ABSTRACT

Title of Document: Simulation and Control of a Passively Articulated, Segmented-Body Rover

Timothy Andrew Wasserman, Master of Science, 2007

Directed By: Professor David L. Akin
Department of Aerospace Engineering

Mobility will be a key aspect of future planetary surface missions. A rover with several segments connected by rotary joints promises much capability in terrain traversal, but is not well understood. In this thesis, a computer model was built to simulate the movements of a passively articulated, segmented-body rover. Its main components are a linearized soil-wheel interaction model, a Newton-Euler based dynamic model, and a PD control module to regulate steering and handle disturbances. The simulation outputs were compared against results from past research on fixed-chassis vehicles. Next, the simulation was used to investigate the driving and turning behavior of articulated vehicles, and their controllability using a simple control system. It was found that the vehicle is relatively stable, and that simple control is possible.

SIMULATION AND CONTROL OF A PASSIVELY ARTICULATED, SEGMENTED-BODY ROVER

By

Timothy Andrew Wasserman

Thesis submitted to the Faculty of the Graduate School of the
University of Maryland, College Park, in partial fulfillment
of the requirements for the degree of
Master of Science
2007

Advisory Committee:
Professor David L. Akin, Chair
Research Associate Craig R. Carignan
Professor Robert M. Sanner

Dedication

To my family.

Acknowledgements

First I would like to thank Dave Akin for taking me on as one of his graduate students at the SSL and letting me pursue the research I wanted to do. I have had an excellent time working at the SSL over the past four years. 90% of that excellent time was spent with its faculty, staff, and students (the other 10% was with robots). Thank you to Agnieszka, Dave, John, and Liz who kept me on my toes and were always up for a distraction or refreshing conversation. Thanks to Rico, Nick, Leon, and Max for keeping the office entertaining and being great reality checks for my research. Thanks to Mike Liszka for being a bad influence.

I also have to thank the SSL staff (present and former) for always taking the time to answer my questions. This is a first for me, but I'm going to thank a piece of furniture, the cot in our office. This thesis would not have been completed yet if it were not for those reinvigorating late afternoon power naps. Also, Go Eulers!

There are several people outside of the lab who I am lucky to call friends, and are my connections to all things non-engineering. Thank you Katie S, Harry, Nadim, Katie M, and Allison. These great people helped to nurture my sanity over the past two years.

To Madeline, thank you. You have been being an amazing support, super-fun, and a fantastic girlfriend. Finally, my family deserves special thanks for letting me figure things out myself, but always being there when I need help. I love you guys.

Table of Contents

Dedication ii

Acknowledgements iii

Table of Contents iv

List of Tables vii

List of Figures viii

Chapter 1: Introduction 1

Chapter 2: Previous Work and Background Theory 4

2.1 Terramechanics 4

2.1.1 Basic Soil/Wheel Relationships 4

2.1.2 Wheel Coordinate System and Slip 5

2.2.3 Primary Wheel Forces 7

2.1.4 Stress Distributions 8

2.1.5 Closed Form Wheel Force Solutions 10

2.2 Articulated Vehicle Modeling and Simulation 12

Chapter 3: Modeling 17

3.1 Kinematics 17

3.1.1 Vehicle Configuration 17

3.1.2 Coordinate Frames and Denavit-Hartenberg Parameters 18

3.2 Soil/Wheel Interactions 21

3.2.1 Finding Sinkage Depth 21

3.2.2 Finding Slip Angle 23

3.2.3 Finding Slip Ratio 25
3.2.4 Wheel Force Vector Construction 26
3.3 Rover Dynamic Model 27
3.3.1 Modified Newton-Euler Method 27
3.3.2 Equation of Motion 28
Chapter 4: Simulation Design 29
4.1 Simulation Outline 29
4.2 Simulation Testing and Validation 30
4.2.1 Comparison to Bekker Model 31
4.2.2 Comparison to Yoshida Model 33
4.2.3 Constant Velocity Driving 39
Chapter 5: Articulated Rover Simulation 42
5.1 Forward Movement 42
5.1.1 Nominal Case 42
5.1.2 Disturbed Behavior 42
5.2 PD Control 46
5.3.1 Controller Design 47
5.3.2 Response to Disturbances 47
5.3.3 Turning Performance 51
Chapter 6: Summary, Conclusions, Future Work 57
6.1 Summary 57
6.2 Conclusions 57
6.3 Future Work 58

Appendices 60

Appendix A: Matlab Simulation Code 60

Appendix B: Mathematica® Dynamic Modeling 74

Appendix C: Soil Parameters 81

References..... 82

List of Tables

Table 3.1: Denavit-Hartenberg Parameters 20

Table 4.1: Rolling Resistance vs. Wheel diameter, Width, and Weight 31

Table 4.2: Wheel Sinkage vs. Wheel Diameter, Width, and Weight 31

Table 4.3: Rolling Resistance vs. Soil Parameters 32

Table 4.4: Wheel Sinkage vs. Soil Parameters 33

Table 4.5: Test Rover Configuration Parameters 39

Table 4.6: Steady State Test Values 39

Table 4.7: Expected Acceleration Test Values 40

Table 4.8: Simulated Acceleration Test Values 40

Table 5.1: Turning Performance at 0.5 m/s 51

Table 5.2: Turning Performance at 3 m/s 52

List of Figures

Figure 2.1: Wheel Frame Coordinate System 6

Figure 2.2: Wheel Slip [Ishigami07] 6

Figure 2.3: Wheel Stress Distributions [Ishigami07] 8

Figure 2.4: Typical Stress Distributions 10

Figure 2.5: Approximated vs. Theoretical Stress Distributions 11

Figure 2.6: Oida's Tractor [Oida,87] 13

Figure 2.7: Waterloo articulated vehicle schematic [He,05] 14

Figure 2.8: Yoshida Articulated Rover Illustration [Yoshida98] 15

Figure 2.9: Yoshida Experimental Rover [Ishigami07] 16

Figure 3.1: Articulated Vehicle Top View 17

Figure 3.2: Articulated Vehicle Side View 18

Figure 3.3: Denavit-Hartenberg Frame Assignments 19

Figure 4.1: Rolling Resistance and Sinkage vs. Wheel Diameter 32

Figure 4.2: Yoshida drawbar pull vs. slip ratio for different slip angles 34

Figure 4.3: Yoshida Side Forces vs. slip ratio for different slip angles 35

Figure 4.4: Simulation Model Drawbar Pull vs. slip ratio for different slip angles 36

Figure 4.5: Simulation Model Side Forces vs. slip ratio for different slip angles 36

Figure 4.6: Approximated vs. Exact Stress Distribution for non-zero θ_2 37

Figure 4.7: Rover at Space Systems Laboratory 37

Figure 4.8: Lunar Roving Vehicle (NASA) 38

Figure 5.1: Low Speed Disturbance, Turning Joint Angle vs. Time 43

Figure 5.2: Low Speed Disturbance, Maximum Turning Joint Angles 44

Figure 5.3: Medium Speed Disturbanc, Turning Joint Angle vs. Time 44

Figure 5.4: Medium Speed Disturbance, Maximum Turning Joint Angles 45

Figure 5.5: High Speed Disturbance, Maximum Turning Joint Angles..... 45

Figure 5.6: High Speed Disturbed Rover Path 46

Figure 5.7: Disturbance Effects for v = 0.5 m/s..... 48

Figure 5.8: Disturbance Effects for v = 3 m/s 49

Figure 5.9: Disturbance Effects for v = 6 m/s 49

Figure 5.10: Response of Turning Joint with PD Control at High Speeds 50

Figure 5.11: Settling Time and Power Requirement Comparisons 50

Figure 5.12: High Speed Turning Joint Response with Alternate Gains 51

Figure 5.13: Low Speed Turning Response 52

Figure 5.14: High Speed Turning Response..... 53

Figure 5.15: Turning Joint Value During Maneuver 54

Figure 5.16: Vehicle Heading Angle During Maneuver 54

Figure 5.17: Rover Path During Maneuver 55

Figure 5.18: Rover Wheel Sinkages During Maneuver 55

Figure 5.19: Wheel Torques During Maneuver..... 56

Chapter 1: Introduction

Mobility will be key to the success of future planetary surface exploration missions, both human and robotic. Quick transits to and from sites of interest and the ability to traverse a variety of challenging terrain are capabilities that scientists, mission planners, and planetary resource prospectors can all agree on. Mass (and ultimately, cost) limitations on recent planetary rover programs have led to the paradigm of low power rovers landed in the most scientifically dense sites achievable by their delivery systems, and fitted with ambitious scientific payloads.

Because of this low power compromise, these rovers cannot achieve large velocities. The maximum speeds of the Sojourner, Mars Exploration Rover (MER), and planned Mars Science Laboratory (MSL) vehicles are on the order of several centimeters per second. Additionally, their "rocker-bogie" type suspensions, while very effective at climbing over obstacles, are typically not suited for high speeds [Miller,02]. The Lunar Roving Vehicle (LRV), used by astronauts during the last three Apollo missions to the moon, averaged 9.6 km/hr during traverses. It was designed to cruise over the lunar regolith on slopes up to 25 degrees and climb over obstacles 30 cm in height, but was never intended to negotiate through the moon's more rocky, crater-pocketed terrain [Sullivan,94].

One method of increasing rover mobility is to add articulation directly to the chassis. A rover would then be comprised of multiple segments; each with its own set of wheels, and connected by a rotational joint. Resembling a train off of its tracks, a segmented-body rover would efficiently conform to terrain as would a "rocker-

bogie" equipped vehicle, but would not preclude accommodation of an independent wheel suspension scheme for use in high-speed, rough terrain traverses.

The idea for this type of vehicle has been around since the early 20th century. Since the 1930s, articulated vehicles have been used extensively in the agricultural, forestry, construction, and earthmoving industries. Military designers have also taken notice. The United States Army researched many concept vehicles during the 1960s. The "Gama Goat", a two segmented, six-wheeled vehicle entered service in 1968 and remained in service until replaced by the Humvee, decades later [Holm,70]. Today, companies such as Caterpillar, Volvo, Terex, Foremost, and Holder all sell their own lines of articulated vehicles.

The vast majority of segmented-body vehicles produced to date have steered via the use of hydraulically actuated joints. On Earth, this solution provides powerful turning capability, even when the vehicle is standing still [Holm,70]. However, hydraulic systems tend to be massive, making them ill-suited for interplanetary payloads. Additionally, the lubrication and sealant agents used on Earth systems typically will not work in the temperature and low-pressure extremes present on the surface of the Moon and Mars. One solution would be to replace the hydraulics with an electrical joint actuator. Another solution would be to implement Ackermann-type steering on the front and rear segments (as in the LRV). However, neither of these suggestions reduces the overall mechanical complexity of the system.

An alternative option—and the one investigated in this thesis—uses passive inter-segment joints. Steering is accomplished by carefully controlling the current output to each of the rover's wheel motors. The resulting wheel torques cause soil

reaction forces that move the segments and affect the overall turning behavior of the vehicle, including turning joint motions. Besides the promise of enhanced mobility, the passively articulated segmented-body rover would be beneficial operationally. Its segments do not need to be shipped to the planet's surface together, or they could arrive in a volume-optimized disconnected configuration. The design could also take on a modular approach, with specialized segments for a range of surface activities including cargo transport, field geology, regolith processing, and base construction. If such a vehicle is easily controllable and can make efficient use of power during the cruising, turning, and obstacle climbing phases of travel, then it should be considered a viable option for future planetary surface exploration rovers.

Background theory in terramechanics and previous work on segmented body rovers is reviewed in Chapter 2. The computer model for the rover is developed in Chapter 3. In Chapter 4, the simulation construction and validation tests are discussed. Chapter 5 describes the simulations that were performed on articulated vehicles. Finally, Chapter 6 contains a summary of the research, conclusions reached, and suggestions for possible future work.

Chapter 2: Previous Work and Background Theory

This chapter introduces the background theory required to model an articulated, off-road vehicle as well as reviews past research into the topic.

2.1 Terramechanics

2.1.1 Basic Soil/Wheel Relationships

Modeling of any terrain-vehicle system begins at the terrain-vehicle interface, the wheels. Empirical work on the nature of soil-wheel interaction began in the early 20th century, with many important breakthroughs and accompanying theory coming after World War II [Holm,70]. The soil-wheel interactions this research is primarily concerned with are that of rigid wheels on deformable terrain (as opposed to deformable wheels on rigid terrain, e.g. pneumatic tires on paved roads).

Mieczysław G. Bekker, a leading researcher in the field of terramechanics introduced the following equation in 1960 [Bekker,60]:

$$p(h) = (k_c / b + k_\phi) h^n \tag{2.1}$$

It relates the pressure that results underneath a flat plate of width b to its sinkage depth h. n, k_c, and k_ϕ are soil parameters (sinkage exponent, cohesive modulus, and frictional modulus respectively) that can be measured experimentally. Using this pressure-sinkage relationship, Bekker derived the following two formulae for wheels:

$$z_0 = \left[\frac{3W}{(3-n)(k_c + bk_\phi)\sqrt{D}} \right]^{\frac{2}{2n+1}} \tag{2.2}$$

$$R_c = \frac{(3W)^{\frac{2n+2}{2n+1}}}{(3-n)^{\frac{2n+2}{2n+1}}(n+1)(k_c + bk_\phi)^{\frac{1}{2n+1}} D^{\frac{n+1}{2n+1}}} \tag{2.3}$$

The first equation allows calculation of a wheel's sinkage. Given weight W on a wheel of diameter D and width b, it will sink to a depth z_0 in a given soil. The second equation describes the wheel's rolling resistance due to soil compaction [Bekker,60]. It takes energy for a rolling wheel to compact soil over a distance. If R_c is the only resistance to movement, then to maintain a velocity v, driving power P = R_cv must be provided to the wheel. While very useful in determining baseline requirements for wheeled vehicles over a variety of measurable terrain, these equations do not take into account wheel slip in loose soils, cornering forces experienced during turning, or the contributions of a torqued wheel. A more complete set of equations is needed for accurate simulation.

2.1.2 Wheel Coordinate System and Slip

Before going further, it will be useful to define a standard set of wheel coordinates and introduce the concept of wheel slip. Figure 2.1 depicts the wheel frame coordinate system.

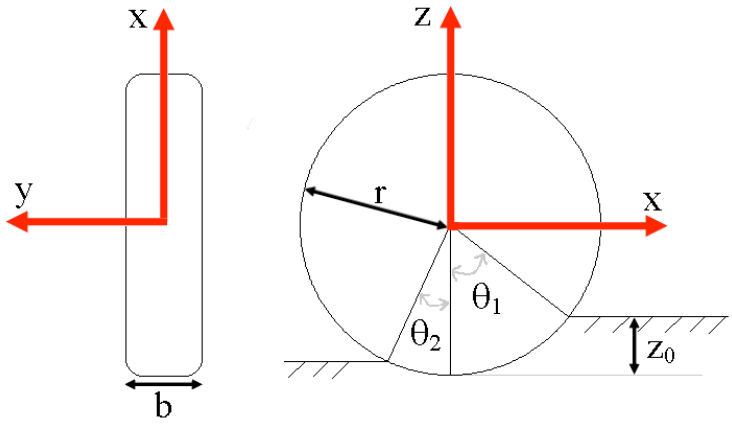

Figure 2.1: Wheel Frame Coordinate System

The x, or longitudinal axis is in the direction of travel, the z-axis is directed vertically, and the y-axis points in the lateral direction, out the wheel's side. Centered at the wheel hub, θ is the angle measured from bottom-dead-center, with positive values being in the direction of travel. θ_1 refers to the entry angle, and θ_2 to the exit angle.

Wheel slip becomes an important factor when traveling through loose soil. A wheel can slip in two ways: longitudinally and laterally, as shown in Figure 2.2.

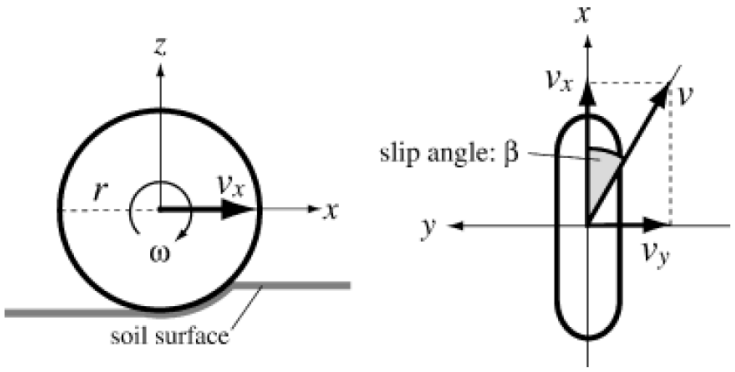

Figure 2.2: Wheel Slip [Ishigami07]

Longitudinal slip occurs when a wheel's traveling velocity v_x differs from its

rim velocity, rω. It can take on a value between -1 and 1. A positive slip ratio (s) indicates a driven wheel, while a negative s indicates braking.

$$s = \begin{cases} (r\omega - v_x)/r\omega & |r\omega| > |v_x| \\ (r\omega - v_x)/v_x & |r\omega| < |v_x| \end{cases} \tag{2.4}$$

Lateral slip occurs when a wheel moves sideways as well as forwards, and is quantified by the slip angle β. It is the angle a wheel's traveling velocity makes with its longitudinal axis [Ishigami,07].

$$\beta = \tan^{-1}(v_y/v_x) \tag{2.5}$$

2.2.3 Primary Wheel Forces

There are three major stress distributions present along the soil contact area of a driven rigid wheel. First is the normal stress σ(θ), acting radially towards the wheel's center. Next is the longitudinal shear stress $\tau_x(\theta)$, which acts tangentially along the wheel's circumference. Finally, the lateral shear stress $\tau_y(\theta)$ acts tangentially across the wheel's width. Figure 2.3 depicts these stress distributions. Note that the maximum value of each stress distribution (σ_m, τ_{xm}, and τ_{ym}) occurs at the same value of $\theta = \theta_m$. This assumption agrees well with experimental results [Wong,67].

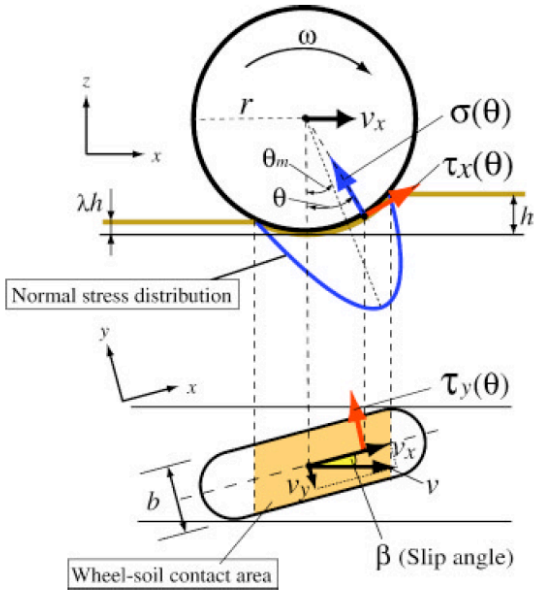

Figure 2.3: Wheel Stress Distributions [Ishigami07]

Given the stress distributions described above, one may calculate forces in the three primary wheel axes and torque about the wheel's axle as follows [Wong,67].

$$F_x = rb\left\{\int_{\theta_2}^{\theta_1} \tau_x(\theta)\cos\theta d\theta - \int_{\theta_2}^{\theta_1} \sigma(\theta)\sin\theta d\theta\right\} \tag{2.6}$$

$$F_z = rb\left\{\int_{\theta_2}^{\theta_1} \sigma(\theta)\cos\theta d\theta + \int_{\theta_2}^{\theta_1} \tau_x(\theta)\sin\theta d\theta\right\} \tag{2.7}$$

$$F_y = rb\int_{\theta_2}^{\theta_1} \tau_y(\theta) d\theta \tag{2.8}$$

$$T = r^2 b\int_{\theta_2}^{\theta_1} \tau_x(\theta) d\theta \tag{2.9}$$

2.1.4 Stress Distributions

Much effort has been put into finding expressions for the various stress

distributions. In 1967, Wong and Reece introduced a new way to model them that includes the effect of wheel slip. Yoshida et al. modified these equations in 2003 to incorporate Bekker's form of the pressure-sinkage relationship. For normal stress [Ishigami,07],

$$\sigma(\theta) = \begin{cases} r^n\left(\frac{k_c}{b} + k_\phi\right)[\cos\theta - \cos\theta_1]^n & (\theta_m \le \theta < \theta_1) \\ r^n\left(\frac{k_c}{b} + k_\phi\right)\left[\cos\left\{\theta_1 - \frac{\theta - \theta_2}{\theta_m - \theta_2}(\theta_1 - \theta_m)\right\} - \cos\theta_1\right]^n & (\theta_2 < \theta \le \theta_m) \end{cases} \quad (2.10)$$

where θ_m is cacluated by using the linear equation:

$$\theta_m = (a_0 + a_1 s)\theta_1 \quad (2.11)$$

The a_0 and a_1 coefficients are empirically measured for a soil. It can be seen that the location of maximum stress moves forward with increasing slip. The normal stress is maximum at θ_m, and falls off on either side to zero at the soil contact points [Wong,67].

Similarly, for shear stress in the longitudinal and lateral directions:

$$\tau_x(\theta) = (c + \sigma(\theta)\tan\phi)\left[1 - e^{-\frac{r}{k_x}(\theta_1 - \theta - (1-s)(\sin\theta_1 - \sin\theta))}\right] \quad (2.12)$$

$$\tau_y(\theta) = (c + \sigma(\theta)\tan\phi)\left[1 - e^{-\frac{r}{k_y}(1-s)(\theta_1 - \theta)\tan\beta}\right] \quad (2.13)$$

c is cohesion, ϕ is the internal friction angle, and k_x and k_y are shear deformation modules; all soil parameters measurable by experiment [Ishigami,07]. Figure 2.4 shows a typical example of what the stress distributions look like over the contact area of the wheel.

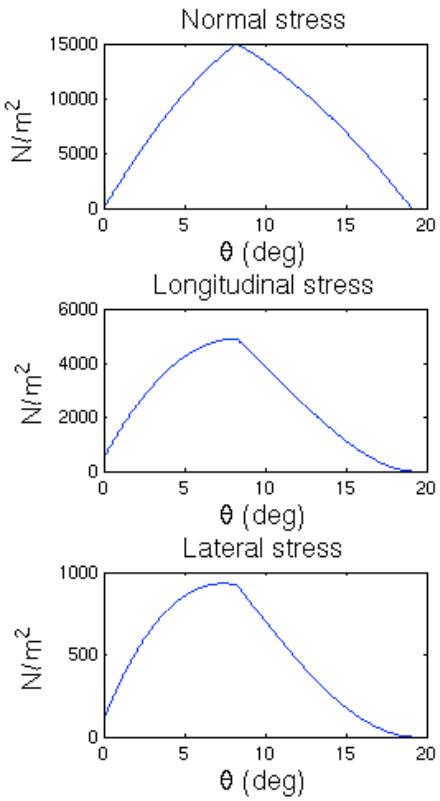

Figure 2.4: Typical Stress Distributions

2.1.5 Closed Form Wheel Force Solutions

Now that the stress distributions are known, Equations 2.10-2.13 can be substituted into Equations 2.6-2.9 to solve for the wheel forces. However, this requires numerical integration, as there are no closed-form solutions to these equations due to the complex form of the stress distributions. For computational purposes, it would be useful to find approximated versions of the stress distributions.

In 1961, Vincent noted that for a variety of soils and slips, the stress distribution curves follow a triangular curve [Vincent,61]. In 2001 Iagnemma expanded on Vincent's observation, assumed that $\theta_2 = 0$ (accurate for soils with low cohesion), and used the following linear approximation to the stress distributions:

$$\alpha(\theta) = \begin{cases} \dfrac{\theta}{\theta_m}\alpha_m & 0 \le \theta \le \theta_m \\ \dfrac{\theta_1 - \theta}{\theta_1 - \theta_m}\alpha_m & \theta_m < \theta \le \theta_1 \end{cases} \tag{2.14}$$

where α can be any of the primary stress distributions (and α_m its maximum value). Figure 2.5 shows a comparison of the approximated distributions and the more exact solutions.

