SIMULATION OF A SOFT BODY NON-GRASP MANIPULATOR USING A SPRING-MASS-DAMPER CLOTH MODEL

by

CRISTIAN ALMENDARIZ

Presented to the Faculty of the Graduate School of
The University of Texas at Arlington in Partial Fulfillment
of the Requirements for the Degree of

MASTER OF SCIENCE

in

MECHANICAL ENGINEERING

THE UNIVERSITY OF TEXAS AT ARLINGTON

August 2021

To my wife Karen Isabel, mother Perla Elizabeth, and my father Sergio Odilio for encouraging me and making me who I am.

ACKNOWLEDGEMENTS

I would like to thank my supervising professor Dr. Alan Bowling for guiding me through this academic endevour with outstanding patience, understanding, and motivation. I wish to also thank Dr. Endel Iarve and Dr. Kent Lawrence for their interest in my research and for taking time to serve in my thesis committee.

I would also like to extend my appreciation to those who acted as mentors through out my studies, Vatsal Joshi and Manoochehr Rabiei for your great support and input when I needed brilliant perspectives. To the other graduate students and members of the Robotics, Biomechanics, and Dynamic Systems Laboratory, Kangji Huang, Ayush Thapa, Thong Nguyen, Dr. Ashley Guy, and Dr. Abhishek Chatterjee, thank you for the support and encouragement.

I will always be forever grateful to all the teachers and mentors that have taught me during all the years I have spent in school. You have instilled in me a love for science and technology and an appreciation for curiosity and life long learning.

Finally, I would like to express my deep gratitude to my wife, Karen, for supporting and inspiring me every day. I am also extremely grateful to my mom, dad, brother, and sister, for their encouragement and patience. If I did not have all their love and support, I would not have been able to succeed in my pursuits.

August 10, 2021

ABSTRACT

SIMULATION OF A SOFT BODY NON-GRASP MANIPULATOR
USING A SPRING-MASS-DAMPER CLOTH MODEL

CRISTIAN ALMENDARIZ, M.S.

The University of Texas at Arlington, 2021

Supervising Professor: Alan Bowling

Simulation and modeling are well known and important tools in the field of robotics, being used to validate design and in the development of robotic control systems. Traditionally, robotic systems have been primarily rigid body systems and used in well controlled environments where interactions are highly predictable, and exceptions are minimized. With the ever-increasing adoption of robotics in the medical field, typical robotic environments are becoming highly unpredictable and requiring the modeling and simulation of not only rigid body systems but soft systems and interactions between the two.

This work addresses model improvement of a "soft and hard" robotic system for the intended use in the medical field. Previous teams in the RBDSL group conceptualized a "soft and hard" robotic manipulator for the reduction of pressure ulcers in immobile hospital patients. The force bed that was developed relied upon a rigid body parallel chain mechanism and a soft body air bladder to accomplish its manipulation of patients. The force bed air bladder model was improved upon using recent techniques in cloth modeling and simulation. Results show an intuitive matching

with the real-world physical force bed and the improved simulated force bed model.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Soft Robotics is a relatively new field where traditional robotics is supplemented with soft body manipulators for more delicate and potentially dexterous manipulation, or are entirely replaced with complete soft body robots for other advantageous reasons. The applications of soft robotics ranges just as widely as traditional systems with plenty of opportunity for refinement and development. Developing dynamic models for simulation and control of soft robotics is one such area of interest.
Techniques used in cloth modeling have been applied to soft robotics model development. The work presented focuses on using traditional cloth modeling techniques in the development of a soft body non-grasp manipulator. The results of which are promising, given that the resulting simulations are shown to be valid as well as visually representative of the physical prototype that the system was modeled after. Despite these gains, the implementation remains to have a need for improvement as certain physical attributes reflected in the model deviate from the prototype's physical properties.

1.1 Soft Robotics

Soft robotics uses various techniques to accurately model complex and highly deformable systems and their interactions. Traditional rigid body robots are well developed in the sense that the modeling, design, and sophisticated control systems are well documented and accesible for the majority of current applications [5]. Soft robotics is considered to still be well within it's infancy with experimental studies

being more attainable than computational dynamic models of the same systems. Due to large deformations and overall motion for most soft robotic systems to achieve the desired goals, more complex computationally dense dynamics are required than in traditional robotics where the inherent stiffness of the robot's rigid body is maintained throughout the application [6].

In addition to the computational intensity of the dynamics of soft robotics, there is no agreed upon general theory for the control of these unconstrained structures. It is not uncommon for soft robotics to take inspiration from nature such as the locomotion of various worms and catipillars or the anatomy and biomechanics of starfish or octupus appendages. Doing so allows for the exploitation of basic mechanical properties that are inherent to similar "soft" materials. Taking inspiration from nature brings forth it's own limitations, one such limitation being that the examples of nearly complete soft bodied animals are usually found in environments where they are within a fully supportive medium like water which aids their functions. To overcome this limitation, typical soft robotic systems take advantage of hybrid robots being comprised of "soft and hard" components [7]. Even still, the need to optimize these designs is currently being pursued. Caasenbrood worked out the implementation of topology optimization in the efficient design of soft robotics and built a model using nonlinear finite element methods [8]. Nealen also demonstrated success utilizing FEM techniques where he was able to produce physically accurate models using SMD systems, mesh free methods, and finite element/ difference/ volume methods [9]. Finite element tools have proven advantageous in the field of simulation and soft robotics or interacting with soft bodies. Especially in the development of real time surgical simulators where soft bodies utilizing cloth modeling techniques can be found [10] [11]. Huang was able to demonstrate faster than real time dynamic simulation of elastic appendages and their interactions between soft bodies and rigid surfaces. The

methods demonstrated were sourced from the popular computer graphics community and were able to still produce real time convergence and show numerical stability throughout the simulation [12]. Honji and Buso showed a real-world practical application of soft robotics by the effective use of real time contact force feedback control, which allowed for dexterous manipulation of various objects [13] [14].

There is current interest and progress in researching the various ways to model soft body robots and their interactions with each other and rigid systems [15] [16]. One such method involves the use of massive particles in a spring mass damper system, typically used for modeling of cloth material [3]. As this technique was developed in the computer graphics community, it has become well studied and optimized for real time simulation and thus, with a reasonable application, should be suitable for the use of modeling soft and hard system interactions for robotics [17]. Understanding the interaction between hard and soft robotics including the soft body materials with cloth like properties such as inflated bladders, membranes, or films, as demonstrated by Liang, can be a vital component in the overall field of soft robotics [18]. As shown in Hsiao, a spring-mass-damper model has successfully been used to implicitly simulate these cloth like bodies to an acceptable degree of authenticity [19]. As this work has been used, others such as Eldbadrawy and others have implemented various techniques in speeding the processing of these simulations [20].

1.2 Cloth Model Techniques

Early cloth modeling research and development was adopted for various applications where computers were used to generate images or video. Such applications included movies, videogames, and recently virtual reality and augmented reality. Companies and groups such as Pixar were early adopters of cloth modeling and simulation. Pixar was one of the first major adopters and developers utilizing

modeling and simulation techniques with *Geri's Game* where the clothing used by the animated character was a complete cloth model simulation. Where characters are hand animated in every scene to devlop their motions throughout the film, the cloth garments are then simulated over each scene. To use the current advances in the field of cloth modeling, Pixar developed their own software which would eventually become *FizT*, the physics engine behind many of their produced films [21]. Utilizing techniques adapted from Finite Element Methods, the evolution of cloth modeling has resulted in attainable accurate and fast simulations of cloth and soft bodies alike. Pixar continues to be a major force within the world of cloth modeling as their recent advances in software and technique continue the speed and resolution at which these cloth models can be simulated [25].

1.2.1 Finite Element Methods

In many computer graphics efforts, deformable bodies, such as those being described for use in soft robotics, have been successful through the implementation of different Finite Element Methods. The core of this method relies on the transformation of partial differential equations (PDEs) into a set of solvable algebraic equations using various numerical integration techniques. The bodies are represented as a continuum that is discretized using irregular meshes of varying topologies. The nodes of these meshes are chosen to represent mass points in the "explicit Finite Element approach" that is commonly used in computer graphics [9]. This form of FEM has proven useful in surgery simulations where accurate results of simulated soft bodied objects are required [10]. Cuboidal element voxelization, proposed by Muller, has been shown to work in simulating fracturing surfaces of FEMs [22]. Other Finite Element techniques used in soft body simulation include The Method of Finite Difference where bodies are discretized by a regularly spaced grid for optimization of the

equations of motion, The Finite Volume Method that utilizes a stress tensor to more easily determine internal forces, and The Boundary Element Method which takes advantage of transforming the equations of motion into a surface integral for decreased computation time. Each of these methods are not without their limitations as for example the Boundary Element Method is limited to bodies whose internals are of a homogenous material and when dealing with fractures results in significant computational difficulty [9]. Despite the proven utility of Finite Element / Volume Methods and Boundary Element Methods for applications in simulating soft bodies, for some, it has been shown that different techniques have proven to be faster and sufficient in accuracy. For the work demonstrated by Meier in the world of surgical simulation, the soft body models were initially proposed in FEM and BEM but eventually the study opted for the use of the faster spring-mass approach as will be described [23].

1.2.2 Geometric Model

Weil was one of the first to demonstrate a "realistic" rendering of cloth using a geometric focused approach. The focus of their work was to achieve an intuitive visual representation of hanging cloth for images and animations. They accomplished this through a two stage process. The first stage constrained the points on the cloth model where the cloth would hang from. A series of curves based on the estimated displacement of the hanging cloth model were then created to serve as the "backbone" for the rest of the model. A rectangular grid coordinate system was then overlaid onto the created shape. The second step used an iterative process for "relaxing" the grid system and creating a finer and finer mesh [1].

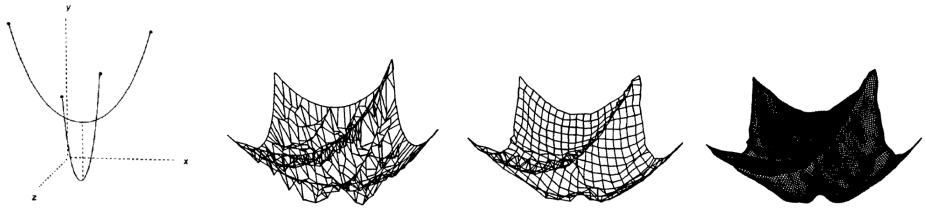


Figure 1.1: Geometric model for constrain hanging cloth [1].

1.2.3 Spring-Mass System

The next innovation in cloth modeling was the inclusion of spring mass coordinate systems as opposed to a purely geometric grid coordinate system, aslo referred to as a particle system [9]. Provot demonstrated that the addition of massive particles and springs assembled to account for physical forces such as tension, compression, shear, and bending, shown in arrangement in Fig. 1.2, were adequate in creating a cloth model that could be simulated using a simple Euler method [2]. An aspect of

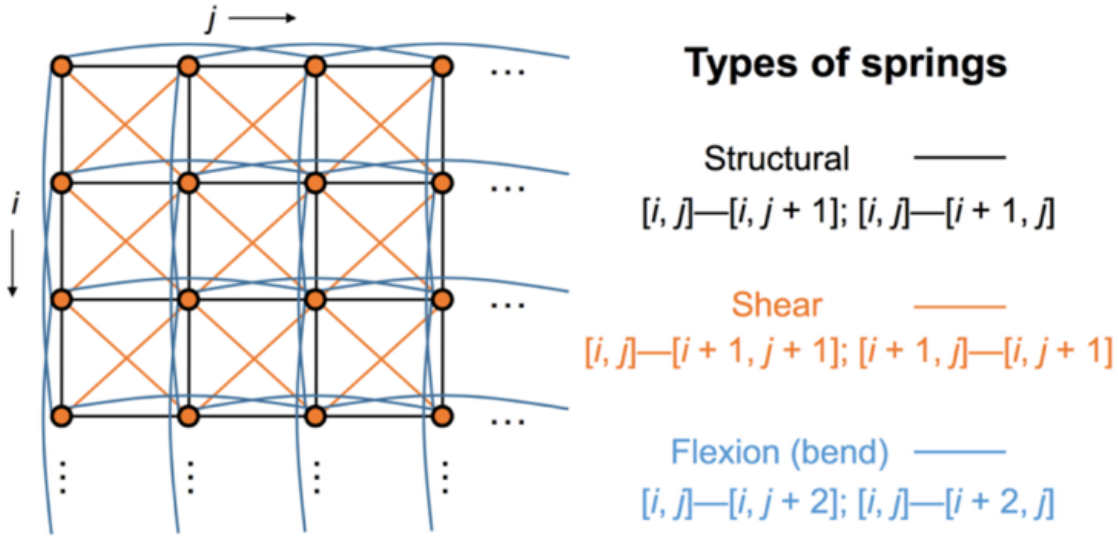


Figure 1.2: Provot's spring mass system deisgn [2].

the Provot design of spring masses is the unattended affect of creating a stiffer than usual cloth model, meaning, that the models generated using this method "act more like sheets of rubber" than actual cloth [2]. The consequence of this is exploited for the development of the air bladder model.

1.2.4 Triangle Based Representation

In the pursuit of more realistic visuals for cloth model simulations, the method of applying triangle based facets (Fig. 1.3) has been shown to be advantageous. This is the forfront of today's cloth model research. Baraff showed that the use of this form of discretization of the grid system is useful in reducing simulation computation time [3]. Choi added onto this work by constraining the compression of the springs and utilizing column based buckling instead which improved upon the computation time and the realistic "look" [24]. English and Bridson would expand upon the previous

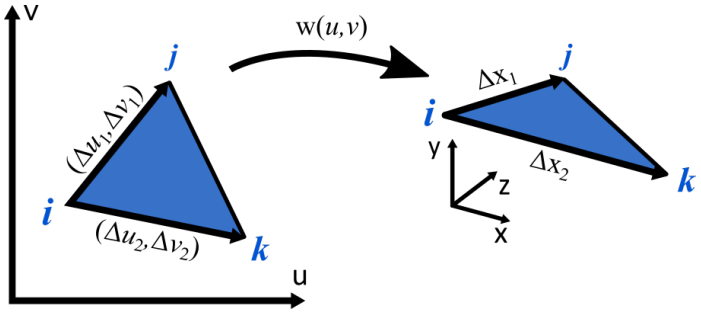


Figure 1.3: Triangle facets undergoing deformation and transformation [3].

concepts by allowing the triangle facets to be joined by midpoints where they can swivel for faster computation compared to column buckling (Fig. 1.4). These triangle meshes are then overlaid with a smoothing mesh at the desired resultion and texture for the application [4].

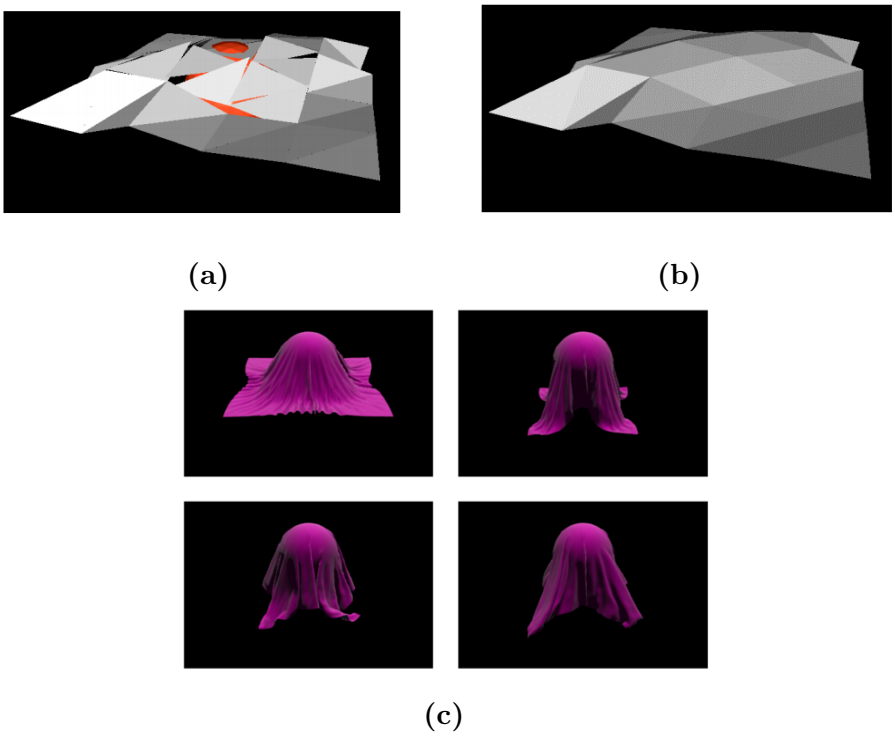


Figure 1.4: Pivoting triangle mesh cloth model and result [4].

1.2.5 Cloth Model for Material Extraction

Extracting material properties and then using them in modeling has depended on the use of finite element models and spring-mass models. In order to accurately represent the different materials in their respective models, the materials of interest need to be characterized through such techniques as matching a FEM to physical material samples as demonstrated by Schumacher [27]. Not only is demonstrating the explicit dynamics and interactions of importance but at times the accuracy of the models representing the cloth like bodies would require a demonstration of near real-world material properties. Various techniques have been shown to be able to extract such properties. Of these techniques, most if not all, utilize a form of advanced linear regression, machine learning, or neural network (deep or convolution) to assist in the "matching" of the cloth materials in question [28] [29] [30]. As the cloths

being investigated have their material properties extracted, they are stored in shared databases for future references. Another technique that was recently demonstrated for extracting material properties relied on a video source of the item in question. Under direct manipulation the item was recorded, and the properties extracted using various techniques [31]. Although they may be computationally expensive, there has been some advancement made in improving the real-time response of mass-spring methods used for this purpose [26].

1.3 Research Application

The primary motivation of this research is to improve upon a model of a soft body non-grasp manipulator. The soft bodied manipulator is a component of a hybrid soft and hard robotics system conceptualized for an application in the medical field. Pressure ulcers have long plagued the medical field requiring constant nurse intervention in attempts to prevent their formation in susceptible patients. The patients who are bedridden and highly immobile are the susceptible patients [32] [33] [34]. The nature of the intervention required by the nursing staff typically involves nurses to physically maneuver or manipulate an individual into varying positions regardless of the patient's body size or weight.

The core of the efforts in this research are to contribute to the development of a novel soft body / rigid body system that automatically manipulates the pressure forces acting on the patients. This long-term effort has not only intended to reduce the physical demands on the nursing staff but also to reduce the possibility of pressure ulcers development on patients [35]. In continuing the recent research of a robotic force bed developed by the Robotics, Biomechanics, Dynamic Simulations Lab (RBDSL) group at The University of Texas at Arlington in partnership with The University of Texas at Dallas and the Presbyterian Wound Care Clinic, Figure.

1.5, this kind of soft non-grasp manipulator can be further developed. Such a

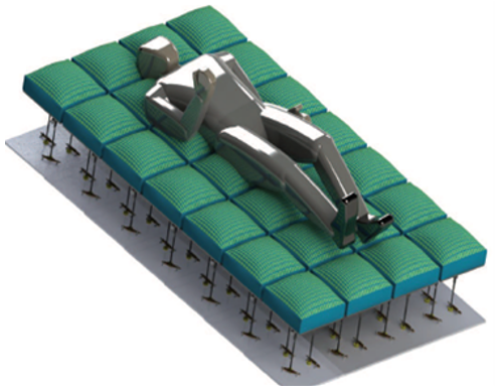


Figure 1.5: Force Bed Concept designed for non-invasive collection of biometrics and non-grasp manipulation of patients.

device would utilize a combination of soft and hard robotics techniques to accomplish the required object manipulation [36]. In their work they were able to couple rigid systems with soft robotics, real time feedback control for manipulation of objects and the control of applied pressure while also using artificial intelligence / machine learning to monitor and adjust the overall system [36]. To further this research, the development of an improved model was necessary and that is where my research adds with the use of the traditional techniques in cloth modeling, a more accurate representation of the soft body air bladder used in the force bed has been developed. The air bladder non-grasp soft manipulator is one component of the system that requires further work and is the focus of this thesis. As shown in Fig. 1.6 the purpose of the air bladder was the be able to manipulate objects on the top surface and or control the application of pressure on those objects. The project requires a model, simulation, and controls to achieve the stated purpose, in order to accomplish this the theoretical model, ultimately, needs to match the experimental prototype under the same conditions, that being the same internal pressure. The experimental prototype

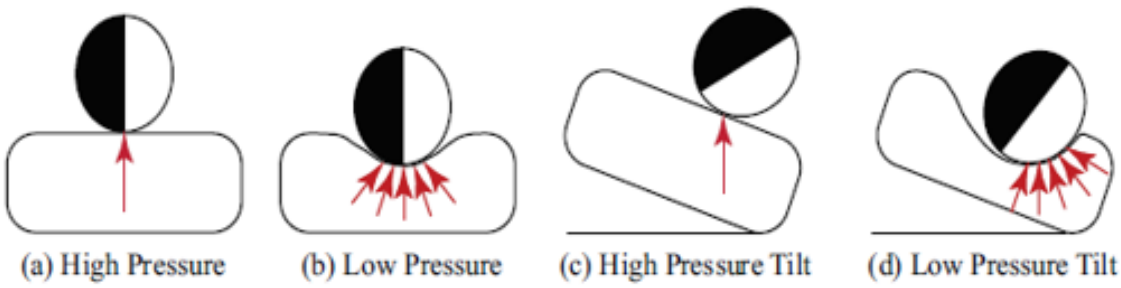


Figure 1.6: Object manipulation for control of applied pressure.

is not void of defects. The air bladder component of the prototype was fabricated in a manner that produced regions of higher rigidity and varying thickenss such as at the edges. The model should match even the prototype and account for such defects and inconsistencies. The work presented here demonstrates: the ability to apply a traditional spring-mass-damper cloth model to create a working air bladder model and simulation, a methodology for tracking and collecting positional information from an experimental prototype, and explores a means of system identification for extracting spring constants from the experimental measurements.

CHAPTER 2

DYNAMIC MODEL

The dynamic model developed for the soft body non-grasp manipulator utilizes the basic principles of a spring-mass-damper system and adapts them through a traditional cloth modeling technique, taking advantage of spring-node interconnections within the mesh topology. The dynamic model was implemented in MATLAB using readily available numerical integration functions. The model developed, also uses a simple bookkeeping technique to implement scalability and allows for independent control of each node, spring, damper, and mass. The model is not perfect as some discrepencies arose that were not accounted for in the theory presented.

2.1 Spring-Mass-Damper

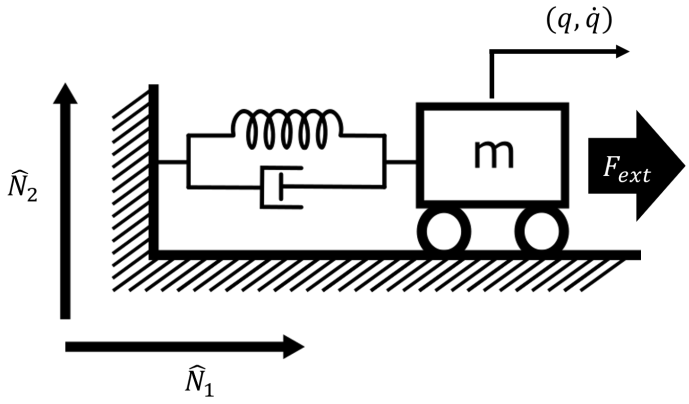


Figure 2.1: Basic spring-mass-damper model.

The basic principle of the Spring Mass Damper dynamic model is represented in Fig. 2.1. Where the spring force F_k is given by:

$$F_k = -k(q - L_0) \cdot \hat{N}_1 \tag{2.1}$$

and the damping force F_d is given by:

$$F_d = -c\dot{q} \cdot \hat{N}_1 \tag{2.2}$$

The displacement of the mass m along the $hatN_1$ direction is given as the coordinate q with the velocity of the mass represented by $\dot{q}$ and acceleration as $\ddot{q}$. The velocity and position are found through integrating the following equation:

$$\ddot{q} = m^{-1}(F_k + F_d) \cdot \hat{N}_1 \tag{2.3}$$

2.2 Applied Cloth Model

Cloth model simulation involves the use of simple springs interconnected between massive particles of a spring-mass-damper model. The force bed air bladder was simplified to a system of massive particles which upon the forces were imparted, these forces included the surrounding spring and damper forces, as well as the forces caused by gravity and the forces from the internal pressure of the air bladder. Each simple spring in the cloth model is orientated and connected such that the topology allows the cloth model to match the intended material properties of the desired physical cloth more closely. The basic structure used in the development of the air bladder model includes structural springs, shearing springs, and bending springs as showin in Fig. 2.2. The means of which they are connected to the massive particle nodes

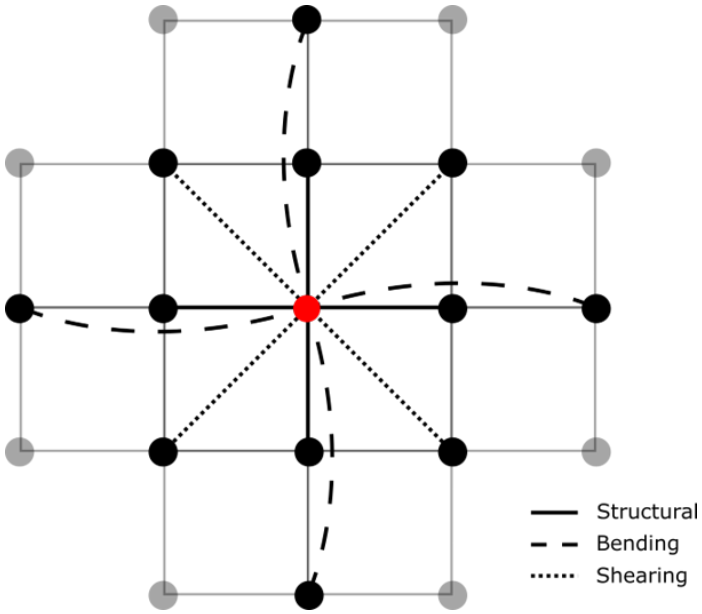


Figure 2.2: Spring-mass-damper cloth model mesh basic unit displayed on one point mass.

allows them to "act" like different springs in order to resist tension and compression, shearing, and bending within the mesh. The governing equation of motion (Equation. 2.4) remains similar to the basic spring-mass-damper model shown in Section 2.1. The difference being that it must account for each external force acting on every massive particle node such as gravity $\mathbf{F}_g = m * g\hat{N}_3$ and the estimated internal pressure $\mathbf{F}_P = P * Area$.

$$\ddot{\mathbf{q}} = A(q)^{-1}(\mathbf{F}_k + \mathbf{F}_d + \mathbf{F}_g + \mathbf{F}_P) \tag{2.4}$$

Where $\mathbf{F}_k$ encompasses all the spring varieties and $\mathbf{F}_d$ encompass all the damping forces corresponding to every spring. The area is calculated based on the courseness of the mesh and is found as taking the average area between a facet where the vertices are the point masses. The pressure is applied at the point masses normal to the mesh

surface. The vectors respresented in Equation 2.4 are tracked for each and every point mass in the cuboidal mesh.

