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

by

James Slonaker

B.S., Massachusetts Institute of Technology (2015)

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

Master of Science in Mechanical Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2016

Signature redacted

Author ....
Department of Mechanical Engineering
August 5, 2016

Signature redacted

Certified by..
Kenneth Kamrin
Assistant Professor of Mechanical Engineering
Thesis Supervisor

Signature redacted

Accepted by ....
Rohan Abeyaratne
Quentin Berg Professor of Mechanics
Chairman, Department Graduate Committee

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

by

James Slonaker

Submitted to the Department of Mechanical Engineering
on August 5, 2016, in partial fulfillment of the
requirements for the degree of
Master of Science in Mechanical Engineering

Abstract

Inspired by hypotheses of Resistive Force Theory, a general dimensionless form for granular locomotion has been discovered, which instructs how to scale size, mass, and driving parameters to relate dynamic behaviors of different locomotors in the same granular media. These scalings are experimentally confirmed with wheel pairs of various shapes and sizes under many driving conditions in a common sand bed. How the relations may be derived alternatively by assuming Coulombic yielding and how the relations can be augmented to predict wheel performance in different ambient gravities is also explained.

Next, a rotating-flap wheel that consists of a central hub connected to five flaps that can actuate to a certain angle open was designed, built, and tested. Experiments were completed on the wheel by performing a series of tests varying the angle of the flaps and the drawbar force the wheel tows. The results indicate a trend toward higher velocities and powers at larger flap angles. Conversely, with larger drawbar forces the trend indicates lower velocities and higher powers. A MATLAB simulation was also created to model granular locomotion with different wheel shapes, including the rotating-flap wheel. Finally, future work extending the analysis of the rotating-flap wheel to encompass a "smart" wheel that is able to actuate given certain external conditions is discussed.

Thesis Supervisor: Kenneth Kamrin
Title: Associate Professor of Mechanical Engineering

Acknowledgments

There are several people that deserve recognition for the help they gave me throughout this process. First, I would like to thank my advisor Professor Ken Kamrin. His guidance and support have been instrumental to the completion of this work. The knowledge and willingness to help that he showed were a great source of inspiration. Additionally, his advice on personal matters, from career trajectory to whether to attend graduate school, have been truly impactful.

I would also like to thank Carrington Motley, who has served as my right-hand man throughout this process. His knowledge of the lab and design experience have been incredibly helpful. It has been a pleasure working with you.

Next, I would like to thank Carmine Senatore and Professor Karl Iagnemma. Carmine's support in the lab and familiarity with the set-up was crucial to the completion of this work. I also really appreciate Professor Iagnemma's willingness to let us run experiments in his lab space.

Finally, I would like to thank my labmate's Ramin, Hesam, Chen-Hung, Patrick, Sachith, Tyler, and Qiong. I have enjoyed getting to know each of you and have appreciated all the help you have given me.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This thesis builds upon work previously done by the author, in partial fulfillment of the requirements for the degree of Bachelor of Science in Mechanical Engineering at the Massachusetts Institute of Technology, which was published under the same title [1]. Parts of that writing will be reproduced and edited here to ensure accuracy and completeness. Additionally, some of the writing below appears in a publication by Slonaker, Motley, Senatore, Iagnemma, and Kamrin that has been submitted for review [2].

1.1 Background

Due to the complexity of the constitutive behavior of granular media [3] dynamic interactions between grains and solid bodies are challenging to model without resorting to grain-by-grain discrete particle methods. A recent approach to simplify these interactions is Resistive Force Theory (RFT) [4]. RFT is an empirical model utilizing a set of hypotheses about local drag forces to approximate resistance on general solid surfaces moving in granular soils near the surface. RFT was initially developed for viscous drag problems [5], however it has shown surprising effectiveness in granular media, where it has been used to simulate the dynamics of legged reptiles and robots [4], swimming sandfish [6], and the distribution of lift forces on curved submerged bodies in granular beds [7].

This thesis seeks to expand the application of RFT to understand and predict the behavior of wheeled locomotors in 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. Section 1.3 briefly explains the experimental equipment used. Chapter 2 focuses on the derived geometrically general scaling relations for granular locomotion. Chapter 3 explores the design and testing of the rotating-flap wheel.

1.1.1 Resistive Force Theory (RFT)

Granular RFT is concerned solely with computing the resistance on a body moving through a gravitationally loaded bed of grains. It does not attempt to describe the flow or stress fields in the granular media. Its basic premise, which is not actually derived physically, though new efforts are being made to provide its foundations [8], is that a simple form governs the force distribution that grains apply along the leading surface of a moving intruder. In a 3D sand bed, assume a quasi-2D intruder (e.g. an arbitrary driven wheel, per Figure 1-1) whose motion is in the xz-plane, where gravity points in $-\hat{z}$ direction and $z = 0$ represents the free surface. The resistive force $\mathbf{F}^{\text{res}} = (F_x^{\text{res}}, F_z^{\text{res}})$ on the leading surface of the intruder, S, is modeled under

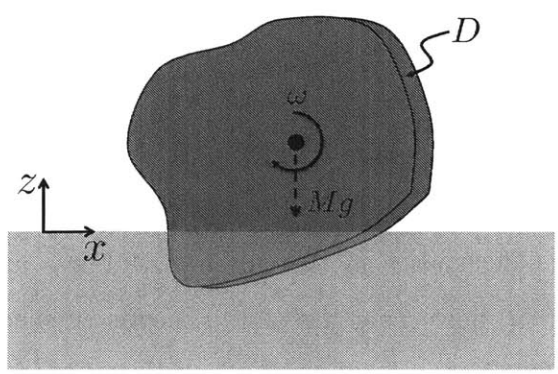


Figure 1-1: General wheel: Driving an arbitrarily shaped wheel of width D and rotational velocity ω, carrying a mass M under gravity g.

RFT to obey

$$\mathbf{F}^{\text{res}} = \int_S \Big(\alpha_x(\beta,\gamma),\ \alpha_z(\beta,\gamma)\Big)\ H(z)\ |z|\ dA_s \tag{1.1}$$

where dA_s is the area of a surface element, β is the surface element's angle of tilt, and γ is the angle of the velocity of the surface element. Both β and γ are measured from the horizontal as shown in Figure 1-2. The Heaviside function, $H(z)$, removes

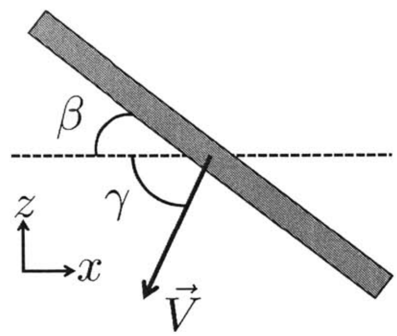


Figure 1-2: The surface element's angle of tilt β and angle of velocity γ are measured from the horizontal.

resistive force on parts of the intruder outside the bed, and the dependence on $|z|$ models added resistance with depth due to gravity. The key constitutive ingredient in the model is the selection of the two force per volume functions α_x and α_z, which is done empirically using experimental force data on small intruding flat plates under various β and γ conditions. The model's effectiveness is somewhat surprising in light of the assumptions made, most notably the assumption that local stress on a surface element is independent of the motion and shape of the other parts of the surface.

By comparing the experimental data for many granular materials it was observed [4] that the $\alpha_{x,z}$ functions have a fairly common shape. Thus, discrete Fourier transforms of these were performed to obtain fixed dimensionless α_x^{gen} and α_z^{gen} functions [4]:

$$\alpha_x^{gen}(\beta,\gamma) = \sum_{m=-1}^{1}\sum_{n=0}^{1}[C_{m,n}cos2\pi(\frac{m\beta}{\pi}+\frac{n\gamma}{2\pi}) + D_{m,n}sin2\pi(\frac{m\beta}{\pi}+\frac{n\gamma}{2\pi})] \tag{1.2}$$

$$\alpha_z^{gen}(\beta,\gamma) = \sum_{m=-1}^{1}\sum_{n=0}^{1}[A_{m,n}cos2\pi(\frac{m\beta}{\pi}+\frac{n\gamma}{2\pi}) + B_{m,n}sin2\pi(\frac{m\beta}{\pi}+\frac{n\gamma}{2\pi})] \tag{1.3}$$

which have generic coefficients as shown in Table 1.1 [4]. The $\alpha_{x,z}$ functions of a

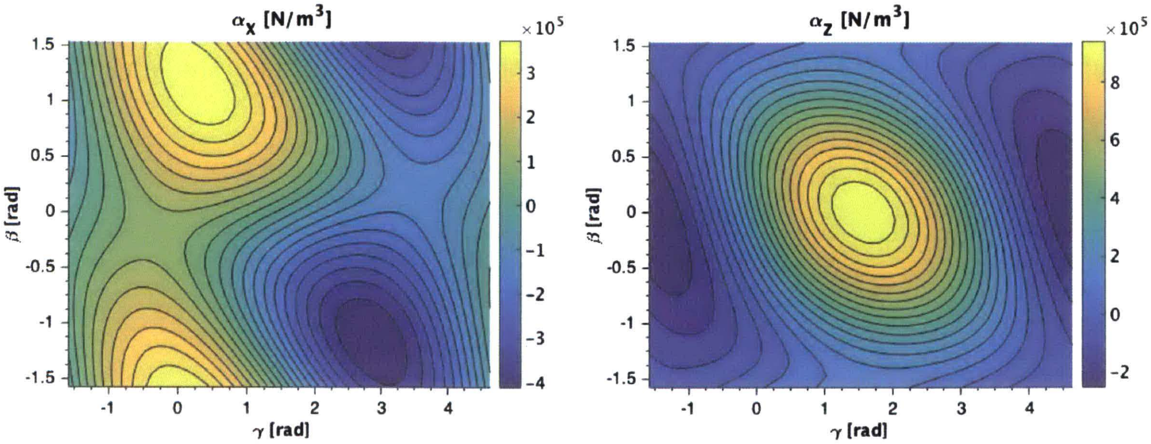
Table 1.1: Generic RFT coefficients.

Generic RFT coefficients.	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

particular material can then be approximated as

$$\alpha_{x,z}(\beta,\gamma) \approx \xi\alpha_{x,z}^{\text{gen}}(\beta,\gamma) \tag{1.4}$$

where ξ is a force/volume quantity denoted the "grain-structure coefficient," which depends on the surface physics of the solid, the mechanical properties of the particular granular media at hand, and the value of gravity. In this way, ξ is the only material parameter needed to execute an RFT calculation in locomotion. A plot of $\alpha_{x,z}$ with $\xi = 1$ is shown in Figure 1-3.



(a) α_x plot. (b) α_z plot.

Figure 1-3: Force per volume plots ($\alpha_{x,z}$), with $\xi = 1$.

By contrast, common engineering terradynamics models like that of Wong and Reece (based on Bekker's work) [9] or the NATO Reference Mobility Model [10] require a much larger number of fit parameters, though they model more than resistive forces. Unlike Bekker's model [11], no knowledge of the material deformation is required for RFT. Similarly, RFT does not require any input of grain size, which differs from other approaches that model nonlocal finite grain size effects [12].

1.1.2 Applying RFT

To calculate the "grain-structure coefficient," Chen Li et al. observed that it can be inferred from a single vertical force measurement [4]. Therefore, to find ξ for a particular material, and thereby the $\alpha_{x,z}$ functions, one only has to measure the stress when the plate is oriented horizontally ($\beta = 0$) and moved vertically down ($\gamma = \pi/2$) [4]. Once the grain-structure coefficient, and thereby the entire resistive force profile, is determined, it can be used to predict the forces experienced by objects interacting with that particular granular material. The resistive forces acting on an intruding body can be approximated by applying the linear superposition principle. 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 $\alpha_{x,z}$ plots to calculate the force on each segment [4]. The total resistive force on the object is then the sum of the individual forces on each smaller linear segment.

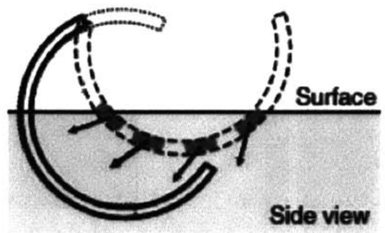


Figure 1-4: "C" shaped leg with linear segments marked darker. Note the image shown here was reprinted from Chen Li et al.'s work [4].

1.2 MATLAB Simulation

To apply RFT to wheeled vehicular locomotion, a MATLAB simulation was created. The entire code for all three wheel shapes is available in Appendix A.

1.2.1 Inputs and Wheel Shapes

The first step in the simulation is to define the inputs and constants necessary. In this 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 inputs include the wheel's rotation speed, mass, initial position, initial velocity, and the gravitational acceleration. Geometric wheel inputs include a characteristic length, the width of the wheel into the plane, the wheel shape, and the number of discretized segments. Additionally if a drawback force, pulley force, or constant spring force is attached to the wheel it is specified. Finally, the grain-structure coefficient for the particular granular material is required.

Multiple wheel shapes were used for the simulation including lug-wheels, superball wheels, and rotating-flap wheels. The lug-wheels have four arms and an elbow half way down their arms bent to a certain angle θ, as shown in Figure 1-5. The character-

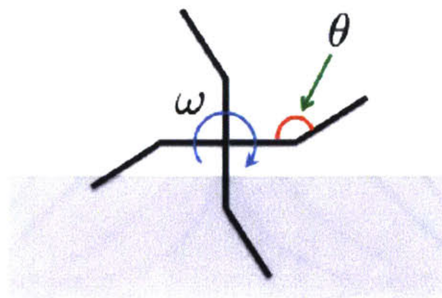


Figure 1-5: General lug-wheel design.

istic length for the lug-wheels is defined as half the total arm length, or the distance from the axle to the bend. Superball wheels are defined by the shape parameter χ, as shown in Equation 1.5.

$$\left|\frac{x}{R}\right|^{2\chi} + \left|\frac{z}{R}\right|^{2\chi} = 1 \tag{1.5}$$

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 $\chi \to \infty$ the shape approaches that of a square again. The actual wheels consist of the superball shapes with a fixed width into the plane. A plot with the shapes corresponding to different χ values is shown in Figure 1-6. The characteristic input length for the superball wheels are their effective radius R. The rotating-flap wheels will be further defined in chapter 3.

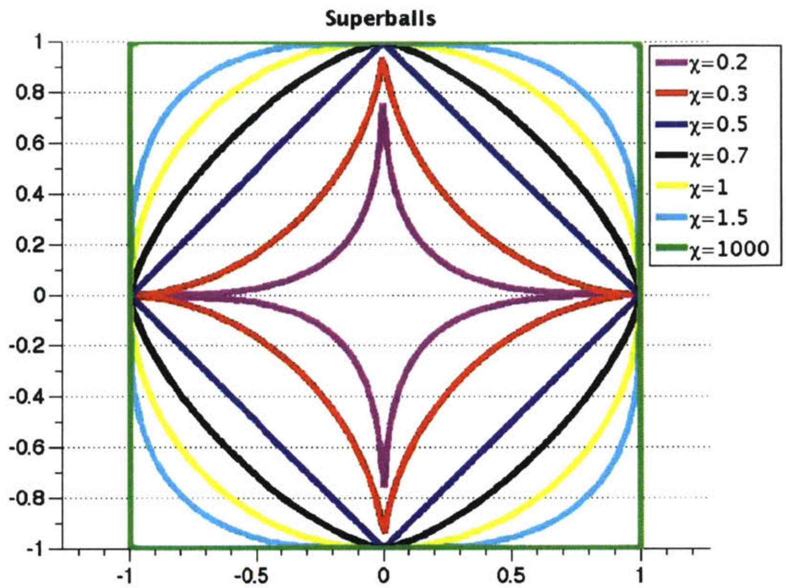


Figure 1-6: "Superball" shapes for different χ values.

1.2.2 Integration of Equations of Motion

With the required inputs, the first step in the simulation is to initialize the wheel design. Based on the geometric 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. To simulate the actual rotation and locomotion of the wheel, the equations of motion are integrated either using the ode45 solver in MATLAB or by directly implementing a Forward Euler method. The integrated function is built of the form of Equation 1.6, 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.

$$\begin{bmatrix} a_x \\ a_z \\ v_x \\ v_z \\ Power \end{bmatrix} = f(\begin{bmatrix} v_x \\ v_z \\ x \\ z \\ Energy \end{bmatrix}, IP) \tag{1.6}$$

Within the actual function, the inputs are used to create a 7 or 9 column index matrix depending on the shape of the wheel (See Appendix A). 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 1.7, 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.

$$\begin{bmatrix} x_i \\ z_i \end{bmatrix} = \begin{bmatrix} x \\ z \end{bmatrix} + \begin{bmatrix} \cos(\omega t) + \sin(\omega t) \\ -\sin(\omega t) + \cos(\omega t) \end{bmatrix} \begin{bmatrix} x_o \\ z_o \end{bmatrix} \tag{1.7}$$

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 1.8 and 1.9.

$$v_{x,i} = \omega(z_i - z) + v_x \tag{1.8}$$

$$v_{z,i} = -\omega(x_i - x) + v_z \tag{1.9}$$

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 $-\pi/2 \leq \beta \leq \pi/2$. Assuming small ω and t, β is calculated as shown in Equation 1.10.

$$\beta_i = \begin{cases} \beta_o + \omega t & \text{if } \beta_o + \omega t \leq \pi/2. \\ \beta_o + \omega t - \pi & \text{if } \beta_o + \omega t > \pi/2. \end{cases} \tag{1.10}$$

If ω is increased to larger values or the time-steps used are much larger, care will need to be taken to rewrite the β calculation to ensure it always stays in the range of $-\pi/2 \leq \beta \leq \pi/2$.

