© Copyright 2017

Jacob Baldassini

An Examination of the Effects of Deformable Foam Contact Surfaces on Robotic Locomotion

Jacob Baldassini

A thesis

submitted in partial fulfillment of the

requirements for the degree of

Master of Science in Electrical Engineering

University of Washington

2017

Reading Committee:

Samuel A. Burden, Chair

Blake Hannaford

Program Authorized to Offer Degree:

Department of Electrical Engineering

University of Washington

Abstract

An Examination of the Effects of Deformable Foam Contact Surfaces on Robotic Locomotion

Jacob Baldassini

Chair of the Supervisory Committee:
Professor Samuel A. Burden
Electrical Engineering

The ability of foam to reduce the impact of terrain irregularities on robotic locomotion was studied. A test bed was developed to conduct static experiments to attempt to measure and quantify how statically positioned foam feet reacted to changing terrain profiles. The average height of the terrain profile was varied linearly by moving a ledge under the foot, which, according to our hypothesis, should have resulted in a linear variation of the height of the foot. These experiments were repeated for two different ledge heights (3 mm and 12.8 mmm), two different foot sizes (2 cm and 3 cm cubes), and five different foot stiffnesses. Foot behavior approached full linearity as stiffness increased for both foot sizes on the 3 mm ledge, but did not approach full linearity for either foot size on the 12.8 mm ledge. Although foam feet have desirable properties, their behavior was not consistent enough to create a predictive model for their behavior for a given stiffness, size, and ledge height. Some insight into a manner for obtaining design curves was obtained, however, and other avenues of research appear promising.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

This work was done under the mentorship of Prof. Samuel Burden. I also want to note the helpful discussions and support that Yana Sosnovskaya of the University of Washington Department of Electrical Engineering provided.

This material is based upon work supported by the U. S. Army Research Laboratory and the U. S. Army Research Office under contract/grant number W911NF-16-1-0158.

Chapter 1. INTRODUCTION

Simple organisms can easily move themselves over uneven terrain despite operating under constraints that robots do not need to consider, such as avoiding predators, hunting prey, and maintaining the ability to heal and reproduce. Despite this, it has proved difficult to create robust control strategies for robotic locomotion. Recent research has suggested that in some organisms, the structure of the limbs provides inherent resistance to disruption; when cockroaches were perturbed while running, they began regaining their original course before they could have reacted via neural feedback [5]. These responses were also faster than pure reflex actions measured in humans [2]. This research attempts to determine whether a similar effect can be obtained for robots by using deformable feet.

Chapter 2. BACKGROUND

2.1 RELATED WORK

There is precedent for using biologically inspired foot designs in robotic locomotion research to improve performance. Many aspects of biological control strategies have been incorporated in modern robotic design, like posture, body proportioning, and the use of tails [15]. A counterintuitive result that came out from this research is that decreased rigidity can lead to increased performance, although this research was performed on a small, insect-inspired platform [14].

The performance of locomotion control strategies is also known to be highly dependent on the interaction of the limbs and terrain [9]. This is especially when considering deformable terrain [4] [10], although it was examined in the context of granular media, which does not have

the same properties as foam. This paper is, to the author's knowledge, the first to examine the use of deformable foam feet in this context. Deformable feet would fit within the broader paradigm of morphological computation [3]- here, correcting for disturbances will be partly outsourced from the control strategy to the foot itself. The smoothing effect the feet will have on disturbances can also outsource filtering, for cases where the limbs are also being used as sensors [8]. Research into decoupling the limbs has shown promise in smoothing trajectory outcomes [11], and deformable feet support the application of this result as well, as the effect of foam compressing can reduce the sharpness of transitioning between different modes in a gait.

The mathematical theory describing the deformation of elastic materials is well-known. For sufficiently small amounts of applied force, elastic body deformations are well-approximated by first order (or *linear*) deformations [12]. As foam is such a material, this means that if second-order effects and other sources of error are negligible, foam feet may be modeled as springs.

2.2 MOTIVATION

The genesis for this research was contemplating ways to improve the performance of a four-legged robot, like a Minitaur or something similar [6], by using foam coatings on its feet. We began by considering an individual leg. We chose to use a leg manufactured by Ghost Robotics, the manufacturers of the Minitaur, as it could be directly controlled via Arduino.

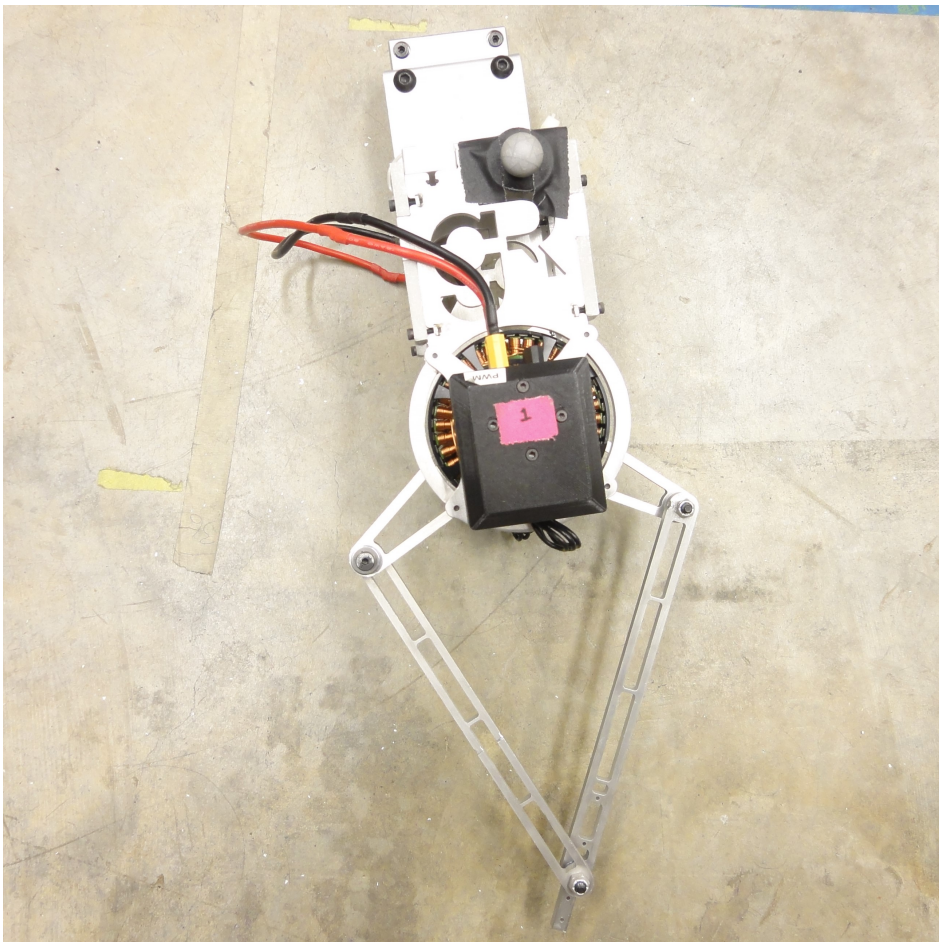

Figure 1: A Picture of the Robotic Leg

Terms used to calculate the dynamics of this robot are defined as follows: Leg 1 is the leg corresponding to motor 1, positioned on the left in figure 1. Leg 2 is the leg corresponding to motor 2, positioned on the right in figure 1. Consider the robot standing on the y-axis in the Cartesian plane. θ_1 and θ_2 are, respectively, the angles that leg 1 and leg 2 make with the hip, as measured clockwise from the negative y-axis. The position of the foot joint is defined as (x_f, y_f), as its dynamics are as follows:

$$\begin{bmatrix} \bar{\theta} \\ \Delta \end{bmatrix} = \frac{1}{2}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}\begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix}$$

$$\begin{bmatrix} x_{f_{rot}} \\ y_{f_{rot}} \end{bmatrix} = \begin{bmatrix} 0 \\ -l_1 \sin\left(\frac{\pi}{2} - \Delta\right) - \sqrt{l_2^2 - l_1^2 \sin^2(\Delta)} \end{bmatrix}$$

$$\begin{bmatrix} x_f \\ y_f \end{bmatrix} = \begin{bmatrix} \cos(\bar{\theta}) & -\sin(\bar{\theta}) \\ \sin(\bar{\theta}) & \cos(\bar{\theta}) \end{bmatrix} \begin{bmatrix} x_{f_{rot}} \\ y_{f_{rot}} \end{bmatrix}$$

We calculated the distance to the foot joint rather than the distance to the foot's contact surface, because the distance from the foot joint to the contact surface of the foot varies based on the type of foot being used.