2.3 The Dynamic Model

The model was built in MATLAB using a custom script to automatically scale the number of massive particles and springs used in the model. The mass matrix $A(q)$ was also calculated in this manner. The spring constant values as well as the damping coefficients of the dampers were assumed to be constant throughout the model. Once each component was created, the bookkeeping involved tracking the position and velocity of each particle. The equations of motion were determined from these values and used in a MATLAB ODE solver to model the full motion of each particle throughout the duration of the simulation.

Each massive particle is positioned in 3-Dimensional space relative to the inertial reference point via $\mathbf{P}_{Nn}$ where N is the intertial reference point with corresponding frame $N = \{\hat{N}_1, \hat{N}_2\}$ and n is any arbitrary massive particle.

$$\mathbf{P}_{Nn} = q_i\hat{N}_1 + q_{i+1}\hat{N}_2 + q_{i+2}\hat{N}_3 \tag{2.5}$$

The position vector $\mathbf{P}_{Nn}$ is comprised of coordinate distances q_i, q_{i+1}, q_{i+2} along their respective directions as shown in Equation 2.5. The derivative of the position vector $\mathbf{P}_{Nn}$ of any arbitrary massive particle is $\mathbf{V}_n$, the velocity of that particle relative to the inertial reference point as shown in Equation 2.6.

$$\mathbf{V}_n = \frac{d\mathbf{P}_{Nn}}{dt} = \frac{d}{dt}\left(q_i\hat{N}_1 + q_{i+1}\hat{N}_2 + q_{i+2}\hat{N}_3\right) = \dot{q}_i\hat{N}_1 + \dot{q}_{i+1}\hat{N}_2 + \dot{q}_{i+2}\hat{N}_3 \tag{2.6}$$

The spring forces $\mathbf{F}_{n,k}$ acting on any of the arbitrary massive particles n are determined by the following:

$$\begin{bmatrix} \mathbf{F}_{1,k} \cdot \hat{N}_1 \\ \mathbf{F}_{1,k} \cdot \hat{N}_2 \\ \mathbf{F}_{1,k} \cdot \hat{N}_3 \\ \vdots \\ \mathbf{F}_{n,k} \cdot \hat{N}_1 \\ \mathbf{F}_{n,k} \cdot \hat{N}_2 \\ \mathbf{F}_{n,k} \cdot \hat{N}_3 \end{bmatrix} = \begin{bmatrix} \boldsymbol{\delta}_{1,1} \cdot \hat{N}_1 & \boldsymbol{\delta}_{1,2} \cdot \hat{N}_1 & \dots & \boldsymbol{\delta}_{1,m} \cdot \hat{N}_1 \\ \boldsymbol{\delta}_{1,1} \cdot \hat{N}_2 & \boldsymbol{\delta}_{1,2} \cdot \hat{N}_2 & \dots & \boldsymbol{\delta}_{1,m} \cdot \hat{N}_2 \\ \boldsymbol{\delta}_{1,1} \cdot \hat{N}_3 & \boldsymbol{\delta}_{1,2} \cdot \hat{N}_3 & \dots & \boldsymbol{\delta}_{1,m} \cdot \hat{N}_3 \\ \vdots & \vdots & \dots & \vdots \\ \boldsymbol{\delta}_{n,1} \cdot \hat{N}_1 & \boldsymbol{\delta}_{n,2} \cdot \hat{N}_1 & \dots & \boldsymbol{\delta}_{n,m} \cdot \hat{N}_1 \\ \boldsymbol{\delta}_{n,1} \cdot \hat{N}_2 & \boldsymbol{\delta}_{n,2} \cdot \hat{N}_2 & \dots & \boldsymbol{\delta}_{n,m} \cdot \hat{N}_2 \\ \boldsymbol{\delta}_{n,1} \cdot \hat{N}_3 & \boldsymbol{\delta}_{n,2} \cdot \hat{N}_3 & \dots & \boldsymbol{\delta}_{n,m} \cdot \hat{N}_3 \end{bmatrix} \begin{bmatrix} k_1 \\ k_2 \\ k_3 \\ k_4 \\ k_5 \\ \vdots \\ k_m \end{bmatrix} \tag{2.7}$$

Where $\boldsymbol{\delta}_{n,m}$ represents the change in length of spring m whose unstretched length is $L_{m,0}$ and is acting on body n:

$$\boldsymbol{\delta}_{n,m} = (\|\mathbf{P}_{Ni} - \mathbf{P}_{Nn}\| - L_{m,0}) \cdot \hat{s}_m \tag{2.8}$$

with

$$\hat{s}_m = \frac{\mathbf{P}_{Ni} - \mathbf{P}_{Nn}}{\|\mathbf{P}_{Ni} - \mathbf{P}_{Nn}\|} \tag{2.9}$$

Similarly to determining the spring forces, the damping forces $\mathbf{F}_{n,d}$ acting on any of the arbitrary massive particles n:

$$\begin{bmatrix} \mathbf{F}_{1,d} \cdot \hat{N}_1 \\ \mathbf{F}_{1,d} \cdot \hat{N}_2 \\ \mathbf{F}_{1,d} \cdot \hat{N}_3 \\ \vdots \\ \mathbf{F}_{n,d} \cdot \hat{N}_1 \\ \mathbf{F}_{n,d} \cdot \hat{N}_2 \\ \mathbf{F}_{n,d} \cdot \hat{N}_3 \end{bmatrix} = \begin{bmatrix} \dot{\boldsymbol{\delta}}_{1,1} \cdot \hat{N}_1 & \dot{\boldsymbol{\delta}}_{1,2} \cdot \hat{N}_1 & \dots & \dot{\boldsymbol{\delta}}_{1,m} \cdot \hat{N}_1 \\ \dot{\boldsymbol{\delta}}_{1,1} \cdot \hat{N}_2 & \dot{\boldsymbol{\delta}}_{1,2} \cdot \hat{N}_2 & \dots & \dot{\boldsymbol{\delta}}_{1,m} \cdot \hat{N}_2 \\ \dot{\boldsymbol{\delta}}_{1,1} \cdot \hat{N}_3 & \dot{\boldsymbol{\delta}}_{1,2} \cdot \hat{N}_3 & \dots & \dot{\boldsymbol{\delta}}_{1,m} \cdot \hat{N}_3 \\ \vdots & \vdots & \dots & \vdots \\ \dot{\boldsymbol{\delta}}_{n,1} \cdot \hat{N}_1 & \dot{\boldsymbol{\delta}}_{n,2} \cdot \hat{N}_1 & \dots & \dot{\boldsymbol{\delta}}_{n,m} \cdot \hat{N}_1 \\ \dot{\boldsymbol{\delta}}_{n,1} \cdot \hat{N}_2 & \dot{\boldsymbol{\delta}}_{n,2} \cdot \hat{N}_2 & \dots & \dot{\boldsymbol{\delta}}_{n,m} \cdot \hat{N}_2 \\ \dot{\boldsymbol{\delta}}_{n,1} \cdot \hat{N}_3 & \dot{\boldsymbol{\delta}}_{n,2} \cdot \hat{N}_3 & \dots & \dot{\boldsymbol{\delta}}_{n,m} \cdot \hat{N}_3 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ \vdots \\ c_m \end{bmatrix} \tag{2.10}$$

Where $\dot{\boldsymbol{\delta}}_{n,m}$ represents the rate of the change in the length of spring m acting on body n and is determined by:

$$\dot{\boldsymbol{\delta}}_{n,m} = \|\mathbf{V}_i - \mathbf{V}_n\| \cdot \hat{s}_m \tag{2.11}$$

At every time step within the numerical solver, the positional changes of the massive particles cause the volume of the air bladder to change dynamically. The changing volume will affect the internal pressure of the air bladder. The desired pressure $P_{desired}$ is set at every time step. This was accounted for by utilizing MATLAB's convexHull function to take the convex hull of the entire cuboidal mesh to get a $V_{current}$ which was used with the given pressure at the corresponding time $P_{desired}$ to determine the P_{actual}.

$$P_{actual} = \frac{P_{desired} V_{initial}}{V_{current}} \tag{2.12}$$

The direction at which the pressure is applied to every massive particle was determined by using MATLAB's convexhulln function to convert the mesh into a trian-

gulation. The traingulation of the mesh is the input into the vertexNorm function which returns the normal unit vector to the surface at every mesh node.

2.3.1 Bookkeeping

The SMD cloth model had a scalable mesh grid that allowed for a control of the model's resolution, the coarseness of the mesh. This was achieved through a algorithm that allows for the relatively easy tracking of every spring-damper and the corresponding terminating massive particle nodes. Fig. 2.3 demonstrates the means of creating and numbering the massive particles with the structural springs shown for a small model example. Figure 2.3a shows that the mesh is constructed in a 2D plane which actually represents the 3D mesh as presented in Figure 2.3b. Each particle node has a coordinate position associated with it for each of the directions on the inertial reference frame. Spring maps were developed that automatically

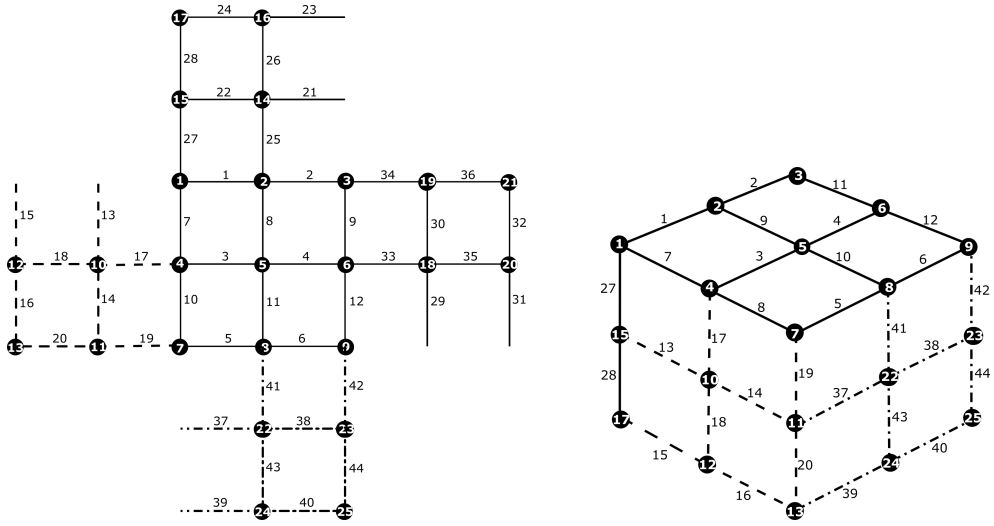


Figure 2.3: Mesh Development.

determined both terminating particle nodes for every spring. The spring maps were

also used in applying the dynamic model when it came to building the force vectors $\mathbf{F}_k$ and $\mathbf{F}_d$, they allowed for the correct application of force direction acting on each massive particle that was coupled to the end of the springs. As an example of the structural mesh spring map:

$$\mathbf{SpringMap}_{Structural} = \begin{bmatrix} 1 & 2 \\ 1 & 4 \\ 1 & 15 \\ 2 & 3 \\ 2 & 5 \\ 2 & 14 \\ 3 & 6 \\ 3 & 34 \\ \vdots & \vdots \\ 24 & 25 \end{bmatrix} \tag{2.13}$$

Every repeated value pair is removed as every spring is duplicated with opposing directions.

2.3.2 Numerical Solver

The model developed is a discrete model consisting of point masses and a series of interconnected springs and dampers. This is advantageous since the resulting mass matrix becomes a diagonal matrix allowing for the solution to be that of an Ordinary Differential Equation (ODE) solvable through a numerical integration. The model developed is inherently stiff, numerically speaking, limiting the options available between the different ODE solvers within MATLAB. The small time steps requried in

the numerical integration reduces the practicality of some solvers such as the well-known ode45 solver. The two ODE solvers used were ode23t and ode113. The ODE solver chosen for a majority of the efforts was ode23t as this proved to be relatively faster than the latter while still maintaining sufficient results. The implementation of the ODE solver relied on taking the initial coordinates of each particle in the model and using a series of custom functions to calculate initial spring lengths, velocities, the mapping between each spring, and separating the constrained particles from the unconstrained ones then passing all the pertinent information into the ODE solver. So long as the solver was producing results that fell within the tolerances specified, it was able to complete the simulation. As the simulation made progress, at every time step, the coordinate vector was saved into an external file for future analysis.

2.3.3 Energy Check

At every integration of the numerical solver the total energy and the kinetic energy are calculated and exported for future validation of the simulation. This is done to determine if the simulation is "energetically consistent" through the application of the Work-Energy Theorem as described by Bowling [37]. Equation 2.14 is the Work-Energy Theorem:

$$T_1 = T_2 - W_{1\to 2} \tag{2.14}$$

where $W_{1\to 2}$ is the work done in moving from state 1 to state 2 and where T_1 and T_2 are the kinetic energies at the two corresponding states.

$$T_{\mathbf{q}_1} = T_{\mathbf{q}_2} - W_{\mathbf{q}_{1\to 2}} - W_{P_{1\to 2}} \tag{2.15}$$

Since there is no work relating to explicitly time dependent constraints $W_{P_{1\to 2}}$, Equation 2.15 becomes:

$$T_{\mathbf{q}_1} = T_{\mathbf{q}_2} - W_{\mathbf{q}_{1\to 2}} \tag{2.16}$$

with

$$W_{\mathbf{q}_{1\to 2}} = \int_{t_1}^{t_2} \mathbf{\Gamma}_{\boldsymbol{q}} \cdot \dot{\boldsymbol{q}}\, dt \tag{2.17}$$

The work related to the generalized speeds $W_{q_{1\to 2}}$ is determined through the numerical integration of the ODE solver resulting in Equation 2.18

$$W_{\mathbf{q}_{1\to 2}} = \mathbf{\Gamma}_{\boldsymbol{q}} \cdot \dot{\boldsymbol{q}} \tag{2.18}$$

and leaving T_2 to be calculated at each time step (Equation 2.19) where i is the evaluated body and n is the total numbers of bodies

$$T_2 = \sum_{i=1}^{n} \frac{1}{2} m_i \|\mathbf{V}_i\|^2 \tag{2.19}$$

So long as the resulting value for T_1 is consistent at every time step and within the tolerances chosen in the numerical integrator throughout the simulation the energy check is valid. The energy check presented is not sufficient in and of itself to determine that the model is physically accurate, it simply provides confidence that the model is valid throughout the simulation. The model requires further analysis such as observing if the motions and deformations associated with the simulation are intuitive and identical to those observed by a physical analog of the model.

2.3.4 System Identification

The purpose of System identification is to extract material property information from the experimental measurements of the physical analog that the model represents. If the model is represented within the physical air bladder, the material existing where the springs are modeled can be closely approximated as a simple spring as well. The spring constant of the material, when considering the topology of the model, should be able to be extracted from the experimental measurements made of the physical air bladder while under different internal pressures. To accomplish this system identification, the positional information from the experimental data can be mapped to the model where a coordinate vector can be extrapolated.

$$\mathbf{Q} = [\mathbf{x}; \mathbf{y}; \mathbf{z}; \dot{\mathbf{x}}; \dot{\mathbf{y}}; \dot{\mathbf{z}}; \mathbf{energyCheck}] \tag{2.20}$$

The coordinate vector is the same as used in the simulation of the model developed. The calculations and methods used to determine each force acting on the massive particles of the model are then used on the coordinates extracted from the experimental data. With each $[\mathbf{x}; \mathbf{y}; \mathbf{z}]$ coordinate trouple vector representing the position of each tracked point in the experimental data. The sample of experimental data is taken at a moment when there is no motion of the experimental system. A sample taken at this moment would result in all points on the experimental analog being at equilibrium which allows for

$$0 = \mathbf{F}_k + \mathbf{F}_g + \mathbf{F}_p \tag{2.21}$$

There is no damping force due to the steady state of the sample being where each representative particle is at a velocity of zero. Since the purpose of the system iden-

tification is to determine the spring constants, the spring forces $\mathbf{F}_k$ are moved to the left-hand side of Equation. 2.21, providing

$$-\mathbf{F}_k = \mathbf{k}(\Delta) = \mathbf{F}_g + \mathbf{F}_p \tag{2.22}$$

To calculate for the spring constants of each spring within the mesh, the pseudo-inverse on the delta matrix is performed to move it to the right-hand side of Equation. 2.22.

$$\mathbf{k} = (\Delta)^+ (\mathbf{F}_g + \mathbf{F}_p) \tag{2.23}$$

where

$$\Delta = \begin{bmatrix} \boldsymbol{\delta}_{1,1} \cdot \hat{N}_1 & \boldsymbol{\delta}_{1,2} \cdot \hat{N}_1 & \dots & \boldsymbol{\delta}_{1,m} \cdot \hat{N}_1 \\ \boldsymbol{\delta}_{1,1} \cdot \hat{N}_2 & \boldsymbol{\delta}_{1,2} \cdot \hat{N}_2 & \dots & \boldsymbol{\delta}_{1,m} \cdot \hat{N}_2 \\ \boldsymbol{\delta}_{1,1} \cdot \hat{N}_3 & \boldsymbol{\delta}_{1,2} \cdot \hat{N}_3 & \dots & \boldsymbol{\delta}_{1,m} \cdot \hat{N}_3 \\ \vdots & \vdots & \dots & \vdots \\ \boldsymbol{\delta}_{n,1} \cdot \hat{N}_1 & \boldsymbol{\delta}_{n,2} \cdot \hat{N}_1 & \dots & \boldsymbol{\delta}_{n,m} \cdot \hat{N}_1 \\ \boldsymbol{\delta}_{n,1} \cdot \hat{N}_2 & \boldsymbol{\delta}_{n,2} \cdot \hat{N}_2 & \dots & \boldsymbol{\delta}_{n,m} \cdot \hat{N}_2 \\ \boldsymbol{\delta}_{n,1} \cdot \hat{N}_3 & \boldsymbol{\delta}_{n,2} \cdot \hat{N}_3 & \dots & \boldsymbol{\delta}_{n,m} \cdot \hat{N}_3 \end{bmatrix} \tag{2.24}$$

This should accomplish the calculation desired for extracting the material property information.

2.3.5 Deficiencies & Countermeasures

Base assumptions in the improved SMD model that utilizes cloth modeling techniques include the simplification of the thick-walled silicone air bladder material to a massive particle SMD model. This simplification removes any real world affects caused by the bending of the thickness of the material or any underlying stresses caused through the manufacturing techniques of the air bladder. The homogeneity of

the real-world material is also varying due to the methods of fabrication, leading to the possibility of varying thickness between different regions of the air bladder walls and at the edges. To counteract these deficiencies, the cloth model technique uses the bending springs and the MATLAB code is written for independent control of spring constants and damping coefficients for use with system identification of the material properties of the physical analog, the prototype air bladder.

2.3.6 Obtaining Simulation Data

The air bladder was simulated under various conditions that comprised of changing the air pressure over a given time. The coordinate vector was sampled when the air bladder had reached a steady state for different internal pressures. Timed properly, the overall kinetic energy of simulation at the sample time is minimized. This process was repeated to achieve measurements at various pressures and with various spring constants and damping coefficients to match the built physical model as coarsely.

CHAPTER 3

EXPERIMENTAL STUDY

To determine if the model developed and corresponding simulations are sufficient in representing the fabricated prototype of the force bed unit, the prototype must be measured under various conditions. The method presented for measuring the prototype relies on modern imaging techniques that are common to many robotics applications. An Intel RealSense D435 Stereoscopic camera is utilized in capturing accurate positional values of the top surface of the bed through the standard image processing applications available in MATLAB.

3.1 The Force Bed Unit

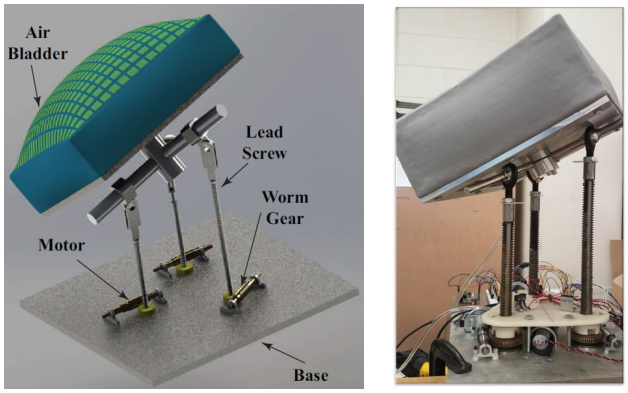
The force bed unit that was conceptualized and built by RBDSL and others in 2011 is primarily comprised of three linear actuators and an inflatable air bladder. It represents one unit out of 28 units that would comprise the entirety of the force bed conceptualized for patient manipulation, Fig. 1.5. The prototype consists of primary and secondary components of which the focus of the work presented is on the primary air bladder component. The telemetry, controls, and other miscellaneous hardware are not within the scope of the work and are considered as part of the secondary components.

3.1.1 Rigid Body Component

The three linear actuators are connected by a center plate with an articulating joint creating a parallel closed chain mechanism. The renderings used in the design

of the force bed unit and the eventual prototype are shown in Fig. 3.1. The linear actuators allow for large vertical displacement as well as angular displacement about the center of the articulating joint.

As currently designed, the linear actuators are lead screws driven by DC motor pow-



(a) Rendering [38] **(b)** Prototype [39]

Figure 3.1: Force bed unit rendering **(a)** and prototype **(b)**.

ered worm gears. Each actuator is topped with a swivel ball joint rod end that allows for 65° of rotational travel, the maximum articulation allowed of the center plate about either horizontal axis.

3.1.2 Soft Body Component

The other primary component of the force bed unit is the inflatable air bladder that is joined to the top of the parallel closed chain mechanism via the center plate on the articulating joint. The air bladder used in the prototype is shown in Fig. 3.2. The air bladder is comprised of an elastic "high-strength silicone-rubber" that was cast into a cuboidal shape [39]. The dimensions of the air bladder are: height, 0.127 [m], length, 0.3 [m], and width, 0.3 [m]. The bottom face of the air bladder is

comprised of the center plate of aluminum that rests ontop of the actuator assembly. The scope of this work focuses on the modeling and simulation of the air bladder.

Figure 3.2: Silicone based cuboidal air bladder.

3.1.3 Secondary Components

The secondary components that provide functionality to the force bed unit prototype include the external air compressor, pressure transducer, pneumatic solenoid, and microcontroller. The external air compressor provides the pressure needed for inflation of the air bladder. The pneumatic solenoid enables the controlled reduction of pressure in the air bladder. The pressure transducer and the microcontroller allow for feedback and control of the motion and inflation of the force bed unit. The scope of this work does not encompass the operation or setup of the secondary functional components. Improving upon the model and simulation of the air bladder to succesfully match the theoretical model to the experimental data collected from the prototype is the primary focus of this research.

3.2 Experimental Setup

The development of the improved model of the force bed air bladder relied on the mapping, analysis, and characterization of the prototype that was built. This was done by tracking the deformation of the prototype air bladder under various conditions, in this case, various pressures. Shown in Fig. 3.3 is the overall setup used

in collecting the positional data of the top surface of the air bladder to accomplish this goal. The air bladder was marked on the top surface as this was the primary

Figure 3.3: Experimental setup of data collection from prototype force bed unit.

region of interest. The top surface was marked with a grid pattern of 2 mm round white markings using a stencil and white ink that was compatible with the air bladder material. The spacing from marking to marking was arbitrarily set to roughly 9 ± 1mm and was organized into a rectangular grid pattern. Originally this setup

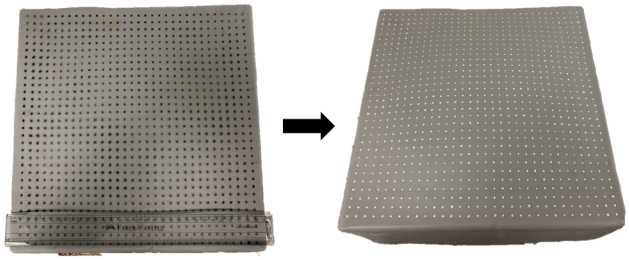


Figure 3.4: Modification of ink marking color to enhance contrast for more reliable positional data collection.

was manually marked with a black ink marking and manually measured (Fig. 3.4). This method was changed for improved accuracy of marking size and spacing while

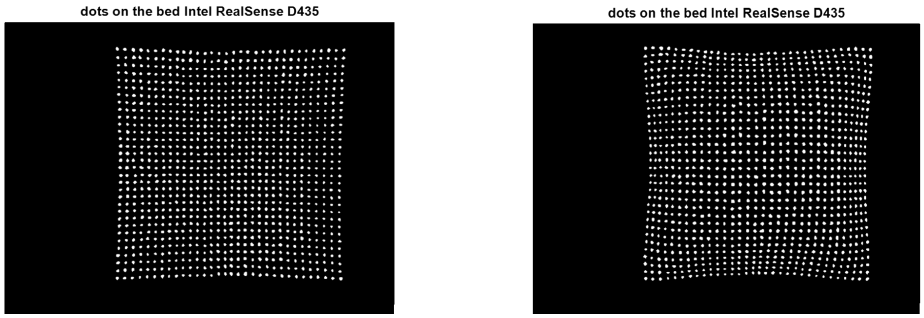
also improving the contrast between the ink marking and the bladder material. This improved contrast allowed for more reliable data collection using an automated image processing technique.

The system was observed using an Intel RealSense D435 Stereoscopic depth camera which, with the combination of a custom MATLAB script, allowed for each of the 1024 white ink markings on the top surface of the air bladder to be tracked in 3D space in relation to the camera. The camera was mounted to an aluminum structure directly overhead the air bladder as displayed in Fig. 3.3. The improved data collection technique allowed for large variations in the lighting of the room where the data collection had occurred, reducing lighting issues during the experiment. The air bladder was inflated to various pressures using a small air compressor under manual control. The pressure internal to the air bladder was monitored via a pressure transducer which fed back signal to a microcontroller and displayed on a connected PC to verify the current pressure within the air bladder. Once the air bladder had

Pressure [kPa]	101	105	107	110	112	117	130	132	145
Samples [N]	5	5	5	5	5	5	5	2	2

Table 3.1: Force Bed Prototype Sampling Conditions

settled and ceased motion due to the inflation and the readings from the pressure transducer had stabilized at a consistent pressure reading, a RGBD snapshot was taken using the RealSense camera. Two images are saved with every snapshot, a depth map and an RGB image. The two images together have the positional information saved for each ink marking as measured for future processing (Fig. 3.5).



(a) 101 kPa **(b)** 132 kPa

Figure 3.5: Processed Image Identifying pixel position from RGB source image.