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 1.11.

$$\gamma_i = \begin{cases} \arctan(\frac{v_{z,i}}{v_{x,i}}) & \text{if } v_{x,i} < 0. \\ \arctan(\frac{v_{z,i}}{v_{x,i}}) + \pi & \text{if } v_{x,i} \geq 0. \end{cases} \tag{1.11}$$

Columns 8 and 9 are only used for wheel shapes where the resistive force can only act on one side of the linear segment. For instance the superball wheels will never feel a force on the internal edge of the linear segment because it is a solid object. Therefore, for these shapes, in the initialization, the outward normal vector of each discretized segment is found. 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. Columns 8 and 9, therefore, correspond to the updated outward normal vectors in the x and z directions, n_x and n_z. The vectors are updated using the rotation matrix:

$$\begin{bmatrix} n_{x,i} \\ n_{z,i} \end{bmatrix} = \begin{bmatrix} \cos(\omega t) + \sin(\omega t) \\ -\sin(\omega t) + \cos(\omega t) \end{bmatrix} \begin{bmatrix} n_{x,o} \\ n_{z,o} \end{bmatrix} \tag{1.12}$$

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 1.2 and 1.3. 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 (D), as shown in Equations 1.13 and 1.14.

$$f_x = -\alpha_x L D z_i \tag{1.13}$$

$$f_z = -\alpha_z L D z_i \tag{1.14}$$

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.

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. Assuming no drawbar or pulley force, the outputs can then be solved for as shown in Equation 1.15, where M is the tire mass and τ is the total torque.

$$\begin{bmatrix} a_x \\ a_z \\ v_x \\ v_z \\ Power \end{bmatrix} = \begin{bmatrix} \frac{F_{x,tot}}{M} \\ \frac{F_{z,tot}-Mg}{M} \\ v_x \\ v_z \\ \omega\tau \end{bmatrix} \tag{1.15}$$

1.2.3 Average Velocity and Power

Once the function is integrated 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 cycles 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 1-7.

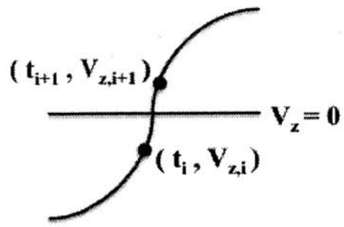


Figure 1-7: Linear interpolation of point where $v_z = 0$.

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 1.16.

$$m_{v_z} = \frac{v_{z,i+1} - v_{z,i}}{t_{i+1} - t_i} \tag{1.16}$$

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 1.17.

$$t_o = \frac{-v_{z,i}}{m_{v_z}} + t_i \tag{1.17}$$

Next, the x-position, x_o, and the energy dissipated, E_o, at that instance can be interpolated as shown in Equations 1.18 and 1.19.

$$x_o = m_x(t_o - t_i) + x_i \tag{1.18}$$

$$E_o = m_E(t_o - t_i) + E_i \tag{1.19}$$

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 1.20 and 1.21.

$$v_{x,avg} = \frac{x_{o,v} - x_{o,u}}{t_{o,v} - t_{o,u}} \tag{1.20}$$

$$P_{avg} = \frac{E_{o,v} - E_{o,u}}{t_{o,v} - t_{o,u}} \tag{1.21}$$

1.3 Experimental Setup

To run physical experiments, a sand bed in the MIT Robotic Mobility Group Lab [13] was used. The test apparatus, as shown in Figure 1-8, consists of a Lexan bin filled with Quikrete medium sand surrounded by an aluminum frame. Attached to the aluminum frame are two low friction rods, which are attached to the carriage and allow for horizontal motion. The carriage is also attached through low-friction vertical rods to the main platform to allow for vertical wheel motion. This allows the wheel to translate freely. The main platform is connected to a motor driven wheel. The wheels used were printed in PLA plastic using a MakerBot 3D printer. Position, torque, velocity, and rotational velocity sensors are attached to the frame to record the relevant data.

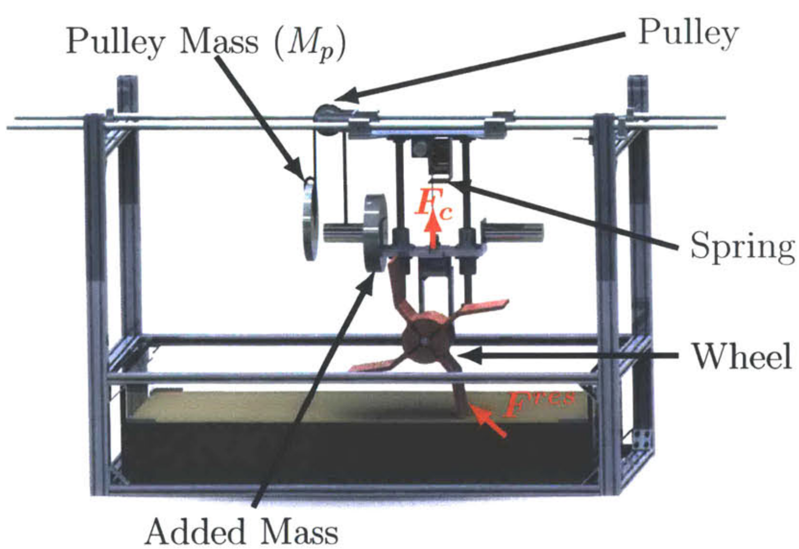


Figure 1-8: Experimental apparatus used in the MIT Robotic Mobility Group Lab [13].

Chapter 2

Geometrically General Scaling Relations for Granular Locomotion

In light of RFT's effectiveness in multiple geometries and its dependence on very few model parameters, a natural question to ask is whether granular RFT, when combined with Newton's laws, produces a set of intruder dynamics possessing scaling behaviors. If these could be identified and validated experimentally, they would provide a physical basis to directly relate different granular locomotion problems in the same soil without performing any simulation, RFT or otherwise. In application, they could be exploited as scaling laws to predict the performance of a "large" locomotor in a sand bed — such as a truck wheel or a tank's caterpillar tracks — by appropriately down-scaled analysis of a smaller locomotor in the same bed. Such a capability could be leveraged in granular design much like scaling techniques in aerodynamics and hydrodynamics.

In this chapter, arbitrarily shaped wheels were studied and a family of geometrically-general scaling laws governing their driving behaviors was derived and experimentally validated. The relations are obtained through analysis of the RFT terradynamical system. As a secondary justification, the same invariances are shown to arise modeling the grains as a frictional continuum. The analysis could be applied to other locomotors with more moving parts, but here the concept is initially tested on solid wheels, which have few internal degrees of freedom, simplifying the analysis.

2.1 Dimensional Analysis

First, dimensional analysis was performed on a generic wheel, such as that shown in Figure 1-1, moving through a granular material that obeys the RFT model. The generic wheel has a dimensionless shape, which we denote by a point-set f, a constant width D into the plane, a characteristic length L that scales the shape f to give the actual wheel cross-section, and a mass M assumed concentrated on the axle. The wheel is given a fixed rotational velocity ω, is acted upon by some gravity g, and interacts with the sand bed through some grain-structure coefficient ξ. The outputs we are interested in are the power expended as the wheel drives in granular media, P, and the wheel's x-translational velocity V, although other outputs could be considered.

Before applying dimensional analysis, note that by using $\alpha_{x,z} = \xi\alpha_{x,z}^{\text{gen}}$ from Equation 1.4, the problem's dependence on ξ and D is seen only through the product ξD. With this and a standard nondimensionalization, the wheel's steady driving limit-cycle is predicted to obey the form:

$$\left[\frac{P}{Mg\sqrt{Lg}}, \frac{V}{\sqrt{Lg}}\right] = \mathbf{\Psi}\left(\sqrt{\frac{g}{L}}t, f, \frac{g}{L\omega^2}, \frac{\xi DL^2}{Mg}\right) \tag{2.1}$$

where, for clarity, $\mathbf{\Psi}$ is a four-input, two-output function as shown. If the gravity, wheel surface texture, and granular media are fixed, g and ξ can be absorbed into the function, giving the reduced form:

$$\left[\frac{P}{M\sqrt{L}}, \frac{V}{\sqrt{L}}\right] = \tilde{\mathbf{\Psi}}\left(\sqrt{\frac{1}{L}}t, f, \frac{1}{L\omega^2}, \frac{DL^2}{M}\right) \tag{2.2}$$

The above forms give the following family of scaling relations: Consider two experiments with the same f, g, and ξ, but one has inputs (L, M, D, ω) and the other has inputs

$$(L', M', D', \omega') = (rL, sM, sr^{-2}D, r^{-1/2}\omega) \tag{2.3}$$

for any positive scalars r and s. Then the corresponding driving cycles should obey:

$$\langle P' \rangle = sr^{1/2} \langle P \rangle \tag{2.4}$$

$$\langle V' \rangle = r^{1/2} \langle V \rangle \tag{2.5}$$

where $\langle \cdot \rangle$ denotes a time-average.

2.2 Connection to Coulomb Plasticity

Although RFT is empirical, interestingly, Equation 2.2 can also be deduced mechanically by considering wheel motion in a 3D bed of ideal Coulomb material [14]. Here, the grains are treated as a rate-independent frictionally yielding continuum. Such a model could be used to predict the entire sand motion field, but instead consider dimensional analysis implications that can be identified without solving for flow. In this model, the wheel inputs remain the same, there is no grain structure coefficient ξ, and there are three granular material parameters: the density of the material ρ, the material's coefficient of internal friction μ, and the coefficient of sliding friction of the wheel-material interface μ_w. Wheels driving through this hypothetical continuum must obey the following dimensionless form:

$$\left[\frac{P}{Mg\sqrt{Lg}}, \frac{V}{\sqrt{Lg}}\right] = \mathbf{\Psi}_{\text{Coul}}\left(\sqrt{\frac{g}{L}}t, f, \frac{g}{L\omega^2}, \frac{D}{L}, \frac{\rho L^3}{M}, \mu, \mu_w\right) \tag{2.6}$$

If the gravity, wheel roughness, and granular media are fixed, the values of g, ρ, μ, and μ_w can be absorbed into the function giving the reduced form:

$$\left[\frac{P}{M\sqrt{L}}, \frac{V}{\sqrt{L}}\right] = \bar{\mathbf{\Psi}}_{\text{Coul}}\left(\sqrt{\frac{1}{L}}t, f, \frac{1}{L\omega^2}, \frac{D}{L}, \frac{L^3}{M}\right) \tag{2.7}$$

This can be further reduced, if it is assumed that granular motion under the wheel is approximately invariant in the out-of-plane dimension. In this case, if the mass M and width D of the wheel are scaled by some C_0, this would be identical to running C_0 copies of the wheel side by side. The resulting power would be $C_0 P$ and the velocity would remain unchanged. From Equation 2.7, this means $\bar{\Psi}_{\text{Coul}}$ is unchanged under such a transformation, which constrains $\bar{\Psi}_{\text{Coul}}$ to depend on M and D only through the ratio D/M, requiring $\bar{\Psi}_{\text{Coul}}$ to depend on its last two inputs only through their product:

$$\left[\frac{P}{M\sqrt{L}}, \frac{V}{\sqrt{L}}\right] = \tilde{\Psi}_{\text{Coul}}\left(\sqrt{\frac{1}{L}}t, f, \frac{1}{L\omega^2}, \frac{DL^2}{M}\right) \tag{2.8}$$

This form is identical to Equation 2.2. Therefore, the scalings implied by RFT can be derived from Coulomb Plasticity if the flow under the wheel is assumed invariant in the out-of-plane dimension, per a wheel with large enough D relative to sinkage.

2.3 Experimental Validation

To test this derived scaling relation, the experimental setup in the MIT Robotic Mobility Lab was used. Additionally, a pulley and constant force spring, as shown in Figure 1-8, were added to allow the effective gravity on the wheel to be varied. This was used to check Equation 2.1 under different selections of g, while keeping ξ held fixed. Lowering the gravity also prevents overload of the wheel motor as mass increases. An SDP/SI Neg'ator constant-force "spring" (F_c = 66.7N) was attached between the carriage and the main platform. A pulley was attached to the carriage platform and a belt around the pulley was attached to the main platform and a mass, M_p.

The spring and pulley were used in tandem to vary the effective gravity on the wheel in a way that ensures the wheel's gravitational masses always agree. This does not change the gravity experienced by the grains; hence the ξ value remains fixed. In view of Figure 1-8, the gravity change can be seen by applying Newton's second law to the main carriage coupled to the hanging weight:

$$\sum F_z = F_{R,z} + F_c + M_p g_e - M_T g_e = (M_T + M_p)\ddot{z} \tag{2.9}$$

The result is that the wheel's translational motion always matches that of a free wheel whose mass is

$$M = M_T + M_p \tag{2.10}$$

and gravity is

$$g = \frac{M_T g_e - M_p g_e - F_c}{M_T + M_p} \tag{2.11}$$

where g_e is earth gravity and M_T is the total mass of the wheel, motor, main platform, and added mass.

Using these forms, Equation 2.1 was tested systematically with a set of 288 experiments involving pairs of cylindrical wheels and four-arm lug wheels (defined in Figure 2-1) of varying size dimensions, mass loadings, and rotation speeds. Both cylindrical wheels are covered in sandpaper to increase interface friction. The lug wheels have four arms and an elbow half way down their lengths bent 150°; work by Chen Li et al. suggests an elbow bend improves wheel effiencency [4]. The interior circle on the lug wheels is for mounting and never came into contact with the sand in any tests. These two wheel shapes were chosen to demonstrate the scaling over two distinct driving motions.

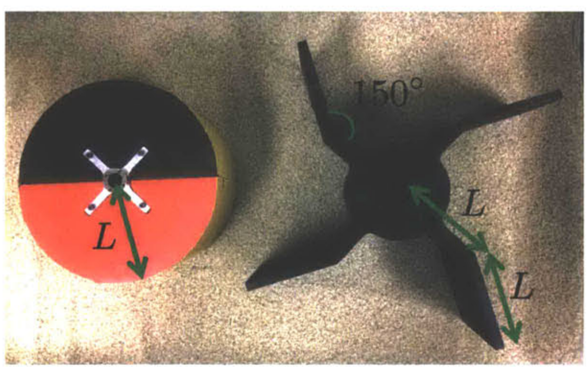
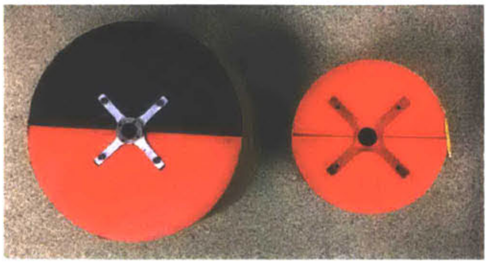


Figure 2-1: The two wheel shapes, f, used in our study: cylinders (left) and lugs (right), and their corresponding definitions of L.

The different test inputs are shown in Table 2.1. In each test, a pair of wheels of the same geometrical family but different size dimensions were driven through the sand; one wheel is 'big' and one is 'small' as denoted with subscripts b and s in the table. Three pairs of wheels were used in total and are shown in Figure 2-2. In each test, the two masses (M_b and M_s) and rotation speeds (ω_b and ω_s) were chosen to

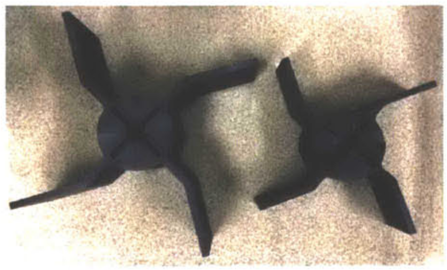
Table 2.1: Experimental tests performed and inputs used. Each row represents a different wheel pair for which 12 combinations of driving parameters were applied; mass and rotational velocity pairs represented as cartesian products.