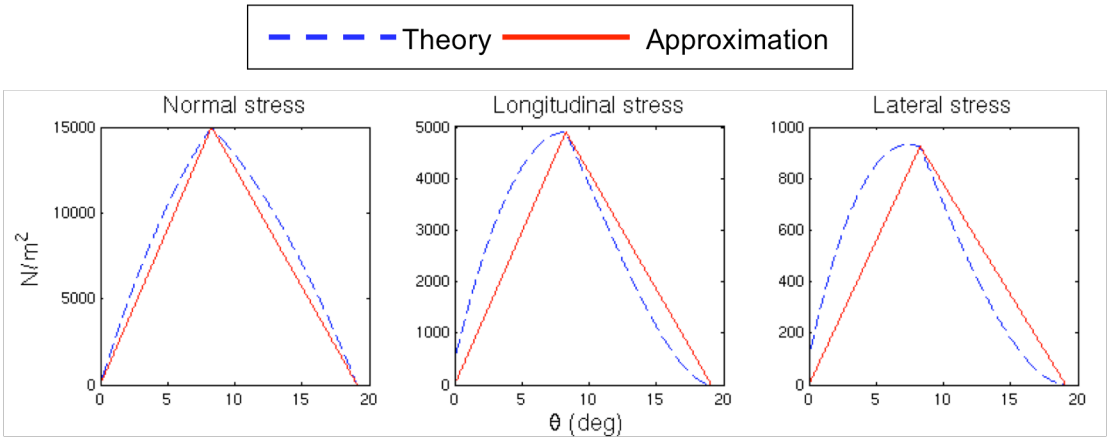

Figure 2.5: Approximated vs. Theoretical Stress Distributions

Substituting these equations into Equations 2.6-2.9 and integrating, the following closed-form solutions can now be found [Iagnemma,01].

$$F_x = \frac{rb}{\theta_m(\theta_1 - \theta_m)}\left[\tau_{xm}(\theta_1\cos\theta_m - \theta_m\cos\theta_1 - \theta_1 + \theta_m) - \sigma_m(\theta_1\sin\theta_m - \theta_m\sin\theta_1)\right] \tag{2.15}$$

$$F_z = \frac{rb}{\theta_m(\theta_1 - \theta_m)}\left[\sigma_m(\theta_1\cos\theta_m - \theta_m\cos\theta_1 - \theta_1 + \theta_m) + \tau_{xm}(\theta_1\sin\theta_m - \theta_m\sin\theta_1)\right] \tag{2.16}$$

$$F_y = \frac{1}{2}rb\tau_{ym}\theta_1 \tag{2.17}$$

$$T = \frac{1}{2}r^2b\tau_{xm}\theta_1 \tag{2.18}$$

The maximum stresses are found by substituting θ_m into the original stress

equations.

$$\sigma_m = r^n \left(\frac{k_c}{b} + k_\phi \right) [\cos\theta_m - \cos\theta_1]^n \tag{2.19}$$

$$\tau_{xm} = (c + \sigma_m \tan\phi) \left[1 - e^{-\frac{r}{k_x}((\theta_1 - \theta_m) - (1-s)(\sin\theta_1 - \sin\theta_m))} \right] \tag{2.20}$$

$$\tau_{ym} = (c + \sigma_m \tan\phi) \left[1 - e^{-\frac{r}{k_y}(1-s)(\theta_1 - \theta_m)\tan\beta} \right] \tag{2.21}$$

Now all forces on the wheel can be represented by algebraic functions of just soil parameters, wheel dimensions, and three "state" variables: slip ratio, slip angle, and entry angle. These simplified wheel force equations will make it much easier to simulate the motions of a passively articulated, segmented body rover.

2.2 Articulated Vehicle Modeling and Simulation

Beginning in the 1980s, several studies were conducted by researchers to model and simulate articulated vehicles to gain insight into their behavior. Most of these models incorporate joint actuation, but are still relevant to the passively articulated problem.

Oida's 1987 paper modeled an articulated tractor (see Figure 2.6); specifically its turning characteristics. Wheel forces were heavily simplified, with traction, rolling resistance, and cornering forces directly proportional to wheel load. Cornering forces also had slip angle dependency, but longitudinal slip was ignored. Additionally, during simulation the turning joint angle was rigidly fixed. The vehicle's equations of motion were determined in the planar case by summing the

forces in x and y directions and the moments about the turning joint.

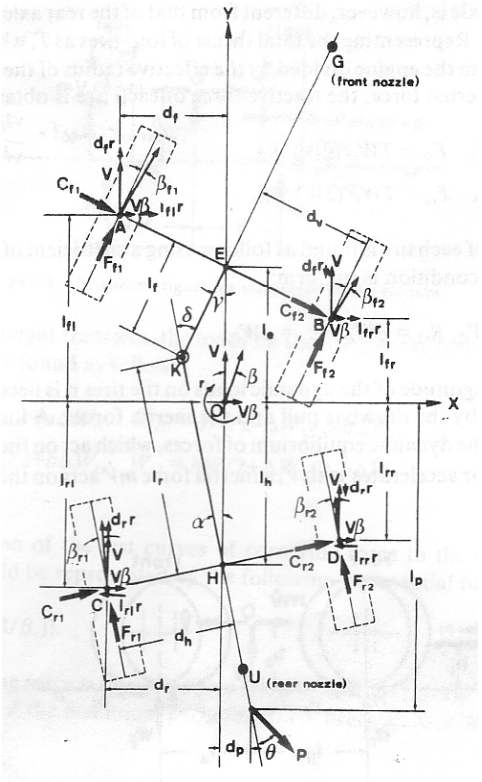

Figure 2.6: Oida's Tractor [Oida,87]

A tractor was simulated taking turns at various joint angles and speeds. Simulation results were compared with the results of real-world experiments involving an actual tractor. The simulation and experiment agreed well with observation, so Oida used the simulation to determine the effects of changing the turning joint and center of gravity (CG) locations. While this research demonstrated the feasibility of articulated vehicle simulation, it left much room for improvement in the accurate modeling of soil forces and dynamics [Oida,87].

More recently, research conducted at the University of Waterloo has focused on the dynamic modeling and stability analysis of articulated-steer vehicles both on and off-road (see Figure 2.7).

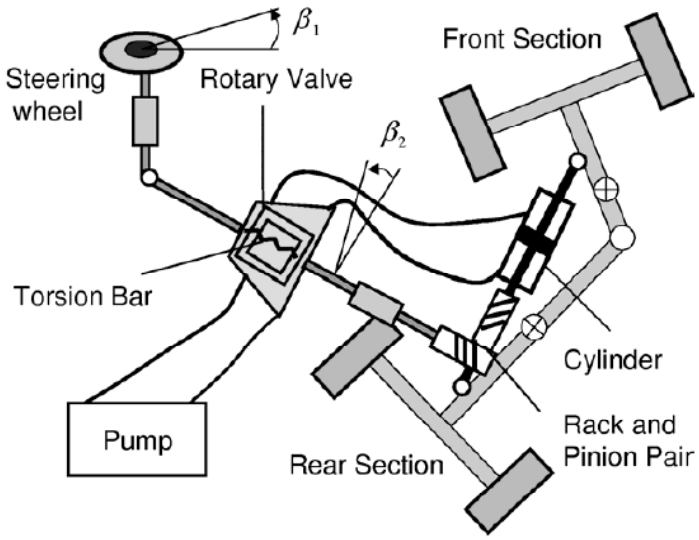

Figure 2.7: Waterloo articulated vehicle schematic [He,05]

Their papers have given insight into the conditions that lead to jack-knife and snaking, two dynamic instabilities experienced by articulated vehicles. Their wheel model is more advanced than Oida's, and incorporates slip, but is still simplified and less theoretically backed than the Bekker-Wong-Yoshida equations [He,05] and [Azad,05]. Azad et al.'s 2007 paper introduces a feedback controller to stabilize an articulated vehicle undergoing snaking motion [Azad,07].

In 1998, Yoshida and Shiwa simulated a rubber-tired articulated vehicle with three segments traveling on a hard surface (See Figure 2.8). The front inter-segment joint had roll and yaw degrees of freedom (DOF), while the rear joint had 3 DOF. Pitch and yaw joints were compliant (torsional springs), while roll joints were free. The four wheels on the front two segments were driven. The two wheels on the third segment rotated freely and were used as odometers to help determine slip rates and improve dead-reckoning.

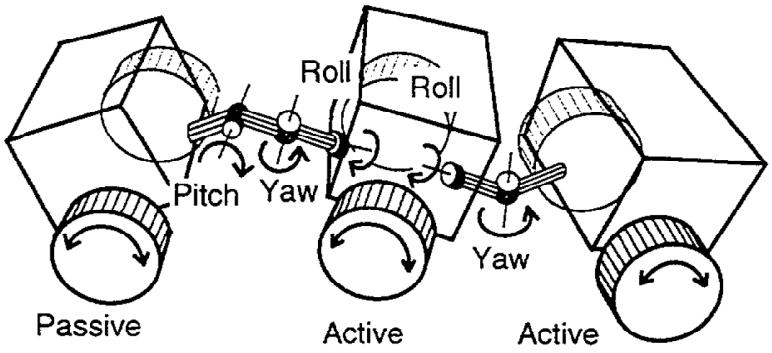

Figure 2.8: Yoshida Articulated Rover Illustration [Yoshida98]

The tire model used includes the effect of wheel slip, as well as rubber wheel stiffness and damping. Vehicle and articulation dynamics were computed by the SpaceDyn™ toolbox, which accommodates an arbitrarily articulated system with multiple branches. Given the vehicle's state, the tire forces were calculated and applied at the ends of the branches. The researchers also constructed an experimental testbed vehicle to complement the simulations. The one simulation reported in their paper involved commanding sinusoidal velocity commands to the left and right wheels of opposite signs to elicit a weaving response. Results from the simulation and experiment were qualitatively consistent with each other, and the path traveled and range of slip ratio values showed good agreement [Yoshida,98].

Since then, Yoshida's group has gone on to advance their rover vehicle simulation and integrate it with the non-simplified terramechanics equations discussed in the previous subsection, but not for a segmented-body vehicle. They compare their simulations with experimental tests conducted with a 35 kg four-wheel drive, four-wheel steer rover with rocker suspension (as shown in Figure 2.9).

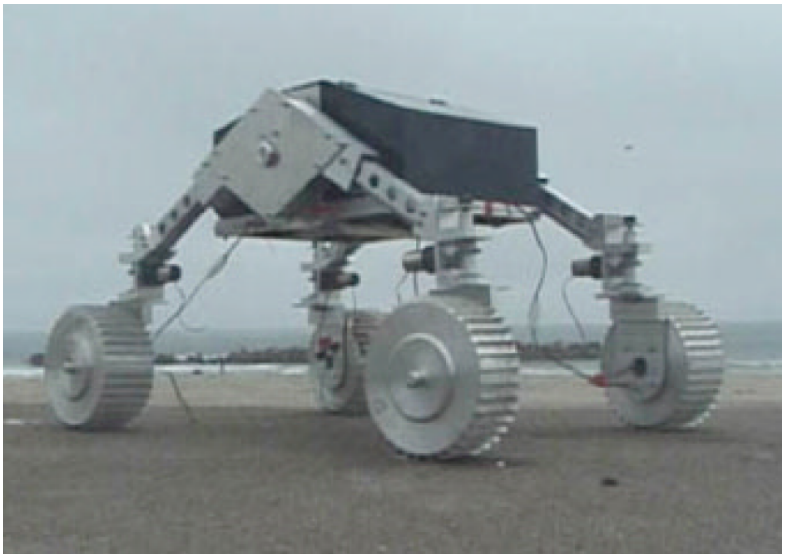

Figure 2.9: Yoshida Experimental Rover [Ishigami07]

Yoshida's group has also had success simulating the motion of a rover on sloped ground. Inputs to the simulation are wheel rotation rates, which are kept constant during simulation runs. Typical ground speeds ranged from 1 to 8 cm/s [Yoshida,04], [Ishigami,05] and [Ishigami,07].

Chapter 3: Modeling

Chapter 3 discusses the various models that will be used as components in the simulation. Section 3.1 details the overall vehicle configuration and its kinematic description. The wheel/soil interaction model used is shown in Section 3.2, and the dynamic model is described in section 3.3.

3.1 Kinematics

3.1.1 Vehicle Configuration

A two-segmented rover is modeled in this research. Each segment has mass, rotational inertia, and two independently driven wheels. The segments—each of length ℓ—are connected by a passive yaw joint. Figures 3.1 and 3.2 show the vehicle schematic.

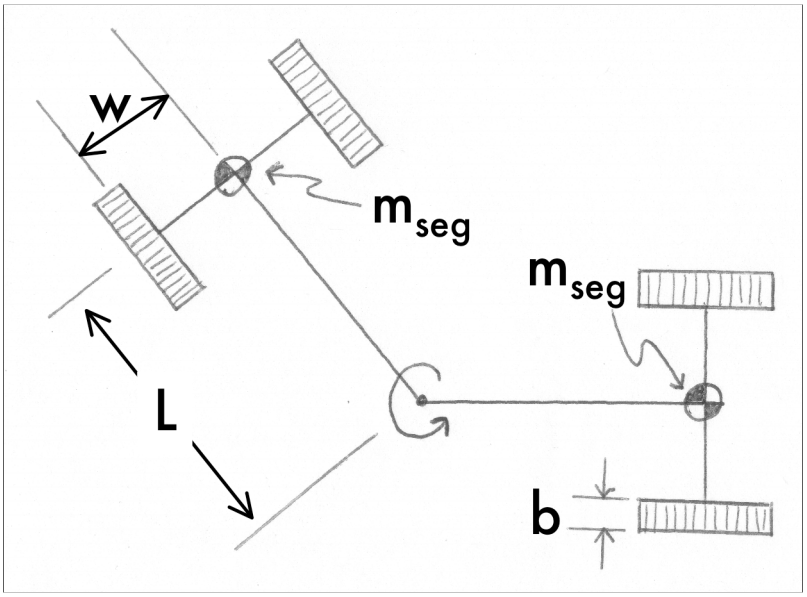

Figure 3.1: Articulated Vehicle Top View

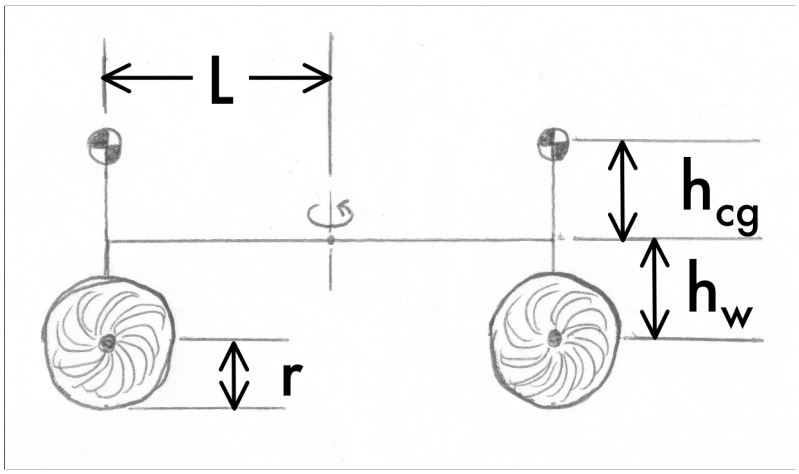

Figure 3.2: Articulated Vehicle Side View

3.1.2 Coordinate Frames and Denavit-Hartenberg Parameters

This description may be represented analytically using Denavit-Hartenberg (DH) notation [Craig,05]. This type of notation is widely used to specify the kinematics of manipulators, but can be applied to any serial-chain mechanism. The rover can be moved to any location and oriented toward any direction in space, which requires 6 degrees of freedom (DOF). In addition, the rover has one mechanical DOF at its yaw joint, bringing the total to 7 DOF. In DH notation, each DOF is modeled as a joint. The first three joints are chosen to be prismatic, and the following three joints are rotational. The last joint is the inter-segment joint. In this scheme, the rear segment is assigned to be the "base" and is coincident in space with the three orientation joints. In essence, the rover can be thought of as a 3 DOF Cartesian manipulator with a 3 DOF wrist, and a 1 DOF end effector. Following the DH convention, the coordinate frames were assigned as shown in Figure 3.3. Table 3.1 shows the DH parameters derived from this arrangement.

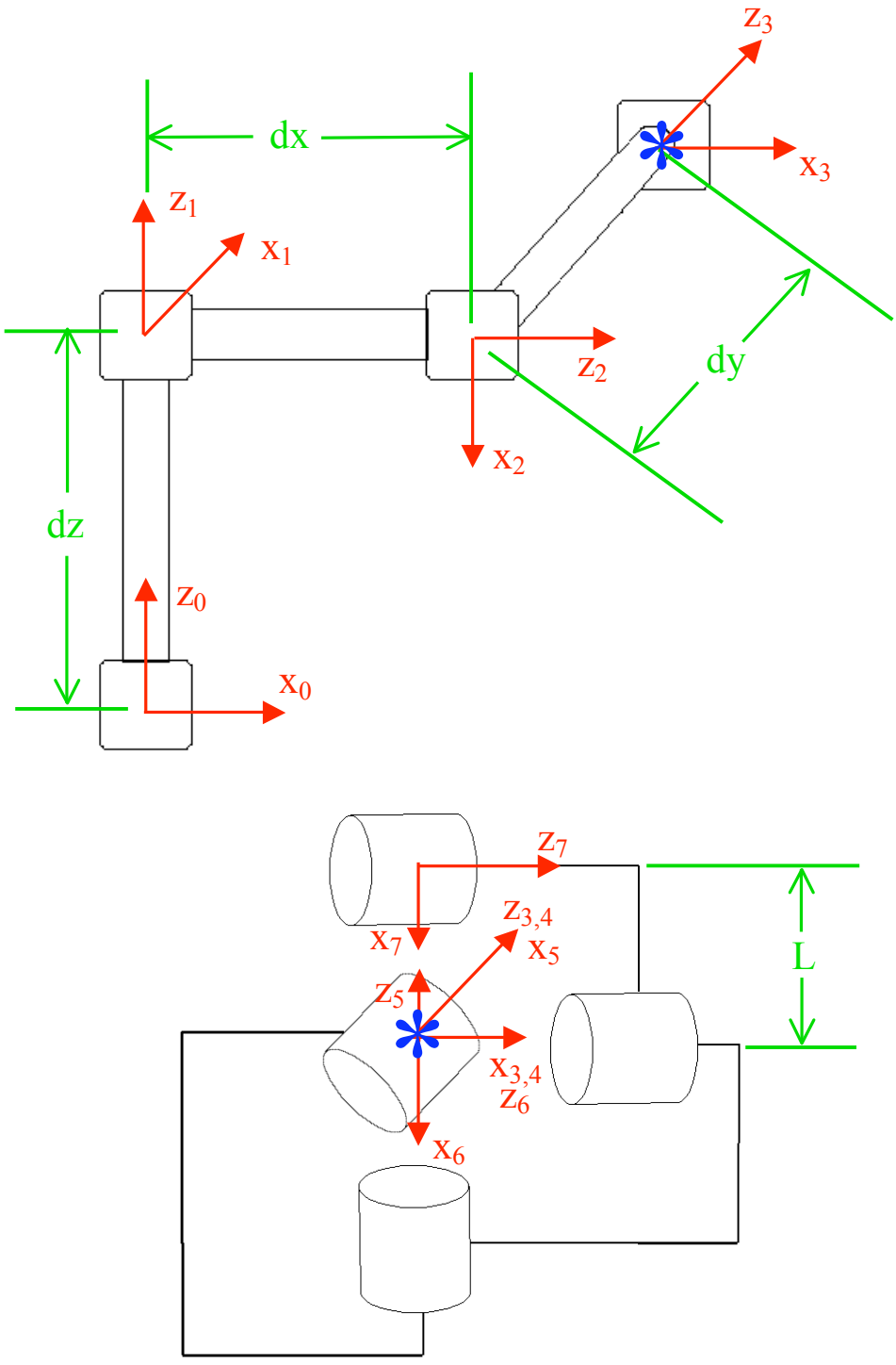

Figure 3.3: Denavit-Hartenberg Frame Assignments

Table 3.1: Denavit-Hartenberg Parameters

i	α_{i-1}	a_{i-1}	d_i	θ_i
1	0	0	dz	$\pi/2$
2	$\pi/2$	0	dx	$-\pi/2$
3	$-\pi/2$	0	dy	$-\pi/2$
4	0	0	0	ϕ_y
5	$\pi/2$	0	0	$\pi/2 + \phi_z$
6	$\pi/2$	0	0	$-\pi/2 + \phi_x$
7	0	L	0	θ_7

^{0}[dx dy dz] are the coordinates of the rear segment in the base frame. ϕ_x, ϕ_y, and ϕ_z specify the yaw, pitch, and roll, respectively, of the rear segment. θ_7 is the value of the rover's turning joint.

Positions of wheels and masses will be important for the simulation, and now that link frames are defined, they may be specified.

$$ {}^6p_w = \begin{bmatrix} 0 \\ \pm w \\ h_w \end{bmatrix}, \; {}^7p_w = \begin{bmatrix} \ell \\ \pm w \\ h_w \end{bmatrix} \tag{3.1} $$

^6p_w and ^7p_w are the wheel hub locations in the rear and front segment frames, respectively. The wheels are located at the ends of each segment, a height h_w below the turning joint plane, and distance ±w (left and right) in the lateral direction. Each wheel is a rigid cylinder with radius r and width b.

The positions of the segment centers of gravity (CG) are as follows:

$$ {}^{6}p_{cg} = \begin{bmatrix} 0 \\ 0 \\ h_{cg} \end{bmatrix}, \; {}^{7}p_{cg} = \begin{bmatrix} \ell \\ 0 \\ h_{cg} \end{bmatrix} \tag{3.2} $$

The masses, m_{seg}, are located a distance h above the crossing of the segment centerline and axle. A 3 x 3 diagonal inertia matrix I accompanies each mass.

$$ I = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \tag{3.3} $$

The masses are approximated as spheres with radius $w/2$, so that

$$ I_{xx} = I_{yy} = I_{zz} = \frac{1}{10} m_{seg} w^2 \tag{3.4} $$

It should be noted that this kinematic model is easily extendable to rovers with three or more segments, and arbitrary values and distributions of mass, inertia, and wheel properties.

3.2 Soil/Wheel Interactions

As outlined in the previous work section, Equations 2.15-2.18 will be used to calculate the wheel forces. The inputs to these equations are sinkage depth z_0, slip ratio s, and slip angle β. The following sections will show how these are derived from the rover's kinematic model and other known state variables. Then, how these forces are aligned with a wheel reaction frame, and interact with the dynamic model, will be explained.

3.2.1 Finding Sinkage Depth

The sinkage depth is defined as the distance from the lowest point on the

wheel (in the base frame), along the line towards the axle, to the surface. For this calculation, the wheel is assumed to be a flat disk of radius r. Points on the rim in the local wheel frame can be expressed as follows:

$$ {}^{w}p_{rim} = \begin{bmatrix} r\cos\gamma \\ 0 \\ r\sin\gamma \end{bmatrix} \tag{3.5} $$

γ specifies an angle about the wheel's y axis. The rim positions in the base frame are found by operating on ${}^{w}p_{rim}$ with the tranformation matrix between the wheel and inertial frame:

$$ {}^{0}p_{rim} = {}^{0}_{w}T\,{}^{w}p_{rim} \tag{3.6} $$

The rim's vertical position is the first element of ${}^{0}p_{rim}$. For a given rover pose (joint value set), there will be a γ_{min} which minimizes ${}^{0}p_{rim}(1)$, locating the lowest point on the wheel. To find this angle, take the expression for ${}^{0}p_{rim}(1)$

$$ {}^{0}p_{rim}(1) = {}^{0}_{w}T(1,:)\,{}^{w}p_{rim} = Ar\cos\gamma_{\min} + Cr\sin\gamma_{\min} + D \tag{3.7} $$

(A, B, C, and D are the elements of the first row of ${}^{0}_{w}T$), set its derivative equal to zero, and solve for γ_{min}. This results in the following:

$$ \gamma_{\min} = \tan^{-1}\left(\frac{C}{A}\right) \tag{3.8} $$

Now γ_{min} may be substituted into Equation 3.7 to find the lowest point on the wheel, ${}^{0}p_{low}$. Next, the unit vector pointing from ${}^{0}p_{low}$ to the wheel's axle (located at ${}^{0}p_{w}$) is needed:

$$ {}^{0}r_{line} = {}^{0}p_{w} - {}^{0}p_{low} \tag{3.9} $$

$$ {}^{0}\hat{r}_{slope} = \frac{{}^{0}r_{line}}{\left|{}^{0}r_{line}\right|} \tag{3.10} $$

The depth of a point on r_{line} with the origin at the surface can be expressed as

$$ depth = {}^{0}p_{low}(1) + z \cdot {}^{0}\hat{r}_{slope}(1) \tag{3.11} $$

Solving for when depth = 0 and z = z_0,

$$ z_0 = -\frac{{}^{0}p_{low}(1)}{{}^{0}\hat{r}_{slope}(1)} \tag{3.12} $$

To find the entry angle, which is an input to the wheel force equations, use

$$ \theta_1 = \cos^{-1}\left[1 - \frac{z_0}{r}\right] \tag{3.13} $$

3.2.2 Finding Slip Angle

As defined above, the slip angle is the angle between a wheel's velocity vector and its longitudinal axis (measured about its vertical axis). Analytically (see Equation 2.5), β is the inverse tangent of the ratio of lateral to longitudinal velocity. These velocity components must be expressed in a wheel frame, requiring the calculation of Jacobians. Given joint rates $\dot{q}$, Cartesian velocities can be solved for via ${}^{W}v_{W} = {}^{W}J\dot{q}$. ${}^{W}J$ is the translation Jacobian in the wheel frame. The method used to find the Jacobian was outlined in [Craig,05].

$$ \begin{aligned} {}^{N+1}J_{tran} = &\left[{}^{N+1}z_1 \times {}^{N+1}\left({}^{1}p_{N+1}\right), {}^{N+1}z_2 \times {}^{N+1}\left({}^{2}p_{N+1}\right) \ldots {}^{N+1}z_{N-1} \times {}^{N+1}\left({}^{N-1}p_{N+1}\right), {}^{N+1}z_N \times {}^{N+1}\left({}^{N}p_{N+1}\right)\right]\Lambda \\ &+\left[{}^{N+1}z_1, {}^{N+1}z_2 \ldots {}^{N+1}z_{N-1}, {}^{N+1}z_N\right](\mathrm{I} - \Lambda) \end{aligned} \tag{3.14} $$

Frame N is the link the wheel is on (6 or 7). N+1 is the wheel frame. The z's represent the frame's z axis described in the wheel frame. The p's are the position of

the wheel in each frame. I is the identity matrix, and $\Lambda = diag(\delta_j)$. δ_j= 1 for revolute joints, and δ_j= 0 for prismatic joints [Craig,05]. Notice the lowest frame referenced is frame 1. The z's can be calculated via

$$ {}^{N+1}\hat{z}_n = {}^{N+1}_{n}R\,{}^{n}\hat{z}_n = {}^{N+1}_{n}R\begin{bmatrix}0\\0\\1\end{bmatrix}, \text{ where } 1 \le n \le N \tag{3.15} $$

The p terms can be written as ${}^{N}_{n}R^{n}_{N}T\,{}^{N}p_{N+1}$. ${}^{N}p_{N+1}$ is the wheel position with respect to the link origin.