These kinematics differ from those calculated for an identical leg in [6]. The kinematics calculated in that paper are valid in the regime where the leg's knees (the joints where the leg 1 and leg 2 segments meet) are above the hip (the joint where both leg 1 segments originate). The kinematics we calculated are valid in the regime where $\theta_1 > \theta_2,\ \theta_1 \in \left[-\frac{\pi}{2}, \pi\right],\ \theta_2 \in [-\pi, \frac{\pi}{2}]$; in other words, where the knees are below the hip. A MATLAB script to implement these dynamics is attached as Appendix A.

Chapter 3. METHODOLOGY

3.1 HYPOTHESIS

Our hypothesis, based on our reading of [12], is that the use of foam can smooth the effects of interacting with discontinuous terrain. Consider dividing the foam into smaller pieces, and modeling the pieces as a collection of springs, as follows:

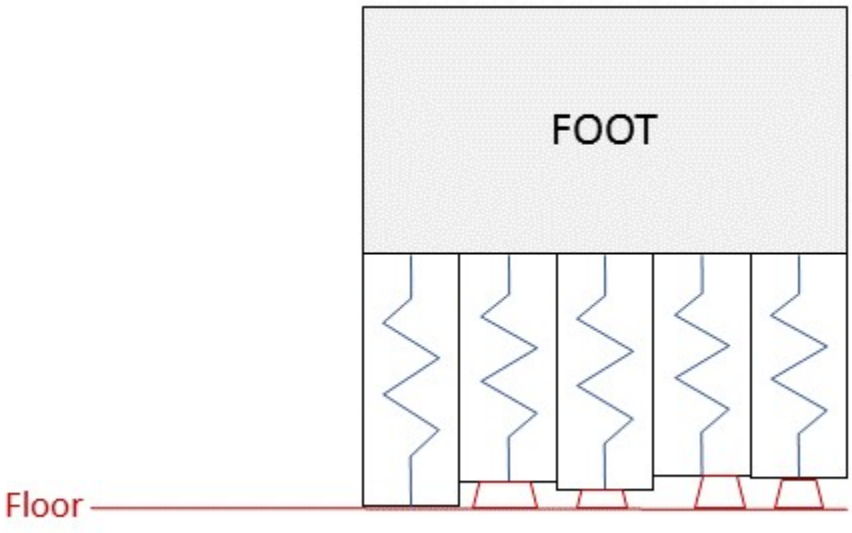

Figure 2: Foot Model

Here, the foot is trying to stand on flat ground, but there are small, surface asperities and irregularities. We will call the height of these asperities, with respect to the floor position r, as $T(r)$, and we will define the region T is defined over as G. Each spring constant is proportional to the cross-sectional area of each piece of foam- we will call the total area A, and the section of ground that each piece G_j. Define the height of the lower edge of the foot above the ground as a variable h.

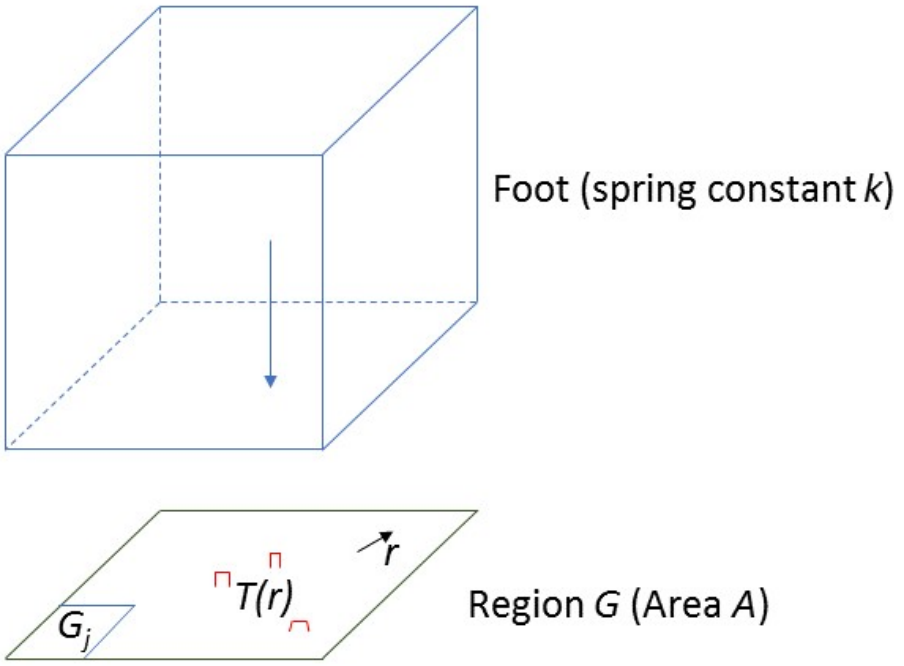

Figure 3: Illustration of Terms

When standing on perfectly flat ground (i.e. $T = 0$), the height of the foot is a constant, which we will call $\bar{x}$. For a flat, rigid foot with area A, the height of the bottom of the foot when standing on flat ground will be zero. When standing on the gravel, the height will be $\underset{r \in G}{max}\, T(r)$, which we will define as T_{max}. This means that if a control system for a rigid-footed robot assumes the future location of its foot to be flat on the floor, the gravel will cause its actual position to be T_{max} above the floor, introducing T_{max} amount of error. When the foam is added to the foot, on flat ground, we can solve for $\bar{x}$ as follows:

$$k\bar{x} = mg \rightarrow \bar{x} = \frac{mg}{k}$$

When using the model in figure 2, we get the following equation for h:

$$\sum_j k \frac{G_j}{A} \left(h - \underset{r \in G_j}{max} T(r) \right) = mg = k\bar{x}$$

As we divide the foam into infinitesimally small pieces, this sum becomes a surface integral

$$\iint_{r \in G} \frac{k}{A}\big(h - T(r)\big)dr = k\bar{x}$$

As the foot is stationary, we can simplify the equation as follows:

$$kh - k\iint_{r \in G} \frac{T(r)}{A}dr = k\bar{x}$$

$\iint_{r \in G} \frac{T(r)}{A}dr$ is simply the average height across T, which we will define as T_{avg}, so the equation becomes

$$h = \bar{x} + T_{avg}$$

This means that if a control system for a foam-footed robot assumes the future location of its foot to be $\bar{x}$ above the ground, the gravel will cause it to be $\bar{x}$ + T_{avg} above the floor, introducing T_{avg} amount of error. For a few scattered pieces of gravel, T_{avg} will be much smaller than T_{max}, so if our hypothesis is experimentally verifiable, this will mean that foam coverings can be used to improve the performance of robotic locomotion control systems.