	Common Parameters [cm], [cm], [m/s^2], [ ]	Driving Parameters [kg] x [deg/s]
Key	$\begin{bmatrix} L_b \\ L_s \end{bmatrix}, \begin{bmatrix} D_b \\ D_s \end{bmatrix}, [g], (\{\text{shape}\})$	$\left(\begin{bmatrix} M_b^1 \\ M_s^1 \end{bmatrix} \begin{bmatrix} M_b^2 \\ M_s^2 \end{bmatrix} \begin{bmatrix} M_b^3 \\ M_s^3 \end{bmatrix} \right) \text{ x } \left(\begin{bmatrix} \omega_b^1 \\ \omega_s^1 \end{bmatrix} \begin{bmatrix} \omega_b^2 \\ \omega_s^2 \end{bmatrix} \begin{bmatrix} \omega_b^3 \\ \omega_s^3 \end{bmatrix} \begin{bmatrix} \omega_b^4 \\ \omega_s^4 \end{bmatrix} \right)$
Pair A	$\begin{bmatrix} 12.50 \\ 8.08 \end{bmatrix}, \begin{bmatrix} 15 \\ 15 \end{bmatrix}, [3.71], (\{\text{Cyl}\})$	$\left(\begin{bmatrix} 35.9 \\ 14.9 \end{bmatrix} \begin{bmatrix} 42.2 \\ 17.6 \end{bmatrix} \begin{bmatrix} 45.7 \\ 19.0 \end{bmatrix} \right) \text{ x } \left(\begin{bmatrix} 14.0 \\ 17.4 \end{bmatrix} \begin{bmatrix} 17.0 \\ 21.2 \end{bmatrix} \begin{bmatrix} 20.0 \\ 24.9 \end{bmatrix} \begin{bmatrix} 23.0 \\ 28.6 \end{bmatrix} \right)$
Pair B	$\begin{bmatrix} 11.25 \\ 7.50 \end{bmatrix}, \begin{bmatrix} 14 \\ 14 \end{bmatrix}, [1.31], (\{\text{Lug}\})$	$\left(\begin{bmatrix} 30.2 \\ 13.4 \end{bmatrix} \begin{bmatrix} 34.9 \\ 15.5 \end{bmatrix} \begin{bmatrix} 39.7 \\ 17.6 \end{bmatrix} \right) \text{ x } \left(\begin{bmatrix} 14.0 \\ 17.1 \end{bmatrix} \begin{bmatrix} 17.0 \\ 20.8 \end{bmatrix} \begin{bmatrix} 20.0 \\ 24.5 \end{bmatrix} \begin{bmatrix} 23.0 \\ 28.2 \end{bmatrix} \right)$
Pair C	$\begin{bmatrix} 11.25 \\ 9.00 \end{bmatrix}, \begin{bmatrix} 14 \\ 10 \end{bmatrix}, [1.31], (\{\text{Lug}\})$	$\left(\begin{bmatrix} 29.3 \\ 13.4 \end{bmatrix} \begin{bmatrix} 34.0 \\ 15.5 \end{bmatrix} \begin{bmatrix} 38.6 \\ 17.6 \end{bmatrix} \right) \text{ x } \left(\begin{bmatrix} 14.0 \\ 15.6 \end{bmatrix} \begin{bmatrix} 17.0 \\ 19.0 \end{bmatrix} \begin{bmatrix} 20.0 \\ 22.4 \end{bmatrix} \begin{bmatrix} 23.0 \\ 25.7 \end{bmatrix} \right)$



(a) Pair A: cylindrical wheels.

(b) Pair B: lug wheels.



(c) Pair C: lug wheels.

Figure 2-2: The three different wheel pairs tested.

ensure the big and small systems have the same dimensionless inputs to $\mathbf{\Psi}$. Hence, each pair of tests corresponds to two experiments that relate through a choice of the r and s scaling parameters described previously. The time-averaged non-dimensional power and velocity:

$$\langle \tilde{P} \rangle = \langle \frac{P}{Mg\sqrt{Lg}} \rangle \tag{2.12}$$

$$\langle \tilde{V} \rangle = \langle \frac{V}{\sqrt{Lg}} \rangle \tag{2.13}$$

were measured when the wheel motion reached cyclic behavior. The power was calculated by multiplying the constant rotation speed by the average torque measured over one cycle. The velocity was measured directly and averaged over one cycle. The wheels were started in the same position, such that the torque and velocity measurements could be averaged over corresponding cycles. In each case, the proposed scaling relation is satisfied if $\langle \tilde{P_b} \rangle = \langle \tilde{P_s} \rangle$ and $\langle \tilde{V_b} \rangle = \langle \tilde{V_s} \rangle$.

The first tests performed were with cylindrical wheels (Pair A). The non-dimensional powers and velocities arising from each of the 12 pairs of driving parameters are plotted in Figure 2-3(a). Four repetitions of the same test are performed each time to obtain useful error-bars. In general the expected trend is strongly observed at low rotational velocity, with some deviation at the highest rotational rate. The next series of tests utilized two four-arm lug wheels (Pair B). Again, the expected scaling relation is observed rather clearly (Figure 2-3(b)). The final series of experiments performed (Pair C) also used four-arm lug wheels, however the widths and lengths of the two wheels were not equal and not proportional. Hence, each test in this series involves two wheels with different lengths, widths, masses, and rotations — all differing by different factors — making this series the most stringent, and arguably most interesting, test of the proposed scaling relation. The agreement is quite strong (Figure 2-3(c)). The best-fit slopes of all six datasets in Figure 2-3 are all within 3% of the predicted value of 1.

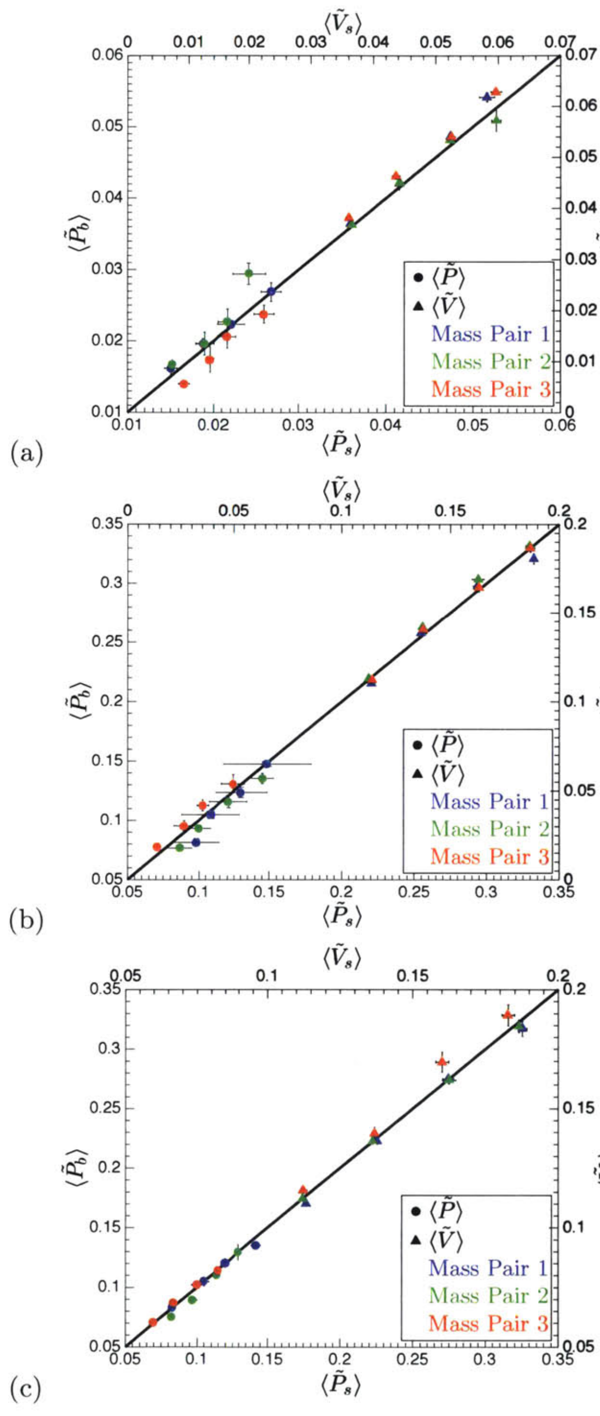


Figure 2-3: Time-averaged non-dimensional power and velocity values for Pair A (a), Pair B (b), and Pair C (c). Each color in each plot represents a different mass pair selection and the corresponding four points of each color represent different rotational velocity pairings. The solid line is the model prediction of $\frac{\langle \tilde{P}_b \rangle}{\langle \tilde{P}_s \rangle} = \frac{\langle \tilde{V}_b \rangle}{\langle \tilde{V}_s \rangle} = 1$.

From Equation 2.1, in addition to the time-average behaviors, the actual time-dependence of these quantities should relate when time is accordingly scaled. Figure 2-4 shows the non-dimensional power plotted against dimensionless time:

$$\tilde{t} = \sqrt{\frac{g}{L}}t \tag{2.14}$$

over the course of a single cycle from two pairs of four-arm lug experiments in Pair C. The predicted scaling agreement between trajectories is observed. The result is non-trivial; the dimensional powers differ by a factor of 2.45 for each pair.

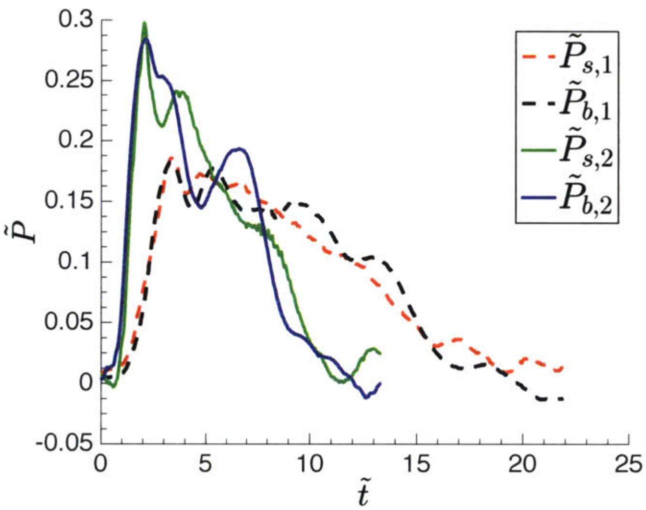


Figure 2-4: Non-dimensional power trajectories for two pairs of lug-wheel experiments over a single cycle (90° of rotation). One pair is shown in solid lines, while the other is shown in dotted lines.

2.4 Potential Extraplanetary Applications

It would be advantageous for future applications to extend the scalings arising from Equation 2.1 so that two experiments in the same sand but with different ambient gravities can be related to each other; e.g. to use an earthbound experiment to predict an extraplanetary experiment. To do this correctly, one requires a theory to explain how ξ varies with gravity. Assuming an ideal Coulomb material, dimensional analysis implies that for a given sand, ξ takes the form

$$\xi = \rho g \hat{\xi}\left(\mu, \mu_w\right). \tag{2.15}$$

Therefore, to the extent that RFT is described by Coulomb Plasticity theory, Equation 2.15 is a good approximation for the grain structure coefficient ξ from RFT.

By substituting Equation 2.15 into Equation 2.1, we obtain:

$$\left[\frac{P}{Mg\sqrt{Lg}}, \frac{V}{\sqrt{Lg}}\right] = \mathbf{\Psi}\left(\sqrt{\frac{g}{L}}t, f, \frac{g}{L\omega^2}, \frac{\rho\hat{\xi}\left(\mu, \mu_w\right)DL^2}{M}\right) \tag{2.16}$$

With this new form, we obtain the following expanded scaling law to relate wheels in the same sand but under two different gravities: Consider two experiments with common f, ρ, μ, and μ_w, where one is described by the inputs (g, L, M, D, ω) and the other by the inputs:

$$(g', L', M', D', \omega') = (qg, rL, sM, sr^{-2}D, q^{1/2}r^{-1/2}\omega) \tag{2.17}$$

for any positive scalars q, r, and s. Then the steady driving cycles of the corresponding outputs should obey:

$$\langle P' \rangle = q^{3/2}r^{1/2}s\langle P \rangle \tag{2.18}$$

$$\langle V' \rangle = q^{1/2}r^{1/2}\langle V \rangle \tag{2.19}$$

When wheel pairs are properly scaled, this relation could be used to predict behaviors in different gravities [15].

2.5 Towing a Drawbar Force

One useful consideration in wheel design is the ability of a wheel to pull a load acting in the opposite direction of motion. This can be described as a constant drawbar force F_d, which acts in the negative horizontal direction. With this new consideration, we can extend the relationship found in Equation 2.16 to include F_d

by adding an additional non-dimensional group:

$$\left[\frac{P}{Mg\sqrt{Lg}}, \frac{V}{\sqrt{Lg}}\right] = \mathbf{\Psi}\left(\sqrt{\frac{g}{L}}t, f, \frac{g}{L\omega^2}, \frac{\rho\hat{\xi}\left(\mu, \mu_w\right)DL^2}{M}, \frac{F_d}{Mg}\right) \tag{2.20}$$

Using this new form, we can expand the scaling law to include the drawbar force. Again, both experiments would have common f, ρ, μ, and μ_w, but one is described by the inputs $(g, L, M, D, \omega, F_d)$ and the other by the inputs:

$$(g', L', M', D', \omega', F_d') = (qg, rL, sM, sr^{-2}D, q^{1/2}r^{-1/2}\omega, sqF_d) \tag{2.21}$$

for any positive scalars q, r, and s. The steady driving cycles of the corresponding outputs should then again obey:

$$\langle P' \rangle = q^{3/2}r^{1/2}s\langle P \rangle \tag{2.22}$$

$$\langle V' \rangle = q^{1/2}r^{1/2}\langle V \rangle \tag{2.23}$$

2.6 Inertial and Gravitational Masses

In the discussed experimental work from Section 2.3, a pulley and constant force spring were used to ensure the gravitational and vertical inertial masses were scaled properly. This should be sufficient for scaling purposes in all cases in which the material used obeys Coulomb-Plasticity and the mass of the wheel and pulley are assumed to be much larger than the mass of the top horizontal carriage. However, if that assumption is not valid, then the horizontal inertial mass of the wheel would also have to be scaled properly to truly replicate the scaling of a free wheel. Therefore, it is useful for future applications to understand the proper scaling mechanism.

Recalling Figure 1-8, and assuming a drawbar mass is attached, when the wheel translates in the vertical direction the main platform and pulley mass (M_p) translate as well. However, when the wheel translates in the horizontal direction, not only does the main platform move, but also the drawbar mass (M_{DB}) and the top carriage, which the the low friction vertical rods attach to. Therefore, to determine what the acceleration in the x and z direction should be, we can use free body diagrams to calculate the total forces on four sections: the pulley mass, the drawbar mass, the wheel and main platform, and the top carriage and vertical bars. The mass of the top carriage and vertical bars is denoted M_c and the total mass of the wheel, motor, main platform, and added mass is denoted M_T.

Using the sum of the forces in the x and z directions for all four free body diagrams, we can solve for the total acceleration of the wheel in both directions:

$$a_x = \frac{F_x^{res} - M_{DB}g}{M_T + M_{DB} + M_c} \tag{2.24}$$

$$a_z = \frac{F_z^{res} - M_T g + M_p g + F_c}{M_T + M_p} \tag{2.25}$$

where F_c is again the constant force of the spring, g is the gravity, and F_x^{res} and F_z^{res} are the resistive forces in the x and z directions. As is seen in the denominator of Equations 2.24 and 2.25 the inertial masses in each direction are actually different. This is because the top carriage is not allowed to move vertically, since it is attached to the horizontal rods. Additionally, the drawbar force and pulley force only act in the horizontal and vertical directions respectively. Therefore, at high speeds when inertial forces can have large impact, care will need to be taken to scale both the horizontal and vertical inertial masses to truly replicate the scaling of a free wheel.

2.7 Discussion

We have proposed and experimentally validated a set of invariances in granular locomotion for the case of rigid, arbitrarily shaped wheels, which was initially obtained by analyzing RFT system dynamics. The scaling analysis has been reconciled with Coulomb Plasticity and an extension has been proposed that could relate locomotion processes in different ambient gravity.

Like RFT itself, our forms neglect rate-sensitivity of the material, which is known to exist (i.e. the $\mu(I)$ rheology [16, 17]). For more rapidly spinning wheels this effect could add another degree of complexity to the scaling, however it is possible the same form will work in a range of larger speeds, since modifications for rate-dependency change the solution only minimally in certain cases [18]. For example, robots that run on sand using "c-legs" [4] move many times their body-length per second yet remain well-described by rate-independent RFT. Moving forward, it will be important to further test this scaling relation at higher speeds and with a greater variability in the inputs. Work is currently being done at Dan Goldman's Complex Rheology and Biomechanics Lab at Georgia Tech to build a test rig to further test the scaling relation over larger input ranges.

Other considerations not accounted for are the effect of internal texture variables within the deforming granular system and nonlocal effects due to finite grain size [12]. The former suggests that a more complex scaling relationship may be needed to go beyond monotonic driving, e.g. oscillatory wheel motion, but the latter is likely to matter only as wheel feature size competes with the grain size [19, 20]. Though rigid wheels were studied here, the scaling could be extended to more complex locomotion, such as undulating self-propulsion, with more moving parts by adding additional non-dimensional groups for each new degree of freedom. One very interesting case to study would be the application of this scaling to tank treads. Testing these extended scaling laws should be important future work.

Chapter 3

Rotating-Flap Wheels

In addition to deriving and experimentally testing the proposed scaling law, another research focus was on the optimization of wheel shapes for granular locomotion. One idea for this work was to develop a "smart" wheel that is able to deform or actuate to an optimal shape depending on the local conditions. Due to RFT's simplicity, it was believed that the wheels motion could be simulated over many conditions. Using this simulation data, it could later be programmed into the physical wheel's control such that when a certain condition arose, such as a drawbar force or required velocity, it could respond accordingly. This could allow a wheel to travel efficiently over many different conditions and for many desired outputs.

3.1 Design and Production

Many iterations of the "smart" wheel design were considered [21]. The first idea was to create a wheel that could change from one superball shape to another. This however proved difficult to mechanically design as it needed to actuate in the in-plane dimension while keeping the out of plane thickness constant. Therefore proposed actuation methods using air or gas pressure were infeasible as their isotropic nature would result in deformations in all directions. Other actuation methods, including linear actuators and timing belts, and wheel designs, such as a rotating or extendable lug wheels, were considered [21]. In an effort to keep moving parts to a minimum and the production of the wheel feasible most of these designs were rejected.

