controller

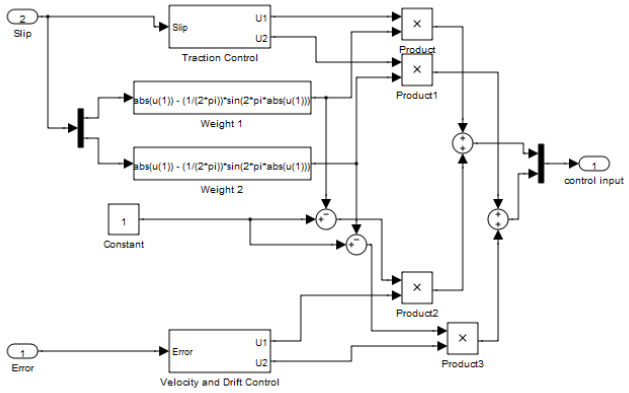


Table 1. Constant Block Properties

Name	Value	Sampling Mode	Out Data Type Str	Lock Scale	Sample Time	Frame Period
Constant	1	Sample based	Inherit: Inherit from 'Constant value'	off	inf	inf

Table 2. Demux Block Properties

Name	Outputs	Display Option	Bus Selection Mode
Demux	2	bar	off

Table 3. Fcn Block Properties

Name	Expr
Weight 1	abs(u(1)) - (1/(2*pi))*sin(2*pi*abs(u(1)))
Weight 2	abs(u(1)) - (1/(2*pi))*sin(2*pi*abs(u(1)))

Table 4. Inport Block Properties

Name	Port	Defined In Blk
Error	1	Sum
Slip	2	Switch1, Switch1

Table 5. Mux Block Properties

Name	Inputs	Display Option
Mux	2	bar

Table 6. Outport Block Properties

Name	Port	Icon Display	Lock Scale	Var Size Sig	Signal Type	Sampling Mode	Source Of Initial Output Value	Output When Disabled	Used By Blk
control input	1	Port number	off	Inherit	auto	auto	Dialog	held	Demux

Table 7. Product Block Properties

Name	Inputs	Multiplication	Collapse Mode	Collapse Dim	Input Same DT	Out Data Type Str	Lock Scale	Rnd Meth	Saturate On Integer Overflow
Product	2	Element-wise(.*)	All dimensions	1	off	Inherit: Inherit via internal rule	off	Zero	off
Product1	2	Element-wise(.*)	All dimensions	1	off	Inherit: Inherit via internal rule	off	Zero	off
Product2	2	Element-wise(.*)	All dimensions	1	off	Inherit: Inherit via internal rule	off	Zero	off
Product3	2	Element-wise(.*)	All dimensions	1	off	Inherit: Inherit via internal rule	off	Zero	off

Table 8. Sum Block Properties

Name	Icon Shape	Inputs	Collapse Mode	Collapse Dim	Input Same DT	Accum Data Type Str	Out Data Type Str	Lock Scale	Rnd Meth	Saturate On Integer Overflow	Sample Time
Sum	round	+|+	All dimensions	1	off	Inherit: Inherit via internal rule	Inherit: Inherit via internal rule	off	Floor	off	Ts
Sum1	round	+|+	All dimensions	1	off	Inherit: Inherit via internal rule	Inherit: Inherit via internal rule	off	Floor	off	Ts
Sum2	round	-+|	All dimensions	1	off	Inherit: Inherit via internal rule	Inherit: Inherit via internal rule	off	Floor	off	
Sum3	round	-+|	All dimensions	1	off	Inherit: Inherit via internal rule	Inherit: Inherit via internal rule	off	Floor	off	

System - NonlinearDynamics_ContinuousTraction_Slip/Suporvisory Controller/Traction Control

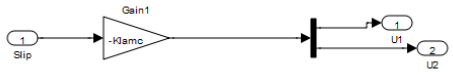


Table 9. Demux Block Properties

Name	Outputs	Display Option	Bus Selection Mode
Demux	2	bar	off

Table 10. Gain Block Properties

Name	Gain	Multiplication	Param Data Type Str	Out Data Type Str	Lock Scale	Rnd Meth	Saturate On Integer Overflow
Gain1	-Klamc	Element-wise(K.*u)	Inherit: Inherit via internal rule	Inherit: Inherit via internal rule	off	Floor	off