3.2 Experiment Design

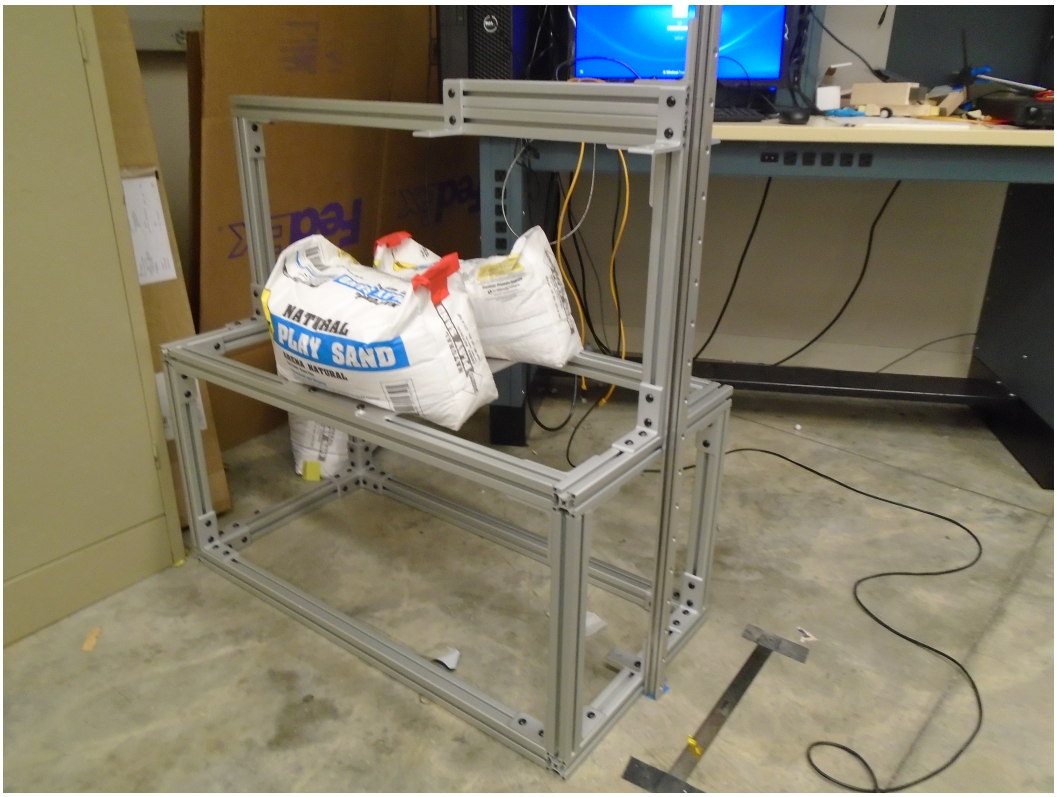

Figure 4: Experimental Testbed

We built the framework shown in figure 4 to hold a vertical guide and carriage steady. The framework was built out of off-the-shelf 40 mm x 40 mm 80/20 parts, and measures 1 m in length, .5 m in width, and 1.04 in height (not counting the height of the guide). To prevent error caused by motion of the frame due to the force of the leg's movements, two 50 lb. bags of sand were added to dampen any vibrations. The leg was attached to the carriage and positioned relative to a ruler secured to the ground as follows:

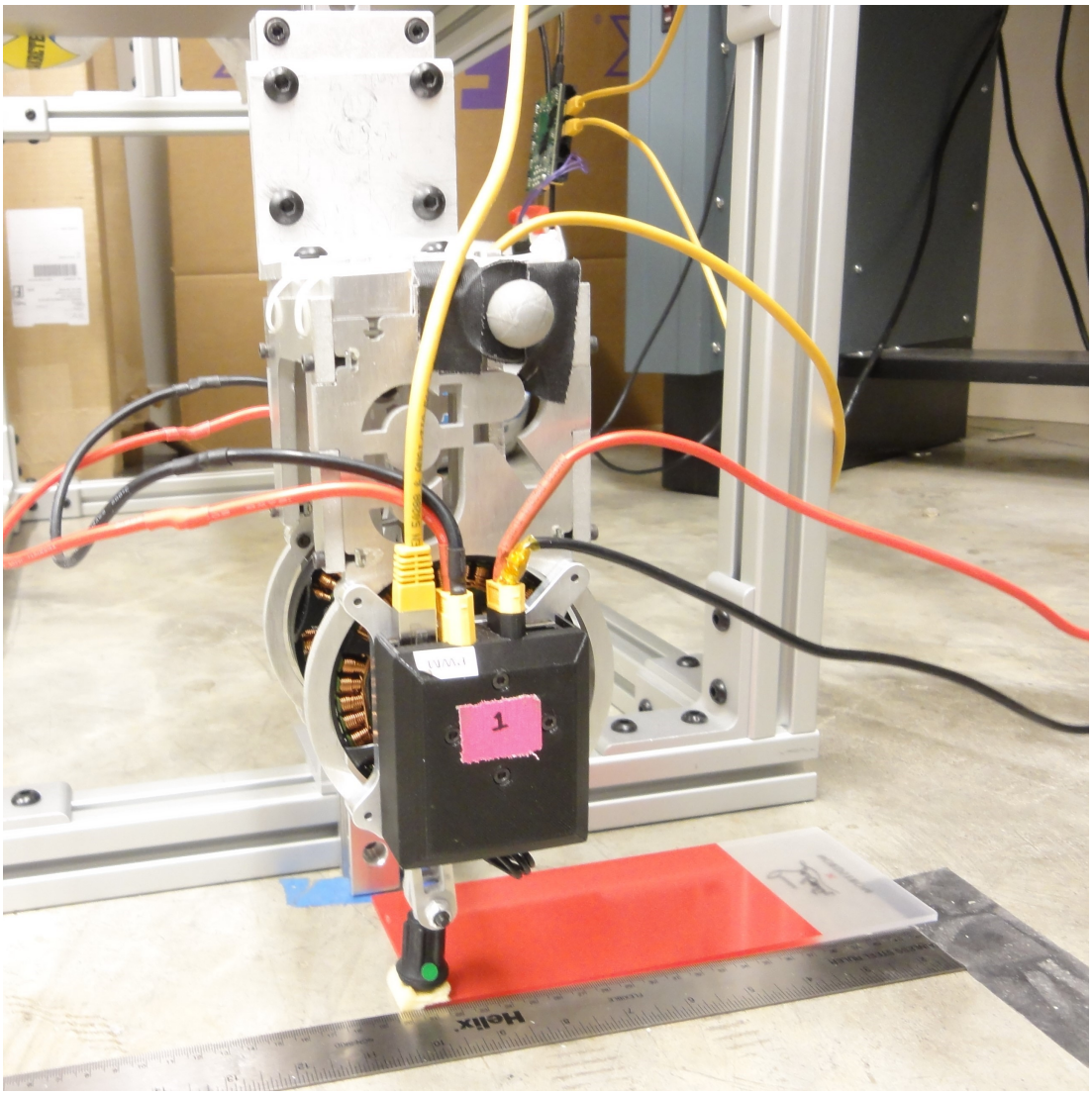

Figure 5: Experiment Setup

With the leg attached in this manner, the hip can move up and down along the vertical axis, and the foot can move in the x-z plane, as controlled by the Arduino. The leg was powered by a BK Precision 9115 power supply producing 16 V and 50 A. Arduino code routines to command the leg to stay stationary (as pictured in Figure 5) and to jump up down were written, and attached as Appendix C.

The experiments we ran all followed the same basic form- the leg was lifted in the air, the edge of a step was positioned relative to the ruler, the leg was lowered and left to stand freely on

its own, and the height of the body above the ground was measured with a Qualsys Oqus 510 high-speed motion capture camera. We conducted these experiments for two different foot styles (2 cm and 3 cm cubes), and two different ledge heights, (3 mm and 12.8 mm). For all experiments, the step edge was moved in increments of 1/10 of the cube's width (either 2 mm or 3 mm, as appropriate), from right to left, starting with the foot completely off the ledge and ending with the foot completely on the ledge.