Ultimately, the rotating-flap wheel design was chosen. The rotating-flap wheel consisted of a central cylindrical wheel hub, with five flaps attached to the hub as shown in Figure 3-1. Each flap consisted of one-fifth the total circumference of the

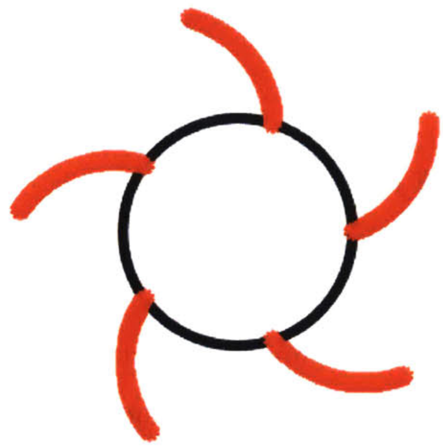


Figure 3-1: Rotating-flap wheel design shown with flaps colored red and actuated open to 70°.

wheel hub and was attached to the hub at one end. To actuate the flaps open a gear system was used consisting of a large central gear, with radius R_b, that was connected to a motor. The large central gear meshed with five smaller gears, with radii R_s that where attached to each flap. In this way, when the motor actuated the central gear a certain rotational distance θ_b, it caused the flaps to rotate a distance:

$$\theta_s = \frac{R_b}{R_s}\theta_b \tag{3.1}$$

allowing full control of the flaps location through one motor.

Once the "smart" wheel design was chosen and fully dimensioned, actual production of the wheel began. The large central gear and five smaller gears were ordered directly, though the central gear was too thick, so a lathe was used to reduce its size. A motor with sufficient torque and position accuracy, as well as a motor controller, was ordered through Maxon Motors. The hub, all five flaps, and the connection piece used to attach the motor to the hub were all 3D printed in PLA plastic using the same MakerBot as the previous experiments. Five axles for each flap's rotation were used. Each axle had two holes drilled and tapped on each end to attach to the flaps via a screw and one hole drilled in the middle of the axle to attach to the small gear through a screw. The same "cross" mounting system was used to attach the wheel to the experimental setup in the MIT Robotic Mobility Lab. The fully assembled rotating-flap wheel is shown in both open and closed configurations in Figure 3-2 for reference.

(a) Closed view of the rotating-flap wheel.

(b) Open view of the rotating-flap wheel.

(c) Isometric view of the rotating-flap wheel.

Figure 3-2: Multiple views of the rotating-flap wheel [21].

3.2 Experimentation

Once the rotating-flap wheel was assembled, experiments on the wheel could be conducted. First, to increase traction sand paper was added to every flap, similar to the cylindrical wheels tested previously. Second, the Maxon Motor came with its own software for position control, which used the motor's internal Hall sensor. Unfortunately, for correct data to be collected the control had to be integrated into the existing LabView environment already in use in the lab. Therefore, the data collection and control software Read&Display_MER_RIG.vi currently in use had to be altered [22]. The new motor controller was connected to the desktop via a USB source, which was determined to be sufficient for the necessary control. Software that integrated the motor into LabView was used and is shown in Figure 3-3.

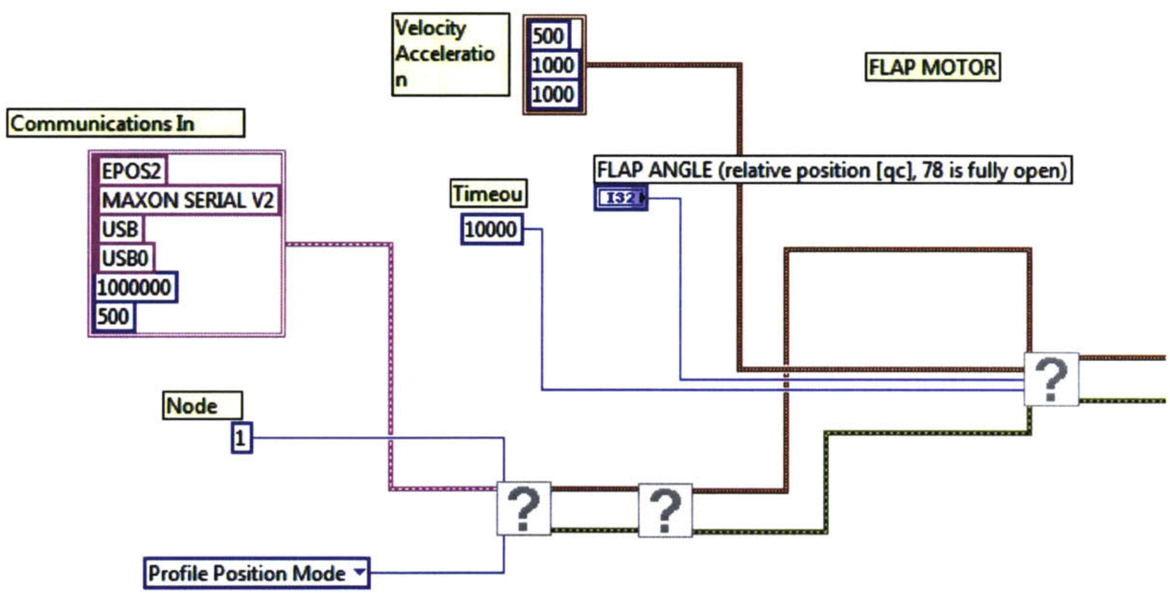


Figure 3-3: LabView code used to integrate the Maxon motor into the LabView system.

The motor used was actuated to a certain position by measuring in quad counts. This was the precision of the hall sensor on which the position control was based. Due to the gear ratio of the rotating-flap wheel, moving the motor 78 qc completely opened up the flap angles to 90°. While the LabView data collection system was running, additional code had to be written to keep the motor engaged once it had reached its desired flap position. This new code is shown in Figure 3-4. This code kept

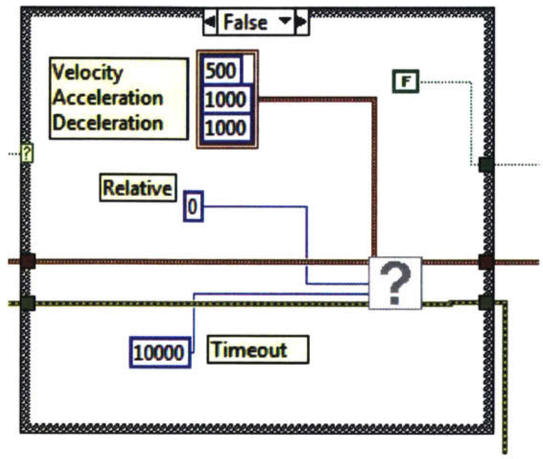


Figure 3-4: LabView code used to keep the motor engaged in the proper position.

the relative position of the motor set to 0 as long as the data collection system was running, ensuring it stayed engaged and had proper torque reactions to the resistive force. All of this new control software was saved as R&D_MER_Flap.vi and used to perform all experiments. The front plate of the control software was also altered to allow the user to input the desired flap position. This is shown in Figure 3-5, where a box labeled "Flap Angle" is shown highlighted. This allows the user to input the flap angle in quad count units with again 78 qc equal to 90° open. Finally, the motor controller had to be connected to a DC power source that could supply the necessary 27.2 volts.

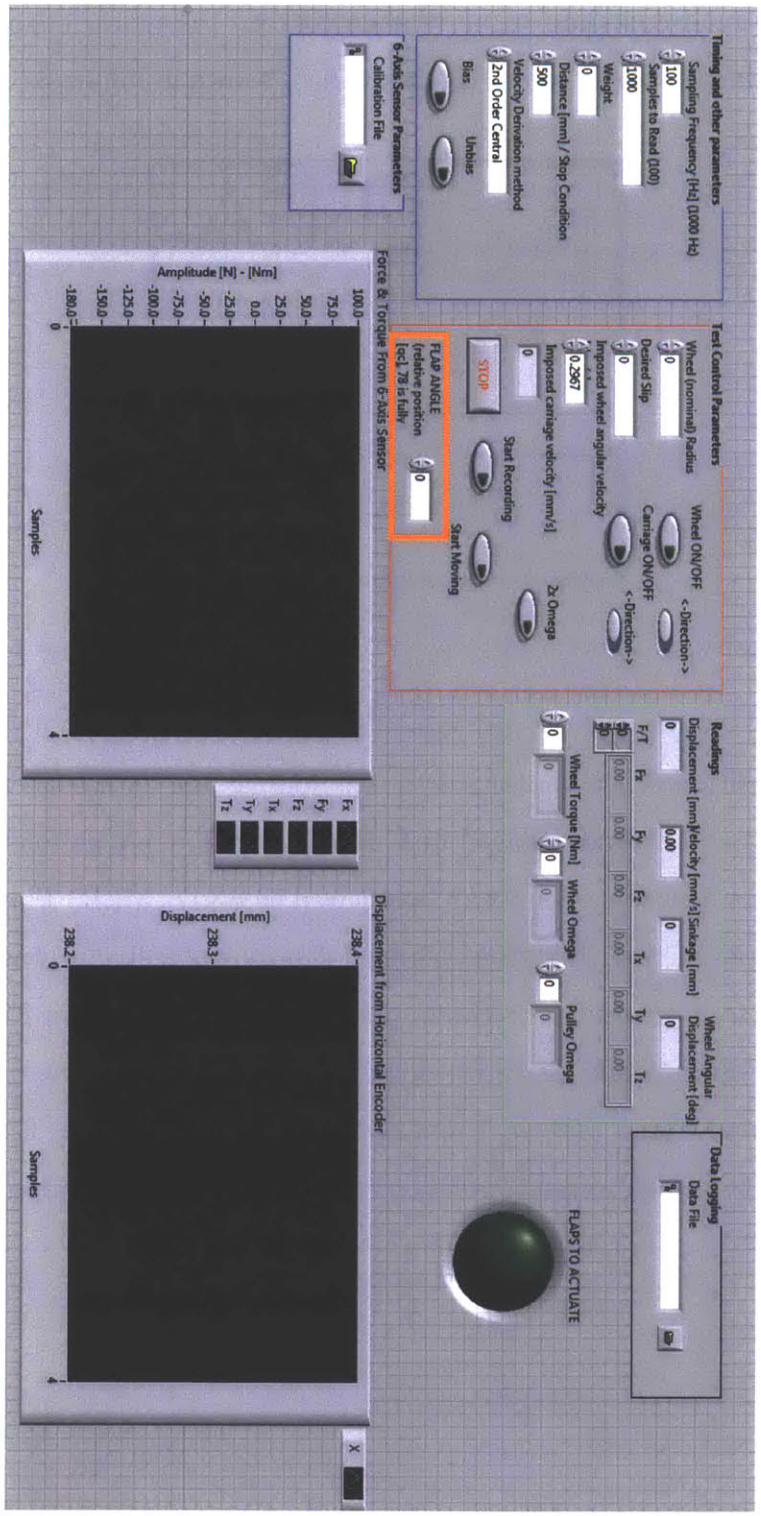
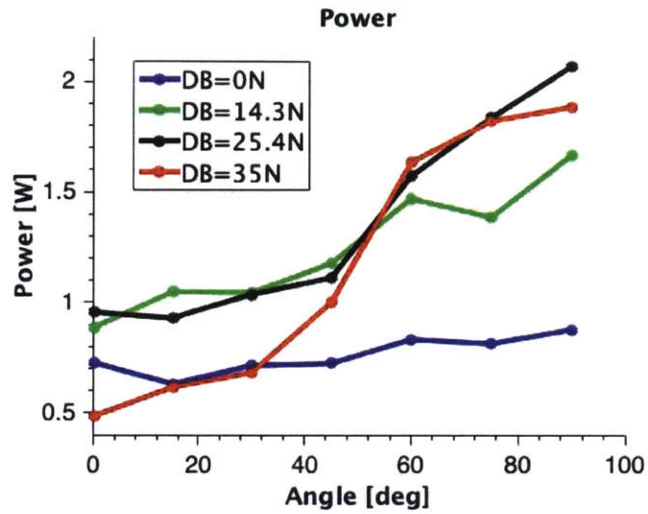
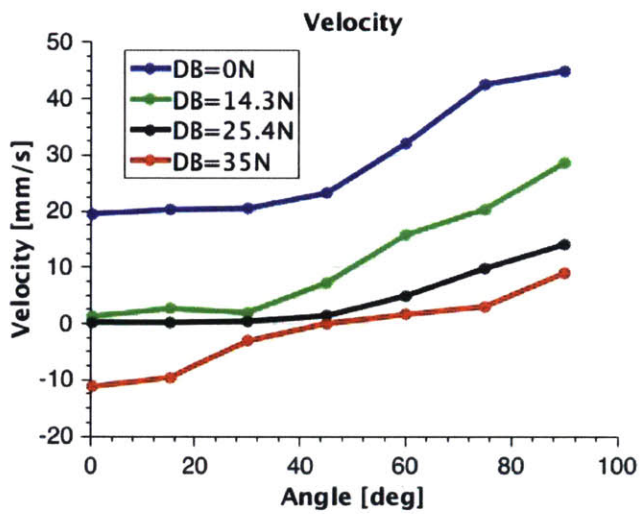


Figure 3-5: Front panel of LabView code with new input to change the flap angle.

The actual experiments performed consisted of testing the rotating-flap wheel at various flap angles open and various drawbar forces. The idea was to determine if whether the wheel was able to achieve higher velocities or more efficient motion when the flaps were actuated further out. To do so, the flap was tested at flap configurations of 0, 15, 30, 45, 60, 75, and 90 degrees open. Four different drawbar forces were tested as well: 0, 14.3, 25.4, and 35 newtons. The wheel was rotated at a constant 20 degrees/sec for each test. Again the average power and velocity were recorded over a single cycle. The average power was calculated by multiplying the constant rotation speed times the average torque measured over a cycle, while the average velocity was measured directly using the linear encoder. The experimentally measured power and velocity measurements are shown in Figure 3-6.



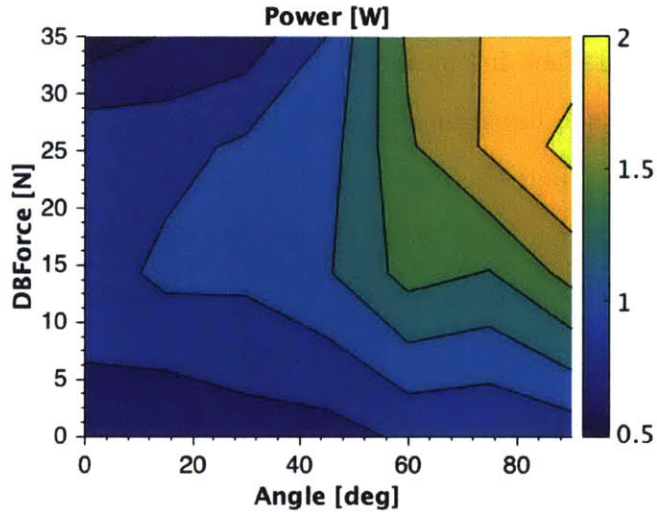
(a) Experimentally measured power vs. flap angle.



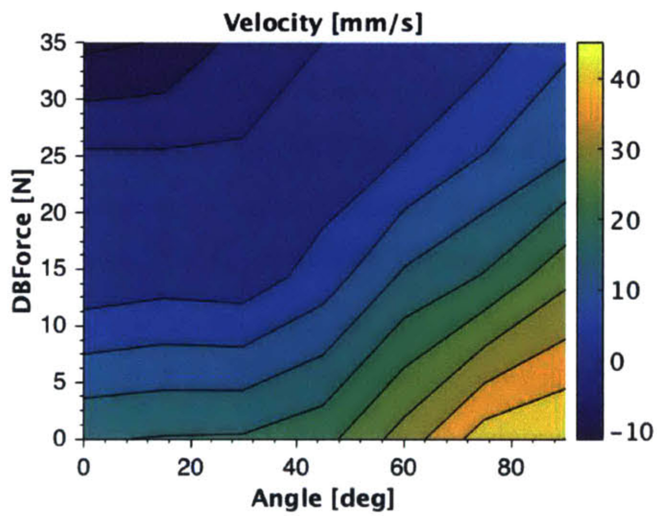
(b) Experimentally measured velocity vs. flap angle.

Figure 3-6: Experimentally measured power and velocity of rotating-flap wheel. Different drawbar forces are plotted in different colors.

Additionally, the results are shown as contour plots in Figure 3-7. As can be seen the general trend in the velocity measurements is followed quite nicely: for a given drawbar force if the flap angle increases the velocity increases and conversely for a given flap angle if the drawbar force decreases the velocity increases. Note at some



(a) Power contour plot.



(b) Velocity contour plot.

Figure 3-7: Power and Velocity contour plots for rotating-flap experiments.