Table 11. Inport Block Properties

Name	Port	Defined In Blk
Slip	1	Switch1, Switch1

Table 12. Outport Block Properties

Name	Port	Icon Display	Lock Scale	Var Size Sig	Signal Type	Sampling Mode	Source Of Initial Output Value	Output When Disabled	Used By Blk
U1	1	Port number	off	Inherit	auto	auto	Dialog	held	Product
U2	2	Port number	off	Inherit	auto	auto	Dialog	held	Product1

System - NonlinearDynamics_ContinuousTraction_Slip/Suporvisory Controller/Velocity and Drift Control

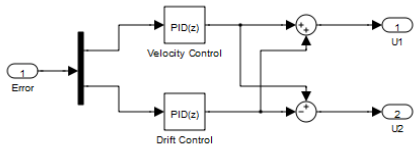


Table 13. Demux Block Properties

Name	Outputs	Display Option	Bus Selection Mode
Demux	2	bar	off

Table 14. Inport Block Properties

Name	Port	Defined In Blk
Error	1	Sum

Table 15. Outport Block Properties

Name	Port	Icon Display	Lock Scale	Var Size Sig	Signal Type	Sampling Mode	Source Of Initial Output Value	Output When Disabled	Used By Blk
U1	1	Port number	off	Inherit	auto	auto	Dialog	held	Product2
U2	2	Port number	off	Inherit	auto	auto	Dialog	held	Product3

Table 16. PID 1dof Block Properties

Name	Controller	Time Domain	Sample Time	Integrator Method	Filter Method	Form	P	I	D	N	Initial Condition Source	Initial Condition For Integrator	Initial Condition For Filter	External Reset	Zero Cross	Linearize As Gain	Anti Windup Mode	Kb	Tracking Mode	Kt	Rnd Meth	Saturate On Integer Overflow
Drift Control	PID	Discrete-time	Ts	Trapezoidal	Forward Euler	Parallel	Kpphi	Kiphi	Kdphi	100	internal	0	0	none	on	off	none	1	off	1	Floor	off
Velocity Control	PID	Discrete-time	Ts	Trapezoidal	Forward Euler	Parallel	Kpv	Kiv	Kdv	100	internal	0	0	none	on	off	none	1	off	1	Floor	off

Table 17. Sum Block Properties

Name	Icon Shape	Inputs	Collapse Mode	Collapse Dim	Input Same DT	Accum Data Type Str	Out Data Type Str	Lock Scale	Rnd Meth	Saturate On Integer Overflow
Sum	round	|++	All dimensions	1	off	Inherit: Inherit via internal rule	Inherit: Inherit via internal rule	off	Floor	off
Sum1	round	+-|	All dimensions	1	off	Inherit: Inherit via internal rule	Inherit: Inherit via internal rule	off	Floor	off

Appendix

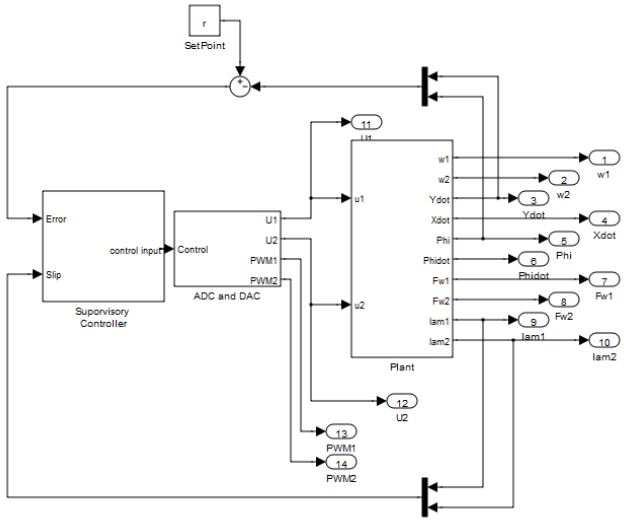
Table 18. Block Type Count

BlockType	Count	Block Names
Sum	6	Sum, Sum1, Sum2, Sum3, Sum, Sum1
Outport	5	U1, U2, U1, U2, control input
Product	4	Product, Product1, Product2, Product3
Inport	4	Error, Slip, Slip, Error
SubSystem	3	Supervisory Controller, Traction Control, Velocity and Drift Control
Demux	3	Demux, Demux, Demux
PID 1dof (m)	2	Drift Control, Velocity Control
Fcn	2	Weight 1, Weight 2
Mux	1	Mux
Gain	1	Gain1
Constant	1	Constant

D.6 Discrete Traction Control and Slip Instants

NonlinearDynamics_DiscreteTraction_Slip

Details for NonlinearDynamics_DiscreteTraction_Slip/Suporvisory Controller and below