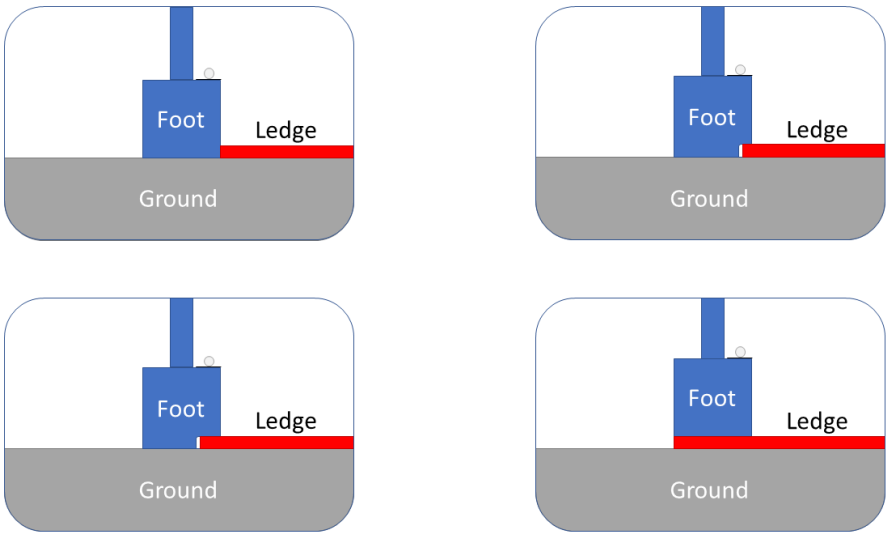

Figure 6: Steps of the Experiment

As the area of the ledge under the foot increased linearly with each movement, the height of the robot body should increase linearly as well, as predicted by the equations we derived in the Hypothesis section. Therefore, the metric we chose to measure the applicability of the model to each scenario tested was the coefficient of determination (R^2 value), which measures how well simple linear regression explains the data.

We originally attempted to run the experiment using the robotic leg, the combined weight of the leg and carriage (2.377 kg) was compressed all the foam feet to such a degree that it became immediately apparent that their behavior did not comply with the simple spring model,

even in the case of the stiffest foams. To improve the data quality, we replaced the leg with balsa wood dowels with the foam feet glued to the bottom. The carriage and dowel combination weighed 1.056 kg when a small dowel was used, and 1.087 kg when a large dowel was used.

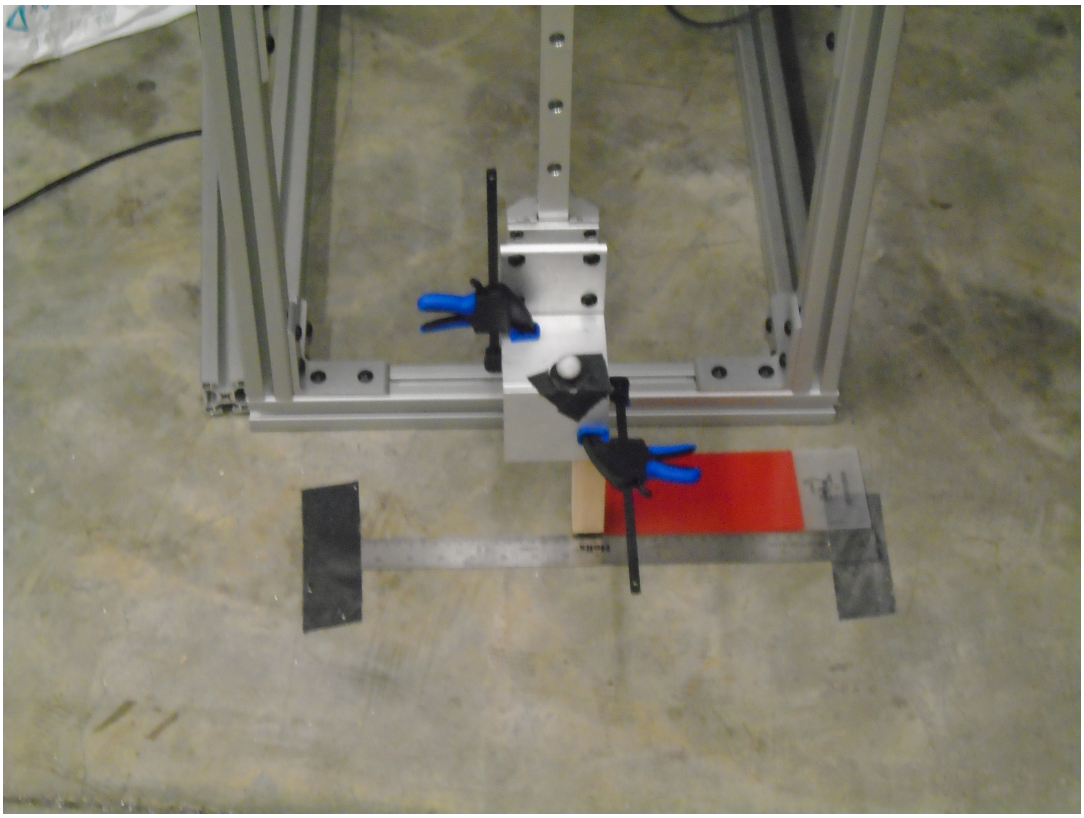

Figure 7: Revised Experimental Setup

3.3 Foot Types

The leg terminates in a square metal prong that is sized to accommodate standard McMaster-Carr 2517T32 rubber non-skid tips, which are approximately 2 cm in diameter. We chose the foams we used based on their firmness, as follows:

Foam Type	Firmness (PSI)
35280	.32-.67

60280	.55-.65
80280	.75-.85
200100	.90-1.05
300135	1.05-1.50

Table 1: Foam Type vs. Firmness

The firmness values given here are per the Polyurethane Foam Association's 25% deflection test [13]. They refer to the force necessary to cause a 25% indentation in the foam at the thickness sold. All foam samples used in this experiment were sold in sheets three inches thick, and cut to shape using a hot wire foam cutter.

Chapter 4. RESULTS AND ANALYSIS

4.1 RESULTS

Our experiments on a 3 mm ledge yielded data in the following form:

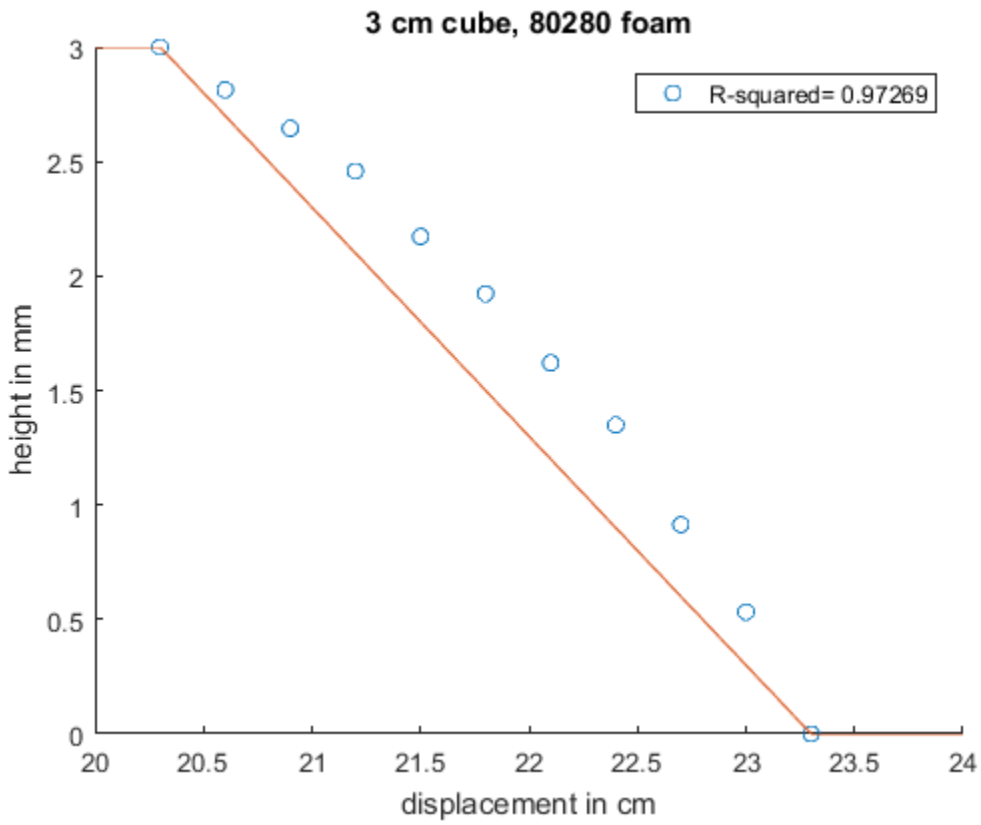

Figure 8: Sample Data

Here, the orange line represents the behavior predicted by the simple spring model, and the blue dots represent the experimental data. All experimental data is attached as Appendix B. We ran the experiment for all foam types for both foot sizes and ledge heights, and generated the following data:

Foam \ Ledge Height	2 cm cube	3 cm cube
35280	.8422	.8978
60280	.8428	.9103
80280	.8652	.9727
200100	.9191	.9822

300135	.9235	.9917

Table 2: 3 mm Ledge Data

Foam \ Ledge Height	2 cm cube	3 cm cube
35280	.4326	.5356
60280	.4456	.6386
80280	.4761	.7182
200100	.4877	.7244
300135	.6176	.7821

Table 3: 12.8 mm Ledge Data

4.2 ANALYSIS

Raising the ledge height and reducing the size of the foot both resulted in decreased performance. Regardless of the ledge height or foot size, in all cases, increasing foam stiffness improved performance. The largest source of error appeared to be the gap between the ledge and where the foam first contacted the ground- a good measure for how much this affects performance is the ratio between ledge height and foot edge length (α). This larger this value was, the worse performance became. For all ledge height and foot size combinations, foot performance responded to softening foam as follows:

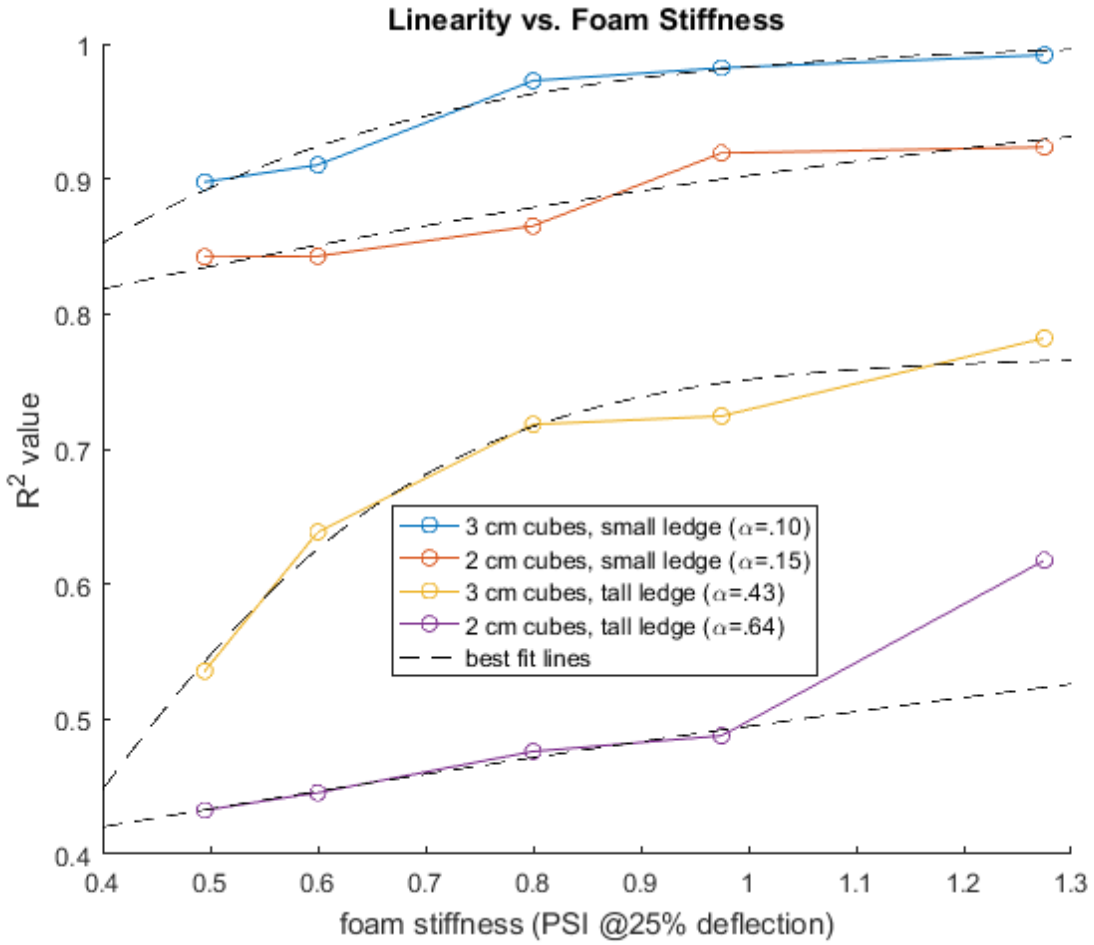

Figure 9: Linearity vs. Foam Stiffness (All Conditions)

Fitting sigmoid curves of the form $\frac{a}{1+e^{-b(x+c)}}$ to all data sets individually using simple least-squares regression, we get the following results (for $\alpha = .64$, this required weighting the individual data points to achieve reasonable results):

	a	b	c
12.8 mm ledge, 2 cm foot ($\alpha = .64$)	.6671	.8706	.2107
12.8 mm ledge, 3 cm foot ($\alpha = .43$)	.7688	5.7182	-.3417
3 mm ledge, 2 cm foot ($\alpha = .15$)	1.0146	1.0953	.9042
3 mm ledge, 3 cm foot ($\alpha = .10$)	1.0023	3.6489	.0764

Table 3: Sigmoid Curve Coefficients

There is no clear pattern to the constants in these sigmoid curves, so no model to predict the response to foam stiffness for a given ledge height and foot size can be created. However, sigmoid curve models do fit the data quite well for reasonable α values, so for a known foot size and terrain irregularity height, this setup could be used to run experiments to find an accurate model for how linearity is related to foam stiffness under those specific conditions, which could be used as a design curve.

Chapter 5. APPLICATIONS AND CONCLUSIONS

These experiments show that there are circumstances in which foam feet smooth out discontinuities in a predictable manner. However, the regime in which this is the case is narrowly bounded in terms of foam stiffness, discontinuity size, and foot size. Although there may be robots for which foam feet could fall in those regimes, foam feet do not appear to be a useful modification for a Minitaur-type robot.

These experiments only looked at static conditions- foam may behave in a more tractable manner in dynamic experiments. Data from such experiments could be used to further the search for a way to quantify agility [1]. Such experiments could be conducted on the testbed created for this research, using the Arduino code from Appendix C, if a means of consistently altering the terrain under a jumping robot could be created. Other possible experiments in dynamic conditions could examine how the smoothing effect affects trajectory differentiability [10], especially as foam feet would allow for limb decoupling even in situations where there are design constraints that make it impossible to modify the limbs where they attach to the body. The

testbed also has potential applications for unrelated experiments in areas such as rehabilitation- it could be used to test designs for prosthetic feet.

BIBLIOGRAPHY

[1] Duperret, J.M., Kenneally, G.D., Pusey, J.L., Koditschek, D.E. “Towards a Comparative Measure of Legged Agility.” Proceedings of the International Symposium on Experimental Robotics. 2014.

[2] Eng, J.J., Winter, D.A., Patla, A.E. “Strategies for recovery from a trip in early and late swing during human walking.” Experimental Brain Research 102 (1994) 339-349.

[3] Hauser, H., Sumioka, H., Füchslin, R., Pfeifer, R. "Introduction to the Special Issue on Morphological Computation." Artificial Life 19 (2013) 1-8.