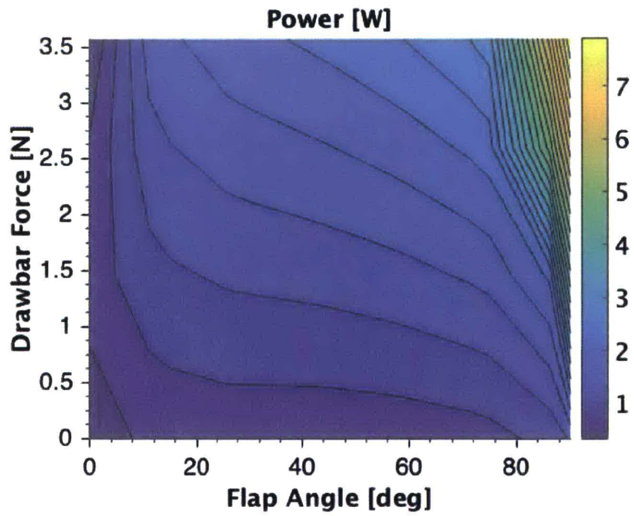
instances, such as when the wheel flaps are fully closed (flap angle = 0°) and towing the highest drawbar, the velocity is actually negative as the wheel is pulled backwards. The general trend in the power is that the larger the flap angle and the drawbar force the larger the power. This is not, however, perfectly observed. For instance, when the wheel is pulled backwards with a negative velocity, the power actually decreases.

One cause of potential error in the experiments, however, is that, due to the meshing of the central and external gears, tolerance issues in the hub resulting from the 3D printing, and slack in the motor, there is some play in the actual flap angle. Therefore, although the flaps might be actuated to a certain angle, when they come into contact with the sand the real angle is a slightly less value as the flaps collapse somewhat. An effort has been made to measure this effect using image processing and it is estimated that the real angle is $4°$ to $7°$ less than the intended.

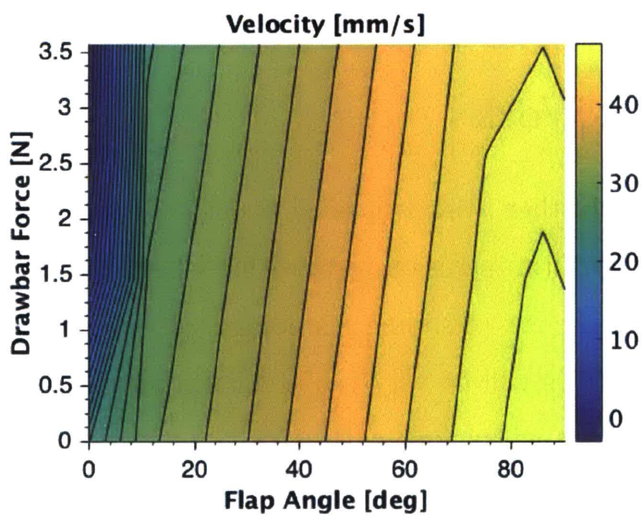
3.3 Simulation

In addition to the physical experiments, a MATLAB simulation was run to model the exact same test parameters. The simulation is exactly that as described in Section 1.2, except that the shape input is a rotation-flap wheel. The simulation code is shown in appendix A.5 and A.6. The hub is assumed to never come into contact with the sand, and thus only the flaps are modeled. This is a good approximation as the flaps tend to dig sand out of the way such that even if the hub is below the sand level it is never in contact with any grains. Similarly, resistive force is assumed to only act on the front side of the flaps as the backside either is never in contact with grains or is moving in the direction opposite that of its velocity such that it would have no resistive force. This is modeled using the outward normal vectors described in Section 1.2.

All parameters were input identical to the physical experiments including the mass, rotation speed, pulley force, constant spring force, drawbar force, and all geometric considerations. The grain-structure coefficient ξ was set at 2.06 based on intrusion experiments performed by Carmine Senatore. The simulations were run for sufficient duration to ensure steady state, before power and velocity averages were calculated. The results of the simulations are shown in Figure 3-8.



(a) Power contour plot.



(b) Velocity contour plot.

Figure 3-8: Power and Velocity contour plots for rotating-flap simulations.

In both cases, the same general trend that was observed in the physical experiments is seen here. For a given drawbar force as the flap angle increases the velocity increases and for a given flap angle as the drawbar force decreases the velocity increases. For the power, in general as the flap angle and drawbar force increase the power increases. Despite observing the same general trend as the experiments, the actual magnitude of the power and velocity values seem to be much larger for the simulation.

There are several possible explanations for this. First, the grain-structure coefficient used was based off of intrusion measurements using an aluminum plate. Therefore, it is quite possible that the rotating-flap wheels covered in sandpaper have a different coefficient of sliding friction of the wheel-material interface μ_w than smooth aluminum. Second, the simulation does not take into account some of the geometric irregularities of the actual wheel, such as the protruding small gears that come into contact with the sand occasionally. Finally, there could also be a bug in the simulation code that has yet to be found. RFT has been used with great accuracy to predict physical experiments as shown by Chen Li et al., so it was expected to produce similar results here [4].

3.4 Further Work

Moving forward, further work is needed to investigate this research. First, effort should be made to perform intrusion tests with a PLA plate covered in sandpaper to mimic the actual coefficient of sliding friction of the wheel-material interface of the rotating-flap wheel. This will be very useful for further simulation to determine if it is possible to model this wheel type with RFT.

Next, it would be very interesting to actually code the contour plots from Figure 3-7 into the control of the rotating-flap wheel. Thus, one could perform an experiment where they set a desired velocity and changed the drawbar force as the wheel was moving. Each time the drawbar force was changed, it would be sensed by the load cell on the experimental setup and could be used to actuate the wheel to the appropriate flap angle for that drawbar to achieve the desired velocity. Thus, this would actually achieve the intended goal of the "smart" wheel, to actuate as it is moving by responding to surrounding conditions. This could lead to overall more efficient motion for a vehicle utilizing the wheel.

Appendix A

Simulation Code

A.1 Lug Wheel Animation Code

```
%% Simulation of 4-spoke tire
% x is defined as positive to the right
% z is defined as positive upwards

%% Clear Everything
commandwindow
close all
clear all
clf
clc
format long

%% Inputs
theta=150*pi/180; % [radians]
TreadWidth = 0.07; % [m], in plane
TreadLength=0.085; % [m]
TireAxleInitCoord = [0,0]; %[x,z], [m]
omega=27*pi/180; % [rad/s]
init_velocity = [0,0]; %[vx, vz], [m/s]
Fspring=6.80389; %Spring force [kg]
```

```
TireMass = (70/9.81); % [kg]
duration = 25;% [sec], length of simulation
savepics=0; %make 1 to create pics, and 0 for no pics
anim=1; %1 there is an animation, and 0 for no animation
fps=30; % Animation frames per second
dbforce=0;% Drawback Force [N]
g = 9.81; % [m/s^2], gravitational acceleration
scaleFactor = 2.576*(g/9.81);
elbows=1;
NumTreads = 4;

%% Set Constants
NumSegs = 45*(elbows+1);
NumPieces=NumSegs*NumTreads;
SegLength = (TreadLength*(elbows+1))/NumSegs;  % [m]
OrderMag = 10^6;

A00 = 0.206*OrderMag;
A10 = 0.169*OrderMag;
B11 = 0.212*OrderMag;
B01 = 0.358*OrderMag;
Bm11 = 0.055*OrderMag;
C11 = -0.124*OrderMag;
C01 = 0.253*OrderMag;
Cm11 = 0.007*OrderMag;
D10 = 0.088*OrderMag;

%% Intitialization
ForceXs=[];
ForceZs=[];
XPos=[];
ZPos=[];
j=1:2:2*(NumSegs/(elbows+1));
i=1;
startingx=zeros((NumSegs/(elbows+1)),1);
startingz=zeros((NumSegs/(elbows+1)),1);
```

```
for i=1:(NumSegs/(elbows+1));
    startingx(i)=(j(i)/(2*((NumSegs)/(elbows+1))))*TreadLength;
end
xposadd=TreadLength;
zposadd=0;
startingx1=[];
startingz1=[];
for j=1:elbows
    startingx1(((j-1)*(NumSegs/(elbows+1)))+1:(j*(NumSegs/(elbows+1))),1)=...
    ((startingx.*cos(j*(pi-theta)))+xposadd);
    startingz1(((j-1)*(NumSegs/(elbows+1)))+1:(j*(NumSegs/(elbows+1))),1)=...
    (startingx.*sin(j*(pi-theta)))+zposadd;
    xposadd=xposadd+(TreadLength*cos(j*(pi-theta)));
    zposadd=zposadd+(TreadLength*sin(j*(pi-theta)));
end
startingz=[startingz; startingz1];
startingx=[startingx; startingx1];
betainit=zeros(NumSegs,1);
for jr=1:1:NumSegs/2;
    betainit(jr)=0;
    betainit((NumSegs/2)+jr)=-atan(startingz(end)/...
    (startingx(end)-TreadLength));
end
initialtread=zeros(NumPieces,3);
j=1;
for j=0:NumTreads-1;
    initialtread((1+j*NumSegs):(NumSegs+j*NumSegs),1)=...
    (cos(j*2*pi/NumTreads).*...
    (startingx)+(sin(j*2*pi/NumTreads).*(startingz)));
    initialtread((1+j*NumSegs):(NumSegs+j*NumSegs),2)=...
    (-sin(j*2*pi/NumTreads).*...
    (startingx)+(cos(j*2*pi/NumTreads).*(startingz)));
    origx(j+1)=(cos(j*2*pi/NumTreads)*(TreadLength))+TireAxleInitCoord(1);
    origz(j+1)=(-sin(j*2*pi/NumTreads)*(TreadLength))+TireAxleInitCoord(2);
    initialtread((j*NumSegs+1):((j+1)*NumSegs),3)=((betainit+(j*pi/2)).*...
    ((betainit+(j*pi/2))<=pi/2))+...
```

```
    (((betainit+(j*pi/2))-pi).*((betainit+(j*pi/2))>pi/2));
end


xs=(origx-TireAxleInitCoord(1))';
zs=(origz-TireAxleInitCoord(2))';

%%Key
% Startingx and startingz are midpts of segment on one tread
% initialtread is midpts of segments
% origx and origz are 4 elbow locations
% xs and zs are origx and origz minus the location of the axle initial
% position

allx=[];
allz=[];
allvx=[];
allvz=[];
newx=[];
newz=[];

%% Function
Vo = zeros(1,5);
Vo(1) = init_velocity(1); % initial velocity in x-dir
Vo(2) = init_velocity(2); % initial velocity in z-dir
Vo(3) = TireAxleInitCoord(1); % initial axle position in x
Vo(4) = TireAxleInitCoord(2); % initial axle position in z
Vo(5) = 0; % initial dissipated Power

options = odeset('RelTol',1e-4,'AbsTol',1e-7);
odefix = @(t, V) FunctionVODEFourBar(t, V, TireMass, TreadWidth, SegLength,...
 omega, NumPieces, NumTreads, OrderMag, scaleFactor,...
  initialtread, g, dbforce, Fspring);
[TOUT,VOUT] = ode45(odefix,[0 duration],Vo,options);

%% Average Power and Velocity measurements
```

```
pdpos=find((diff(sign(VOUT(:,2))))==2);

startavg=3;
endavg=6;

mz1=(VOUT(pdpos(endavg)+1,2)-VOUT(pdpos(endavg),2))/...
(TOUT(pdpos(endavg)+1)...
-TOUT(pdpos(endavg))); %slope of (t,z)
t01=((1/mz1)*-VOUT(pdpos(endavg),2))+TOUT(pdpos(endavg));
mv1=(VOUT(pdpos(endavg)+1,3)-VOUT(pdpos(endavg),3))/...
(TOUT(pdpos(endavg)+1)...
-TOUT(pdpos(endavg))); %slope of (t,vx)
mp1=(VOUT(pdpos(endavg)+1,5)-VOUT(pdpos(endavg),5))/...
(TOUT(pdpos(endavg)+1)...
-TOUT(pdpos(endavg))); %slope of (t,pow)
vxfix1=(mv1*(t01-TOUT(pdpos(endavg))))+VOUT(pdpos(endavg),3);
powfix1=(mp1*(t01-TOUT(pdpos(endavg))))+VOUT(pdpos(endavg),5);

mz2=(VOUT(pdpos(startavg)+1,2)-VOUT(pdpos(startavg),2))/...
(TOUT(pdpos(startavg)+1)...
-TOUT(pdpos(startavg))); %slope of (t,z)
t02=((1/mz2)*-VOUT(pdpos(startavg),2))+TOUT(pdpos(startavg));
mv2=(VOUT(pdpos(startavg)+1,3)-VOUT(pdpos(startavg),3))/...
(TOUT(pdpos(startavg)+1)...
-TOUT(pdpos(startavg))); %slope of (t,vx)
mp2=(VOUT(pdpos(startavg)+1,5)-VOUT(pdpos(startavg),5))/...
(TOUT(pdpos(startavg)+1)...
-TOUT(pdpos(startavg))); %slope of (t,pow)
vxfix2=(mv2*(t02-TOUT(pdpos(startavg))))+VOUT(pdpos(startavg),3);
powfix2=(mp2*(t02-TOUT(pdpos(startavg))))+VOUT(pdpos(startavg),5);

vxavg=(vxfix1-vxfix2)/(t01-t02) % [m/s]
Power=(powfix1-powfix2)/(t01-t02) % [W]

%% Animation Initialization
u=find(diff(sign(diff(mod((TOUT),1/fps))))==2);
```

```
TOUTfps=TOUT(u);
VOUT=VOUT(u,:);

Torque=[];
kd=1;
for k=1:1:length(TOUTfps)

    index = zeros(NumPieces,7); %Local number, x,z,vx,vz,Beta,gamma
    index(:,1) = [1:NumPieces];
    jr=1;

    V(1)=VOUT(k,1);
    V(2)=VOUT(k,2);
    V(3)=VOUT(k,3);
    V(4)=VOUT(k,4);
    t=TOUTfps(k);
    timetot(kd)=t;

    for jr=1:NumPieces;
        index(jr,2)=V(3)+((cos(omega*t)*(initialtread(jr,1)))+...
        (sin(omega*t)*...
        (initialtread(jr,2))));
         %New X position using rotation matrix
        index(jr,3)=V(4)+((-sin(omega*t)*(initialtread(jr,1)))+...
        (cos(omega*t)*...
        (initialtread(jr,2))));
        %New Z position using rotation matrix
        index(jr,4)=V(1)+(omega*(index(jr,3)-V(4)));
        %New velocity in x-dir
        index(jr,5)=V(2)+(-omega*(index(jr,2)-V(3)));
        %New velocity in z-dir
        index(jr,6)=((initialtread(jr,3)+(omega*t))*((initialtread(jr,3)+...
        (omega*t))<=pi/2))+...
        (((initialtread(jr,3)+(omega*t)-pi))*((initialtread(jr,3)+...
        (omega*t))>pi/2)); %beta,
        index(jr,7)=((atan(index(jr,5)/index(jr,4))*(index(jr,4)<0))+...
```

```
        ((atan(index(jr,5)/index(jr,4))+pi)*(index(jr,4)>=0))); %gamma
    end

    A00 = 0.206*OrderMag;
    A10 = 0.169*OrderMag;
    B11 = 0.212*OrderMag;
    B01 = 0.358*OrderMag;
    Bm11 = 0.055*OrderMag;
    C11 = -0.124*OrderMag;
    C01 = 0.253*OrderMag;
    Cm11 = 0.007*OrderMag;
    D10 = 0.088*OrderMag;

    jr=1;
    ForceX=zeros(1,NumPieces);
    ForceZ=zeros(1,NumPieces);
    torque=zeros(1,NumPieces);
    for jr=1:NumPieces;
        B=index(jr,6);
        G=index(jr,7);

        alphaX = scaleFactor*(Cm11*cos(-2*B+G) + C01*cos(G) +...
         C11*cos(2*B+G) + D10*sin(2*B));
        alphaZ = scaleFactor*(A10*cos(2*B) + A00 + Bm11*sin((-2*B)+G) +...
         B01*sin(G) + B11*sin((2*B)+G));

        if index(jr,3)<=0;
            ForceX(jr) = alphaX*SegLength*TreadWidth*-index(jr,3);
            ForceZ(jr) = alphaZ*SegLength*TreadWidth*-index(jr,3);
        else
            ForceX(jr) = 0;
            ForceZ(jr) = 0;
        end
        posvec=[index(jr,2)-V(3);index(jr,3)-V(4);0];
        forcevec=[ForceX(jr);ForceZ(jr);0];
        torquecross=cross(posvec, forcevec);
```

```
        torque(jr)=torquecross(3);

    end


    Torquet=sum(torque); %positive is z axis out of screen/page
    Torque=[Torque Torquet];
    ForceXs=[ForceXs ForceX];
    ForceZs=[ForceZs ForceZ];
    ForceXTot=sum(ForceX);
    ForceZTot=sum(ForceZ);

    allx=[allx; index(:,2)]; %x positions of each tread
    allz=[allz; index(:,3)]; %z positions of each tread
    allvx=[allvx; index(:,4)]; %vx of each tread
    allvz=[allvz; index(:,5)]; %vz of each tread
    newx=[newx (((cos(omega*t).*xs)+(sin(omega*t).*zs))+...
    V(3))]; %New X position of elbows using rotation matrix
    newz=[newz (((-sin(omega*t).*xs)+(cos(omega*t).*zs))+...
    V(4))]; %New Z position of elbows using rotation matrix
    kd=kd+1;
end

if anim==1;
    allvx=allvx*10^-(0.5); %scale vectors yourself
    allvz=allvz*10^-(0.5); %scale vectors yourself
    ForceXs=ForceXs*10^-2; %scale vectors yourself
    ForceZs=ForceZs*10^-2; %scale vectors yourself

    % Animation of Tire
    allx2=allx';
    allz2=allz';
    j=1;
    k=1;
    kk=sprintf('%.4d', k);
    while j<=(length(timetot));
```

```
    plot([newx(2,j)-0.25,newx(2,j)+0.5],[0,0],'y') %plot sand level
    axis([newx(2,j)-0.25,newx(2,j)+0.5,-0.25,0.5]) %set axis
    axis equal
    if NumTreads==4
        figure(1)
        hold on
        set(gca,'FontSize',18)
        xlabel('X')
        ylabel('Z')
        rectangle('Position',[newx(2,j)-0.75,-0.5,1.25,0.5],...
        'FaceColor',...
        'y','edgecolor','y') %sand base
        legend(num2str(timetot(j)),'FontSize',16)
        %Time in upper corner
        plot([newx(1,j),newx(3,j)],[newz(1,j),newz(3,j)],...
        [newx(2,j),...
        newx(4,j)],[newz(2,j),newz(4,j)],'k') %plot tire
        quiver(allx(1+(j-1)*NumPieces:NumPieces+(j-1)*...
        NumPieces),...
        allz(1+(j-1)*NumPieces:NumPieces+(j-1)*NumPieces),...
        allvx(1+(j-1)*NumPieces:NumPieces+(j-1)*NumPieces),...
        allvz(1+(j-1)*NumPieces:NumPieces+(j-1)*NumPieces),0,'r')
        %plot velocity vectors
        quiver(allx2(1+(j-1)*NumPieces:NumPieces+(j-1)*NumPieces),...
        allz2(1+(j-1)*NumPieces:NumPieces+(j-1)*NumPieces),...
        ForceXs(1+(j-1)*...
        NumPieces:NumPieces+(j-1)*NumPieces),ForceZs(1+(j-1)*...
        NumPieces:NumPieces+(j-1)*NumPieces),0,'b') %plot force vectors
        if savepics==1;
            hand = figure(1);
            if k < 10
                numstr = ['0000',num2str(k)];
              elseif k < 100
                  numstr = ['000',num2str(k)];
              elseif k < 1000
                  numstr = ['00',num2str(k)];
```

```
                    elseif k < 10000
                        numstr = ['0',num2str(k)];
                    else
                        numstr = num2str(k);
                end
                saveas(hand,['f_' numstr],'jpg')
                k=k+1;
                kk=sprintf('%.4d', k);
            end
            hold off
        end
        drawnow;
        j=j+1;
    end
end
```