Using ${}^{W}J$, the wheel velocities in their own frame may be obtained. However, the wheel frames do not change orientation with respect to the rest of the segment if the rover's orientation with respect to the ground changes. For example, if the rover is pitched forward, then the longitudinal axis will point slightly down into the ground, not parallel to it as is expected by the terramechanics equations. A frame is needed whose longitudinal direction remains parallel to the surface of the ground, and whose vertical axis passes through the wheel's lowest point and axle. This is important for accurately determining β, and will be used later when aligning soil forces with the vehicle.

This new coordinate frame is called the wheel reaction (wr) frame. The vertical axis of this new frame points in the same direction as ${}^{0}\hat{r}_{slope}$. The lateral axes for the wheel and wr frames point in the same direction. The wr x-axis is then orthogonal to the other two, which is parallel to the ground's surface. This is the case because the situation can be described as a planar disk (wheel) intersecting a plane

(ground). If the disk's z-axis crosses the its lowest point and center, and its y-axis is oriented perpendicular to the face of the disk, then its longitudinal axis must be pointed parallel to the plane it is intersecting. To build this frame, ${}^{link}_{wr}R$ is needed, which operates on coordinates in the wr frame and expresses them in the local link frame.

$$ {}^{link}_{wr}R = {}^{link}_{0}R\begin{bmatrix} {}^{0}X_{wr} & {}^{0}Y_{wr} & {}^{0}Z_{wr} \end{bmatrix} \tag{3.16} $$

${}^{0}X_{wr}$, ${}^{0}Y_{wr}$, and ${}^{0}Z_{wr}$ are the primary axes of the wr frame expressed in base frame coordinates. These can be calculated as follows:

$$ {}^{0}Y_{wr} = {}^{0}_{link}R\begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \tag{3.17} $$

$$ {}^{0}Z_{wr} = {}^{0}\hat{r}_{slope} \tag{3.18} $$

$$ {}^{0}X_{wr} = \frac{{}^{0}Y_{wr} \times {}^{0}Z_{wr}}{\left| {}^{0}Y_{wr} \times {}^{0}Z_{wr} \right|} \tag{3.19} $$

To calculate velocities of the wheel bub and β in the wheel reaction frame, use

$$ {}^{wr}J = {}^{wr}_{link}R\,{}^{W}J \tag{3.20} $$

$$ {}^{wr}v_{W} = {}^{wr}J\dot{q} \tag{3.21} $$

$$ \beta = \tan^{-1}\left(\frac{{}^{wr}v_{W}(2)}{{}^{wr}v_{W}(1)} \right) \tag{3.22} $$

3.2.3 Finding Slip Ratio

The slip ratio s may be solved for numerically using the torque equation, Equation 2.18. T can be considered as the torque commanded to a rover's wheel,

which is a known (or derivable) quantity. Entry angle, the other input to the torque equation is known by the simulation through the process outlined in Section 3.2.1. The only remaining unknown is s. Since torque generally increases with slip ratio, the bisection search method is used, and yields reliable results.

3.2.4 Wheel Force Vector Construction

Once a value for s is obtained, the remaining force magnitudes may be easily calculated with Equations 2.15-2.17, and aligned with the wr frame. However, while the force equations yield numerical results for any vehicle state, additional steps are needed to ensure their proper application and direction. Equation 2.15 describes forces along the longitudinal axis of the wheel reaction frame. Drawbar pull F_x has two terms. The first (dependent on shear stress) is soil thrust H, and the second (dependent on radial stress) is rolling resistance R.

$$H = \frac{rb}{\theta_m(\theta_1 - \theta_m)}\left[\tau_{xm}(\theta_1 \cos\theta_m - \theta_m \cos\theta_1 - \theta_1 + \theta_m)\right] \tag{3.23}$$

$$R = \frac{rb}{\theta_m(\theta_1 - \theta_m)}\left[-\sigma_m(\theta_1 \sin\theta_m - \theta_m \sin\theta_1)\right] \tag{3.24}$$

The soil thrust term can evaluate positive or negative depending on the applied torque and wheel sinkage. The rolling resistance term always evaluates negative. For a wheel with zero longitudinal velocity ($^{wr}v_x = 0$), soil thrust must overcome rolling resistance ($|H| > |R|$) for F_x to be nonzero. In that case,

$$F_x = H + sign(H) \cdot R \tag{3.25}$$

Otherwise, $F_x = 0$. For nonzero $^{wr}v_x$, $|R|$ is applied opposite the direction of motion, while H's sign is maintained. For this case,

$$F_x = H + sign({}^{wr}v_x) \cdot R \tag{3.26}$$

Likewise, for lateral forces, F_y acts in the direction opposite ${}^{wr}v_y$, and only when ${}^{wr}v_y$ is nonzero. Vertical forces are always on, however an additional damping term was added for this research that approximates a suspension system.

Now the forces may be aligned with the wheel reaction frame as follows:

$${}^{wr}F = \begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} \tag{3.27}$$

3.2.5 Forces and Moments on Link CG

For the dynamics calculations in the next section, it will be convenient to combine each segment's wheel forces into a single force and single moment at that segment's cg.

$${}^{link}F_C = \sum {}^{link}F_i \tag{3.28}$$

$${}^{link}N_C = \sum {}^{link}p_i \times {}^{link}F_i \tag{3.29}$$

F_i are the individual wheel forces in the link frame. ${}^{link}p_i$ is the position vector from each wheel to its segment's cg in link frame coordinates.

3.3 Rover Dynamic Model

3.3.1 Modified Newton-Euler Method

The rover dynamic equations were derived using a modified form of the iterative Newton-Euler dynamic formulation shown in [Craig,05]. In the standard approach, the forces on each segment CG due to the link's accelerative, coriolis, and centrifugal motions. The modification is introduced at this step by including the

combined wheel forces and moments on the segment cg in Equations 3.28 and 3.29.

$$^{i+1}F_{i+1} = m_{i+1}{}^{i+1}\dot{v}_{C_{i+1}} + {}^{i+1}F_{C_w} \tag{3.30}$$

$$^{i+1}N_{i+1} = {}^{C_{i+1}}I_{i+1}{}^{i+1}\dot{\omega}_{i+1} + {}^{i+1}\omega_{i+1} \times {}^{C_{i+1}}I_{i+1}{}^{i+1}\omega_{i+1} + {}^{i+1}N_{C_w} \tag{3.31}$$

3.3.2 Equation of Motion

After obtaining the joint torque equations, similar terms can be collected, and a configuration-space equation may be formed:

$$\tau = M(q)\ddot{q} + B(q)[\dot{q}\dot{q}] + C(q)[\dot{q}^2] + G(q) + F(q)[F_w] + N(q)[N_w] \tag{3.32}$$

τ is the vector of torques applied at each joint. For the passively articulated rover case, this is a zero vector. M is the (7 x 7) mass matrix. B is a (7 x 6) matrix of coriolis terms. While there are many combinations of joint velocity products, only six of these have coefficients that evaluate to nonzero values. C is a (7 x 4) matrix of centrifugal terms (4 columns for the 4 nonzero coefficients). G is a vector of gravity terms. F and N are (7 x 6) and are multiplied by the stacked wheel forces and torques on the link cg in the link frame. With $\tau = 0$, the joint accelerations at each time step can be solved for by using

$$\ddot{q} = M^{-1}(q)\left[-B(q)[\dot{q}\dot{q}] - C(q)[\dot{q}^2] - G(q) - F(q)[F_w] - N(q)[N_w]\right] \tag{3.33}$$

See Appendix A for the dyamic equation coefficients.

Chapter 4: Simulation Design

This chapter describes the simulation's design and testing. Section 4.1 walks through the simulation loop. Section 4.2 compares the simulation results with results from previous research.

4.1 Simulation Outline

Now that the kinematic, dynamic, and terrain interaction models are defined, they can be used as components of a computer program to simulate the motion of a segmented-body, rigid-wheeled rover through deformable terrain. This section describes how the simulation developed for this thesis is constructed.

First, soil properties, wheel and rover dimensions, masses, and inertias are loaded into Matlab. Next, initial conditions are set. These include the initial position, orientation, and body joint values and their rates. All zero values describe a static rover facing forwards, and sitting upright, with a straightened turning joint. The other initial conditions to be set are the torques applied by the motors to each of the wheels.

As shown in the closed-form applied torque equation, it is solely a function of slip ratio and sinkage. Each wheel's sinkage can be found by following the process described in Section 3.2.1. Using the bisection method, the torque equation is solved for slip ratio. Having obtained wheel sinkage and slip, the magnitudes of the other non-lateral forces (D and W) may now be calculated.

To calculate C, the wheel's slip angle must be known. Slip angle is a function of the ratio of lateral and longitudinal velocity components, so the wheel's velocity in

its own frame must be determined. Following the method in [Craig,05], the Jacobian for each wheel is obtained. Operating on the joint rate vector with these Jacobians yields the desired velocity vector for each wheel, from which β is easily calculated. Finally, side force C is calculated for each wheel.

Now that all of the wheel forces are known, the next step is to compute their combined force and moment on the local link center of gravity (Equations 3.30 and 3.31). Once these forces and moments are known, they can be used along with the vehicle joint values and rates in Equation 3.33 to find the joint accelerations. The final step is to update the joint values and rates via Euler integration.

$$q(t+\Delta t) = q(t) + \dot{q}(t)\Delta t + \tfrac{1}{2}\ddot{q}(t)\Delta t^2 \tag{4.1}$$

$$\dot{q}(t+\Delta t) = \dot{q}(t) + \ddot{q}(t)\Delta t \tag{4.2}$$

Also at this step—how often depends on bandwidth selection—new wheel powers can be commanded according to a control law. For the simulations performed in this research, each timestep was 0.001 seconds, and the controller bandwidth (when present) was 10 Hz.

4.2 Simulation Testing and Validation

Before using the simulation to gain insight into the dynamics of passively articulated rover systems, it must be validated against previous work and common sense in order to be accepted. Since part of the soil model is new, its outputs should be compared with those of other soil models. The model must also be consistent with itself. For example, if steady state conditions and wheel torques are solved for, those conditions should be maintained throughout a simulation. This would indicate

that the dynamic and soil interaction models are stable together. All validation tests were conducted with a fixed central joint.

4.2.1 Comparison to Bekker Model

First, the soil model was compared to Bekker's equations (2.2-2.3). Bekker's equations assume that rolling resistance (R_c) and wheel sinkage (z_0) are related to the wheel diameter (D), width (b), and weight on the wheel (W) through a power law, such that

$$Q = Ax^B \tag{4.3}$$

where Q is the quantity being calculated, x is the variable, and A and B are the power law parameters. A suitable range of variables were chosen, and three cases were tested to find the R_c and z_0 dependencies of both Bekker's and the author's models. The results are summarized in Tables 4.1 and 4.2. Figure 4.1 shows typical relationships between rolling resistance, sinkage, and wheel diameter.

Table 4.1: Rolling Resistance vs. Wheel diameter, Width, and Weight

x →		D		b		W	
A	Bekker	131.6	24%	34.41	27%	0.01475	35%
	Simulation	162.7		43.74		0.01988	
B	Bekker	-0.6563	4.0%	-0.3115	4.0%	1.312	0.8%
	Simulation	-0.6298		-0.2989		1.301	

Table 4.2: Wheel Sinkage vs. Wheel Diameter, Width, and Weight

x →		D		b		W	
A	Bekker	0.03191	21%	0.008723	23%	0.0003846	27%
	Simulation	0.03874		0.01075		0.0004874	
B	Bekker	-0.3125	4.7%	-0.6229	1.1%	0.625	0.9%
	Simulation	-0.2978		-0.6161		0.6192	

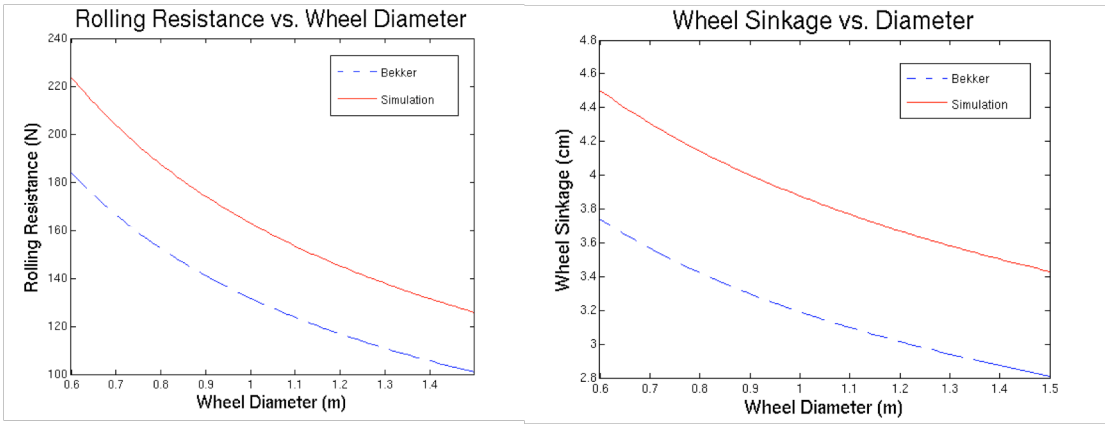

Figure 4.1: Rolling Resistance and Sinkage vs. Wheel Diameter

It can be seen from the tables that the exponent terms match fairly well (to within a few percent), so that the general trends of Q vs. x are similar. However, the A coefficient from the simulation differs from Bekker's equations by 20-35%. Typically, this indicates that the simulated rolling resistances and sinkages are greater than that which would be calculated using Bekker's equations. Wong's model takes into account forces generated by the wheel's tangential stress whereas Bekker's does not, but removing these extra terms from the equations yielded negligible changes.

Next, dependency of R_c and z_0 on soil parameters n and k_ϕ was checked (k_c always appears in conjunction with k_ϕ and for low cohesion soils is much smaller).

Table 4.3: Rolling Resistance vs. Soil Parameters

x →		n		k_ϕ	
A	Bekker	182.8	25%	17470	39%
	Simulation	228.4		10630	
B	Bekker	1.331	18%	-0.3113	16%
	Simulation	1.09		-0.2617	

Table 4.4: Wheel Sinkage vs. Soil Parameters

x →		n		k_ϕ	
A	Bekker	0.03075	26%	280.9	17%
	Simulation	0.03869		231.8	
B	Bekker	2.638	14%	-0.6226	4.3%
	Simulation	2.275		-0.5957	

These results show that the simulation's dependence on soil parameters differs significantly from Bekker's model, and are probably the cause of the 25% offset in sinkage and rolling resistances. For this study, overall trends related to vehicle weight and dimensions are more important than exacting numeric values, so the simulation will be sufficient.

4.2.2 Comparison to Yoshida Model

Next, the simulation was compared against the [Ishigami,07] results for turning and drawbar pull at various slip ratios and slip angles on a single wheel. The wheel has a radius of 9 cm, a width of 11 cm, and a mass of 6.6 kg. Their experimental measurements are plotted over their model predictions in Figures 4.2 and 4.3. Figures 4.4 and 4.5 show the results from the author's simulation.

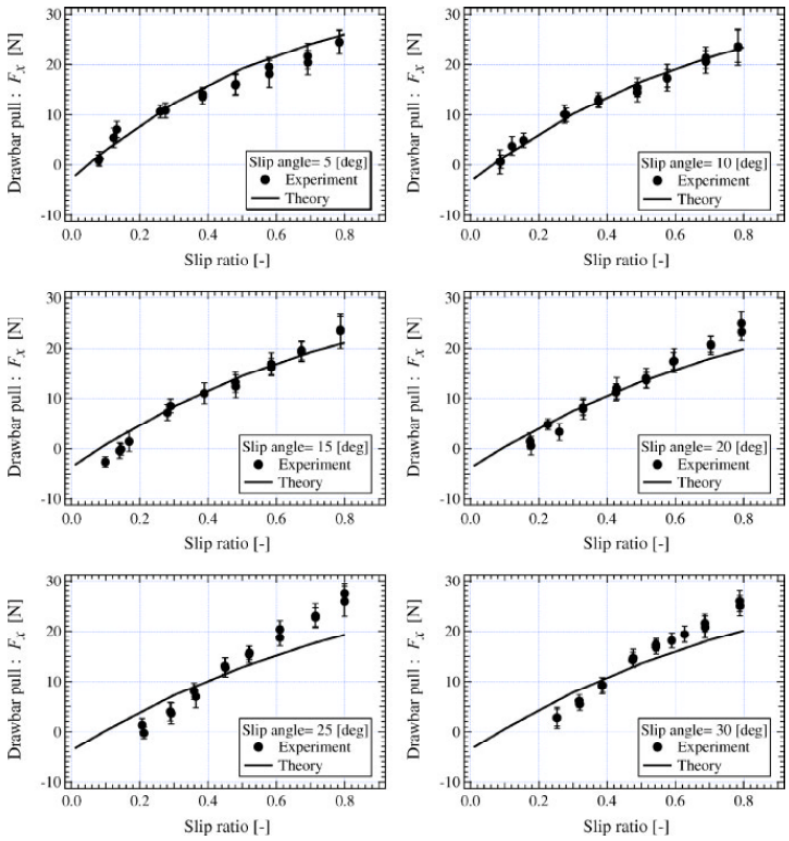

Figure 4.2: Yoshida drawbar pull vs. slip ratio for different slip angles

At first glance, the trends are similar. Drawbar pull increases with increasing slip ratio, and decreases with increasing sideslip. Side forces decrease with increasing slip ratio, and increase with increasing sideslip. For low slip ratios and sideslip, the author's side force results are within 10-15% of Yoshida's. However, the drawbar pull magnitudes of the simulation plots differ significantly from Yoshida's: by ~ 14 N with s = 0, and ~ 25 N when s = 0.8, nearly independent of sideslip. This is due to Yoshida's inclusion of rut recovery in their wheel/soil interaction model. Unlike the assumption in this thesis that the wheel stops contacting the soil behind bottom-dead-center ($\theta_2 = 0$), Yoshida assumes a nonzero θ_2 defined by

$$\theta_2 = \cos^{-1}\left(1 - \lambda z_0 / r\right) \tag{4.4}$$

where λ is called the wheel sinkage ratio, denoting the ratio between the front and rear sinkages of the wheel. Rut recovery depends on many variables and is not well understood. For Yoshida's calculations, $0.9 < \lambda < 1.1$ [Ishigami07]. This means that $\theta_2 \approx \theta_1$, which significantly stretches out the stress distributions discussed in section 2.1.4. This stretching effect compared to the approximated distribution is shown in Figure 4.6.

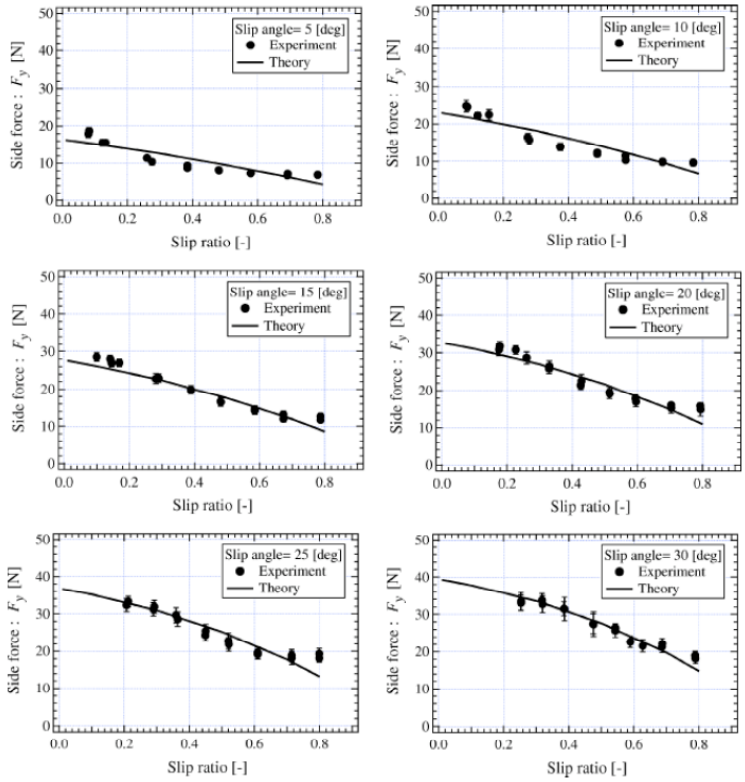

Figure 4.3: Yoshida Side Forces vs. slip ratio for different slip angles

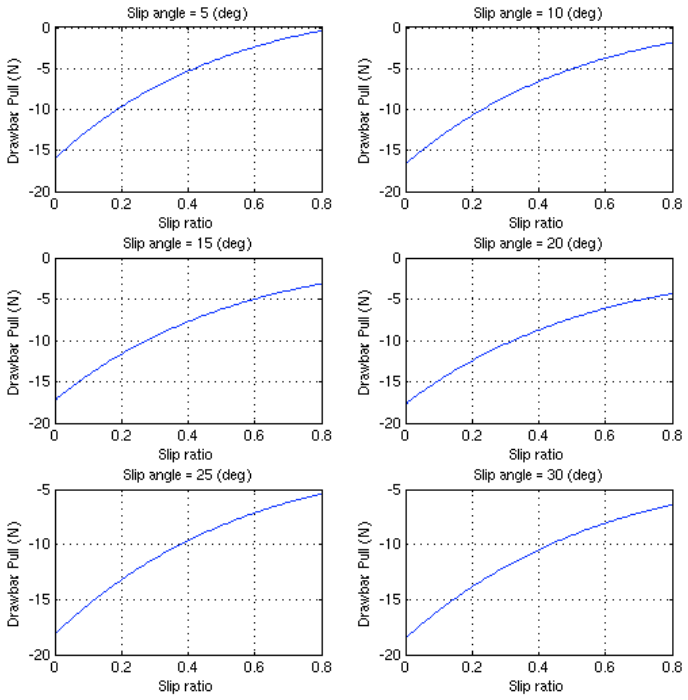

Figure 4.4: Simulation Model Drawbar Pull vs. slip ratio for different slip angles

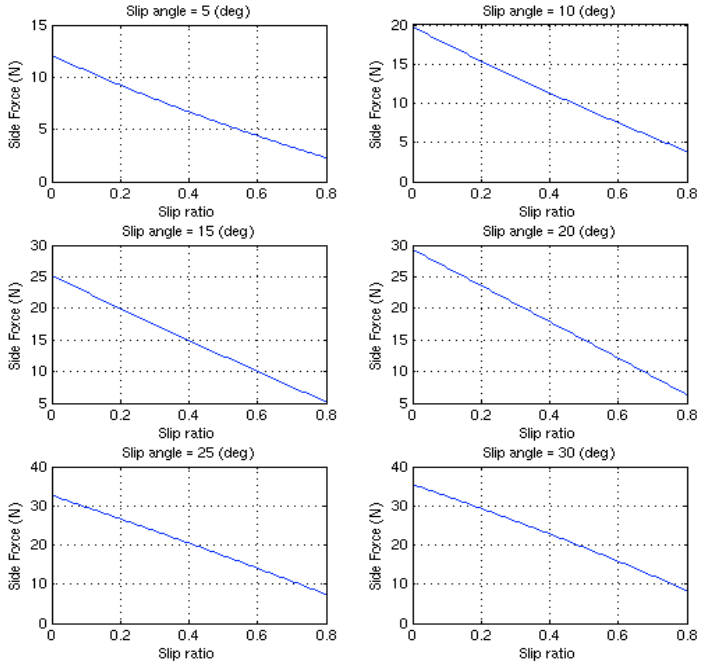

Figure 4.5: Simulation Model Side Forces vs. slip ratio for different slip angles

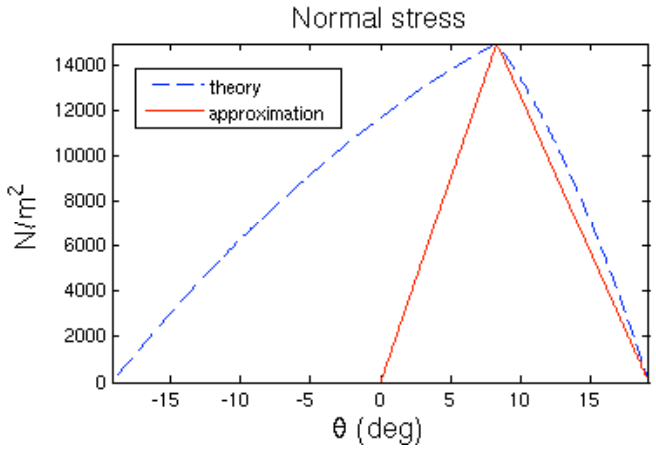

Figure 4.6: Approximated vs. Exact Stress Distribution for non-zero θ_2

For Yoshida's experiments with a small rover on lunar simulant soil, the rut recovery method worked well. To look for other regimes where rut recovery has significant effects, two other rovers were considered. The first was a four-wheeled rocker-suspension rover at the Space Systems Laboratory (Figure 4.7) with wheels 13.3 cm in diameter and 8.9 cm wide. The rover's weight is 8.5 kg, yielding an average load of 20.8 N on each wheel.