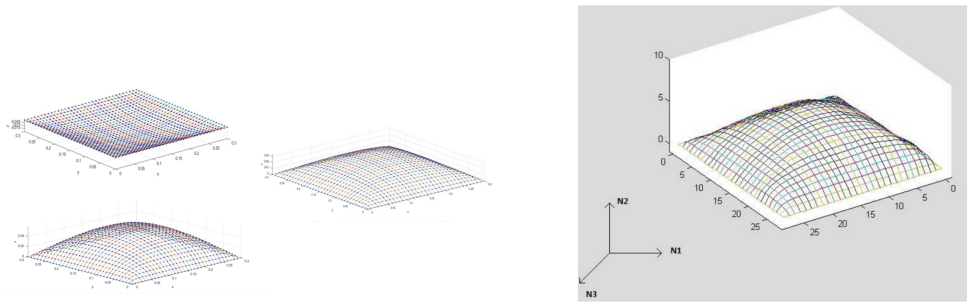
CHAPTER 4

RESULTS & DISCUSSION

Through the methods implemented and improved upon in the work presented, the deformations ovserved in the physical prototype of the air bladder appeared to be captured. Not only were the deformations of the model through its simulation intuitive, but the entire simulation was energetically consistent per the Work-Energy Theorem. In comparison to previous models, the captured deformation and the resulting structual integrity from the implementation of a cloth modeling technique proved successful. Despite these improvements, the system identification of the physical air bladder was not demonstrated as the results from the methods used to extract the spring constant from the experimental data were not conclusive. The matching between the theoretical model and the experimental data worked under certain internal pressures but not at others as there appears to be missing topology information or unaccounted variables.

4.1 Previous Model Deficiencies

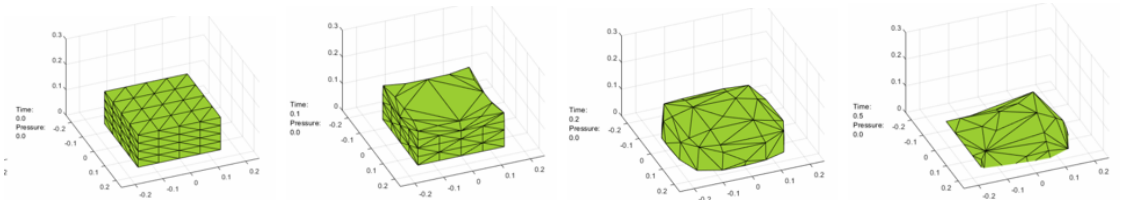
In initial attempts at modeling the air bladder as shown in Fig. 4.1, only the top surface was modeled using a simple spring mass damper matrix model comprised of only structural springs. This model assumed that each of the four edges of the top surface were fully constrained. This was shown to be insufficient as the data collected from the physical air bladder shows clear deformation at the edges as the air bladder "pillowing" becomes more apparent under increasing pressures (Fig. 3.5). From the positions measured of the top surface of the air bladder under atmospheric



(a) Kuicek Model [39] **(b)** Parihk Model [38]

Figure 4.1: Previous air Bladder Models.

pressure (101 kPa) there is a perceptible concave deformation at the edges of the top surface of the air bladder. This mode of deformation is lacking in the edge-constrained models previously developed. To accommodate the deformation observed in the



(a) 0.0 [s] **(b)** 0.1 [s] **(c)** 0.2 [s] **(d)** 0.5 [s]

Figure 4.2: Structural spring air bladder model under collapse.

physical measurements, the model was altered to a cuboidal structure where the four side faces were added to the model, the bottom face of the air bladder was omitted because this face was a rigid plate of aluminum in the physical model allowing this face to be ignored. When this approach was made, the model demonstrated inadequate structural integrity, leading to a folding and complete collapse of the SMD model under simulation as portrayed in Fig. 4.2. At this point, the application of cloth modeling techniques mentioned by Provot [2] were used. The shear and bending springs were added to the cuboidal mesh to achieve the structural integrity required.

4.2 Experimental Procedure Validation

Over a comparison of multiple samples taken of the air bladder at the various internal pressures specified in table 3.1, positional deviation between samples were calculated below $1x10^{-6}$m (Fig. 4.3). The repeatability of the image capturing setup and identification of the markings on the air bladder are shown to be very reliable. Deviations in position of less than 1e-6 m would be inconsequential in the comparison between the simulation and the prototype. Shown in Fig. 4.4 is the progression of deformation of the top surface of the prototype air bladder at a few of the pressures demonstrated.

4.3 Energetic Consistency in Simulation

Through the energy check that was calculated and tracked throughout each simulation Fig. 4.5, the simulation is validated as the energy of the system remained well within the tolerances of the ODE solver. The energy check uses the Work-Energy Theorem to determine the energetic consistency, a near constant value below the tolerances of the numerical solver demonstrates a physically consistent model. As stated previously, this is not a guarantee for sufficient modeling, instead it provides confidence in the model.

4.4 Captured Deformation

Comparing the improved SMD cloth model simulation results to the physical prototype that was fabricated, the simulation was successful in demonstrating intuitive deformations of the cuboidal air bladder as well as sufficient structural integrity to withstand its own weight in a manner observed in the prototype under similar conditions. The concave deformations observed at the edges of the prototype were

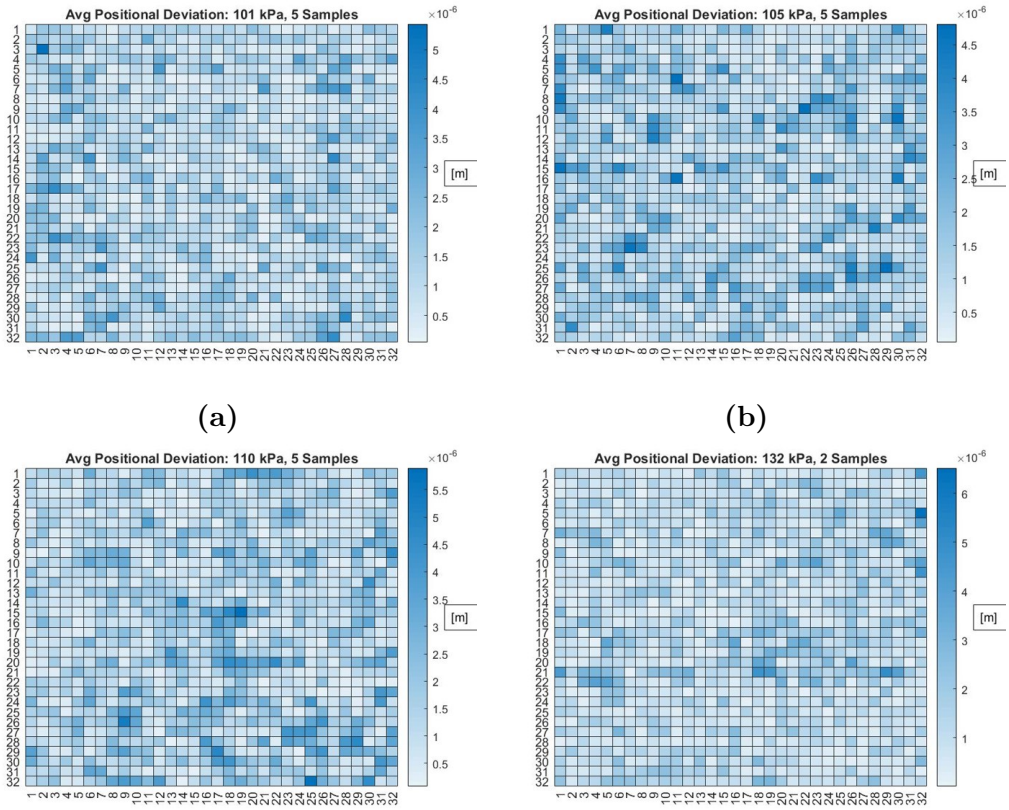


Figure 4.3: Heatmaps showing the average positional deviation between various samples taken of the air bladder top surface under corresponding absolute pressures: **(a)** 101 kPa, **(b)** 105 kPa, **(c)** 110 kPa, **(d)** 132 kPa.

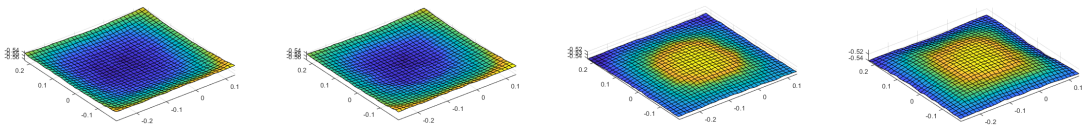


Figure 4.4: Progression of air bladder top surface deformation under corresponding absolute pressures **(a)**:101 kPa, **(b)**: 105 kPa, **(c)**: 110 kPa, **(d)**: 132 kPa.

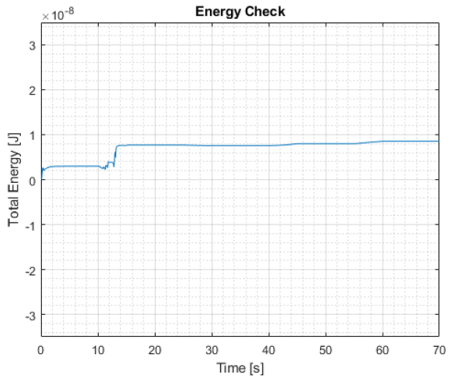


Figure 4.5: Simulation Energy Check

also observed in the simulated model as shown in Fig. 4.6. It is noted that the nature of the concave edge deformation is subtle while the air bladder's internal pressure is equal to atmospheric. When under increasing pressure, the concavity at the edges becomes more pronounced. With the addition of the shear and bending springs to the structural springs in the cuboidal mesh, the simulated model is able to support its own weight at the chosen spring constant values (K=1000 N/m). The simulation displayed in Fig. 4.7 shows that as time progresses, the model will move from the initiated state into a steady state. There exist slight vibrations in the simulation that are not shown in the images due to their scale. The model settles into position within 1 s with the chosen damping constant values (c=800 Ns/m). The model appears to match the look and feel of the physical system where the improved cuboidal mesh shown in Fig. 4.8 demonstrates the intuitive "pillowing" deformation observed in the prototype. From Fig. 4.9, and Fig. 4.10, it is observed that the simulated model, under similar conditions to the prototype, shows similar deformation on the side walls of the cuboidal mesh as the prototype's side walls.

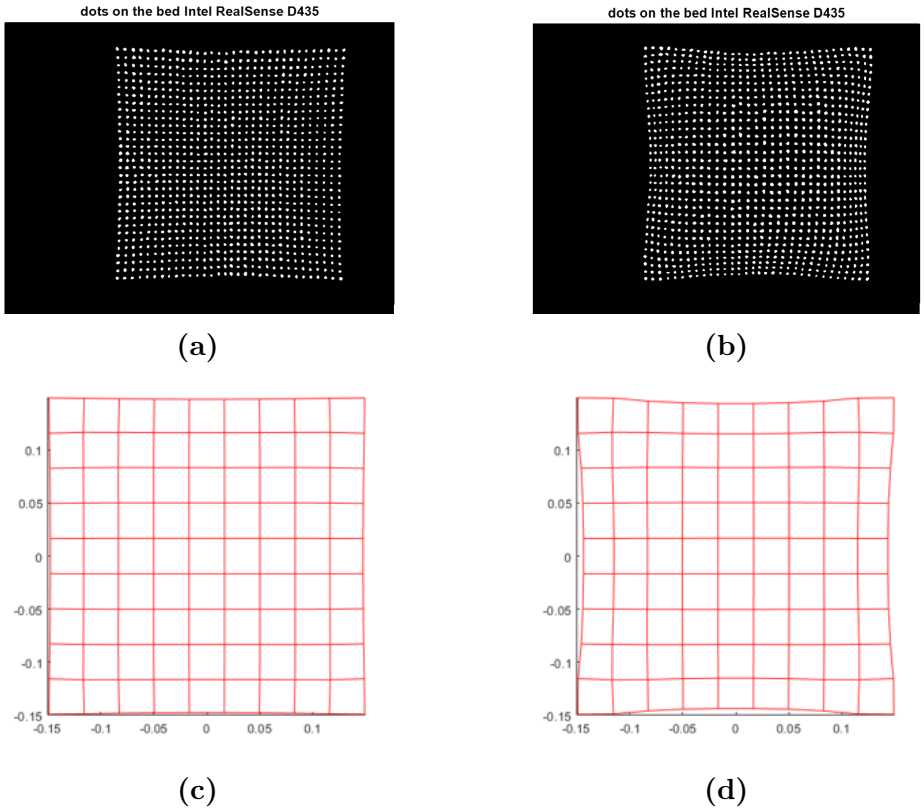
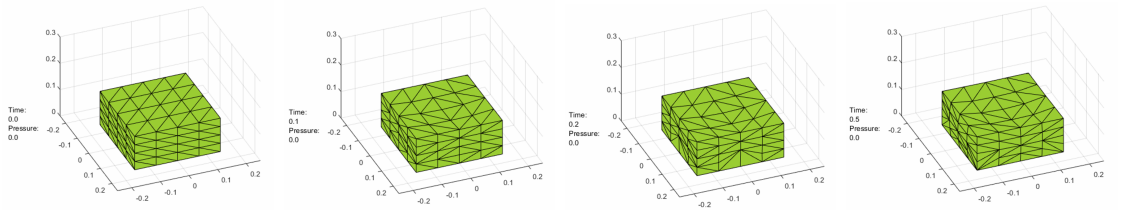


Figure 4.6: Concave Top Surface Air Bladder Deformation Observed in Prototype and Simulated Model.

4.5 System Identification

It was observed that the method used to calculate the spring constants from the experimental data was insufficient. The Δ matrix, as defined in Equation. 2.24, even in the coarsest form of the meshes modeled, maintains more springs than particles. As the resolution of model mesh is increased, creating a finer mesh, the rectangular issue is only exacerbated. The resulting Δ matrix becomes overconstrained and when inverted in Equation. 2.23. Using the pseudo-inverse produces results for the spring constants that are neither consistent nor realistic. Various techniques were tried including: the pseudo inverse, singular value decomposition, least square norms, and more. These methods are not discussed as the results only converged to the same



(a) 0.0 [s] **(b)** 0.1 [s] **(c)** 0.2 [s] **(d)** 0.5 [s]

Figure 4.7: Structurally sufficient air bladder model under.

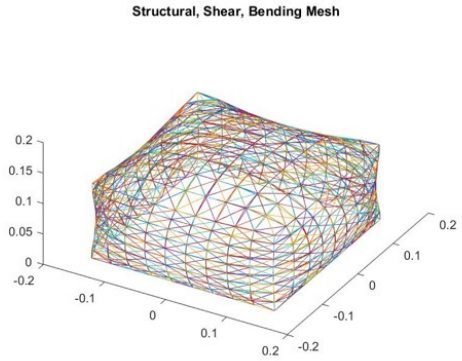
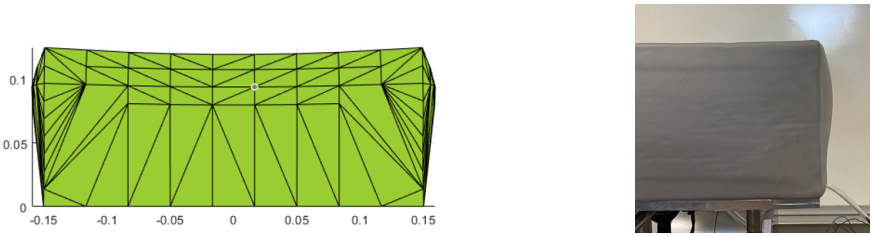
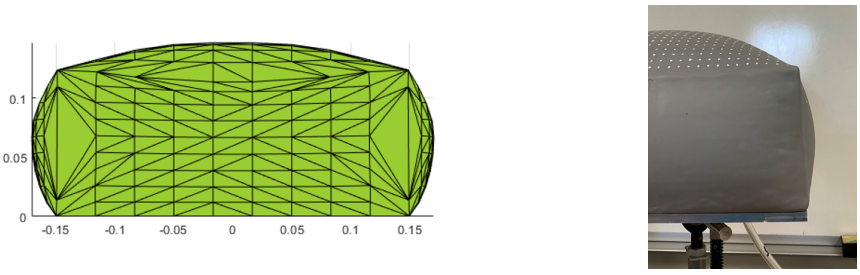


Figure 4.8: Complete Air Bladder Cloth Model Mesh



(a) 101 kPa absolute **(b)** 101 kPa absolute

Figure 4.9: Deflated Comparison



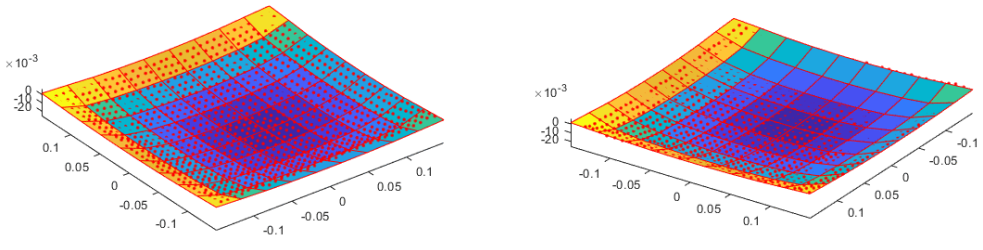
(a) 101.5 kPa absolute **(b)** 132 kPa absolute

Figure 4.10: Inflated Comparison.

values as the pseudo-inverse. The lack of a definite realistic answer appears to be an affect of the Δ matrix being ill conditioned, meaning, any small variations in the input leads to a wide range of solutions in the output that are insufficient for representing the spring constants needed for proper system identification. Not within the scope of this work, but shown to be successful by others, is the implementation of a neural network to perform the system identification. This could be a possible remedy for extracting the correct material properties from the experimental data, providing a proper solution to the issue of system identification.

4.6 Model Deficiencies

There was an unresolved deviation between the resulting deformation from an applied real-world air pressure and the simulated model under similar conditions. The model, while successfully matching the intuitive "look" of the prototype air bladder in simulation, would always seem to deform more than the prototype air bladder under similar conditions. From Fig. 4.11, the simulated model facets and the data collected from the prototype, when overlaid on each other, seem to reflect adequate matching under the internal pressure equaling atmospheric. The model can be adjusted as needed to also match the prototype under pressure as shown in Fig. 4.12. While visually similar, the model in Fig. 4.12 is held to a significantly lower pressure than the prototype was when the sample was taken. If portrayed under similar internal pressures, the model devolves and mismatches as shown in Fig. 4.13. The spring constant values were adjusted in an attempt to match a similar real world calculated spring constant of the material that was reported by another member of the RBDSL group. Despite this effort, the material properties between these two models did not coincide. There seemed to be a lack of rigidity in the model that has yet to be accounted for. The issues present in the system identification, once solved,



(a) Top View **(b)** Bottom View

Figure 4.11: Deflated Overlay of Data to Simulation.

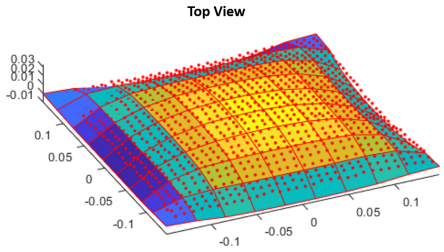
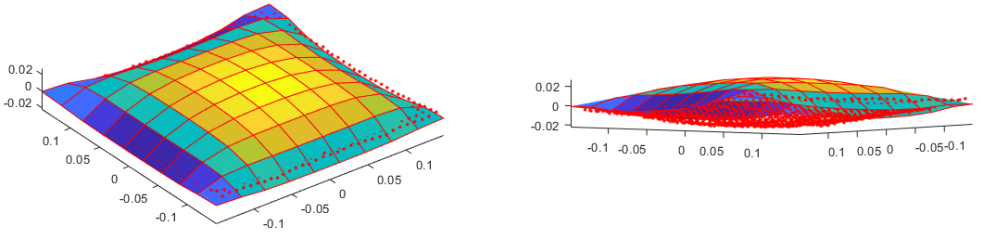


Figure 4.12: Matching Overlay Example



(a) Top View **(b)** Side View

Figure 4.13: Inflated Overlay of Data to Simulation.

could produce a model that succeeds in capturing the apparent deformations from the experimental prototype while under similar conditions, the same internal pressure.

4.7 Future Work

Despite the deficiencies in matching the results of the theoretical model with the experimental data, the model appears to remain a sufficient foundation for future work on the Force Bed project. If such work were to continue I would recommend exploring key areas further such as Topology Optimization, Learning Algorithms, Additional Hardware, and a method for including impact and object interaction in the model. There exists well known optimization techniques for mesh topologies which should be implemented to account for spring-node connections not present in the cloth model technique as well as controlling mesh resolution around regions of the air bladder that require more accurate representation such as the stiffer air bladder edges. Learning algorithms such as neural networks have been shown to be sufficient in extracting material properties for system identification. Similar methods could be applied to this study with the result showing proper matching between the theoretical model and the experimental measurements. With the addition of an Inertial Measurement Unit, roll and tilt of the air bladder prototype could be measured and fed back to the image processing system to automatically account for any needed rotation corrections. With the main purpose of the Force Bed project being patient handling, modeling object interactions through impact and contact seems the next likely step.

CHAPTER 5

CONCLUSION

5.1 Conclusion

In total, the work demonstrated in this thesis encompasses one aspect of the world of soft robotics, that being the development and use of a soft body model to simulate the actions of a real-world prototype. The method demonstrated for collecting position data of surface motion of the physical air bladder proved valid as the repeatability of the data showed a sufficient average positional deviation at every point. The theoretical model also demonstrated the ability to capture the observed deformations in the experimental measurements of the physical air bladder. The simulation of the model was also validated through the Work-Energy Theorem as the simulation showed energetic consistency throughout. These two facts imply that the cloth model of the non-grasp manipulator is sufficient for further use in the Force Bed project.

In later work, one can begin to tie in the use of an artificial intelligence, deep neural networks, for system identification to achieve matching deformations under similar conditions, the same internal pressure. The application of modeling impacts, contact, and coupling a real-time controller would encapsulate the remaining steps needed to implement the Force Bed Unit prototype. Through implementing these improvements, the project should accomplish all the tasks originally set out in 2011, that being, precise control and manipulation of objects with a soft body non-grasp robotic manipulator.

APPENDIX A

SIMULATION PARAMETERS

Parameter	Description	Value	Unit
$K_{structural}$	Spring Constant	1000	N/m
$K_{shear,bending}$	Spring Constant	0	N/m
DMP	Damping Constant	800	$N \cdot /m$
N	# of nodes along $\hat{N}_1$ and $\hat{N}_2$	5	Unitless
M	# of nodes along $\hat{N}_3$	5	Unitless
$xLength$	Length of air bladder	0.3	m
$yLength$	Width of air bladder	0.3	m
$zLength$	Height of air bladder	0.127	m
$setDynamicPressure$	Set dynamic/static pressure	0	Unitless
$dynPres\_Time\_1$	P1 active pressure time	10	s
$dynPres\_Time\_2$	Time to go from P1 to P2	15	s
$dynPres\_Time\_3$	P2 active pressure time	25	s
$dynPres\_Time\_4$	Time to go from P2 to P3	30	s
$dynPres\_Time\_5$	P3 active pressure time	40	s
$dynPres\_Time\_6$	Time to go from P3 to P4	45	s
$dynPres\_Time\_7$	P4 active pressure time	55	s
$dynPres\_Time\_8$	Time to go from P4 to P5	60	s
$TFINAL$	Simulation run time	10	s
$P1$	Pressure 1	0	Pa
$P2$	Pressure 2	125	Pa
$P3$	Pressure 3	250	Pa
$P4$	Pressure 4	375	Pa
$P5$	Pressure 5	500	Pa
$totalMass$	Mass of the air bladder	0.1785	kg
$grav$	Gravitational acceleration	9.80665	m/s^2
$TINITIAL$	Simulation start time	0.0	s
$INTEGSTP$	Integration step	0.1	s
$RELERR$	Relative error	$1x10^{-8}$	Unitless
$ABSERR$	Absolute error	$1x10^{-9}$	Unitless

Table A.1: Parameters set for collapsing air bladder simulation.

Parameter	Description	Value	Unit
$K_{structural}$	Spring Constant	1000	N/m
$K_{shear,bending}$	Spring Constant	1000	N/m
DMP	Damping Constant	800	$N \cdot /m$
N	# of nodes along $\hat{N}_1$ and $\hat{N}_2$	5	Unitless
M	# of nodes along $\hat{N}_3$	5	Unitless
$xLength$	Length of air bladder	0.3	m
$yLength$	Width of air bladder	0.3	m
$zLength$	Height of air bladder	0.127	m
$setDynamicPressure$	Set dynamic/static pressure	0	Unitless
$dynPres\_Time\_1$	P1 active pressure time	10	s
$dynPres\_Time\_2$	Time to go from P1 to P2	15	s
$dynPres\_Time\_3$	P2 active pressure time	25	s
$dynPres\_Time\_4$	Time to go from P2 to P3	30	s
$dynPres\_Time\_5$	P3 active pressure time	40	s
$dynPres\_Time\_6$	Time to go from P3 to P4	45	s
$dynPres\_Time\_7$	P4 active pressure time	55	s
$dynPres\_Time\_8$	Time to go from P4 to P5	60	s
$TFINAL$	Simulation run time	10	s
$P1$	Pressure 1	0	Pa
$P2$	Pressure 2	125	Pa
$P3$	Pressure 3	250	Pa
$P4$	Pressure 4	375	Pa
$P5$	Pressure 5	500	Pa
$totalMass$	Mass of the air bladder	0.1785	kg
$grav$	Gravitational acceleration	9.80665	m/s^2
$TINITIAL$	Simulation start time	0.0	s
$INTEGSTP$	Integration step	0.1	s
$RELERR$	Relative error	$1x10^{-8}$	Unitless
$ABSERR$	Absolute error	$1x10^{-9}$	Unitless

Table A.2: Parameters set for non-collapsing air bladder simulation.

Parameter	Description	Value	Unit
K	Spring Constant	1000	N/m
DMP	Damping Constant	800	$N \cdot /m$
N	# of nodes along $\hat{N}_1$ and $\hat{N}_2$	10	Unitless
M	# of nodes along $\hat{N}_3$	10	Unitless
$xLength$	Length of air bladder	0.3	m
$yLength$	Width of air bladder	0.3	m
$zLength$	Height of air bladder	0.127	m
$setDynamicPressure$	Set dynamic/static pressure	1	Unitless
$dynPres\_Time\_1$	P1 active pressure time	10	s
$dynPres\_Time\_2$	Time to go from P1 to P2	15	s
$dynPres\_Time\_3$	P2 active pressure time	25	s
$dynPres\_Time\_4$	Time to go from P2 to P3	30	s
$dynPres\_Time\_5$	P3 active pressure time	40	s
$dynPres\_Time\_6$	Time to go from P3 to P4	45	s
$dynPres\_Time\_7$	P4 active pressure time	55	s
$dynPres\_Time\_8$	Time to go from P4 to P5	60	s
$TFINAL$	Simulation run time	70	s
$P1$	Pressure 1	0	Pa
$P2$	Pressure 2	125	Pa
$P3$	Pressure 3	250	Pa
$P4$	Pressure 4	375	Pa
$P5$	Pressure 5	500	Pa
$totalMass$	Mass of the air bladder	0.1785	kg
$grav$	Gravitational acceleration	9.80665	m/s^2
$TINITIAL$	Simulation start time	0.0	s
$INTEGSTP$	Integration step	0.1	s
$RELERR$	Relative error	$1x10^{-8}$	Unitless
$ABSERR$	Absolute error	$1x10^{-9}$	Unitless

Table A.3: Parameters set for all other air bladder simulation examples.