A.2 Lug Wheel Function (FunctionVODEFourBar)

```
function [ Vdot ] = FunctionVODEFourBar( t,V,TireMass,...
TreadWidth, SegLength, omega, NumPieces, NumTreads,...
 OrderMag, scaleFactor, initialtread, g, dbforce, Fspring )
% This is the function is used by ode45 function
index = zeros(NumPieces,7);
%Local number, x,z,vx,vz,Beta,gamma
index(:,1) = [1:NumPieces];
jr=1;

for jr=1:NumPieces;
    index(jr,2)=V(3)+((cos(omega*t)*(initialtread(jr,1)))+...
    (sin(omega*t)*(initialtread(jr,2))));
    %New X position using rotation matrix
    index(jr,3)=V(4)+((-sin(omega*t)*(initialtread(jr,1)))+...
    (cos(omega*t)*(initialtread(jr,2))));
    %New Z position using rotation matrix
```

```
    index(jr,4)=V(1)+(omega*(index(jr,3)-V(4)));
    %New velocity in x-dir
    index(jr,5)=V(2)+(-omega*(index(jr,2)-V(3)));
    %New velocity in z-dir
    index(jr,6)=((initialtread(jr,3)+(omega*t))*...
    ((initialtread(jr,3)+(omega*t))<=pi/2))+...
    (((initialtread(jr,3)+(omega*t)-pi))*...
    ((initialtread(jr,3)+(omega*t))>pi/2)); %beta,
    index(jr,7)=((atan(index(jr,5)/index(jr,4))*...
    (index(jr,4)<0))+((atan(index(jr,5)/index(jr,4))+...
    pi)*(index(jr,4)>=0))); %gamma
end

A00 = 0.206*OrderMag;
A10 = 0.169*OrderMag;
B11 = 0.212*OrderMag;
B01 = 0.358*OrderMag;
Bm11 = 0.055*OrderMag;
C11 = -0.124*OrderMag;
C01 = 0.253*OrderMag;
Cm11 = 0.007*OrderMag;
D10 = 0.088*OrderMag;

jr=1;
ForceX=zeros(1,NumPieces);
ForceZ=zeros(1,NumPieces);
torque=zeros(1,NumPieces);
for jr=1:NumPieces;
    B=index(jr,6);
    G=index(jr,7);

    alphaX = scaleFactor*(Cm11*cos(-2*B+G) +...
     C01*cos(G) + C11*cos(2*B+G) +...
      D10*sin(2*B));
    alphaZ = scaleFactor*(A10*cos(2*B) + A00 +...
     Bm11*sin((-2*B)+G) + B01*sin(G) +...
```

```
      B11*sin((2*B)+G));

    if index(jr,3)<=0;
        ForceX(jr) = alphaX*SegLength*...
        TreadWidth*-index(jr,3);
        ForceZ(jr) = alphaZ*SegLength*...
        TreadWidth*-index(jr,3);
    else
        ForceX(jr) = 0;
        ForceZ(jr) = 0;
    end
    posvec=[index(jr,2)-V(3);index(jr,3)-V(4);0];
    forcevec=[ForceX(jr);ForceZ(jr);0];
    torquecross=cross(posvec, forcevec);
    torque(jr)=torquecross(3);

end

max(ForceX);
Torquetotal=sum(torque);
ForceXTot=sum(ForceX);
ForceZTot=sum(ForceZ);
Vdot = zeros(4,1);
Vdot(1) = (ForceXTot-dbforce)/TireMass;
%Accel in x-direction
Vdot(2) = (ForceZTot-(TireMass*g)+...
(Fspring*g))/TireMass;
%Accel in z-direction +(15*4.448)+(6.80389*g)
Vdot(3) = V(1); %Velocity in x-dir
Vdot(4) = V(2); %Velocity in z-dir
Vdot(5) = omega*Torquetotal;
%Derivitive of Energy dissipated

end
```

A.3 Superball Animation Code

```
%% Simulation of superball wheel
% x is defined as positive to the right
% z is defined as positive upwards

%% Clear Everything
commandwindow
clear all
close all
clf
clc
format long

%% Inputs
g = 9.81; %[m/s^2], gravitational acceleration
TireAxleInitCoord = [0,0]; %[x,z]
TreadWidth = 0.15*10^2; % [m], in plane
NumSegs = 4;
p=1; % chi wheel shape parameter
omega=30*pi/180; % [rad/s], positive in clockwise direction
init_velocity = [0,0]; %[vx, vz] %[m/s]
rad=(1)^(2*p); %radius to 2p
TireMass =1000;% [kg], mass of wheel
duration = 2; %[sec] length of simulation
dbforce=0; %Drawback force [N]
OrderMag = 10^6;
scaleFactor = 2.576;

A00 = 0.206*OrderMag;
A10 = 0.169*OrderMag;
B11 = 0.212*OrderMag;
B01 = 0.358*OrderMag;
Bm11 = 0.055*OrderMag;
```

```
C11 = -0.124*OrderMag;
C01 = 0.253*OrderMag;
Cm11 = 0.007*OrderMag;
D10 = 0.088*OrderMag;

%% Initialization
theta=linspace(0,pi,NumSegs);
theta=sort(theta);

initialtread=[];
midpts=[];
grad=[];
ms=[];
points=[];
rho=(rad./((abs(sin(theta)).^(2*p))+(abs(cos(theta))....
^(2*p)))).^(1/(2*p));
x1=rho.*cos(theta);
y1=rho.*sin(theta);
xsfix=fliplr(x1);
xsfix=xsfix(2:end);
x=[x1 xsfix];
ysfix=fliplr(y1);
ysfix=ysfix(2:end);
y=[y1 -ysfix];
points(:,1)=x;
points(:,2)=y;
initialtread=(points(1:end-1,:)+points(2:end,:))/2;
for jz=1:(length(initialtread)-1);
    midpts(jz,1)=(initialtread(jz,1)+initialtread(jz+1,1))/2;
    midpts(jz,2)=(initialtread(jz,2)+initialtread(jz+1,2))/2;
    ms(jz)=(initialtread(jz+1,2)-initialtread(jz,2))/...
     (initialtread(jz+1,1)-initialtread(jz,1));
end
midpts(end+1,1)=(initialtread(end,1)+initialtread(1,1))/2;
midpts(end,2)=(initialtread(end,2)+initialtread(1,2))/2;
ms(end+1)=(initialtread(1,2)-initialtread(end,2))/...
```

```
(initialtread(1,1)-initialtread(end,1));
for jx=1:length(midpts);
    if (ms(jx)==Inf)
        grad(jx,:)=[1,0];
    elseif (ms(jx)==-Inf)
        grad(jx,:)=[-1,0];
    else
        if midpts(jx,2)>=0
            a=[1;ms(jx)];
            aperp=[0,-1;1,0]*a;
            grad(jx,:)=[aperp(1),aperp(2)];
        else
            a=[1;ms(jx)];
            aperp=[0,1;-1,0]*a;
            grad(jx,:)=[aperp(1),aperp(2)];
        end
    end
    grad(jx,:)=[grad(jx,1)/sqrt((grad(jx,1)^2)+...
    (grad(jx,2)^2)),grad(jx,2)/sqrt((grad(jx,1)^2)...
    +(grad(jx,2)^2))];
end
SegLengths=((points(2:end,1)-points(1:end-1,1)).^2+...
(points(2:end,2)-points(1:end-1,2)).^2).^0.5;
vec2=[-1;0];
for jvv=1:1:length(midpts);
    vec1=[midpts(jvv,1)-initialtread(jvv,1);...
    midpts(jvv,2)-initialtread(jvv,2)];
    midpts(jvv,3)=((pi-acos(dot(vec1,vec2)/...
    (sqrt((vec1(1)^2)+(vec1(2)^2))*sqrt((vec2(1)^2)+...
    (vec2(2)^2)))))*(vec1(1)>=0&&vec1(2)<=0))+...
    ((acos(dot(vec1,vec2)/(sqrt((vec1(1)^2)+...
    (vec1(2)^2))*sqrt((vec2(1)^2)+(vec2(2)^2)))))*...
    (vec1(1)<0&&vec1(2)>=0))+((-acos(dot(vec1,vec2)/...
    (sqrt((vec1(1)^2)+(vec1(2)^2))*sqrt((vec2(1)^2)+...
    (vec2(2)^2)))))*(vec1(1)<0&&vec1(2)<0))+...
    (((acos(dot(vec1,vec2)/(sqrt((vec1(1)^2)+...
```

```
    (vec1(2)^2))*sqrt((vec2(1)^2)+(vec2(2)^2)))))-pi)*...
    (vec1(1)>=0&&vec1(2)>0));
end
xs=initialtread(:,1);
zs=initialtread(:,2);

%% Function
Vo = zeros(1,5);
Vo(1) = init_velocity(1); % initial velocity in x-dir
Vo(2) = init_velocity(2); % initial velocity in z-dir
Vo(3) = TireAxleInitCoord(1); % initial axle position in x
Vo(4) = TireAxleInitCoord(2); % initial axle position in z
Vo(5) = 0; % initial dissipated Power

odefix = @(t, V) FunctionVODEShapesShadow(t, V,...
 TireMass, TreadWidth, omega, OrderMag,...
  scaleFactor, midpts, g, SegLengths, grad);
[TOUT,VOUT] = ode45(odefix,[0 duration],Vo);

pdpos=find((diff(sign(VOUT(:,2))))==2);

mz1=(VOUT(pdpos(24)+1,2)-VOUT(pdpos(24),2))/...
 (TOUT(pdpos(24)+1)-TOUT(pdpos(24))); %slope of (t,z)
t01=((1/mz1)*-VOUT(pdpos(24),2))+TOUT(pdpos(24));
mv1=(VOUT(pdpos(24)+1,3)-VOUT(pdpos(24),3))/...
 (TOUT(pdpos(24)+1)-TOUT(pdpos(24))); %slope of (t,vx)
mp1=(VOUT(pdpos(24)+1,5)-VOUT(pdpos(24),5))/...
 (TOUT(pdpos(24)+1)-TOUT(pdpos(24))); %slope of (t,pow)
vxfix1=(mv1*(t01-TOUT(pdpos(24))))+VOUT(pdpos(24),3);
powfix1=(mp1*(t01-TOUT(pdpos(24))))+VOUT(pdpos(24),5);

mz2=(VOUT(pdpos(4)+1,2)-VOUT(pdpos(4),2))/...
 (TOUT(pdpos(4)+1)-TOUT(pdpos(4))); %slope of (t,z)
t02=((1/mz2)*-VOUT(pdpos(4),2))+TOUT(pdpos(4));
mv2=(VOUT(pdpos(4)+1,3)-VOUT(pdpos(4),3))/...
 (TOUT(pdpos(4)+1)-TOUT(pdpos(4))); %slope of (t,vx)
```

```
mp2=(VOUT(pdpos(4)+1,5)-VOUT(pdpos(4),5))/...
(TOUT(pdpos(4)+1)-TOUT(pdpos(4))); %slope of (t,pow)
vxfix2=(mv2*(t02-TOUT(pdpos(4))))+VOUT(pdpos(4),3);
powfix2=(mp2*(t02-TOUT(pdpos(4))))+VOUT(pdpos(4),5);

vxavg=(vxfix1-vxfix2)/(t01-t02) % [m/s]
Power=(powfix1-powfix2)/(t01-t02) % [W]

%% Animation Initialization
ForceXs=[];
ForceZs=[];
Torque=[];
allx=[];
allz=[];
allvx=[];
allvz=[];
newx=[];
newz=[];
initialtread=midpts;
for k=1:length(TOUT)
    V(1)=VOUT(k,1);
    V(2)=VOUT(k,2);
    V(3)=VOUT(k,3);
    V(4)=VOUT(k,4);
    t=TOUT(k);
    [dim dims]=size(initialtread);
    index = zeros(dim,9);
    %Local number, x,z,vx,vz,Beta,gamma
    index(:,1) = [1:dim];
    jr=1;

    for jr=1:dim;
        index(jr,2)=V(3)+((cos(omega*t)*...
        (initialtread(jr,1)))+(sin(omega*t)*...
        (initialtread(jr,2))));
        %New X position using rotation matrix
```

```
        index(jr,3)=V(4)+((-sin(omega*t)*...
        (initialtread(jr,1)))+(cos(omega*t)*...
        (initialtread(jr,2))));
        %New Z position using rotation matrix
        index(jr,4)=V(1)+(omega*(index(jr,3)-V(4)));
        %New velocity in x-dir
        index(jr,5)=V(2)+(-omega*(index(jr,2)-V(3)));
        %New velocity in z-dir
         index(jr,6)=((initialtread(jr,3)+...
         (omega*t))*((initialtread(jr,3)+...
         (omega*t))<=pi/2))+(((initialtread(jr,3)...
         +(omega*t)-pi))*((initialtread(jr,3)+...
         (omega*t))>pi/2)); %beta,
        index(jr,7)=((atan(index(jr,5)/index(jr,4))*...
        (index(jr,4)<0))+((atan(index(jr,5)/index(jr,4))+...
        pi)*(index(jr,4)>=0))); %gamma
        gradrot=[cos(omega*t), sin(omega*t);...
         -sin(omega*t) cos(omega*t)]*[grad(jr,1);grad(jr,2)];
        index(jr,8)=gradrot(1);
        index(jr,9)=gradrot(2);
    end

    A00 = 0.206*OrderMag;
    A10 = 0.169*OrderMag;
    B11 = 0.212*OrderMag;
    B01 = 0.358*OrderMag;
    Bm11 = 0.055*OrderMag;
    C11 = -0.124*OrderMag;
    C01 = 0.253*OrderMag;
    Cm11 = 0.007*OrderMag;
    D10 = 0.088*OrderMag;

    jr=1;
    ForceX=zeros(1,dim);
    ForceZ=zeros(1,dim);
    torque=zeros(1,dim);
```

```
for jr=1:dim;
    velocs=index(jr,4:5);
    B=index(jr,6);
    G=index(jr,7);
    grads=index(jr,8:9);
    coeff=1;
    alphaX = coeff*scaleFactor*(Cm11*cos(-2*B+G)...
     + C01*cos(G) + C11*cos(2*B+G) + D10*sin(2*B));
    alphaZ = coeff*scaleFactor*(A10*cos(2*B)...
     + A00 + Bm11*sin((-2*B)+G) + B01*sin(G)...
      + B11*sin((2*B)+G));
    if index(jr,3)<=0;
        ForceX(jr) = (dot([grads(1);grads(2)],...
        [velocs(1);velocs(2)])>0)*alphaX*SegLengths(jr)*...
        TreadWidth*-index(jr,3);
        ForceZ(jr) = (dot([grads(1);grads(2)],...
        [velocs(1);velocs(2)])>0)*alphaZ*...
        SegLengths(jr)*TreadWidth*-index(jr,3);
    else
        ForceX(jr) = 0;
        ForceZ(jr) = 0;
    end
    posvec=[index(jr,2)-V(3);index(jr,3)-V(4);0];
    forcevec=[ForceX(jr);ForceZ(jr);0];
    torquecross=cross(posvec, forcevec);
    torque(jr)=torquecross(3);

end

Torquetotal=sum(torque);
Torque=[Torque Torquetotal];
ForceXs=[ForceXs ForceX];
ForceZs=[ForceZs ForceZ];

ForceXTot=sum(ForceX);
ForceZTot=sum(ForceZ);
```

```
    allx=[allx; index(:,2)];
    allz=[allz; index(:,3)];
    allvx=[allvx; index(:,4)];
    allvz=[allvz; index(:,5)];
    newxs=(((cos(omega*t).*xs)...
    +(sin(omega*t).*zs))+V(3));
    newxs(end+1)=(((cos(omega*t)*...
    xs(1))+(sin(omega*t)*zs(1)))+V(3));
    newx=[newx newxs];

    newzs=(((-sin(omega*t).*xs)+...
    (cos(omega*t).*zs))+V(4));
    newzs(end+1)=(((-sin(omega*t)*xs(1))+...
    (cos(omega*t)*zs(1)))+V(4));
    newz=[newz newzs];
end

%% Animation

allvx=allvx*10^-1;
allvz=allvz*10^-1;
ForceXs=ForceXs*10^-4;
ForceZs=ForceZs*10^-4;

allx2=allx';
allz2=allz';
j=1;
k=1;
kk=sprintf('%.4d', k);
NumPieces=(NumSegs*2)-2;
while j<=(length(TOUT))
    axis([newx(2,j)-0.5,newx(2,j)+0.5,-0.5,1])
    axis equal

    figure(1)
```

```
hold on
rectangle('Position',[newx(2,j)-0.5,-0.5,6,0.5],...
'FaceColor','y','edgecolor','y')
legend(num2str(TOUT(j)))
plot(newx((((j-1)*(NumPieces+1))+1):...
(j*(NumPieces+1))),newz((((j-1)*...
(NumPieces+1))+1):(j*(NumPieces+1))),'k-')
quiver(allx(1+(j-1)*NumPieces:NumPieces+...
(j-1)*NumPieces),allz(1+(j-1)*NumPieces:NumPieces+...
(j-1)*NumPieces),allvx(1+(j-1)*NumPieces:NumPieces+...
(j-1)*NumPieces),allvz(1+(j-1)*NumPieces:NumPieces+...
(j-1)*NumPieces),0,'r')
quiver(allx2(1+(j-1)*NumPieces:NumPieces+(j-1)*...
NumPieces),allz2(1+(j-1)*NumPieces:NumPieces+...
(j-1)*NumPieces),ForceXs(1+(j-1)*NumPieces:...
NumPieces+(j-1)*NumPieces),ForceZs(1+(j-1)*...
NumPieces:NumPieces+(j-1)*NumPieces),0,'b')

hand = figure(1);
if k < 10
   numstr = ['0000',num2str(k)];
    elseif k < 100
       numstr = ['000',num2str(k)];
    elseif k < 1000
       numstr = ['00',num2str(k)];
    elseif k < 10000
       numstr = ['0',num2str(k)];
    else
       numstr = num2str(k);
end
saveas(hand,['f_' numstr],'jpg')
k=k+1;
kk=sprintf('%.4d', k);
hold off

drawnow;
```

```
    j=j+4;

end
```

