

Wheel Design Optimization for Locomotion in Granular Beds using Resistive Force Theory

by

James Slonaker

Submitted to the Department of Mechanical Engineering
in partial fulfillment of the requirements for the degree of

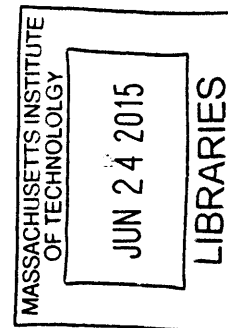
Bachelor of Science in Mechanical Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2015

© Massachusetts Institute of Technology 2015. All rights reserved.



Author **Signature redacted**
Department of Mechanical Engineering
May 8, 2015

Certified by **Signature redacted**
Ken Kamrin
Assistant Professor of Mechanical Engineering
Thesis Supervisor

Accepted by **Signature redacted**
Anette Hosoi
Associate Professor of Mechanical Engineering
Undergraduate Officer



77 Massachusetts Avenue
Cambridge, MA 02139
<http://libraries.mit.edu/ask>

DISCLAIMER NOTICE

Due to the condition of the original material, there are unavoidable flaws in this reproduction. We have made every effort possible to provide you with the best copy available.

Thank you.

The images contained in this document are of the best quality available.

Wheel Design Optimization for Locomotion in Granular Beds using Resistive Force Theory

by

James Slonaker

Submitted to the Department of Mechanical Engineering
on May 8, 2015, in partial fulfillment of the
requirements for the degree of
Bachelor of Science in Mechanical Engineering

Abstract

The application of Resistive Force Theory to further understand the dynamics of wheeled locomotion through granular materials was explored. Resistive Force Theory is a new terradynamic model that simplifies the calculation of the forces applied to bodies moving through granular media because it utilizes linear superposition and is easily scalable to any material of interest. First, a MATLAB simulation was created to test different wheel designs rotating through sand. The designs tested include a four-spoke design, consisting of four treads that have a hinge halfway down their length set to a specific angle, and a "superball" design, consisting of different solid shapes defined by the "superball" equation. The average velocity and power were found for each case to find an optimal design. It was found that four-spoke designs with an angle $\theta < 180^\circ$ were optimal as they reached the highest velocities, while requiring the least power.

Next, dimensional analysis was performed to find a global scaling relationship for the RFT wheel designs. Scaling laws for the power and velocity were found that allow different wheel designs and conditions to be simulated with an entirely new system. Using the simulation, it was found that the scaling law for a tire rotating on Mars could be tested on Earth up to a high degree of accuracy. Physical experiments, using 3D printed wheels and a sand testing bed, were carried out to further validate the scaling relationship. Both four-spoke and "superball" designs were tested and seem to show the general scaling trend expected.

Thesis Supervisor: Ken Kamrin
Title: Assistant Professor of Mechanical Engineering

Acknowledgments

I would like to thank several people who helped me along this journey to complete my undergraduate education and specifically this research. First and foremost, I would like to thank Professor Kamrin who has not only helped guide the direction of this research, but has also served as a invaluable mentor to me. His advice on graduate school, career directions, and life in general have been a great source of insight, especially while navigating this intense education. I am extremely glad that he choose to ask me to work with him after only interacting with me through classwork. I can safely say that this decision has altered my life trajectory for the better.

I would also like to thank Carrington Motley and Carmine Senatore, who were both great sources of help in the lab. Without their work and expertise, none of the experimental results could have been achieved. A special thanks to Carmine for building the entire experimental rig that we used. In addition, I would like to thank Professor Iagnemma for providing the lab space in which we were able to perform all of our experiments. Having a dedicated lab definitely aided our ability to test. Finally, I would like to thank Sachith Dunatunga, whose knowledge of remote computing proved extremely useful while I was studying in Italy, and Amy Guyomard who laid the groundwork for the MATLAB simulations.

Contents

1	Introduction	13
1.1	Resistive Force Theory (RFT)	14
1.1.1	Early Terramechanical Models	14
1.1.2	Stress Plots	14
1.1.3	RFT Scalability	16
1.1.4	Applying RFT	18
1.2	MATLAB Simulation	19
1.2.1	Stress Plot Expansion	19
1.2.2	Simulation Inputs and Initialization	20
1.2.3	Integration of Equations of Motion	21
1.2.4	Average Velocity and Power	23
2	Simulation Applications	25
2.1	Four-Spoke	25
2.1.1	Initial Optimization	26
2.1.2	Changing Length	28
2.1.3	Changing Mass	33
2.1.4	Optimal Design	35
2.2	Superballs	37
2.2.1	Shape Optimization	39
2.3	Global Scaling Relations	41
2.3.1	Dimensional Analysis	41
2.3.2	Defining ζ Using Plasticity	43

2.3.3	Functional Scaling Law	44
2.3.4	Non-Intuitiveness of Scaling Law	45
2.3.5	Simulation Verification	45
2.3.6	Time Based Scaling Law	46
2.4	Triangle	48
2.4.1	Mathematical Derivation	48
2.4.2	Simulation Testing	50
2.4.3	Rotating Optimum	52
2.5	Further Exploration	55
3	Experimental Validation	57
3.1	Experimental Setup	57
3.2	Cylindrical Wheels	58
3.2.1	Experimental Results	59
3.3	4-Bar Wheels	60
3.3.1	Experimental Results	62
3.4	Further Experimentation	65
A	Simulation Code	67
A.1	Four-Spoke Animation Code	67
A.2	Four-Spoke Function (FunctionVODEFourBar)	76
A.3	"Superball" Animation Code	79
A.4	"Superball" Function (FunctionVODEShapesShadow)	88

List of Figures

1-1	Forces in the x and z direction measured for different β and γ . [1] . . .	15
1-2	Stress per unit depth plots in the x and z direction. [1]	16
1-3	Scalable stress per unit depth plots.	17
1-4	"C" shaped leg with linear segments marked darker. [1]	18
1-5	Extended stress per unit depth plots.	20
1-6	Linear interpolation of point where $v_z = 0$	23
2-1	General four-spoke design.	25
2-2	Velocity and Power contour plots.	27
2-3	Velocity and Power contour.	28
2-4	Velocity and Power contour plots for varying lengths.	30
2-5	Contour plots for varying lengths.	32
2-6	Velocity and Power contour plots for varying masses.	34
2-7	Contour plots for varying masses.	35
2-8	Maximum velocity and power tire designs.	36
2-9	"Superball" shapes for different χ values.	38
2-10	"Superball" Velocity contour plot.	39
2-11	"Superball" Power contour plot.	40
2-12	"Superball" Velocity and Power contour plot.	41
2-13	Arbitrarily shaped wheel design.	42
2-14	Position scaling.	47
2-15	Velocity scaling.	48
2-16	Horizontally pulled curve.	49

2-17	Optimal curve.	50
2-18	"Triangle" tread design pulled horizontally.	51
2-19	Force (F_x) and torque (τ) plot.	51
2-20	Force to torque ratio (F_x/τ).	52
2-21	"Triangle" wheel design.	53
2-22	"Triangle" wheel velocity contour plot.	53
2-23	"Triangle" wheel power contour plot.	54
2-24	"Triangle" wheel velocity and power contour plot.	55
3-1	Cylindrical wheels.	59
3-2	Velocity measurements.	59
3-3	Power measurements.	60
3-4	Four-spoke wheels.	61
3-5	Velocity measurements.	64
3-6	Power measurements.	65

List of Tables

- 1.1 Generic Coefficients 17

- 2.1 Effect of Length on Maximum Angle 31
- 2.2 Effect of Mass on Maximum Angle 34
- 2.3 Simulation Inputs 46
- 2.4 Simulation Outputs 46

- 3.1 Cylindrical Wheel Inputs 58
- 3.2 Four-Spoke Wheel Inputs 61
- 3.3 Four-Spoke Masses 62
- 3.4 Four-Spoke Drawbar Forces 62
- 3.5 Small Wheel Results 62
- 3.6 Large Wheel Results 63

Chapter 1

Introduction

The engineering principles of Solid and Fluid Mechanics have been developed over years, such that the interaction of bodies with a solid or fluid can be explained, predicted, and optimized. When an object comes into contact with a solid, the resultant forces can be solved for with knowledge of the material properties and dynamics of the object. Similarly, when a body moves through a fluid, such as a plane moving in air or a submarine moving through water, the lift, drag, and thrust forces can be solved for.

In contrast, modeling bodies as they interact with granular media, such as sand, dirt, and gravel, is a relatively young field. The dynamics of granular materials, which flow once the yield stress is exceeded, is more complicated as they exhibit both solid and fluid properties.[1] The variety of the particle shapes, sizes, and densities makes it difficult to model as each particle interacts with each other through dissipative contact forces. Models, such as granular plasticity, do exist for granular flows, but simulating interactions between bodies and granular media is usually very computationally intensive. Processes, such as the Discrete Element Method (DEM), often require each individual particle to be tracked and accounted for, which can make it very difficult to simulate complex situations.

In 2013, Chen Li et al. proposed a new "terradyamic" model to explain the interaction of bodies with non-cohesive granular materials called Resistive Force Theory.[1] This theory, originally developed to model animal legged locomotion, is not very com-

putationally demanding, but has been validated by numerous experiments.[1] Under the right conditions, Resistive Force Theory can be used for quick and detailed analysis of the interactions that bodies moving through granular media experience. This thesis seeks to expand the application of Resistive Force Theory to examine its use in predicting and optimizing the wheel design of vehicles moving through granular media.

Section 1.1 explains the background and formulation of Resistive Force Theory. Section 1.2 expands upon this foundation by describing the simulation created in MATLAB to apply RFT to wheeled locomotion. Chapter 2 focuses on the optimization routines and scientific principles that were explored through the simulation. Chapter 3 discusses the validation of these ideas, through physical experiments and their results.

1.1 Resistive Force Theory (RFT)

1.1.1 Early Terramechanical Models

Before Resistive Force Theory, two types of terramechanical models were used to simulate locomotion through granular media. The first approximated the object's interaction with the media as a horizontal flat plate.[1] The second assumed that the resistive forces experienced by the object were purely a function of the object's depth and area.[1] Both of these models, though, neglected that the resistive forces also depended on the objects orientation and velocity direction.[1] Therefore, these early models were not able to accurately represent the resistive force profile.

1.1.2 Stress Plots

To improve upon earlier methods, Chen Li et al. first assumed that the net forces on an object could be well approximated by linear superposition of the resistive forces on infinitesimally small segments.[1] This assumption was corroborated by earlier experiments that they had performed.[1] To utilize this linear superposition property,

they first needed to create the stress profile of a flat plate moving through sand in different directions and orientations. To do this, they measured the lift (F_z) and drag (F_x) forces on a plate at different angles of attack (β) and angles of intrusion (γ) as shown in Figure 1-1.[1]

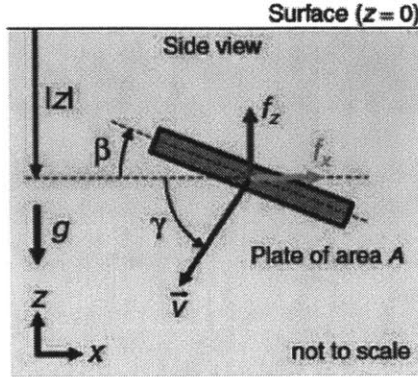


Figure 1-1: Forces in the x and z direction measured for different β and γ . [1]

Varying the angle of attack changed the orientation of the plate, while varying the angle of intrusion changed the direction of the velocity of the plate. Once the forces were measured, the stress could be found by dividing the force by the area of the plate, as shown in equation 1.1.

$$\sigma_{z,x} = \frac{f_{z,x}}{A} \quad (1.1)$$

Finally, since the resistive forces were proportional to the depth of the intruding object, the stresses were modeled per unit depth. The resistive forces depend on the depth because the frictional forces between the granular particles scale with the "hydrostatic-like" pressure.[1] Therefore, the stress was modeled as shown in equation 1.2.

$$\sigma_{z,x}(|z|, \beta, \gamma) = \begin{cases} \alpha_{z,x}(\beta, \gamma)|z| & \text{if } z < 0. \\ 0, & \text{if } z > 0. \end{cases} \quad (1.2)$$

After all the measurements were performed, the resistive stress plots could be

constructed as shown in Figure 1-2.

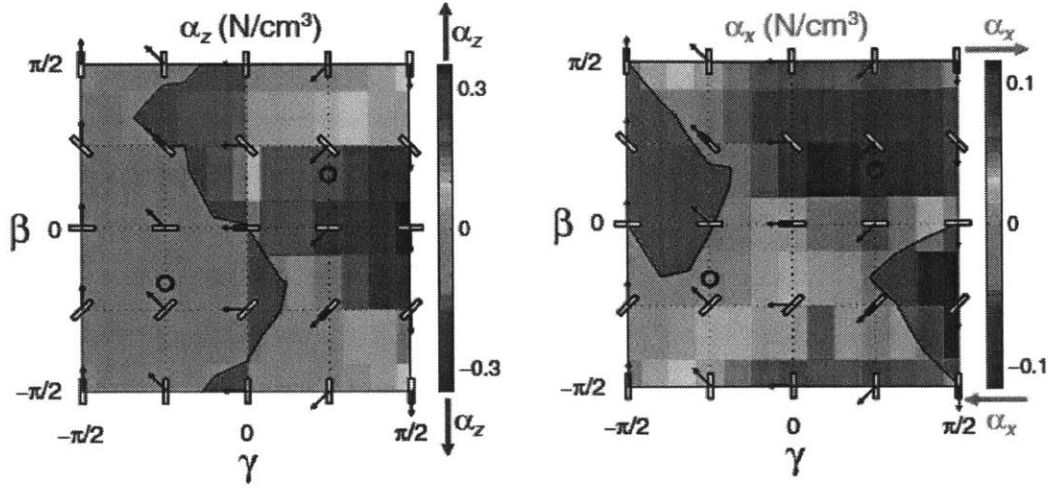


Figure 1-2: Stress per unit depth plots in the x and z direction.[1]

1.1.3 RFT Scalability

Chen Li et al. repeated the flat plate intrusion test with different non-cohesive granular materials and found that the general patterns were very similar, even though the magnitudes differed.[1] Thus, they performed a discrete Fourier transform of the stress profiles to obtain scalable fitting functions, which are shown as equations 1.3 and 1.4.[1]

$$\alpha_z^{fit}(\beta, \gamma) = \sum_{m=-1}^1 \sum_{n=0}^1 [A_{m,n} \cos 2\pi \left(\frac{m\beta}{\pi} + \frac{n\gamma}{2\pi} \right) + B_{m,n} \sin 2\pi \left(\frac{m\beta}{\pi} + \frac{n\gamma}{2\pi} \right)] \quad (1.3)$$

$$\alpha_x^{fit}(\beta, \gamma) = \sum_{m=-1}^1 \sum_{n=0}^1 [C_{m,n} \cos 2\pi \left(\frac{m\beta}{\pi} + \frac{n\gamma}{2\pi} \right) + D_{m,n} \sin 2\pi \left(\frac{m\beta}{\pi} + \frac{n\gamma}{2\pi} \right)] \quad (1.4)$$

The fitting functions have generic coefficients, which can be multiplied by a scaling factor (ζ) to reflect individual materials. The generic coefficients are shown in Table 1.1.

Table 1.1: Generic Coefficients

Generic Coefficient	Value
$A_{0,0}$	0.206
$A_{1,0}$	0.169
$B_{1,1}$	0.212
$B_{0,1}$	0.358
$B_{-1,1}$	0.055
$C_{1,1}$	-0.124
$C_{0,1}$	0.253
$C_{-1,1}$	0.007
$D_{1,0}$	0.088

Using the generic scalable functions, the stress per unit depth plots are recreated in Figure 1-3.

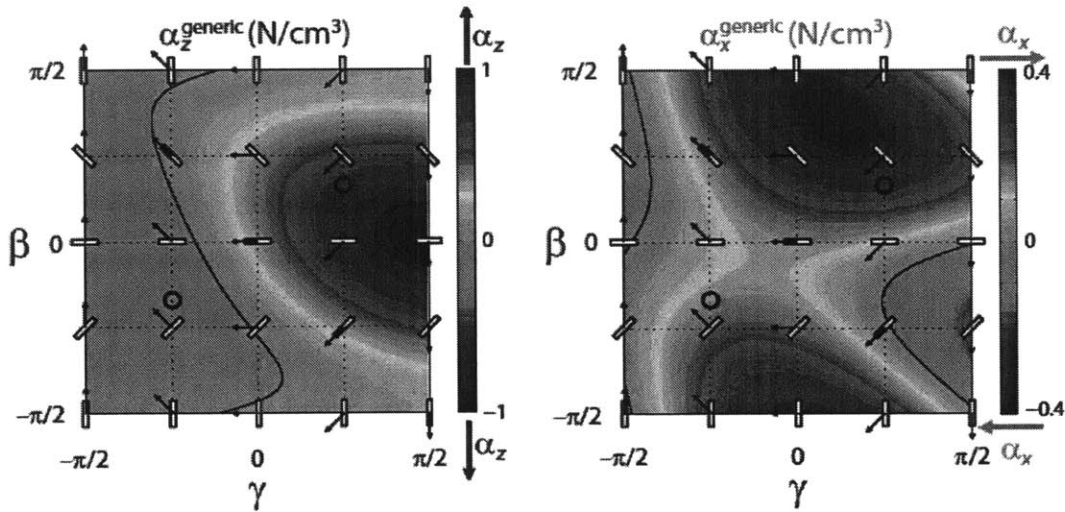


Figure 1-3: Scalable stress per unit depth plots.

This scalable stress profile is extremely useful, as it allows the entire stress profile of a new granular material to be inferred from a single measurement. Chen Li et al. observed that the ratio of the maximum vertical stress from the experimental and fitted data was nearly constant for all materials.[1] The ratio of maximum vertical stresses, which occurs when a plate is oriented horizontally and moved straight downward, can be approximated as χ as shown in equation 1.5.

$$\chi = \alpha_z(0, \pi/2) / \alpha_z^{fit}(0, \pi/2) = 1.26 \pm 0.14 \quad (1.5)$$

Therefore, to find the scaling factor (ζ) for a particular material and thereby the entire stress profile, one only has to measure the stress when the plate is oriented horizontally ($\beta = 0$) and moved straight downward ($\gamma = \pi/2$) as shown in equation 1.6.[1]

$$\zeta = \alpha_z(0, \pi/2) / \chi \approx 0.8\alpha_z(0, \pi/2) \quad (1.6)$$

Once the scaling factor, and therefore the entire resistive force profile, is determined, it can be used to predict the forces experienced by objects interacting with the particular granular material.

1.1.4 Applying RFT

Using the scaled stress plots, the resistive forces acting on an intruding body can be approximated by applying the linear superposition principle. For instance, to estimate the resistive force on the actuating leg shown in Figure 1-4, one can discretize the object into small linear segments that have specific angles of attack, angles of intrusion, depths, and areas and use the stress plot to calculate the force on each segment.[1] The total resistive force on the object is then the sum of the individual forces on each smaller segment.

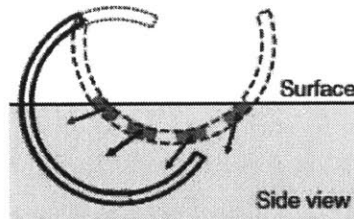


Figure 1-4: "C" shaped leg with linear segments marked darker.[1]

To ensure that the linear superposition assumption holds, Chen Li et al. compared the measured forces on an actual "C" shaped body to the integration of the RFT

predictive forces on the same body as shown in equation 1.7.[1]

$$F_{z,x} = \int_S \alpha_{z,x}(\beta_s, \gamma_s) |z|_s dA_s \quad (1.7)$$

They found the results were a large improvement over the earlier terramechanical models and therefore supported their assumptions about RFT.

1.2 MATLAB Simulation

A MATLAB simulation was created to expand the RFT theory, proposed by Chen Li et al., to apply it to wheeled vehicular locomotion. A few different tire designs were simulated, so this section will only provide an overview of the general principles used. The entire code is available in Appendix A for complete analysis.