APPENDIX B

EXPERIMENTAL DATA

The following data was collected from the imaging technique where the internal air bladder pressure was equal to atmospheric:

X	Y	Z	X	Y	Z	X	Y	Z
-0.067959	-0.012211	0.062295	0.128414	-0.179341	-0.178526	-0.563000	-0.553000	-0.550000
0.027145	-0.187613	-0.076637	0.203472	0.076541	-0.176608	-0.552000	-0.559000	-0.560000
0.062022	-0.140090	0.104244	-0.181847	0.088966	0.209278	-0.562000	-0.560000	-0.556000
-0.093093	-0.240874	0.131935	0.222514	-0.165733	-0.141810	-0.539000	-0.556000	-0.559000
-0.267943	0.024878	0.035816	0.128871	-0.142908	0.217809	-0.561000	-0.559000	-0.537000
0.064532	-0.013650	0.130160	-0.103487	-0.126704	-0.181587	-0.561000	-0.561000	-0.563000
-0.229904	0.131150	-0.141622	-0.164371	0.179438	-0.176270	-0.559000	-0.548000	-0.561000
-0.090628	-0.024211	-0.240112	-0.178105	0.079131	0.166193	-0.563000	-0.563000	-0.559000
-0.142964	-0.027919	0.023342	0.075799	0.143396	-0.014208	-0.551000	-0.541000	-0.551000
-0.156932	-0.067501	-0.217433	-0.126239	-0.114461	-0.126291	-0.556000	-0.560000	-0.558000
-0.205882	-0.178759	-0.182065	0.052080	-0.038210	0.202922	-0.559000	-0.560000	-0.562000
-0.079919	-0.065196	-0.039566	0.215730	0.127549	0.204218	-0.554000	-0.547000	-0.553000
-0.028491	-0.014984	-0.000058	-0.178708	-0.178262	-0.117261	-0.555000	-0.561000	-0.562000
0.013514	0.023764	0.052215	-0.180551	-0.181103	-0.154181	-0.562000	-0.562000	-0.561000
0.078597	0.026434	0.061576	0.141642	-0.087250	-0.176925	-0.559000	-0.544000	-0.543000
-0.218408	-0.163216	-0.168592	-0.050203	0.143722	0.168000	-0.547000	-0.550000	-0.547000
-0.168592	-0.091032	-0.028510	0.011040	0.157924	0.054087	-0.560000	-0.562000	-0.562000
-0.254103	-0.241703	-0.228394	-0.152635	0.105870	0.145084	-0.562000	-0.561000	-0.558000
-0.231113	-0.230454	-0.229619	0.170435	0.157783	0.092342	-0.546000	-0.547000	-0.552000
-0.228741	-0.229096	-0.216287	-0.114002	0.079429	-0.177529	-0.559000	-0.561000	-0.561000
-0.217737	-0.200563	-0.205130	-0.101749	-0.087514	0.157338	-0.562000	-0.541000	-0.550000
-0.206155	-0.205225	-0.205762	0.066708	0.079715	-0.125625	-0.551000	-0.561000	-0.561000
-0.205394	-0.191749	-0.193233	0.145087	0.157622	0.117782	-0.562000	-0.558000	-0.561000
-0.192539	-0.179141	-0.167128	-0.101823	-0.063251	0.065627	-0.561000	-0.545000	-0.555000
-0.168337	-0.168226	-0.166907	0.105218	-0.062052	-0.138233	-0.556000	-0.561000	-0.560000
-0.155513	-0.128998	-0.130467	-0.113326	0.040697	0.014208	-0.544000	-0.553000	-0.556000
-0.130418	-0.117174	-0.116966	0.065744	0.157162	-0.101068	-0.556000	-0.560000	-0.559000
-0.116133	-0.105253	-0.106200	0.078824	-0.088463	0.117096	-0.549000	-0.548000	-0.560000
-0.104506	-0.104880	-0.093168	0.000963	-0.150627	-0.012236	-0.559000	-0.560000	-0.559000
-0.078956	-0.079805	-0.079948	0.052523	0.077366	0.180144	-0.544000	-0.559000	-0.556000
-0.079663	-0.077227	-0.066252	-0.076491	0.051110	0.102579	-0.557000	-0.559000	-0.560000
-0.066608	-0.066608	-0.052364	-0.127148	-0.089617	-0.025433	-0.560000	-0.560000	-0.560000
-0.053126	-0.053411	-0.052174	0.204621	0.038048	0.051201	-0.558000	-0.553000	-0.550000
-0.041682	-0.041608	-0.041682	0.064536	0.167062	-0.000359	-0.547000	-0.546000	-0.557000
-0.041164	-0.040214	-0.040214	0.012838	-0.064909	-0.013580	-0.559000	-0.554000	-0.558000
-0.028286	-0.028387	-0.028135	0.154260	-0.090767	0.037980	-0.559000	-0.561000	-0.562000
-0.015032	-0.015167	-0.015140	0.077504	-0.052958	-0.026705	-0.545000	-0.549000	-0.552000
-0.001935	-0.001939	-0.002168	0.178722	-0.164520	0.142467	-0.556000	-0.551000	-0.546000
0.000672	0.011174	0.011134	0.168067	-0.152635	0.011199	-0.541000	-0.551000	-0.554000
0.012193	0.023374	0.023091	0.037845	-0.052958	0.166762	-0.555000	-0.556000	-0.555000
0.024365	0.024190	0.026125	-0.165792	-0.102565	0.063849	-0.546000	-0.553000	-0.556000
0.049849	0.062276	0.077032	-0.154338	-0.142383	0.039251	-0.556000	-0.556000	-0.556000
0.103893	0.128412	0.129689	0.156769	0.144341	0.093109	-0.553000	-0.544000	-0.550000
-0.265434	-0.242975	-0.129640	-0.050343	-0.073909	0.156387	-0.547000	-0.544000	-0.558000
-0.116442	0.077248	0.103638	-0.155244	-0.138041	-0.061079	-0.551000	-0.554000	-0.553000
-0.265562	-0.253066	-0.253066	0.171828	0.016187	-0.165996	-0.554000	-0.556000	-0.554000
-0.252700	-0.239764	-0.242328	-0.114937	-0.022640	0.197564	-0.551000	-0.549000	-0.543000
-0.241955	-0.227330	-0.218116	-0.024047	0.223355	0.222075	-0.542000	-0.551000	-0.549000
-0.200489	-0.174895	-0.153902	0.170732	0.041616	0.079887	-0.551000	-0.551000	-0.545000
-0.153966	-0.153966	-0.130690	-0.024891	-0.163033	0.041838	-0.548000	-0.549000	-0.546000
-0.116036	-0.116313	-0.101951	-0.160562	-0.123742	0.002024	-0.544000	-0.548000	-0.548000
-0.090300	-0.078742	-0.052255	0.129804	0.203755	-0.151153	-0.547000	-0.538000	-0.546000
-0.040415	-0.038174	-0.038662	-0.112687	0.115314	0.204227	-0.546000	-0.548000	-0.549000
-0.015393	-0.001064	-0.000876	-0.062418	-0.165733	-0.062307	-0.545000	-0.546000	-0.545000
0.013106	0.012252	0.025288	-0.077779	0.219515	-0.128693	-0.536000	-0.543000	-0.545000
0.025998	0.026309	0.035856	-0.026865	0.027002	-0.063505	-0.545000	-0.546000	-0.546000
0.037259	0.038436	0.048982	-0.000612	-0.127168	-0.062821	-0.545000	-0.546000	-0.543000
0.052090	0.063946	0.064178	-0.037032	-0.012315	-0.165733	-0.544000	-0.544000	-0.540000
0.064178	0.064887	0.078923	-0.127991	-0.166694	-0.102255	-0.542000	-0.543000	-0.543000
0.079037	0.090504	0.090670	-0.089551	-0.141626	-0.037778	-0.544000	-0.544000	-0.544000
0.091455	0.117097	0.129441	-0.037639	-0.049253	0.002214	-0.538000	-0.542000	-0.542000
-0.267152	-0.266065	-0.203475	-0.152416	0.090123	-0.049826	-0.540000	-0.544000	-0.538000
-0.169403	-0.142831	-0.142370	0.013805	-0.113719	0.191951	-0.540000	-0.544000	-0.545000
-0.104189	-0.089905	-0.055183	0.050919	-0.077945	0.115999	-0.544000	-0.546000	-0.547000
-0.028430	0.075072	0.114621	-0.052192	0.142942	-0.180926	-0.544000	-0.541000	-0.546000
0.115281	0.117041	0.129707	0.182618	0.166559	-0.036814	-0.546000	-0.546000	-0.548000
-0.014878	-0.267712	-0.255056	-0.179742	-0.166531	0.183122	-0.548000	-0.548000	-0.550000
-0.228680	-0.202700	-0.103383	0.180386	0.216630	-0.065875	-0.550000	-0.551000	-0.551000
-0.088033	-0.065995	-0.014390	-0.103155	-0.039628	-0.116864	-0.552000	-0.551000	-0.550000
0.024142	0.038122	0.039487	-0.179662	0.153669	0.220389	-0.548000	-0.551000	-0.551000
0.062627	0.077069	0.091834	-0.180990	0.221930	0.065949	-0.554000	-0.553000	-0.553000
0.117156	0.128399	0.130418	0.209106	0.079525	-0.153077	-0.557000	-0.557000	-0.547000
-0.264238	-0.254883	-0.255761	-0.140548	-0.036391	-0.088565	-0.556000	-0.554000	-0.555000
-0.253849	-0.243973	-0.226993	-0.178557	0.183745	0.117367	-0.556000	-0.547000	-0.556000
-0.228826	-0.229599	-0.205405	0.052693	-0.165793	0.051736	-0.547000	-0.552000	-0.557000
-0.189497	-0.168357	-0.114772	-0.175039	0.103221	-0.161952	-0.554000	-0.558000	-0.558000
-0.105557	-0.077901	-0.080757	0.040043	0.191955	-0.052411	-0.560000	-0.560000	-0.559000
-0.065913	-0.067166	-0.066638	0.153532	0.103165	0.180544	-0.557000	-0.557000	-0.558000
-0.055239	-0.041057	-0.028896	-0.039434	-0.065564	0.037111	-0.544000	-0.546000	-0.549000

-0.014590	-0.002907	0.012401	0.205452	-0.066200	-0.077110	-0.551000	-0.557000	-0.559000
0.024407	0.037356	0.063324	-0.090932	0.206358	0.220268	-0.548000	-0.547000	-0.557000
0.064944	0.091350	0.129343	-0.077806	0.195934	-0.168518	-0.559000	-0.560000	-0.558000
0.130646	0.130654	0.075318	-0.039228	0.025581	-0.075667	-0.557000	-0.561000	-0.542000
0.088280	-0.266630	-0.155459	-0.036515	-0.114088	-0.139270	-0.547000	-0.560000	-0.544000
-0.091298	-0.066220	-0.053078	0.000306	-0.116720	0.117670	-0.551000	-0.557000	-0.557000
0.075989	0.115535	-0.267417	-0.179892	-0.127505	-0.114497	-0.556000	-0.552000	-0.539000
-0.267926	-0.267433	-0.267926	-0.101911	0.222783	0.026895	-0.560000	-0.561000	-0.559000
-0.266615	-0.266090	-0.267035	0.184211	-0.011074	0.041099	-0.553000	-0.546000	-0.561000
-0.264490	-0.265290	-0.265805	0.093432	0.105366	0.117974	-0.548000	-0.555000	-0.558000
-0.265995	-0.265851	-0.265715	0.131456	0.157583	0.053975	-0.561000	-0.558000	-0.561000
-0.265553	-0.264647	-0.254877	0.209918	-0.126851	-0.114103	-0.561000	-0.556000	-0.553000
-0.255079	-0.254454	-0.254672	-0.050467	0.222495	-0.088947	-0.558000	-0.560000	-0.562000
-0.253475	-0.255199	-0.254173	0.054282	0.093150	-0.010711	-0.562000	-0.560000	-0.559000
-0.254571	-0.254613	-0.253685	0.002195	0.028240	0.040953	-0.556000	-0.547000	-0.561000
-0.254353	-0.253712	-0.254078	0.117354	0.131517	0.143895	-0.563000	-0.560000	-0.562000
-0.253876	-0.253712	-0.252328	0.065669	0.079325	0.105832	-0.561000	-0.556000	-0.546000
-0.252997	-0.253991	-0.242037	-0.153309	0.014397	-0.127052	-0.546000	-0.561000	-0.563000
-0.245195	-0.242148	-0.242989	-0.101865	0.170639	-0.138807	-0.558000	-0.547000	-0.556000
-0.242948	-0.241440	-0.243167	0.105744	0.223195	-0.062812	-0.563000	-0.561000	-0.561000
-0.240333	-0.241976	-0.242807	-0.010530	0.041400	0.131571	-0.561000	-0.548000	-0.551000
-0.241306	-0.241926	-0.242330	0.158115	0.184726	0.079904	-0.561000	-0.557000	-0.556000
-0.241627	-0.242394	-0.228643	0.222702	-0.139905	-0.087842	-0.555000	-0.562000	-0.561000
-0.229907	-0.230518	-0.231775	0.027458	-0.101834	-0.049859	-0.562000	-0.562000	-0.553000
-0.229427	-0.230289	-0.230904	0.065858	0.208960	-0.036454	-0.553000	-0.560000	-0.542000
-0.227048	-0.229845	-0.231514	-0.010607	0.000475	0.014395	-0.562000	-0.563000	-0.562000
-0.231147	-0.230696	-0.230680	0.053175	0.079193	-0.177210	-0.558000	-0.561000	-0.559000
-0.229379	-0.214496	-0.215665	-0.165449	-0.152495	-0.139865	-0.558000	-0.557000	-0.558000
-0.216062	-0.215997	-0.219571	-0.011053	0.155748	0.208696	-0.561000	-0.562000	-0.562000
-0.218305	-0.215278	-0.219125	0.053317	0.065772	0.195958	-0.563000	-0.560000	-0.557000
-0.218977	-0.215432	-0.217277	-0.036520	0.026636	0.040947	-0.559000	-0.554000	-0.560000
-0.218902	-0.217119	-0.217337	0.132243	0.144448	0.170493	-0.559000	-0.558000	-0.560000
-0.217427	-0.216792	-0.217935	0.092869	0.118953	-0.165709	-0.561000	-0.547000	-0.550000
-0.216683	-0.201664	-0.203910	-0.126936	0.001383	0.104667	-0.560000	-0.558000	-0.561000
-0.207350	-0.204912	-0.204357	0.170150	0.208770	-0.049930	-0.560000	-0.559000	-0.562000
-0.202356	-0.205569	-0.205502	-0.023799	0.014918	0.028095	-0.552000	-0.550000	-0.556000
-0.206563	-0.206933	-0.206308	0.041776	0.131577	0.145020	-0.556000	-0.559000	-0.559000
-0.206279	-0.205772	-0.205334	0.093678	0.195665	-0.152441	-0.559000	-0.547000	-0.553000
-0.190681	-0.189878	-0.194051	-0.063613	0.001103	0.066244	-0.547000	-0.543000	-0.556000
-0.194752	-0.193551	-0.189756	0.207759	-0.011113	0.182806	-0.557000	-0.556000	-0.560000
-0.193835	-0.190660	-0.188972	0.220795	0.027501	-0.050312	-0.560000	-0.556000	-0.553000
-0.192761	-0.180728	-0.181356	0.039552	0.182166	-0.178130	-0.546000	-0.556000	-0.549000
-0.179099	-0.175746	-0.182071	0.026465	0.143748	0.170075	-0.550000	-0.536000	-0.553000
-0.181015	-0.179679	-0.180905	0.001154	0.065822	0.104850	-0.554000	-0.548000	-0.550000
-0.181228	-0.180259	-0.166659	-0.126495	-0.075298	0.026771	-0.548000	-0.551000	-0.550000
-0.168592	-0.168822	-0.164308	0.207159	-0.088782	-0.036968	-0.550000	-0.541000	-0.547000
-0.167238	-0.167804	-0.169100	-0.024549	-0.012070	0.000703	-0.548000	-0.548000	-0.547000
-0.168405	-0.168182	-0.168706	0.078968	0.014673	0.118188	-0.548000	-0.547000	-0.547000
-0.167499	-0.167432	-0.154553	-0.163021	-0.150248	-0.112738	-0.547000	-0.548000	-0.546000
-0.154987	-0.155110	-0.155380	-0.125458	-0.076211	-0.050184	-0.544000	-0.544000	-0.541000
-0.155710	-0.156269	-0.154455	0.129984	-0.088464	-0.011564	-0.546000	-0.546000	-0.545000
-0.154812	-0.155920	-0.155964	0.001163	0.026876	0.065822	-0.546000	-0.545000	-0.535000
-0.156205	-0.156154	-0.155509	0.104043	0.193210	0.218876	-0.538000	-0.544000	-0.544000
-0.152535	-0.151126	-0.142143	-0.150248	-0.113053	-0.063523	-0.543000	-0.543000	-0.536000
-0.142469	-0.143479	-0.139318	0.219265	-0.125030	0.040040	-0.561000	-0.559000	-0.544000
-0.141497	-0.143338	-0.142507	0.129804	0.143122	0.155594	-0.544000	-0.553000	-0.552000
-0.143119	-0.141998	-0.140559	0.193513	-0.037100	0.026807	-0.548000	-0.553000	-0.550000
-0.142058	-0.142670	-0.143211	0.052719	0.066148	0.090897	-0.556000	-0.554000	-0.555000
-0.142906	-0.142957	-0.142226	0.117315	0.206150	-0.087416	-0.556000	-0.548000	-0.547000
-0.139352	-0.130930	-0.127895	-0.175561	-0.163421	-0.100338	-0.555000	-0.558000	-0.560000
-0.129061	-0.130036	-0.130783	0.051938	0.064826	0.104288	-0.547000	-0.560000	-0.540000
-0.131104	-0.130211	-0.129747	0.130217	0.192461	0.205538	-0.562000	-0.561000	-0.561000
-0.127426	-0.126729	-0.126265	0.218862	-0.010787	0.002425	-0.552000	-0.558000	-0.555000
-0.130874	-0.130434	-0.129644	0.079393	-0.037135	0.180271	-0.559000	-0.561000	-0.552000
-0.129722	-0.127476	-0.129121	0.014472	-0.113147	-0.100338	-0.557000	-0.559000	-0.557000
-0.116646	-0.118265	-0.115973	-0.137868	-0.125663	-0.088880	-0.554000	-0.550000	-0.550000
-0.116864	-0.116659	-0.117674	-0.036907	-0.011766	-0.062079	-0.546000	-0.553000	-0.551000
-0.117796	-0.117672	-0.117367	-0.050015	0.092216	0.193982	-0.552000	-0.553000	-0.552000
-0.117882	-0.114826	-0.117014	0.079887	0.104757	0.181299	-0.552000	-0.555000	-0.554000
-0.116536	-0.114950	-0.112831	0.218610	0.001253	-0.137897	-0.553000	-0.550000	-0.549000
-0.116107	-0.103613	-0.106107	0.026553	0.129363	-0.050274	-0.551000	-0.551000	-0.551000
-0.105136	-0.105172	-0.105735	0.039517	0.064617	0.091131	-0.550000	-0.550000	-0.549000
-0.106135	-0.105528	-0.104498	-0.063397	-0.026325	0.001160	-0.547000	-0.549000	-0.550000
-0.104909	-0.105058	-0.104834	0.014816	0.142333	0.167900	-0.551000	-0.549000	-0.546000
-0.104078	-0.103708	-0.102596	0.192602	0.219026	-0.087917	-0.547000	-0.547000	-0.547000
-0.101073	-0.092760	-0.091076	-0.164027	-0.150437	-0.075895	-0.547000	-0.548000	-0.549000
-0.091610	-0.091950	-0.092613	-0.050764	0.013598	0.026425	-0.544000	-0.545000	-0.545000
-0.092443	-0.091939	-0.092266	0.077573	-0.175524	-0.138349	-0.545000	-0.544000	-0.542000
-0.089136	-0.090822	-0.091998	-0.062130	-0.037049	-0.012004	-0.543000	-0.544000	-0.545000
-0.092642	-0.092827	-0.092660	0.053047	0.102723	0.204148	-0.544000	-0.547000	-0.549000
-0.092162	-0.088774	-0.091049	0.142246	0.154711	-0.112806	-0.544000	-0.556000	-0.559000
-0.090961	-0.079255	-0.080290	-0.076090	-0.050105	-0.174518	-0.559000	-0.546000	-0.561000
-0.080846	-0.076147	-0.079593	-0.100461	-0.063200	-0.024697	-0.545000	-0.560000	-0.557000
-0.079119	-0.079737	-0.079402	0.089785	0.154647	0.013268	-0.551000	-0.559000	-0.550000
-0.078181	-0.079133	-0.079305	0.064711	0.103251	0.117524	-0.544000	-0.544000	-0.551000
-0.079376	-0.079834	-0.079164	0.129146	0.141811	0.191463	-0.561000	-0.549000	-0.560000
-0.078881	-0.076958	-0.065599	0.216316	-0.164131	-0.151241	-0.560000	-0.561000	-0.559000