A.4 Superball Function (FunctionVODEShapesShadow)

```
function [ Vdot ] = FunctionVODEShapesShadow( t,...
V,TireMass, TreadWidth, omega, OrderMag,...
 scaleFactor, initialtread, g, SegLengths, grad, dbforce )
% This is the function is used by ode45 function
[dim dims]=size(initialtread);
index = zeros(dim,9);
%Local number, x,z,vx,vz,Beta,gamma
index(:,1) = [1:dim];
jr=1;

for jr=1:dim;
    index(jr,2)=V(3)+((cos(omega*t)*...
    (initialtread(jr,1)))+(sin(omega*t)*...
    (initialtread(jr,2))));
    %New X position using rotation matrix
    index(jr,3)=V(4)+((-sin(omega*t)*...
    (initialtread(jr,1)))+(cos(omega*t)*...
    (initialtread(jr,2))));
    %New Z position using rotation matrix
    index(jr,4)=V(1)+(omega*...
    (index(jr,3)-V(4)));
    %New velocity in x-dir
    index(jr,5)=V(2)+(-omega*...
    (index(jr,2)-V(3)));
    %New velocity in z-dir
    index(jr,6)=((initialtread(jr,3)+...
    (omega*t))*((initialtread(jr,3)+(omega*t))<=pi/2))...
    +(((initialtread(jr,3)+(omega*t)-pi))*...
```

```
    ((initialtread(jr,3)+(omega*t))>pi/2)); %beta,
    index(jr,7)=((atan(index(jr,5)/index(jr,4))*...
    (index(jr,4)<0))+((atan(index(jr,5)/index(jr,4))...
    +pi)*(index(jr,4)>=0))); %gamma
    gradrot=[cos(omega*t), sin(omega*t);...
     -sin(omega*t) cos(omega*t)]*[grad(jr,1);grad(jr,2)];
    index(jr,8)=gradrot(1);
    index(jr,9)=gradrot(2);
end
A00 = 0.206*OrderMag;
A10 = 0.169*OrderMag;
B11 = 0.212*OrderMag;
B01 = 0.358*OrderMag;
Bm11 = 0.055*OrderMag;
C11 = -0.124*OrderMag;
C01 = 0.253*OrderMag;
Cm11 = 0.007*OrderMag;
D10 = 0.088*OrderMag;

jr=1;
ForceX=zeros(1,dim);
ForceZ=zeros(1,dim);
torque=zeros(1,dim);
for jr=1:dim;
    velocs=index(jr,4:5);
    B=index(jr,6);
    G=index(jr,7);
    grads=index(jr,8:9);
    coeff=1;
    alphaX = coeff*scaleFactor*(Cm11*...
    cos(-2*B+G) + C01*cos(G) + C11*...
    cos(2*B+G) + D10*sin(2*B));
    alphaZ = coeff*scaleFactor*(A10*...
    cos(2*B) + A00 + Bm11*sin((-2*B)+G)...
     + B01*sin(G) + B11*sin((2*B)+G));
    if index(jr,3)<=0;
```

```
        ForceX(jr) = (dot([grads(1);grads(2)],...
        [velocs(1);velocs(2)])>0)*alphaX*...
        SegLengths(jr)*TreadWidth*-index(jr,3);
        ForceZ(jr) = (dot([grads(1);grads(2)],...
        [velocs(1);velocs(2)])>0)*alphaZ*...
        SegLengths(jr)*TreadWidth*-index(jr,3);
    else
        ForceX(jr) = 0;
        ForceZ(jr) = 0;
    end
    posvec=[index(jr,2)-V(3);index(jr,3)-V(4);0];
    forcevec=[ForceX(jr);ForceZ(jr);0];
    torquecross=cross(posvec, forcevec);
    torque(jr)=torquecross(3);
end

Torquetotal=sum(torque);
ForceXTot=sum(ForceX);
ForceZTot=sum(ForceZ);
Vdot = zeros(4,1);
Vdot(1) = (ForceXTot-dbforce)/TireMass;
%Accel in x-direction
Vdot(2) = (ForceZTot-(TireMass*g))/TireMass;
%Accel in z-direction
Vdot(3) = V(1); %Velocity in x-dir
Vdot(4) = V(2); %Velocity in z-dir
Vdot(5) = omega*Torquetotal;
% Derivitive of Energy dissipated

end
```

A.5 Rotating Flap Wheel Animation Code

```
%% Simulation of rotating-flap wheel
```

```
% x is defined as positive to the right
% z is defined as positive upwards

%% Clear Everything
commandwindow
clear all
close all
clf
clc
format long

%% Inputs
savepics=1;
g = 9.81; % [m/s^2], gravitational acceleration
TreadWidth = 0.1; % [m], in plane.
NumSegs = 300; % Number of discrete linear segments:
% must be even and divisible by 5
p=1; % chi wheel shape parameter
omega=20*pi/180; % [rad/s], positive in clockwise direction
effrad=0.08; % effective tire radius [m]
TireAxleInitCoord = [0,effrad]; % [x,z] [m]
rad=(effrad)^(2*p); % radius to 2p
init_velocity = [0,0]; % [vx, vz], [m/s]
TireMass=14.88;% [kg], mass of wheel
duration=(0.4*pi/omega)*50; %[sec] length of simulation
dbmass=14.3/9.81;% [kg] drawback mass used as force
Fspring=66.72; % [N] force of spring
Mpulley=1.134+0.3; %[kg]
OrderMag=10^6; % units for RFT coeffs
scaleFactor = 2.06;  % sand material specific scaling factor
deltat=(0.4*pi/omega)/5000; % timestep [sec]
mplatform=9; % [kg], mass of horizontally sliding platform

%Flap Specific
numflaps=5; % number of flaps
rotang=70*pi/180; % flap angle [rad]
```

```
% RFT coefficients
A00 = 0.206*OrderMag;
A10 = 0.169*OrderMag;
B11 = 0.212*OrderMag;
B01 = 0.358*OrderMag;
Bm11 = 0.055*OrderMag;
C11 = -0.124*OrderMag;
C01 = 0.253*OrderMag;
Cm11 = 0.007*OrderMag;
D10 = 0.088*OrderMag;

%% Initialization of Flap
theta=sort(linspace(0,pi,(NumSegs+2)/2));
initialtread=[]; %midpts of segments formed by points
grad=[]; % outward unit vectors from points outlines
ms=[]; %slopes of points
points=[]; % coordinates of segment points

% Create Circle
rho=(rad./((abs(sin(theta)).^(2*p))+(abs(cos(theta)).^...
(2*p)))).^(1/(2*p)); %radian in polar coordinate
x1=rho.*cos(theta); %polar to cartesian x
y1=rho.*sin(theta); %polar to cartesian y
xsfix=fliplr(x1);
xsfix=xsfix(2:end);
x=[x1 xsfix];
ysfix=fliplr(y1);
ysfix=ysfix(2:end);
y=[y1 -ysfix];
points(:,1)=x;
points(:,2)=y;
xs=points(1:end-1,1);
zs=points(1:end-1,2);
points2=points(1:end-1,:);
pointsnew=[];
```

```
% Rotate Segments
fsegs=length(points2)/numflaps;
rotmat=[cos(rotang) sin(rotang); -sin(rotang) cos(rotang)];
for j=1:1:numflaps;
    coord=points2((fsegs*(j-1))+1:fsegs*j,:)';
    if j==numflaps
        coord(:,end+1)=points2(1,:)';
    else
        coord(:,end+1)=points2((fsegs*j)+1,:)';
    end
    rotcoord=rotmat*[coord(1,:)-coord(1,1);coord(2,:)-coord(2,1)];
    newcoord=[rotcoord(1,:)+coord(1,1);rotcoord(2,:)+coord(2,1)];
    pointsnew((fsegs*(j-1))+j:(fsegs*j)+j,1:2)=newcoord';
end

points2=[];
points2=pointsnew;

% Get midpts
initialtread=(points2(1:end-1,:)+points2(2:end,:))...
/2; %midpts of segments formed by points
ms=(points2(2:end,2)-points2(1:end-1,2))./...
(points2(2:end,1)-points2(1:end-1,1));
ydiff=(points2(2:end,2)-points2(1:end-1,2));
xdiff=(points2(2:end,1)-points2(1:end-1,1));
SegLengths=((points2(2:end,1)-points2(1:end-1,1)).^2...
+(points2(2:end,2)-points2(1:end-1,2)).^2).^0.5; % of points lines

k=(1:1:numflaps-1)*(fsegs+1);
initialtread(k,:)=[];
ms(k,:)=[];
xdiff(k,:)=[];
ydiff(k,:)=[];
SegLengths(k,:)=[];

```

```
% Get outward normal vectors
for jx=1:length(initialtread);
    if (ms(jx)==Inf)
        grad(jx,:)=[1,0];
    elseif (ms(jx)==-Inf)
        grad(jx,:)=[-1,0];
    else
        a=[xdiff(jx);ydiff(jx)];
        aperp=[0,1;-1,0]*a;
        grad(jx,:)=[aperp(1),aperp(2)];
    end
    grad(jx,:)=[grad(jx,1)/sqrt((grad(jx,1)^2)+(grad(jx,2)^2)),...
    grad(jx,2)/sqrt((grad(jx,1)^2)+(grad(jx,2)...
    ^2))]; % outward facing vectors from shape outlines
end

vec2=[-1;0];
for jvv=1:1:length(initialtread);
    vec1=[initialtread(jvv,1)-points(jvv,1);initialtread(jvv,2)-...
    points(jvv,2)];
    initialtread(jvv,3)=((pi-acos(dot(vec1,vec2)/(sqrt((vec1(1)^2)+...
    (vec1(2)^2))*sqrt((vec2(1)^2)+(vec2(2)^2))))))*...
    (vec1(1)>=0&&vec1(2)<=0))...
        +((acos(dot(vec1,vec2)/(sqrt((vec1(1)^2)+...
        (vec1(2)^2))*sqrt((vec2(1)^2)+(vec2(2)^2)))))*...
        (vec1(1)<0&&vec1(2)>=0))+((-acos(dot(vec1,vec2)...
        /(sqrt((vec1(1)^2)+(vec1(2)^2))*...
        sqrt((vec2(1)^2)+(vec2(2)^2)))))...
        *(vec1(1)<0&&vec1(2)<0))+...
        (((acos(dot(vec1,vec2)/(sqrt((vec1(1)^2)+(vec1(2)^2))...
        *sqrt((vec2(1)^2)+(vec2(2)^2)))))-pi)*...
        (vec1(1)>=0&&vec1(2)>0)); %betas of intitial tread
end

%%
% Plot Flap Wheel
```

```
figure(2)
hold on
plot(xs,zs,'k-',points2(:,1),points2(:,2),'ro',...
initialtread(:,1),initialtread(:,2),'ro','LineWidth',6)
axis([-0.2, 0.2, -0.2, 0.2]);%equal
quiver(initialtread(:,1),initialtread(:,2),grad(:,1),grad(:,2))
quiver(initialtread(:,1),initialtread(:,2),ones(length(ms),1),ms)

% figure(1)
% hold on
% axis equal
% plot(initialtread2(:,1),initialtread2(:,2),'ro',xs,zs,'bo');
% size(initialtread)

%% Get Initial Index of Axle
Vo = zeros(1,5);
Vo(1) = init_velocity(1); % initial velocity in x-dir
Vo(2) = init_velocity(2); % initial velocity in z-dir
Vo(3) = TireAxleInitCoord(1); % initial axle position in x
Vo(4) = TireAxleInitCoord(2); % initial axle position in z
Vo(5) = 0; % initial dissipated Power


%options = odeset('RelTol',1e-3,'AbsTol',1e-6);
% Default 1e-3 and 1e-6
odefix = @(t, V) DBFlapFunctionVODE(t, V, TireMass,...
 TreadWidth, omega, OrderMag, scaleFactor, initialtread,...
  g, SegLengths, grad, dbmass, Fspring,...
   Mpulley,effrad,mplatform);
%[TOUT,VOUT] = ode23(odefix,[0 duration],Vo,options );

TOUT=0:deltat:duration;
VOUT=zeros(length(TOUT),5);
VOUT(1,:)=Vo;
for tstp=2:1:length(TOUT);
    VOUT(tstp,:)=VOUT(tstp-1,:)+deltat*...
```

```
    odefix(TOUT(tstp-1),VOUT(tstp-1,:))';
end
%%
figure(3)
plot(VOUT(:,3),VOUT(:,4),'.-')

figure(4)
plot(TOUT,VOUT(:,3),'.-')

figure(5)
plot(TOUT,VOUT(:,4),'.-')
%%

pdpos=find((diff(sign(VOUT(:,2))))==2);
PDPosNum=length(pdpos)

cyctime=((2*pi)/omega)/5;
endavg=PDPosNum;
cyctimebeg=TOUT(pdpos(endavg))-cyctime;
startavg=find(abs(TOUT(pdpos)-cyctimebeg)<=0.1);
if length(startavg)>1
    startavg=startavg(ceil((startavg(end)-startavg(1))/2));
end

mz1=(VOUT(pdpos(endavg)+1,2)-VOUT(pdpos(endavg),2))...
/(TOUT(pdpos(endavg)+1)-TOUT(pdpos(endavg))); %slope of (t,z)
t01=((1/mz1)*-VOUT(pdpos(endavg),2))+TOUT(pdpos(endavg));
mv1=(VOUT(pdpos(endavg)+1,3)-VOUT(pdpos(endavg),3))...
/(TOUT(pdpos(endavg)+1)-TOUT(pdpos(endavg))); %slope of (t,vx)
mp1=(VOUT(pdpos(endavg)+1,5)-VOUT(pdpos(endavg),5))...
/(TOUT(pdpos(endavg)+1)-TOUT(pdpos(endavg))); %slope of (t,pow)
vxfix1=(mv1*(t01-TOUT(pdpos(endavg))))+VOUT(pdpos(endavg),3);
powfix1=(mp1*(t01-TOUT(pdpos(endavg))))+VOUT(pdpos(endavg),5);

mz2=(VOUT(pdpos(startavg)+1,2)-VOUT(pdpos(startavg),2))...
/(TOUT(pdpos(startavg)+1)-TOUT(pdpos(startavg))); %slope of (t,z)
```

```
t02=((1/mz2)*-VOUT(pdpos(startavg),2))+TOUT(pdpos(startavg));
mv2=(VOUT(pdpos(startavg)+1,3)-VOUT(pdpos(startavg),3))...
/(TOUT(pdpos(startavg)+1)-TOUT(pdpos(startavg))); %slope of (t,vx)
mp2=(VOUT(pdpos(startavg)+1,5)-VOUT(pdpos(startavg),5))...
/(TOUT(pdpos(startavg)+1)-TOUT(pdpos(startavg))); %slope of (t,pow)
vxfix2=(mv2*(t02-TOUT(pdpos(startavg))))+VOUT(pdpos(startavg),3);
powfix2=(mp2*(t02-TOUT(pdpos(startavg))))+VOUT(pdpos(startavg),5);

vxavg=(vxfix1-vxfix2)/(t01-t02) %m/s
Power=(powfix1-powfix2)/(t01-t02) %Newton
Endpos=VOUT(end,3) %Ending Position

%% Animation Initialization

ForceXs=[];
ForceZs=[];
Torque=[];
allx=[];
allz=[];
allvx=[];
allvz=[];
newx=[];
newz=[];

fps=5;
u=find(diff(sign(diff(mod((TOUT),1/fps))))==2);
TOUTfps=TOUT(u);
VOUTfps=VOUT(u,:);
kd=1;

for k=1:1:length(TOUTfps)
    V(1)=VOUTfps(k,1);
    V(2)=VOUTfps(k,2);
    V(3)=VOUTfps(k,3);
    V(4)=VOUTfps(k,4);
    t=TOUTfps(k);
```

```
timetot(kd)=t;

[dim dims]=size(initialtread);
index = zeros(dim,9); %Local number, x,z,vx,vz,Beta,gamma
index(:,1) = [1:dim];
jr=1;

for jr=1:dim;
    index(jr,2)=V(3)+((cos(omega*t)*(initialtread(jr,1)))...
    +(sin(omega*t)*...
    (initialtread(jr,2)))); %New X position using rotation matrix
    index(jr,3)=V(4)+((-sin(omega*t)*(initialtread(jr,1)))...
    +(cos(omega*t)*...
    (initialtread(jr,2)))); %New Z position using rotation matrix
    index(jr,4)=V(1)+(omega*...
    (index(jr,3)-V(4))); %New velocity in x-dir
    index(jr,5)=V(2)+(-omega*...
    (index(jr,2)-V(3))); %New velocity in z-dir
    index(jr,6)=((initialtread(jr,3)+(omega*t))...
    *((initialtread(jr,3)...
    +(omega*t))<=pi/2))+(((initialtread(jr,3)+(omega*t)-pi))*...
    ((initialtread(jr,3)+(omega*t))>pi/2)); %beta,
    index(jr,7)=((atan(index(jr,5)/index(jr,4))...
    *(index(jr,4)<0))+...
    ((atan(index(jr,5)/index(jr,4))+pi)*...
    (index(jr,4)>=0))); %gamma
    gradrot=[cos(omega*t), sin(omega*t); -sin(omega*t) ...
    cos(omega*t)]*[grad(jr,1);grad(jr,2)];
    index(jr,8)=gradrot(1);
    index(jr,9)=gradrot(2);
end

A00 = 0.206*OrderMag;
A10 = 0.169*OrderMag;
B11 = 0.212*OrderMag;
B01 = 0.358*OrderMag;
```

```
Bm11 = 0.055*OrderMag;
C11 = -0.124*OrderMag;
C01 = 0.253*OrderMag;
Cm11 = 0.007*OrderMag;
D10 = 0.088*OrderMag;

jr=1;
ForceX=zeros(1,dim);
ForceZ=zeros(1,dim);
torque=zeros(1,dim);
for jr=1:dim;
    velocs=index(jr,4:5);
    B=index(jr,6);
    G=index(jr,7);
    grads=index(jr,8:9);
    alphaX = scaleFactor*(Cm11*cos(-2*B+G) + C01*cos(G) +...
     C11*cos(2*B+G) + D10*sin(2*B));
    alphaZ = scaleFactor*(A10*cos(2*B) + A00 + Bm11*sin((-2*B)+G)...
     + B01*sin(G) + B11*sin((2*B)+G));
    if index(jr,3)<=0;
        ForceX(jr) = (dot([grads(1);grads(2)],...
        [velocs(1);velocs(2)])>0)*...
        alphaX*SegLengths(jr)*TreadWidth*-index(jr,3);
        ForceZ(jr) = (dot([grads(1);grads(2)],...
        [velocs(1);velocs(2)])>0)*...
        alphaZ*SegLengths(jr)*TreadWidth*-index(jr,3);
    else
        ForceX(jr) = 0;
        ForceZ(jr) = 0;
    end
    posvec=[index(jr,2)-V(3);index(jr,3)-V(4);0];
    forcevec=[ForceX(jr);ForceZ(jr);0];
    torquecross=cross(posvec, forcevec);
    torque(jr)=torquecross(3);
end