1.2.1 Stress Plot Expansion

The first step to applying the RFT framework is to expand the general stress plots. The plots produced in Figure 1-3 only display values for $-\pi/2 \leq \beta \leq \pi/2$ and $-\pi/2 \leq \gamma \leq \pi/2$. This is the full range of β , however the full range of γ extends from $-\pi/2 \leq \gamma \leq 3\pi/2$. Therefore, the plots had to be extended. Using logical symmetrical arguments, for instance that the $\alpha_z(-\pi/4, -\pi/4) = \alpha_z(\pi/4, 5\pi/4)$ and $\alpha_x(-\pi/4, -\pi/4) = -\alpha_x(\pi/4, 5\pi/4)$, it was discovered that the original scalable equations presented in 1.3 and 1.4 were valid over the full range of γ . Thus, the stress plots were reproduced to reflect the full range as shown in Figure 1-5.

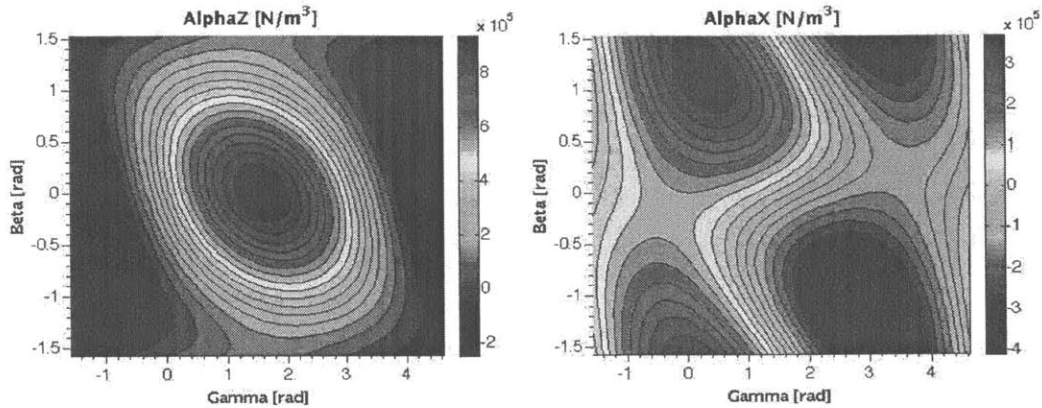


Figure 1-5: Extended stress per unit depth plots.

1.2.2 Simulation Inputs and Initialization

With the full RFT framework available, the simulation could now be created. The first step in the simulation is to define the inputs and constants necessary. In the MATLAB simulation the wheel is assumed to be rotating alone, i.e. with no attached vehicle, at a fixed rotation rate and with a fixed mass. Therefore, the required inputs include the tire mass, fixed rotation speed, initial position and velocity of the wheel, and the gravitational acceleration. Details about the tire design including a characteristic length, the width of the wheel in the plane, and the wheel design parameter (discussed in Chapter 2) are also required. Finally, the scaling factor for the particular granular material and the number of discretized segments are needed. One other input that was later added is the drawback force, which provides a constant force in the x-direction that helps to simulate the tire moving up or down an incline, as it essentially "tilts" the gravity field. With the required inputs, the first step in the simulation is to initialize the design. Based on the wheel design parameters, the starting Cartesian coordinates of the midpoints of each discretized segment are found. Next, the initial β angle values are calculated for each segment.

1.2.3 Integration of Equations of Motion

To simulate the actual rotation and locomotion of the wheel, the equations of motion are integrated using the ode45 solver in MATLAB. The integrated function is built of the form of equation 1.8, where the initial conditions, along with the initialization parameters (*IP*), are input and the output is of the form of the derivative of the initial conditions.

$$\begin{bmatrix} a_x \\ a_z \\ v_x \\ v_z \\ Power \end{bmatrix} = f\left(\begin{bmatrix} v_x \\ v_z \\ x \\ z \\ Energy \end{bmatrix}, IP\right) \quad (1.8)$$

Within the actual function, the inputs are used to create a 7 column index matrix. The first column consists purely of the number assigned to each individual segment. The second and third column correspond to the new x and z coordinates of the rotated tire after some time-step (t). These coordinates are found using the rotation matrix, where the angle the tire is rotated is equal to the fixed rotation rate (ω) multiplied by the time-step. This is shown in equation 1.9, where the i subscript denotes it is after the time-step, while the o subscript denotes it is before the time-step. The values without any subscript reflect the position of the axle, or wheel center, rather than the segment midpoint.

$$\begin{bmatrix} x_i \\ z_i \end{bmatrix} = \begin{bmatrix} x \\ z \end{bmatrix} + \begin{bmatrix} \cos(\omega t) + \sin(\omega t) \\ -\sin(\omega t) + \cos(\omega t) \end{bmatrix} \begin{bmatrix} x_o \\ z_o \end{bmatrix} \quad (1.9)$$

Columns 4 and 5 correspond to the velocities in the x and z direction of the segments after the time-step. These are solved for by calculating and adding the tangential velocity that the fixed rotation of the wheel provides, as shown in equations 1.10 and 1.11.

$$v_{x,i} = \omega(z_i - z) + v_x \quad (1.10)$$

$$v_{z,i} = -\omega(x_i - x) + v_z \quad (1.11)$$

Column 6 corresponds to the new β values, which are calculated by adding the rotation of the tire (ωt) to the original value. Some manipulation is required to keep the β values in the range of $-\pi/2 \leq \beta \leq \pi/2$ as shown in equation 1.12.

$$\beta_i = \begin{cases} \beta_o + \omega t & \text{if } \beta_o + \omega t \leq \pi/2. \\ \beta_o + \omega t - \pi & \text{if } \beta_o + \omega t > \pi/2. \end{cases} \quad (1.12)$$

Finally, column 7 corresponds to the new γ values which are calculated by taking the arctangent of the velocity in the z direction over the velocity in the x direction. Similar to the β calculation, the code is set up to ensure that the γ stays in the range $-\pi/2 \leq \gamma \leq 3\pi/2$ as shown in equation 1.13.

$$\gamma_i = \begin{cases} \arctan\left(\frac{v_{z,i}}{v_{x,i}}\right) & \text{if } v_{x,i} < 0. \\ \arctan\left(\frac{v_{z,i}}{v_{x,i}}\right) + \pi & \text{if } v_{x,i} \geq 0. \end{cases} \quad (1.13)$$

Using this index of values, the function then calculates the stress per unit depth acting on each segment by plugging in the angles of attack and intrusion into the scalable RFT formula from equations 1.3 and 1.4. Next, this stress per unit depth in the x and z direction is converted to the forces in each direction using the segment length (L) and the tire width (W), as shown in equations 1.14 and 1.15. The code also ensures that the force is set to zero if the z -position of the segment is above the surface of the granular material as there is no resistive force in that case.

$$f_z = -\alpha_z LW z_i \quad (1.14)$$

$$f_x = -\alpha_x LW z_i \quad (1.15)$$

Using the position vector of the midpoint of each segment and crossing it with the

forces obtained, gives the torque that acts on each segment. The total torque and forces in both directions acting on the entire wheel can then be found by summing the individual elements acting on each segment. Finally, the outputs are solved for as shown in equation 1.16, where DB is the drawback force, m is the tire mass, τ is the total torque, and all other parameters are defined above.

$$\begin{bmatrix} a_x \\ a_z \\ v_x \\ v_z \\ Power \end{bmatrix} = \begin{bmatrix} \frac{F_{x,tot}-DB}{m} \\ \frac{F_{z,tot}-mg}{m} \\ v_x \\ v_z \\ \omega\tau \end{bmatrix} \quad (1.16)$$

1.2.4 Average Velocity and Power

Once the function is integrated by the ode45 solver over the simulation duration, an array of velocities (v_x, v_z) , positions (x, z) , and energies dissipated (E) at each time-step (t) is output. With these outputs, the average velocity in the x-direction $(v_{x,avg})$ and the average power expended (P_{avg}) can be calculated. As the wheel translates, it eventually reaches a steady state, where it oscillates with a constant period and amplitude. Using this property, the time-steps where the velocity in the z direction (v_z) change from negative to positive are found. These represent instances where a new period of oscillation is beginning. These time-steps are not right at the point when $v_z = 0$, though, so the values at that point need to be linearly interpolated, as shown in Figure 1-6.

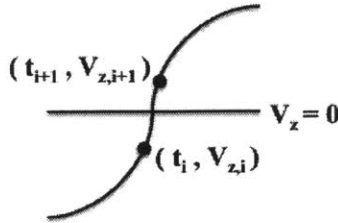


Figure 1-6: Linear interpolation of point where $v_z = 0$.

To do so, first the local slope (m_{v_z}) between the points $(t_i, v_{z,i})$ and $(t_{i+1}, v_{z,i+1})$ is calculated. This is done as shown in equation 1.17.

$$m_{v_z} = \frac{v_{z,i+1} - v_{z,i}}{t_{i+1} - t_i} \quad (1.17)$$

This process is repeated to find the local slope between the same two points for the (t, x) plot, m_x , and (t, E) plot, m_E , as well. Using these slopes, the time, t_o , at the point where $v_z = 0$ is found as shown in equation 1.18.

$$t_o = \frac{-v_{z,i}}{m_{v_z}} + t_i \quad (1.18)$$

Next, the x-position, x_o , and the energy dissipated, E_o , at that instance can be interpolated as shown in equations 1.19 and 1.20.

$$x_o = m_x * (t_o - t_i) + x_i \quad (1.19)$$

$$E_o = m_E * (t_o - t_i) + E_i \quad (1.20)$$

Using these interpolated values at each instance where $v_z = 0$, $v_{x,avg}$ and P_{avg} can be calculated. The averages are calculated over arbitrarily chosen steady state period numbers, shown here as u and v , that are high enough to ensure the wheel has reached steady state, as shown in equations 1.21 and 1.22.

$$v_{x,avg} = \frac{x_{o,v} - x_{o,u}}{t_{o,v} - t_{o,u}} \quad (1.21)$$

$$P_{avg} = \frac{E_{o,v} - E_{o,u}}{t_{o,v} - t_{o,u}} \quad (1.22)$$

Chapter 2

Simulation Applications

Using the basic simulation framework described in Chapter 1, different wheel shapes and designs could be simulated. As mentioned, the designs must be such that they have a single wheel design parameter. The various designs used in this research will be discussed in the sections below.

2.1 Four-Spoke

The first wheel design simulated was a simple four-spoke one, where each spoke is assumed to have a hinge halfway down its length. The hinge can be bent at any angle θ , as shown in Figure 2-1.

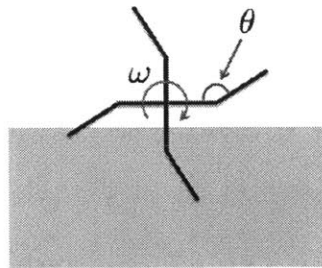


Figure 2-1: General four-spoke design.

The dimensionless wheel design parameter in this design is the spoke bend angle

θ . The characteristic input length is defined as half the total spoke length, or the distance from the axle to the bend. Using this design, various angles, lengths, masses, and rotation speeds were simulated to attempt to find an optimal and least optimal angle.

2.1.1 Initial Optimization

The initial optimization routine involved simulating over a range of angles (θ) and rotational velocities (ω), while keeping the tread length, mass, and other inputs constant. Using MATLAB's parallel computation ability, the $v_{x,avg}$ and P_{avg} were calculated for every wheel in this design space. With this data, average power and velocity contour plots could be constructed. The first simulation tested over the range of angles from $45^\circ \leq \theta \leq 315^\circ$ and rotational velocities from $3 \leq \omega \leq 7.5$ radians per second. In reality, RFT is only expected to be accurate over the range of angles $90^\circ \leq \theta \leq 270^\circ$ because as the two bars approach 0° or 360° , RFT will assume that the force on the bar is double, when in reality the two bars are in essence becoming a single one. Therefore, RFT is not expected to be valid for acute angles, however the range was extended for instructive purposes.

For this simulation, the length was set to 0.25 meters, the mass to 68 kilograms, and there was assumed to be no drawback force. The contour plots for the x-direction velocity and power are shown in Figure 2-2. The black stars marked on the contour plots represent the angles at which the velocity and power are maximized for each rotational speed. Conversely, the red stars represent the angles at which the velocity and power are minimized.

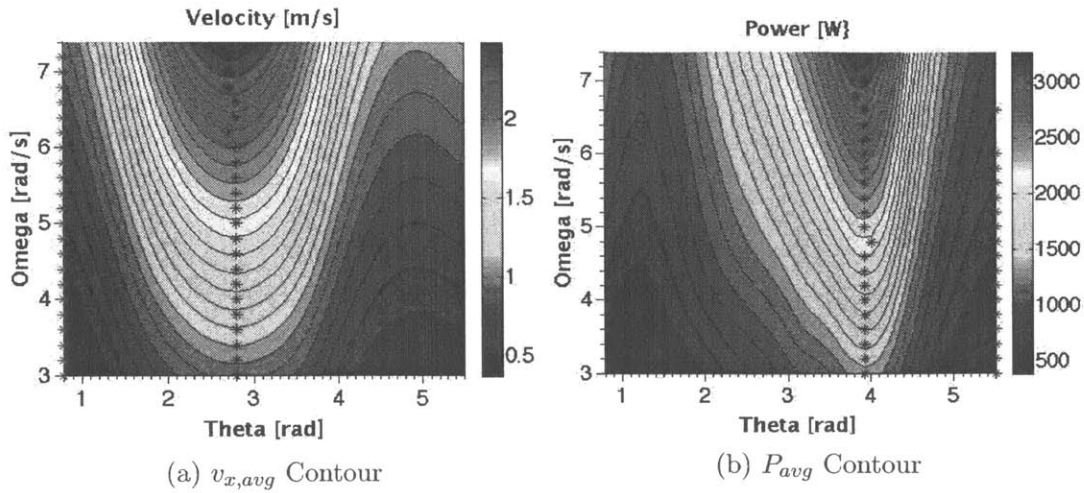


Figure 2-2: Velocity and Power contour plots.

It was found that designs with an angle less than 180° were able to generate the highest velocity. Conversely, designs with an angle greater than 180° required the most power. To determine the optimal angle that minimizes power for a given velocity v_x , both contours are plotted. The optima, defined as the points in which the gradients of the contour plots are parallel, occur at local tangency points of the contour plots. These were found in the code by taking the cross product of the gradients of each plot. Whenever the cross product was below a certain threshold (close to zero), it was approximated as a point of tangency and a point (shown as a blue dot) was plotted. This is shown in Figure 2-3.

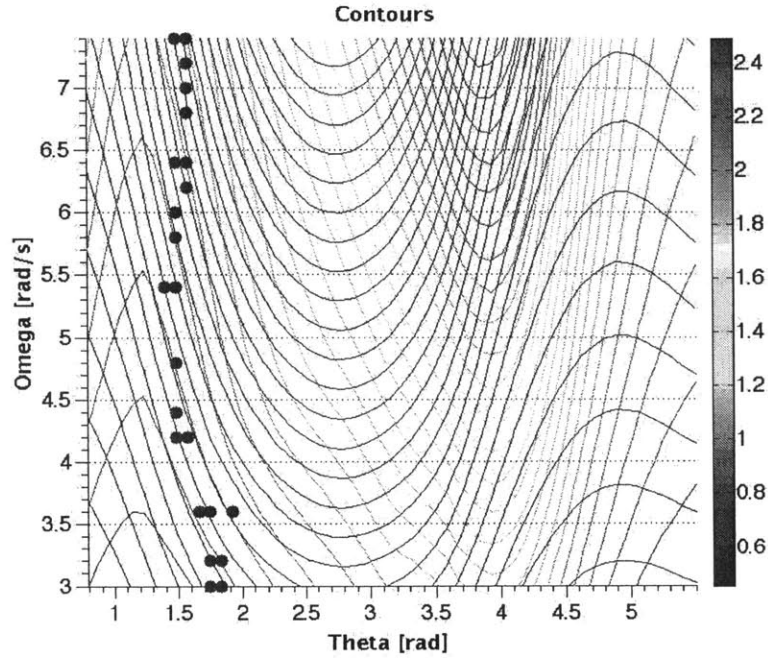
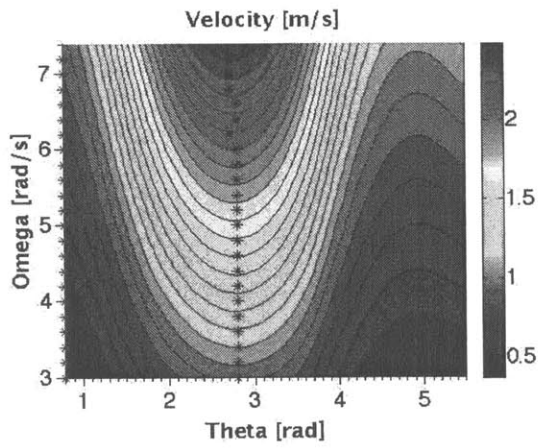


Figure 2-3: Velocity and Power contour.

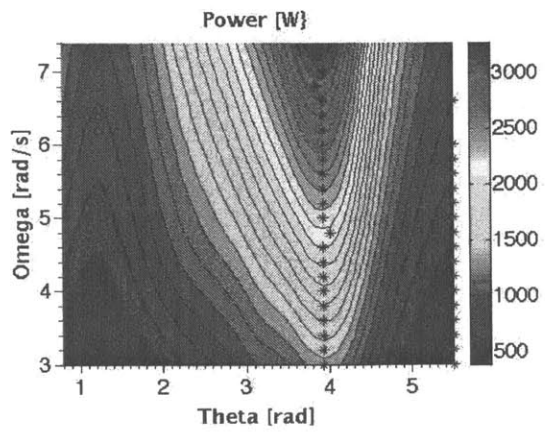
The optima occur around 86° , which is not within the ideal range of RFT usage. This plot is still very useful, though, as one can follow a given velocity contour and manually find at which angle the power is minimized.

2.1.2 Changing Length

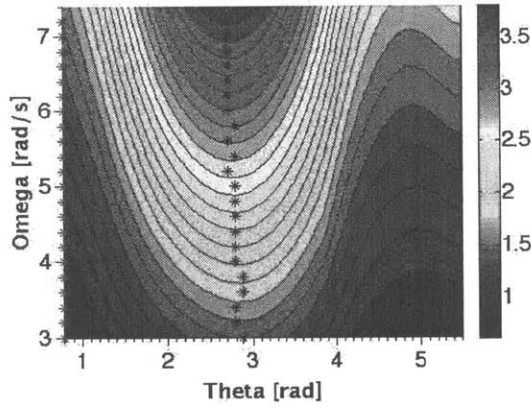
After the initial optimization, further exploration was performed to determine if changing the tread length had any effect on the optimal wheel design. Using the same range of angles and rotational velocities, the simulation was performed again for different lengths varying from 0.25m to 1m. The mass (68kg) and all other inputs remained constant. The velocity and power contour plots for each case are shown in Figure 2-4.