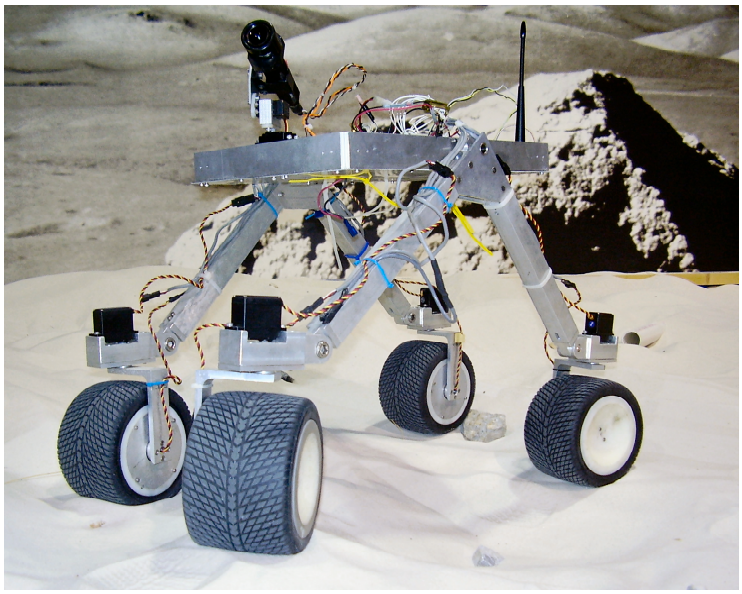

Figure 4.7: Rover at Space Systems Laboratory

Using the model developed for the simulation, for zero drawbar pull, sinkage in loose dry sand is 1.31 cm, slip ratio is 0.37, and rolling resistance is 5.88 N. During testing, the rover reached a velocity of 20 cm/s, drawing a power of 1.55 W per wheel. No significant rut recovery was noticed. Assuming a motor efficiency of 0.8, the power available to each wheel was 1.24 W, indicating a rolling resistance of 6.2 N. The rolling resistance calculated by the simulation differs from this value by only 5.2%. While this test was crude, the results are encouraging.

The second rover considered was the LRV used during the Apollo missions to the moon (Figure 4.8). The LRV had wheels 0.41 m in radius, 0.23 m in width, and had a mass of 700 kg. According to the author's model, the LRV would attain a drawbar pull of zero when s = 0.07. Sinkage would be 2.5 cm, and rolling resistance would be 46.2 N per wheel. The rover was provided with 0.25 hp (186 W) drive motors at each wheel. This would enable a top speed of about 4 m/s, or 14.4 kph, which is near the rover's maximum design velocity on flat ground. For this case also, rut recovery does not appear to have had a large effect on vehicle performance.

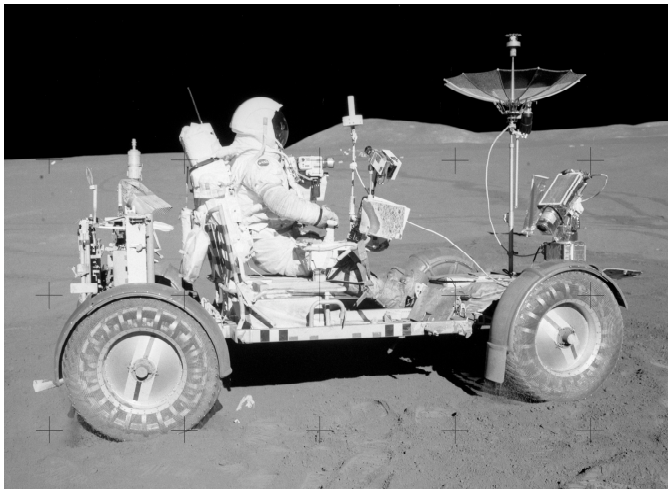

Figure 4.8: Lunar Roving Vehicle (NASA)

Since the general trends of the simulation follow previous work done, and produce results numerically similar to experimental data where rut recovery was not a major factor, it will be considered sufficient for the investigations that follow.

4.2.3 Constant Velocity Driving

Given a set of steady state conditions, the simulation should maintain those conditions. For straight, constant velocity driving, the net force in the longitudinal direction (drawbar pull) must be equal to zero. To test this, three two-segment rovers were specified, and their steady state driving torques were numerically solved for. These torques were then used as initial conditions for the simulation, which was run for 5 seconds. The rover parameters were as follows:

Table 4.5: Test Rover Configuration Parameters

Rover	Segment Mass (kg)	Wheel Radius (m)	Wheel Width (m)
A	200	0.5	0.2
B	100	0.4	0.15
C	25	0.15	0.08

The steady-state parameters used as initial conditions for each rover wheel are shown in the table below:

Table 4.6: Steady State Test Values

Rover	Torque (Nm)	Velocity (m/s)	Slip Ratio	Sinkage (cm)	Rolling Resistance (N)
A	88.86	3.0	0.00258	4.445	174.1
B	36.36	1.0	0.0121	3.703	89.0
C	5.222	0.5	0.0753	3.105	33.1

At the end of the simulation, the rover's state was virtually the same as its initial conditions, indicating that accelerations were near zero, and the wheel/soil interaction and rover dynamic models interacted in a stable way.

4.2.4 Constant Drawbar Pull

Next, the simulation was tested with constant accelerations. This was done for Rover A and solving for the required torques to 1) accelerate at 0.1 m/s^2 and 2) overcome the rolling resistance of an additional, unpowered segment. This translates to per-wheel drawbar pulls of 10 N and 175 N, respectively.

Table 4.7: Expected Acceleration Test Values

Test #	Drawbar Pull (N)	Torque (Nm)	Sim Time (s)	Initial Velocity (m/s)	Expected Final Velocity (m/s)	Slip Ratio	z_0 (cm)	R_c (N)
1	10	93.83	10	0.0	1.0	0.00564	4.445	173.8
2	175	170.4	5	1.0	8.75	0.0708	4.401	176.4

The results of the final iteration of the simulation and deviation from the expected values are shown in Table 4.8.

Table 4.8: Simulated Acceleration Test Values

Test #	Drawbar Pull (N)		Velocity (m/s)		Slip Ratio		Sinkage (cm)		Rolling Resistance (N)	
1	10.80	8.0%	1.0002	0.02%	0.00598	6.0%	4.4356	2.6%	173.1	0.45%
2 (F)	175.8	0.5%	9.1456	4.5%	0.0785	11%	4.2486	3.5%	158.3	10%
2 (B)	150.4	14%			0.0535	24%	4.5560	3.5%	183.3	3.9%

For the low drawbar pull case, the expected final velocity was reached nearly exactly. For the high drawbar pull case, a higher velocity is obtained than expected. Drawbar pull, slip, sinkage, and rolling resistances also differ by nontrivial amounts. This is due to the fact that the steady state torques were solved for under the assumption that all wheels were at the same depth. Due to the vehicle's forward acceleration, weight is transferred to the rear, causing the back wheels to sink deeper than the front wheels. A steady state 0.07 degree pitch-up was the result. This effect was not accounted for when calculating the torques, and will be attributed the deviations.

Chapter 5: Articulated Rover Simulation

This chapter details an investigation into passively articulated rover motion using the simulation described above. Unlike the validation tests, the turning joint is free to rotate in these simulations. Section 5.1 looks at forward movement, and its stability when disturbed by an external impulse. Section 5.2 describes the turning nature of an articulated vehicle. Section 5.3 implements a simple control system to deal with disturbances and to help complete desired vehicle motions. All simulations occur under Earth gravity conditions and in uniform dry sand.

5.1 Forward Movement

5.1.1 Nominal Case

The first articulated simulation runs were targeted at determining whether the rover could move in a straight line. In a perfect simulation world, if the rover is moving at constant velocity and provides equal and sufficient powers to its wheels, then it will continue without changing its velocity vector, or exhibiting turning joint motion. Three simulations were performed with initial velocities of 0.5 m/s, 3 m/s, and 6 m/s for 60 seconds each. At the end of each 60 second simulation, turning joint angles were on the order of 10^{-9} degrees, the vehicle's heading had shifted by 10^{-8} degrees, and the velocities were the same to about one part in one million.

5.1.2 Disturbed Behavior

Traveling over a planet's surface, a rover will encounter non-homogenous soil

and rock distributions. Rocks impacting the wheel sides have the ability to disturb the rover's motion, and impart angular velocity on its turning joint. Disturbances were introduced into the simulation as a non-zero initial turning joint rate. The rover was tested over rates ranging from 0 deg/s to one such that its lateral velocity matched its longitudinal velocity. This range was chosen to represent probable disturbance magnitudes encountered in the field. The rover's responses to these disturbances changed as initial velocity was increased. For each disturbance, the maximum turning angle, settled turning angle, final heading change rate, and settling time were recorded.

For the low speed case, the turning joint approached its settled value without overshoot (Figure 5.1), and the relationship between maximum turning joint angle and disturbance magnitude was nearly linear (Figure 5.2).

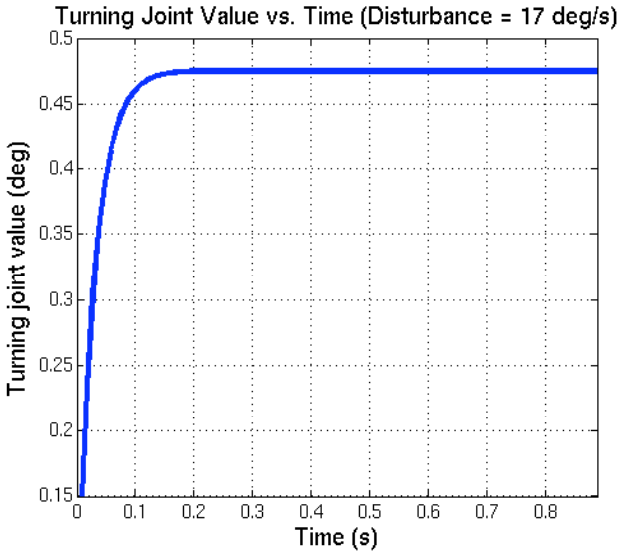

Figure 5.1: Low Speed Disturbance, Turning Joint Angle vs. Time

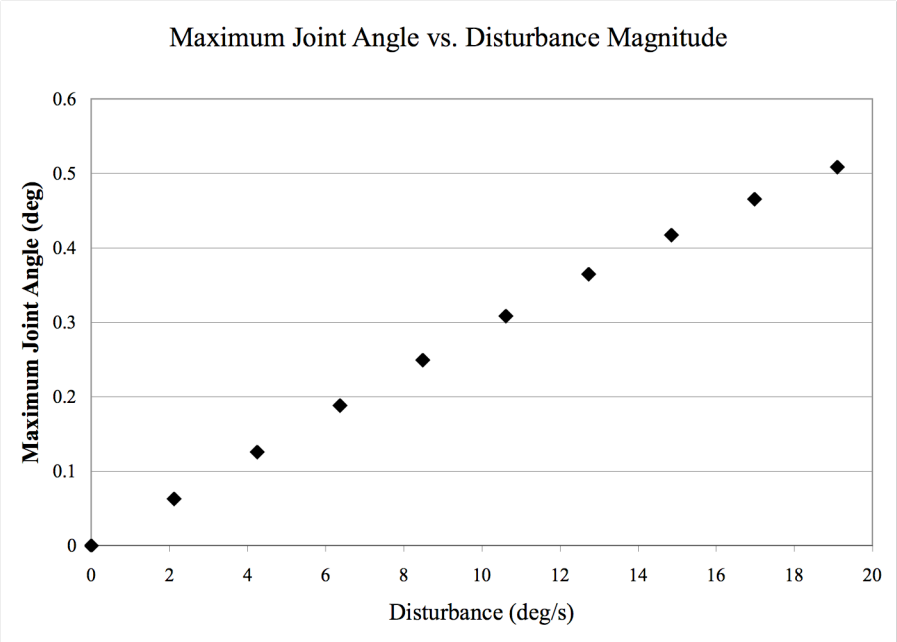

Figure 5.2: Low Speed Disturbance, Maximum Turning Joint Angles

For the medium speed case, there is some overshoot (Figure 5.3). Maximum vs. final turning joint angles differ by up to 6.3 percent, diverging with increasing disturbance. Figure 5.4 shows that the maximum disturbance plot begins to turn for higher magnitude disturbances.

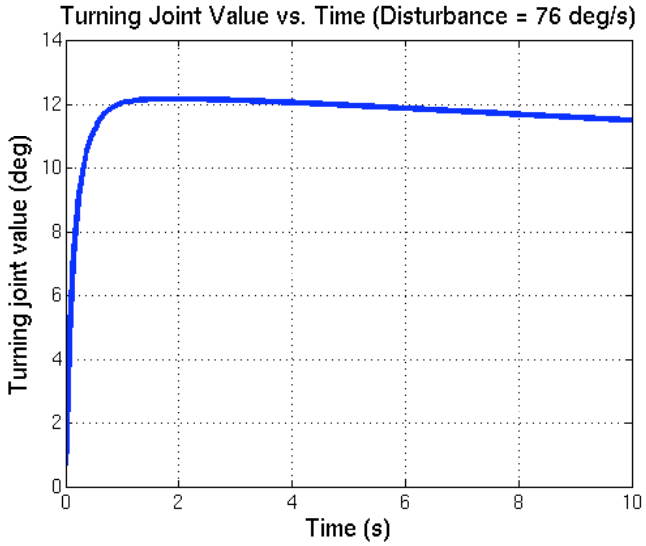

Figure 5.3: Medium Speed Disturbanc, Turning Joint Angle vs. Time

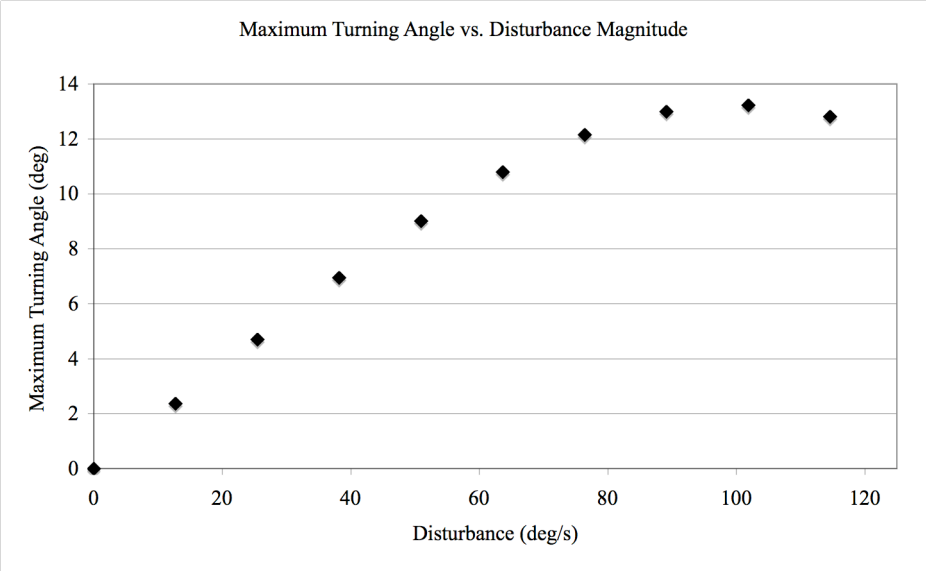

Figure 5.4: Medium Speed Disturbance, Maximum Turning Joint Angles

For the high speed case, the maximum turning angle plot continues its turn, but eventually levels out at a maximum disturbance of about 17 degrees, independent of the initial displacement (Figure 5.5). The difference between maximum and final turning joint angles also diverged by significant amounts for the larger disturbances.

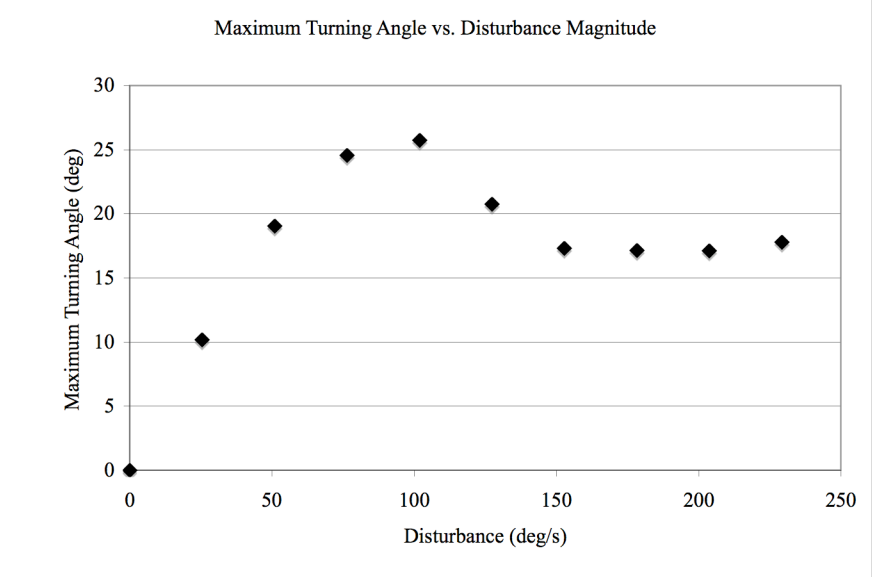

Figure 5.5: High Speed Disturbance, Maximum Turning Joint Angles

Essentially, each disturbance introduces a steady state turning joint, or q_7 displacement, and therefore, turning radius to the rover's motion. For a given q_7, the rover's segments line up with a circle of radius

$$r = \frac{\ell}{\tan(q_7/2)} \tag{5.1}$$

The observed vs. expected turning radii differed only by an average of 3.8%. This difference could be due in part to the same powers being delivered to each wheel, whereas inner and outer wheel powers would be slightly different for a perfectly turning vehicle. Figure 5.6 shows the circular path produced for one of the high speed simulations.

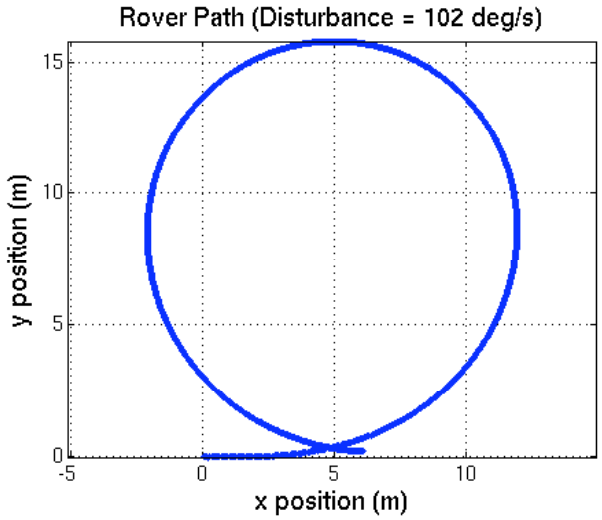

Figure 5.6: High Speed Disturbed Rover Path

5.2 PD Control

In order to truly be in control of an articulated rover and to handle disturbances, there must be a means of steering. The steering system should be as

simple as possible, and rely on feedback easily available from the rover. It is assumed that the rover has a high quality absolute rotary encoder at its turning joint to measure angles, and to derive joint rates.

5.3.1 Controller Design

The controller developed for this test was a PD controller. The turning joint value and rate are multiplied by gains, and added to the steady-state power delivered to the front left wheel, and subtracted from the front left one.

$$P_{left} = P_{nom} + K_p\left(q_7 - q_{7des}\right) + K_d\left(\dot{q}_7 - \dot{q}_{7des}\right) \tag{5.2}$$

$$P_{right} = P_{nom} - K_p\left(q_7 - q_{7des}\right) - K_d\left(\dot{q}_7 - \dot{q}_{7des}\right) \tag{5.3}$$

The initial gains were chosen arbitrarily such that $K_p = 100$, and $K_d = 100$. The controller may command new powers at a frequency of 10 Hz.

5.3.2 Response to Disturbances

The first test for the control system was to see how well it handled the same disturbances described in Section 5.1.2. The simulations were stopped when the turning joint angle and rate were negligible. Results in Figures 5.7-5.9 show the maximum q_7 overshoot and final vehicle heading q_6 as a function of disturbance magnitude, for each of the velocities tested.

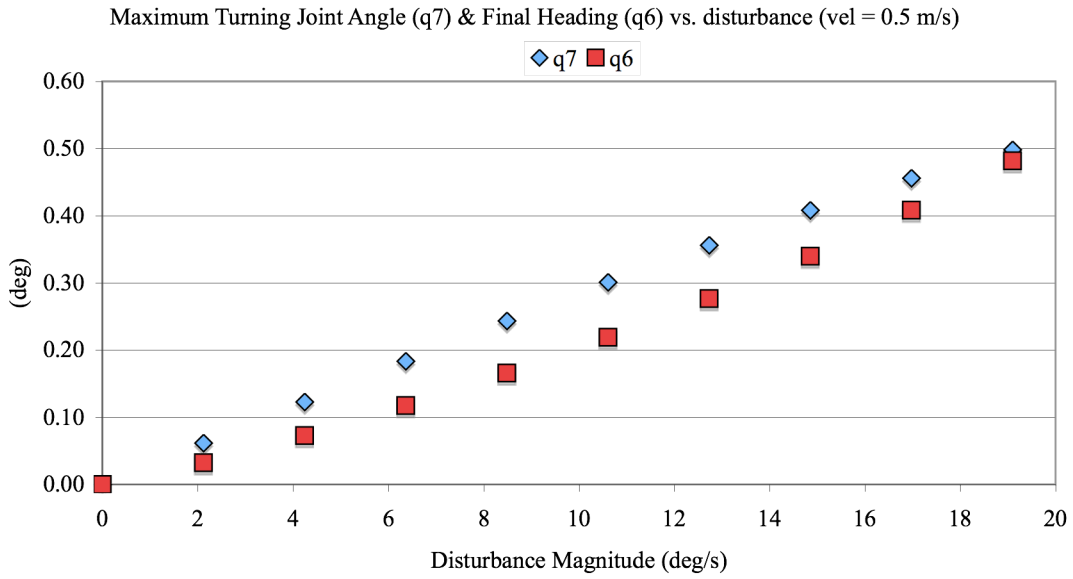

Figure 5.7: Disturbance Effects for v = 0.5 m/s

For low and medium velocities, the general trend is for higher disturbances to cause larger overshoots and heading changes. Also, for the given set of gains, the control system was less able to handle the higher speed cases. Additionally, power requirements per motor increased by up to 2% and 19% for the 0.5 m/s and 3 m/s velocities, respectively, while for the 6 m/s case, power requirements increased by up to 15%.

Settling times were on the order of 9 seconds for v = 0.5 m/s, 17 seconds for v = 3 m/s, and 18 seconds for v = 6 m/s. While maximum turning joint angles increase with increasing disturbance magnitude for the high velocity case, final heading increases until about 100 deg/s disturbance, and then decreases back to near zero at 230 deg/s. A typical trace of turning joint angle vs. time (for the high speed case) is shown in figure 5.10.