Ryan

27-May-2014 13:06:32

Table of Contents

Model - NonlinearDynamics_DiscreteTraction_Slip
System - NonlinearDynamics_DiscreteTraction_Slip/Suporvisory Controller
System - NonlinearDynamics_DiscreteTraction_Slip/Suporvisory Controller/Traction Control
System - NonlinearDynamics_DiscreteTraction_Slip/Suporvisory Controller/Velocity and Drift Control
Appendix

List of Tables

1. Abs Block Properties
2. Inport Block Properties
3. MinMax Block Properties
4. Outport Block Properties
5. Switch Block Properties
6. Gain Block Properties
7. Inport Block Properties
8. Outport Block Properties
9. Sum Block Properties
10. UnitDelay Block Properties
11. Demux Block Properties
12. Inport Block Properties
13. Mux Block Properties
14. Outport Block Properties
15. PID 1dof Block Properties
16. Sum Block Properties
17. Block Type Count

Model - NonlinearDynamics_DiscreteTraction_Slip

Full Model Hierarchy

1. NonlinearDynamics_DiscreteTraction_Slip
    1. ADC and DAC
        1. ADC
        2. ADC1
        3. DAC
        4. DAC1

2. Plant
    1. Bulk Vehicle Dynamics
        1. Lateral Slide Dynamics
        2. Rotation Dynamics
        3. Translation Dynamics
    2. Motor and Traction Dynamics
        1. Motor Dynamics
            1. Motor/Wheel Dynamics1
            2. Motor/Wheel Dynamics2
        2. Slip And Traction
            1. Slip Estimation1
                1. Determine Max Magnitude
            2. Slip Estimation2
                1. Determine Max Magnitude
            3. Traction Estimation1
            4. Traction Estimation2
        3. Wheel Velocity Estimator
3. Supervisory Controller
    1. Traction Control
    2. Velocity and Drift Control

Simulation Parameter	Value
Solver	ode45
RelTol	1e-3
Refine	1
MaxOrder	5
ZeroCross	on

[more info]

System - NonlinearDynamics_DiscreteTraction_Slip/Supervisory Controller

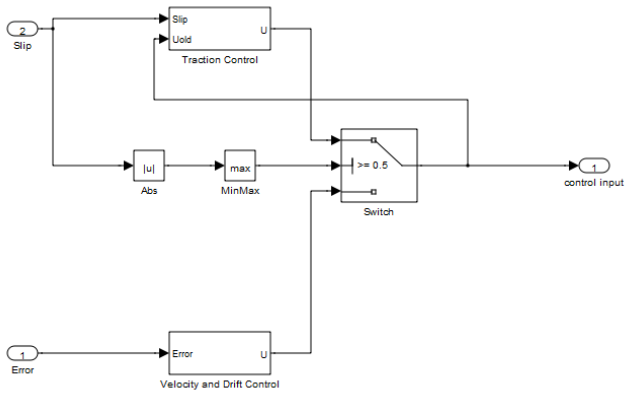


Table 1. Abs Block Properties

Name	Zero Cross	Out Data Type Str	Lock Scale	Rnd Meth	Saturate On Integer Overflow
Abs	on	Inherit: Same as input	off	Floor	off

Table 2. Inport Block Properties

Name	Port	Defined In Blk
Error	1	Sum
Slip	2	Switch1, Switch1

Table 3. MinMax Block Properties

Name	Function	Inputs	Input Same DT	Out Data Type Str	Lock Scale	Rnd Meth	Saturate On Integer Overflow	Zero Cross
MinMax	max	1	off	Inherit: Inherit via internal rule	off	Floor	off	on

Table 4. Outport Block Properties

Name	Port	Icon Display	Lock Scale	Var Size Sig	Signal Type	Sampling Mode	Source Of Initial Output Value	Output When Disabled	Used By Blk
control input	1	Port number	off	Inherit	auto	auto	Dialog	held	Unit Delay, Demux

Table 5. Switch Block Properties

Name	Criteria	Threshold	Input Same DT	Out Data Type Str	Lock Scale	Rnd Meth	Saturate On Integer Overflow	Zero Cross	Allow Diff Input Sizes
Switch	u2 >= Threshold	0.5	off	Inherit: Inherit via internal rule	off	Floor	off	on	off

System - NonlinearDynamics_DiscreteTraction_Slip/Suporvisory Controller/Traction Control