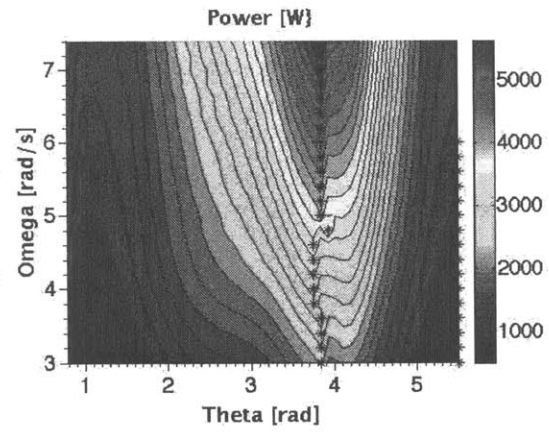
(a) Length=0.25m
Velocity [m/s]



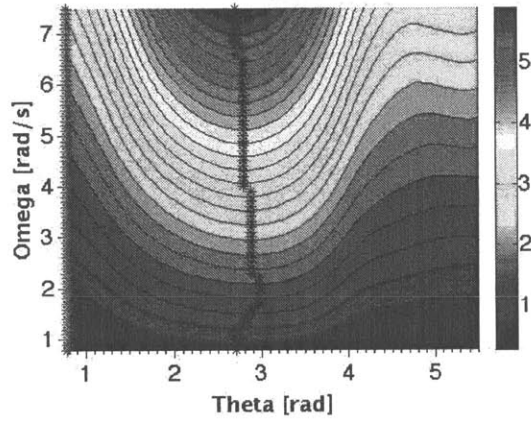
(b) Length=0.25m



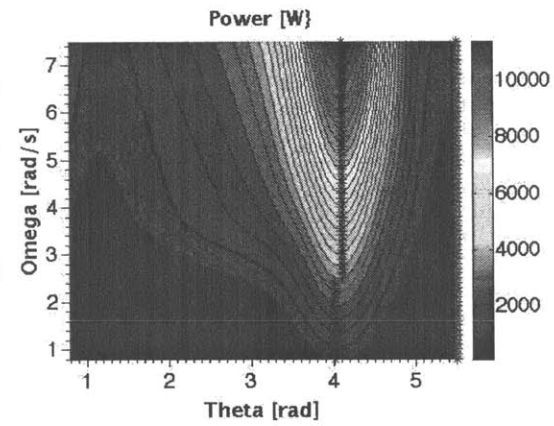
(c) Length=0.35m
Velocity [m/s]



(d) Length=0.35m



(e) Length=0.5m



(f) Length=0.5m

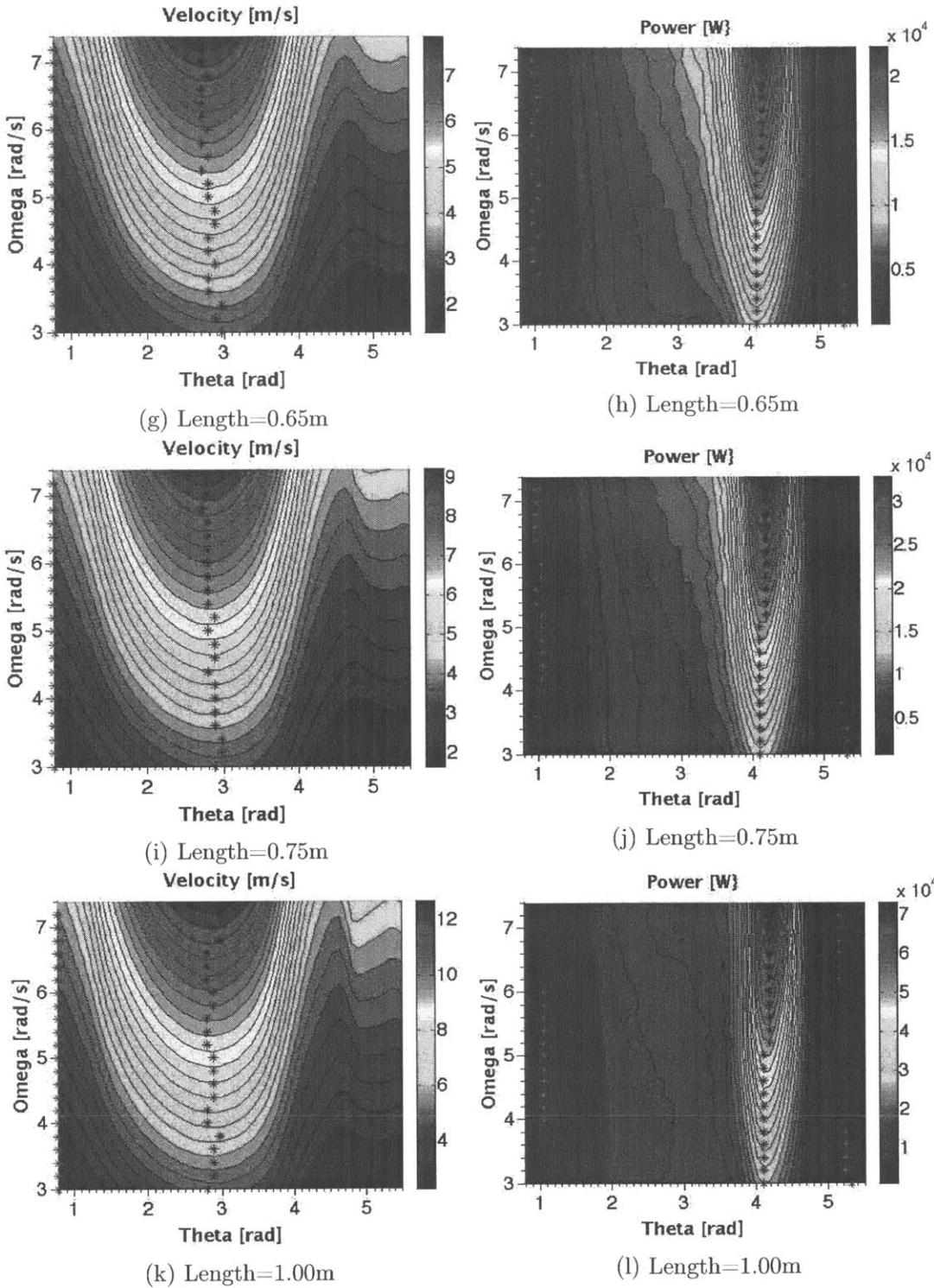


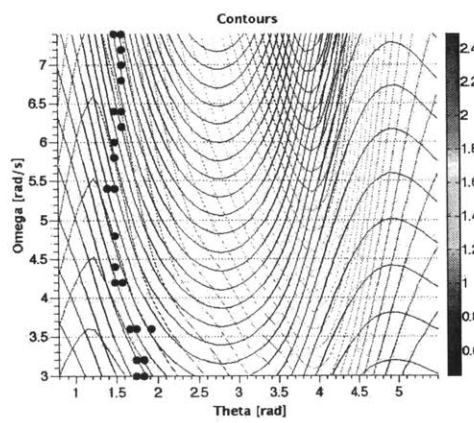
Figure 2-4: Velocity and Power contour plots for varying lengths.

The average of the angles that produced the maximum, within the range of RFT validity, are taken for each case. The results are shown in Table 2.1.

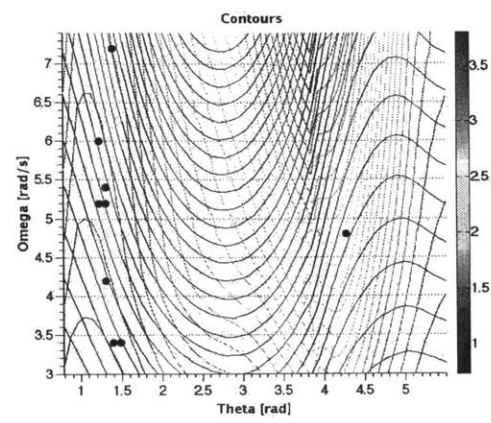
Table 2.1: Effect of Length on Maximum Angle

Length	Maximum Velocity	Maximum Power
0.25m	158.91°	224.57°
0.35m	158.48°	219.13°
0.5m	161.25°	233.75°
0.65m	160.22°	237.17°
0.75m	162.39°	237.61°
1.00m	163.70°	237.61°
Overall	160.83°	231.64°

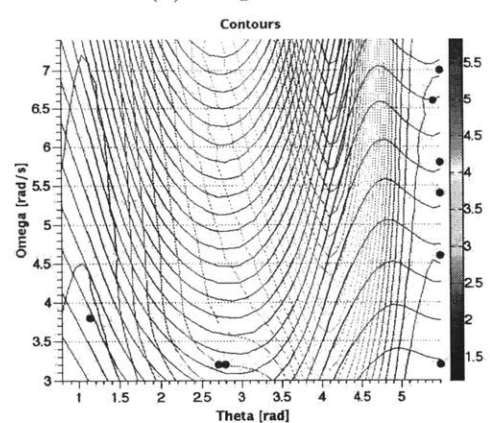
Increasing the length seems to slightly increase the angle that produces the maximum velocity and power, however it is clear the designs with $\theta < 180^\circ$ produce the highest velocity and designs with $\theta > 180^\circ$ require the highest power. The contour plots for all cases are plotted together in Figure 2-5.



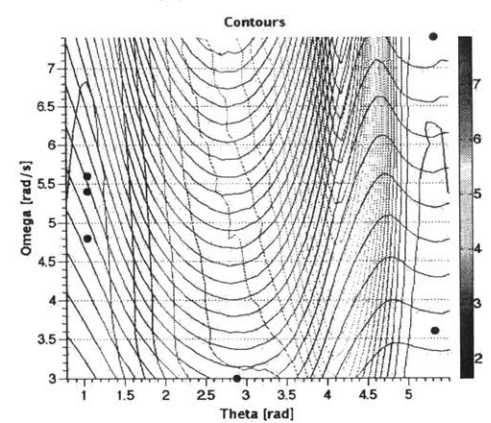
(a) Length=0.25m



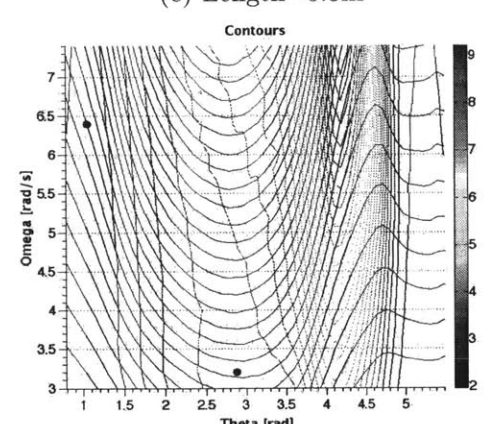
(b) Length=0.35m



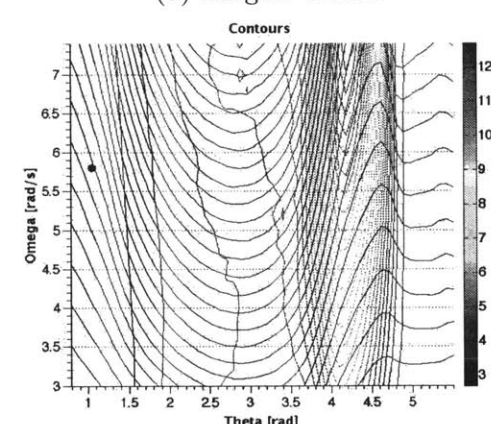
(c) Length=0.5m



(d) Length=0.65m



(e) Length=0.75m

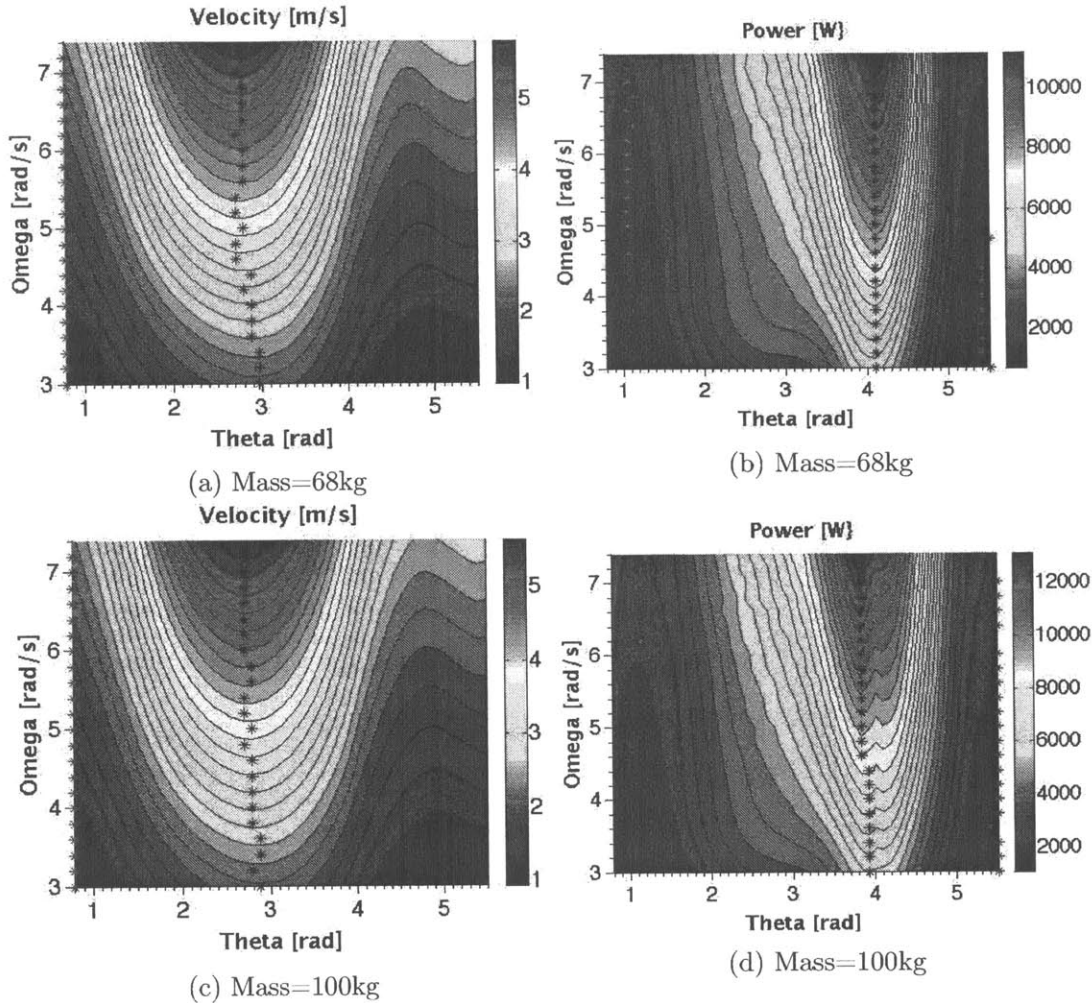


(f) Length=1.00m

Figure 2-5: Contour plots for varying lengths.

2.1.3 Changing Mass

The same type of experiment was performed to determine the effect of changing the mass. In this case, the same angle and rotational velocity ranges were used, while the length was kept constant at 0.5m. The mass was varied from 68kg to 200kg. Again, the velocity and power contour plots are shown in Figure 2-6.



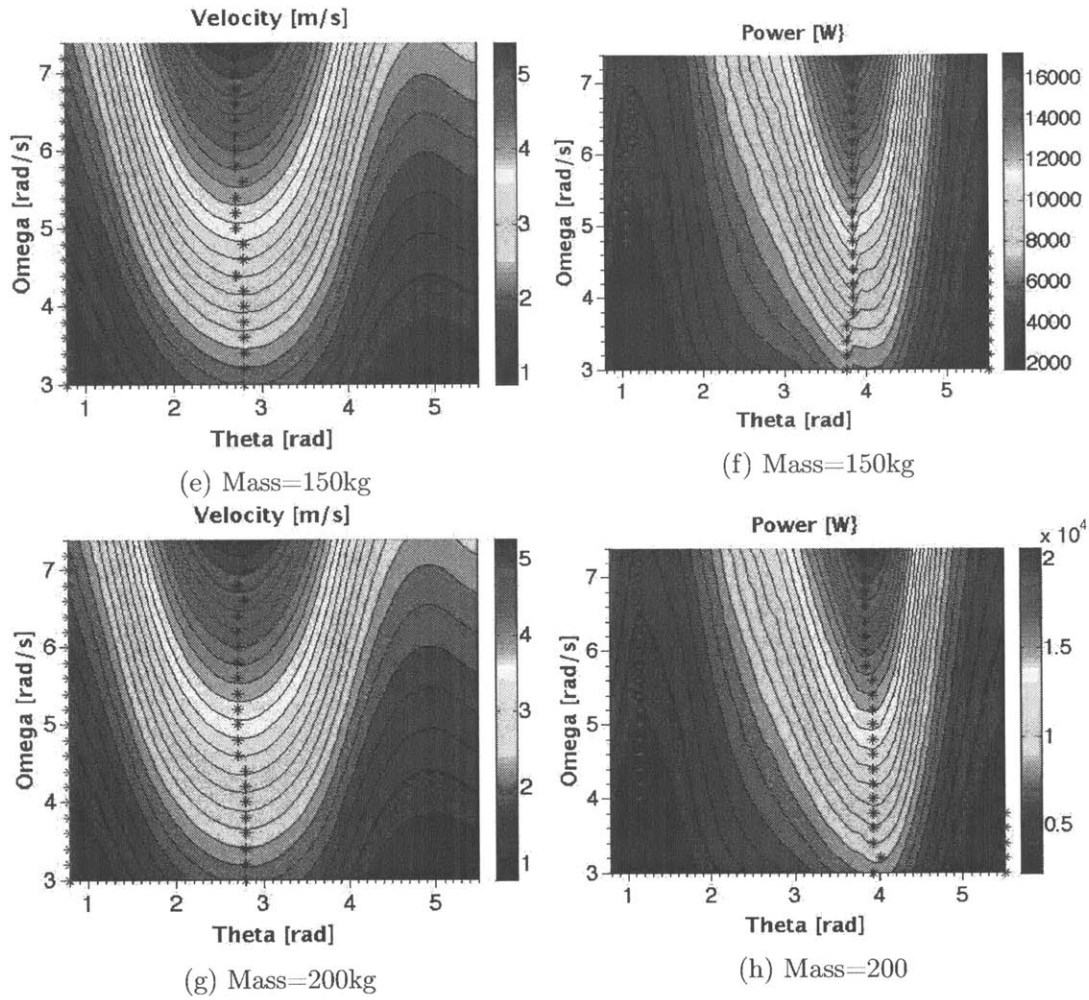


Figure 2-6: Velocity and Power contour plots for varying masses.

The average of the angles that produced the maximum for each case are shown in Table 2.2.

Table 2.2: Effect of Mass on Maximum Angle

Mass	Maximum Velocity	Maximum Power
68kg	160.87°	235°
100kg	158.48°	221.74°
150kg	157.17°	219.13°
200kg	157.39°	223.26°
Overall	158.48°	224.78°

Changing the mass does not appear to drastically change the maximum angle. It also seems to confirm the idea that designs with $\theta < 180^\circ$ achieve greater velocities and designs with $\theta > 180^\circ$ require the highest powers. The contour plots for all cases are again plotted together in Figure 2-7.

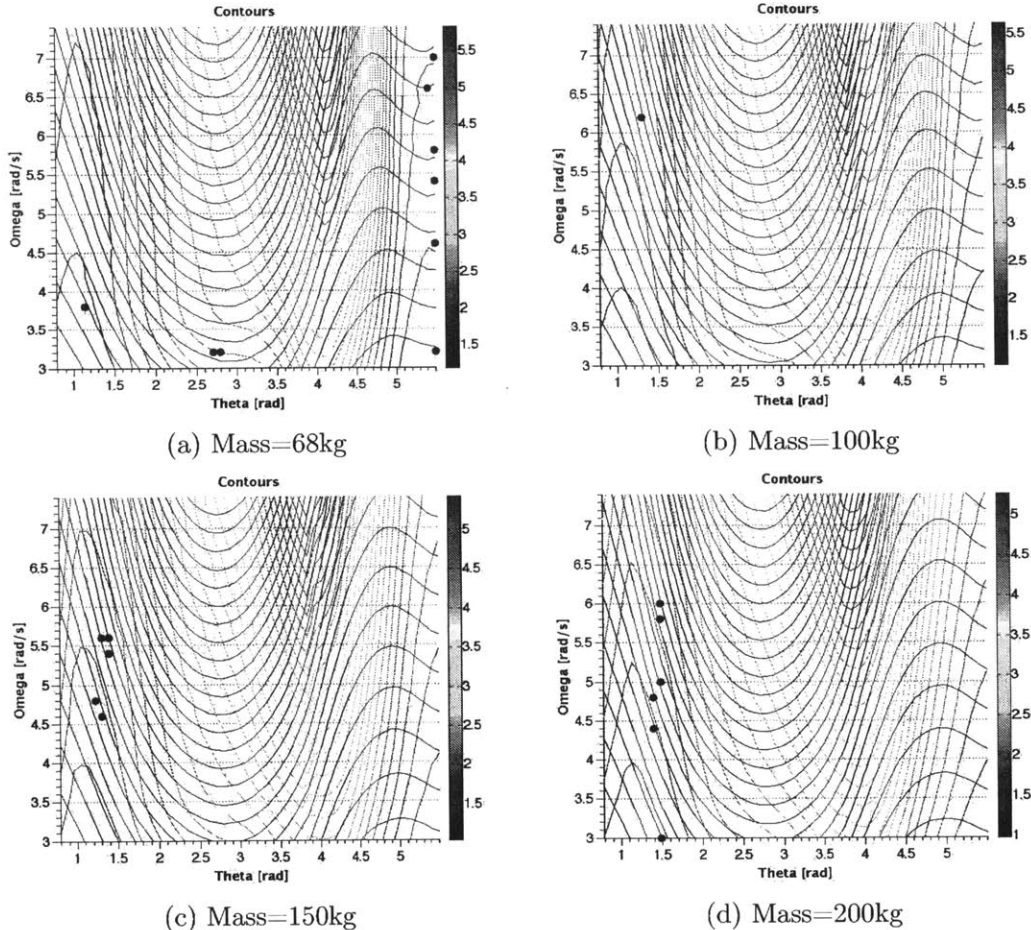


Figure 2-7: Contour plots for varying masses.