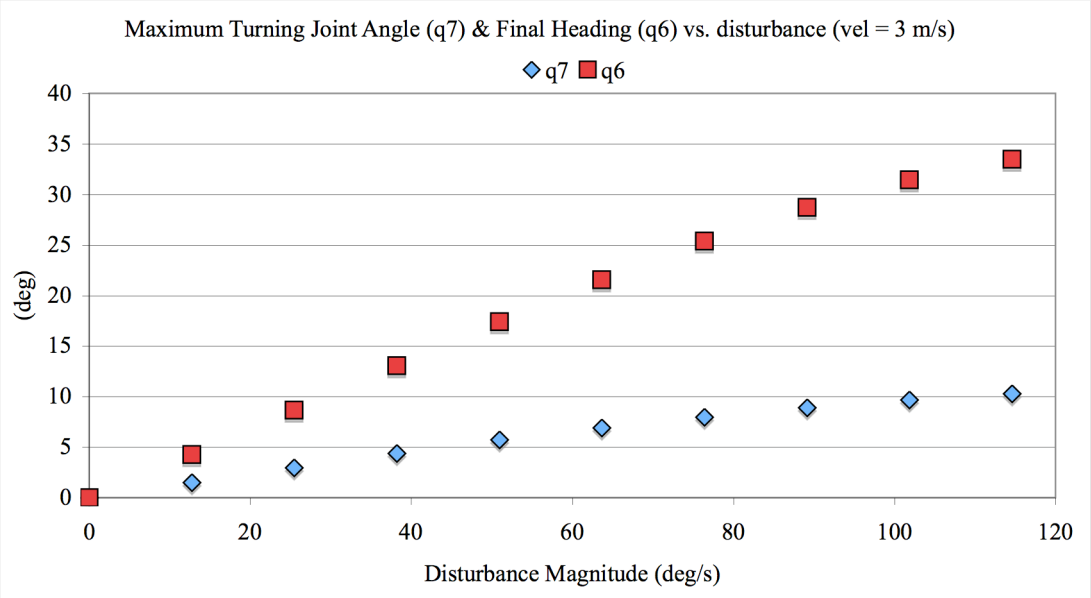

Figure 5.8: Disturbance Effects for v = 3 m/s

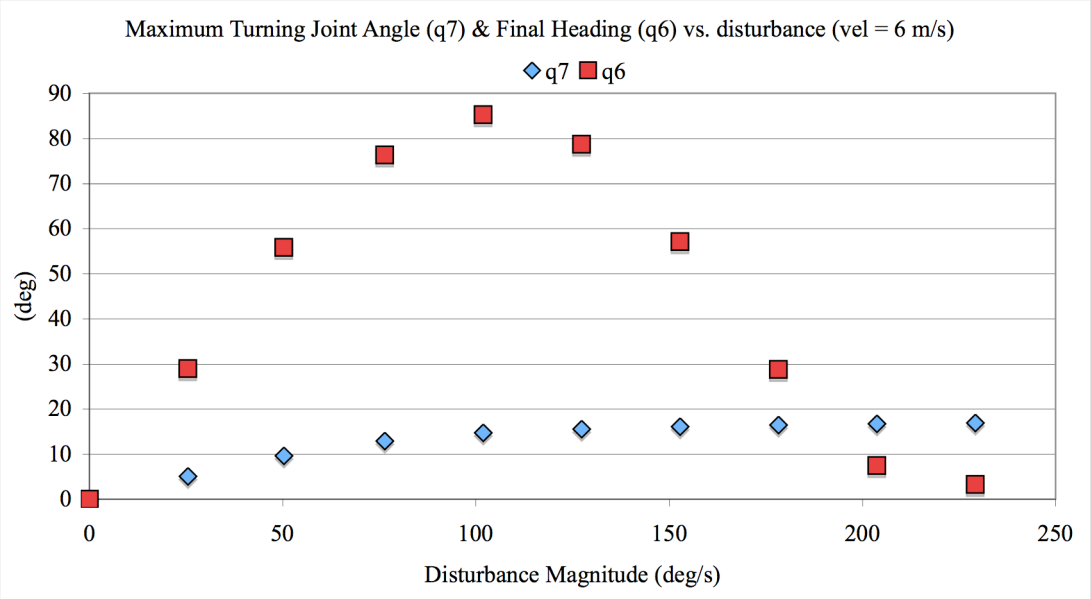

Figure 5.9: Disturbance Effects for v = 6 m/s

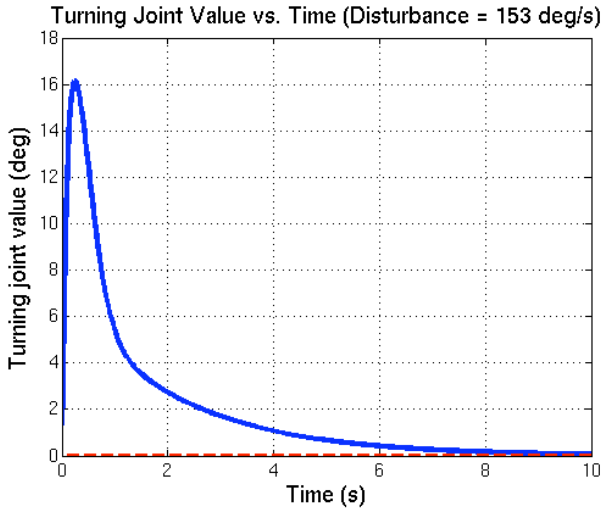

Figure 5.10: Response of Turning Joint with PD Control at High Speeds

Next an attempt was made to achieve a quicker settling time by altering the gains of the system for the 6 m/s case. Now, $K_p = 300$ and $K_d = 300$. Figure 5.11 shows the settling time and power requirement percent differences between the two gain sets.

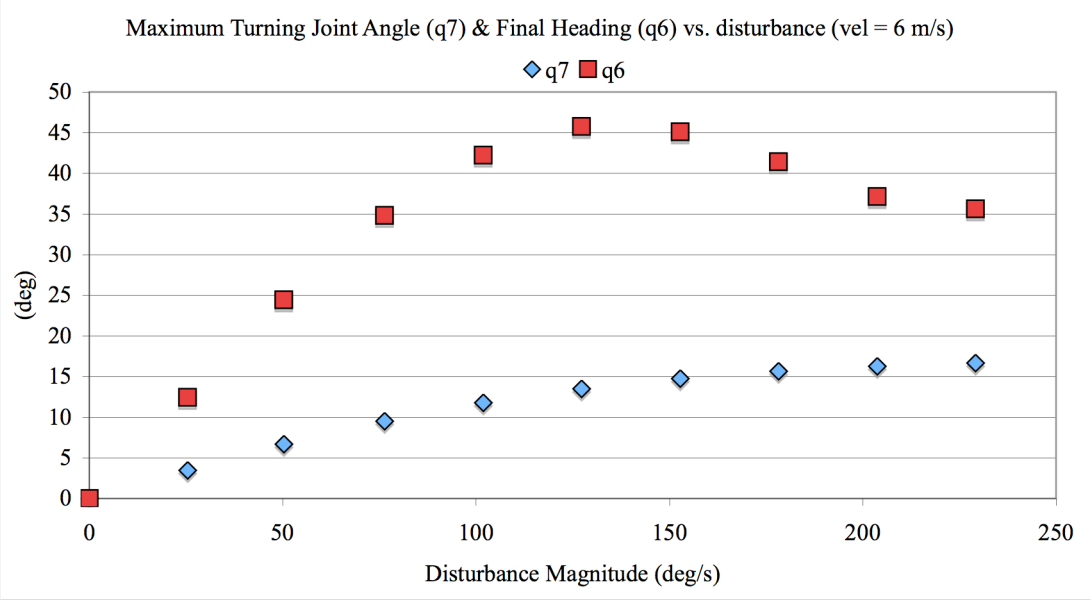

Figure 5.11: Settling Time and Power Requirement Comparisons

The new gains improve settling time by 25-55 percent across the disturbance

range. Power requirements increased by a maximum of 25 percent. The turning joint response is shown in Figure 5.12

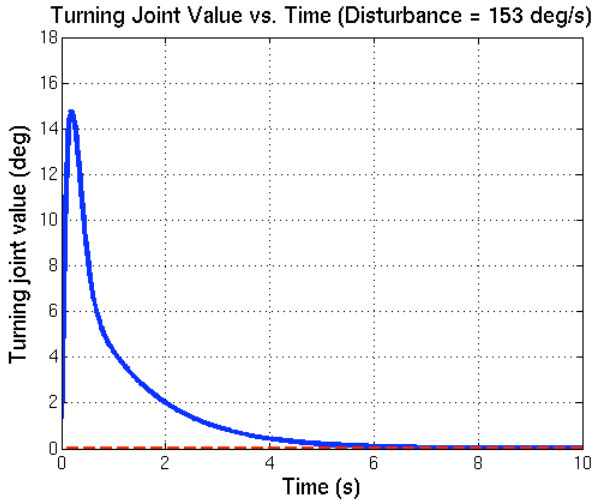

Figure 5.12: High Speed Turning Joint Response with Alternate Gains

5.3.3 Turning Performance

The next group of simulations set the rover moving at 0.5 or 3 m/s (nominal per-wheel driving powers of 89 W and 535 W, respectively) in the longitudinal direction, and then told the control system to move the turning joint angle to 5, 10, 15, 30, or 45 degrees. Tables 5.1 and 5.2 show each case's settled turning joint angle percentage, resulting turn radius, and maximum power per wheel.

Table 5.1: Turning Performance at 0.5 m/s

q_7 commanded (deg)	q_7 final (deg)	q_7 % difference	Turn Radius (m)	Maximum Power (W)
5	4.999	0.00	34.7	98
10	9.9994	0.01	17.3	107
15	14.9979	0.01	11.5	115
30	29.9823	0.06	5.7	141
45	44.931	0.15	3.7	168

Table 5.2: Turning Performance at 3 m/s

q_7 commanded (deg)	q_7 final (deg)	q_7 % difference	Turn Radius (m)	Maximum Power (W)
5	4.9881	0.24	35.2	543
10	9.9058	0.94	17.7	552
15	14.6882	2.08	11.9	561
30	27.7773	7.41	6.3	587
45	38.6856	14.03	4.4	613

Traveling velocities remained within 5% of their initial values. The PD controller had an easier time achieving the desired joint angles for the smaller velocity case than for the higher one (See Figures 5.13 and 5.14). Powers peaked during initial phases of disturbance handling, but then returned to within a few Watts of nominal steady-state powers.

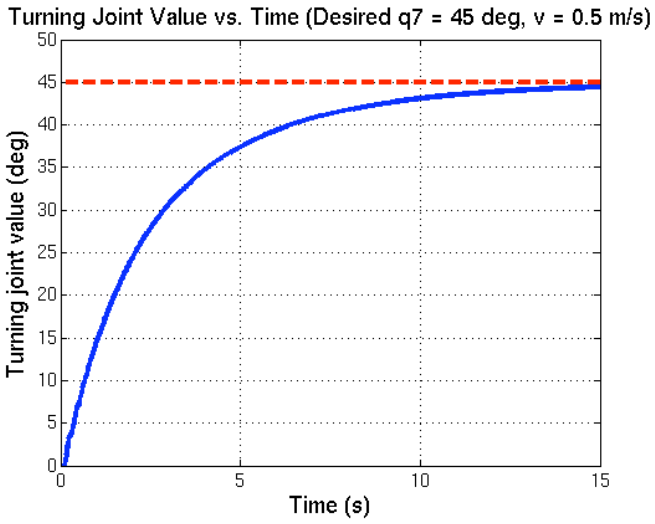

Figure 5.13: Low Speed Turning Response

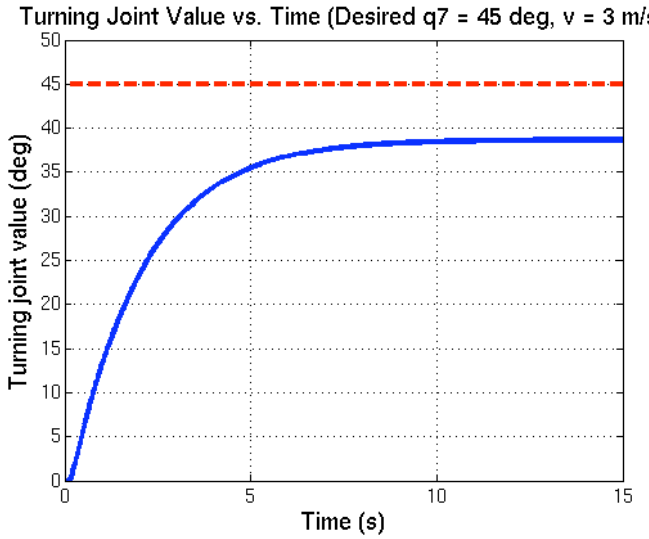

Figure 5.14: High Speed Turning Response

Next, the vehicle (traveling at 1.5 m/s) was commanded to perform a maneuver. The maneuver set the desired turning joint angle to 20 degrees for 6 seconds (starting at t = 2 s), and then returned to driving straight. As can be seen in Figure 5.15, the turning joint angle reached 20 degrees by the end of being commanded to do so, and then returned to zero within a few seconds. The overall heading change (see Figure 5.16) was 60 degrees.

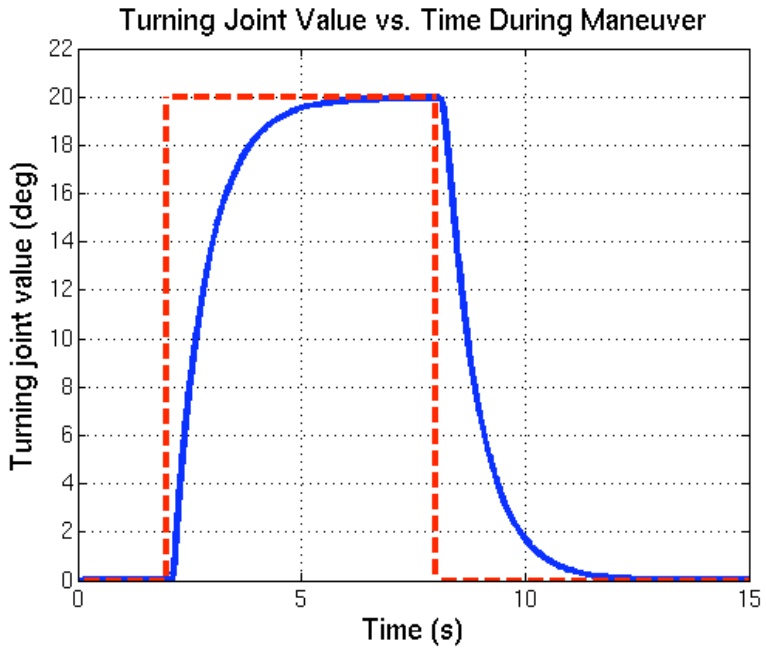

Figure 5.15: Turning Joint Value During Maneuver

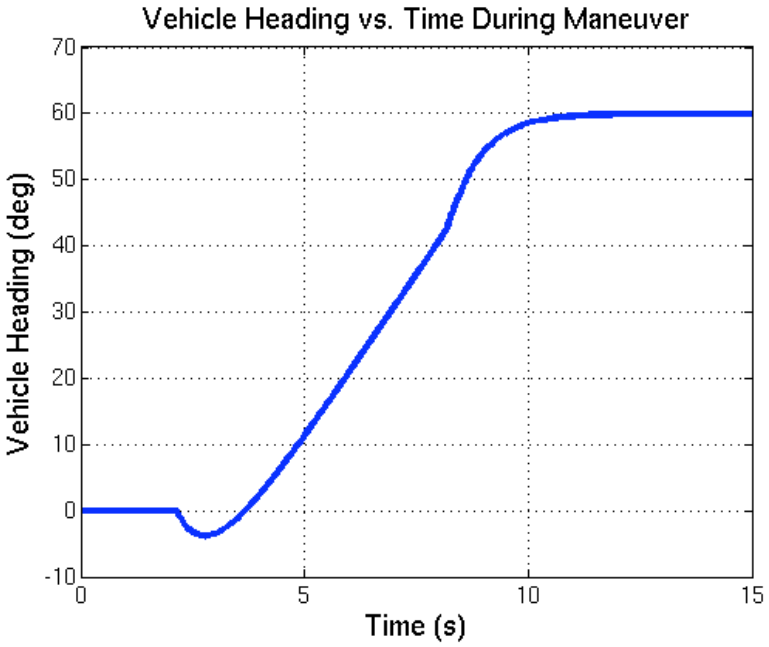

Figure 5.16: Vehicle Heading Angle During Maneuver

To verify, the path is shown in Figure 5.17. The wheel sinkages and torques are shown in Figures 5.18 and 5.19, respectively. It can be seen that during the left

turn, the wheels were deeper on the right than on the left, which is expected.

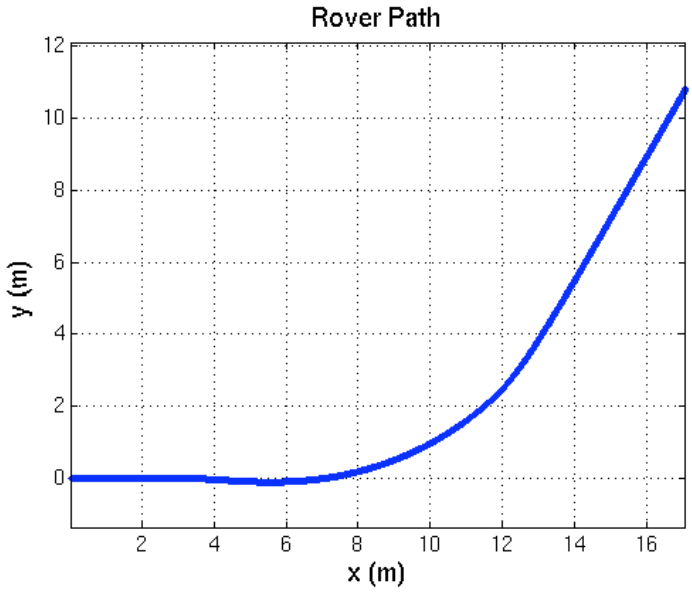

Figure 5.17: Rover Path During Maneuver

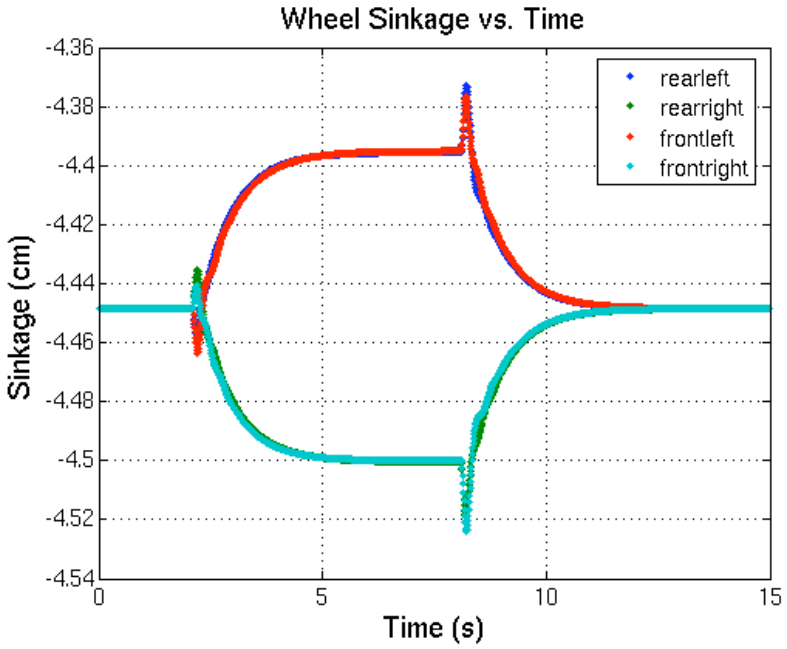

Figure 5.18: Rover Wheel Sinkages During Maneuver

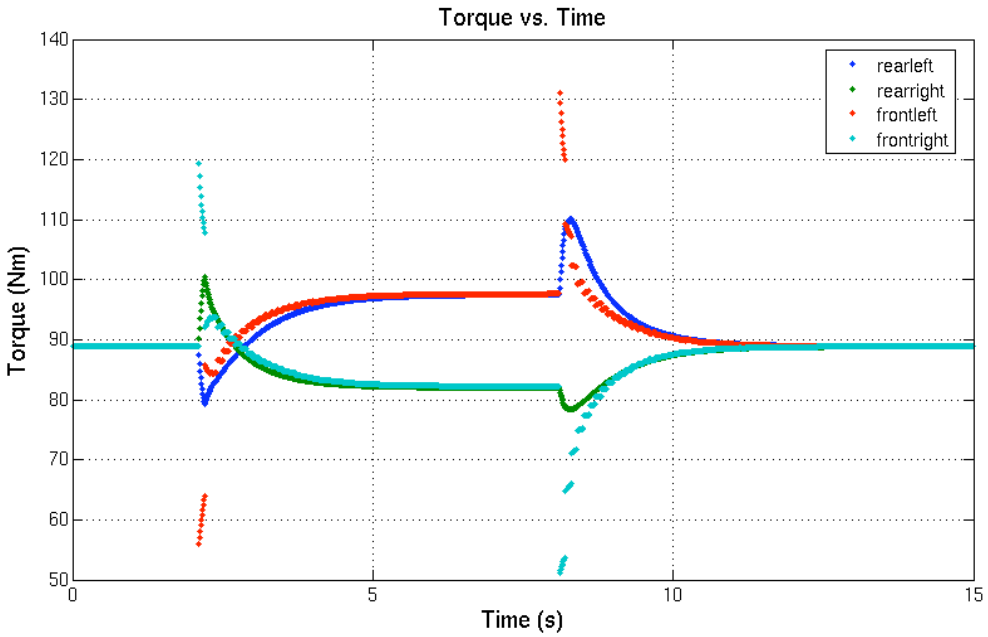

Figure 5.19: Wheel Torques During Maneuver

Chapter 6: Summary, Conclusions, Future Work

6.1 Summary

In this thesis, a computer model was built to simulate the movements of a passively articulated, segmented-body rover. Its main components are a linearized (to reduce computation time) soil-wheel interaction model, a Newton-Euler based dynamic model (easily implemented using DH parameters), and a PD control module that regulates turning the turning joint's position and velocity. The simulation was tested against results for fixed-chassis rovers from past research. Next, the simulation was used to investigate the driving and turning behavior of articulated vehicles. Straight driving cases with disturbances were looked at first. Finally, the feedback control loop was added to the system, and its ability to handle disturbances and turn the vehicle was determined.

6.2 Conclusions

The simulation developed for this research provides qualitatively and to some extent, numerically consistent results with those of previous studies. It also corresponded well to one rough experimental test in the lab. It was sufficient for the purpose of investigating the general behavior of articulated vehicles.

It was found that if disturbed from a straight path, a passively articulated rover will typically not become unstable, but rather settle into a stable turn radius which decreases with increasing velocity and disturbance severity. Additionally, it is possible for such a vehicle to be controlled via a simple PD controller running at 10

Hz that is dependent only on feedback from a turning joint encoder. For higher velocities and larger disturbances or turn commands, the controller required power significantly above the straight driving power, however this may have only been due to the specific gains chosen.

Overall, it was found that simulation is a powerful way to look at vehicle behavior and derive vehicle requirements. It was also found that for the passively-articulated case, that it is controllable by simple control laws, but that optimal choice of gains is important in order to reduce required power and torque, yet keep the steering responsive.

6.3 Future Work

There is much that can still be done to improve the accuracy and capability of the simulation itself. First, if the simulation is ported to a faster language, such as C or C++, then it might be feasible to use more accurate terramechanics stress distributions. It might also be possible to speed the simulation up to real-time and use it as a simulator with humans and other real-world systems in the loop. Currently the simulation runs between 30 and 40 Hz. The timestep used for most work in this research was 0.001 seconds.

Another area where the simulation could be expanded on is the number of vehicle segments and turning joint degrees of freedom modeled. The kinematics module is already extendable to additional segments and joints, but the dynamic model was solved in Mathematica® for specifically two segments and one yaw degree of turning freedom.

For this research a damping term was introduced to approximate a suspension system. A more accurate model of the suspension would improve the prediction power of the simulation, as well as give it the capability to evaluate ride quality and structural design requirements. Rover movement on sloped ground should also be investigated. Turning the gravity vector in the dynamic formulation would simulate sloped behavior.

Simulations should also be performed on other types of rovers to make fair comparisons between configurations. Additionally, it may be interesting to look at the effects of changing segment dimensions, mass distributions, and wheel/soil parameters for the articulated vehicle.

This research modeled the ground's surface as a flat plane. However, it would be useful to add vertical dimension to the terrain and simulate obstacle climbing performance.

Finally, the simulation should be tested against experimental results in the laboratory and out in the field. Good agreement between simulation and actual testing would significantly improve the design process for new rover concepts. For this, a research-grade meter-scale prototype vehicle should be constructed, basing its design on the results from available simulations. Depending on the joint configuration, this prototype could also begin testing the articulated vehicle's obstacle climbing abilities.