-0.065117	-0.065599	-0.067659	0.077385	0.116512	0.140680	-0.557000	-0.558000	-0.549000
-0.067859	-0.067697	-0.066923	0.167462	-0.177142	-0.127618	-0.541000	-0.553000	-0.550000
-0.063398	-0.065161	-0.065707	-0.090100	-0.038159	-0.025259	-0.547000	-0.545000	-0.544000
-0.065990	-0.066090	-0.066326	0.089670	-0.152803	-0.139183	-0.544000	-0.538000	-0.538000
-0.052875	-0.053190	-0.053572	-0.051346	0.026483	0.064500	-0.141895	-0.116583	-0.561000
-0.054699	-0.054661	-0.053823	0.179564	0.191531	-0.114433	-0.537000	-0.554000	-0.537000
-0.053262	-0.053661	-0.052752	-0.076104	-0.038418	0.013265	-0.538000	-0.546000	-0.557000
-0.053795	-0.054327	-0.054439	0.115764	0.166154	-0.178105	-0.539000	-0.534000	-0.560000
-0.053747	-0.051040	-0.052650	-0.102630	-0.064598	-0.013153	-0.544000	-0.540000	-0.540000
-0.053026	-0.052911	-0.052889	0.141652	-0.178213	-0.051759	-0.547000	-0.557000	-0.545000
-0.039434	-0.040798	-0.040810	-0.039215	0.025138	0.114380	-0.544000	-0.540000	-0.547000
-0.041693	-0.041397	-0.039853	-0.140158	-0.026081	0.077299	-0.533000	-0.544000	-0.553000
-0.041105	-0.041537	-0.041251	0.090375	0.128642	0.191232	-0.538000	-0.539000	-0.562000
-0.040987	-0.040511	-0.038394	-0.164395	-0.127187	-0.102630	-0.556000	-0.559000	-0.559000
-0.039306	-0.039570	-0.040171	-0.064481	-0.090542	0.141732	-0.561000	-0.548000	-0.550000
-0.028440	-0.028817	-0.027883	-0.027620	0.038125	0.179754	-0.548000	-0.560000	-0.560000
-0.028868	-0.028019	-0.026810	-0.139181	-0.114640	-0.103370	-0.547000	-0.544000	-0.559000
-0.027740	-0.027638	-0.028161	0.051343	0.062451	0.077162	-0.546000	-0.537000	-0.537000
-0.027957	-0.028243	-0.028243	0.090383	0.102099	0.116033	-0.551000	-0.537000	-0.536000
-0.028192	-0.027808	-0.026209	0.216248	-0.164737	-0.153351	-0.544000	-0.553000	-0.552000
-0.025666	-0.025952	-0.026516	-0.128022	0.203539	-0.027443	-0.540000	-0.557000	-0.557000
-0.025891	-0.015850	-0.015751	0.011608	0.025231	0.089945	-0.556000	-0.561000	-0.547000
-0.016207	-0.016209	-0.016178	0.101729	0.141798	-0.153129	-0.551000	-0.544000	-0.556000
-0.015314	-0.014018	-0.014490	-0.140215	-0.115595	-0.077799	-0.552000	-0.550000	-0.551000
-0.014908	-0.015563	-0.015071	-0.052411	-0.013127	-0.000616	-0.555000	-0.549000	-0.556000
-0.015787	-0.014878	-0.012845	-0.165403	-0.127399	0.191656	-0.538000	-0.551000	-0.553000
-0.013800	-0.013345	-0.002526	-0.001610	0.025451	0.050140	-0.554000	-0.559000	-0.558000
-0.002538	-0.002930	-0.002187	0.141798	-0.013412	0.102811	-0.549000	-0.557000	-0.555000
-0.002422	-0.002631	-0.000226	-0.153165	-0.102049	-0.089772	-0.558000	-0.553000	-0.557000
-0.001090	-0.001782	-0.002199	0.077248	0.115972	0.217345	-0.561000	-0.561000	-0.558000
-0.001440	-0.001069	0.010841	-0.102740	-0.091526	-0.077779	-0.547000	-0.552000	-0.563000
0.010900	0.010233	0.010306	-0.026961	0.128582	0.192147	-0.563000	-0.562000	-0.558000
0.010633	0.010443	0.011713	-0.140799	-0.014010	0.089467	-0.556000	-0.561000	-0.559000
0.010061	0.010542	0.010112	0.102591	0.116033	0.218876	-0.561000	-0.562000	-0.546000
0.010327	0.012679	0.023118	0.129247	-0.091487	-0.026369	-0.561000	-0.562000	-0.560000
0.023592	0.023161	0.023713	-0.013313	-0.001005	0.051690	-0.552000	-0.558000	-0.561000
0.023519	0.023519	0.023235	0.101917	0.115972	0.141641	-0.561000	-0.550000	-0.558000
0.023590	0.023183	0.024387	-0.103377	0.178857	0.190885	-0.561000	-0.548000	-0.563000
0.024393	0.024544	0.025500	0.205813	-0.013391	0.039236	-0.562000	-0.563000	-0.556000
0.035491	0.035722	0.036926	0.168149	-0.040381	-0.000835	-0.561000	-0.561000	-0.560000
0.036562	0.036343	0.036758	0.051782	0.065263	-0.153806	-0.562000	-0.557000	-0.557000
0.036538	0.038249	0.037666	-0.129358	-0.102811	0.141569	-0.562000	-0.561000	-0.559000
0.037439	0.037739	0.037399	0.180478	0.205977	-0.166652	-0.560000	-0.553000	-0.541000
0.038067	0.038824	0.049040	-0.090811	-0.064467	-0.051527	-0.557000	-0.555000	-0.544000
0.049293	0.049382	0.049362	0.116452	0.142341	-0.141257	-0.560000	-0.560000	-0.559000
0.050049	0.050521	0.049483	-0.076683	-0.026372	-0.013035	-0.555000	-0.541000	-0.550000
0.049918	0.049581	0.049278	0.025997	0.129798	0.168905	-0.557000	-0.541000	-0.543000
0.049510	0.049648	0.050652	-0.154480	-0.115973	0.193060	-0.543000	-0.543000	-0.544000
0.049991	0.051352	0.052321	0.220037	0.050905	-0.052774	-0.551000	-0.560000	-0.559000
0.061808	0.062317	0.062326	-0.025369	0.011683	-0.000227	-0.551000	-0.540000	-0.541000
0.061818	0.062336	0.062307	0.090726	0.117362	-0.077932	-0.547000	-0.547000	-0.549000
0.062605	0.063093	0.064192	0.180628	-0.154784	0.219768	-0.540000	-0.546000	-0.551000
0.064931	0.064564	0.074914	0.129506	-0.077699	0.103484	-0.547000	-0.553000	-0.544000
0.075119	0.075393	0.075201	0.142627	-0.142908	-0.090399	-0.544000	-0.556000	-0.561000
0.076698	0.075641	0.076008	0.168200	0.195024	0.207619	-0.561000	-0.561000	-0.544000
0.077277	0.077875	0.089061	0.155702	0.168593	0.181411	-0.564000	-0.544000	-0.551000
0.089061	0.089456	0.091232	-0.169241	0.129757	-0.104637	-0.561000	-0.560000	-0.548000
0.101898	0.102527	0.101952	-0.000932	0.011930	0.156205	-0.547000	-0.554000	-0.560000
0.101838	0.101495	0.102966	0.182180	0.194697	-0.169180	-0.546000	-0.559000	-0.540000
0.103469	0.104442	0.115390	-0.104714	-0.026557	0.013187	-0.558000	-0.556000	-0.544000
0.114849	0.114849	0.114498	0.064573	0.078717	0.091136	-0.550000	-0.558000	-0.559000
0.115059	0.115055	0.115133	0.104507	-0.090852	0.169227	-0.557000	-0.558000	-0.549000
0.115827	0.116267	0.116666	0.182792	-0.129693	-0.116836	-0.551000	-0.553000	-0.553000
0.116877	0.116839	0.117331	0.195741	-0.000122	0.077855	-0.540000	-0.547000	-0.539000
0.127304	0.128183	0.128183	0.090652	0.117843	-0.130315	-0.547000	-0.548000	-0.545000
0.128921	0.128882	0.128585	-0.052030	0.169370	-0.090565	-0.545000	-0.543000	-0.544000
0.128987	0.129384	0.130117	-0.065425	-0.142769	-0.151759	-0.547000	-0.545000	-0.561000
0.130167	-0.266750	-0.267249	-0.088911	0.196828	0.210557	-0.559000	-0.562000	-0.553000
-0.267740	-0.267446	-0.266732	-0.024132	0.014343	0.067229	-0.549000	-0.563000	-0.558000
-0.265505	-0.264033	-0.254517	-0.166040	-0.101680	-0.076380	-0.551000	-0.544000	-0.545000
-0.255172	-0.255172	-0.254301	0.197203	-0.023416	-0.075384	-0.532000	-0.541000	-0.556000
-0.254818	-0.243431	-0.241940	0.196845	-0.049769	-0.036831	-0.546000	-0.539000	-0.540000
-0.242809	-0.242418	-0.242859	0.001988	0.027008	0.066905	-0.550000	-0.554000	-0.545000
-0.243251	-0.242611	-0.242371	0.118645	0.144306	-0.115334	-0.559000	-0.555000	-0.557000
-0.242418	-0.229934	-0.230706	-0.075783	-0.062429	0.040187	-0.550000	-0.536000	-0.551000
-0.230289	-0.230239	-0.218116	-0.076058	-0.063049	0.002010	-0.543000	-0.535000	-0.540000
-0.218116	-0.218805	-0.218805	0.105524	-0.139986	-0.012395	-0.544000	-0.539000	-0.544000
-0.204168	-0.206438	-0.205478	-0.075802	-0.062392	-0.037232	-0.541000	-0.544000	-0.547000
-0.205849	-0.205824	-0.202881	0.196952	0.117993	-0.139986	-0.548000	-0.539000	-0.548000
-0.204912	-0.191300	-0.192168	-0.101892	-0.050402	-0.088834	-0.555000	-0.558000	-0.543000
-0.193994	-0.192445	-0.193438	-0.037366	-0.023843	0.040210	-0.559000	-0.540000	-0.561000
-0.193835	-0.192999	-0.194131	0.079745	0.091437	0.104213	-0.543000	-0.563000	-0.550000
-0.193290	-0.192599	-0.192599	0.118627	0.131966	-0.164401	-0.560000	-0.557000	-0.561000
-0.193438	-0.177866	-0.177718	-0.139317	-0.112961	-0.101315	-0.554000	-0.562000	-0.561000
-0.178695	-0.179737	-0.180806	-0.063505	-0.011787	0.194058	-0.561000	-0.546000	-0.556000
-0.181667	-0.177884	-0.177560	0.207858	-0.036903	-0.024706	-0.554000	-0.550000	-0.545000
-0.180384	-0.181032	-0.180958	0.013877	0.130654	0.156625	-0.542000	-0.541000	-0.541000

-0.180311	-0.179988	-0.180360	0.079463	-0.165285	-0.139986	-0.534000	-0.551000	-0.551000
-0.164065	-0.165565	-0.169101	0.040224	0.219723	-0.113790	-0.545000	-0.559000	-0.554000
-0.163391	-0.165479	-0.166885	0.130419	0.156625	0.169424	-0.548000	-0.561000	-0.548000
-0.166885	-0.166586	-0.165780	0.181190	-0.177254	0.051860	-0.546000	-0.538000	-0.544000
-0.152223	-0.155023	-0.156605	0.090844	0.116423	0.181519	-0.543000	-0.544000	-0.541000
-0.155649	-0.153979	-0.152030	0.206194	0.014301	-0.137664	-0.546000	-0.544000	-0.545000
-0.155286	-0.142663	-0.143194	-0.099623	-0.088163	-0.011173	-0.547000	-0.545000	-0.546000
-0.143954	-0.143833	-0.142641	-0.075856	0.002024	0.104709	-0.545000	-0.545000	-0.545000
-0.142577	-0.142175	-0.142130	0.168511	0.181100	-0.074543	-0.547000	-0.544000	-0.544000
-0.141363	-0.130827	-0.131296	-0.061636	0.027361	0.090606	-0.544000	-0.547000	-0.541000
-0.130457	-0.130868	-0.130403	0.116631	0.142503	0.168116	-0.547000	-0.548000	-0.549000
-0.129761	-0.129963	-0.116278	-0.151153	-0.024156	0.052345	-0.549000	-0.547000	-0.547000
-0.117028	-0.117217	-0.117229	0.129804	0.027547	0.117238	-0.547000	-0.548000	-0.547000
-0.117222	-0.117014	-0.115084	0.169424	0.205750	-0.149592	-0.547000	-0.547000	-0.548000
-0.112601	-0.103221	-0.105561	-0.037567	0.051249	0.155594	-0.549000	-0.543000	-0.551000
-0.105525	-0.104574	-0.102513	0.205782	-0.101541	0.038861	-0.544000	-0.547000	-0.549000
-0.092442	-0.092354	-0.091953	0.115733	0.128930	0.064762	-0.544000	-0.553000	-0.542000
-0.091953	-0.091421	-0.079106	-0.138538	-0.126335	0.026213	-0.549000	-0.551000	-0.551000
-0.079539	-0.080489	-0.079491	-0.089315	0.166808	-0.102016	-0.549000	-0.550000	-0.549000
-0.078461	-0.066572	-0.066998	0.013705	0.024673	0.037710	-0.552000	-0.544000	-0.545000
-0.066879	-0.066770	-0.066998	0.063057	0.178917	-0.114203	-0.550000	-0.553000	-0.547000
-0.066330	-0.065911	-0.066746	0.001101	-0.163590	0.037938	-0.551000	-0.554000	-0.544000
-0.051863	-0.053777	-0.054080	0.077509	0.090385	0.154735	-0.551000	-0.555000	-0.554000
-0.053777	-0.053381	-0.041804	0.103421	0.141188	0.154268	-0.553000	-0.553000	-0.553000
-0.041655	-0.041354	-0.040196	-0.090810	-0.077846	-0.012713	-0.538000	-0.542000	-0.544000
-0.040268	-0.040484	-0.028865	-0.052191	-0.001504	0.024555	-0.546000	-0.555000	-0.552000
-0.029072	-0.028709	-0.015382	-0.039922	0.153350	0.128258	-0.545000	-0.555000	-0.556000
-0.015835	-0.014944	-0.013914	0.179022	0.065254	0.128258	-0.547000	-0.552000	-0.556000
-0.002682	-0.003108	-0.001790	-0.078059	-0.039241	0.013477	-0.555000	-0.556000	-0.554000
-0.002195	-0.002199	-0.000857	-0.139986	-0.127815	0.038686	-0.551000	-0.557000	-0.556000
-0.000862	0.009647	0.010024	-0.051806	-0.001040	0.012559	-0.542000	-0.548000	-0.558000
0.009679	0.010555	0.009644	0.024770	0.050944	0.180102	-0.556000	-0.552000	-0.545000
0.010095	0.010808	0.012063	-0.128483	0.155036	0.153485	-0.556000	-0.557000	-0.558000
0.011296	0.024048	0.035394	0.077631	0.012537	0.103273	-0.559000	-0.558000	-0.558000
0.036342	0.036342	0.037733	-0.141074	-0.090787	0.191130	-0.556000	-0.559000	-0.548000
0.036859	0.036871	0.038734	0.219120	-0.039070	0.012071	-0.545000	-0.556000	-0.560000
0.049517	0.048919	0.049718	0.038629	0.051657	0.065254	-0.560000	-0.545000	-0.558000
0.049182	0.049580	0.049226	0.104026	0.154200	0.206886	-0.551000	-0.544000	-0.560000
0.050753	0.052310	0.061897	0.038841	-0.129750	-0.103719	-0.556000	-0.560000	-0.552000
0.063298	0.062536	0.064714	-0.180693	-0.000606	0.077587	-0.541000	-0.561000	-0.557000
0.074978	0.075604	0.075845	-0.104271	-0.052357	0.181166	-0.556000	-0.560000	-0.561000
0.075474	0.076489	0.088157	-0.000604	0.012335	-0.013150	-0.558000	-0.548000	-0.557000
0.087997	0.088413	0.090784	-0.154995	-0.026335	-0.013492	-0.561000	-0.546000	-0.554000
0.101274	0.101372	0.101372	0.050964	0.064612	0.090927	-0.558000	-0.560000	-0.560000
0.100798	0.101857	0.102085	-0.078221	-0.064550	-0.052485	-0.560000	-0.561000	-0.560000
0.102085	0.102085	0.102272	0.024550	0.142844	-0.116664	-0.559000	-0.542000	-0.546000
0.101898	0.102805	0.104891	0.209278	0.220637	0.052063	-0.551000	-0.550000	-0.557000
0.104770	0.114330	0.114743	-0.000599	0.039147	0.130413	-0.559000	-0.558000	-0.556000
0.114533	0.115614	0.116318	0.156507	-0.167770	-0.155401	-0.561000	-0.561000	-0.561000
0.117452	0.116858	0.128607	-0.026204	0.013069	0.025818	-0.561000	-0.561000	-0.548000
0.128399	0.128163	0.128163	0.104535	-0.154823	-0.025305	-0.544000	-0.546000	-0.551000
0.130294	-0.092095	0.023246	0.090368	-0.062023	0.170886	-0.560000	-0.545000	-0.550000
-0.266414	-0.266414	-0.230431	0.131144	-0.023979	-0.101223	-0.562000	-0.560000	-0.559000
-0.230077	-0.216858	-0.217485	-0.050040	-0.114461	-0.076471	-0.558000	-0.549000	-0.560000
-0.191723	-0.193814	-0.191814	0.170118	-0.088757	-0.075837	-0.562000	-0.562000	-0.562000
-0.179737	-0.180061	-0.165243	0.194058	-0.137346	-0.099834	-0.561000	-0.560000	-0.547000
-0.154683	-0.155603	-0.155158	0.155964	-0.025712	-0.150510	-0.557000	-0.541000	-0.545000
-0.142454	-0.130178	-0.117484	0.142978	-0.177166	-0.012515	-0.556000	-0.562000	-0.562000
-0.103126	-0.106005	-0.016076	0.064051	0.049466	-0.115810	-0.561000	-0.559000	-0.549000
-0.014697	-0.001560	-0.001551	0.154281	0.165132	-0.039426	-0.546000	-0.544000	-0.561000
-0.001593	0.009644	0.010267	0.062660	-0.116176	-0.052388	-0.561000	-0.562000	-0.561000
0.011451	0.037126	0.036582	0.116423	0.091110	0.024087	-0.552000	-0.562000	-0.553000
0.049554	0.062378	0.063163	-0.117048	0.167595	0.193416	-0.556000	-0.548000	-0.550000
0.063163	0.064046	0.075190	0.038787	-0.064723	-0.025983	-0.557000	-0.561000	-0.561000
0.075971	0.075398	0.075886	-0.012985	0.012380	0.025365	-0.560000	-0.560000	-0.561000
0.075080	0.075080	0.075488	0.051615	0.065206	0.090456	-0.550000	-0.561000	-0.562000
0.075372	0.075216	0.088777	-0.051756	-0.040012	0.038910	-0.553000	-0.544000	-0.563000
0.088252	0.088574	0.088574	0.051871	0.064833	0.077727	-0.548000	-0.562000	-0.561000
0.088574	0.088069	0.088413	0.090592	0.103317	-0.117073	-0.560000	-0.563000	-0.562000
0.088616	0.089217	0.088878	-0.078091	-0.065295	0.116701	-0.561000	-0.560000	-0.561000
0.089040	0.089222	0.088878	0.129838	-0.129739	0.077205	-0.563000	-0.562000	-0.559000
0.089991	0.101161	0.102085	-0.091338	-0.039203	0.038647	-0.557000	-0.550000	-0.545000
0.101422	0.101442	0.101607	0.103819	0.116923	-0.130508	-0.558000	-0.544000	-0.546000
0.101793	0.102805	0.102333	0.168636	-0.012797	0.025665	-0.559000	-0.561000	-0.562000
0.114533	0.114533	0.115602	-0.065242	0.208508	-0.012750	-0.561000	-0.561000	-0.540000
0.117070	0.128163	0.128399	0.052492	0.131456	0.144012	-0.549000	-0.560000	-0.560000
0.129257	0.129020	-0.254588	0.184261	0.183746	0.221751	-0.561000	-0.563000	-0.561000
-0.216900	-0.214971	-0.218657	0.014172	0.014510	0.052879	-0.546000	-0.558000	-0.558000
-0.193218	-0.192515	-0.178341	-0.151311	0.092118	-0.152684	-0.553000	-0.558000	-0.561000
-0.180785	-0.165151	-0.155360	-0.024985	0.155970	-0.125456	-0.538000	-0.555000	-0.559000
-0.129952	-0.128839	-0.105433	-0.077541	-0.125759	0.215713	-0.561000	-0.561000	-0.557000
-0.103491	-0.075743	-0.052382	0.216316	0.192918	0.089998	-0.561000	-0.561000	-0.561000
-0.027447	-0.002440	-0.001205	0.191183	0.076451	0.023955	-0.561000	-0.560000	-0.558000
0.010628	0.023344	0.023844	0.063391	0.090475	0.154496	-0.550000	-0.544000	-0.541000
0.036836	0.036930	0.049969	0.077491	0.180851	-0.166972	-0.545000	-0.562000	-0.561000
0.050455	0.052194	0.062530	0.102815	-0.025845	0.141907	-0.559000	-0.556000	-0.538000
0.088802	0.088810	0.089982	0.194758	0.207198	-0.077792	-0.550000	-0.557000	-0.560000

0.091096	0.115945	0.117351	-0.560000	-0.562000	-0.546000	-0.548000	-0.560000	-0.562000
0.130007	0.128717	-0.178492						

The following data was collected from the imaging technique where the internal air bladder pressure was 132 kPa absolute:

X	Y	Z	X	Y	Z	X	Y	Z
-0.092631	0.071602	0.035538	0.107695	-0.025772	-0.142875	-0.526000	-0.515000	-0.511000
-0.080139	-0.001524	-0.117526	-0.039530	0.091426	0.000004	-0.509000	-0.508000	-0.521000
0.072129	-0.236665	-0.093727	-0.010680	0.077501	-0.166719	-0.533000	-0.513000	-0.515000
0.036144	-0.136357	-0.094516	-0.077241	-0.065986	-0.175952	-0.534000	-0.513000	-0.538000
-0.238429	-0.066654	-0.264889	0.104429	0.211363	-0.061443	-0.509000	-0.535000	-0.533000
-0.262548	-0.263745	-0.249071	-0.179056	-0.113090	0.157955	-0.538000	-0.531000	-0.530000
-0.236134	-0.225403	-0.224608	-0.087683	0.105735	-0.127030	-0.528000	-0.532000	-0.523000
-0.208505	-0.198180	-0.196880	0.184881	0.119007	-0.114052	-0.521000	-0.519000	-0.520000
-0.197523	-0.197606	-0.198101	-0.049445	-0.011342	-0.140330	-0.527000	-0.534000	-0.514000
-0.157760	-0.147936	-0.129883	-0.178451	0.040782	-0.125587	-0.522000	-0.509000	-0.515000
-0.119299	-0.122576	-0.105355	0.015240	-0.074231	0.206157	-0.535000	-0.512000	-0.509000
-0.104738	-0.092764	-0.093128	-0.061754	0.041583	-0.011102	-0.509000	-0.511000	-0.514000
-0.093675	-0.091926	-0.084525	0.092234	0.118168	-0.062836	-0.511000	-0.531000	-0.513000
-0.080218	-0.079123	-0.080843	0.193034	-0.075149	-0.024130	-0.506000	-0.517000	-0.515000
-0.068416	-0.068814	-0.066160	0.131020	-0.088769	0.143459	-0.518000	-0.507000	-0.535000
-0.061982	-0.061004	-0.060662	-0.011058	0.203349	-0.175281	-0.535000	-0.532000	-0.525000
-0.058629	-0.053250	-0.053145	0.192390	-0.138659	-0.037445	-0.509000	-0.508000	-0.534000
-0.048328	-0.042119	-0.040000	-0.024226	-0.176209	-0.089894	-0.516000	-0.508000	-0.532000
-0.035632	-0.034122	-0.028105	-0.050516	0.191138	0.179764	-0.528000	-0.509000	-0.508000
-0.026855	-0.026802	-0.017296	0.078702	0.000870	0.013988	-0.507000	-0.516000	-0.509000
-0.014934	-0.005470	-0.001762	0.129366	-0.000326	-0.127552	-0.520000	-0.509000	-0.531000
0.004407	0.010252	0.028677	0.039188	-0.163977	-0.062836	-0.511000	-0.532000	-0.525000
0.031528	0.034763	0.035704	-0.178358	0.168863	-0.103909	-0.518000	-0.514000	-0.514000
0.035704	0.035634	0.044941	-0.025721	0.039573	0.052771	-0.513000	-0.523000	-0.518000
0.046949	0.048074	0.056443	-0.140591	-0.102690	-0.076952	-0.517000	-0.527000	-0.518000
0.060166	0.084407	0.084568	-0.154064	-0.090233	0.040361	-0.523000	-0.524000	-0.528000
0.096393	0.118524	0.133285	0.066325	0.105336	-0.050495	-0.533000	-0.533000	-0.523000
-0.159386	-0.110533	-0.029636	-0.154294	-0.126974	-0.161133	-0.531000	-0.516000	-0.536000
-0.250499	-0.235227	-0.196915	-0.102218	0.172156	0.094175	-0.530000	-0.521000	-0.519000
-0.169299	-0.156787	-0.156470	-0.086364	0.120301	-0.100490	-0.518000	-0.521000	-0.515000
-0.130438	-0.128918	-0.129678	-0.113328	-0.074466	0.029376	-0.509000	-0.512000	-0.511000
-0.116202	-0.116202	-0.106620	0.093385	-0.048644	0.042715	-0.511000	-0.517000	-0.508000
-0.053973	-0.053174	-0.045715	-0.099080	-0.051597	-0.064774	-0.510000	-0.530000	-0.518000
-0.043461	-0.064627	-0.019459	0.181450	-0.112675	0.357018	-0.841000	-0.528000	-0.514000
-0.015315	-0.002174	0.034526	-0.150871	0.119142	-0.064923	-0.512000	-0.518000	-0.532000
0.042148	0.044054	0.048023	-0.115113	-0.178951	-0.165157	-0.530000	-0.519000	-0.517000
0.049054	0.132268	0.133027	-0.115335	-0.089350	0.223838	-0.539000	-0.542000	-0.535000
0.133650	-0.239018	-0.195833	-0.050320	-0.182623	-0.114558	-0.530000	-0.520000	-0.516000
-0.170688	-0.130476	-0.120153	0.053314	-0.049635	0.078623	-0.510000	-0.512000	-0.509000
-0.016656	-0.011505	-0.008736	-0.060809	-0.038513	0.217157	-0.538000	-0.523000	-0.514000
0.035469	0.060010	0.082497	0.166750	-0.052662	-0.075867	-0.517000	-0.529000	-0.529000
0.095719	0.120561	-0.147323	0.170634	0.157412	-0.168392	-0.532000	-0.519000	-0.508000
-0.103364	-0.072985	-0.066314	0.143498	0.001704	0.203107	-0.534000	-0.509000	-0.516000
-0.056671	-0.053269	-0.046887	0.066725	0.130509	-0.012241	-0.508000	-0.531000	-0.525000
-0.032772	-0.008164	0.023115	-0.162793	0.167564	0.191607	-0.531000	-0.512000	-0.518000
0.033308	0.119579	-0.251274	0.000800	0.117632	-0.181314	-0.534000	-0.534000	-0.534000
-0.237002	-0.223622	-0.184626	-0.179205	-0.178966	-0.179039	-0.533000	-0.534000	-0.523000
-0.172015	-0.065905	-0.030776	-0.178094	-0.126894	-0.023984	-0.507000	-0.517000	-0.512000
-0.003646	0.093872	-0.236179	0.130551	0.102791	-0.181653	-0.532000	-0.530000	-0.535000
-0.186130	-0.185074	-0.159075	0.079462	0.208586	-0.010804	-0.519000	-0.516000	-0.511000
-0.144956	-0.131470	-0.091895	0.091783	0.027829	0.067276	-0.511000	-0.507000	-0.525000
-0.058962	-0.019821	-0.264735	0.040090	0.168143	-0.140036	-0.525000	-0.538000	-0.541000
-0.266262	-0.265463	-0.260870	0.196845	0.224170	0.184317	-0.539000	-0.534000	-0.536000
-0.260586	-0.260776	-0.259376	-0.074701	0.079466	0.002566	-0.539000	-0.537000	-0.538000
-0.259859	-0.251442	-0.250668	-0.010263	0.015030	-0.152957	-0.532000	-0.532000	-0.538000
-0.252387	-0.247843	-0.250859	-0.140341	0.210453	0.079022	-0.533000	-0.535000	-0.533000
-0.247462	-0.247414	-0.248342	0.131992	-0.035904	-0.010186	-0.533000	-0.535000	-0.531000
-0.238169	-0.238948	-0.225693	0.002361	0.158058	0.224970	-0.540000	-0.532000	-0.526000
-0.222315	-0.222315	-0.210801	-0.166568	0.014272	0.118211	-0.526000	-0.522000	-0.523000
-0.211412	-0.211748	-0.210739	-0.087758	-0.023576	-0.153558	-0.530000	-0.528000	-0.525000
-0.210319	-0.211558	-0.208131	-0.140934	-0.125718	-0.165680	-0.531000	-0.523000	-0.533000
-0.210830	-0.197540	-0.197161	0.041617	-0.177748	0.015362	-0.521000	-0.520000	-0.521000
-0.196915	-0.197775	-0.197894	0.042244	0.105432	-0.166238	-0.532000	-0.528000	-0.535000
-0.198643	-0.184041	-0.184156	0.171253	0.209973	-0.048799	-0.517000	-0.519000	-0.519000
-0.183950	-0.183532	-0.183579	0.092591	0.105650	-0.101487	-0.519000	-0.519000	-0.518000
-0.182738	-0.183179	-0.184640	-0.088681	0.041496	0.066870	-0.518000	-0.522000	-0.532000
-0.185114	-0.184928	-0.184359	0.145348	-0.166102	-0.153509	-0.530000	-0.529000	-0.536000
-0.184549	-0.171790	-0.171323	0.184472	0.219857	0.144549	-0.521000	-0.519000	-0.521000
-0.170758	-0.158295	-0.158610	-0.101487	-0.114134	0.052937	-0.514000	-0.516000	-0.514000
-0.157995	-0.156780	-0.156990	-0.074611	-0.050139	0.015423	-0.513000	-0.514000	-0.514000
-0.157581	-0.159773	-0.157589	-0.036471	0.079654	-0.165168	-0.532000	-0.518000	-0.535000
-0.159828	-0.145257	-0.144413	0.119268	-0.176317	-0.076444	-0.516000	-0.513000	-0.512000