2.1.4 Optimal Design

In the design space for which RFT is expected to be physically valid, $90^\circ \leq \theta \leq 270^\circ$, it is clear from the simulations that regardless of length, mass, or rotational velocity, the optimal design occurs when $\theta < 180^\circ$. These wheels are able to generate

higher velocities while expending less power for given rotational velocities. Conversely, designs with $\theta > 180^\circ$ produce lower velocities and require higher power. The wheel design that produces the maximum velocity is at approximately $\theta = 160^\circ$, while the design that requires the maximum power is at approximately $\theta = 230^\circ$. Both of these designs, assumed to be rotating clockwise, are shown in Figure 2-8.

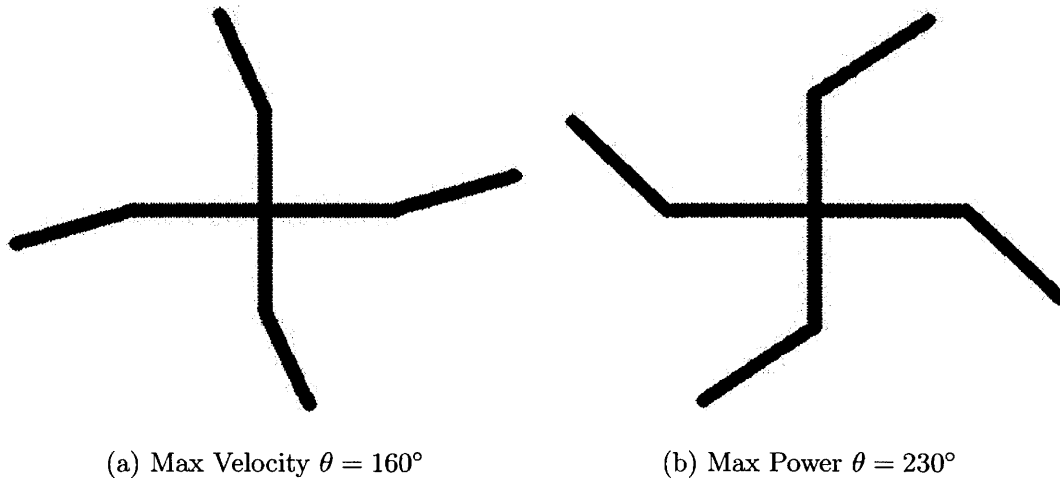


Figure 2-8: Maximum velocity and power tire designs.

There are several proposed reasonings for this effect. Chen Li et al. tested with "C" leg designs, like that seen in Figure 1-4, as well as ones that rotated the other direction and "clawed" into the sand.[1] What they found is that the "claw" design was much less efficient because it not only reached greater depth into the sand, which caused higher resistive forces as the forces are proportional to depth, but also because it reached higher stress regions of the RFT plot.[1] The four-spoke design with $\theta > 180^\circ$ is believed to produce a similar effect, as it claws into the sand and expends more power overcoming higher resistive forces, so it is not able to reach as high a velocity.

Conversely, wheels with $\theta < 180^\circ$ drive more efficiently. This effect is believed to be due to the fact that these designs better distribute forces, which reduces the magnitude of stresses that act on them. This is very similar to the practice of reducing the air in one's tire when driving a car on the beach. By increasing the flexibility

of one's tire, the forces can be better distributed to reduce the peak stress. These designs, and likewise a deflated tire, do not reach as large depths because of this better force distribution, which allows them to face smaller resistive forces. Therefore, they require less power to overcome these forces and can reach higher velocities.

2.2 Superballs

Expanding upon the simple four-spoke design, the next wheel geometry tested in the simulation were "superballs." The shape of the "superball" wheels are defined by the shape parameter χ , as shown in equation 2.1.

$$\left|\frac{x}{R}\right|^{2\chi} + \left|\frac{z}{R}\right|^{2\chi} = 1 \quad (2.1)$$

When $\chi = 1$, the equation becomes that of a circle with radius R . As χ decreases to 0.5, the shape becomes a square. As it decreases more, the sides of the square become concave. Conversely, when $\chi \rightarrow \infty$ the shape approaches that of a square again. The actual wheels are the "superball" shapes with a fixed width into the plane. A plot with the shapes of different χ values plotted is shown in Figure 2-9.

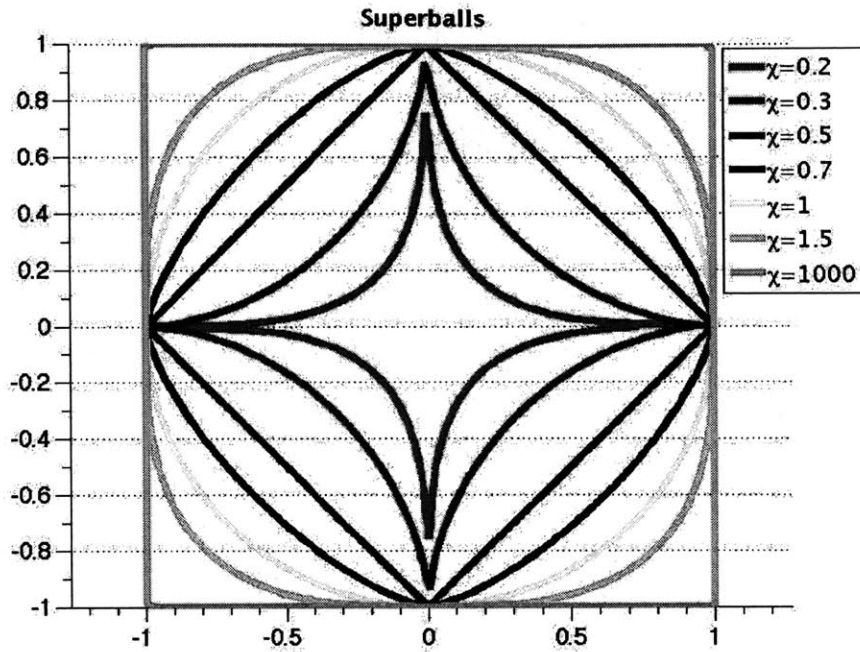


Figure 2-9: "Superball" shapes for different χ values.

The characteristic input length for this tire design is the effective radius R , used in equation 2.1. With this input geometry, a similar process as that used for the four-spoke design could be used. Other than replacing the angle θ with the shape parameter χ and replacing the tread length L with the effective radius R , one more step was needed to utilize the simulation.

Unlike the four-spoke design, where resistive forces from the granular media could act on both sides of the tread, the superball shapes are assumed to be filled in solid wheels. Therefore, the simulation needed to be altered slightly so that resistive forces act only on the outside of the wheel, and not within it. This was achieved by calculating the outward normal vector of each discretized segment of the wheel shape. Within the integrated function, the dot product of the outward normal vector and the velocity vector was taken at each time-step. If that dot product was negative, meaning the outward normal vector was in the direction opposite that of motion, then any resistive force on that segment was assumed to be zero. Essentially, this

assumed that any surface that was moving away from the sand, rather than pushing into it, does not generate any resistive forces.

2.2.1 Shape Optimization

Similar to the four spoke design, a range of rotational velocities (ω) and shape parameters (χ) were simulated to find the average velocity and power. The rotational velocities were tested over the range of $3 \leq \omega \leq 7.5$ radians per second, while the shape parameters were tested over the range from $0.3 \leq \chi \leq 3$. The results are shown in Figures 2-10 and 2-11.

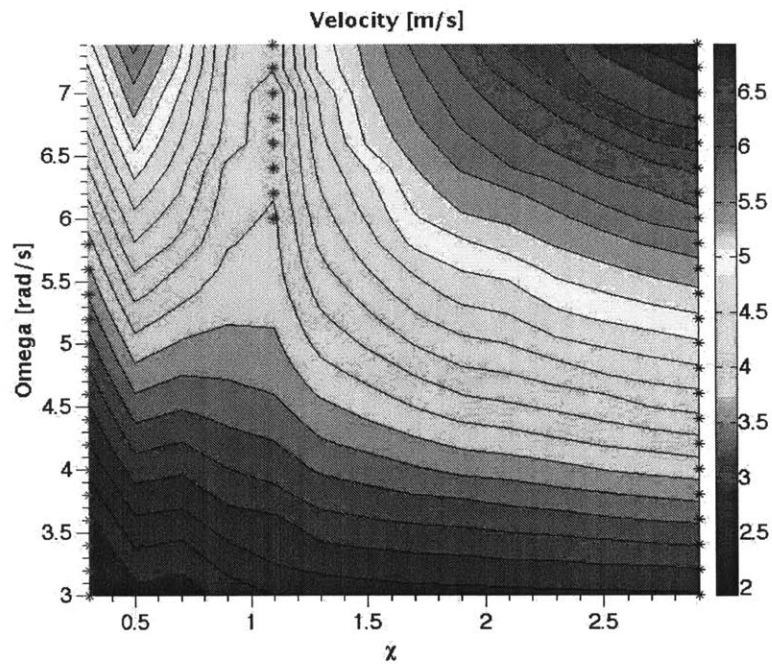


Figure 2-10: "Superball" Velocity contour plot.

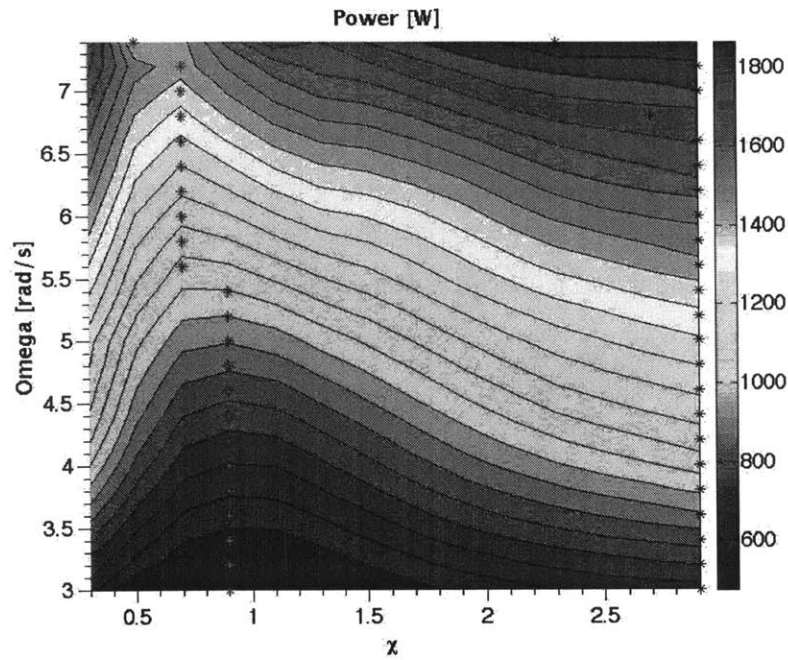


Figure 2-11: "Superball" Power contour plot.

The simulation indicates, that at high rotational speeds the minimum velocity occurs when $\chi = 1.1$, while the maximum power occurs when $\chi = 0.7$. Conversely, at low rotational speeds, the minimum velocity occurs when $\chi = 0.3$ and the maximum power occurs at $\chi = 0.9$. Further analysis is required to determine why these optima occur where they do, however it is clear that unlike the four spoke design, the optima do seem to change as the rotational velocity changes. Both contours are shown in Figure 2-12.

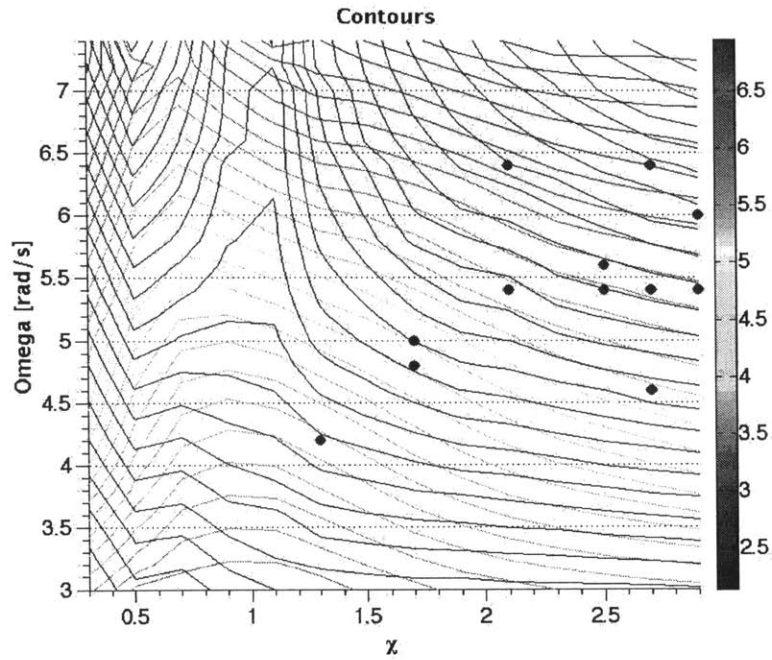


Figure 2-12: "Superball" Velocity and Power contour plot.

2.3 Global Scaling Relations

Building upon these previous simulation results, we believed that the simplicity of RFT might allow for the derivation of global scaling laws. Finding ways to model both the velocity (V_x) and power (P) of a wheel design by testing with a completely different one could be incredibly useful in future terradynamic applications.

2.3.1 Dimensional Analysis

First, to perform dimensional analysis of the system the inputs have to be defined. Rather than using any specific shape design, an arbitrarily shaped wheel, such as that shown in Figure 2-13, is used.

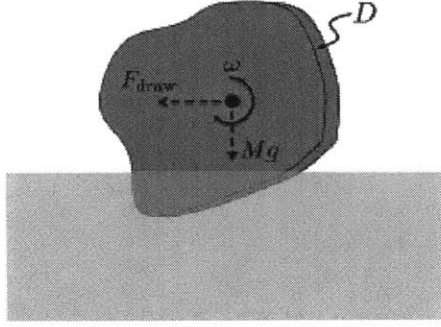


Figure 2-13: Arbitrarily shaped wheel design.

The wheel boundary is defined by the polar set $r = Lf(\theta)$, where L is the characteristic wheel length and f the function which gives the wheel design parameter. The wheel is assumed to have a mass m and rotate at some constant rotational velocity ω . The wheel has an out of plane thickness defined by D and is acted upon by a gravitational acceleration g . There is assumed to be a horizontal drawback force F_{draw} acting on the axle. Finally, the type of granular material the wheel is moving through is defined by the RFT scaling constant ζ . Due to the nature of RFT, the dependence on ζ and D arises from the product ζD , which can be simplified into a single variable. Using these inputs, the average horizontal velocity and power can be written as functions ψ_v and ψ_p as shown in equations 2.2 and 2.3.

$$v_{avg} = \psi_v(\omega, L, f, m, g, F_{draw}, \zeta D) \quad (2.2)$$

$$P_{avg} = \psi_p(\omega, L, f, m, g, F_{draw}, \zeta D) \quad (2.3)$$

Next, non-dimensional groups can be written using L , m , and w as the characteristic length, mass, and time units. The groups are defined in equations 2.4, 2.5, and 2.6.

$$\tilde{g} = \frac{g}{L\omega^2} \quad (2.4)$$

$$F_{draw}^{\sim} = \frac{mg}{F_{draw}} \quad (2.5)$$

$$\zeta^{\sim} D = \frac{\zeta DL^2}{mg} \quad (2.6)$$

This leads to the global scaling forms shown in equations 2.7 and 2.8.

$$v_{avg} = L\omega\tilde{\psi}_v(f, \frac{g}{L\omega^2}, \frac{mg}{F_{draw}}, \frac{\zeta DL^2}{mg}) \quad (2.7)$$

$$P_{avg} = mL^2\omega^3\tilde{\psi}_p(f, \frac{g}{L\omega^2}, \frac{mg}{F_{draw}}, \frac{\zeta DL^2}{mg}) \quad (2.8)$$

2.3.2 Defining ζ Using Plasticity

The scaling constant ζ could be further expanded, though, using Plasticity theory. Within Plasticity, it is understood that the only external properties that affect the force of a body intruding into a granular material is the density of the sand ρ , the gravitational acceleration g , the coefficient of friction of the sand μ , and the coefficient of sand material interface μ_w . Therefore, the RFT scaling constant ζ could be rewritten in these terms. Using dimensional analysis, the relation shown in equation 2.9 was derived.

$$\zeta = \rho g \hat{\zeta}(\mu, \mu_w) \quad (2.9)$$

Therefore, the global scaling relations can be rewritten as shown in equations 2.10 and 2.11.

$$v_{avg} = L\omega\tilde{\psi}_v(f, \frac{g}{L\omega^2}, \frac{mg}{F_{draw}}, \frac{\rho\hat{\zeta}(\mu, \mu_w)DL^2}{m}) \quad (2.10)$$

$$P_{avg} = mL^2\omega^3\tilde{\psi}_p(f, \frac{g}{L\omega^2}, \frac{mg}{F_{draw}}, \frac{\rho\hat{\zeta}(\mu, \mu_w)DL^2}{m}) \quad (2.11)$$

2.3.3 Functional Scaling Law

Using the scaling relations defined in equations 2.10 and 2.11 a functional scaling law can be found. First, to reduce the complexity of the law, the shape and type of sand are assumed to be constant. This means the wheel shape parameter (f) and the sand type ($\hat{\zeta}(\mu, \mu_w)$ and ρ) remain constant, resulting in a three degree of freedom system. We therefore introduce three scaling factors A , B , and C as defined in equation 2.12.

$$\bar{g} = Ag, \bar{L} = BL, \bar{m} = Cm \quad (2.12)$$

Utilizing the global scaling relations, we can solve for the scale factors for the other non-constant inputs as shown in equation 2.13.

$$\bar{\omega} = \sqrt{\frac{A}{B}}\omega, \bar{F}_{draw} = CAF_{draw}, \bar{D} = \frac{C}{B^2}D \quad (2.13)$$

Inputting these scale factors back into the global scaling relations, we can derive the functional scaling laws as shown in equations 2.14 and 2.15.

$$\begin{aligned} v_{avg}(\omega, L, f, m, g, F_{draw}, \hat{\zeta}(\mu, \mu_w), D, \rho) = \\ \frac{1}{\sqrt{AB}}v_{avg}\left(\sqrt{\frac{A}{B}}\omega, BL, f, Cm, Ag, CAF_{draw}, \hat{\zeta}(\mu, \mu_w), \frac{C}{B^2}D, \rho\right) \end{aligned} \quad (2.14)$$

$$\begin{aligned} P_{avg}(\omega, L, f, m, g, F_{draw}, \hat{\zeta}(\mu, \mu_w), D, \rho) = \\ \frac{1}{CA\sqrt{AB}}P_{avg}\left(\sqrt{\frac{A}{B}}\omega, BL, f, Cm, Ag, CAF_{draw}, \hat{\zeta}(\mu, \mu_w), \frac{C}{B^2}D, \rho\right) \end{aligned} \quad (2.15)$$

This means for instance, to test how a wheel design would behave on Mars, one could test a different design on Earth as long as the scaling law is fulfilled. So, to simulate changes in the gravitational acceleration, one needs only to change accordingly the other inputs ω , L , m , F_{draw} , and D .

2.3.4 Non-Intuitiveness of Scaling Law

As a first check of the scaling laws derived in equations 2.14 and 2.15, the case where a cylindrical wheel is rolling without slipping on a solid surface ($\zeta \rightarrow \infty$) can be used. In this case, one would expect the velocity of both cases to scale purely with the length over the time scale. Therefore, multiplying the length BL by the rotational velocity $\sqrt{\frac{A}{B}}\omega$ gives an expected scaling of \sqrt{AB} , which is what is found. The power, in the rolling without slipping case, would be 0 for both tires, so the scaling law again holds.

Similarly, in the case of a non-round wheel rolling without slipping on a solid surface, one would expect that the velocity would scale the same as the cylindrical wheel. The power, though, would be expected to scale with the torque, defined as the length BL multiplied by the mass Cm and gravity Ag , multiplied by the rotational velocity $\sqrt{\frac{A}{B}}\omega$. Therefore, the expected power scaling would be $CA\sqrt{AB}$, which is exactly what is found.