```

```
    Torquetotal=sum(torque);
    Torque=[Torque Torquetotal];
    ForceXs=[ForceXs ForceX];
    ForceZs=[ForceZs ForceZ];
    ForceXTot=sum(ForceX);
    ForceZTot=sum(ForceZ);

    allx=[allx; index(:,2)];
    allz=[allz; index(:,3)];
    allvx=[allvx; index(:,4)];
    allvz=[allvz; index(:,5)];
    newx=[newx (((cos(omega*t).*xs)+(sin(omega*t).*zs))+...
    V(3))]; %New X position of elbows using rotation matrix
    newz=[newz (((-sin(omega*t).*xs)+(cos(omega*t).*zs))+...
    V(4))]; %New Z position of elbows using rotatoin matrix
    kd=kd+1;
end

% Animation of Wheel
anim=1;

if anim==1;
    allvx=allvx*10^-0.3;
    allvz=allvz*10^-0.3;
    ForceXs=ForceXs*10^-2;
    ForceZs=ForceZs*10^-2;

    allx2=allx';
    allz2=allz';
    j=1;
    k=1;
    kk=sprintf('%.4d', k);
    NumPieces=NumSegs;
    while j<=(length(timetot))
        plot([newx(2,j)-0.2,newx(2,j)+0.2],[0,0],...
        'y') %key to making plot reset each time
```

```
        axis([newx(2,j)-0.2,newx(2,j)+0.2,-0.2,0.4])
        axis equal

        figure(1)
        hold on
        set(gca,'FontSize',18)
        xlabel('X')
        ylabel('Z')
        rectangle('Position',[newx(2,j)-0.4,-1,0.7,1]...
        ,'FaceColor','y','edgecolor','y')
        legend(num2str(timetot(j)))


        plot(newx((((j-1)*(NumPieces))+1):(j*(NumPieces))),...
        newz((((j-1)*(NumPieces))+1):(j*(NumPieces))),'k-')
        quiver(allx(1+(j-1)*NumPieces:NumPieces+(j-1)*NumPieces)...
        ,allz(1+(j-1)*NumPieces:NumPieces+(j-1)*NumPieces),...
        allvx(1+(j-1)*NumPieces:NumPieces+(j-1)*NumPieces),...
        allvz(1+(j-1)*NumPieces:NumPieces+(j-1)*NumPieces),0,'r')
        quiver(allx2(1+(j-1)*NumPieces:NumPieces+(j-1)*NumPieces)...
        ,allz2(1+(j-1)*NumPieces:NumPieces+(j-1)*NumPieces),...
        ForceXs(1+(j-1)*NumPieces:NumPieces+(j-1)*NumPieces),...
        ForceZs(1+(j-1)*NumPieces:NumPieces+(j-1)*NumPieces),0,'b')

        if savepics==0;
            hand = figure(1);
            % Attach the right suffix to the image
            if k < 10
               numstr = ['0000',num2str(k)];
                elseif k < 100
                   numstr = ['000',num2str(k)];
                elseif k < 1000
                   numstr = ['00',num2str(k)];
                elseif k < 10000
                   numstr = ['0',num2str(k)];
                else
```

```
                numstr = num2str(k);
            end
            saveas(hand,['f_' numstr],'jpg')
            k=k+1;
            kk=sprintf('%.4d', k);
        end

        drawnow;
        j=j+1;
        hold off

    end
end

 %% Plot torque vs time, and position and velocity vs time
 % figure(1)
 % plot(TOUT,Torque,'x-')
 %
 % figure(2)
 % plot(TOUT,VOUT(:,1),'r',TOUT,VOUT(:,2),'b',TOUT,VOUT(:,3),...
 %'g',TOUT,VOUT(:,4),'y','LineWidth',2)
 % legend('vx','vz','x','z')
 % xlabel('Time')
```

A.6 Rotating Flap Wheel Function (DBFlapFunctionVODE)

```
function [ Vdot ] = FlapFunctionVODE( t,V,TireMass,...
 TreadWidth, omega, OrderMag, scaleFactor, initialtread,...
  g, SegLengths, grad, dbmass, Fspring, Mpulley,effrad, mplatform )
% This is the function is used by ode45 function
[dim dims]=size(initialtread);
index = zeros(dim,9); %Local number, x,z,vx,vz,Beta,gamma
index(:,1) = [1:dim];
```

```
jr=1;

for jr=1:dim;
    index(jr,2)=V(3)+((cos(omega*t)*(initialtread(jr,1)))...
    +(sin(omega*t)*(initialtread(jr,2))));
    %New X position using rotation matrix
    index(jr,3)=V(4)+((-sin(omega*t)*...
    (initialtread(jr,1)))+(cos(omega*t)*...
    (initialtread(jr,2)))); %New Z position using rotation matrix
    index(jr,4)=V(1)+(omega*(index(jr,3)-V(4)));
    %New velocity in x-dir
    index(jr,5)=V(2)+(-omega*(index(jr,2)-V(3)));
    %New velocity in z-dir
    index(jr,6)=((initialtread(jr,3)+(omega*t))*((initialtread(jr,3)+...
    (omega*t))<=pi/2))+(((initialtread(jr,3)+(omega*t)-pi))*...
    ((initialtread(jr,3)+(omega*t))>pi/2)); %beta,
    index(jr,7)=((atan(index(jr,5)/index(jr,4))*(index(jr,4)<0))...
    +((atan(index(jr,5)/index(jr,4))+pi)*(index(jr,4)>=0))); %gamma
    gradrot=[cos(omega*t), sin(omega*t); -sin(omega*t)...
     cos(omega*t)]*[grad(jr,1);grad(jr,2)];
    index(jr,8)=gradrot(1);
    index(jr,9)=gradrot(2);
end

A00 = 0.206*OrderMag;
A10 = 0.169*OrderMag;
B11 = 0.212*OrderMag;
B01 = 0.358*OrderMag;
Bm11 = 0.055*OrderMag;
C11 = -0.124*OrderMag;
C01 = 0.253*OrderMag;
Cm11 = 0.007*OrderMag;
D10 = 0.088*OrderMag;

jr=1;
ForceX=zeros(1,dim);
```

```
ForceZ=zeros(1,dim);
torque=zeros(1,dim);
for jr=1:dim;
    velocs=index(jr,4:5);
    B=index(jr,6);
    G=index(jr,7);
    grads=index(jr,8:9);
    alphaX = scaleFactor*(Cm11*cos(-2*B+G) +...
     C01*cos(G) + C11*cos(2*B+G) + D10*sin(2*B));
    alphaZ = scaleFactor*(A10*cos(2*B) + A00 +...
     Bm11*sin((-2*B)+G) + B01*sin(G) + B11*sin((2*B)+G));
    if index(jr,3)<=0;
        ForceX(jr) = (dot([grads(1);grads(2)],...
        [velocs(1);velocs(2)])>0)*...
        alphaX*SegLengths(jr)*TreadWidth*-index(jr,3);
        ForceZ(jr) = (dot([grads(1);grads(2)],...
        [velocs(1);velocs(2)])>0)*...
        alphaZ*SegLengths(jr)*TreadWidth*-index(jr,3);
    else
        ForceX(jr) = 0;
        ForceZ(jr) = 0;
    end
    posvec=[index(jr,2)-V(3);index(jr,3)-V(4);0];
    forcevec=[ForceX(jr);ForceZ(jr);0];
    torquecross=cross(posvec, forcevec);
    torque(jr)=torquecross(3);
end

Torquetotal=sum(torque);
ForceXTot=sum(ForceX);
ForceZTot=sum(ForceZ);
Vdot = zeros(5,1);

Vdot(1) = (ForceXTot-(dbmass*g))/(TireMass+dbmass+...
mplatform); %Accel in x-direction
Vdot(2) = (ForceZTot-(TireMass*g)+Fspring+(Mpulley*g))...
```

```
/(TireMass+Mpulley); %Accel in z-direction
Vdot(3) = V(1); %Velocity in x-dir
Vdot(4) = V(2); %Velocity in z-dir
Vdot(5) = omega*Torquetotal; % Derivitive of Energy dissipated

% if sqrt((Vdot(1)^2)+(Vdot(2)^2))>(g);
%      scalfix=(17)/sqrt((Vdot(1)^2)+(Vdot(2)^2));
%      Vdot(1)=Vdot(1)*scalfix;
%      Vdot(2)=Vdot(2)*scalfix;
% end
end
```

Bibliography

[1] Slonaker, J. (2015). Wheel design optimization for locomotion in granular beds using resistive force theory (Dissertation, Massachusetts Institute of Technology).

[2] Slonaker, J., Motley, D. C., Senatore, C., Iagnemma, K., & Kamrin, K. (2016). Geometrically general scaling relations for locomotion on granular beds. arXiv preprint arXiv:1604.02490.

[3] Goddard, J. D. (2014). Continuum modeling of granular media. Applied Mechanics Reviews, 66(5), 050801.

[4] Li, C., Zhang, T., & Goldman, D. I. (2013). A terradynamics of legged locomotion on granular media. Science, 339(6126), 1408-1412.

[5] Lighthill, J. (1975). Mathematical biofluiddynamics (Vol. 17). Siam.

[6] Goldman, D. I. (2014). Colloquium: Biophysical principles of undulatory self-propulsion in granular media. Reviews of Modern Physics, 86(3), 943.

[7] Ding, Y., Gravish, N., & Goldman, D. I. (2011). Drag induced lift in granular media. Physical Review Letters, 106(2), 028001.

[8] Askari, H., & Kamrin, K. (2015). Intrusion in heterogeneous materials: Simple global rules from complex micro-mechanics. arXiv preprint arXiv:1510.02966.

[9] Wong, J. Y., & Reece, A. R. (1967). Prediction of rigid wheel performance based on the analysis of soil-wheel stresses part I. Performance of driven rigid wheels. Journal of Terramechanics, 4(1), 81-98.

[10] R. B. Ahlvin and P. W. Haley, Nato Reference Mobility Model: Edition II. NRMM User's Guide (US Army Engineer Waterways Experiment Station, 1992).

[11] Bekker, M. G. (1956). Theory of land locomotion.

[12] Goldsmith, J., Guo, H., Hunt, S. N., Tao, M., & Koehler, S. (2013). Drag on intruders in granular beds: A boundary layer approach. Physical Review E, 88(3), 030201.

[13] Senatore, C., & Iagnemma, K. (2014). Analysis of stress distributions under lightweight wheeled vehicles. Journal of Terramechanics, 51, 1-17.

[14] Nedderman, R. M. (1992). Statics and Kinematics of Granular Materials Cambridge Univ. Press, Cambridge.

[15] Kobayashi, T., Fujiwara, Y., Yamakawa, J., Yasufuku, N., & Omine, K. (2010). Mobility performance of a rigid wheel in low gravity environments. Journal of Terramechanics, 47(4), 261-274.

[16] da Cruz, F., Emam, S., Prochnow, M., Roux, J. N., & Chevoir, F. (2005). Rheophysics of dense granular materials: Discrete simulation of plane shear flows. Physical Review E, 72(2), 021309.

[17] Jop, P., Forterre, Y., & Pouliquen, O. (2006). A constitutive law for dense granular flows. Nature, 441(7094), 727-730.

[18] Lagrée, P. Y., Staron, L., & Popinet, S. (2011). The granular column collapse as a continuum: validity of a two-dimensional Navier-Stokes model with a $\mu(i)$-rheology. Journal of Fluid Mechanics, 686, 378-408.

[19] Kamrin, K., & Koval, G. (2012). Nonlocal constitutive relation for steady granular flow. Physical Review Letters, 108(17), 178301.

[20] Kamrin, K., & Henann, D. L. (2015). Nonlocal modeling of granular flows down inclines. Soft matter, 11(1), 179-185.

[21] Motley, D.C. (2016). Physical Experimentation and Actuated Wheel Design for Granular Locomotion Using Resistive Force Theory (Dissertation, Massachusetts Institute of Technology).

[22] Okafor, C. A. (2011). Integration of a testbed for examining the interaction of Mars rover wheels with a Mars soil simulant (Dissertation, Massachusetts Institute of Technology).