[4] Hubicki, C.M., Aguilar, J.J., Goldman, D.I., Ames, A.D. “Tractable Terrain-Aware Motion Planning on Granular Media: An Impulsive Jumping Study.” Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). 2016.

[5] Jindrich, D., Full, R. “Dynamic Stabilization of Rapid Hexapodal Locomotion.” Journal of Experimental Biology 205 (2002) 2803-2823.

[6] Kenneally, G., De, A., Koditschek, D.E. “Design Principles for a Family of Direct-Drive Legged Robots.” IEEE Robotics and Automation Letters 1.2 (2016) 900-907.

[7] Kenneally, G. and Koditschek, D.E. "Leg Design for Energy Management in an Electromechanical Robot." 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Congress Center Hamburg, Sept 28 - Oct 2, 2015.

[8] Krotkov, E. “Active Perception for Legged Locomotion: Every Step is an Experiment.” Proceedings of the IEEE International Symposium on Intelligent Control. 1990.

[9] Li, C., Umbanhowar, P.B., Komsuoglu, H., Koditschek, D.E., Goldman, D.I. “Sensitive Dependence of the Motion of a Legged Robot on Granular Media.” Proceedings of the National Academy of Sciences, U.S.A. 2009.

[10] Li, C., Zhang, T., Goldman, D.I. “A Terradynamics of Legged Locomotion on Granular Media.” Science, Vol. 339 (2013) 1408-1412.

[11] Pace, A.M., Burden, S.A. “Decoupled Limbs Yield Differentiable Trajectory Outcomes Through Intermittent Contact in Locomotion and Manipulation.” Proceedings of the IEEE International Conference on Robotics and Automation (ICRA). 2017.

[12] Podio-Guidugli, P. “A Primer in Elasticity,” in Journal of Elasticity 58, 2000.

[13] Polyurethane Foam Association Joint Industry Foam Standards and Guidelines, http://www.pfa.org/jifsg/jifsgs4.html, July 1994.

[14] Sastra, J., Revzen, S., Yim, M. “Softer Legs Allow a Modular Hexapod to Run Faster.” Adaptive Mobile Robotics, 2012.

[15] Spenko, M.J., Haynes, G.C., Saunders, J.A., Cutkosky, M.R., Rizzi, A.A., Full, R.J., Koditschek, D.E. “Biologically Inspired Climbing with a Hexapedal Robot.” Journal of Field Robotics 25 (2008) 223-242.

APPENDIX A

The following function (A2P.m) takes the angles of a Ghost Robotics leg and determines the x-y coordinates of the foot joint (xf, yf) relative to the leg's hip, for the regime where the knees are extended past the hip (as opposed to the dynamics calculated in [7]).

```
function [xf,yf]=A2P(theta1,theta2)
close all

seg1=10;
seg2=20;

%IMPORTANT: all angles are measured clockwise from the negative y-axis

%theta1 must be between -pi/2 and pi
if (theta1<-pi/2) | (theta1>pi)
    error('theta1 must be in [-pi\2,pi]')
end

%theta2 must be between -pi and pi/2
if (theta2<-pi) | (theta2>pi/2)
    error('theta1 must be in [-pi,pi/2]')
end

%theta1 must be larger than theta2
if (theta1<theta2)
    error('theta1 must be larger than theta2')
end

thetabar=(theta1+theta2)/2;
dtheta=(theta1-theta2)/2;

%find position of leg1 knee
xk1=-seg1*sin(dtheta);
yk1=-seg1*sin(pi/2-dtheta);

%find position of leg2 knee
xk2=-xk1;
yk2=yk1;

%find position of foot
xf=0;
yf=yk1-sqrt(seg2^2-xk1^2);

%rotate counterclockwise
rotmat=[cos(thetabar) -sin(thetabar); sin(thetabar) cos(thetabar)];
pointmat=[xk1 xk2 xf; yk1 yk2 yf];
rotpointmat=rotmat*pointmat;
xk1=-rotpointmat(1,1);
xk2=-rotpointmat(1,2);
```

```
xf=-rotpointmat(1,3);
yk1=rotpointmat(2,1);
yk2=rotpointmat(2,2);
yf=rotpointmat(2,3);

%plot position of leg
figure(1)
hold on
plot([0 xk2],[0 yk2],'b')
plot([0 xk1],[0 yk1],'r')
plot([xk2 xf],[yk2 yf],'b')
plot([xk1 xf],[yk1 yf],'r')
legend('leg 1','leg 2')
axis equal
title(['Foot position= (' num2str(xf) ', ' num2str(yf) ')'])
```