-0.144794	-0.145257	-0.148917	-0.036176	-0.010669	0.117772	-0.516000	-0.529000	-0.515000
-0.145389	-0.144617	-0.144335	-0.149912	-0.061727	-0.049675	-0.513000	-0.512000	-0.512000
-0.143719	-0.143719	-0.144280	0.067253	0.002260	0.079344	-0.512000	-0.514000	-0.534000
-0.148443	-0.133324	-0.130983	0.105046	-0.176086	-0.087595	-0.516000	-0.510000	-0.516000
-0.132488	-0.130574	-0.130869	0.041432	-0.099248	-0.024190	-0.510000	-0.509000	-0.515000
-0.131295	-0.133161	-0.132837	-0.010943	0.118765	0.144103	-0.520000	-0.522000	-0.532000
-0.135382	-0.130517	-0.131744	0.156607	0.194649	-0.036898	-0.512000	-0.517000	-0.534000
-0.124585	-0.120569	-0.117471	0.131199	-0.173941	-0.124974	-0.521000	-0.510000	-0.517000
-0.119629	-0.117377	-0.117467	0.080119	-0.100501	-0.036187	-0.510000	-0.508000	-0.509000
-0.117778	-0.117500	-0.117750	-0.025262	0.002832	0.028533	-0.508000	-0.510000	-0.511000
-0.118160	-0.118241	-0.116181	0.053073	0.065948	0.091797	-0.511000	-0.508000	-0.522000
-0.109087	-0.106826	-0.107336	0.015200	-0.125452	0.131633	-0.518000	-0.519000	-0.516000
-0.106059	-0.105681	-0.104350	-0.113207	-0.088573	0.092894	-0.511000	-0.509000	-0.508000
-0.103923	-0.104129	-0.104555	-0.023782	0.015599	0.028517	-0.508000	-0.510000	-0.513000
-0.104946	-0.106831	-0.096734	0.080549	0.106262	0.169534	-0.524000	-0.526000	-0.517000
-0.095258	-0.092337	-0.093484	-0.137926	-0.101096	0.015717	-0.509000	-0.509000	-0.528000
-0.096468	-0.092633	-0.092193	0.065957	0.180765	-0.049853	-0.509000	-0.508000	-0.524000
-0.095363	-0.085374	-0.079339	0.027770	0.168093	0.204939	-0.535000	-0.509000	-0.508000
-0.079374	-0.083036	-0.083511	0.014409	0.027827	0.168093	-0.524000	-0.527000	-0.510000
-0.079618	-0.079149	-0.079149	0.178973	-0.061949	-0.036538	-0.507000	-0.507000	-0.510000
-0.079618	-0.079930	-0.080005	0.039794	0.079620	0.104021	-0.512000	-0.513000	-0.529000
-0.073315	-0.071113	-0.069807	0.116892	-0.150232	-0.124915	-0.521000	-0.519000	-0.516000
-0.069391	-0.068829	-0.067120	-0.112893	-0.100620	0.118842	-0.514000	-0.508000	-0.506000
-0.066868	-0.067127	-0.066995	-0.048868	0.001826	0.027602	-0.509000	-0.508000	-0.509000
-0.067364	-0.071789	-0.067937	0.053839	0.079129	0.191935	-0.532000	-0.513000	-0.508000
-0.065902	-0.065859	-0.066523	-0.076196	-0.037957	0.013977	-0.507000	-0.508000	-0.537000
-0.062357	-0.057195	-0.057784	0.041269	0.215214	-0.127435	-0.521000	-0.518000	-0.512000
-0.055312	-0.056073	-0.057694	0.141701	0.103926	0.118535	-0.514000	-0.520000	-0.518000
-0.056495	-0.056058	-0.054613	0.154396	-0.114100	-0.102120	-0.517000	-0.514000	-0.507000
-0.053474	-0.054210	-0.054058	-0.077459	0.002191	0.078113	-0.508000	-0.510000	-0.521000
-0.044338	-0.043078	-0.044244	0.091161	-0.126810	-0.103030	-0.517000	-0.520000	-0.513000
-0.041232	-0.040769	-0.040539	0.155141	-0.075987	-0.038311	-0.508000	-0.508000	-0.508000
-0.041078	-0.041206	-0.040990	-0.024755	0.027523	0.065704	-0.508000	-0.510000	-0.515000
-0.042795	-0.041145	-0.039557	0.091108	0.129480	-0.064222	-0.509000	-0.507000	-0.508000
-0.041018	-0.040984	-0.040974	0.001321	0.014192	0.052913	-0.507000	-0.509000	-0.532000
-0.036186	-0.036955	-0.033205	0.078691	-0.176114	0.215428	-0.537000	-0.526000	-0.519000
-0.030738	-0.031698	-0.029573	-0.140025	-0.114114	0.142912	-0.519000	-0.516000	-0.513000
-0.028336	-0.027697	-0.028060	-0.090596	-0.077258	-0.038798	-0.508000	-0.508000	-0.508000
-0.028060	-0.028005	-0.028060	-0.025119	0.026267	0.039335	-0.507000	-0.508000	-0.510000
-0.028991	-0.029018	-0.029377	0.052558	0.091440	0.104827	-0.512000	-0.514000	-0.508000
-0.026865	-0.027839	-0.021524	0.115910	-0.011974	0.065561	-0.508000	-0.533000	-0.523000
-0.020713	-0.017872	-0.018745	-0.175404	0.167124	-0.114516	-0.518000	-0.520000	-0.515000
-0.017175	-0.018781	-0.017468	-0.127371	-0.090216	0.155501	-0.521000	-0.517000	-0.511000
-0.015957	-0.016173	-0.016100	-0.102934	-0.064932	-0.051612	-0.510000	-0.510000	-0.514000
-0.016157	-0.015447	-0.015972	0.091523	-0.078154	0.012360	-0.508000	-0.509000	-0.511000
-0.015739	-0.014675	-0.014631	0.078913	0.103420	-0.026439	-0.508000	-0.509000	-0.533000
-0.008988	-0.009197	-0.007011	0.052349	-0.177010	-0.164431	-0.531000	-0.526000	-0.517000
-0.005043	-0.003889	-0.003514	-0.140401	-0.115251	-0.077214	-0.514000	-0.511000	-0.517000
-0.004008	-0.003820	-0.002473	0.091702	-0.103546	-0.090751	-0.516000	-0.510000	-0.511000
-0.002604	-0.002594	-0.002197	-0.038329	-0.026405	0.077758	-0.509000	-0.509000	-0.510000
-0.002602	0.007718	0.006354	0.012977	0.065777	0.130414	-0.518000	-0.523000	-0.512000
0.010038	0.009282	0.009395	0.167124	0.091566	0.104268	-0.514000	-0.516000	-0.511000
0.010607	0.010538	0.011017	0.117785	-0.025804	0.026305	-0.510000	-0.511000	-0.509000
0.010389	0.010686	0.010707	-0.012216	0.039365	0.051629	-0.510000	-0.511000	-0.534000
0.016291	0.020391	0.020360	0.078177	-0.178169	-0.128243	-0.521000	-0.519000	-0.521000
0.020391	0.021451	0.022049	0.143916	0.156100	-0.102934	-0.517000	-0.516000	-0.512000
0.022672	0.022628	0.022988	-0.091236	-0.063193	0.040311	-0.511000	-0.512000	-0.515000
0.021911	0.021713	0.021729	0.064775	0.104524	0.118261	-0.517000	-0.518000	-0.514000
0.022582	0.023652	0.030825	0.130174	-0.038843	0.013688	-0.512000	-0.529000	-0.525000
0.032736	0.033230	0.034188	-0.154563	-0.141963	0.142923	-0.520000	-0.516000	-0.516000
0.034886	0.035469	0.033743	-0.091236	-0.078859	0.000028	-0.514000	-0.519000	-0.516000
0.035507	0.036303	0.040694	0.131083	0.077599	0.065589	-0.516000	-0.535000	-0.516000
0.047445	0.047746	0.047931	0.207074	-0.012405	0.026733	-0.516000	-0.518000	-0.521000
0.046775	0.048352	0.047501	0.079235	0.143198	-0.064408	-0.516000	-0.520000	-0.515000
0.048021	0.048376	0.048301	0.130676	-0.039538	0.000288	-0.516000	-0.518000	-0.520000
0.058822	0.059037	0.058609	0.066247	-0.116479	0.131743	-0.522000	-0.523000	-0.519000
0.060232	0.059910	0.060459	-0.128735	0.013459	-0.102557	-0.518000	-0.519000	-0.519000
0.060232	0.060533	0.067301	-0.000024	0.117859	-0.063516	-0.518000	-0.533000	-0.523000
0.070767	0.072132	0.071867	-0.180709	0.143924	0.012547	-0.521000	-0.521000	-0.520000
0.073271	0.072719	0.073140	0.051923	-0.090402	-0.063516	-0.518000	-0.521000	-0.535000
0.080091	0.081182	0.083159	-0.012408	0.209378	-0.181138	-0.533000	-0.525000	-0.526000
0.083108	0.084322	0.084608	0.144527	-0.142500	0.157490	-0.527000	-0.524000	-0.532000
0.106795	0.107206	0.107211	-0.116654	0.052592	-0.169216	-0.531000	-0.533000	-0.538000
0.117372	0.117898	0.128792	-0.181742	0.222790	0.065621	-0.534000	-0.535000	-0.539000
0.128872	0.129781	0.132419	0.078060	0.000462	0.040443	-0.540000	-0.537000	-0.537000
0.131587	0.133184	0.133311	-0.090280	0.196862	-0.130217	-0.535000	-0.538000	-0.535000
-0.264280	-0.264988	-0.262126	-0.104370	-0.139027	0.211158	-0.539000	-0.534000	-0.536000
-0.261847	-0.259809	-0.258944	-0.087263	0.145077	0.028060	-0.537000	-0.536000	-0.532000
-0.251132	-0.250350	-0.250820	0.053631	-0.126606	0.144458	-0.533000	-0.534000	-0.537000
-0.252229	-0.252777	-0.248629	0.157695	0.185109	0.224253	-0.540000	-0.532000	-0.534000
-0.248712	-0.249564	-0.249097	-0.061907	0.028307	0.093629	-0.534000	-0.533000	-0.533000
-0.247843	-0.237733	-0.237672	0.105117	0.054181	-0.139183	-0.530000	-0.529000	-0.532000
-0.238827	-0.237791	-0.237791	-0.101379	0.170858	0.014407	-0.531000	-0.531000	-0.528000
-0.235581	-0.235227	-0.235227	0.131826	-0.061841	0.053876	-0.530000	-0.530000	-0.527000
-0.225217	-0.223475	-0.224875	0.066745	-0.101861	-0.115272	-0.527000	-0.530000	-0.531000
-0.224428	-0.222315	-0.223160	0.171475	-0.153849	0.079853	-0.526000	-0.528000	-0.530000
-0.224006	-0.211204	-0.212416	0.106240	0.159008	-0.010416	-0.523000	-0.526000	-0.523000

-0.209974	-0.209974	-0.210404	0.156944	0.079397	0.105606	-0.523000	-0.525000	-0.529000
-0.211985	-0.212199	-0.197760	0.145157	0.171152	0.222157	-0.538000	-0.520000	-0.524000
-0.197676	-0.196167	-0.196915	0.001876	-0.127540	0.066709	-0.520000	-0.521000	-0.523000
-0.197671	-0.198049	-0.198049	0.079925	0.132301	0.145717	-0.524000	-0.524000	-0.519000
-0.185171	-0.184343	-0.184288	0.158044	-0.063228	0.080011	-0.519000	-0.521000	-0.518000
-0.183596	-0.182848	-0.184304	-0.114554	0.015273	0.027459	-0.518000	-0.520000	-0.525000
-0.184842	-0.186025	-0.171167	0.119309	-0.127326	0.197523	-0.533000	-0.516000	-0.516000
-0.171307	-0.171079	-0.170748	-0.010277	0.027829	-0.062985	-0.517000	-0.516000	-0.516000
-0.170381	-0.170051	-0.170417	-0.024020	0.001862	0.015185	-0.515000	-0.515000	-0.516000
-0.170748	-0.171042	-0.171741	0.040626	0.066563	0.093261	-0.518000	-0.519000	-0.527000
-0.172307	-0.171256	-0.172887	0.105650	-0.140667	0.170231	-0.525000	-0.530000	-0.534000
-0.172304	-0.173260	-0.158092	0.183542	-0.177103	0.208332	-0.535000	-0.517000	-0.516000
-0.158610	-0.157629	-0.156869	-0.089350	-0.063319	-0.011446	-0.514000	-0.513000	-0.514000
-0.157175	-0.156688	-0.159511	0.027194	0.065939	0.106446	-0.517000	-0.527000	-0.522000
-0.157998	-0.160717	-0.147237	-0.140667	0.157440	0.219626	-0.538000	-0.520000	-0.516000
-0.144434	-0.144413	-0.143486	-0.113111	-0.087963	-0.024334	-0.513000	-0.511000	-0.512000
-0.144131	-0.144413	-0.147509	0.015067	0.041207	0.092361	-0.513000	-0.524000	-0.528000
-0.147393	-0.148919	-0.133905	0.169534	0.182007	0.218360	-0.538000	-0.519000	-0.528000
-0.136214	-0.135440	-0.133795	-0.113097	-0.150028	-0.138061	-0.525000	-0.521000	-0.510000
-0.130384	-0.130627	-0.131028	-0.126097	0.001639	-0.049483	-0.511000	-0.514000	-0.535000
-0.135503	-0.136263	-0.121683	0.105841	0.206196	0.218360	-0.538000	-0.519000	-0.516000
-0.120157	-0.118251	-0.119168	-0.113745	-0.087963	0.105635	-0.513000	-0.515000	-0.519000
-0.119634	-0.122631	-0.110131	0.118162	0.143498	0.193398	-0.532000	-0.528000	-0.526000
-0.109714	-0.105806	-0.105395	-0.150028	-0.138324	-0.074334	-0.514000	-0.511000	-0.509000
-0.104983	-0.104776	-0.104177	-0.049838	-0.036286	-0.011347	-0.509000	-0.510000	-0.509000
-0.103973	-0.107280	-0.109541	0.053416	0.066483	0.157835	-0.522000	-0.533000	-0.538000
-0.110951	-0.095767	-0.094200	0.193761	0.218360	-0.125825	-0.522000	-0.511000	-0.519000
-0.094823	-0.092631	-0.094090	0.079980	-0.114114	-0.036720	-0.507000	-0.517000	-0.520000
-0.095006	-0.095372	-0.097929	0.130783	0.144145	0.156607	-0.522000	-0.536000	-0.532000
-0.087187	-0.086420	-0.082149	0.206200	-0.162429	-0.173886	-0.535000	-0.516000	-0.522000
-0.083092	-0.081845	-0.081156	-0.101308	-0.125454	-0.112893	-0.519000	-0.515000	-0.508000
-0.079276	-0.079524	-0.082101	-0.087793	-0.011699	0.001630	-0.507000	-0.521000	-0.538000
-0.084841	-0.079825	-0.079668	0.156307	0.217094	-0.050486	-0.509000	-0.508000	-0.509000
-0.079463	-0.081393	-0.073491	0.053581	0.066495	0.143867	-0.519000	-0.531000	-0.516000
-0.068797	-0.066798	-0.061115	-0.160485	0.130727	0.091620	-0.510000	-0.532000	-0.529000
-0.059747	-0.055337	-0.053764	-0.163291	-0.149912	-0.090020	-0.515000	-0.508000	-0.507000
-0.053285	-0.053390	-0.053390	0.014592	0.025876	0.039072	-0.508000	-0.508000	-0.533000
-0.048481	-0.045768	-0.046057	0.052218	0.192104	-0.139163	-0.526000	-0.524000	-0.511000
-0.043075	-0.043126	-0.043545	0.167069	0.104009	0.117544	-0.514000	-0.519000	-0.508000
-0.040245	-0.040166	-0.034559	0.143106	-0.012314	0.038996	-0.507000	-0.531000	-0.521000
-0.032484	-0.031086	-0.028406	-0.163372	0.155069	-0.126810	-0.521000	-0.510000	-0.508000
-0.027921	-0.023492	-0.019164	-0.063951	-0.050749	0.203298	-0.535000	-0.520000	-0.507000
-0.015306	-0.015179	-0.015179	0.143381	-0.012495	0.025792	-0.509000	-0.509000	-0.509000
-0.015166	-0.010886	-0.008296	0.064108	0.039318	0.204174	-0.534000	-0.529000	-0.526000
-0.007409	-0.005328	-0.003825	-0.152401	0.178843	0.143498	-0.519000	-0.514000	-0.517000
-0.004459	-0.002195	-0.002573	0.116098	0.129935	-0.012545	-0.509000	-0.509000	-0.534000
0.003786	0.003299	0.002937	0.026386	-0.178966	0.192146	-0.532000	-0.535000	-0.530000
0.005403	0.006921	0.007733	0.204557	-0.153559	-0.128281	-0.522000	-0.519000	-0.516000
0.008902	0.008902	0.010005	0.143498	-0.102163	-0.089922	-0.516000	-0.511000	-0.510000
0.010361	0.011164	0.011198	0.065749	-0.052148	-0.000364	-0.509000	-0.510000	-0.537000
0.014318	0.017838	0.018974	0.014651	0.217954	-0.153269	-0.529000	-0.526000	-0.518000
0.020730	0.022211	0.022446	-0.140799	-0.116331	-0.077279	-0.515000	-0.512000	-0.513000
0.022944	0.022446	0.022854	-0.051148	-0.025905	0.025936	-0.512000	-0.511000	-0.513000
0.022490	0.023287	0.027746	0.052332	0.091154	-0.012993	-0.512000	-0.537000	-0.534000
0.028056	0.029147	0.030167	0.219217	0.205810	0.192163	-0.531000	-0.527000	-0.516000
0.034393	0.035538	0.043021	0.180423	0.103435	0.091120	-0.515000	-0.528000	-0.528000
0.043888	0.044179	0.047548	0.182007	-0.153755	0.170505	-0.527000	-0.514000	-0.517000
0.047447	0.047080	0.048663	-0.026158	0.091865	0.117632	-0.518000	-0.517000	-0.517000
0.047838	0.048112	0.047931	-0.051648	0.040393	0.052844	-0.516000	-0.518000	-0.537000
0.053478	0.055379	0.057114	0.104596	0.220074	0.194283	-0.531000	-0.524000	-0.523000
0.059070	0.058720	0.060943	-0.141496	0.144604	0.157207	-0.524000	-0.518000	-0.517000
0.060827	0.060116	0.060232	-0.050529	-0.012729	0.027459	-0.518000	-0.519000	-0.519000
0.060232	0.060232	0.061335	0.052758	0.066188	0.105650	-0.519000	-0.518000	-0.517000
0.061216	0.066111	0.067979	-0.037125	-0.024891	0.220862	-0.537000	-0.535000	-0.530000
0.068990	0.069971	0.070796	0.207074	-0.166805	-0.154997	-0.528000	-0.525000	-0.523000
0.070911	0.071756	0.072187	0.156280	-0.129757	0.130040	-0.523000	-0.520000	-0.521000
0.072326	0.071865	0.073411	-0.103324	0.000654	0.118313	-0.521000	-0.520000	-0.520000
0.072581	0.072581	0.081611	-0.077636	-0.037662	-0.024666	-0.520000	-0.531000	-0.532000
0.081765	0.082548	0.083717	-0.168839	0.197152	-0.155758	-0.529000	-0.524000	-0.523000
0.084907	0.084233	0.084233	0.104820	-0.090387	-0.012065	-0.524000	-0.524000	-0.524000
0.084233	0.084073	0.084394	0.000658	0.078316	0.091700	-0.523000	-0.525000	-0.531000
0.094950	0.094658	0.095873	0.131572	-0.169216	0.183386	-0.532000	-0.527000	-0.528000
0.096055	0.097297	0.097113	-0.039010	0.092577	-0.129756	-0.528000	-0.527000	-0.524000
0.096560	0.105850	0.106138	-0.117113	-0.077000	0.222157	-0.538000	-0.535000	-0.530000
0.106393	0.106393	0.108046	0.208332	0.000666	0.065875	-0.530000	-0.532000	-0.532000
0.108046	0.108046	0.109926	0.104542	0.158356	0.170871	-0.532000	-0.529000	-0.534000
0.118098	0.117846	0.117465	-0.116313	-0.025710	0.014516	-0.535000	-0.535000	-0.536000
0.118540	0.118659	0.118280	0.027101	0.208722	-0.038200	-0.533000	-0.533000	-0.532000
0.119688	0.119534	0.120561	0.079662	-0.076924	-0.064531	-0.533000	-0.532000	-0.531000
0.121961	0.121813	0.128792	-0.103608	-0.141735	-0.129487	-0.532000	-0.535000	-0.539000
0.131023	0.130431	0.131555	0.066116	-0.063989	0.117717	-0.535000	-0.536000	-0.535000
0.131309	0.133813	-0.237668	-0.077503	0.169317	-0.169029	-0.535000	-0.528000	-0.536000
-0.238717	-0.172554	-0.264638	-0.087749	0.198870	-0.151402	-0.529000	-0.534000	-0.537000
-0.265513	-0.263067	-0.263521	-0.164539	-0.152179	-0.125618	-0.533000	-0.535000	-0.537000
-0.262925	-0.265214	-0.260783	-0.100686	0.118507	0.171217	-0.539000	-0.535000	-0.538000
-0.263476	-0.261216	-0.261216	-0.061337	0.131033	-0.049133	-0.536000	-0.536000	-0.535000
-0.259525	-0.261723	-0.261236	-0.036524	-0.023581	0.092892	-0.537000	-0.536000	-0.536000

-0.259666	-0.258887	-0.250780	0.105328	0.066816	0.041675	-0.537000	-0.532000	-0.533000
-0.250689	-0.250526	-0.248786	-0.113888	-0.165519	-0.100703	-0.532000	-0.531000	-0.533000
-0.248514	-0.251617	-0.247716	-0.074906	-0.049131	0.197742	-0.537000	-0.532000	-0.534000
-0.248936	-0.247876	-0.247699	-0.022539	0.041497	0.015145	-0.535000	-0.534000	-0.530000
-0.238967	-0.236823	-0.237817	0.065993	-0.127149	-0.074502	-0.528000	-0.531000	-0.528000
-0.235237	-0.236698	-0.237116	0.144898	-0.023516	0.002266	-0.532000	-0.530000	-0.529000
-0.235405	-0.235850	-0.235405	0.119110	-0.048491	-0.036115	-0.530000	-0.529000	-0.530000
-0.235850	-0.237649	-0.234317	-0.009913	0.027471	0.185332	-0.534000	-0.530000	-0.528000
-0.225300	-0.224790	-0.224171	0.042379	-0.140272	-0.087831	-0.526000	-0.526000	-0.528000
-0.223731	-0.222883	-0.223835	-0.075438	0.001575	0.065912	-0.526000	-0.528000	-0.528000
-0.225023	-0.222510	-0.222510	0.131988	0.145366	-0.061710	-0.525000	-0.525000	-0.525000
-0.222510	-0.221797	-0.222934	-0.049359	-0.035774	-0.023329	-0.524000	-0.526000	-0.527000
-0.223357	-0.221696	-0.221715	-0.009857	0.053350	0.040875	-0.526000	-0.526000	-0.534000
-0.224440	-0.224020	-0.225907	0.092226	0.197246	0.210594	-0.536000	-0.539000	-0.522000
-0.210146	-0.210539	-0.210633	0.223204	-0.035843	0.027397	-0.523000	-0.523000	-0.523000
-0.209974	-0.210608	-0.210589	0.053907	-0.114397	-0.100230	-0.523000	-0.523000	-0.521000
-0.209784	-0.209402	-0.209803	-0.075008	-0.062465	0.002234	-0.523000	-0.524000	-0.523000
-0.209974	-0.209412	-0.209111	0.014565	0.066460	0.092598	-0.523000	-0.522000	-0.524000
-0.209143	-0.212981	-0.211920	-0.049108	0.131919	0.185889	-0.532000	-0.535000	-0.535000
-0.211016	-0.197201	-0.197099	0.198849	0.210201	-0.101209	-0.520000	-0.520000	-0.521000
-0.197547	-0.197167	-0.197149	0.028463	-0.075334	-0.061733	-0.520000	-0.520000	-0.520000
-0.196537	-0.196321	-0.196065	-0.036657	-0.023219	0.093801	-0.521000	-0.522000	-0.529000
-0.197450	-0.198196	-0.199656	0.119136	-0.153043	0.184896	-0.531000	-0.534000	-0.538000
-0.199193	-0.185171	-0.184560	0.198467	0.221566	-0.023175	-0.519000	-0.519000	-0.517000
-0.183241	-0.184223	-0.184659	-0.075655	-0.036464	0.001869	-0.518000	-0.521000	-0.518000
-0.182377	-0.184771	-0.185296	0.132389	0.053639	0.158138	-0.523000	-0.528000	-0.526000
-0.184594	-0.170431	-0.171079	-0.140934	0.171419	-0.075635	-0.517000	-0.517000	-0.518000
-0.170191	-0.169534	-0.170520	-0.036464	-0.088932	0.080137	-0.516000	-0.519000	-0.531000
-0.172633	-0.170910	-0.173293	0.131881	-0.163868	0.157835	-0.522000	-0.533000	-0.537000
-0.173295	-0.157995	-0.159905	0.195930	0.220481	-0.024161	-0.514000	-0.520000	-0.515000
-0.157715	-0.157449	-0.158031	0.144320	0.001858	0.131925	-0.518000	-0.524000	-0.528000
-0.159814	-0.160700	-0.160674	0.169881	0.182609	0.195642	-0.533000	-0.535000	-0.530000
-0.149866	-0.148408	-0.145820	0.207684	-0.162312	-0.138281	-0.525000	-0.518000	-0.512000
-0.143493	-0.144602	-0.145765	-0.099899	0.052372	0.131627	-0.518000	-0.521000	-0.532000
-0.147884	-0.148718	-0.131550	0.156970	0.194024	0.206445	-0.535000	-0.512000	-0.510000
-0.130989	-0.133950	-0.134150	-0.061923	0.053429	0.168899	-0.524000	-0.527000	-0.528000
-0.124468	-0.123821	-0.122492	0.181327	-0.149965	-0.162054	-0.531000	-0.525000	-0.522000
-0.120819	-0.121044	-0.119640	-0.138899	0.157262	0.168552	-0.523000	-0.518000	-0.528000
-0.121930	-0.111174	-0.107564	0.132255	0.181386	-0.173007	-0.533000	-0.520000	-0.528000
-0.108268	-0.109927	-0.100264	0.144436	0.181386	0.206200	-0.536000	-0.535000	-0.531000
-0.098890	-0.097544	-0.098097	-0.174915	-0.161097	-0.149334	-0.529000	-0.532000	-0.539000
-0.098815	-0.092234	-0.092281	0.192728	0.218184	0.001833	-0.508000	-0.508000	-0.530000
-0.086172	-0.084136	-0.073834	0.051960	-0.150727	-0.137992	-0.525000	-0.535000	-0.525000
-0.072112	-0.073427	-0.069318	-0.173657	-0.138240	0.215092	-0.537000	-0.520000	-0.524000
-0.070467	-0.071024	-0.067667	0.155377	0.167685	0.180765	-0.528000	-0.512000	-0.528000
-0.059845	-0.050759	-0.048663	0.104225	0.179523	0.215428	-0.537000	-0.535000	-0.534000
-0.037335	-0.033807	-0.024276	0.203908	0.202222	-0.151779	-0.529000	-0.538000	-0.531000
-0.022068	-0.022735	-0.021249	0.215829	-0.163614	0.192772	-0.532000	-0.528000	-0.511000
-0.002003	-0.001360	0.001055	0.180135	-0.051668	0.050867	-0.509000	-0.538000	-0.526000
0.005962	0.007191	0.007718	0.216462	-0.140401	0.155421	-0.521000	-0.518000	-0.515000
0.010096	0.010597	0.015436	-0.115741	-0.077514	-0.038764	-0.512000	-0.532000	-0.535000
0.014874	0.017164	0.018266	0.192753	0.205815	0.180765	-0.528000	-0.524000	-0.530000
0.029781	0.032821	0.036110	0.168301	-0.165879	-0.129195	-0.521000	-0.514000	-0.537000
0.040195	0.042064	0.045757	0.026100	0.219217	0.194065	-0.532000	-0.521000	-0.522000
0.045212	0.048428	0.054203	-0.129893	0.155379	0.014027	-0.517000	-0.535000	-0.533000
0.054942	0.055949	0.055879	0.207755	-0.179884	0.183612	-0.530000	-0.530000	-0.527000
0.056803	0.060116	0.060824	-0.166404	0.170505	0.092633	-0.518000	-0.519000	-0.533000
0.068734	0.070573	0.071495	0.078790	0.196269	0.170828	-0.528000	-0.521000	-0.520000
0.072006	0.073174	0.079504	-0.116411	0.041084	-0.051947	-0.520000	-0.538000	-0.531000
0.082328	0.083508	0.083658	0.222157	0.183729	0.026166	-0.523000	-0.524000	-0.521000
0.084414	0.084644	0.084806	0.118103	-0.076885	-0.051900	-0.523000	-0.524000	-0.524000
0.084317	0.083905	0.084219	-0.038440	-0.024894	0.013651	-0.524000	-0.525000	-0.526000
0.085173	0.084976	0.084651	0.052876	-0.129883	-0.103255	-0.523000	-0.521000	-0.535000
0.092904	0.093830	0.094538	-0.063641	0.208332	0.196896	-0.533000	-0.526000	-0.527000
0.095209	0.095751	0.095901	0.026981	0.053080	0.170580	-0.530000	-0.529000	-0.528000
0.095719	0.095725	0.095234	-0.155566	-0.142851	-0.024749	-0.528000	-0.527000	-0.528000
0.095415	0.095253	0.095253	-0.011736	0.000663	0.013679	-0.527000	-0.527000	-0.528000
0.095434	0.095434	0.095817	0.039713	0.065872	0.079535	-0.528000	-0.528000	-0.523000
0.096043	0.096310	0.096859	0.143965	-0.063988	-0.050691	-0.526000	-0.529000	-0.528000
0.096676	0.097528	0.097547	0.118263	0.131703	-0.103279	-0.526000	-0.526000	-0.531000
0.106020	0.105714	0.106524	-0.090287	0.012826	0.026604	-0.530000	-0.534000	-0.531000
0.106946	0.105770	0.107471	0.195735	-0.012493	0.039939	-0.530000	-0.532000	-0.529000
0.107521	0.107218	0.107218	0.117976	-0.051264	-0.038681	-0.531000	-0.531000	-0.531000
0.107199	0.108423	0.107805	0.078738	0.091854	0.144670	-0.531000	-0.534000	-0.530000
0.108263	0.109354	0.109053	0.184075	-0.155806	-0.142777	-0.529000	-0.528000	-0.528000
0.109043	0.108140	0.107855	-0.103322	-0.090345	-0.077021	-0.528000	-0.528000	-0.531000
0.108468	0.109284	0.118381	-0.063304	0.131202	-0.130001	-0.529000	-0.536000	-0.534000
0.117600	0.118040	0.118075	0.000936	0.040196	0.053191	-0.535000	-0.535000	-0.535000
0.119308	0.119404	0.119181	-0.011914	0.105486	0.118641	-0.535000	-0.534000	-0.535000
0.119395	0.119111	0.120567	0.130981	0.196096	0.092373	-0.534000	-0.535000	-0.533000
0.120107	0.120116	0.119982	0.144501	0.157061	0.170292	-0.533000	-0.535000	-0.532000
0.121238	0.120561	0.121187	0.183772	-0.116054	-0.090084	-0.532000	-0.532000	-0.538000
0.128827	0.129996	0.129996	-0.155143	0.014255	-0.038086	-0.540000	-0.540000	-0.539000
0.129755	0.128882	0.128932	-0.025383	-0.011388	0.026621	-0.538000	-0.537000	-0.536000
0.129609	0.129663	0.131220	0.053045	0.091117	0.104697	-0.536000	-0.537000	-0.535000
0.130661	0.131884	0.130856	0.144349	0.130302	0.156392	-0.535000	-0.536000	-0.535000
0.131919	0.133826	0.134327	0.208654	0.183161	-0.142193	-0.535000	-0.537000	-0.533000

```
-0.249735      -0.238180      -0.223513      -0.116196       0.117654      -0.154030      -0.532000      -0.527000      -0.516000
-0.171596      -0.159938      -0.136245       0.027751       0.053440      -0.153125      -0.530000      -0.530000      -0.515000
-0.105310      -0.093457      -0.053696      -0.162228       0.119034      -0.087803      -0.514000      -0.508000      -0.529000
-0.047495      -0.006343       0.004742       0.064572      -0.151488       0.154225      -0.521000      -0.527000      -0.530000
 0.017477       0.023010       0.033221       0.179556      -0.165472       0.078483      -0.513000      -0.522000      -0.514000
 0.036289       0.036427       0.060942       0.155062      -0.012332       0.012156      -0.515000      -0.518000      -0.530000
 0.069602       0.072357       0.072533       0.039971       0.183072       0.025928      -0.520000      -0.522000      -0.520000
 0.071958       0.072333       0.092474       0.065124       0.091543       0.104522      -0.521000      -0.538000      -0.532000
 0.221524      -0.025937      -0.507000
```