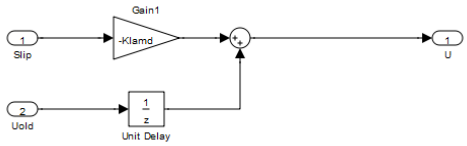


Table 6. Gain Block Properties

Name	Gain	Multiplication	Param Data Type Str	Out Data Type Str	Lock Scale	Rnd Meth	Saturate On Integer Overflow	Sample Time
Gain1	-Klamd	Element-wise(K.*u)	Inherit: Inherit via internal rule	Inherit: Inherit via internal rule	off	Floor	off	Ts

Table 7. Inport Block Properties

Name	Port	Defined In Blk
Slip	1	Switch1, Switch1
Uold	2	Switch

Table 8. Outport Block Properties

Name	Port	Icon Display	Lock Scale	Var Size Sig	Signal Type	Sampling Mode	Source Of Initial Output Value	Output When Disabled	Used By Blk
U	1	Port number	off	Inherit	auto	auto	Dialog	held	Switch

Table 9. Sum Block Properties

Name	Icon Shape	Inputs	Collapse Mode	Collapse Dim	Input Same DT	Accum Data Type Str	Out Data Type Str	Lock Scale	Rnd Meth	Saturate On Integer Overflow
Sum	round	|++	All dimensions	1	off	Inherit: Inherit via internal rule	Inherit: Inherit via internal rule	off	Floor	off

Table 10. UnitDelay Block Properties

Name	X0	Input Processing	Sample Time
Unit Delay	0	Elements as channels (sample based)	Ts

System - NonlinearDynamics_DiscreteTraction_Slip/Suporvisory Controller/Velocity and Drift Control

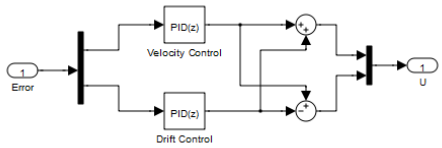


Table 11. Demux Block Properties

Name	Outputs	Display Option	Bus Selection Mode
Demux	2	bar	off

Table 12. Inport Block Properties

Name	Port	Defined In Blk
Error	1	Sum

Table 13. Mux Block Properties

Name	Inputs	Display Option
Mux	2	bar

Table 14. Outport Block Properties

Name	Port	Icon Display	Lock Scale	Var Size Sig	Signal Type	Sampling Mode	Source Of Initial Output Value	Output When Disabled	Used By Blk
U	1	Port number	off	Inherit	auto	auto	Dialog	held	Switch

Table 15. PID 1dof Block Properties

Name	Controller	Time Domain	Sample Time	Integrator Method	Filter Method	Form	P	I	D	N	Initial Condition Source	Initial Condition For Integrator	Initial Condition For Filter	External Reset	Zero Cross	Linearize As Gain	Anti Windup Mode	Kb	Tracking Mode	Kt	Rnd Meth	Saturate On Integer Overflow
Drift Control	PID	Discrete-time	Ts	Trapezoidal	Forward Euler	Parallel	Kpphi	Kiphi	Kdphi	100	internal	0	0	none	on	off	none	1	off	1	Floor	off
Velocity Control	PID	Discrete-time	Ts	Trapezoidal	Forward Euler	Parallel	Kpv	Kiv	Kdv	100	internal	0	0	none	on	off	none	1	off	1	Floor	off

Table 16. Sum Block Properties

Name	Icon Shape	Inputs	Collapse Mode	Collapse Dim	Input Same DT	Accum Data Type Str	Out Data Type Str	Lock Scale	Rnd Meth	Saturate On Integer Overflow
Sum	round	|++	All dimensions	1	off	Inherit: Inherit via internal rule	Inherit: Inherit via internal rule	off	Floor	off
Sum1	round	+-|	All dimensions	1	off	Inherit: Inherit via internal rule	Inherit: Inherit via internal rule	off	Floor	off

Appendix

Table 17. Block Type Count

BlockType	Count	Block Names
Inport	5	Error, Slip, Slip, Uold, Error
Sum	3	Sum, Sum, Sum1
SubSystem	3	Supervisory Controller, Traction Control, Velocity and Drift Control
Outport	3	U, U, control input
PID 1dof (m)	2	Drift Control, Velocity Control
UnitDelay	1	Unit Delay
Switch	1	Switch
Mux	1	Mux
MinMax	1	MinMax
Gain	1	Gain1
Demux	1	Demux
Abs	1	Abs