APPENDIX B

The data I collected is attached here in the form of the code used to plot it:

```
close all

% 2 cm cube, small ledge, 300135 foam
x1=21.6:.2:23.6;
y1=-[414.06 414.44 415.13 416.34 416.79 418.17 419.13 420.24 422.52 425.55
428.18];
y1=y1+428.18;
scale1=y1(1)-y1(end);
y1=3*y1/scale1;
figure(1)
hold on
plot(x1,y1,'o')
plot([21 21.6 23.6 24],[3 3 0 0])
xlabel('displacement in cm')
ylabel('height in mm')
title('2 cm cube, 300135 foam')
lm=fitlm(x1,y1);
Rsq1=lm.Rsquared.Ordinary;
legend(['R-squared= ' num2str(Rsq1)])

% 2 cm cube, small ledge, 200100 foam
x2=21.6:.2:23.6;
% y2=-[353.11 353.24 353.64 354.28 354.83 355.62 355.93 357.56 359.25 361.25
367.38];
% y2=y2+367.38;
y2=-[342.24 342.62 342.80 343.73 344.49 345.49 346.55 347.90 349.45 350.97
354.62];
y2=y2+354.62;
scale2=y2(1)-y2(end);
y2=3*y2/scale2;
figure(2)
hold on
plot(x2,y2,'o')
plot([21 21.6 23.6 24],[3 3 0 0])
xlabel('displacement in cm')
ylabel('height in mm')
title('3 cm cube, 200100 foam')
lm=fitlm(x2,y2);
Rsq2=lm.Rsquared.Ordinary;
legend(['R-squared= ' num2str(Rsq2)])

% 2 cm cube, small ledge, 80280 foam
x3=21.3:.2:23.3;
y3=-[424.62 425.45 425.90 426.83 427.87 428.28 429.42 430.14 432.94 436.77
440.53];
y3=y3+440.53;
scale3=y3(1)-y3(end);
y3=12.8*y3/scale3;
figure(3)
hold on
```

```
plot(x3,y3,'o')
plot([21 21.3 23.3 24],[12.8 12.8 0 0])
xlabel('displacement in cm')
ylabel('height in mm')
title('2 cm cube, 80280 foam')
lm=fitlm(x3,y3);
Rsq3=lm.Rsquared.Ordinary;
legend(['R-squared= ' num2str(Rsq3)])

% 2 cm cube, small ledge, 60280 foam
x4=21.5:.2:23.5;
y4=-[433.53 433.70 434.18 435.05 435.74 436.87 437.70 439.36 441.12 443.81
449.96];
y4=y4+449.96;
scale4=y4(1)-y4(end);
y4=3*y4/scale4;
figure(4)
hold on
plot(x4,y4,'o')
plot([21 21.5 23.5 24],[3 3 0 0])
xlabel('displacement in cm')
ylabel('height in mm')
title('2 cm cube, 60280 foam')
lm=fitlm(x4,y4);
Rsq4=lm.Rsquared.Ordinary;
legend(['R-squared= ' num2str(Rsq4)])

% 2 cm cube, small ledge, 35280 foam
x5=21.6:.2:23.6;
y5=-[345.93 346.18 346.83 347.42 348.21 349.04 349.52 350.93 352.66 354.59
360.14];
y5=y5+360.14;
scale5=y5(1)-y5(end);
y5=3*y5/scale5;
figure(5)
hold on
plot(x5,y5,'o')
plot([21 21.6 23.6 24],[3 3 0 0])
xlabel('displacement in cm')
ylabel('height in mm')
title('2 cm cube, 35280 foam')
lm=fitlm(x5,y5);
Rsq5=lm.Rsquared.Ordinary;
legend(['R-squared= ' num2str(Rsq5)])


close all

% 3 cm cube, small ledge, 300135 foam
x1=[20 20.4:.1:23.4  24];
y1=-[804.33 804.33 804.78 805.03 805.20 805.93 806.73 807.53 807.67 807.91...
    808.68 809.37 810.20 810.52 811.11 811.84 812.78 813.47 813.92 814.38...
    815.11 815.49 816.22 816.95 817.82 818.17 818.86 819.49 820.29 820.88...
    822.09 823.52 823.52];
y1=y1+823.52;
scale1=y1(1)-y1(end);
```

```
y1=3*y1/scale1;
x1=20.4:.3:23.4;
y1=y1([2 5 8 11 14 17 20 23 26 29 32]);
figure(1)
hold on
plot(x1,y1,'o')
plot([20 20.4 23.4 24],[3 3 0 0])
xlabel('displacement in cm')
ylabel('height in mm')
title('3 cm cube, 300135 foam')
lm=fitlm(x1,y1);
Rsq1=lm.Rsquared.Ordinary;
legend(['R-squared= ' num2str(Rsq1)])

% 3 cm cube, small ledge, 200100 foam
x2=[20.5:.3:23.5];
y2=-[383.56 384.08 385.56 386.91 388.43 389.57 391.08 393.67 395.19 397.43
399.81];
y2=y2+399.81;
scale2=y2(1)-y2(end);
y2=3*y2/scale2;
figure(2)
hold on
plot(x2,y2,'o')
plot([20 20.5 23.5 24],[3 3 0 0])
xlabel('displacement in cm')
ylabel('height in mm')
title('3 cm cube, 200100 foam')
lm=fitlm(x2,y2);
Rsq2=lm.Rsquared.Ordinary;
legend(['R-squared= ' num2str(Rsq2)])


% 3 cm cube, small ledge, 80280 foam
x3=[20.3:.3:23.3];
y3=-[399.71 400.92 402.02 403.23 405.09 406.71 408.68 410.44 413.27 415.75
419.21];
y3=y3+419.21;
scale3=y3(1)-y3(end);
y3=3*y3/scale3;
figure(3)
hold on
plot(x3,y3,'o')
plot([20 20.3 23.3 24],[3 3 0 0])
xlabel('displacement in cm')
ylabel('height in mm')
title('3 cm cube, 80280 foam')
lm=fitlm(x3,y3);
Rsq3=lm.Rsquared.Ordinary;
legend(['R-squared= ' num2str(Rsq3)])


% 3 cm cube, small ledge, 60280 foam
x4=19.8:.3:22.8;
y4=-[392.08 392.26 393.12 393.64 394.85 395.95 397.81 399.81 401.26 404.09
408.78];
```

```
y4=y4+408.78;
scale4=y4(1)-y4(end);
y4=3*y4/scale4;
figure(4)
hold on
plot(x4,y4,'o')
plot([19 19.8 22.8 23.5],[3 3 0 0])
xlabel('displacement in cm')
ylabel('height in mm')
title('3 cm cube, 60280 foam')
lm=fitlm(x4,y4);
Rsq4=lm.Rsquared.Ordinary;
legend(['R-squared= ' num2str(Rsq4)])


% 3 cm cube, small ledge, 35280 foam
x5=19.8:.3:22.8;
y5=-[407.20 407.33 407.75 408.54 409.54 410.23 412.23 413.99 415.93 418.48
422.76];
y5=y5+422.76;
scale5=y5(1)-y5(end);
y5=3*y5/scale5;
figure(5)
hold on
plot(x5,y5,'o')
plot([19 19.8 22.8 23.5],[3 3 0 0])
xlabel('displacement in cm')
ylabel('height in mm')
title('3 cm cube, 35280 foam')
lm=fitlm(x5,y5);
Rsq5=lm.Rsquared.Ordinary;
legend(['R-squared= ' num2str(Rsq5)])


close all

% 2 cm cube, tall ledge, 300135 foam
x1=21.3:.2:23.3;
y1=-[303.60 304.70 305.63 306.84 308.18 310.25 312.70 314.53 316.73 336.38
368.71];
y1=y1+368.71;
scale1=y1(1)-y1(end);
y1=12.8*y1/scale1;
figure(1)
hold on
plot(x1,y1,'o')
plot([21 21.3 23.3 24],[12.8 12.8 0 0])
xlabel('displacement in cm')
ylabel('height in mm')
title('3 cm cube, 300135 foam')
lm=fitlm(x1,y1);
Rsq1=lm.Rsquared.Ordinary;
legend(['R-squared= ' num2str(Rsq1)])

% 2 cm cube, tall ledge, 200100 foam
x2=21.4:.2:23.4;
```

```
y2=-[296.33 296.67 296.88 297.91 299.01 300.29 302.39 304.15 306.46 310.63
356.62];
y2=y2+356.62;
scale2=y2(1)-y2(end);
y2=12.8*y2/scale2;
figure(2)
hold on
plot(x2,y2,'o')
plot([21 21.4 23.4 24],[12.8 12.8 0 0])
xlabel('displacement in cm')
ylabel('height in mm')
title('2 cm cube, 200100 foam')
lm=fitlm(x2,y2);
Rsq2=lm.Rsquared.Ordinary;
legend(['R-squared= ' num2str(Rsq2)])

% 2 cm cube, tall ledge, 80280 foam
x3=21.3:.2:23.3;
y3=-[379.11 379.28 379.60 380.94 381.87 382.80 383.87 385.49 387.98 395.57
441.05];
y3=y3+441.05;
scale3=y3(1)-y3(end);
y3=12.8*y3/scale3;
figure(3)
hold on
plot(x3,y3,'o')
plot([21 21.3 23.3 24],[12.8 12.8 0 0])
xlabel('displacement in cm')
ylabel('height in mm')
title('2 cm cube, 80280 foam')
lm=fitlm(x3,y3);
Rsq3=lm.Rsquared.Ordinary;
legend(['R-squared= ' num2str(Rsq3)])

% 2 cm cube, tall ledge, 60280 foam
x4=21.2:.2:23.2;
y4=-[322.42 322.52 323.14 323.87 325.04 326.32 327.94 330.14 333.07 334.45
389.77];
y4=y4+389.77;
scale4=y4(1)-y4(end);
y4=12.8*y4/scale4;
figure(4)
hold on
plot(x4,y4,'o')
plot([20 21.2 23.2 24],[12.8 12.8 0 0])
xlabel('displacement in cm')
ylabel('height in mm')
title('3 cm cube, 60280 foam')
lm=fitlm(x4,y4);
Rsq4=lm.Rsquared.Ordinary;
legend(['R-squared= ' num2str(Rsq4)])

% 2 cm cube, tall ledge, 35280 foam
x5=21.5:.2:23.5;
y5=-[312.91 313.59 314.18 314.39 315.35 316.21 317.08 318.73 320.80 327.90
379.28];
```

```
y5=y5+379.28;
scale5=y5(1)-y5(end);
y5=12.8*y5/scale5;
figure(5)
hold on
plot(x5,y5,'o')
plot([21 21.5 23.5 24],[12.8 12.8 0 0])
xlabel('displacement in cm')
ylabel('height in mm')
title('2 cm cube, 35280 foam')
lm=fitlm(x5,y5);
Rsq5=lm.Rsquared.Ordinary;
legend(['R-squared= ' num2str(Rsq5)])


close all

% 3 cm cube, tall ledge, 300135 foam
x1=20.1:.3:23.1;
y1=-[334.62 335.35 337.38 340.87 344.00 346.80 350.28 354.11 359.35 368.70
399.33];
y1=y1+399.33;
scale1=y1(1)-y1(end);
y1=12.8*y1/scale1;
figure(1)
hold on
plot(x1,y1,'o')
plot([19 20.1 23.1 24],[12.8 12.8 0 0])
xlabel('displacement in cm')
ylabel('height in mm')
title('3 cm cube, 300135 foam')
lm=fitlm(x1,y1);
Rsq1=lm.Rsquared.Ordinary;
legend(['R-squared= ' num2str(Rsq1)])

% 3 cm cube, tall ledge, 200100 foam
x2=20:.3:23;
y2=-[340.56 340.93 342.49 344.87 346.83 349.76 352.49 356.90 362.18 371.18
404.19];
y2=y2+404.19;
scale2=y2(1)-y2(end);
y2=12.8*y2/scale2;
figure(2)
hold on
plot(x2,y2,'o')
plot([19 20 23 24],[12.8 12.8 0 0])
xlabel('displacement in cm')
ylabel('height in mm')
title('3 cm cube, 200100 foam')
lm=fitlm(x2,y2);
Rsq2=lm.Rsquared.Ordinary;
legend(['R-squared= ' num2str(Rsq2)])

% 3 cm cube, tall ledge, 80280 foam
x3=19.8:.3:22.8;
```

```
y3=-[359.56 360.38 361.42 363.90 366.21 368.59 371.25 374.87 379.11 402.61
426.38];
y3=y3+426.38;
scale3=y3(1)-y3(end);
y3=12.8*y3/scale3;
figure(3)
hold on
plot(x3,y3,'o')
plot([19 19.8 22.8 24],[12.8 12.8 0 0])
xlabel('displacement in cm')
ylabel('height in mm')
title('3 cm cube, 80280 foam')
lm=fitlm(x3,y3);
Rsq3=lm.Rsquared.Ordinary;
legend(['R-squared= ' num2str(Rsq3)])

% 3 cm cube, tall ledge, 60280 foam
x4=19.6:.3:22.6;
y4=-[362.31 362.59 362.66 363.97 366.28 369.70 371.56 375.46 379.32 387.46
426.80];
y4=y4+426.80;
scale4=y4(1)-y4(end);
y4=12.8*y4/scale4;
figure(4)
hold on
plot(x4,y4,'o')
plot([19 19.6 22.6 23],[12.8 12.8 0 0])
xlabel('displacement in cm')
ylabel('height in mm')
title('3 cm cube, 60280 foam')
lm=fitlm(x4,y4);
Rsq4=lm.Rsquared.Ordinary;
legend(['R-squared= ' num2str(Rsq4)])

% 3 cm cube, tall ledge, 35280 foam
x5=19.5:.3:22.5;
y5=-[356.25 356.80 357.66 358.97 360.52 362.04 363.94 366.83 370.04 375.21
423.24];
y5=y5+423.24;
scale5=y5(1)-y5(end);
y5=12.8*y5/scale5;
figure(5)
hold on
plot(x5,y5,'o')
plot([19 19.5 22.5 23],[12.8 12.8 0 0])
xlabel('displacement in cm')
ylabel('height in mm')
title('2 cm cube, 35280 foam')
lm=fitlm(x5,y5);
Rsq5=lm.Rsquared.Ordinary;
legend(['R-squared= ' num2str(Rsq5)])


xf=[.495 .6 .8 .975 1.275];
y2s=[.8422 .8428 .8652 .9191 .9235];
y2t=[.4326 .4456 .4761 .4877 .6176];
```

```
y3s=[.8978 .9103 .9727 .9822 .9917];
y3t=[.5356 .6386 .7182 .7244 .7821];

x=.4:.0001:1.3;
fit3t=.7688./(1+exp(-5.7182*(x-.3417)));
fit2s=1.0146./(1+exp(-1.0953*(x+.9042)));
fit3s=1.0023./(1+exp(-3.6489*(x+.0764)));
fit2t=.6671./(1+exp(-.8706*(x+.2107)));

figure(1)
hold on
plot(xf,y3s,'o-')
plot(xf,y2s,'o-')
plot(xf,y3t,'o-')
plot(xf,y2t,'o-')
xlabel('foam stiffness (PSI @25% deflection)')
ylabel('R^2 value')
plot(x,fit2s,'k--')
plot(x,fit2t,'k--')
plot(x,fit3s,'k--')
plot(x,fit3t,'k--')
legend('3 cm cubes, small ledge','2 cm cubes, small ledge','3 cm cubes, tall
ledge','2 cm cubes, tall ledge', 'best fit lines')
title('Linearity vs. Foam Stiffness')
```