APPENDIX C

SOURCE CODE

LIST OF SOURCE CODE

The runModel Matlab program was written to build the mesh topology discussed as well as to run the model in a simulation given certain parameters using the ode23t numerical sovler. The first section presented in the program requires the user to define all input parameters. Next, the mesh is developed through a series of functions as well as the initial condition of the mesh model. Spring lengths and the initial condition triangulation is determined. The model is than ran using ode23t and the @dermesh function. A portion of the work energy theorem is held within the function NiCheck, calculating the work related to the generalized speeds.

Listing C.1: Build Mesh and Run Simulation: runModel

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                    RUN MODEL                                    %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear
clc

K  = 1000;
DMP= 800;

N = 5;
M = 5;

xLength = 0.3;
yLength = 0.3;
zLength = 0.127;
```

```
17
18  startPressure = 0;
19  pressure = 0; %Pa [N/m^2]
20  endPressure = 0;
21
22  setDynamicPressure = 0;
23
24  %                  (T2)---(P2)---(T3)                  (T6)---(P4)---(T7)
25  %                    /            \                      /            \
26  %                   /              \                    /              \
27  % (T0)--(P1)--(T1)/                \(T4)--(P3)--(T5)/                \(T8)---(P5)--
28
29  dynPres_Time_1 = 10;
30  dynPres_Time_2 = 15;
31  dynPres_Time_3 = 25;
32  dynPres_Time_4 = 30;
33  dynPres_Time_5 = 40;
34  dynPres_Time_6 = 45;
35  dynPres_Time_7 = 55;
36  dynPres_Time_8 = 60;
37  TFINAL = 5;
38
39  P1 = 6000;
40  P2 = 125;
41  P3 = 250;
42  P4 = 375;
43  P5 = 500;
44
45  pMX = P5;
46
47  totalMass = 0.1785;%Kg
48  L_Bed = 0.30; %m
49
50  L=0;%m
51  grav =9.81; %m/s^2
52
53  TINITIAL = 0;
54  INTEGSTP = 0.1;
55  RELERR = 1e-8; %1e-3 Default
56  ABSERR = 1e-9; %1e-6 Default
57
58  %-------------------------------------------------------------------------
59
60  numBods = N^2+4*(N-1)^2;
61  mass = totalMass / numBods;
62  area = (2*(xLength*zLength)+2*(yLength*zLength)+(yLength*xLength))/numBods;
63  Lo=L_Bed/(N-1) - L;
64
65  pVector = zeros(3*(N^2),1);
66  gVector = repmat([0;0;-grav*mass],N^2+4*(N-1)^2,1);
67
68  KSt = repmat(K,    2*N*(N-1)+2*((N-1)^2)*4    ,1);
69  KSh = repmat(K,    2*((N-1)^2 + 4*(N-1)*(M-1)),1);
70  KBe = repmat(K,    2*(N)*(N-2) + 4*((N-2)*(M-1) + (M-2)*(N-1)),1);
71  %KRi = repmat(0,    8*(N-2),1);
72
73  dampSt = repmat(DMP,   2*N*(N-1)+2*((N-1)^2)*4,1);
74  dampSh = repmat(DMP,   2*((N-1)^2 + 4*(N-1)*(M-1)),1);
75  dampBe = repmat(DMP,   2*(N)*(N-2) + 4*((N-2)*(M-1) + (M-2)*(N-1)),1);
76  %dampRI = repmat(0,   8*(N-2),1);
77
78  %-------------------------------------------------------------------------
79
80  numConstrainedBods = 2*N+2*(M-2);
81
82  Masses = repmat(-mass,1,3*(N^2+4*(N-1)^2 - numConstrainedBods));
83  A = sparse(diag(Masses));
84  clear Masses
85  eCheck = 0;
86
87  %-------------------------------------------------------------------------
88
89  [FT, F1, F2, F3, F4] = buildBedBodyMeshes(N,M);
90
91  structural_bodyMap = structuralSprings(FT, F1, F2, F3, F4, N, M);
92  shear_bodyMap = shearSprings(FT, F1, F2, F3, F4, N, M);
93  bend_bodyMap = bendSprings(FT, F1, F2, F3, F4, N, M);
94  ridge_bodyMap = ridgeSprings(FT, F1, F2, F3, F4, N, M);
95
96  [springMap_ST, springCntST] = springMapBuilder_Structural(N, M, structural_bodyMap);
97  [springMap_SH, springCntSH] = springMapBuilder_Shear(N, M, shear_bodyMap);
98  [springMap_BE, springCntBE] = springMapBuilder_Bend(N, M, bend_bodyMap);
99  [springMap_RI, springCntRI] = springMapBuilder_Ridge(N, M, ridge_bodyMap);
100
101 [ x_FT,y_FT,z_FT, x_F1,y_F1,z_F1, x_F2,y_F2,z_F2, x_F3,y_F3,z_F3, x_F4,y_F4,z_F4] =
        buildBedCoordMesh_Init(N,M, xLength, yLength, zLength);
```

```
Q = qBuild(N,M,x_FT,y_FT,z_FT, x_F1,y_F1,z_F1, x_F2,y_F2,z_F2, x_F3,y_F3,z_F3, x_F4,y_F4,z_F4,
    eCheck);

LoSt = lengthBuilder(springCntST, N, M, Q);
LoSh = lengthBuilder(springCntSH, N, M, Q);
LoBe = lengthBuilder(springCntBE, N, M, Q);
LoRi = lengthBuilder(springCntRI, N, M, Q);

%-------------------------------------------------------------------------

[x, y, z, Dx, Dy, Dz, ~] = buildBedCoords(N,M,Q);
DT = delaunayTriangulation(x,y,z);
Con = freeBoundary(DT);
[~,Vo] = convexHull(DT);

%-------------------------------------------------------------------------

[conQ,varQ,qList] = breakDownQ(N,M,Q,eCheck);

%-------------------------------------------------------------------------

options = odeset('RelTol',RELERR, 'AbsTol',ABSERR, 'MaxStep',INTEGSTP);
T = TINITIAL;
PrintCounter = 0;

Qr(1,:) = [x',y',z',Dx',Dy',Dz',eCheck];
qcnt=2;
qEnd = 0;
qBegin = 1;

fname1 = strcat(num2str(N),'x',num2str(M),'_press_',num2str(pMX),'_t',num2str(TFINAL),'_k',num2str
    (K),'thesis',".txt");
fname2 = strcat(num2str(N),'x',num2str(M),'_press_',num2str(pMX),'_t',num2str(TFINAL),'_k',num2str
    (K),'thesis',".mat");
fSaveName1 = strcat('C:\Users\CAlmendariz\Documents\0204\',fname1);
fSaveName2 = strcat('C:\Users\CAlmendariz\Documents\0204\',fname2);
save(fSaveName2,'LoSt','LoSh','LoBe','LoRi','pressure','mass', 'KSh', 'KSt', 'KBe', 'KRi', 'dampSt
    ', 'dampSh', 'dampRI', 'dampBe', 'springCntST', 'springCntSH', 'springCntBE', 'springCntRI', '
    springMap_ST', 'springMap_SH', 'springMap_BE', 'springMap_RI')
FID = fopen(fSaveName1,'a');
writeToFile(FID, Qr(1,:), T, pressure)
fclose(FID);

while 1

  if( TFINAL>=TINITIAL && T+0.01*INTEGSTP>=TFINAL ) PrintCounter = -1; end
  if( TFINAL<=TINITIAL && T+0.01*INTEGSTP<=TFINAL ) PrintCounter = -1; end

  [TimeOdeArray,varQ] = ode23t( @dermesh, [T T+INTEGSTP], varQ, options, conQ, qList, N, M,
      springMap_ST, springMap_SH, springMap_BE, springMap_RI, springCntST, springCntSH,
      springCntBE, springCntRI, Lo, gVector, KSt, dampSt, KSh, dampSh, KBe, dampBe, dampRI, KRi, A
      , pressure, Vo, area, LoSt, LoSh, LoBe, LoRi, Con);
  TimeAtEndOfArray = TimeOdeArray( length(TimeOdeArray) );
  if( abs(TimeAtEndOfArray - (T+INTEGSTP) ) >= abs(0.001*INTEGSTP) ) warning('numerical
      integration failed'); fprintf("Booo"); pause();break;  end

  T = TimeAtEndOfArray;
  Lt = length(TimeOdeArray)-1;
  if( T >= TFINAL) break; end

    %                 (T2)---(P2)---(T3)                (T6)---(P4)---(T7)
    %                   /           \                     /           \
    %                 /               \                 /               \
    % (T0)--(P1)--(T1)/                 \(T4)--(P3)--(T5)/                 \(T8)---(P5)--

  if setDynamicPressure ==1
    if (T<=dynPres_Time_1) pressure = P1; end
    if (T>dynPres_Time_1 && T<=dynPres_Time_2) pressure = P1 + ((P2-P1)/(dynPres_Time_2-
        dynPres_Time_1))*(T-dynPres_Time_1); end
    if (T>dynPres_Time_2)&&(T<=dynPres_Time_3) pressure = P2; end
    if (T>dynPres_Time_3)&&(T<=dynPres_Time_4) pressure = P2 + ((P3-P2)/(dynPres_Time_4-
        dynPres_Time_3))*(T-dynPres_Time_3); end
    if (T>dynPres_Time_4)&&(T<=dynPres_Time_5) pressure = P3; end
    if (T>dynPres_Time_5 && T<=dynPres_Time_6) pressure = P3 + ((P4-P3)/(dynPres_Time_6-
        dynPres_Time_5))*(T-dynPres_Time_5); end
    if (T>dynPres_Time_6)&&(T<=dynPres_Time_7) pressure = P4; end
    if (T>dynPres_Time_7)&&(T<=dynPres_Time_8) pressure = P4 + ((P5-P4)/(dynPres_Time_8-
        dynPres_Time_7))*(T-dynPres_Time_7); end
    if (T>dynPres_Time_8) pressure = P5; end
  else
    pressure = 0;
  end

  varQ = varQ(size(varQ,1),:)';
  Q = rebuildQ(N,M,varQ, conQ, qList)';

  FID = fopen(fSaveName1,'a');
```

```
176     writeToFile(FID, Q(1,:), T, pressure)
177     fclose(FID);
178
179     clear Q
180     qcnt=qcnt+1;
181
182     fprintf('%f\n',T);
183
184 end
185
186 function DQ = dermesh(~, varQ, conQ, qList , N, M, springMap_ST, springMap_SH, springMap_BE,
        springMap_RI, springCntST, springCntSH, springCntBE, springCntRI, ~, gVector, KSt, dampSt, KSh
        , dampSh, KBe, dampBe, dampRI, KRi, A, pressure, Vo, area, LoSt, LoSh, LoBe, LoRi,Con)
187     %Rebuild Q from varQ and conQ
188     Q = rebuildQ(N,M,varQ, conQ, qList);
189
190     [ x, y, z, Dx, Dy, Dz, ~] = buildBedCoords(N,M,Q);
191
192     [pVector, pressure,Vol] = calcPressure2(x,y,z,pressure,area,Vo,Con);
193
194     [magVector_ST, xVector_ST, yVector_ST, zVector_ST, stretch_ST, stretchP_ST] = vectorBuilder(
            LoSt, springCntST, x,y,z,Dx,Dy,Dz);
195     [magVector_SH, xVector_SH, yVector_SH, zVector_SH, stretch_SH, stretchP_SH] = vectorBuilder(
            LoSh, springCntSH, x,y,z,Dx,Dy,Dz);
196     [magVector_BE, xVector_BE, yVector_BE, zVector_BE, stretch_BE, stretchP_BE] = vectorBuilder(
            LoBe, springCntBE, x,y,z,Dx,Dy,Dz);
197     [magVector_RI, xVector_RI, yVector_RI, zVector_RI, stretch_RI, stretchP_RI] = vectorBuilder(
            LoRi, springCntRI, x,y,z,Dx,Dy,Dz);
198
199     [St1, St2] = springForceBuilder(N, M, xVector_ST, yVector_ST, zVector_ST, magVector_ST,
            springMap_ST, stretch_ST, stretchP_ST);
200     [Sh1, Sh2] = springForceBuilder(N, M, xVector_SH, yVector_SH, zVector_SH, magVector_SH,
            springMap_SH, stretch_SH, stretchP_SH);
201     [Sb1, Sb2] = springForceBuilder(N, M, xVector_BE, yVector_BE, zVector_BE, magVector_BE,
            springMap_BE, stretch_BE, stretchP_BE);
202     [Se1, Se2] = springForceBuilder(N, M, xVector_RI, yVector_RI, zVector_RI, magVector_RI,
            springMap_RI, stretch_RI, stretchP_RI);
203
204     %nonlinearKs
205
206     [RHS] = miscCalcs(St1, St2, dampSt, KSt, Sh1, Sh2, dampSh, KSh, Sb1, Sb2, dampBe, KBe, Se1,
            Se2, dampRI, KRi, gVector, pVector);
207     [RHS2] = filterConstrainedBodies(N,M,RHS);
208     SolutionTo = inverseARHS(A,RHS2);
209     eCheck = NiCheck(N,M,Dx, Dy, Dz, RHS2);
210     [conQ,varQ,qList] = breakDownQ(N,M,Q,eCheck);
211     DQ = solPosFun(x,y,z,Dx,Dy,Dz,SolutionTo,N,M,eCheck, varQ, qList);
212 end
213
214 %-------------------------------------------------------------------------
215
216 function      [conQ,varQ,qList] = breakDownQ(N,M,Q,eCheck)
217     numBods = N^2+4*(N-1)^2;
218     Qtemp = reshape(Q(1:size(Q,1)-1),numBods,[]);
219     eCheck = Q(size(Q,1));
220     fixedPoints = (N^2 + (M-1)):(M-1):(N^2 + 4*(M-1)^2);
221     conQ = zeros(length(fixedPoints),6);
222     varQ = zeros(numBods-length(fixedPoints),6);
223
224     qList(:,1) = fixedPoints;
225     dynamicPoints = (1:numBods)';
226     fp = flip(qList(:,1));
227
228     for i=1:length(fp)
229         dynamicPoints(fp(i))=[];
230     end
231
232     qList(1:size(dynamicPoints,1),2) = dynamicPoints;
233
234     for i=1:numBods-size(qList,1)
235         conQ(i,:) = Qtemp(qList(i,1),:);
236     end
237     for j=1:size(qList,1)
238         varQ(j,:) = Qtemp(qList(j,2),:);
239     end
240
241     conQ = reshape(conQ,[],1);
242     varQ = reshape(varQ,[],1);
243     varQ = [varQ ; eCheck];
244
245 end
246
247 %-------------------------------------------------------------------------
248
249 function Q = rebuildQ(N,M,varQ, conQ, qList)
250     numBods = N^2+4*(N-1)^2;
251     Q = zeros(numBods,6);
```

```
252
253     eCheck = varQ(size(varQ,1));
254     varQ = reshape(varQ(1:size(varQ,1)-1),[],6);
255     conQ = reshape(conQ,[],6);
256
257     for i=1:numBods-size(qList,1)
258         Q(qList(i,1),:) = conQ(i,:);
259     end
260     for j=1:size(qList,1)
261         Q(qList(j,2),:) = varQ(j,:);
262     end
263
264     Q = reshape(Q,[],1);
265     Q = [Q ; eCheck];
266 end
267
268 %-------------------------------------------------------------------------
269
270 function [RHS2] = filterConstrainedBodies(N,M,RHS)
271
272     charlie = (N^2 + (M-1));
273     delta = (M-1);
274     sigma = (N^2 + 4*(M-1)^2);
275     fixedPoints = charlie:delta:sigma;
276     fixedPoints =  flip(fixedPoints);
277     RHS2 = RHS;
278
279     fp1 = fixedPoints*3;
280     fp2 = fixedPoints*3-1;
281     fp3 = fixedPoints*3-2;
282
283     fixedPoints2 = reshape([fp1' fp2' fp3']',1,length(fp1)*3)';
284
285
286     for i=1:length(fixedPoints2)
287         RHS2(fixedPoints2(i)) = [];
288     end
289
290
291
292
293  end
294
295 %-------------------------------------------------------------------------
296
297 function writeToFile(fileID, Qr,T,pressure)
298     lenQr = size(Qr,2);
299
300     fprintf(fileID,'%f ',T);
301     fprintf(fileID,'%f ',pressure);
302
303     for i=1:lenQr
304         fprintf(fileID,'%14.12E ',Qr(i));
305     end
306     fprintf(fileID,'\n');
307
308
309 end
310
311 %-------------------------------------------------------------------------
312
313 %Body Connection Mapping : Structural, shear, and bending springs
314 function bodyMap =       structuralSprings(FT, F1, F2, F3, F4, N, M)
315     bodyMap = zeros(N^2+4*(N-1)*(M-1),20);
316
317     %Top Face
318     for xi=2:N+1
319        for yi=2:N+1
320             i = FT(xi,yi);
321             bodyMap(i,1)=FT(xi-1,yi);
322             bodyMap(i,2)=FT(xi,yi+1);
323             bodyMap(i,3)=FT(xi+1,yi);
324             bodyMap(i,4)=FT(xi,yi-1);
325        end
326     end
327
328     %Other Faces
329     for xi=2:M+1
330        for yi=2:N+1
331             i = F1(xi,yi);
332             bodyMap(i,5)=F1(xi-1,yi);
333             bodyMap(i,6)=F1(xi,yi+1);
334             bodyMap(i,7)=F1(xi+1,yi);
335             bodyMap(i,8)=F1(xi,yi-1);
336
337             i = F2(xi,yi);
338             bodyMap(i,9) =F2(xi-1,yi);
```

```
            bodyMap(i,10)=F2(xi,yi+1);
            bodyMap(i,11)=F2(xi+1,yi);
            bodyMap(i,12)=F2(xi,yi-1);

            i = F3(xi,yi);
            bodyMap(i,13)=F3(xi-1,yi);
            bodyMap(i,14)=F3(xi,yi+1);
            bodyMap(i,15)=F3(xi+1,yi);
            bodyMap(i,16)=F3(xi,yi-1);

            i = F4(xi,yi);
            bodyMap(i,17)=F4(xi-1,yi);
            bodyMap(i,18)=F4(xi,yi+1);
            bodyMap(i,19)=F4(xi+1,yi);
            bodyMap(i,20)=F4(xi,yi-1);
        end
    end
    bodyMap = sparse(bodyMap);
end

%-------------------------------------------------------------------------

function bodyMap =      shearSprings(FT, F1, F2, F3, F4, N, M)
    bodyMap = zeros(N^2+4*(N-1)*(M-1),20);

    %Top Face
    for xi=2:N+1
        for yi=2:N+1
            i = FT(xi,yi);
            bodyMap(i,1)=FT(xi-1,yi+1);
            bodyMap(i,2)=FT(xi-1,yi-1);
            bodyMap(i,3)=FT(xi+1,yi+1);
            bodyMap(i,4)=FT(xi+1,yi-1);
        end
    end

    %Other Faces
    for xi=2:M+1
        for yi=2:N+1
            i = F1(xi,yi);
            bodyMap(i,5)=F1(xi-1,yi+1);
            bodyMap(i,6)=F1(xi-1,yi-1);
            bodyMap(i,7)=F1(xi+1,yi+1);
            bodyMap(i,8)=F1(xi+1,yi-1);

            i = F2(xi,yi);
            bodyMap(i,9) =F2(xi-1,yi+1);
            bodyMap(i,10)=F2(xi-1,yi-1);
            bodyMap(i,11)=F2(xi+1,yi+1);
            bodyMap(i,12)=F2(xi+1,yi-1);

            i = F3(xi,yi);
            bodyMap(i,13)=F3(xi-1,yi+1);
            bodyMap(i,14)=F3(xi-1,yi-1);
            bodyMap(i,15)=F3(xi+1,yi+1);
            bodyMap(i,16)=F3(xi+1,yi-1);

            i = F4(xi,yi);
            bodyMap(i,17)=F4(xi-1,yi+1);
            bodyMap(i,18)=F4(xi-1,yi-1);
            bodyMap(i,19)=F4(xi+1,yi+1);
            bodyMap(i,20)=F4(xi+1,yi-1);
        end
    end
    bodyMap = sparse(bodyMap);
end