Appendices

Appendix A: Matlab Simulation Code

A.1: simcore.m

```
% simcore.m simulates the motions of an articulated vehicle

clear all
close all
clc

% LOAD SOIL PROPERTIES
soil = loadsoil('drysand2.soil')
g = 9.81;   % grav accel (m/s^2)

% VEHICLE PARAMETERS

rover.n = 2;        % number of segments
rover.l = 1.5;        % rover length dimension (distance to next link)
rover.hw = -0.375;  % vertical distance from link plane to wheel hub
rover.hcg = 0.125;  % vertical distance from link plane to segment cg
rover.w = 0.8;     % lateral distance from segment centerline to wheel hub

% wheel and cg coords in link frame
for i=1:rover.n;
    pw{i,1}=[sign(i-1)*rover.l;rover.w;rover.hw]; % wheel coords
    pw{i,2}=[sign(i-1)*rover.l;-rover.w;rover.hw]; % wheel coords
    pcg{i}=[sign(i-1)*rover.l;0;rover.hcg];   % cg coords
end

joints = [0 0 0 1 1 1 ones(1,rover.n-1)];   % joint types (0=P,1=R)

% segment masses
mseg = 200;   % kg
segmass = mseg*ones(rover.n,1);

% wheel properties
wheel.r = 0.5;     % wheel radius
wheel.b = 0.2;     % wheel width
D = 800;    % vertical damping at each wheel (Ns/m)

% solve for wheel state given desired drawbar pull
[sinkage,Tmag,Rconst,slip0] = constvel(mseg,g,wheel,0);

% INITIAL CONDITIONS

vel_init = 0.5; % initial rover velocity

dx = wheel.r-sinkage-rover.hw;    % vertical position
dy = 0; dz = 0;
phix = 0; phiy = 0; phiz = 0;

theta_init = 0;     % initial turning joint angle (deg)
jointrate = 0;      % initial turning joint rate (deg/s)

% initial joint values
q = [dz,dx,dy,phiy,phiz,phix,theta_init]';
qdot = [-vel_init;0;0;0;0;0;jointrate];

% VARIABLE HACKS FOR DYNAMICS EQNS
ms = mseg;
```

```
L = rover.l;
H = pcg{1}(3);
sphereI = 2/5*ms*(pw{1,1}(2)/2)^2;
I11=sphereI;I12=0;I13=0;
I21=0;I22=sphereI;I23=0;
I31=0;I32=0;I33=sphereI;

% Wheel Powers & Torques
motor.Pmax0 = Tmag*(vel_init/((1-slip0)*wheel.r))*ones(rover.n,2);
motor.Pmax = motor.Pmax0;
T = Tmag*ones(rover.n,2);

% Desired values
q7des = 0;      % desired turning joint angle
q7dotdes = 0;   % desired turning jonit rate

% PD Controller gains
PgainP = 100;   % position
PgainD = 100;   % velocity

% SIMULATION PARAMETERS
dt = 0.001;         % timestep length (s)
simseconds = 1;     % length of simulation (s)

num_iter = simseconds/dt;   % number of iterations
iterations = 1;             % start iteration counter at 1
records = 1;                % start records counter at 1
t = 0;                      % start time at 0 seconds
tic                         % start timer

stable = 1;

% MAIN SIMULATION LOOP
while(iterations < num_iter)

    DH = getDH(rover,q);                    % get updated DH parameters

    % Transformation Matrices
    Tadjacent = getTadj(DH);                % All T's between adjacent frames
    Tjoint = getTjoint(Tadjacent,rover);    % T's from joint to base frame
    Twheel = getTwheel(Tjoint,rover,pw);    % T's from wheel to base frame

    plow0 = getplow0(Twheel,rover,wheel);         % base coords of deepest wheel point
    [z0, Rwr2w] = getWRframe(plow0,Twheel,Tjoint,rover);    % max wheel depth &
rotation from wheel frame to wheel reaction frame

    % break out of simulation if wheel breaks surface
    if sum(sum(z0>=0))>0
        stable = 0;
        break;
    end

    % Jacobians
    Jtransw = getJTWheel(rover,wheel,joints,Tadjacent,pw);  % Wheel Jacobians in wheel
frame
    Jtrans = getJTWR(Jtransw,Rwr2w,rover);                  % Wheel Jacobians in wheel
reaction frame

    % Wheel velocities
    wheelvel = getvwheel(rover,qdot,Jtrans);

    % WHEEL FORCES

    th1 = real(acos(1-(-z0)./wheel.r)); % wheel entry angle

    % solve for slip angle and slip ratios
    for i=1:rover.n
        for j=1:2
            betaslip(i,j) = getBeta(wheelvel{i,j});
            soil.kx(i,j) = betaslip(i,j)*soil.dkx+soil.kx0;
```

```
            soil.ky(i,j) = betaslip(i,j)*soil.dky+soil.ky0;
            slip(i,j) = sbisect(th1(i,j),soil.kx(i,j),soil,T(i,j),wheel);
        end
    end

    % location of maximum stresses
    thm = thetamax(th1, slip, soil);

    % max stress values
    sigm = wheel.r^soil.n * (soil.kc/wheel.b+soil.kphi) .* (cos(thm)-
cos(th1)).^soil.n;
    jx = wheel.r.*(th1-thm-(1-slip).*(sin(th1)-sin(thm)));
    taum = (soil.c+sigm*tan(soil.phi)) .* (1-exp((-1*jx)./soil.kx));
    tauym = (soil.c+sigm.*tan(soil.phi)) .* (1-exp( -(wheel.r./soil.ky) .* ((1-
slip).*(th1-thm).*tan(betaslip))));

    % Wheel Force Magnitudes
    FxH = DPH(th1,thm,taum,wheel);              % soil thrust
    FxR = DPR(th1,thm,sigm,wheel);          % soil resistance
    Fz = W(th1, thm, sigm, taum, wheel);        % vertical reaction
    Fytau = abs(Ctau(th1, tauym, wheel));       % lateral stress
    %Fybull(i,j) = quad(@(th) Cbull(th, th1(i,j), wheel, soil),0,th1(i,j));  % lateral
bulldozing
    %Fy = Fytau+Fybull;                          % total lateral forces
    Fy = Fytau;

    % Determine force vectors
    FxHsign = sign(FxH);
    for i=1:rover.n
        Fcw{i}=zeros(3,1); Ncw{i} = zeros(3,1);
        for j=1:2

            % Vectorize in local coords
            if (wheelvel{i,j}(1)==0 && FxHsign(i,j)==1)
                WFx_vec{i,j} = [max([FxH(i,j)+FxR(i,j),0]);0;0];
            elseif (wheelvel{i,j}(1)==0 && FxHsign(i,j)==-1)
                WFx_vec{i,j} = [min([FxH(i,j)-FxR(i,j),0]);0;0];
            else
                WFx_vec{i,j} = [FxH(i,j)+FxR(i,j)*sign(wheelvel{i,j}(1));0;0];
            end
            WFy_vec{i,j} = [0;Fy(i,j)*-sign(wheelvel{i,j}(2));0];
            WFz_vec{i,j} = [0;0;Fz(i,j)-D*wheelvel{i,j}(3)];

            % Transform to link (cg)
            WFx_veclink{i,j} = Rwr2w{i,j}*WFx_vec{i,j};
            WFy_veclink{i,j} = Rwr2w{i,j}*WFy_vec{i,j};
            WFz_veclink{i,j} = Rwr2w{i,j}*WFz_vec{i,j};

            pwheelwrtcg{i,j} = pw{i,j} - pcg{i};    % position of wheel wrt cg (link
frame)
            Fcwlocal = WFx_veclink{i,j} + WFy_veclink{i,j} + WFz_veclink{i,j};
            Fcw{i}= Fcw{i} + Fcwlocal;
            Ncw{i}= Ncw{i} + cross(pwheelwrtcg{i,j},Fcwlocal);
        end
    end

    % BUILD HANDY STATE MATRICES

    % forces & moments
    Fs = [];
    Ns = [];
    for i=1:rover.n
        Fs = [Fs;Fcw{i}];
        Ns = [Ns;Ncw{i}];
    end

    % square of joint rates
    qdotsquared = qdot(4:rover.n+5).^2;

    % joint rate products
```

```
    tempcounter = 0;
    for i=4:rover.n+5-1
        for j=i+1:rover.n+5
            tempcounter = tempcounter+1;
            qdotprods(tempcounter,1)=qdot(i)*qdot(j);
        end
    end

    % DYNAMICS!!!

    % more variable hacks
    q1=q(1);q2=q(2);q3=q(3);q4=q(4);q5=q(5);q6=q(6);q7=q(7);

    % equations from mathematica
    Minv=inv([2*ms,0,0,ms*(2*L*cos(q6 + q7/2.)*cos(q7/2.)*sin(q4) + cos(q4)*(-
2*H*cos(q5) + L*sin(q5)*(sin(q6) + sin(q6 + q7)))),ms*sin(q4)*(2*H*sin(q5) +
L*cos(q5)*(sin(q6) + sin(q6 + q7))),L*ms*(2*cos(q6 + q7/2.)*cos(q7/2.)*sin(q4)*sin(q5)
+ cos(q4)*(sin(q6) + sin(q6 + q7))),L*ms*(cos(q6 + q7)*sin(q4)*sin(q5) +
cos(q4)*sin(q6 + q7));0,2*ms,0,ms*(-2*L*cos(q4)*cos(q6 + q7/2.)*cos(q7/2.) +
sin(q4)*(-2*H*cos(q5) + L*sin(q5)*(sin(q6) + sin(q6 + q7)))),-(ms*cos(q4)*(2*H*sin(q5)
+ L*cos(q5)*(sin(q6) + sin(q6 + q7)))),L*ms*(-2*cos(q4)*cos(q6 +
q7/2.)*cos(q7/2.)*sin(q5) + sin(q4)*(sin(q6) + sin(q6 + q7))),L*ms*(-(cos(q4)*cos(q6 +
q7)*sin(q5)) + sin(q4)*sin(q6 + q7));0,0,2*ms,0,ms*(2*H*cos(q5) - L*sin(q5)*(sin(q6) +
sin(q6 + q7))),2*L*ms*cos(q5)*cos(q6 + q7/2.)*cos(q7/2.),L*ms*cos(q5)*cos(q6 +
q7);ms*(2*L*cos(q6 + q7/2.)*cos(q7/2.)*sin(q4) + cos(q4)*(-2*H*cos(q5) +
L*sin(q5)*(sin(q6) + sin(q6 + q7)))),ms*(-2*L*cos(q4)*cos(q6 + q7/2.)*cos(q7/2.) +
sin(q4)*(-2*H*cos(q5) + L*sin(q5)*(sin(q6) + sin(q6 + q7)))),0,I33 +
(cos(q5)^2*(2*(2*I11 + I22 + 2*H^2*ms + L^2*ms) + (-I11 + I22 + L^2*ms)*cos(2*q6) + (-
I11 + I22)*cos(2*(q6 + q7)) + L^2*ms*(2*cos(q7) + cos(2*(q6 + q7)) + 2*cos(2*q6 +
q7))) + 2*(I22 + 2*I33 + 2*L^2*ms + 2*L^2*ms*cos(q7))*sin(q5)^2 -
4*H*L*ms*cos(q7/2.)*sin(2*q5)*sin(q6 + q7/2.))/2.,2*H*L*ms*cos(q6 +
q7/2.)*cos(q7/2.)*sin(q5) + cos(q5)*(L^2*ms + (-I11 + I22 + L^2*ms)*cos(q7))*sin(2*q6
+ q7),2*(I33 + L^2*ms + L^2*ms*cos(q7))*sin(q5) - H*L*ms*cos(q5)*(sin(q6) + sin(q6 +
q7)),(I33 + L^2*ms + L^2*ms*cos(q7))*sin(q5) - H*L*ms*cos(q5)*sin(q6 +
q7);ms*sin(q4)*(2*H*sin(q5) + L*cos(q5)*(sin(q6) + sin(q6 + q7))),-
(ms*cos(q4)*(2*H*sin(q5) + L*cos(q5)*(sin(q6) + sin(q6 + q7)))),ms*(2*H*cos(q5) -
L*sin(q5)*(sin(q6) + sin(q6 + q7))),2*H*L*ms*cos(q6 + q7/2.)*cos(q7/2.)*sin(q5) +
cos(q5)*(L^2*ms + (-I11 + I22 + L^2*ms)*cos(q7))*sin(2*q6 + q7),(2*(I11 + I22 + I33 +
2*H^2*ms + L^2*ms) + (I11 - I22 - L^2*ms)*cos(2*q6) + 2*L^2*ms*cos(q7) + (I11 - I22 -
L^2*ms)*cos(2*(q6 + q7)) - 2*L^2*ms*cos(2*q6 + q7))/2.,H*L*ms*(cos(q6) + cos(q6 +
q7)),H*L*ms*cos(q6 + q7);L*ms*(2*cos(q6 + q7/2.)*cos(q7/2.)*sin(q4)*sin(q5) +
cos(q4)*(sin(q6) + sin(q6 + q7))),L*ms*(-2*cos(q4)*cos(q6 + q7/2.)*cos(q7/2.)*sin(q5)
+ sin(q4)*(sin(q6) + sin(q6 + q7))),2*L*ms*cos(q5)*cos(q6 + q7/2.)*cos(q7/2.),2*(I33 +
L^2*ms + L^2*ms*cos(q7))*sin(q5) - H*L*ms*cos(q5)*(sin(q6) + sin(q6 +
q7)),H*L*ms*(cos(q6) + cos(q6 + q7)),2*(I33 + L^2*ms + L^2*ms*cos(q7)),I33 + L^2*ms +
L^2*ms*cos(q7);L*ms*(cos(q6 + q7)*sin(q4)*sin(q5) + cos(q4)*sin(q6 + q7)),L*ms*(-
(cos(q4)*cos(q6 + q7)*sin(q5)) + sin(q4)*sin(q6 + q7)),L*ms*cos(q5)*cos(q6 + q7),(I33
+ L^2*ms + L^2*ms*cos(q7))*sin(q5) - H*L*ms*cos(q5)*sin(q6 + q7),H*L*ms*cos(q6 +
q7),I33 + L^2*ms + L^2*ms*cos(q7),I33 + L^2*ms]);
    B=[2*ms*cos(q4)*(2*H*sin(q5) + L*cos(q5)*(sin(q6) + sin(q6 +
q7))),2*L*ms*(2*cos(q4)*cos(q6 + q7/2.)*cos(q7/2.)*sin(q5) - sin(q4)*(sin(q6) + sin(q6
+ q7))),2*L*ms*(cos(q4)*cos(q6 + q7)*sin(q5) - sin(q4)*sin(q6 +
q7)),4*L*ms*cos(q5)*cos(q6 + q7/2.)*cos(q7/2.)*sin(q4),2*L*ms*cos(q5)*cos(q6 +
q7)*sin(q4),2*L*ms*(cos(q4)*cos(q6 + q7) - sin(q4)*sin(q5)*sin(q6 +
q7));2*ms*sin(q4)*(2*H*sin(q5) + L*cos(q5)*(sin(q6) + sin(q6 + q7))),2*L*ms*(2*cos(q6
+ q7/2.)*cos(q7/2.)*sin(q4)*sin(q5) + cos(q4)*(sin(q6) + sin(q6 + q7))),2*L*ms*(cos(q6
+ q7)*sin(q4)*sin(q5) + cos(q4)*sin(q6 + q7)),-4*L*ms*cos(q4)*cos(q5)*cos(q6 +
q7/2.)*cos(q7/2.),-2*L*ms*cos(q4)*cos(q5)*cos(q6 + q7),2*L*ms*(cos(q6 + q7)*sin(q4) +
cos(q4)*sin(q5)*sin(q6 + q7));0,0,0,-4*L*ms*cos(q6 + q7/2.)*cos(q7/2.)*sin(q5),-
2*L*ms*cos(q6 + q7)*sin(q5),-2*L*ms*cos(q5)*sin(q6 + q7);(-
8*H*L*ms*cos(2*q5)*cos(q7/2.)*sin(q6 + q7/2.) + 2*sin(2*q5)*(-2*I11 + 2*I33 - 2*H^2*ms
+ L^2*ms + (I11 - I22 - L^2*ms)*cos(q7)*cos(2*q6 + q7) + 2*L^2*ms*sin(q6)*sin(q6 +
q7)))/2.,-2*H*L*ms*cos(q6 + q7/2.)*cos(q7/2.)*sin(2*q5) - 2*cos(q5)^2*(L^2*ms + (-I11
+ I22 + L^2*ms)*cos(q7))*sin(2*q6 + q7),(2*cos(q6 + q7)*(-(H*L*ms*sin(2*q5)) +
cos(2*q5)*((I11 - I22)*sin(q6 + q7) - L^2*ms*(sin(q6) + sin(q6 + q7)))) + (I11 -
I22)*sin(2*(q6 + q7)) - L^2*ms*(3*sin(q7) + sin(2*(q6 + q7)) + sin(2*q6 +
q7)))/2.,cos(q5)*(2*(I33 + L^2*ms) + (-I11 + I22 + L^2*ms)*cos(2*q6) + (-I11 +
I22)*cos(2*(q6 + q7)) + L^2*ms*(2*cos(q7) + cos(2*(q6 + q7)) + 2*cos(2*q6 +
q7))),cos(q5)*(I33 + L^2*ms + (-I11 + I22)*cos(2*(q6 + q7)) + L^2*ms*(cos(q7) +
cos(2*(q6 + q7)) + cos(2*q6 + q7))),-2*L*ms*(H*cos(q5)*cos(q6 + q7) +
```

```
L*sin(q5)*sin(q7));0,(-8*H*L*ms*cos(q7/2.)*sin(q5)*sin(q6 + q7/2.) - 4*cos(q5)*(I33 +
L^2*ms + (I11 - I22 - L^2*ms)*cos(q7)*cos(2*q6 + q7) + 2*L^2*ms*sin(q6)*sin(q6 +
q7)))/2.,(-2*(I33 + L^2*ms)*cos(q5) + 2*(-I11 + I22 + L^2*ms)*cos(q5)*cos(2*(q6 + q7))
- 4*L*ms*(H*sin(q5) + L*cos(q5)*sin(q6))*sin(q6 + q7))/2.,2*(L^2*ms + (-I11 + I22 +
L^2*ms)*cos(q7))*sin(2*q6 + q7),2*cos(q6 + q7)*(L^2*ms*sin(q6) + (-I11 + I22 +
L^2*ms)*sin(q6 + q7)),-2*H*L*ms*sin(q6 + q7);2*H*L*ms*sin(q5)*(sin(q6) + sin(q6 + q7))
+ 2*cos(q5)*(I33 + L^2*ms + (I11 - I22 - L^2*ms)*cos(q7)*cos(2*q6 + q7) +
2*L^2*ms*sin(q6)*sin(q6 + q7)),0,-2*L^2*ms*sin(q5)*sin(q7),0,0,-
2*L^2*ms*sin(q7);(2*cos(q5)*(I33 + L^2*ms + (I11 - I22)*cos(2*(q6 + q7)) +
L^2*ms*(cos(q7) - cos(2*(q6 + q7)) - cos(2*q6 + q7))) + 4*H*L*ms*sin(q5)*sin(q6 +
q7))/2.,2*L^2*ms*sin(q5)*sin(q7),0,0,0,0];
    C=[ms*(2*L*cos(q4)*cos(q6 + q7/2.)*cos(q7/2.) + sin(q4)*(2*H*cos(q5) -
L*sin(q5)*(sin(q6) + sin(q6 + q7)))),ms*sin(q4)*(2*H*cos(q5) - L*sin(q5)*(sin(q6) +
sin(q6 + q7))),L*ms*(2*cos(q4)*cos(q6 + q7/2.)*cos(q7/2.) - sin(q4)*sin(q5)*(sin(q6) +
sin(q6 + q7))),L*ms*(cos(q4)*cos(q6 + q7) - sin(q4)*sin(q5)*sin(q6 +
q7));ms*(2*L*cos(q6 + q7/2.)*cos(q7/2.)*sin(q4) + cos(q4)*(-2*H*cos(q5) +
L*sin(q5)*(sin(q6) + sin(q6 + q7)))),ms*cos(q4)*(-2*H*cos(q5) + L*sin(q5)*(sin(q6) +
sin(q6 + q7))),L*ms*(2*cos(q6 + q7/2.)*cos(q7/2.)*sin(q4) + cos(q4)*sin(q5)*(sin(q6) +
sin(q6 + q7))),L*ms*(cos(q6 + q7)*sin(q4) + cos(q4)*sin(q5)*sin(q6 + q7));0,-
(ms*(2*H*sin(q5) + L*cos(q5)*(sin(q6) + sin(q6 + q7)))),-
2*L*ms*cos(q5)*cos(q7/2.)*sin(q6 + q7/2.),-(L*ms*cos(q5)*sin(q6 +
q7));0,2*H*L*ms*cos(q5)*cos(q6 + q7/2.)*cos(q7/2.) - (L^2*ms + (-I11 + I22 +
L^2*ms)*cos(q7))*sin(q5)*sin(2*q6 + q7),-2*H*L*ms*cos(q5)*cos(q6 + q7/2.)*cos(q7/2.),-
(L*ms*(H*cos(q5)*cos(q6 + q7) +
L*sin(q5)*sin(q7)));(16*H*L*ms*cos(2*q5)*cos(q7/2.)*sin(q6 + q7/2.) - 4*sin(2*q5)*(-
2*I11 + 2*I33 - 2*H^2*ms + L^2*ms + (I11 - I22 - L^2*ms)*cos(q7)*cos(2*q6 + q7) +
2*L^2*ms*sin(q6)*sin(q6 + q7)))/8.,0,-(H*L*ms*(sin(q6) + sin(q6 + q7))),-
(H*L*ms*sin(q6 + q7));H*L*ms*cos(q6 + q7/2.)*cos(q7/2.)*sin(2*q5) + cos(q5)^2*(L^2*ms
+ (-I11 + I22 + L^2*ms)*cos(q7))*sin(2*q6 + q7),-((L^2*ms + (-I11 + I22 +
L^2*ms)*cos(q7))*sin(2*q6 + q7)),0,-(L^2*ms*sin(q7));(2*cos(q6 + q7)*(H*L*ms*sin(2*q5)
+ cos(2*q5)*(L^2*ms*sin(q6) + (-I11 + I22 + L^2*ms)*sin(q6 + q7))) + (-I11 +
I22)*sin(2*(q6 + q7)) + L^2*ms*(3*sin(q7) + sin(2*(q6 + q7)) + sin(2*q6 + q7)))/4.,-
(cos(q6 + q7)*(L^2*ms*sin(q6) + (-I11 + I22 + L^2*ms)*sin(q6 +
q7))),L^2*ms*sin(q7),0];
    G=[0,2*ms,0,ms*(-2*L*cos(q4)*cos(q6 + q7/2.)*cos(q7/2.) + sin(q4)*(-2*H*cos(q5) +
L*sin(q5)*(sin(q6) + sin(q6 + q7)))),-(ms*cos(q4)*(2*H*sin(q5) + L*cos(q5)*(sin(q6) +
sin(q6 + q7)))),L*ms*(-2*cos(q4)*cos(q6 + q7/2.)*cos(q7/2.)*sin(q5) + sin(q4)*(sin(q6)
+ sin(q6 + q7))),L*ms*(-(cos(q4)*cos(q6 + q7)*sin(q5)) + sin(q4)*sin(q6 + q7))]';
    Fwcoeff=[-(cos(q4)*cos(q6)) + sin(q4)*sin(q5)*sin(q6),cos(q6)*sin(q4)*sin(q5) +
cos(q4)*sin(q6),-(cos(q5)*sin(q4)),-(cos(q4)*cos(q6 + q7)) + sin(q4)*sin(q5)*sin(q6 +
q7),cos(q6 + q7)*sin(q4)*sin(q5) + cos(q4)*sin(q6 + q7),-(cos(q5)*sin(q4));-
(cos(q6)*sin(q4)) - cos(q4)*sin(q5)*sin(q6),-(cos(q4)*cos(q6)*sin(q5)) +
sin(q4)*sin(q6),cos(q4)*cos(q5),-(cos(q6 + q7)*sin(q4)) - cos(q4)*sin(q5)*sin(q6 +
q7),-(cos(q4)*cos(q6 + q7)*sin(q5)) + sin(q4)*sin(q6 +
q7),cos(q4)*cos(q5);cos(q5)*sin(q6),cos(q5)*cos(q6),sin(q5),cos(q5)*sin(q6 +
q7),cos(q5)*cos(q6 + q7),sin(q5);H*cos(q5)*cos(q6),-
(H*cos(q5)*sin(q6)),0,H*cos(q5)*cos(q6 + q7) + L*sin(q5)*sin(q7),L*(1 +
cos(q7))*sin(q5) - H*cos(q5)*sin(q6 + q7),-2*L*cos(q5)*cos(q6 +
q7/2.)*cos(q7/2.);H*sin(q6),H*cos(q6),0,H*sin(q6 + q7),H*cos(q6 + q7),-(L*(sin(q6) +
sin(q6 + q7)));0,0,0,L*sin(q7),L*(1 + cos(q7)),0;0,0,0,0,L,0];

Nwcoeff=[0,0,0,0,0,0;0,0,0,0,0,0;0,0,0,0,0,0;cos(q5)*sin(q6),cos(q5)*cos(q6),sin(q5),c
os(q5)*sin(q6 + q7),cos(q5)*cos(q6 + q7),sin(q5);-cos(q6),sin(q6),0,-cos(q6 +
q7),sin(q6 + q7),0;0,0,1,0,0,1;0,0,0,0,0,1];

    qddot = Minv*(-B*qdotprods-C*qdotsquared-g*G+Fwcoeff*Fs+Nwcoeff*Ns);

    % UPDATE JOINT VALUES & RATES
    q = q+qdot*dt+1/2*qddot*dt^2;
    qdot = qdot+qddot*dt;
    %q(7)=0;qdot(7)=0;qddot(7)=0;    % use if want to fix turning joint

    % PD Controller
    if (mod(iterations,100)==0)

        deltatheta = q(7)-q7des;
        deltathetadot = qdot(7)-q7dotdes;
        PdeltaP = PgainP*deltatheta;
        PdeltaD = PgainD*deltathetadot;
```

```
        motor.Pmax=motor.Pmax0;
        motor.Pmax(2,2) = motor.Pmax0(2,2)-PdeltaP-PdeltaD;
        motor.Pmax(2,1) = motor.Pmax0(2,1)+PdeltaP+PdeltaD;

    end

    % Calculate Wheel Torques
    for i=1:rover.n
        for j=1:2
            if slip(i,j)>=0
                T(i,j) = motor.Pmax(i,j)*(wheel.r*(slip(i,j)-1))/(-
abs(wheelvel{i,j}(1)));
            else
                T(i,j) =
motor.Pmax(i,j)/((slip(i,j)+1)*abs(wheelvel{i,j}(1))/wheel.r);
            end
        end
    end

    % PRINT OUT INFORMATION TO SCREEN & RECORD SIMULATION DATA
    iterations = iterations + 1;
    if (mod(iterations,10)==0)

        % Screen Output
        if (mod(iterations,10)==0)
            minsleft = (num_iter-iterations)/(iterations/toc)/60;
            minsleftrnd = fix(minsleft);
            secondsleft = round((minsleft-minsleftrnd)*60);
            clc
            disp([num2str(minsleftrnd), ' min ', num2str(secondsleft), ' sec
remaining, Completed ',num2str(iterations), ' iterations in ', num2str(round(toc)), '
seconds at ',num2str(round(iterations/toc)),' Hz']);
            disp(['simtime = ',num2str(t),' / ',num2str(simseconds)]);
            disp(['turning joint angle: ',num2str(180/pi*q(7))]);
            disp(['turning joint angle rate: ',num2str(180/pi*qdot(7))]);
            disp(['vel = ',num2str(sqrt(qdot(1).^2+qdot(3).^2))])
            disp(['DP = ',num2str(FxH(1,1)+FxR(1,1))])
            disp(['slip = ',num2str(slip(1,1))])
            disp(['sinkage = ',num2str(z0(1,1))])
            disp(['R = ',num2str(FxR(1,1))])
            disp(['heading: ',num2str(180/pi*q(6))]);
            disp(['heading rate: ',num2str(180/pi*qdot(6))]);
        end

        % Record Data
        trec(records) = t;
        qrec(:,records) = q;
        qrecdot(:,records) = qdot;
        qrecddot(:,records) = qddot;

        counter = 1;
        for i=1:rover.n
            for j=1:2
                z0rec(records,counter)=z0(i,j);
                th1rec(records,counter)=th1(i,j);
                betarec(records,counter)=betaslip(i,j);
                sliprec(records,counter)=slip(i,j);
                Hrec(records,counter)=FxH(i,j);
                Rrec(records,counter)=FxR(i,j);
                %Fytaurec(records,counter)=Fytau(i,j);
                %Fybullrec(records,counter)=Fybull(i,j);
                %Fzrec(records,counter)=Fz(i,j);
                Torec(records,counter)=T(i,j);
                Porec(records,counter)=motor.Pmax(i,j);
                %wvrecx(records,counter)=wheelvel{i,j}(1);
                %wvrecy(records,counter)=wheelvel{i,j}(2);
                wfxrec(records,counter)=WFx_vec{i,j}(1);
                wfyrec(records,counter)=WFy_vec{i,j}(2);
                wfzrec(records,counter)=WFz_vec{i,j}(3);
                counter = counter+1;
```

```
            end
        end
        records = records + 1;
    end

    % update time
    t = t + dt;

end
toc

plotit
```