APPENDIX C

The Arduino routines to make the leg stand (PositionerFinal) and jump (JumperFinal) are attached here:

POSITIONERFINAL

```
#include <Motor.h>

const uint8_t outPin[] = {PA1, PB10};
const uint8_t inPin[] = {PB2, PB0};
const float motZeros[] = {4.719, 6.102};
const int NMOT = 2;
BlCon34 M[NMOT];

void controlLoop() {
  for (int i=0; i<NMOT; ++i)
    M[i].update();
}

void debug() {
  for (int i=0; i<NMOT; ++i)
    Serial1 << _FLOAT(M[i].getPosition(),4) << "\t";
  for (int i=0; i<NMOT; ++i) {
    M[i].setTorqueEstParams(0.0954, 0.186, 16);
    Serial1 << _FLOAT(M[i].getTorque(),4) << "\t";
//    Serial1 << _FLOAT(M[i].getOpenLoop(),4) << "\t";
//    Serial1 << _FLOAT(M[i].update(),2) << "\t";
}
  Serial1 << "\n";
}

void setup() {
  Serial1.begin(115200);

  Motor::updateRate = 1000;
  Motor::velSmooth = 0.95;
  BlCon34::useEXTI = true;
  for (int i=0; i<NMOT; ++i)
    M[i].init(outPin[i], inPin[i], motZeros[i], -1);

  attachTimerInterrupt(0, controlLoop, 1000);
  attachTimerInterrupt(1, debug, 50);

  for (int i=0; i<NMOT; ++i) {
```

```
    M[i].enable(true);
  }

  delay(100);
}

void loop() {

  delay(1000);
    for (int i=0; i<NMOT; ++i) {
    M[i].setGain(0.3);
 //   M[i].setPosition(1);
  }
  M[0].setPosition(.2);
  M[1].setPosition(.2);
}
```

JUMPERFINAL

```
#include <Motor.h>

const uint8_t outPin[] = {PA1, PB10};
const uint8_t inPin[] = {PB2, PB0};
const float motZeros[] = {4.719, 6.102};
const int NMOT = 2;
BlCon34 M[NMOT];

void controlLoop() {
  for (int i=0; i<NMOT; ++i)
    M[i].update();
}

void debug() {
  for (int i=0; i<NMOT; ++i)
    Serial1 << _FLOAT(M[i].getPosition(),4) << "\t";
  for (int i=0; i<NMOT; ++i) {
    M[i].setTorqueEstParams(0.0954, 0.186, 16);
    Serial1 << _FLOAT(M[i].getTorque(),4) << "\t";
//    Serial1 << _FLOAT(M[i].getOpenLoop(),4) << "\t";
    Serial1 << _FLOAT(M[i].update(),2) << "\t";
}
  Serial1 << "\n";
}

void setup() {
  Serial1.begin(115200);
```

```
  Motor::updateRate = 1000;
  Motor::velSmooth = 0.95;
  BlCon34::useEXTI = true;
  for (int i=0; i<NMOT; ++i)
    M[i].init(outPin[i], inPin[i], motZeros[i], -1);

  attachTimerInterrupt(0, controlLoop, 1000);
  attachTimerInterrupt(1, debug, 50);

  for (int i=0; i<NMOT; ++i) {
    M[i].enable(true);
  }

  delay(100);
}

void loop() {

  delay(250);
    for (int i=0; i<NMOT; ++i) {
    M[i].setGain(0.5);
    M[i].setPosition(.5);
  }
    delay(500);
    for (int i=0; i<NMOT; ++i) {
    M[i].setGain(0.5);
    M[i].setPosition(2.5);
  }

}
```