Although these scaling relations are valid in both these cases, they are not intuitive. For instance, one would expect the velocity to scale with ω and L , but it is not expected that changing the mass, gravity, and thickness of the wheel would have any effect. This relationship between the mass m and the thickness D is especially interesting, because it seems to come directly from RFT. This scaling could not be found using only Plasticity.

2.3.5 Simulation Verification

This scaling law was tested within the simulation. Using the four-spoke design, one wheel was assumed to be on mars, with the other on Earth. All three scaling factors A, B , and C were varied. In this case, $A = 2.644$, $B = 5$, and $C = 8$. The inputs for both tests are shown in Table 2.3.

Table 2.3: Simulation Inputs

Input	Mars Wheel	Input	Earth Wheel
ω	$25^\circ/sec$	$\sqrt{\frac{A}{B}}\omega$	$18.18^\circ/sec$
L	0.1m	BL	0.5m
f	150°	f	150°
m	50kg	Cm	400kg
g	$3.71m/s^2$	Ag	$9.81m/s^2$
F_{draw}	0.5N	CAF_{draw}	10.58N
$\zeta(\mu, \mu_w)$	0.2626	$\zeta(\mu, \mu_w)$	0.2626
D	0.2m	$\frac{C}{B^2}D$	0.064m
ρ	1	ρ	1

The velocity and power results for each test, along with the expected velocity and power is shown in Table 2.4.

Table 2.4: Simulation Outputs

Mars Output	Mars Wheel	Earth Output	Expected Earth Output	Earth Wheel
v_x	$0.07331m/s$	$\sqrt{AB}v_x$	$0.26655m/s$	$0.26655m/s$
P	$7.91W$	$CA\sqrt{ABP}$	$608.59W$	$608.64W$

The results match exactly what the scaling law would predict up to three significant figures for the power and up to five significant figures for the velocity.

2.3.6 Time Based Scaling Law

This scaling effect can not only be applied to the average output velocity and power, but it can also be used to find a parameter at any instant of time. This means, with the velocity, position, and time at every step of two scalable designs, one can alter the time and multiply by a scaling factor to replicate the entire time dependent trajectory. From the dimensional analysis, we know that the time scales with the rotational velocity ω , such that the same scale factor applied to ω can be applied to the time t . This effect is the same for the position, which is dimensionally similar to the length L and scales as such. The velocity as well scales with the length

component divided by the time. Therefore, the position and velocity trajectories can be written as shown in equations 2.16 and 2.17, where the two wheels are defined by the subscripts m and e , which refers to the Mars and Earth scaling example in the previous section.

$$x_m(t_m) = \frac{1}{B}x_e\left(\sqrt{\frac{A}{B}}t_e\right) \quad (2.16)$$

$$v_m(t_m) = \frac{1}{\sqrt{AB}}v_e\left(\sqrt{\frac{A}{B}}t_e\right) \quad (2.17)$$

Using the same data that was used for the scaling of the Mars and Earth wheel, the position and velocity over time were plotted. Then, using the time based scaling, the Earth's wheel position and velocity trajectories were scaled to match the Mars'. The position and velocity in the z-direction are shown in Figures 2-14 and 2-15, however the x-direction has the same effect.

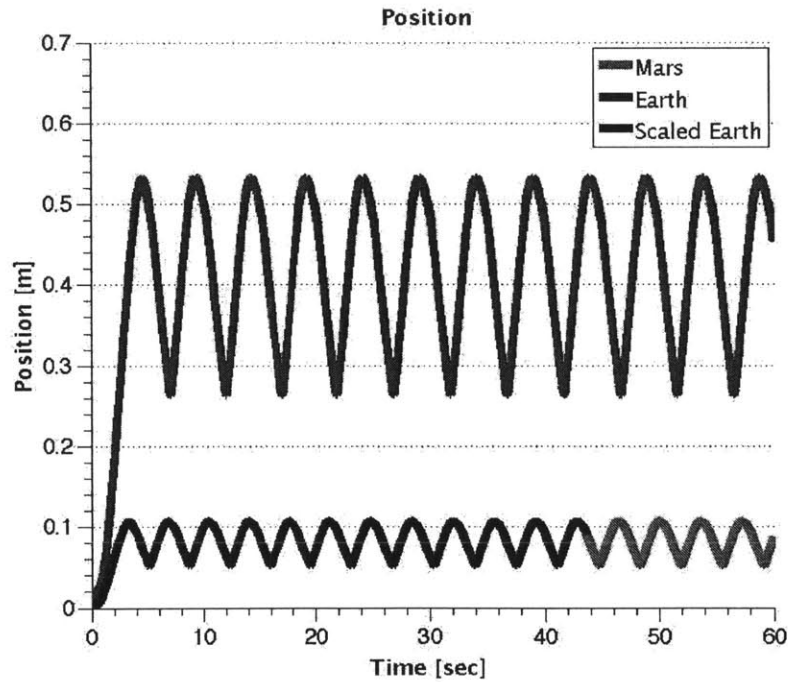


Figure 2-14: Position scaling.

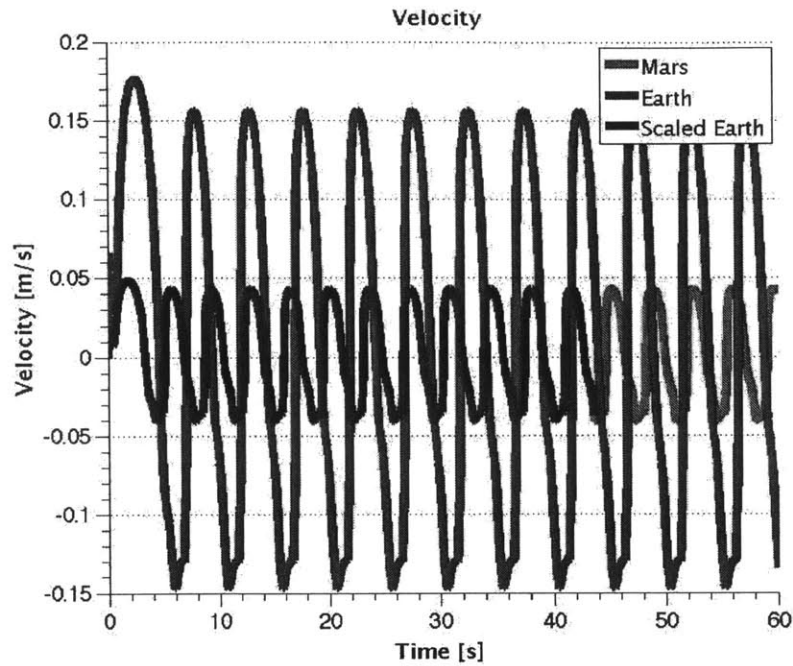


Figure 2-15: Velocity scaling.

As can be seen from the plots, the scaled trajectories match almost exactly the other wheels output, validating the time based scaling law.

2.4 Triangle

2.4.1 Mathematical Derivation

Another experiment that was performed was to determine what the optimal curve $\psi(z)$, with height H , would be if it were to be completely submerged in a granular material and pulled horizontally, as shown in Figure 2-16.

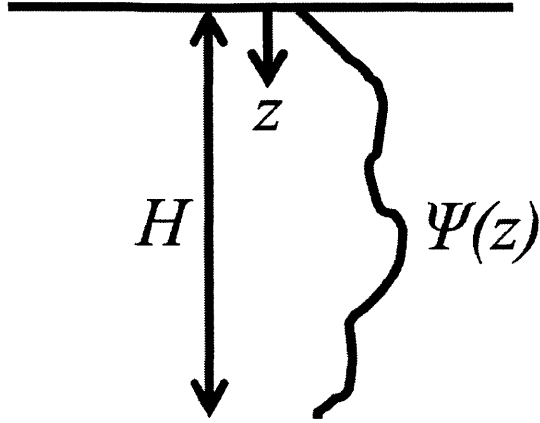


Figure 2-16: Horizontally pulled curve.

To optimal was defined as the curve that was able to minimize the torque, around the top "axle", acting on it while subject to the resistive force of the sand in the x-direction. Since this curve is pulled horizontally, the angle of intrusion is defined as $\gamma = 0$. Therefore, the force and torque acting on the curve are as defined in equations 2.18 and 2.19, where z is the depth and all other variables are defined above.

$$F_x = \int_0^H \alpha_x(\beta) Dz dz \quad (2.18)$$

$$\tau = \int_0^H \alpha_x(\beta) Dz^2 dz \quad (2.19)$$

To find the angle of attack β , equation 2.20 is used.

$$\beta = \pi - \arctan \frac{d\psi}{dz} \quad (2.20)$$

Then, plugging the equation for β into the RFT force plots, variational calculus can be used to derive what the optimal curve is. The Euler-Lagrange equation was calculated, as shown in equation 2.21, where $F = M - \lambda F_x$.

$$\frac{dF}{dz} - \frac{d}{dz} \left(F - \psi' \frac{dF}{d\psi'} \right) = 0 \quad (2.21)$$

This was a very complex equation, so Mathematica was utilized to solve for the curve $\psi(z)$. Additionally, several assumptions had to be made. The optimal curve found is shown in Figure 2-17.

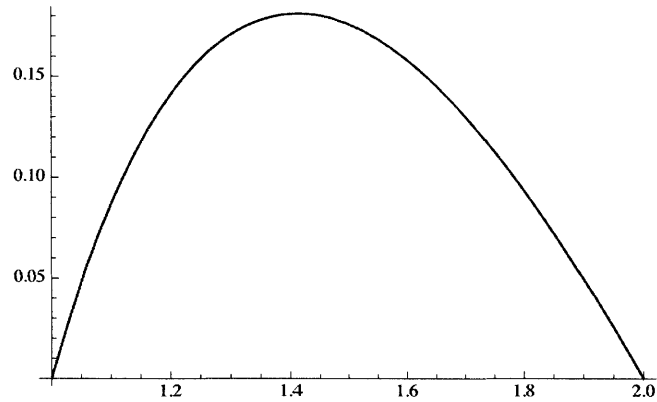


Figure 2-17: Optimal curve.

The result found was unexpected as the curve was not perfectly flat, but rather had some width to it (about 15% of the height). Additionally, the vertex of the curve did not occur at half the height, but rather slightly above the midpoint of the height (around 40% of the height).

2.4.2 Simulation Testing

To further validate this result, a simple triangle design was tested in the simulation. The design consisted of a single tread, essentially two bars that met at a certain point, submerged into the granular material and pulled horizontally. This shape is shown in Figure 2-18.

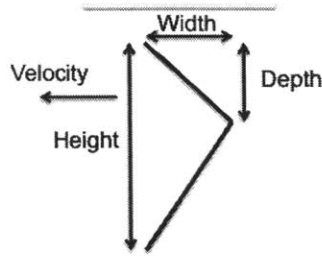


Figure 2-18: "Triangle" tread design pulled horizontally.

The depth was varied from 0.01 to 0.99 of the height. For each design, the total force in the x-direction (F_x) acting on the three bars and the total torque (τ), defined as the sum of the force in the x direction on each discretized segment of the tread multiplied by the tread's z-position relative to the top of the design, was calculated. Afterwards, the total force (F_x), torque (τ), and force to torque ratio (F_x/τ) was plotted against the depth. The results are shown in Figures 2-19 and 2-20.

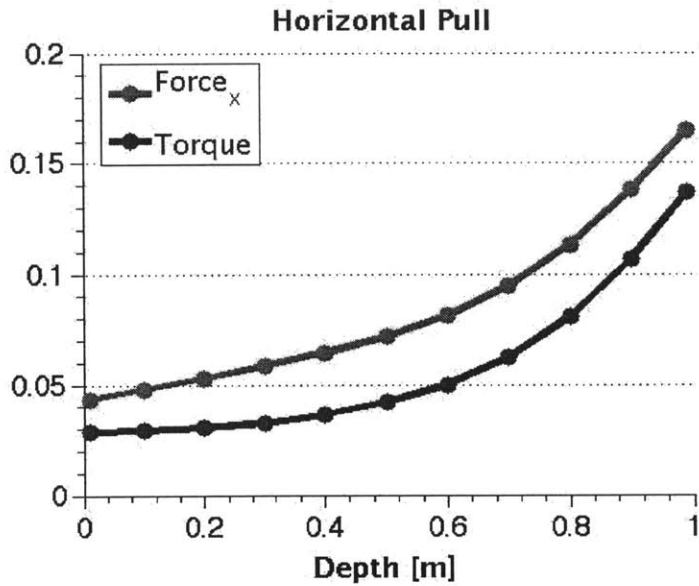


Figure 2-19: Force (F_x) and torque (τ) plot.

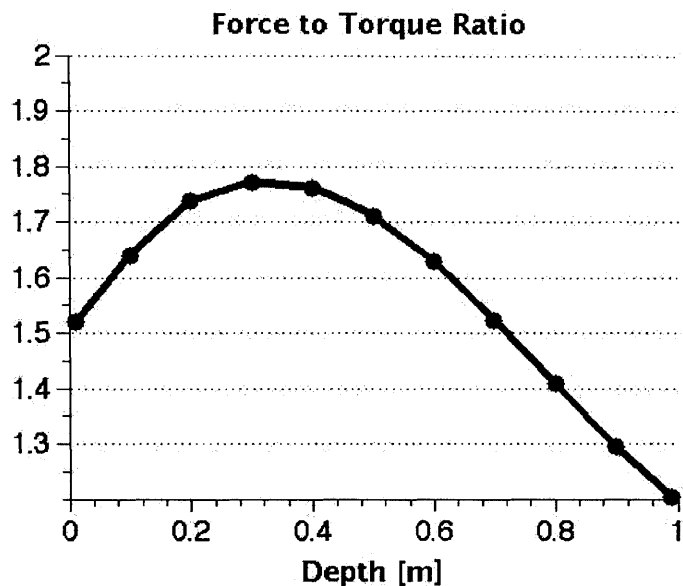


Figure 2-20: Force to torque ratio (F_x/τ).

As you can see, the optimal is again at a point above the center of $depth = 0.5$. The true optimum occurs at $depth = 0.3$. At that point, the force to torque ratio is maximized. This confirmed the mathematical work done in Mathematica.

2.4.3 Rotating Optimum

Expanding upon the simple horizontally pulled triangle, a new wheel design was utilized. This new "triangle" wheel was essentially four of the simple triangle designs put together as treads and rotated. The treads again consist of two bars that attach at an angle. The depth of the point at which the two bars meet is varied, while the width and height is kept constant. This four-spoke design, shown in Figure 2-21, was chosen to ensure that the spokes do not interfere with the resistive force that the next spoke incurs as it enters the granular material.

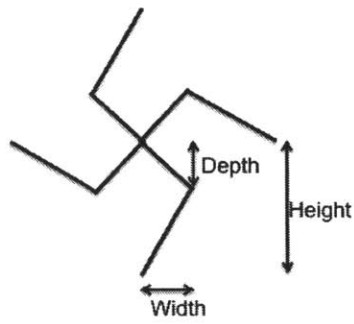


Figure 2-21: "Triangle" wheel design.

The velocity and power contour plots produced are shown in Figures 2-22 and 2-23.

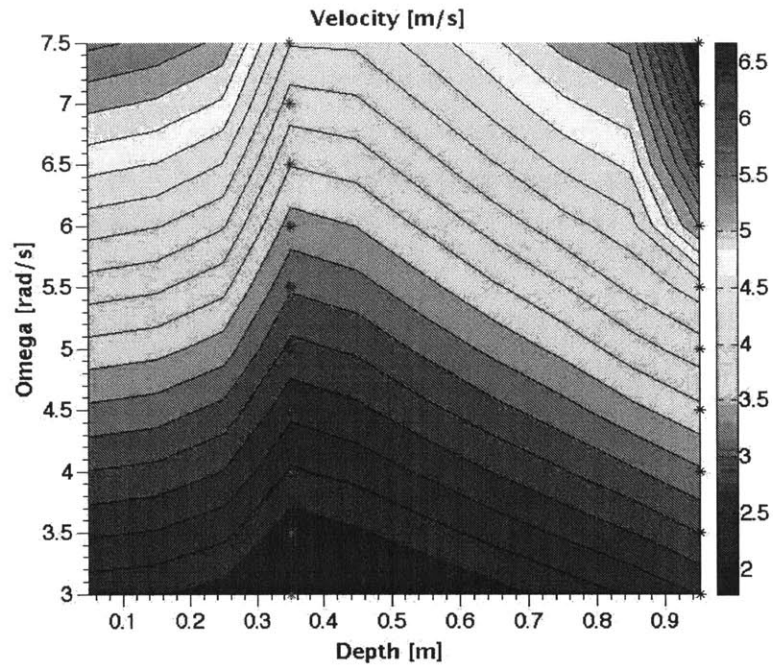


Figure 2-22: "Triangle" wheel velocity contour plot.

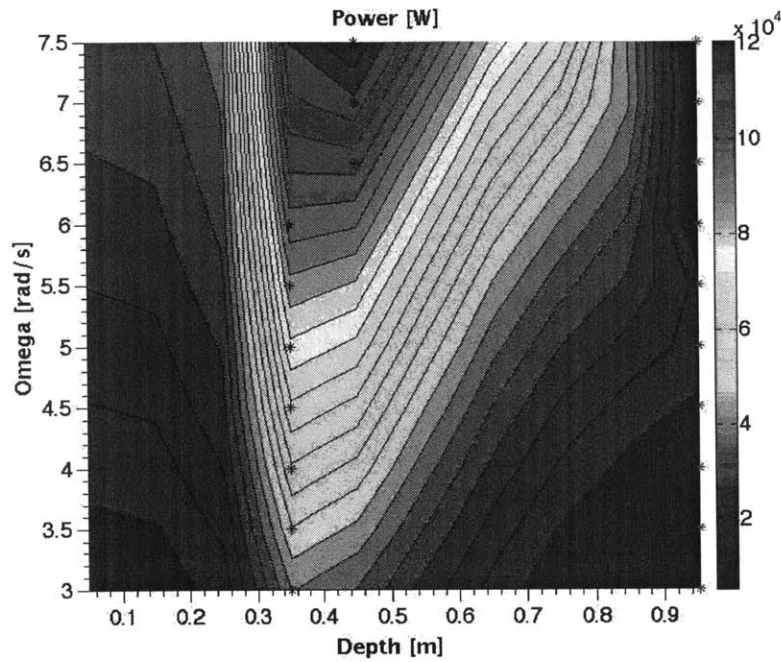


Figure 2-23: "Triangle" wheel power contour plot.

The results show that the velocity is minimized at a depth of 0.35m, while the power is maximized at about 0.4m. Therefore, designs with a depth slightly above the midpoint are actually the least optimal, as they travel the slowest and require the most power. Some proposed reasonings for this effect is that when the tire is rotating, only a portion of the tread is actually submerged in the sand. Therefore the experiment with the single triangle fully submerged may not be an accurate representation. Further analysis is required to determine whether this result will change as the mass of the tire varies or whether it is a global result. Both plots are graphed in Figure 2-24.

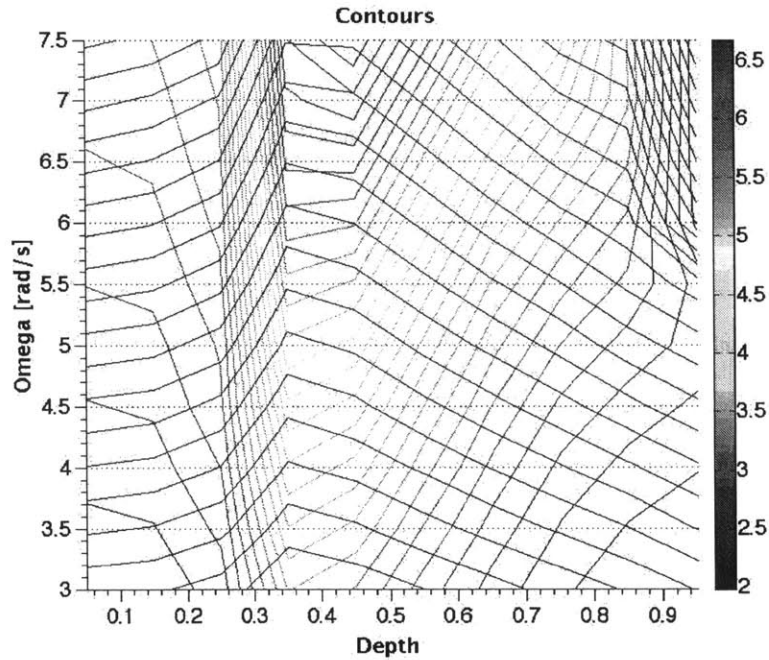


Figure 2-24: "Triangle" wheel velocity and power contour plot.