A.2: loadsoil.m

```
% soil = loadsoil(file)
%
% loads soil properties from a file (mks units)
% each property name and value on one line separated by a space
%
% 5/1/07: phi (friction angle) is stored in file in degrees, but converted
% to radians in this routine

function soil = loadsoil(file)

% default soil parameters if not specified in file
soil.name = ['default'];
soil.n = 1;
soil.c = 1000;
soil.phi = 45*pi/180;
soil.kc = 1000;
soil.kphi = 1000000;
soil.kx0 = 0.036;
soil.ky0 = 0.013;
soil.dkx = 0.043;
soil.dky = 0.020;
soil.a0 = 0.15;
soil.a1 = 0.4;
soil.density = 2000;

% read data from .soil file
[param_name,value]=textread(file,'%s %s','commentstyle','matlab');

% assign data values to soil structure
for i=1:length(param_name)
    switch param_name{i}
        case 'name'
            soil.name = value{i};
        case 'n'
            soil.n = str2num(value{i});
        case 'c'
            soil.c = str2num(value{i});
        case 'phi'
            soil.phi = str2num(value{i})*pi/180;
        case 'kc'
            soil.kc = str2num(value{i});
        case 'kphi'
            soil.kphi = str2num(value{i});
        case 'kx0'
            soil.kx0 = str2num(value{i});
        case 'ky0'
            soil.ky0 = str2num(value{i});
        case 'dkx'
            soil.dkx = str2num(value{i});
        case 'dky'
            soil.dky = str2num(value{i});
        case 'a0'
```

```
            soil.a0 = str2num(value{i});
        case 'a1'
            soil.a1 = str2num(value{i});
        case 'density'
            soil.density = str2num(value{i});
    end
end

% set kx & ky values for zero slip angle
soil.kx = soil.kx0;
soil.ky = soil.ky0;
```

A.3: constvel.m

```
% function [z,T,R,s] = constvel(mseg,g,wheel,DP_des)
%
% for constant drawbar pull driving, determines sinkage, driving torque,
% rolling resistance, and slip per wheel
%
% mseg = segment mass
% g = gravity
% wheel = wheel properties
% DP_des = desired drawbar pull

function [z,T,R,s] = constvel(mseg,g,wheel,DP_des)

m = mseg/2; % mass per wheel
soil = loadsoil('drysand2.soil');

s = fzero(@(s) zeroDP(s, wheel,soil, g, m, DP_des, 0),[0 1]);

zguess = 0.04;
for i=1:length(s)
    z = fzero(@(z) zeroFz(z, s, wheel, soil, g, m, 0),zguess);
end

th1 = acos(1-z/wheel.r);
thm = thetamax(th1, s, soil);
sigm = wheel.r^soil.n * (soil.kc/wheel.b+soil.kphi) .* (cos(thm)-cos(th1)).^soil.n;

jx = wheel.r*(th1-thm-(1-s)*(sin(th1)-sin(thm)));
taum = (soil.c+sigm.*tan(soil.phi)) .* (1-exp(-jx/soil.kx));

R = abs(DPR(th1, thm, sigm, wheel));
T = 1/2*wheel.r^2*wheel.b*taum.*th1;
```

A.4: zeroFz.m

```
% function Fz = zeroFz(z, slip, wheel, soil, g, m,lamda)
%
% minimized to find vertical force equilibrium

function Fz = zeroFz(z, slip, wheel, soil, g, m,lamda)

th1 = real(acos(1-z./wheel.r));
thm = thetamax(th1, slip, soil);
jx = wheel.r*(th1-thm-(1-slip)*(sin(th1)-sin(thm)));
sigm = wheel.r^soil.n * (soil.kc/wheel.b+soil.kphi) .* (cos(thm)-cos(th1)).^soil.n;
taum = (soil.c+sigm.*tan(soil.phi)) .* (1-exp(-jx/soil.kx));

Fz = W(th1, thm, sigm, taum, wheel) - m*g*(1+lamda);
```

A.5: zeroDP.m

```
% function DP = zeroDP(s, wheel,soil, g, m, DP_des,lamda)
%
% minimized to find desire drawbar pull solution

function DP = zeroDP(s, wheel,soil, g, m, DP_des,lamda)

zguess = 0.05;
z = fzero(@(z) zeroFz(z, s, wheel, soil, g, m, lamda),zguess);
th1 = acos(1-z/wheel.r);

thm = thetamax(th1, s, soil);
sigm = wheel.r^soil.n * (soil.kc/wheel.b+soil.kphi) .* (cos(thm)-cos(th1)).^soil.n;
jx = wheel.r*(th1-thm-(1-s)*(sin(th1)-sin(thm)));
taum = (soil.c+sigm.*tan(soil.phi)) .* (1-exp(-jx/soil.kx));

R = abs(DPR(th1, thm, sigm, wheel));
H = DPH(th1,thm,taum,wheel);
DP = H-R-DP_des;
```

A.6: getDH.m

```
% function DHparams = getDH(rover,q)
%
% assigns DH parameters based on current joint values q

function DHparams = getDH(rover,q)

DHparams.i = [1:rover.n+5];
DHparams.alpha = [0,pi/2,-pi/2,0,pi/2,pi/2,zeros(1,rover.n-1)];
DHparams.a = [0,0,0,0,0,0,rover.l,2*rover.l*ones(1,rover.n-2)];
DHparams.d = [q(1:3)',zeros(1,rover.n+2)];
DHparams.theta = [pi/2,-pi/2,-pi/2,q(4),q(5)+pi/2,q(6)-pi/2,q(7:end)'];
```

A.7: getTadj.m

```
% function Tadjacent = getTadj(DH)
%
% calculates transformation matrices between all adjacent frames (i to i-1)

function Tadjacent = getTadj(DH)

for i = 1:(DH.i(end))
    Tadjacent{i} = [cos(DH.theta(i)),-sin(DH.theta(i)),0,DH.a(i);
        sin(DH.theta(i))*cos(DH.alpha(i)),cos(DH.theta(i))*cos(DH.alpha(i)),-
sin(DH.alpha(i)),-sin(DH.alpha(i))*DH.d(i);

sin(DH.theta(i))*sin(DH.alpha(i)),cos(DH.theta(i))*sin(DH.alpha(i)),cos(DH.alpha(i)),c
os(DH.alpha(i))*DH.d(i);
        0,0,0,1];
end
```

A.8: getTjoint.m

```
% function Tjoint = getTjoint(Tadjacent,rover)
%
% calculates transformation matrices between all frames and the base frame

function Tjoint = getTjoint(Tadjacent,rover)
```

```
for i = 1:rover.n
    Tjoint{i} = 1;
    for j = i+5:-1:1
        Tjoint{i} = Tadjacent{j} * Tjoint{i};
    end
end
```

A.9: getTwheel.m

```
% function Twheel = getTwheel(Tjoint,rover,pw)
%
% calculates transformation matrices from wheel hubs to base frame

function Twheel = getTwheel(Tjoint,rover,pw)

for i=1:rover.n
    for j=1:2
        Tw2j = [eye(3,3),pw{i,j};0,0,0,1];
        Twheel{i,j}=Tjoint{i}*Tw2j;
    end
end
```

A.10: getplow0.m

```
% function plow0 = getplow0(Twheel,rover,wheel)
%
% returns the lowest point of each wheel in the base frame

function plow0 = getplow0(Twheel,rover,wheel)

for i=1:rover.n
    for j=1:2
        thetamin = atan2(Twheel{i,j}(1,3),Twheel{i,j}(1,1));
        if thetamin>0
            thetaminother = thetamin - pi;
        elseif thetamin<=0
           thetaminother = thetamin + pi;
        end

        plow0{i,j} = Twheel{i,j}*[wheel.r*cos(thetamin);0;wheel.r*sin(thetamin);1];
        plow0other{i,j} =
Twheel{i,j}*[wheel.r*cos(thetaminother);0;wheel.r*sin(thetaminother);1];

        if plow0{i,j}(1)>plow0other{i,j}(1)
            plow0{i,j} = plow0other{i,j};
        end
    end
end
```

A.11: getWRframe.m

```
% function [zdepth,Rwr2w] = getWRframe(plow0,Twheel,Tjoint,rover)
%
% returns the sinkage of each wheel, and the transformation between wheel
% reaction frame and wheel (hub) frame

function [zdepth,Rwr2w] = getWRframe(plow0,Twheel,Tjoint,rover)

for i=1:rover.n
    for j=1:2
        rline = Twheel{i,j}(1:3,4)-plow0{i,j}(1:3);
        rslope0 = rline/norm(rline);
```

```
        zdepth(i,j) = plow0{i,j}(1)/rslope0(1);
        Xpart = cross(Tjoint{i}(1:3,1:3)*[0;1;0],rslope0);
        Rwr20{i,j} = [Xpart/norm(Xpart),Tjoint{i}(1:3,1:3)*[0;1;0],rslope0];
        Rwr2w{i,j} = inv(Tjoint{i}(1:3,1:3))*Rwr20{i,j};
    end
end
```

A.12: getJTWheel.m

```
% function JT = getJTWheel(rover,wheel,joints,Tadjacent,pw)
%
% get jacobian for all wheels in wheel (hub) frames

function JT = getJTWheel(rover,wheel,joints,Tadjacent,pw)

wsign = -1;
for i = 1:rover.n
    for j = 1:2
        % indiv. wheel
        for k = 1:i+5
            Trans = eye(4);
            for m = i+5:-1:k+1
                Trans = Tadjacent{m} * Trans;
            end
            z{k} = Trans(1:3,1:3)'*[0;0;1];
            p1{k} = Trans * [pw{i,j};1];
            p1{k} = p1{k}(1:3);
            p2{k} = Trans(1:3,1:3)'*p1{k};
            zp{k} = cross(z{k}(1:3),p2{k});
        end
        term1 = cell2mat(zp);
        term2 = cell2mat(z);
        lam_joints = diag(joints(1:i+5));
        JT{i,j} = term1*lam_joints + term2*(eye(i+5)-lam_joints);
        wsign = -wsign;
    end
end
```

A.13: getJTWR.m

```
% function Jtrans = getJTWR(Jtransw,Rwr2w,rover)
%
% get jacobian for all wheels in wheel reaction frames

function Jtrans = getJTWR(Jtransw,Rwr2w,rover)

for i=1:rover.n
    for j=1:2
        Jtrans{i,j}=Rwr2w{i,j}'*Jtransw{i,j};
    end
end
```

A.14: getvwheel.m

```
% function wheelvel = getvwheel(rover,qdot,Jtrans)
%
% wheel velocities in wheel reaction frames

function wheelvel = getvwheel(rover,qdot,Jtrans)

for i=1:rover.n
    qdotloc=qdot(1:i+5);
```

```
    for j=1:2
        wheelvel{i,j}=Jtrans{i,j}*qdotloc;
    end
end
```

A.15: getBeta.m

```
%function betaslip = getBeta(rover,Jtrans,qdot)
%
% calculates slip angle for each wheel

function betaslip = getBeta(wheelvel);

betaslip = abs(atan2(wheelvel(2),wheelvel(1)));

% if on wrong side of wheel
if (betaslip > pi/2)
    betaslip = pi-betaslip;
end
```

A.16: thetamax.m

```
% function thm = thetamax(th1, s, soil)
%
% calculate the location of maximum stresses

function thm = thetamax(th1, s, soil)

thm = (soil.a0 + soil.a1*s).*th1;
```

A.17: DPH.m

```
% function Fx = DPH(th1, thm, taum, wheel)
%
% calculates soil thrust H

function Fx = DPH(th1, thm, taum, wheel)

Fx = wheel.r*wheel.b./(thm.*(th1-thm)).*(taum.*(th1.*cos(thm)-thm.*cos(th1)-th1+thm));
```

A.18: DPR.m

```
% function Fx = DPR(th1, thm, sigm, wheel)
%
% calculates soil rolling resistance R

function Fx = DPR(th1, thm, sigm, wheel)

Fx = wheel.r*wheel.b./(thm.*(th1-thm)).*(-sigm.*(th1.*sin(thm)-thm.*sin(th1)));
```

A.19: W.m

```
% function Fz = W(th1, thm, sigm, taum, wheel)
%
% calculates wheel vertical load
```

```
function Fz = W(th1, thm, sigm, taum, wheel)

Fz = wheel.r*wheel.b./(thm.*(th1-thm)).*(sigm.*(th1.*cos(thm)-thm.*cos(th1)-
th1+thm)+taum.*(th1.*sin(thm)-thm.*sin(th1)));

% use for no powered wheel contribution to load
%Fz = wheel.r*wheel.b./(thm.*(th1-thm)).*(sigm.*(th1.*cos(thm)-thm.*cos(th1)-
th1+thm));
```

A.20: Ctau.m

```
% function Fytau = Ctau(th1, taum, wheel)
%
% calculates wheel side load

function Fytau = Ctau(th1, taum, wheel)

Fytau = 1/2.*taum*wheel.r*wheel.b.*th1;
```

A.21: plotit.m

```
% plotit plots a multitude of data from the simulation including joint
% values, positions, torques, slips, powers, etc. over the entire sim time

fcount=1;
rect = [1500 50 1666 970];

figure('Position',rect);
% backwards!
figc = 5;
figr = 2;

subplot(figr,figc,fcount);plot(trec,qrec(4:7,:)*180/pi,'.');legend y z 6
7;fcount=fcount+1;

subplot(figr,figc,fcount);plot(trec,qrecdot(4:7,:)*180/pi,'.');legend dy dz d6
d7;fcount=fcount+1;

fwdvelrec = sqrt(qrecdot(1,:).^2+qrecdot(3,:).^2);
subplot(figr,figc,fcount);plot(trec,sqrt(qrecdot(1,:).^2+qrecdot(3,:).^2),'.');title('
vel');fcount=fcount+1;
hold on;
plot(trec,-qrecdot(1,:),'r.');
hold on;
plot(trec,qrecdot(3,:),'g.');
legend fwd x y;
subplot(figr,figc,fcount);plot(-
qrec(1,:),qrec(3,:),'.');fcount=fcount+1;title('path');axis equal;

subplot(figr,figc,fcount);plot(trec,z0rec,'.');title('wheel depth');legend rearleft
rearright frontleft frontright;fcount=fcount+1;
%figure;plot(trec,th1rec*180/pi,'.');title('theta1');legend rearleft rearright
frontleft frontright;fcount=fcount+1;
subplot(figr,figc,fcount);plot(trec,betarec*180/pi,'.');title('beta');legend rearleft
rearright frontleft frontright;fcount=fcount+1;
subplot(figr,figc,fcount);plot(trec,sliprec,'.');title('slip');legend rearleft
rearright frontleft frontright;fcount=fcount+1;
%figure;plot(trec,Hrec,'.');title('FxH');legend rearleft rearright frontleft
frontright;
%figure;plot(trec,Rrec,'.');title('FxR');legend rearleft rearright frontleft
frontright;
%figure;plot(trec,Fytaurec,'.');title('Fytau');legend rearleft rearright frontleft
frontright;
%figure;plot(trec,Fybullrec,'.');title('Fybull');legend rearleft rearright frontleft
```

```
frontright;
%figure;plot(trec,Fzrec,'.');title('Fz');legend rearleft rearright frontleft
frontright;
%figure;plot(trec,Torec,'.');title('Torque');legend rearleft rearright frontleft
frontright;fcount=fcount+1;

subplot(figr,figc,fcount);plot(trec,wfxrec,'.');title('Fx actual');legend rearleft
rearright frontleft frontright;fcount=fcount+1;
subplot(figr,figc,fcount);plot(trec,wfyrec,'.');title('Fy actual');legend rearleft
rearright frontleft frontright;fcount=fcount+1;
%subplot(figr,figc,fcount);plot(trec,wfzrec,'.');title('Fz actual');legend rearleft
rearright frontleft frontright;fcount=fcount+1;
subplot(figr,figc,fcount);plot(trec,Porec,'.');title('Power');legend rearleft
rearright frontleft frontright;fcount=fcount+1;

%figure;figure;plot(trec,wvrec,'.');title('wheel vel x');legend rearleft rearright
frontleft frontright;

disp(' ')
disp(['vel = ',num2str(sqrt(qrecdot(1,end).^2+qrecdot(3,end).^2))])
disp(['DP = ',num2str(FxH(end,end)+FxR(end,end))])
disp(['slip = ',num2str(sliprec(end,end))])
disp(['sinkage = ',num2str(z0rec(end,end))])
disp(['R = ',num2str(FxR(end,end))])
```

A.22: drysand2.soil

```
% This File Contains Soil Data For
% Dry Sand (Wong 2001, pg 136) & (Dimi 2001, pg 40)
% Created 5/2/07
%
%START
name drysand2
n 1.1
c 1040
phi 38
kc 990
kphi 1528430
kx0 0.0254
ky0 0.0254
dkx 0
dky 0
a0 0.28
a1 0.35
density 1500
%END
```

Appendix B: Mathematica® Dynamic Modeling

The following pages show the Mathetmatica® code used to formulate the dynamic equations for the articulated rover.

Joint positions, velocities, accelerations, and type (1=Rotational, 0=Prismatic)

$$q = \begin{pmatrix} q1 \\ q2 \\ q3 \\ q4 \\ q5 \\ q6 \\ q7 \end{pmatrix}; \quad qd = \begin{pmatrix} q1d \\ q2d \\ q3d \\ q4d \\ q5d \\ q6d \\ q7d \end{pmatrix}; \quad qdd = \begin{pmatrix} q1dd \\ q2dd \\ q3dd \\ q4dd \\ q5dd \\ q6dd \\ q7dd \end{pmatrix}; \quad joints = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix};$$

Segment masses & Inertia Tensor

$$m = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ ms \\ ms \end{pmatrix}; \quad Iseg = \begin{pmatrix} I11 & 0 & 0 \\ 0 & I22 & 0 \\ 0 & 0 & I33 \end{pmatrix};$$

Joint velocities & accelerations (divided into revolute and prismatic)

$$ddot = \begin{pmatrix} q1d \\ q2d \\ q3d \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}; \quad dddot = \begin{pmatrix} q1dd \\ q2dd \\ q3dd \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}; \quad thetadot = \begin{pmatrix} 0 \\ 0 \\ 0 \\ q4d \\ q5d \\ q6d \\ q7d \end{pmatrix}; \quad thetaddot = \begin{pmatrix} 0 \\ 0 \\ 0 \\ q4dd \\ q5dd \\ q6dd \\ q7dd \end{pmatrix};$$

Forces and Moments at link cg

$$Fcw[1] = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}; \quad Fcw[2] = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}; \quad Fcw[3] = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}; \quad Fcw[4] = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix};$$

$$Fcw[5] = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}; \quad Fcw[6] = \begin{pmatrix} Fcw6x \\ Fcw6y \\ Fcw6z \end{pmatrix}; \quad Fcw[7] = \begin{pmatrix} Fcw7x \\ Fcw7y \\ Fcw7z \end{pmatrix};$$

$$Ncw[1] = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}; \quad Ncw[2] = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}; \quad Ncw[3] = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}; \quad Ncw[4] = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix};$$

$$Ncw[5] = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}; \quad Ncw[6] = \begin{pmatrix} Ncw6x \\ Ncw6y \\ Ncw6z \end{pmatrix}; \quad Ncw[7] = \begin{pmatrix} Ncw7x \\ Ncw7y \\ Ncw7z \end{pmatrix};$$

DH Parameters

$$\text{alpha} = \begin{pmatrix} 0 \\ \text{Pi}/2 \\ -\text{Pi}/2 \\ 0 \\ \text{Pi}/2 \\ \text{Pi}/2 \\ 0 \end{pmatrix}; \quad a = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ L \end{pmatrix}; \quad d = \begin{pmatrix} q1 \\ q2 \\ q3 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}; \quad \text{theta} = \begin{pmatrix} \text{Pi}/2 \\ -\text{Pi}/2 \\ -\text{Pi}/2 \\ q4 \\ \text{Pi}/2 + q5 \\ -\text{Pi}/2 + q6 \\ q7 \end{pmatrix};$$

Position of link cg in link frame

$$pcg[1] = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}; \quad pcg[2] = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}; \quad pcg[3] = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix};$$

$$pcg[4] = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}; \quad pcg[5] = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}; \quad pcg[6] = \begin{pmatrix} 0 \\ 0 \\ H \end{pmatrix}; \quad pcg[7] = \begin{pmatrix} L \\ 0 \\ H \end{pmatrix};$$

Generate transformation matrices for adjacent links

```
For[i = 1, i ≤ 7, i++,
 Tadj[i] =
  (
   {{Cos[theta[[i, 1]]], -Sin[theta[[i, 1]]],
     Sin[theta[[i, 1]]] * Cos[alpha[[i, 1]]], Cos[theta[[i, 1]]] * Cos[alpha[[i, 1]]], -Sin[al
     Sin[theta[[i, 1]]] * Sin[alpha[[i, 1]]], Cos[theta[[i, 1]]] * Sin[alpha[[i, 1]]],  Cos[alp
     0, 0, 0]}
   ,
   ;
  )
 Radj[i] = Take[Tadj[i], {1, 3}, {1, 3}];
]
```

Velocities and accelerations of inertial frame

$$w[1] = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}; \quad wdot[1] = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}; \quad vdot[1] = \begin{pmatrix} g \\ 0 \\ 0 \end{pmatrix};$$

Outward iterations to compute velocities and accelerations

```
For[i = 1, i ≤ 7, i++,
 w[i + 1] = Transpose[Radj[i]].w[i] + {{0}, {0}, {thetadot[[i, 1]]}};
 wdot[i + 1] = Transpose[Radj[i]].wdot[i] + Cross[Flatten[Transpose[Radj[i]].w[i]],
    {0, 0, thetadot[[i, 1]]}] + {{0}, {0}, {thetaddot[[i, 1]]}};
 vdot[i + 1] = Transpose[Radj[i]].
    (Cross[Flatten[wdot[i]], Flatten[Take[Tadj[i], {1, 3}, {4}]]] + Cross[Flatten[w[i]],
       Cross[Flatten[w[i]], Flatten[Take[Tadj[i], {1, 3}, {4}]]]] + vdot[i]) +
   2 * Cross[Flatten[w[i + 1]], {0, 0, ddot[[i, 1]]}] + {0, 0, dddot[[i, 1]]};
 vcdot[i + 1] = Cross[Flatten[wdot[i + 1]], Flatten[pcg[i]]] +
   Cross[Flatten[w[i + 1]], Cross[Flatten[w[i + 1]], Flatten[pcg[i]]]] + vdot[i + 1];
 Fcg[i + 1] = m[[i, 1]] * vcdot[i + 1] + Fcw[i];
 (*Ncg[i+1]=Ncw[i];*)
 Ncg[i + 1] = Ncw[i] + Iseg.wdot[i + 1] + Cross[Flatten[w[i + 1]], Flatten[Iseg.w[i + 1]]];
]
```

$$f[8] = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}; \; n[8] = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}; \; Tadj[8] = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}; \; Radj[8] = Take[Tadj[8], \{1, 3\}, \{1, 3\}];$$

Inward iterations to compute forces and torques

```
For[i = 7, i ≥ 1, i--,
 f[i] = Radj[i + 1].f[i + 1] + Fcg[i + 1];
 n[i] = Ncg[i + 1] + Radj[i + 1].n[i + 1] + Cross[Flatten[pcg[i]], Flatten[Fcg[i + 1]]] +
   Cross[Flatten[Take[Tadj[i + 1], {1, 3}, {4}]], Flatten[Radj[i + 1].f[i + 1]]];
 If[joints[[i, 1]] == 0, tau[i] = f[i][[3]], tau[i] = n[i][[3]]];
]
```