2.5 Further Exploration

Expanding upon these simulation results, more complex wheel designs will be tested. For instance, a combination of the four-spoke and "superball" designs, essentially solid wheels with treads protruding out, can be simulated. Additionally, higher tread numbers can be used. Further analysis on more complex designs, with greater than one degree of freedom also present an interesting challenge. Alternatively, another design space that requires further research is the effect of tank treads, rather than rotating wheels. With tank treads, the segments move horizontally through the sand, which may provide a unique perspective on the optimal way to move through granular media.

Other than new wheel designs, there is also further work to be done on the effect of greater discretization. For instance, studying whether the simulation results become more accurate as the segments are made smaller and smaller. Similarly, further research needs to be done on different methods to find the optimal designs quickly.

Perhaps, nonlinear optimization routines can be used, which can allow for the optimal design to be found at any instant. This can be extremely useful, especially for potential "smart" tires, which are further described in the next chapter.

Chapter 3

Experimental Validation

To further validate the scaling relationships derived in section 2.3.3, physical experiments were performed.

3.1 Experimental Setup

The experimental set up used was the sand test bed in Professor Karl Iagnemma's Robotic Mobility Laboratory. The test apparatus consisted of a motor driven wheel, which was attached to a torque sensor and load cell. The load cell was used to measure the effective mass of the whole assembly on the sand and was attached to the central platform. The central platform had bearings that connected it to vertical rods, which were attached to the top platform. The rod-bearing system allowed the tire to move in the vertical direction, while the entire system was actuated horizontally through another horizontal rod-bearing system that was attached to the stationary structure. To reduce the system weight a 15 pound spring was connected from the central platform to the top platform.

Connected to the central platform were displacement and velocity sensors for both the x and z directions. The top platform had a connect for a rope-pulley system, which allowed masses to be used as drawback forces. The wheels were designed in SolidWorks and printed in Polylactide (PLA) plastic using a MakerBot 3D printer. The wheels were tested in Quikrete sand. The entire system was controlled through

a LabView program.

3.2 Cylindrical Wheels

The first set of wheels tested were two scaled cylindrical wheels, which were lined with sand paper for traction. The wheels were defined by the "superball" equation with the wheel design parameter $\chi = 1$. For this experiment, the scaling constants of $A = 1$, $B = 1.55$, and $C = 2.4$ were used due to experimental constraints. The mass and drawbar forces were varied over the scaled space to populate a range of velocity and power values. The small wheel was tested under vertical loads of 41, 54, and 66.67N and drawbar forces ranging from -30N to 12N. The large wheel was tested under vertical loads of 100, 130, and 160N and drawbar forces ranging from -50N to 30N (In this case drawback force is defined as positive when it is pulling in the direction opposite that the tire is moving in). The rest of the inputs, and the appropriate scale values, are shown in Table 3.1.

Table 3.1: Cylindrical Wheel Inputs

Small Wheel Scale	Small Wheel	Large Wheel Scale	Large Wheel
ω	$30^\circ/sec$	$\sqrt{\frac{A}{B}}\omega$	$24.1^\circ/sec$
R	8.08 cm	BL	12.5 cm
f	$\chi = 1$	f	$\chi = 1$
g	$9.81m/s^2$	Ag	$9.81m/s^2$
D	15 cm	$\frac{C}{B^2}D$	15 cm

The two wheels tested are shown in Figure 3-1.

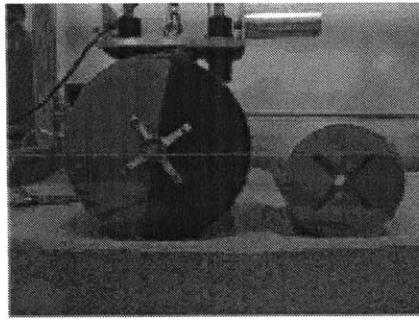


Figure 3-1: Cylindrical wheels.

3.2.1 Experimental Results

Using the sensor outputs, the average velocity and power were measured for each test. The results, along with RFT prediction lines, are shown in Figures 3-2 and 3-3.

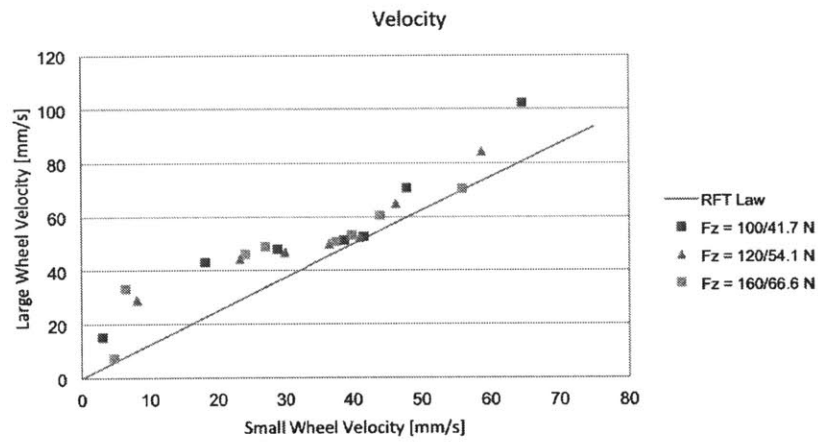


Figure 3-2: Velocity measurements.

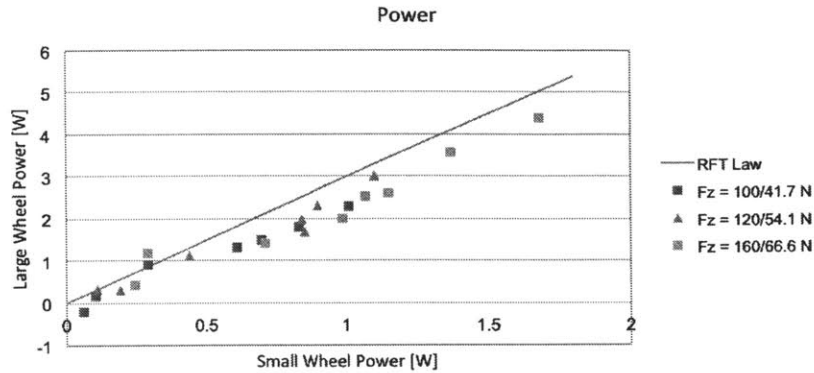


Figure 3-3: Power measurements.

The results seem to indicate the RFT trend prediction, however all of the measured velocities were a little high and the powers were a little low. There are a few proposed reasons for these discrepancies. First, the exact drawbar forces were not always achieved and were in fact always slightly below the intended value. Second, there was measurable friction between the bearings and vertical rods, which affected the downward force the wheel exerted on the sand. Essentially, the mass was not always constant as a result of the friction. Finally, it was later realized that while the spring did remove downward force, it did not change the inertial mass. This means the scaled masses were all slightly off, which again introduced some error into the experiment.

3.3 4-Bar Wheels

To further validate the RFT scaling predictions, four-spoke wheels were printed and tested. The four spoke design was expected to better follow the RFT trend, as it better simulated the conditions under which RFT was based. That is its design closely mirrors the flat plate intrusion tests done to originally produce the RFT stress plots. The two wheels used are shown in Figure 3-4.

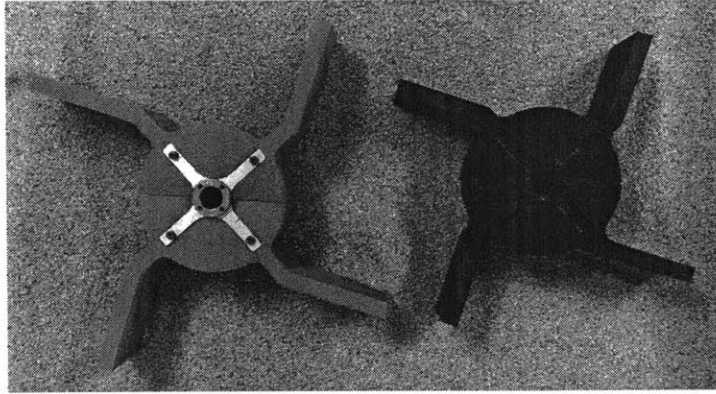


Figure 3-4: Four-spoke wheels.

The scaling constants of $A = 1$, $B = 1.2$, and $C = 1.44$ were used for the experiments. The various inputs for the two tests are shown below in Table 3.2.

Table 3.2: Four-Spoke Wheel Inputs

Small Wheel Scale	Small Wheel	Large Wheel Scale	Large Wheel
ω	$29.6^\circ/sec$	$\sqrt{\frac{A}{B}}\omega$	$27.0^\circ/sec$
L	7.08 cm	BL	8.50 cm
f	$\theta = 150^\circ$	f	$\theta = 150^\circ$
g	$9.81m/s^2$	Ag	$9.81m/s^2$
D	7 cm	$\frac{C}{B^2}D$	7 cm

Again, the mass and drawbar forces were varied to populate a range of power and velocity values. For each tire, three different masses, shown in Table 3.3, and five different drawbar forces, shown in Table 3.4, were used. For every combination of mass and drawbar force, five tests were run. To find the velocity and power, the average of the five tests were taken.

Table 3.3: Four-Spoke Masses

Small Wheel (m)	Large Wheel (Cm)
48.61N	70N
54.16N	78N
59.72N	86N

Table 3.4: Four-Spoke Drawbar Forces

Small Wheel (F_{draw})	Large Wheel (CAF_{draw})
-4.39lbs	-6.32lbs
-1.74lbs	-2.5lbs
0lbs	0lbs
1.74lbs	2.5lbs
4.39lbs	6.32lbs

3.3.1 Experimental Results

The results of the tests are shown in Tables 3.5 and 3.6.

Table 3.5: Small Wheel Results

Mass	F_{draw}	Average Velocity	Average Power
48.61 N	-4.39 lbs	82.72 ± 1.97 mm/s	0.99 ± 0.09 W
48.61 N	-1.74 lbs	67.30 ± 1.47 mm/s	1.35 ± 0.05 W
48.61 N	0 lbs	53.70 ± 1.50 mm/s	2.22 ± 0.14 W
48.61 N	1.74 lbs	33.58 ± 2.96 mm/s	2.84 ± 0.06 W
48.61 N	4.39 lbs	17.38 ± 4.55 mm/s	3.39 ± 0.23 W
54.16 N	-4.39 lbs	81.23 ± 5.45 mm/s	1.07 ± 0.14 W
54.16 N	-1.74 lbs	64.40 ± 3.06 mm/s	2.08 ± 0.37 W
54.16 N	0 lbs	50.01 ± 1.35 mm/s	2.58 ± 0.12 W
54.16 N	1.74 lbs	36.54 ± 4.50 mm/s	3.11 ± 0.18 W
54.16 N	4.39 lbs	24.21 ± 8.42 mm/s	3.26 ± 0.50 W
59.72 N	-4.39 lbs	80.48 ± 2.79 mm/s	1.29 ± 0.35 W
59.72 N	-1.74 lbs	64.11 ± 2.28 mm/s	2.23 ± 0.22 W
59.72 N	0 lbs	51.14 ± 0.59 mm/s	2.82 ± 0.28 W
59.72 N	1.74 lbs	37.13 ± 4.69 mm/s	3.51 ± 0.34 W
59.72 N	4.39 lbs	21.16 ± 2.90 mm/s	3.94 ± 0.26 W

Table 3.6: Large Wheel Results

Mass	F_{draw}	Average Velocity	Average Power
70 N	-6.32 lbs	90.46 ± 1.00 mm/s	1.82 ± 0.17 W
70 N	-2.50 lbs	76.29 ± 5.87 mm/s	2.66 ± 0.05 W
70 N	0 lbs	61.04 ± 2.16 mm/s	3.44 ± 0.10 W
70 N	2.50 lbs	51.17 ± 3.32 mm/s	4.52 ± 0.10 W
70 N	6.32 lbs	26.55 ± 7.61 mm/s	5.18 ± 0.23 W
78 N	-6.32 lbs	91.37 ± 1.90 mm/s	2.00 ± 0.18 W
78 N	-2.50 lbs	75.09 ± 4.52 mm/s	2.82 ± 0.16 W
78 N	0 lbs	60.58 ± 1.82 mm/s	3.69 ± 0.33 W
78 N	2.50 lbs	47.99 ± 1.97 mm/s	4.99 ± 0.27 W
78 N	6.32 lbs	32.94 ± 11.60 mm/s	5.58 ± 0.14 W
86 N	-6.32 lbs	88.52 ± 2.63 mm/s	2.18 ± 0.27 W
86 N	-2.50 lbs	78.51 ± 3.24 mm/s	2.38 ± 0.40 W
86 N	0 lbs	68.91 ± 4.34 mm/s	3.54 ± 0.32 W
86 N	2.50 lbs	49.34 ± 3.17 mm/s	4.76 ± 0.22 W
86 N	6.32 lbs	33.34 ± 2.51 mm/s	5.46 ± 0.28 W

These results are also plotted in Figures 3-5 and 3-6, with the RFT prediction line displayed.

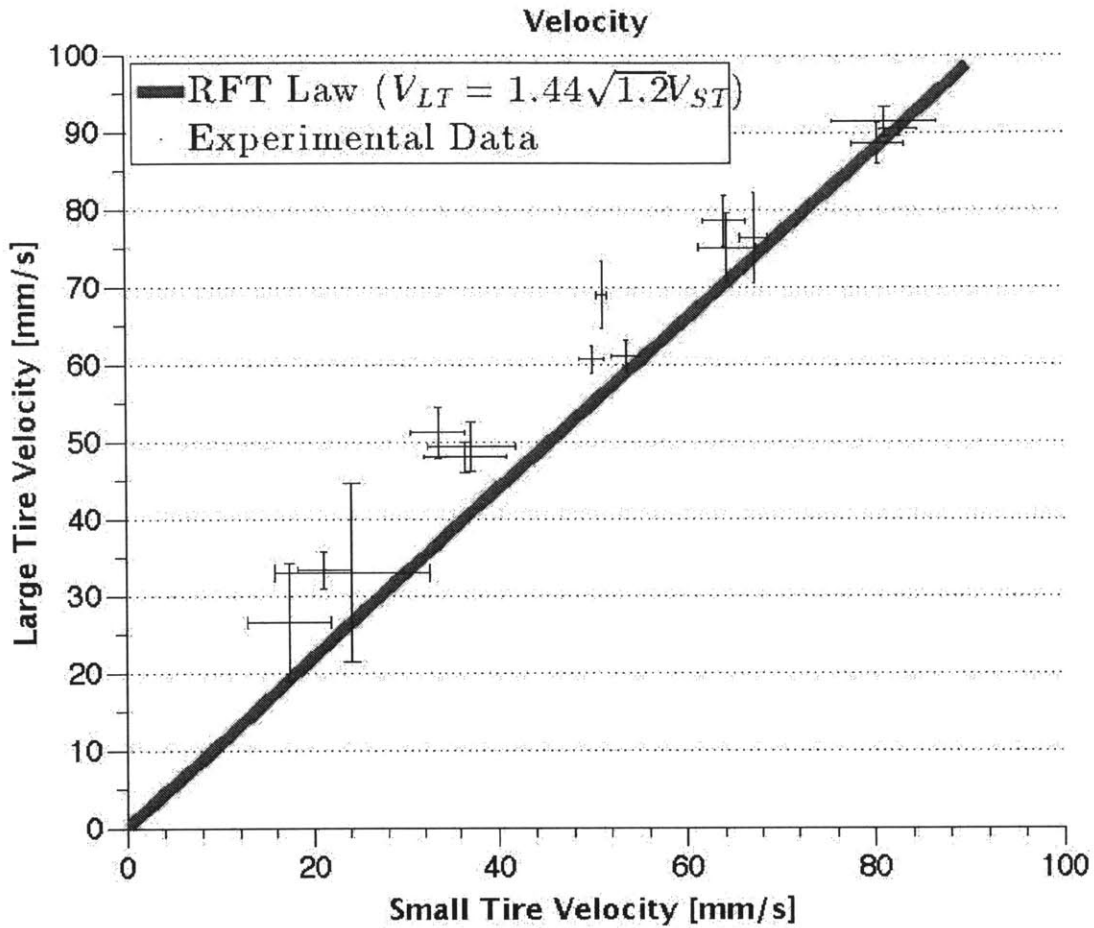


Figure 3-5: Velocity measurements.

The results indicate that the tires do seem to be following the RFT predicted scaling trend fairly well. There are still some sources of error, though, which can be reflected in the fact that the measured velocities tend to be above the RFT predicted line, exactly as the cylindrical wheel test experienced. This error could be due to two potential issues as discussed above. First, the friction in the bearings made it difficult to achieve exactly the intended mass, as there were fluctuations on the downward force the wheel exerted onto the sand. Second, the spring again was not accounted for in these tests, so the inertial mass was not exactly that shown in the scaling laws.

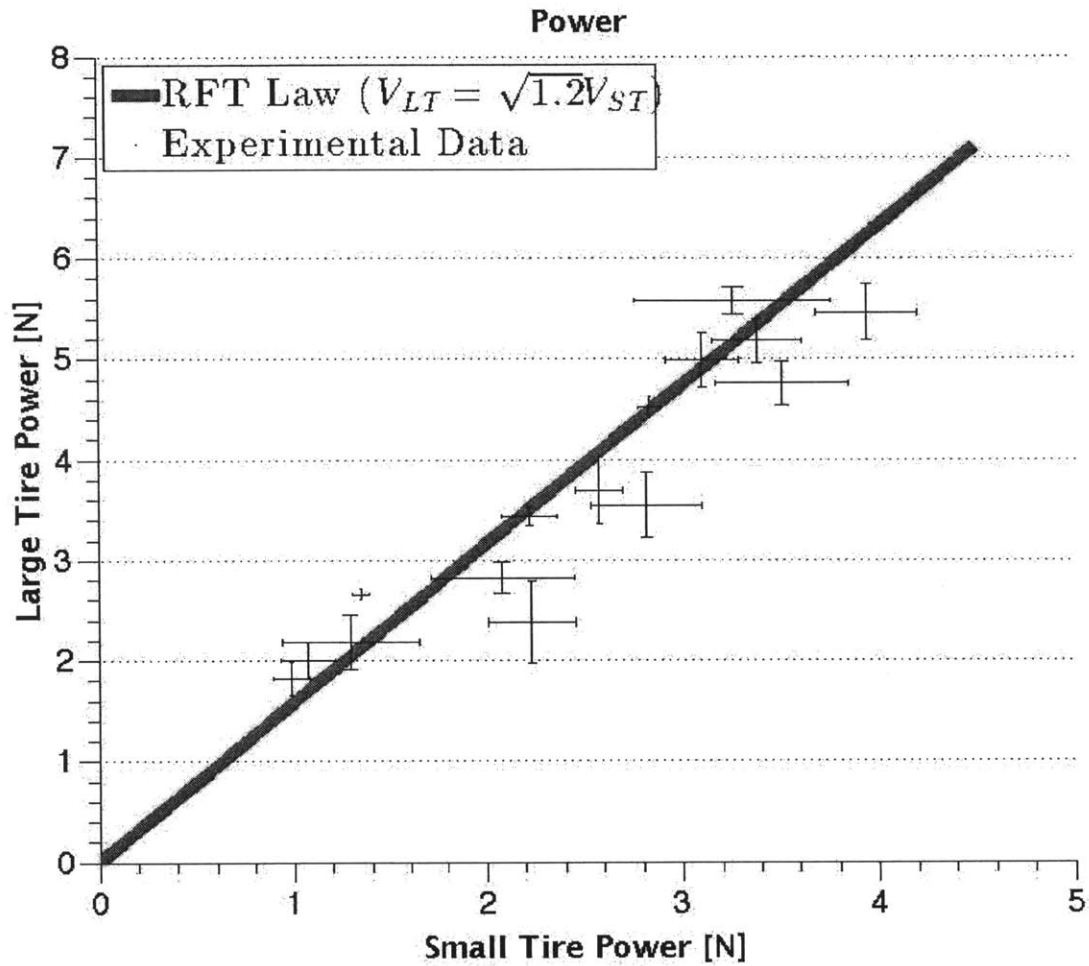


Figure 3-6: Power measurements.