Variables to find coefficients for. qd4^2+, qd4*qd5+

```
qdsquared = Take[qd^2, {4, 7}, {1}];
wheelforces = Flatten[{Fcw[6], Fcw[7]}];
wheeltorques = Flatten[{Ncw[6], Ncw[7]}];
counter = 0;
prods = Table[0, {6}];
For[i = 4, i ≤ 6, i++,
 For[k = i + 1, k ≤ 7, k++,
  counter++;
  prods[[counter]] = (qd[[i, 1]] * qd[[k, 1]]);
 ]
]
varlist = Flatten[{qdd, prods, qdsquared, g, wheelforces, wheeltorques}]
```

```
{q1dd, q2dd, q3dd, q4dd, q5dd, q6dd, q7dd, q4d q5d, q4d q6d,
 q4d q7d, q5d q6d, q5d q7d, q6d q7d, q4d^2, q5d^2, q6d^2, q7d^2, g, Fcw6x, Fcw6y,
 Fcw6z, Fcw7x, Fcw7y, Fcw7z, Ncw6x, Ncw6y, Ncw6z, Ncw7x, Ncw7y, Ncw7z}
```

Find coefficients to use in dynamics equation coefficient matrices

```
For[i = 1, i ≤ 7, i++,
 tau[i] = Collect[tau[i], varlist, FullSimplify];
 taucoeff[i] = Coefficient[tau[i][[1]], varlist];
]
```

Inertia Matrix

```
M = Table[0, {7}, {7}];
For[i = 1, i ≤ 7, i++,
 For[j = 1, j ≤ 7, j++,
  M[[i, j]] = taucoeff[i][[j]];
 ]
]
```

Centrifugal Force Matrix

```
CF = Table[0, {7}, {4}];
For[i = 1, i ≤ 7, i++,
 For[j = 1, j ≤ 4, j++,
  CF[[i, j]] = taucoeff[i][[j + 13]];
 ]
]
```

Coriolis Forces

```
BF = Table[0, {7}, {6}];
For[i = 1, i ≤ 7, i++,
 For[j = 1, j ≤ 6, j++,
  BF[[i, j]] = taucoeff[i][[j + 7]];
 ]
]
```

Gravity Forces

```
G = Table[0, {7}, {1}];
For[i = 1, i ≤ 7, i++,
 G[[i]] = taucoeff[i][[18]];
]
```

```
FWcoeff = Table[0, {7}, {6}];
NWcoeff = Table[0, {7}, {6}];
For[i = 1, i ≤ 7, i++,
 For[j = 1, j ≤ 6, j++,
  FWcoeff[[i, j]] = taucoeff[i][[j + 18]];
  NWcoeff[[i, j]] = taucoeff[i][[j + 24]];
 ]
]

SetOptions[$Output, PageWidth → 20000];
FortranForm[M]
```

```
List(List(2*ms,0,0,ms*(2*L*Cos(q6 + q7/2.)*Cos(q7/2.)*Sin(q4) + Cos(q4)*(-2*H*Cos(q5)
+ L*Sin(q5)*(Sin(q6) + Sin(q6 + q7)))),ms*Sin(q4)*(2*H*Sin(q5) + L*Cos(q5)*(Sin(q6) +
Sin(q6 + q7))),L*ms*(2*Cos(q6 + q7/2.)*Cos(q7/2.)*Sin(q4)*Sin(q5) + Cos(q4)*(Sin(q6) +
Sin(q6 + q7))),L*ms*(Cos(q6 + q7)*Sin(q4)*Sin(q5) + Cos(q4)*Sin(q6 +
q7))),List(0,2*ms,0,ms*(-2*L*Cos(q4)*Cos(q6 + q7/2.)*Cos(q7/2.) +
Sin(q4)*(-2*H*Cos(q5) + L*Sin(q5)*(Sin(q6) + Sin(q6 + q7)))),-(ms*Cos(q4)*(2*H*Sin(q5)
+ L*Cos(q5)*(Sin(q6) + Sin(q6 + q7)))),L*ms*(-2*Cos(q4)*Cos(q6 +
q7/2.)*Cos(q7/2.)*Sin(q5) + Sin(q4)*(Sin(q6) + Sin(q6 + q7))),L*ms*(-(Cos(q4)*Cos(q6 +
q7)*Sin(q5)) + Sin(q4)*Sin(q6 + q7))),List(0,0,2*ms,0,ms*(2*H*Cos(q5) -
L*Sin(q5)*(Sin(q6) + Sin(q6 + q7))),2*L*ms*Cos(q5)*Cos(q6 +
q7/2.)*Cos(q7/2.),L*ms*Cos(q5)*Cos(q6 + q7)),List(ms*(2*L*Cos(q6 +
q7/2.)*Cos(q7/2.)*Sin(q4) + Cos(q4)*(-2*H*Cos(q5) + L*Sin(q5)*(Sin(q6) + Sin(q6 +
q7)))),ms*(-2*L*Cos(q4)*Cos(q6 + q7/2.)*Cos(q7/2.) + Sin(q4)*(-2*H*Cos(q5) +
L*Sin(q5)*(Sin(q6) + Sin(q6 + q7)))),0,I33 + (Cos(q5)**2*(2*(2*I11 + I22 + 2*H**2*ms +
L**2*ms) + (-I11 + I22 + L**2*ms)*Cos(2*q6) + (-I11 + I22)*Cos(2*(q6 + q7)) +
L**2*ms*(2*Cos(q7) + Cos(2*(q6 + q7)) + 2*Cos(2*q6 + q7))) + 2*(I22 + 2*I33 +
2*L**2*ms + 2*L**2*ms*Cos(q7))*Sin(q5)**2 - 4*H*L*ms*Cos(q7/2.)*Sin(2*q5)*Sin(q6 +
q7/2.))/2.,2*H*L*ms*Cos(q6 + q7/2.)*Cos(q7/2.)*Sin(q5) + Cos(q5)*(L**2*ms + (-I11 +
I22 + L**2*ms)*Cos(q7))*Sin(2*q6 + q7),2*(I33 + L**2*ms + L**2*ms*Cos(q7))*Sin(q5) -
H*L*ms*Cos(q5)*(Sin(q6) + Sin(q6 + q7)),(I33 + L**2*ms + L**2*ms*Cos(q7))*Sin(q5) -
H*L*ms*Cos(q5)*Sin(q6 + q7)),List(ms*Sin(q4)*(2*H*Sin(q5) + L*Cos(q5)*(Sin(q6) +
Sin(q6 + q7))),-(ms*Cos(q4)*(2*H*Sin(q5) + L*Cos(q5)*(Sin(q6) + Sin(q6 +
q7)))),ms*(2*H*Cos(q5) - L*Sin(q5)*(Sin(q6) + Sin(q6 + q7))),2*H*L*ms*Cos(q6 +
q7/2.)*Cos(q7/2.)*Sin(q5) + Cos(q5)*(L**2*ms + (-I11 + I22 +
L**2*ms)*Cos(q7))*Sin(2*q6 + q7),(2*(I11 + I22 + I33 + 2*H**2*ms + L**2*ms) + (I11 -
I22 - L**2*ms)*Cos(2*q6) + 2*L**2*ms*Cos(q7) + (I11 - I22 - L**2*ms)*Cos(2*(q6 + q7))
- 2*L**2*ms*Cos(2*q6 + q7))/2.,H*L*ms*(Cos(q6) + Cos(q6 + q7)),H*L*ms*Cos(q6 +
q7)),List(L*ms*(2*Cos(q6 + q7/2.)*Cos(q7/2.)*Sin(q4)*Sin(q5) + Cos(q4)*(Sin(q6) +
Sin(q6 + q7))),L*ms*(-2*Cos(q4)*Cos(q6 + q7/2.)*Cos(q7/2.)*Sin(q5) + Sin(q4)*(Sin(q6)
+ Sin(q6 + q7))),2*L*ms*Cos(q5)*Cos(q6 + q7/2.)*Cos(q7/2.),2*(I33 + L**2*ms +
L**2*ms*Cos(q7))*Sin(q5) - H*L*ms*Cos(q5)*(Sin(q6) + Sin(q6 + q7)),H*L*ms*(Cos(q6) +
Cos(q6 + q7)),2*(I33 + L**2*ms + L**2*ms*Cos(q7)),I33 + L**2*ms +
L**2*ms*Cos(q7)),List(L*ms*(Cos(q6 + q7)*Sin(q4)*Sin(q5) + Cos(q4)*Sin(q6 +
q7)),L*ms*(-(Cos(q4)*Cos(q6 + q7)*Sin(q5)) + Sin(q4)*Sin(q6 + q7)),L*ms*Cos(q5)*Cos(q6
+ q7),(I33 + L**2*ms + L**2*ms*Cos(q7))*Sin(q5) - H*L*ms*Cos(q5)*Sin(q6 +
q7),H*L*ms*Cos(q6 + q7),I33 + L**2*ms + L**2*ms*Cos(q7),I33 + L**2*ms))
```

FortranForm[BF]

```
List(List(2*ms*Cos(q4)*(2*H*Sin(q5) + L*Cos(q5)*(Sin(q6) + Sin(q6 +
q7))),2*L*ms*(2*Cos(q4)*Cos(q6 + q7/2.)*Cos(q7/2.)*Sin(q5) - Sin(q4)*(Sin(q6) + Sin(q6
+ q7))),2*L*ms*(Cos(q4)*Cos(q6 + q7)*Sin(q5) - Sin(q4)*Sin(q6 +
q7)),4*L*ms*Cos(q5)*Cos(q6 + q7/2.)*Cos(q7/2.)*Sin(q4),2*L*ms*Cos(q5)*Cos(q6 +
q7)*Sin(q4),2*L*ms*(Cos(q4)*Cos(q6 + q7) - Sin(q4)*Sin(q5)*Sin(q6 +
q7))),List(2*ms*Sin(q4)*(2*H*Sin(q5) + L*Cos(q5)*(Sin(q6) + Sin(q6 +
q7))),2*L*ms*(2*Cos(q6 + q7/2.)*Cos(q7/2.)*Sin(q4)*Sin(q5) + Cos(q4)*(Sin(q6) + Sin(q6
+ q7))),2*L*ms*(Cos(q6 + q7)*Sin(q4)*Sin(q5) + Cos(q4)*Sin(q6 +
q7)),-4*L*ms*Cos(q4)*Cos(q5)*Cos(q6 + q7/2.)*Cos(q7/2.),-2*L*ms*Cos(q4)*Cos(q5)*Cos(q6
+ q7),2*L*ms*(Cos(q6 + q7)*Sin(q4) + Cos(q4)*Sin(q5)*Sin(q6 +
q7))),List(0,0,0,-4*L*ms*Cos(q6 + q7/2.)*Cos(q7/2.)*Sin(q5),-2*L*ms*Cos(q6 +
q7)*Sin(q5),-2*L*ms*Cos(q5)*Sin(q6 + q7)),List((-8*H*L*ms*Cos(2*q5)*Cos(q7/2.)*Sin(q6
+ q7/2.) + 2*Sin(2*q5)*(-2*I11 + 2*I33 - 2*H**2*ms + L**2*ms + (I11 - I22 -
L**2*ms)*Cos(q7)*Cos(2*q6 + q7) + 2*L**2*ms*Sin(q6)*Sin(q6 + q7)))/2.,-2*H*L*ms*Cos(q6
+ q7/2.)*Cos(q7/2.)*Sin(2*q5) - 2*Cos(q5)**2*(L**2*ms + (-I11 + I22 +
L**2*ms)*Cos(q7))*Sin(2*q6 + q7),(2*Cos(q6 + q7)*(-(H*L*ms*Sin(2*q5)) +
Cos(2*q5)*((I11 - I22)*Sin(q6 + q7) - L**2*ms*(Sin(q6) + Sin(q6 + q7)))) + (I11 -
I22)*Sin(2*(q6 + q7)) - L**2*ms*(3*Sin(q7) + Sin(2*(q6 + q7)) + Sin(2*q6 +
q7)))/2.,Cos(q5)*(2*(I33 + L**2*ms) + (-I11 + I22 + L**2*ms)*Cos(2*q6) + (-I11 +
I22)*Cos(2*(q6 + q7)) + L**2*ms*(2*Cos(q7) + Cos(2*(q6 + q7)) + 2*Cos(2*q6 +
q7))),Cos(q5)*(I33 + L**2*ms + (-I11 + I22)*Cos(2*(q6 + q7)) + L**2*ms*(Cos(q7) +
Cos(2*(q6 + q7)) + Cos(2*q6 + q7))),-2*L*ms*(H*Cos(q5)*Cos(q6 + q7) +
L*Sin(q5)*Sin(q7))),List(0,(-8*H*L*ms*Cos(q7/2.)*Sin(q5)*Sin(q6 + q7/2.) -
4*Cos(q5)*(I33 + L**2*ms + (I11 - I22 - L**2*ms)*Cos(q7)*Cos(2*q6 + q7) +
2*L**2*ms*Sin(q6)*Sin(q6 + q7)))/2.,(-2*(I33 + L**2*ms)*Cos(q5) + 2*(-I11 + I22 +
L**2*ms)*Cos(q5)*Cos(2*(q6 + q7)) - 4*L*ms*(H*Sin(q5) + L*Cos(q5)*Sin(q6))*Sin(q6 +
q7))/2.,2*(L**2*ms + (-I11 + I22 + L**2*ms)*Cos(q7))*Sin(2*q6 + q7),2*Cos(q6 +
q7)*(L**2*ms*Sin(q6) + (-I11 + I22 + L**2*ms)*Sin(q6 + q7)),-2*H*L*ms*Sin(q6 +
q7)),List(2*H*L*ms*Sin(q5)*(Sin(q6) + Sin(q6 + q7)) + 2*Cos(q5)*(I33 + L**2*ms + (I11
- I22 - L**2*ms)*Cos(q7)*Cos(2*q6 + q7) + 2*L**2*ms*Sin(q6)*Sin(q6 +
q7)),0,-2*L**2*ms*Sin(q5)*Sin(q7),0,0,-2*L**2*ms*Sin(q7)),List((2*Cos(q5)*(I33 +
L**2*ms + (I11 - I22)*Cos(2*(q6 + q7)) + L**2*ms*(Cos(q7) - Cos(2*(q6 + q7)) -
Cos(2*q6 + q7))) + 4*H*L*ms*Sin(q5)*Sin(q6 +
q7))/2.,2*L**2*ms*Sin(q5)*Sin(q7),0,0,0,0))
```

FortranForm[CF]

```
List(List(ms*(2*L*Cos(q4)*Cos(q6 + q7/2.)*Cos(q7/2.) + Sin(q4)*(2*H*Cos(q5) -
L*Sin(q5)*(Sin(q6) + Sin(q6 + q7)))),ms*Sin(q4)*(2*H*Cos(q5) - L*Sin(q5)*(Sin(q6) +
Sin(q6 + q7))),L*ms*(2*Cos(q4)*Cos(q6 + q7/2.)*Cos(q7/2.) - Sin(q4)*Sin(q5)*(Sin(q6) +
Sin(q6 + q7))),L*ms*(Cos(q4)*Cos(q6 + q7) - Sin(q4)*Sin(q5)*Sin(q6 +
q7))),List(ms*(2*L*Cos(q6 + q7/2.)*Cos(q7/2.)*Sin(q4) + Cos(q4)*(-2*H*Cos(q5) +
L*Sin(q5)*(Sin(q6) + Sin(q6 + q7)))),ms*Cos(q4)*(-2*H*Cos(q5) + L*Sin(q5)*(Sin(q6) +
Sin(q6 + q7))),L*ms*(2*Cos(q6 + q7/2.)*Cos(q7/2.)*Sin(q4) + Cos(q4)*Sin(q5)*(Sin(q6) +
Sin(q6 + q7))),L*ms*(Cos(q6 + q7)*Sin(q4) + Cos(q4)*Sin(q5)*Sin(q6 +
q7))),List(0,-(ms*(2*H*Sin(q5) + L*Cos(q5)*(Sin(q6) + Sin(q6 +
q7)))),-2*L*ms*Cos(q5)*Cos(q7/2.)*Sin(q6 + q7/2.),-(L*ms*Cos(q5)*Sin(q6 +
q7))),List(0,2*H*L*ms*Cos(q5)*Cos(q6 + q7/2.)*Cos(q7/2.) - (L**2*ms + (-I11 + I22 +
L**2*ms)*Cos(q7))*Sin(q5)*Sin(2*q6 + q7),-2*H*L*ms*Cos(q5)*Cos(q6 +
q7/2.)*Cos(q7/2.),-(L*ms*(H*Cos(q5)*Cos(q6 + q7) +
L*Sin(q5)*Sin(q7)))),List((16*H*L*ms*Cos(2*q5)*Cos(q7/2.)*Sin(q6 + q7/2.) -
4*Sin(2*q5)*(-2*I11 + 2*I33 - 2*H**2*ms + L**2*ms + (I11 - I22 -
L**2*ms)*Cos(q7)*Cos(2*q6 + q7) + 2*L**2*ms*Sin(q6)*Sin(q6 +
q7)))/8.,0,-(H*L*ms*(Sin(q6) + Sin(q6 + q7))),-(H*L*ms*Sin(q6 +
q7))),List(H*L*ms*Cos(q6 + q7/2.)*Cos(q7/2.)*Sin(2*q5) + Cos(q5)**2*(L**2*ms + (-I11 +
I22 + L**2*ms)*Cos(q7))*Sin(2*q6 + q7),-((L**2*ms + (-I11 + I22 +
L**2*ms)*Cos(q7))*Sin(2*q6 + q7)),0,-(L**2*ms*Sin(q7))),List((2*Cos(q6 +
q7)*(H*L*ms*Sin(2*q5) + Cos(2*q5)*(L**2*ms*Sin(q6) + (-I11 + I22 + L**2*ms)*Sin(q6 +
q7))) + (-I11 + I22)*Sin(2*(q6 + q7)) + L**2*ms*(3*Sin(q7) + Sin(2*(q6 + q7)) +
Sin(2*q6 + q7)))/4.,-(Cos(q6 + q7)*(L**2*ms*Sin(q6) + (-I11 + I22 + L**2*ms)*Sin(q6 +
q7))),L**2*ms*Sin(q7),0))
```

FortranForm[G]

```
List(0,2*ms,0,ms*(-2*L*Cos(q4)*Cos(q6 + q7/2.)*Cos(q7/2.) + Sin(q4)*(-2*H*Cos(q5) +
L*Sin(q5)*(Sin(q6) + Sin(q6 + q7))),-(ms*Cos(q4)*(2*H*Sin(q5) + L*Cos(q5)*(Sin(q6) +
Sin(q6 + q7)))),L*ms*(-2*Cos(q4)*Cos(q6 + q7/2.)*Cos(q7/2.)*Sin(q5) + Sin(q4)*(Sin(q6)
+ Sin(q6 + q7))),L*ms*(-(Cos(q4)*Cos(q6 + q7)*Sin(q5)) + Sin(q4)*Sin(q6 + q7)))
```

FortranForm[FWcoeff]

```
List(List(-(Cos(q4)*Cos(q6)) + Sin(q4)*Sin(q5)*Sin(q6),Cos(q6)*Sin(q4)*Sin(q5) +
Cos(q4)*Sin(q6),-(Cos(q5)*Sin(q4)),-(Cos(q4)*Cos(q6 + q7)) + Sin(q4)*Sin(q5)*Sin(q6 +
q7),Cos(q6 + q7)*Sin(q4)*Sin(q5) + Cos(q4)*Sin(q6 +
q7),-(Cos(q5)*Sin(q4))),List(-(Cos(q6)*Sin(q4)) -
Cos(q4)*Sin(q5)*Sin(q6),-(Cos(q4)*Cos(q6)*Sin(q5)) +
Sin(q4)*Sin(q6),Cos(q4)*Cos(q5),-(Cos(q6 + q7)*Sin(q4)) - Cos(q4)*Sin(q5)*Sin(q6 +
q7),-(Cos(q4)*Cos(q6 + q7)*Sin(q5)) + Sin(q4)*Sin(q6 +
q7),Cos(q4)*Cos(q5)),List(Cos(q5)*Sin(q6),Cos(q5)*Cos(q6),Sin(q5),Cos(q5)*Sin(q6 +
q7),Cos(q5)*Cos(q6 +
q7),Sin(q5)),List(H*Cos(q5)*Cos(q6),-(H*Cos(q5)*Sin(q6)),0,H*Cos(q5)*Cos(q6 + q7) +
L*Sin(q5)*Sin(q7),L*(1 + Cos(q7))*Sin(q5) - H*Cos(q5)*Sin(q6 + q7),-2*L*Cos(q5)*Cos(q6
+ q7/2.)*Cos(q7/2.)),List(H*Sin(q6),H*Cos(q6),0,H*Sin(q6 + q7),H*Cos(q6 +
q7),-(L*(Sin(q6) + Sin(q6 + q7)))),List(0,0,0,L*Sin(q7),L*(1 +
Cos(q7)),0),List(0,0,0,0,L,0))
```

FortranForm[NWcoeff]

```
List(List(0,0,0,0,0,0),List(0,0,0,0,0,0),List(0,0,0,0,0,0),List(Cos(q5)*Sin(q6),Cos(q5)
*Cos(q6),Sin(q5),Cos(q5)*Sin(q6 + q7),Cos(q5)*Cos(q6 +
q7),Sin(q5)),List(-Cos(q6),Sin(q6),0,-Cos(q6 + q7),Sin(q6 +
q7),0),List(0,0,1,0,0,1),List(0,0,0,0,0,1))
```

Appendix C: Soil Parameters

Soil Parameter	Symbol	Units	Dry Sand	Lunar Soil Simulant
Exponent of sinkage	n	-	1.1	1.0
Soil cohesion	c	kPa	1.04	0.8
Internal friction angle	ϕ	deg	38	37.2
Cohesion modulus	k_c	kN/m^{n+1}	0.99	1.37
Friction modulus	k_ϕ	kN/m^{n+2}	1528.43	814
Longitudinal shear deformation modulus	k_x	m	0.0254	0.036
Lateral shear deformation modulus	k_y	m	0.0254	0.013
Maximum stress parameter	a_0	-	0.28	0.4
Maximum stress parameter	a_1	-	0.35	0.15
Soil density	ρ	kg/m^2	1500	1600

References: [Apostolopoulos,01], [Wong,01], and [Ishigami,07]

References

[Azad,05] Azad, N.L. et al. "Off-road lateral stability analysis of an articulated steer vehicle with a rear mounted load," *International Journal of Vehicle Systems Modeling and Testing*, Vol. 1, Nos. 1/2/3, pp. 106-130, 2005.

[Azad,07] Azad, N.L. et al. "Robust state feedback stabilization of articulated steer vehicles," *Vehicle Systems Dynamics*, Vol. 45, No. 3, pp. 249-275, 2007.

[Apostolopoulos,01] Apostolopoulos, D.S. "Analytical Configuration of Wheeled Robobtic Locomotion," Ph.D. Thesis, Carnegie Mellon University. Pittsburgh, PA, 2001.

[Bekker,60] Bekker, M.G. *Off-the-road Locomotion*. The University of Michigan Press, Ann Arbor, MI, 1960.

[Craig,05] Craig, J.J. *Introduction to Robotics: Mechanics and Control*, 3rd ed., Prentice Hall, Upper Saddle River, NJ, 2005.

[He,05] He, Y. et al. "Dynamic modeling and stability analysis of articulated frame steer vehicles," *International Journal of Vehicle Systems*, Vol. 12, No. 1, pp. 28-59, 2005.

[Holm,70] Holm, I.C. "Articulated, Wheeled Off-The-Road Vehicles," *Journal of Terramechanics*, Vol. 7, No. 1, pp. 19-54, 1970.

[Iagnemma,01] Iagnemma, K.D. "Rough-Terrain Mobile Robot Planning and Control with Application to Planetary Exploration," Ph.D. Thesis, Massachusetts Institute of Technology. Cambridge, MA, 2001.

[Ishigami,05] Ishigami G. et al. "Steering Trajectory Analysis of Planetary Exploration Rovers Based on All-Wheel Dynamics Model," *Proceedings of The 8th International Symposium on Artificial Intelligence, Robotics, and Automation in Space*, Munich, Gernmany, 2005.

[Ishigami,07] Ishigami, G. et al. "Terramechanics-Based Model for Steering Maneuver of Planetary Exploration Rovers on Loose Soil," *Journal of Field Robotics*, Vol. 24, No. 3, pp. 233-250, 2007.

[Miller,02] Miller, D.P. and Lee, T.L. "High-Speed Traversal of Rough Terrain Using a Rocker-Bogie Mobility System," Proceedings of Robotics 2002: The 5th Inernational Conference on Robotics for Challenging Situations and Environments, Albuquerque, New Mexico, March 2002.

[Oida,87] Oida, A. "Turning Behavior of Articulated Frame Steering Tractors – Part 2. Motion of Tractors With Drawbar Pull," *Journal of*

Terramechanics, Vol. 24, No. 1, pp. 57-73, 1987.

[Sullivan,94] Sullivan, T.A. *Catalog of Apollo Experiment Operations*, NASA Reference Publication 1317, 1994.

[Wong,67] Wong, J. and Reece, A.R. "Prediction of Rigid Wheel Performance Based on the Analysis of Soil-Wheel Stresses Part I. Performance of Driven Rigid Wheels," *Journal of Terramechanics*, Vol. 4, No. 1, pp. 81-98, 1967.

[Wong,01] Wong, J. *Theory of Ground Vehicles*, 3rd ed., Wiley-Interscience, 2001.

[Yoshida,98] Yoshida, K. et al. "Dynamic Simulation of an Articulated Off-Road Vehicle," AIAA Modeling and Simulation Technologies Conference and Exhibit, pp. 257-261, Boston, Massachusetts, 1998.

[Yoshida,04] Yoshida, K. and Ishigami, G. "Steering Characteristics of a Rigid Wheel for Exploration on Loose Soil," *Proceedings of the IEEE International Conference on Intelligent Robotics and Systems*, Sendai, Japan, 2004.