3.4 Further Experimentation

Expanding upon these results, the next set of experiments that are currently being run are to test the scaling law with a different four spoke tire design that does not vary mass. In this test, the effect of the spring altering the inertial mass is removed. Ideally, this will provide a better analysis of the true nature of the RFT derived scaling relationship. After those tests, different four-bar designs will be tested to confirm the simulation results. It was shown in Chapter 2 that four spoke wheel designs with $\theta = 160^\circ$ were able to generate the highest velocities, while designs with $\theta = 230^\circ$ required the most power. Confirmation of these results, though, requires

further experimental verification by testing different wheels with every input held constant except for a changing angle θ .

Ultimately, the goal is to create a "smart" tire that is able to deform or actuate to the optimal shape for whatever condition it is in. This will require further simulation and experimentation to determine what the best shape is for any condition. Once a range of design conditions is met, work can be started on the design of the "smart" wheel. For instance, the four-spoke design can utilize a hinge and spring system to actively actuate and change the angle while in motion. For the "superballs", one can imagine using an air pressurized system to inflate or deflate membranes into different defined shapes as the wheel is moving.

Appendix A

Simulation Code

A.1 Four-Spoke Animation Code

```
1 %% Simulation of 4-spoke tire
2 % x is defined as positive to the right
3 % z is defined as positive upwards
4
5 %% Clear Everything
6 commandwindow
7 close all
8 clear all
9 clf
10 clc
11 format long
12
13 %% Inputs
14 theta=150*pi/180; % [radians]
15 TreadWidth = 0.07; % [m], in plane
16 TreadLength=0.085; % [m]
17 TireAxleInitCoord = [0,0]; %[x,z], [m]
18 omega=27*pi/180; % [rad/s]
19 init_velocity = [0,0]; %[vx, vz], [m/s]
20 Fspring=6.80389; %Spring force [kg]
```

```

21 TireMass = (70/9.81); % [kg]
22 duration = 25;% [sec], length of simulation
23 savepics=0; %make 1 to create pics, and 0 for no pics
24 anim=1; %1 there is an animation, and 0 for no animation
25 fps=30; % Animation frames per second
26 dbforce=0;% Drawback Force [N]
27 g = 9.81; % [m/s^2], gravitational acceleration
28 scaleFactor = 2.576*(g/9.81);
29 elbows=1;
30 NumTreads = 4;
31
32 %% Set Constants
33 NumSegs = 45*(elbows+1);
34 NumPieces=NumSegs*NumTreads;
35 SegLength = (TreadLength*(elbows+1))/NumSegs; % [m]
36 OrderMag = 10^6;
37
38 A00 = 0.206*OrderMag;
39 A10 = 0.169*OrderMag;
40 B11 = 0.212*OrderMag;
41 B01 = 0.358*OrderMag;
42 Bm11 = 0.055*OrderMag;
43 C11 = -0.124*OrderMag;
44 C01 = 0.253*OrderMag;
45 Cm11 = 0.007*OrderMag;
46 D10 = 0.088*OrderMag;
47
48 %% Initialization
49 ForceXs=[];
50 ForceZs=[];
51 XPos=[];
52 ZPos=[];
53 j=1:2:2*(NumSegs/(elbows+1));
54 i=1;
55 startingx=zeros((NumSegs/(elbows+1)),1);
56 startingz=zeros((NumSegs/(elbows+1)),1);

```

```

57 for i=1:(NumSegs/(elbows+1));
58     startingx(i)=(j(i)/(2*((NumSegs)/(elbows+1))))*TreadLength;
59 end
60 xposadd=TreadLength;
61 zposadd=0;
62 startingx1=[];
63 startingz1=[];
64 for j=1:elbows
65     startingx1(((j-1)*(NumSegs/(elbows+1)))+1:(j*(NumSegs/(elbows+1))),1)=...
66     ((startingx.*cos(j*(pi-theta)))+xposadd);
67     startingz1(((j-1)*(NumSegs/(elbows+1)))+1:(j*(NumSegs/(elbows+1))),1)=...
68     (startingx.*sin(j*(pi-theta)))+zposadd;
69     xposadd=xposadd+(TreadLength*cos(j*(pi-theta)));
70     zposadd=zposadd+(TreadLength*sin(j*(pi-theta)));
71 end
72 startingz=[startingz; startingz1];
73 startingx=[startingx; startingx1];
74 betainit=zeros(NumSegs,1);
75 for jr=1:1:NumSegs/2;
76     betainit(jr)=0;
77     betainit((NumSegs/2)+jr)=-atan(startingz(end)/...
78     (startingx(end)-TreadLength));
79 end
80 initialtread=zeros(NumPieces,3);
81 j=1;
82 for j=0:NumTreads-1;
83     initialtread((1+j*NumSegs):(NumSegs+j*NumSegs),1)=...
84     (cos(j*2*pi/NumTreads).*...
85     (startingx)+(sin(j*2*pi/NumTreads).*(startingz)));
86     initialtread((1+j*NumSegs):(NumSegs+j*NumSegs),2)=...
87     (-sin(j*2*pi/NumTreads).*...
88     (startingx)+(cos(j*2*pi/NumTreads).*(startingz)));
89     origx(j+1)=(cos(j*2*pi/NumTreads)*(TreadLength))+TireAxleInitCoord(1);
90     origz(j+1)=(-sin(j*2*pi/NumTreads)*(TreadLength))+TireAxleInitCoord(2);
91     initialtread((j*NumSegs+1):((j+1)*NumSegs),3)=((betainit+(j*pi/2)).*...
92     ((betainit+(j*pi/2))<=pi/2))+...

```

```

93     ((betainit+(j*pi/2))-pi).*((betainit+(j*pi/2))>pi/2));
94 end
95
96
97 xs=(origx-TireAxleInitCoord(1))';
98 zs=(origz-TireAxleInitCoord(2))';
99
100 %%Key
101 % Startingx and startingz are midpts of segment on one tread
102 % initialtread is midpts of segments
103 % origx and origz are 4 elbow locations
104 % xs and zs are origx and origz minus the location of the axle initial
105 % position
106
107 allx=[];
108 allz=[];
109 allvx=[];
110 allvz=[];
111 newx=[];
112 newz=[];
113
114 %% Function
115 Vo = zeros(1,5);
116 Vo(1) = init_velocity(1); % initial velocity in x-dir
117 Vo(2) = init_velocity(2); % initial velocity in z-dir
118 Vo(3) = TireAxleInitCoord(1); % initial axle position in x
119 Vo(4) = TireAxleInitCoord(2); % initial axle position in z
120 Vo(5) = 0; % initial dissipated Power
121
122 options = odeset('RelTol',1e-4,'AbsTol',1e-7);
123 odefix = @(t, V) FunctionVODEFourBar(t, V, TireMass, TreadWidth, SegLength,...
124     omega, NumPieces, NumTreads, OrderMag, scaleFactor,...
125     initialtread, g, dbforce, Fspring);
126 [TOUT,VOUT] = ode45(odefix,[0 duration],Vo,options);
127
128 %% Average Power and Velocity measurements

```

```

129 pdpos=find((diff(sign(VOUT(:,2))))==2);
130
131 startavg=3;
132 endavg=6;
133
134 mz1=(VOUT(pdpos(endavg)+1,2)-VOUT(pdpos(endavg),2))/...
135 (TOUT(pdpos(endavg)+1)...
136 -TOUT(pdpos(endavg))); %slope of (t,z)
137 t01=((1/mz1)*-VOUT(pdpos(endavg),2))+TOUT(pdpos(endavg));
138 mv1=(VOUT(pdpos(endavg)+1,3)-VOUT(pdpos(endavg),3))/...
139 (TOUT(pdpos(endavg)+1)...
140 -TOUT(pdpos(endavg))); %slope of (t,vx)
141 mp1=(VOUT(pdpos(endavg)+1,5)-VOUT(pdpos(endavg),5))/...
142 (TOUT(pdpos(endavg)+1)...
143 -TOUT(pdpos(endavg))); %slope of (t,pow)
144 vxfix1=(mv1*(t01-TOUT(pdpos(endavg))))+VOUT(pdpos(endavg),3);
145 powfix1=(mp1*(t01-TOUT(pdpos(endavg))))+VOUT(pdpos(endavg),5);
146
147 mz2=(VOUT(pdpos(startavg)+1,2)-VOUT(pdpos(startavg),2))/...
148 (TOUT(pdpos(startavg)+1)...
149 -TOUT(pdpos(startavg))); %slope of (t,z)
150 t02=((1/mz2)*-VOUT(pdpos(startavg),2))+TOUT(pdpos(startavg));
151 mv2=(VOUT(pdpos(startavg)+1,3)-VOUT(pdpos(startavg),3))/...
152 (TOUT(pdpos(startavg)+1)...
153 -TOUT(pdpos(startavg))); %slope of (t,vx)
154 mp2=(VOUT(pdpos(startavg)+1,5)-VOUT(pdpos(startavg),5))/...
155 (TOUT(pdpos(startavg)+1)...
156 -TOUT(pdpos(startavg))); %slope of (t,pow)
157 vxfix2=(mv2*(t02-TOUT(pdpos(startavg))))+VOUT(pdpos(startavg),3);
158 powfix2=(mp2*(t02-TOUT(pdpos(startavg))))+VOUT(pdpos(startavg),5);
159
160 vxavg=(vxfix1-vxfix2)/(t01-t02) % [m/s]
161 Power=(powfix1-powfix2)/(t01-t02) % [W]
162
163 %% Animation Initialization
164 u=find(diff(sign(diff(mod((TOUT),1/fps))))==2);

```



```

165 TOUTfps=TOUT(u);
166 VOUT=VOUT(u,:);
167
168 Torque=[];
169 kd=1;
170 for k=1:1:length(TOUTfps)
171
172     index = zeros(NumPieces,7); %Local number, x,z,vx,vz,Beta,gamma
173     index(:,1) = [1:NumPieces];
174     jr=1;
175
176     V(1)=VOUT(k,1);
177     V(2)=VOUT(k,2);
178     V(3)=VOUT(k,3);
179     V(4)=VOUT(k,4);
180     t=TOUTfps(k);
181     timetot(kd)=t;
182
183     for jr=1:NumPieces;
184         index(jr,2)=V(3)+((cos(omega*t)*(initialtread(jr,1)))+...
185             (sin(omega*t)*...
186             (initialtread(jr,2))));
187         %New X position using rotation matrix
188         index(jr,3)=V(4)+((-sin(omega*t)*(initialtread(jr,1)))+...
189             (cos(omega*t)*...
190             (initialtread(jr,2))));
191         %New Z position using rotation matrix
192         index(jr,4)=V(1)+(omega*(index(jr,3)-V(4)));
193         %New velocity in x-dir
194         index(jr,5)=V(2)+(-omega*(index(jr,2)-V(3)));
195         %New velocity in z-dir
196         index(jr,6)=((initialtread(jr,3)+(omega*t))*((initialtread(jr,3)+...
197             (omega*t))<=pi/2))+...
198             (((initialtread(jr,3)+(omega*t)-pi))*((initialtread(jr,3)+...
199             (omega*t))>pi/2)); %beta,
200         index(jr,7)=((atan(index(jr,5)/index(jr,4))*(index(jr,4)<0))+...

```

```

201         ((atan(index(jr,5)/index(jr,4))+pi)*(index(jr,4)>=0))); %gamma
202     end
203
204     A00 = 0.206*OrderMag;
205     A10 = 0.169*OrderMag;
206     B11 = 0.212*OrderMag;
207     B01 = 0.358*OrderMag;
208     Bm11 = 0.055*OrderMag;
209     C11 = -0.124*OrderMag;
210     C01 = 0.253*OrderMag;
211     Cm11 = 0.007*OrderMag;
212     D10 = 0.088*OrderMag;
213
214     jr=1;
215     ForceX=zeros(1,NumPieces);
216     ForceZ=zeros(1,NumPieces);
217     torque=zeros(1,NumPieces);
218     for jr=1:NumPieces;
219         B=index(jr,6);
220         G=index(jr,7);
221
222         alphaX = scaleFactor*(Cm11*cos(-2*B+G) + C01*cos(G) + ...
223             C11*cos(2*B+G) + D10*sin(2*B));
224         alphaZ = scaleFactor*(A10*cos(2*B) + A00 + Bm11*sin((-2*B)+G) + ...
225             B01*sin(G) + B11*sin((2*B)+G));
226
227         if index(jr,3)<=0;
228             ForceX(jr) = alphaX*SegLength*TreadWidth*-index(jr,3);
229             ForceZ(jr) = alphaZ*SegLength*TreadWidth*-index(jr,3);
230         else
231             ForceX(jr) = 0;
232             ForceZ(jr) = 0;
233         end
234         posvec=[index(jr,2)-V(3);index(jr,3)-V(4);0];
235         forcevec=[ForceX(jr);ForceZ(jr);0];
236         torquecross=cross(posvec, forcevec);

```

```

237         torque(jr)=torquecross(3);
238
239     end
240
241
242     Torquet=sum(torque); %positive is z axis out of screen/page
243     Torque=[Torque Torquet];
244     ForceXs=[ForceXs ForceX];
245     ForceZs=[ForceZs ForceZ];
246     ForceXTot=sum(ForceX);
247     ForceZTot=sum(ForceZ);
248
249     allx=[allx; index(:,2)]; %x positions of each tread
250     allz=[allz; index(:,3)]; %z positions of each tread
251     allvx=[allvx; index(:,4)]; %vx of each tread
252     allvz=[allvz; index(:,5)]; %vz of each tread
253     newx=[newx ((cos(omega*t).*xs)+(sin(omega*t).*zs))+...
254     V(3)]; %New X position of elbows using rotation matrix
255     newz=[newz ((-sin(omega*t).*xs)+(cos(omega*t).*zs))+...
256     V(4)]; %New Z position of elbows using rotation matrix
257     kd=kd+1;
258 end
259
260 if anim==1;
261     allvx=allvx*10^-(0.5); %scale vectors yourself
262     allvz=allvz*10^-(0.5); %scale vectors yourself
263     ForceXs=ForceXs*10^-2; %scale vectors yourself
264     ForceZs=ForceZs*10^-2; %scale vectors yourself
265
266     % Animation of Tire
267     allx2=allx';
268     allz2=allz';
269     j=1;
270     k=1;
271     kk=sprintf('%d', k);
272     while j<=(length(timetot));

```

```

273     plot([newx(2,j)-0.25,newx(2,j)+0.5],[0,0],'y') %plot sand level
274     axis([newx(2,j)-0.25,newx(2,j)+0.5,-0.25,0.5]) %set axis
275     axis equal
276     if NumTreads==4
277         figure(1)
278         hold on
279         set(gca,'FontSize',18)
280         xlabel('X')
281         ylabel('Z')
282         rectangle('Position',[newx(2,j)-0.75,-0.5,1.25,0.5],...
283             'FaceColor',...
284             'y','edgecolor','y') %sand base
285         legend(num2str(timetot(j)), 'FontSize',16)
286         %Time in upper corner
287         plot([newx(1,j),newx(3,j)],[newz(1,j),newz(3,j)],...
288             [newx(2,j),...
289             newx(4,j)],[newz(2,j),newz(4,j)], 'k') %plot tire
290         quiver(allx(1+(j-1)*NumPieces:NumPieces+(j-1)*...
291             NumPieces),...
292             allz(1+(j-1)*NumPieces:NumPieces+(j-1)*NumPieces),...
293             allvx(1+(j-1)*NumPieces:NumPieces+(j-1)*NumPieces),...
294             allvz(1+(j-1)*NumPieces:NumPieces+(j-1)*NumPieces),0,'r')
295         %plot velocity vectors
296         quiver(allx2(1+(j-1)*NumPieces:NumPieces+(j-1)*NumPieces),...
297             allz2(1+(j-1)*NumPieces:NumPieces+(j-1)*NumPieces),...
298             ForceXs(1+(j-1)*...
299             NumPieces:NumPieces+(j-1)*NumPieces),ForceZs(1+(j-1)*...
300             NumPieces:NumPieces+(j-1)*NumPieces),0,'b') %plot force vectors
301         if savepics==1;
302             hand = figure(1);
303             if k < 10
304                 numstr = ['0000',num2str(k)];
305             elseif k < 100
306                 numstr = ['000',num2str(k)];
307             elseif k < 1000
308                 numstr = ['00',num2str(k)];

```

```

309             elseif k < 10000
310                 numstr = ['0',num2str(k)];
311             else
312                 numstr = num2str(k);
313             end
314             saveas(hand,['f_' numstr],'jpg')
315             k=k+1;
316             kk=sprintf('%.4d', k);
317         end
318         hold off
319     end
320     drawnow;
321     j=j+1;
322 end
323 end

```

A.2 Four-Spoke Function (FunctionVODEFourBar)

```

1 function [ Vdot ] = FunctionVODEFourBar( t,V,TireMass,...
2 TreadWidth, SegLength, omega, NumPieces, NumTreads,...
3 OrderMag, scaleFactor, initialtread, g, dbforce, Fspring )
4 % This is the function is used by ode45 function
5 index = zeros(NumPieces,7);
6 %Local number, x,z,vx,vz,Beta,gamma
7 index(:,1) = [1:NumPieces];
8 jr=1;
9
10 for jr=1:NumPieces;
11     index(jr,2)=V(3)+((cos(omega*t)*(initialtread(jr,1)))+...
12     (sin(omega*t)*(initialtread(jr,2))));
13     %New X position using rotation matrix
14     index(jr,3)=V(4)+((-sin(omega*t)*(initialtread(jr,1)))+...
15     (cos(omega*t)*(initialtread(jr,2))));
16     %New Z position using rotation matrix

```

```

17     index(jr,4)=V(1)+(omega*(index(jr,3)-V(4)));
18     %New velocity in x-dir
19     index(jr,5)=V(2)+(-omega*(index(jr,2)-V(3)));
20     %New velocity in z-dir
21     index(jr,6)=((initialtread(jr,3)+(omega*t))*...
22     ((initialtread(jr,3)+(omega*t))<=pi/2))+...
23     (((initialtread(jr,3)+(omega*t)-pi))*...
24     ((initialtread(jr,3)+(omega*t))>pi/2)); %beta,
25     index(jr,7)=((atan(index(jr,5)/index(jr,4))*...
26     (index(jr,4)<0))+((atan(index(jr,5)/index(jr,4))+...
27     pi)*(index(jr,4)>=0))); %gamma
28 end
29
30 A00 = 0.206*OrderMag;
31 A10 = 0.169*OrderMag;
32 B11 = 0.212*OrderMag;
33 B01 = 0.358*OrderMag;
34 Bm11 = 0.055*OrderMag;
35 C11 = -0.124*OrderMag;
36 C01 = 0.253*OrderMag;
37 Cm11 = 0.007*OrderMag;
38 D10 = 0.088*OrderMag;
39
40 jr=1;
41 ForceX=zeros(1,NumPieces);
42 ForceZ=zeros(1,NumPieces);
43 torque=zeros(1,NumPieces);
44 for jr=1:NumPieces;
45     B=index(jr,6);
46     G=index(jr,7);
47
48     alphaX = scaleFactor*(Cm11*cos(-2*B+G) +...
49     C01*cos(G) + C11*cos(2*B+G) +...
50     D10*sin(2*B));
51     alphaZ = scaleFactor*(A10*cos(2*B) + A00 +...
52     Bm11*sin((-2*B)+G) + B01*sin(G) +...

```

```

53     B11*sin((2*B)+G));
54
55     if index(jr,3)<=0;
56         ForceX(jr) = alphaX*SegLength*...
57         TreadWidth*-index(jr,3);
58         ForceZ(jr) = alphaZ*SegLength*...
59         TreadWidth*-index(jr,3);
60     else
61         ForceX(jr) = 0;
62         ForceZ(jr) = 0;
63     end
64     posvec=[index(jr,2)-V(3);index(jr,3)-V(4);0];
65     forcevec=[ForceX(jr);ForceZ(jr);0];
66     torquecross=cross(posvec, forcevec);
67     torque(jr)=torquecross(3);
68
69 end
70
71 max(ForceX);
72 Torquetotal=sum(torque);
73 ForceXTot=sum(ForceX);
74 ForceZTot=sum(ForceZ);
75 Vdot = zeros(4,1);
76 Vdot(1) = (ForceXTot-dbforce)/TireMass;
77 %Accel in x-direction
78 Vdot(2) = (ForceZTot-(TireMass*g)+...
79 (Fspring*g))/TireMass;
80 %Accel in z-direction +(15*4.448)+(6.80389*g)
81 Vdot(3) = V(1); %Velocity in x-dir
82 Vdot(4) = V(2); %Velocity in z-dir
83 Vdot(5) = omega*Torquetotal;
84 %Derivative of Energy dissipated
85
86 end

```

A.3 "Superball" Animation Code

```
1 %% Simulation of superball wheel
2 % x is defined as positive to the right
3 % z is defined as positive upwards
4
5 %% Clear Everything
6 commandwindow
7 clear all
8 close all
9 clf
10 clc
11 format long
12
13 %% Inputs
14 g = 9.81; %[m/s^2], gravitational acceleration
15 TireAxleInitCoord = [0,0]; %[x,z]
16 TreadWidth = 0.15*10^2; % [m], in plane
17 NumSegs = 4;
18 p=1; % chi wheel shape parameter
19 omega=30*pi/180; % [rad/s], positive in clockwise direction
20 init_velocity = [0,0]; %[vx, vz] %[m/s]
21 rad=(1)^(2*p); %radius to 2p
22 TireMass =1000;% [kg], mass of wheel
23 duration = 2; %[sec] length of simulation
24 dbforce=0; %Drawback force [N]
25 OrderMag = 10^6;
26 scaleFactor = 2.576;
27
28 A00 = 0.206*OrderMag;
29 A10 = 0.169*OrderMag;
30 B11 = 0.212*OrderMag;
31 B01 = 0.358*OrderMag;
32 Bm11 = 0.055*OrderMag;
```



```

33 C11 = -0.124*OrderMag;
34 C01 = 0.253*OrderMag;
35 Cm11 = 0.007*OrderMag;
36 D10 = 0.088*OrderMag;
37
38 %% Initialization
39 theta=linspace(0,pi,NumSegs);
40 theta=sort(theta);
41
42 initialtread=[];
43 midpts=[];
44 grad=[];
45 ms=[];
46 points=[];
47 rho=(rad./((abs(sin(theta)).^(2*p))+(abs(cos(theta))....
48 ^ (2*p)))) .^(1/(2*p));
49 x1=rho.*cos(theta);
50 y1=rho.*sin(theta);
51 xsfix=fliplr(x1);
52 xsfix=xsfix(2:end);
53 x=[x1 xsfix];
54 ysfix=fliplr(y1);
55 ysfix=ysfix(2:end);
56 y=[y1 -ysfix];
57 points(:,1)=x;
58 points(:,2)=y;
59 initialtread=(points(1:end-1,:)+points(2:end,:))/2;
60 for jz=1:(length(initialtread)-1);
61     midpts(jz,1)=(initialtread(jz,1)+initialtread(jz+1,1))/2;
62     midpts(jz,2)=(initialtread(jz,2)+initialtread(jz+1,2))/2;
63     ms(jz)=(initialtread(jz+1,2)-initialtread(jz,2))/...
64         (initialtread(jz+1,1)-initialtread(jz,1));
65 end
66 midpts(end+1,1)=(initialtread(end,1)+initialtread(1,1))/2;
67 midpts(end,2)=(initialtread(end,2)+initialtread(1,2))/2;
68 ms(end+1)=(initialtread(1,2)-initialtread(end,2))/...

```

```

69 (initialtread(1,1)-initialtread(end,1));
70 for jx=1:length(midpts);
71     if (ms(jx)==Inf)
72         grad(jx,:)=[1,0];
73     elseif (ms(jx)==-Inf)
74         grad(jx,:)=[-1,0];
75     else
76         if midpts(jx,2)>=0
77             a=[1;ms(jx)];
78             aperp=[0,-1;1,0]*a;
79             grad(jx,:)=[aperp(1),aperp(2)];
80         else
81             a=[1;ms(jx)];
82             aperp=[0,1;-1,0]*a;
83             grad(jx,:)=[aperp(1),aperp(2)];
84         end
85     end
86     grad(jx,:)=[grad(jx,1)/sqrt((grad(jx,1)^2+...
87         (grad(jx,2)^2)),grad(jx,2)/sqrt((grad(jx,1)^2)...
88         +(grad(jx,2)^2))];
89 end
90 SegLengths=((points(2:end,1)-points(1:end-1,1)).^2+...
91 (points(2:end,2)-points(1:end-1,2)).^2).^0.5;
92 vec2=[-1;0];
93 for jvv=1:1:length(midpts);
94     vec1=[midpts(jvv,1)-initialtread(jvv,1);...
95     midpts(jvv,2)-initialtread(jvv,2)];
96     midpts(jvv,3)=(pi-acos(dot(vec1,vec2)/...
97     (sqrt((vec1(1)^2)+(vec1(2)^2))*sqrt((vec2(1)^2)+...
98     (vec2(2)^2))))*(vec1(1)>=0&&vec1(2)<=0))+...
99     ((acos(dot(vec1,vec2)/(sqrt((vec1(1)^2)+...
100     (vec1(2)^2))*sqrt((vec2(1)^2)+(vec2(2)^2)))))*...
101     (vec1(1)<0&&vec1(2)>=0))+((-acos(dot(vec1,vec2)/...
102     (sqrt((vec1(1)^2)+(vec1(2)^2))*sqrt((vec2(1)^2)+...
103     (vec2(2)^2))))*(vec1(1)<0&&vec1(2)<0))+...
104     ((acos(dot(vec1,vec2)/(sqrt((vec1(1)^2)+...

```

```

105     (vec1(2)^2))*sqrt((vec2(1)^2)+(vec2(2)^2))))-pi)*...
106     (vec1(1)>=0&&vec1(2)>0));
107 end
108 xs=initialtread(:,1);
109 zs=initialtread(:,2);
110
111 %% Function
112 Vo = zeros(1,5);
113 Vo(1) = init_velocity(1); % initial velocity in x-dir
114 Vo(2) = init_velocity(2); % initial velocity in z-dir
115 Vo(3) = TireAxleInitCoord(1); % initial axle position in x
116 Vo(4) = TireAxleInitCoord(2); % initial axle position in z
117 Vo(5) = 0; % initial dissipated Power
118
119 odefix = @(t, V) FunctionVODEShapesShadow(t, V,...
120     TireMass, TreadWidth, omega, OrderMag,...
121     scaleFactor, midpts, g, SegLengths, grad);
122 [TOUT,VOUT] = ode45(odefix,[0 duration],Vo);
123
124 pdpos=find((diff(sign(VOUT(:,2))))==2);
125
126 mz1=(VOUT(pdpos(24)+1,2)-VOUT(pdpos(24),2))/...
127 (TOUT(pdpos(24)+1)-TOUT(pdpos(24))); %slope of (t,z)
128 t01=((1/mz1)*-VOUT(pdpos(24),2))+TOUT(pdpos(24));
129 mv1=(VOUT(pdpos(24)+1,3)-VOUT(pdpos(24),3))/...
130 (TOUT(pdpos(24)+1)-TOUT(pdpos(24))); %slope of (t,vx)
131 mp1=(VOUT(pdpos(24)+1,5)-VOUT(pdpos(24),5))/...
132 (TOUT(pdpos(24)+1)-TOUT(pdpos(24))); %slope of (t,pow)
133 vxfix1=(mv1*(t01-TOUT(pdpos(24))))+VOUT(pdpos(24),3);
134 powfix1=(mp1*(t01-TOUT(pdpos(24))))+VOUT(pdpos(24),5);
135
136 mz2=(VOUT(pdpos(4)+1,2)-VOUT(pdpos(4),2))/...
137 (TOUT(pdpos(4)+1)-TOUT(pdpos(4))); %slope of (t,z)
138 t02=((1/mz2)*-VOUT(pdpos(4),2))+TOUT(pdpos(4));
139 mv2=(VOUT(pdpos(4)+1,3)-VOUT(pdpos(4),3))/...
140 (TOUT(pdpos(4)+1)-TOUT(pdpos(4))); %slope of (t,vx)

```

```

141 mp2=(VOUT(pdpos(4)+1,5)-VOUT(pdpos(4),5))/...
142 (TOUT(pdpos(4)+1)-TOUT(pdpos(4))); %slope of (t,pow)
143 vxfix2=(mv2*(t02-TOUT(pdpos(4))))+VOUT(pdpos(4),3);
144 powfix2=(mp2*(t02-TOUT(pdpos(4))))+VOUT(pdpos(4),5);
145
146 vxavg=(vxfix1-vxfix2)/(t01-t02) % [m/s]
147 Power=(powfix1-powfix2)/(t01-t02) % [W]
148
149 %% Animation Initialization
150 ForceXs=[];
151 ForceZs=[];
152 Torque=[];
153 allx=[];
154 allz=[];
155 allvx=[];
156 allvz=[];
157 newx=[];
158 newz=[];
159 initialtread=midpts;
160 for k=1:length(TOUT)
161     V(1)=VOUT(k,1);
162     V(2)=VOUT(k,2);
163     V(3)=VOUT(k,3);
164     V(4)=VOUT(k,4);
165     t=TOUT(k);
166     [dim dims]=size(initialtread);
167     index = zeros(dim,9);
168     %Local number, x,z,vx,vz,Beta,gamma
169     index(:,1) = [1:dim];
170     jr=1;
171
172     for jr=1:dim;
173         index(jr,2)=V(3)+((cos(omega*t)*...
174         (initialtread(jr,1)))+(sin(omega*t)*...
175         (initialtread(jr,2))));
176         %New X position using rotation matrix

```

```

177     index(jr,3)=V(4)+((-sin(omega*t))*...
178     (initialtread(jr,1)))+(cos(omega*t))*...
179     (initialtread(jr,2)));
180     %New Z position using rotation matrix
181     index(jr,4)=V(1)+(omega*(index(jr,3)-V(4)));
182     %New velocity in x-dir
183     index(jr,5)=V(2)+(-omega*(index(jr,2)-V(3)));
184     %New velocity in z-dir
185     index(jr,6)=((initialtread(jr,3)+...
186     (omega*t))*((initialtread(jr,3)+...
187     (omega*t))<=pi/2))+(((initialtread(jr,3)...
188     +(omega*t)-pi))*((initialtread(jr,3)+...
189     (omega*t))>pi/2)); %beta,
190     index(jr,7)=((atan(index(jr,5)/index(jr,4))*...
191     (index(jr,4)<0))+((atan(index(jr,5)/index(jr,4))+...
192     pi)*(index(jr,4)>=0))); %gamma
193     gradrot=[cos(omega*t), sin(omega*t);...
194     -sin(omega*t) cos(omega*t)]*[grad(jr,1);grad(jr,2)];
195     index(jr,8)=gradrot(1);
196     index(jr,9)=gradrot(2);
197     end
198
199     A00 = 0.206*OrderMag;
200     A10 = 0.169*OrderMag;
201     B11 = 0.212*OrderMag;
202     B01 = 0.358*OrderMag;
203     Bm11 = 0.055*OrderMag;
204     C11 = -0.124*OrderMag;
205     C01 = 0.253*OrderMag;
206     Cm11 = 0.007*OrderMag;
207     D10 = 0.088*OrderMag;
208
209     jr=1;
210     ForceX=zeros(1,dim);
211     ForceZ=zeros(1,dim);
212     torque=zeros(1,dim);

```

```

213     for jr=1:dim;
214         velocs=index(jr,4:5);
215         B=index(jr,6);
216         G=index(jr,7);
217         grads=index(jr,8:9);
218         coeff=1;
219         alphaX = coeff*scaleFactor*(Cm1*cos(-2*B+G)...
220             + C01*cos(G) + C11*cos(2*B+G) + D10*sin(2*B));
221         alphaZ = coeff*scaleFactor*(A10*cos(2*B)...
222             + A00 + Bm11*sin((-2*B)+G) + B01*sin(G)...
223             + B11*sin((2*B)+G));
224         if index(jr,3)<=0;
225             ForceX(jr) = (dot([grads(1);grads(2)],...
226                 [velocs(1);velocs(2)])>0)*alphaX*SegLengths(jr)*...
227                 TreadWidth*-index(jr,3);
228             ForceZ(jr) = (dot([grads(1);grads(2)],...
229                 [velocs(1);velocs(2)])>0)*alphaZ*...
230                 SegLengths(jr)*TreadWidth*-index(jr,3);
231         else
232             ForceX(jr) = 0;
233             ForceZ(jr) = 0;
234         end
235         posvec=[index(jr,2)-V(3);index(jr,3)-V(4);0];
236         forcevec=[ForceX(jr);ForceZ(jr);0];
237         torquecross=cross(posvec, forcevec);
238         torque(jr)=torquecross(3);
239
240     end
241
242     Torquetotal=sum(torque);
243     Torque=[Torque Torquetotal];
244     ForceXs=[ForceXs ForceX];
245     ForceZs=[ForceZs ForceZ];
246
247     ForceXTot=sum(ForceX);
248     ForceZTot=sum(ForceZ);

```

```

249
250     allx=[allx; index(:,2)];
251     allz=[allz; index(:,3)];
252     allvx=[allvx; index(:,4)];
253     allvz=[allvz; index(:,5)];
254     newxs=((cos(omega*t).*xs)...
255     +(sin(omega*t).*zs))+V(3));
256     newxs(end+1)=((cos(omega*t)*...
257     xs(1))+sin(omega*t)*zs(1))+V(3));
258     newx=[newx newxs];
259
260     newzs=((-sin(omega*t).*xs)+...
261     (cos(omega*t).*zs))+V(4));
262     newzs(end+1)=((-sin(omega*t)*xs(1))+...
263     (cos(omega*t)*zs(1))+V(4));
264     newz=[newz newzs];
265 end
266
267 %% Animation
268
269 allvx=allvx*10^-1;
270 allvz=allvz*10^-1;
271 ForceXs=ForceXs*10^-4;
272 ForceZs=ForceZs*10^-4;
273
274 allx2=allx';
275 allz2=allz';
276 j=1;
277 k=1;
278 kk=sprintf('%4d', k);
279 NumPieces=(NumSegs*2)-2;
280 while j<=(length(TOUT))
281     axis([newx(2,j)-0.5,newx(2,j)+0.5,-0.5,1])
282     axis equal
283
284     figure(1)

```

```

285     hold on
286     rectangle('Position',[newx(2,j)-0.5,-0.5,6,0.5],...
287     'FaceColor','y','edgecolor','y')
288     legend(num2str(TOUT(j)))
289     plot(newx(((j-1)*(NumPieces+1))+1):...
290     (j*(NumPieces+1)),newz(((j-1)*...
291     (NumPieces+1))+1):(j*(NumPieces+1)),'k-')
292     quiver(allx(1+(j-1)*NumPieces:NumPieces+...
293     (j-1)*NumPieces),allz(1+(j-1)*NumPieces:NumPieces+...
294     (j-1)*NumPieces),allvx(1+(j-1)*NumPieces:NumPieces+...
295     (j-1)*NumPieces),allvz(1+(j-1)*NumPieces:NumPieces+...
296     (j-1)*NumPieces),0,'r')
297     quiver(allx2(1+(j-1)*NumPieces:NumPieces+(j-1)*...
298     NumPieces),allz2(1+(j-1)*NumPieces:NumPieces+...
299     (j-1)*NumPieces),ForceXs(1+(j-1)*NumPieces:...
300     NumPieces+(j-1)*NumPieces),ForceZs(1+(j-1)*...
301     NumPieces:NumPieces+(j-1)*NumPieces),0,'b')
302
303     hand = figure(1);
304     if k < 10
305         numstr = ['0000',num2str(k)];
306     elseif k < 100
307         numstr = ['000',num2str(k)];
308     elseif k < 1000
309         numstr = ['00',num2str(k)];
310     elseif k < 10000
311         numstr = ['0',num2str(k)];
312     else
313         numstr = num2str(k);
314     end
315     saveas(hand,['f_' numstr],'jpg')
316     k=k+1;
317     kk=sprintf('%.4d', k);
318     hold off
319
320     drawnow;

```



```

321     j=j+4;
322
323 end

```

A.4 "Superball" Function (FunctionVODEShapesShadow)

```

1 function [ Vdot ] = FunctionVODEShapesShadow( t,...
2 V,TireMass, TreadWidth, omega, OrderMag,...
3 scaleFactor, initialtread, g, SegLengths, grad, dbforce )
4 % This is the function is used by ode45 function
5 [dim dims]=size(initialtread);
6 index = zeros(dim,9);
7 %Local number, x, z, vx, vz, Beta, gamma
8 index(:,1) = [1:dim];
9 jr=1;
10
11 for jr=1:dim;
12     index(jr,2)=V(3)+((cos(omega*t))*...
13     (initialtread(jr,1)))+(sin(omega*t))*...
14     (initialtread(jr,2)));
15     %New X position using rotation matrix
16     index(jr,3)=V(4)+((-sin(omega*t))*...
17     (initialtread(jr,1)))+(cos(omega*t))*...
18     (initialtread(jr,2)));
19     %New Z position using rotation matrix
20     index(jr,4)=V(1)+(omega*...
21     (index(jr,3)-V(4)));
22     %New velocity in x-dir
23     index(jr,5)=V(2)+(-omega*...
24     (index(jr,2)-V(3)));
25     %New velocity in z-dir
26     index(jr,6)=((initialtread(jr,3)+...
27     (omega*t))*((initialtread(jr,3)+(omega*t))<=pi/2))...
28     +(((initialtread(jr,3)+(omega*t))-pi))*...

```

```

29     ((initialtread(jr,3)+(omega*t))>pi/2)); %beta,
30     index(jr,7)=((atan(index(jr,5)/index(jr,4))*...
31     (index(jr,4)<0))+((atan(index(jr,5)/index(jr,4))*...
32     +pi)*(index(jr,4)>=0))); %gamma
33     gradrot=[cos(omega*t), sin(omega*t);...
34     -sin(omega*t) cos(omega*t)]*[grad(jr,1);grad(jr,2)];
35     index(jr,8)=gradrot(1);
36     index(jr,9)=gradrot(2);
37 end
38 A00 = 0.206*OrderMag;
39 A10 = 0.169*OrderMag;
40 B11 = 0.212*OrderMag;
41 B01 = 0.358*OrderMag;
42 Bm11 = 0.055*OrderMag;
43 C11 = -0.124*OrderMag;
44 C01 = 0.253*OrderMag;
45 Cm11 = 0.007*OrderMag;
46 D10 = 0.088*OrderMag;
47
48 jr=1;
49 ForceX=zeros(1,dim);
50 ForceZ=zeros(1,dim);
51 torque=zeros(1,dim);
52 for jr=1:dim;
53     velocs=index(jr,4:5);
54     B=index(jr,6);
55     G=index(jr,7);
56     grads=index(jr,8:9);
57     coeff=1;
58     alphaX = coeff*scaleFactor*(Cm11*...
59     cos(-2*B+G) + C01*cos(G) + C11*...
60     cos(2*B+G) + D10*sin(2*B));
61     alphaZ = coeff*scaleFactor*(A10*...
62     cos(2*B) + A00 + Bm11*sin((-2*B)+G)...
63     + B01*sin(G) + B11*sin((2*B)+G));
64     if index(jr,3)<=0;

```

```

65     ForceX(jr) = (dot([grads(1);grads(2)],...
66     [velocs(1);velocs(2)])>0)*alphaX*...
67     SegLengths(jr)*TreadWidth*-index(jr,3);
68     ForceZ(jr) = (dot([grads(1);grads(2)],...
69     [velocs(1);velocs(2)])>0)*alphaZ*...
70     SegLengths(jr)*TreadWidth*-index(jr,3);
71     else
72         ForceX(jr) = 0;
73         ForceZ(jr) = 0;
74     end
75     posvec=[index(jr,2)-V(3);index(jr,3)-V(4);0];
76     forcevec=[ForceX(jr);ForceZ(jr);0];
77     torquecross=cross(posvec, forcevec);
78     torque(jr)=torquecross(3);
79 end
80
81 Torquetotal=sum(torque);
82 ForceXTot=sum(ForceX);
83 ForceZTot=sum(ForceZ);
84 Vdot = zeros(4,1);
85 Vdot(1) = (ForceXTot-dbforce)/TireMass;
86 %Accel in x-direction
87 Vdot(2) = (ForceZTot-(TireMass*g))/TireMass;
88 %Accel in z-direction
89 Vdot(3) = V(1); %Velocity in x-dir
90 Vdot(4) = V(2); %Velocity in z-dir
91 Vdot(5) = omega*Torquetotal;
92 % Derivative of Energy dissipated
93
94 end

```

Bibliography

- [1] Li, C., T. Zhang, and D. I. Goldman. "A Terradynamics of Legged Locomotion on Granular Media." *Science* 339.6126 (2013): 1408-412. Web.