%-------------------------------------------------------------------------

function bodyMap =      bendSprings(FT, F1, F2, F3, F4, N, M)
    blankMat_NN = zeros(N+4,N+4);
    blankMat_NM = zeros(N+4,M+4);

    blankMat_NN(2:size(FT,1)+1,2:size(FT,2)+1) = FT;
    FT = blankMat_NN;

    blankMat_NM(2:size(F1,1)+1,2:size(F1,2)+1) = F1;
    F1 = blankMat_NM;
    blankMat_NM(2:size(F2,1)+1,2:size(F2,2)+1) = F2;
    F2 = blankMat_NM;
    blankMat_NM(2:size(F3,1)+1,2:size(F3,2)+1) = F3;
    F3 = blankMat_NM;
    blankMat_NM(2:size(F4,1)+1,2:size(F4,2)+1) = F4;
    F4 = blankMat_NM;

    bodyMap = zeros(N^2+4*(N-1)*(M-1),20);
```

```
426     %Top Face
427     for xi=3:N+2
428         for yi=3:N+2
429             i = FT(xi,yi);
430             bodyMap(i,1)=FT(xi-2,yi);
431             bodyMap(i,2)=FT(xi+2,yi);
432             bodyMap(i,3)=FT(xi,yi+2);
433             bodyMap(i,4)=FT(xi,yi-2);
434         end
435     end
436
437     %Other Faces
438     for xi=3:M+2
439         for yi=3:N+2
440             i = F1(xi,yi);
441             bodyMap(i,5)=F1(xi-2,yi);
442             bodyMap(i,6)=F1(xi+2,yi);
443             bodyMap(i,7)=F1(xi,yi+2);
444             bodyMap(i,8)=F1(xi,yi-2);
445
446             i = F2(xi,yi);
447             bodyMap(i,9) =F2(xi-2,yi);
448             bodyMap(i,10)=F2(xi+2,yi);
449             bodyMap(i,11)=F2(xi,yi+2);
450             bodyMap(i,12)=F2(xi,yi-2);
451
452             i = F3(xi,yi);
453             bodyMap(i,13)=F3(xi-2,yi);
454             bodyMap(i,14)=F3(xi+2,yi);
455             bodyMap(i,15)=F3(xi,yi+2);
456             bodyMap(i,16)=F3(xi,yi-2);
457
458             i = F4(xi,yi);
459             bodyMap(i,17)=F4(xi-2,yi);
460             bodyMap(i,18)=F4(xi+2,yi);
461             bodyMap(i,19)=F4(xi,yi+2);
462             bodyMap(i,20)=F4(xi,yi-2);
463         end
464     end
465     bodyMap = sparse(bodyMap);
466 end
467
468 %-------------------------------------------------------------------------
469
470 function bodyMap =      ridgeSprings(FT, F1, F2, F3, F4, N, M)
471     %Top edge springs need to have differing LoTop vs LoSide
472     bodyMap = zeros((N-2)*8,2);
473
474     FT = FT(3:N,3:N);
475     F1 = F1(3:M,3:N);
476     F2 = F2(3:M,3:N);
477     F3 = F3(3:M,3:N);
478     F4 = F4(3:M,3:N);
479
480     %Top Edge Mapping
481     bodyMap(1+0*(N-2):1*(N-2),1:2) = [ FT(1,:)' , F2(1,:)' ];
482     bodyMap(1+1*(N-2):2*(N-2),1:2) = [ FT(:,(N-2)) , flip(F3(1,:)') ];
483     bodyMap(1+2*(N-2):3*(N-2),1:2) = [ FT((N-2),:)' , F4(1,:)' ];
484     bodyMap(1+3*(N-2):4*(N-2),1:2) = [ FT(:,1) , flip(F1(1,:)') ];
485
486     bodyMap(1+4*(N-2):5*(N-2),1:2) = [ F1(:,1) , F4(:,1) ];
487     bodyMap(1+5*(N-2):6*(N-2),1:2) = [ F2(:,1) , F1(:,(N-2)) ];
488     bodyMap(1+6*(N-2):7*(N-2),1:2) = [ F3(:,(N-2)) , F2(:,(N-2)) ];
489     bodyMap(1+7*(N-2):8*(N-2),1:2) = [ F4(:,(N-2)) , F3(:,1) ];
490
491 end
492
493
494 %-------------------------------------------------------------------------
495
496 function [springMap, springCnt] =        springMapBuilder_Structural(N, M, structural_bodyMap)
497     numBods = N^2+4*(N-1)*(M-1);
498     numSprings = 2*N*(N-1)+8*(N-1)*(M-1);
499     springCnt = zeros(numSprings,2);
500     springMap = zeros(numBods,numSprings);
501
502     structural_bodyMap = [ (1:size(structural_bodyMap,1))' , structural_bodyMap];
503     cnt=1;
504
505     while cnt<numSprings
506         for i=1:numBods
507          a = unique(structural_bodyMap(i,2:size(structural_bodyMap,2)));
508          a(a==0)=[];
509
510          for j=1:length(a)
511              springCnt(cnt,1)=structural_bodyMap(i,1);
512              springCnt(cnt,2)=a(j);
```

```
            cnt=cnt+1;
        end
       end
    end

    i=1;
    while i<=size(springCnt,1)
        body1 = springCnt(i,1);
        body2 = springCnt(i,2);

        j=1;
        while j<size(springCnt,1)
            if springCnt(j,1)== body2 &&  springCnt(j,2)==body1
                springCnt(j,:) = [];
            end
            j=j+1;
        end
        i=i+1;
    end

    for i=1:numBods
        for j=1:size(springCnt,1)
            if springCnt(j,1)==i
               springMap(i,j) = 1;
            end
            if springCnt(j,2)==i
               springMap(i,j) = -1;
            end
        end
    end
    springMap = sparse(springMap);
end

%-------------------------------------------------------------------------

function [springMap, springCnt] =         springMapBuilder_Shear(N, M, shear_bodyMap)
    numBods = N^2+4*(N-1)*(M-1);
    numSprings = 2*((N-1)^2 + 4*(M-1)*(N-1));
    springCnt = zeros(numSprings,2);
    springMap = zeros(numBods,numSprings);

    shear_bodyMap = [ (1:size(shear_bodyMap,1))' , shear_bodyMap];
    cnt=1;

    while cnt<numSprings
       for i=1:numBods
        a = unique(shear_bodyMap(i,2:size(shear_bodyMap,2)));
        a(a==0)=[];

        for j=1:length(a)
            springCnt(cnt,1)=shear_bodyMap(i,1);
            springCnt(cnt,2)=a(j);
            cnt=cnt+1;
        end
       end
    end

    i=1;
    while i<=size(springCnt,1)
        body1 = springCnt(i,1);
        body2 = springCnt(i,2);

        j=1;
        while j<size(springCnt,1)
            if springCnt(j,1)== body2 &&  springCnt(j,2)==body1
                springCnt(j,:) = [];
            end
            j=j+1;
        end
        i=i+1;
    end

    for i=1:numBods
        for j=1:size(springCnt,1)
            if springCnt(j,1)==i
               springMap(i,j) = 1;
            end
            if springCnt(j,2)==i
               springMap(i,j) = -1;
            end
        end
    end
    springMap = sparse(springMap);
end

%-------------------------------------------------------------------------
```

```
function [springMap, springCnt] =          springMapBuilder_Bend(N, M, Bend_bodyMap)
    numBods = N^2+4*(N-1)*(M-1);
    numSprings = 2*(N)*(N-2) + 4*((N-2)*M + (M-2)*N);
    springCnt = zeros(numSprings,2);


    Bend_bodyMap = [ (1:size(Bend_bodyMap,1))' , Bend_bodyMap];
    cnt=1;

    while cnt<numSprings
       for i=1:numBods
        a = unique(Bend_bodyMap(i,2:size(Bend_bodyMap,2)));
        a(a==0)=[];

        for j=1:length(a)
            springCnt(cnt,1)=Bend_bodyMap(i,1);
            springCnt(cnt,2)=a(j);
            cnt=cnt+1;
        end
       end
    end

    i=1;
    while i<=size(springCnt,1)
        body1 = springCnt(i,1);
        body2 = springCnt(i,2);

        j=1;
        while j<size(springCnt,1)
            if springCnt(j,1)== body2 &&  springCnt(j,2)==body1
                springCnt(j,:) = [];
            end
            j=j+1;
        end
        i=i+1;
    end

    springMap = zeros(numBods,size(springCnt,1));

    for i=1:numBods
        for j=1:size(springCnt,1)
            if springCnt(j,1)==i
               springMap(i,j) = 1;
            end
            if springCnt(j,2)==i
               springMap(i,j) = -1;
            end
        end
    end
    springMap = sparse(springMap);
end

%-------------------------------------------------------------------------

function [springMap_RI, springCntRI] =   springMapBuilder_Ridge(N, M, ridge_bodyMap)
    numBods = N^2+4*(N-1)*(M-1);

    springCntRI = ridge_bodyMap;
    springMap_RI = zeros(numBods,1);
    for i=1:length(ridge_bodyMap)
        springMap_RI(ridge_bodyMap(i,1),i) = 1;
        springMap_RI(ridge_bodyMap(i,2),i) = -1;
    end
    springMap_RI = sparse(springMap_RI);
end

%-------------------------------------------------------------------------

function [FT, F1, F2, F3, F4] =     buildBedBodyMeshes(N,M)
    FT = zeros(N+2);
    F1 = zeros(N+2,M+2);
    F2 = zeros(N+2,M+2);
    F3 = zeros(N+2,M+2);
    F4 = zeros(N+2,M+2);

    topFaceCount = 1:N^2;
    FT(2:N+1,2:N+1) = reshape(topFaceCount,N,N)';

    F1(2,2:N+1) = flip(FT(2:N+1,2))';
    faceOneCount = (1+N^2):(N^2 + N*(M-1));
    F1(3:M+1,2:N+1) = reshape(faceOneCount,M-1,[]);

    F2(2,2:N+1) = (FT(2,2:N+1))';
    a=(1+ N^2 + (N-1)*(M-1));
    b=(N^2 + (N-1)*(M-1) + N*(M-1));
    faceTwoCount = a:b;
    F2(3:M+1,2:N+1) = reshape(faceTwoCount,M-1,[]);
```

```
687
688         F3(2,2:N+1) = flip(FT(2:N+1,N+1))';
689         a=(1+ N^2 + 2*(N-1)*(M-1));
690         b=(N^2 + 2*(N-1)*(M-1) + N*(M-1));
691         faceThreeCount = a:b;
692         F3(3:M+1,2:N+1) = flip(reshape(faceThreeCount,M-1,[]),2);
693
694         F4(2,2:N+1) = (FT(N+1,2:N+1))';
695         a=(1+ N^2 + 3*(N-1)*(M-1));
696         b=(N^2 + 4*(N-1)*(M-1));
697         faceFourCount = a:b;
698         F4(3:M+1,3:N+1) = flip(reshape(faceFourCount,M-1,[]),2);
699         F4(2:M+1,2) = F1(2:M+1,2);
700     end
701
702     %-------------------------------------------------------------------------
703
704     function [ x_FT,y_FT,z_FT, x_F1,y_F1,z_F1, x_F2,y_F2,z_F2, x_F3,y_F3,z_F3, x_F4,y_F4,z_F4] =
          buildBedCoordMesh_Init(N,M, xLength, yLength, zLength)
705         %TopFace
706         x = -xLength/2:xLength/(N-1):xLength/2;
707         y = flip(-yLength/2:yLength/(M-1):yLength/2);
708         [x_FT, y_FT, z_FT] = meshgrid(x,y,zLength);
709
710         %Face One
711         x = -xLength/2;
712         y = (-yLength/2:yLength/(M-1):yLength/2);
713         z = flip(0:zLength/(M-1):zLength);
714         [y_F1, z_F1, x_F1] = meshgrid(y,z,x);
715
716         %Face Two
717         x = (-xLength/2:xLength/(N-1):xLength/2);
718         y = (yLength/2);
719         z = flip(0:zLength/(M-1):zLength);
720         [x_F2, z_F2, y_F2] = meshgrid(x,z,y);
721
722         %Face Three
723         x = xLength/2;
724         y = (-yLength/2:yLength/(M-1):yLength/2);
725         z = flip(0:zLength/(M-1):zLength);
726         [y_F3, z_F3, x_F3] = meshgrid(y,z,x);
727
728         %Face Four
729         x = (-xLength/2:xLength/(N-1):xLength/2);
730         y = (-yLength/2);
731         z = flip(0:zLength/(M-1):zLength);
732         [x_F4, z_F4, y_F4] = meshgrid(x,z,y);
733
734     end
735
736     %-------------------------------------------------------------------------
737
738     function Q = qBuild(N,M,x_FT,y_FT,z_FT, x_F1,y_F1,z_F1, x_F2,y_F2,z_F2, x_F3,y_F3,z_F3, x_F4,y_F4,
          z_F4, eCheck)
739         x = [ reshape(x_FT',N^2,[]) ; reshape(x_F1(2:M,1:N-1),(N-1)*(M-1),[]) ; reshape(x_F2(2:M,1:N
              -1),(N-1)*(M-1),[]) ; reshape(flip(x_F3(2:M,2:N),2),(N-1)*(M-1),[]) ; reshape(flip(x_F4(2:
              M,2:N),2),(N-1)*(M-1),[]) ];
740         y = [ reshape(y_FT',N^2,[]) ; reshape(y_F1(2:M,1:N-1),(N-1)*(M-1),[]) ; reshape(y_F2(2:M,1:N
              -1),(N-1)*(M-1),[]) ; reshape(flip(y_F3(2:M,2:N),2),(N-1)*(M-1),[]) ; reshape(flip(y_F4(2:
              M,2:N),2),(N-1)*(M-1),[]) ];
741         z = [ reshape(z_FT',N^2,[]) ; reshape(z_F1(2:M,1:N-1),(N-1)*(M-1),[]) ; reshape(z_F2(2:M,1:N
              -1),(N-1)*(M-1),[]) ; reshape(flip(z_F3(2:M,2:N),2),(N-1)*(M-1),[]) ; reshape(flip(z_F4(2:
              M,2:N),2),(N-1)*(M-1),[]) ];
742         Dx = zeros((N^2 + 4*(N-1)*(M-1)),1);
743         Dy = zeros((N^2 + 4*(N-1)*(M-1)),1);
744         Dz = zeros((N^2 + 4*(N-1)*(M-1)),1);
745
746         Q = [x;y;z;Dx;Dy;Dz;eCheck];
747     end
748
749     %-------------------------------------------------------------------------
750
751     function [ x, y, z, Dx, Dy, Dz, eCheck ] =       buildBedCoords(N,M,Q)
752         numBod = (N^2 + 4*(N-1)*(M-1));
753
754         xpop = 1;
755         ypop = numBod;
756         x = Q(    xpop   :   ypop    );
757
758         xpop = 1+1*numBod;
759         ypop = 2*numBod;
760         y = Q(    xpop   :   ypop    );
761
762         xpop = 1+2*numBod;
763         ypop = 3*numBod;
764         z = Q(    xpop   :   ypop    );
765
```

```
766     xpop = 1+3*numBod;
767     ypop = 4*numBod;
768     Dx= Q(    xpop   :   ypop    );
769
770     xpop = 1+4*numBod;
771     ypop = 5*numBod;
772     Dy= Q(    xpop   :   ypop    );
773
774     xpop = 1+5*numBod;
775     ypop = 6*numBod;
776     Dz= Q(    xpop   :   ypop    );
777
778     eCheck = Q(ypop+1);
779 end
780
781 %-------------------------------------------------------------------------
782
783 function [pVector, pressure, Vol] = calcPressure2(x,y,z,pressure,area,Vo,Con)
784     P = [x,y,z];
785     [~,Vol]=convhulln(P);
786     TR = triangulation(Con,P);
787     V = vertexNormal(TR);
788
789     deltaV = Vo/Vol;
790     pressure = pressure*deltaV;
791     pForce = pressure*area;
792
793     quiver3(x,y,z,V(:,1),V(:,2),V(:,3))
794
795     pVector = reshape(V',size(V,1)*size(V,2),[]);
796     pVector = pVector*pForce;
797
798 end
799
800 %-------------------------------------------------------------------------
801
802 function [magVector, xVector, yVector, zVector, stretch, stretchP] =        vectorBuilder(Lo,
      springCnt,x,y,z,Dx,Dy,Dz)
803     length=size(springCnt,1);
804     xVector = zeros(length,1);
805     yVector = zeros(length,1);
806     zVector = zeros(length,1);
807     xVelVec = zeros(length,1);
808     yVelVec = zeros(length,1);
809     zVelVec = zeros(length,1);
810
811     for i=1:length
812        xVector(i,1) = x(springCnt(i,2)) - x(springCnt(i,1));
813        yVector(i,1) = y(springCnt(i,2)) - y(springCnt(i,1));
814        zVector(i,1) = z(springCnt(i,2)) - z(springCnt(i,1));
815        xVelVec(i,1) = Dx(springCnt(i,2)) - Dx(springCnt(i,1));
816        yVelVec(i,1) = Dy(springCnt(i,2)) - Dy(springCnt(i,1));
817        zVelVec(i,1) = Dz(springCnt(i,2)) - Dz(springCnt(i,1));
818     end
819
820     magVector = vecnorm([xVector yVector zVector],2,2);
821     stretch = magVector-Lo;
822     stretchP = (xVector.*xVelVec + yVector.*yVelVec + zVector.*zVelVec)./magVector;
823 end
824
825 %-------------------------------------------------------------------------
826
827 function Lo = lengthBuilder(springCnt,N,M,Q)
828      num =  N^2+4*(N-1)*(M-1);
829     x = Q(1:num);
830     y = Q(1+1*num:2*num);
831     z = Q(1+2*num:3*num);
832     length=size(springCnt,1);
833
834     xVector = zeros(length,1);
835     yVector = zeros(length,1);
836     zVector = zeros(length,1);
837
838     for i=1:length
839        xVector(i,1) = x(springCnt(i,2)) - x(springCnt(i,1));
840        yVector(i,1) = y(springCnt(i,2)) - y(springCnt(i,1));
841        zVector(i,1) = z(springCnt(i,2)) - z(springCnt(i,1));
842     end
843     Lo = sqrt(xVector.^2 + yVector.^2 + zVector.^2);
844 end
845
846 %-------------------------------------------------------------------------
847
848 function eCheck = NiCheck(N,M,Dx, Dy, Dz, RHS)
849     velVect = reshape([Dx, Dy, Dz]',3*length(Dx),1);
850     [velVect] = filterConstrainedBodies(N,M,velVect);
851     eCheck = sum(RHS.*velVect);
```

```
852 end
853
854 %-------------------------------------------------------------------------
855
856 function [Sh1, Sh2] =       springForceBuilder(~,~, xVector, yVector, zVector, magVector,
      springMapMini1, stretch, stretchP)
857
858     xMat1 = sparse(springMapMini1*diag(xVector./magVector));
859     yMat1 = sparse(springMapMini1*diag(yVector./magVector));
860     zMat1 = sparse(springMapMini1*diag(zVector./magVector));
861
862     springMapShear = reshape( [xMat1, yMat1, zMat1]' , length(xVector) , [] )';
863     Sh1 = sparse(springMapShear*diag(stretch));
864     Sh2 = sparse(springMapShear*diag(stretchP));
865
866 end
867
868 %-------------------------------------------------------------------------
869
870 function [RHS] =     miscCalcs(St1, St2, dampSt, KSt, Sh1, Sh2, dampSh, KSh, Sb1, Sb2, dampBe, KBe,
       Se1, Se2, dampRI, KRi, gVector, pVector)
871     RHS = -St1*KSt -St2*dampSt -Sh1*KSh -Sh2*dampSh -Sb1*KBe -Sb2*dampBe -Se1*KRi -Se2*dampRI -
          pVector -gVector;
872 end
873
874 %-------------------------------------------------------------------------
875
876 function SolutionTo =   inverseARHS(A,RHS)
877     SolutionTo = A\RHS;
878 end
879
880 %-------------------------------------------------------------------------
881
882 function DQ =    solPosFun(x,y,z,Dx,Dy,Dz,SolutionTo,N,M,eCheck, varQ, qList)
883     SOLpos = reshape(SolutionTo, 3, []) ';
884
885     varQ = reshape(varQ(1:length(varQ)-1),[],6);
886     varQ(:,1:3) = varQ(:,4:6);
887     varQ(:,4:6) = SOLpos;
888      %       Dx        Dy       Dz       Ax       Ay       Az%
889     DQ = [reshape(varQ,[],1) ; eCheck];
890
891 end
```

After a simulation is built and ran under the runModel program, the simulation can be quickly loaded in its entirety using this program. The input requires the parameters defined in the simulation to be entered as the filename and the correct filepath specified for that specific simulation to be loaded into the MATLAB workspace. All plotting and evaluation programs present should not clear the workspace of the instantiated variables or their values.

Listing C.2: Load Simulations and Pertinent Data: loadPreviousSimulations

```
1  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2  %                                LOAD PROGRAMS                                %
3  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
4
5  clear
6  clc
7
8  N = 10;              %Enter Resolution in length
9  M = 10;              %Enter Resolution in width
10 pressure = 500;      %Enter Pressure
11 TFINAL = 70;         %Enter Simulation Length
12 K = 1000;            %Enter Spring Constants Present in File
13
14
15 directory = strcat(num2str(N),'x',num2str(M),'_press_',num2str(pressure),'_t',num2str(TFINAL),'_k'
       ,num2str(K));
16
17 fLoad = strcat("FileLocation\",directory,".mat");
18 fSaveName1 = strcat("FileLocation\",directory,".txt");
19
20 params = load(fLoad);
```

```

LoSt = params.LoSt;
LoSh = params.LoSh;
LoBe = params.LoBe;
LoRi = params.LoRi;
pressure = params.pressure;
mass = params.mass;
KSh = params.KSh;
KSt = params.KSt;
KBe = params.KBe;
KRi = params.KRi;
dampSt = params.dampSt;
dampSh = params.dampSh;
dampRI = params.dampRI;
dampBe = params.dampBe;
springCntST = params.springCntST;
springCntSH = params.springCntSH;
springCntRI = params.springCntRI;
springCntBE = params.springCntBE;
springMap_ST = full(params.springMap_ST);
springMap_SH = full(params.springMap_SH);
springMap_RI = full(params.springMap_RI);
springMap_BE = full(params.springMap_BE);

FID = fopen(fSaveName1,'r');
numBods = (N^2 + 4*(N-1)*(M-1))*6+3;
lengthQ = TFINAL*10;%- 1;
formatSpec = '%f';
Q0 = fscanf(FID,formatSpec);
Q = reshape(Q0,numBods,[])';
```

The energy check program is where the remaining portion of the Work-Energy Theorem is presented. The NiCheck function within the runModel program only calculates and stores the work relating to the generalized speeds. The remaining portion of the theorem in this program takes the difference between the work related to the generalized speeds and the total kinetic energy at each time step. The total energy is then plotted over the time of the loaded simulation.

Listing C.3: Plot the Simulation's Total Energy Over Time: energyCheck

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                              RUN ENERGY CHECK                               %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clf

[t, TotalE, SpringE, KinE, GravPot, workDamp, eCheck] = checkEnergy(mass, KSh, KSt, KBe, KRi, Q, N
    , M, LoSt, LoSh, LoBe, LoRi, springCntST, springCntSH, springCntBE, springCntRI, dampSt,
    dampSh, dampRI, dampBe);

plot(t/10,eCheck+KinE)
title("Energy Check")
xlabel("Time [s]")
ylabel("Total Energy [J]")

grid on
grid minor

%------------------------------------------------------------------------------

function [t, TotalE, SpringE, KinE, GravPot, workDamp, eCheck] = checkEnergy(mass, KSh, KSt, KBe,
    KRi, Q, N, M, LoSt, LoSh, LoBe, LoRi, springCntST, springCntSH, springCntBE, springCntRI,
    dampSt, dampSh, dampRI, dampBe)
   numBods = N^2 + 4*(N-1)*(M-1);
   t=1:size(Q,1);
   Qt=zeros(numBods,6,size(Q,1));

   for i=1:max(t)
        Qt(:,:,i) = reshape(Q(i,3:size(Q,2)-1),numBods,[]);
        [~, ~, ~, ~, stretch_ST(:,i), stretch_STP(:,i)] = vectorBuilder(LoSt, springCntST, Qt(:,1,
            i),Qt(:,2,i),Qt(:,3,i),Qt(:,4,i),Qt(:,5,i),Qt(:,6,i));
        [~, ~, ~, ~, stretch_SH(:,i), stretch_SHP(:,i)] = vectorBuilder(LoSh, springCntSH, Qt(:,1,
            i),Qt(:,2,i),Qt(:,3,i),Qt(:,4,i),Qt(:,5,i),Qt(:,6,i));
        [~, ~, ~, ~, stretch_BE(:,i), stretch_BEP(:,i)] = vectorBuilder(LoBe, springCntBE, Qt(:,1,
            i),Qt(:,2,i),Qt(:,3,i),Qt(:,4,i),Qt(:,5,i),Qt(:,6,i));
```

```
        [~, ~, ~, ~, stretch_RI(:,i), stretch_RIP(:,i)] = vectorBuilder(LoRi, springCntRI, Qt(:,1,
            i),Qt(:,2,i),Qt(:,3,i),Qt(:,4,i),Qt(:,5,i),Qt(:,6,i));

        Te(:,i) = 0.5*mass*( Qt(:,4,i).^2 + Qt(:,5,i).^2 + Qt(:,6,i).^2 );



        Vg(:,i) = 9.81*mass*(Qt(:,3,i)-Qt(:,3,1));
        Vs1(:,i) = 0.5*( KSh.*(stretch_SH(:,i).^2) );
        Vs2(:,i) = 0.5*( KSt.*(stretch_ST(:,i).^2) );
        Vs3(:,i) = 0.5*( KBe.*(stretch_BE(:,i).^2) );
        Vs4(:,i) = 0.5*( KRi.*(stretch_RI(:,i).^2) );

        Ds1(:,i) = -( dampSh.*(stretch_SHP(:,i)) );
        Ds2(:,i) = -( dampSt.*(stretch_STP(:,i)) );
        Ds3(:,i) = -( dampBe.*(stretch_BEP(:,i)) );
        Ds4(:,i) = -( dampRI.*(stretch_RIP(:,i)) );

        SpringE(i) = sum(Vs1(:,i)) + sum(Vs2(:,i)) + sum(Vs3(:,i)) + sum(Vs4(:,i));
        GravPot(i) = sum(Vg(:,i));
        KinE(i) = sum(Te(:,i));
        workDamp(i) = sum(Ds1(:,i)) + sum(Ds2(:,i)) + sum(Ds3(:,i)) + sum(Ds4(:,i));
        TotalE(i) =  SpringE(i) + GravPot(i) + KinE(i);
        eCheck(i) = Q(i,size(Q,2));
    end
end

%-------------------------------------------------------------------------

function [magVector, xVector, yVector, zVector, stretch, stretchP] =        vectorBuilder(Lo,
     springCnt,x,y,z,Dx,Dy,Dz)
    length=size(springCnt,1);
    xVector = zeros(length,1);
    yVector = zeros(length,1);
    zVector = zeros(length,1);
    xVelVec = zeros(length,1);
    yVelVec = zeros(length,1);
    zVelVec = zeros(length,1);

    for i=1:length
        xVector(i,1) = x(springCnt(i,2)) - x(springCnt(i,1));
        yVector(i,1) = y(springCnt(i,2)) - y(springCnt(i,1));
        zVector(i,1) = z(springCnt(i,2)) - z(springCnt(i,1));
        xVelVec(i,1) = Dx(springCnt(i,2)) - Dx(springCnt(i,1));
        yVelVec(i,1) = Dy(springCnt(i,2)) - Dy(springCnt(i,1));
        zVelVec(i,1) = Dz(springCnt(i,2)) - Dz(springCnt(i,1));
    end

    magVector = vecnorm([xVector yVector zVector],2,2);

    stretch = magVector-Lo;
    %calculate initial Lo for each spring prior to running code
    stretchP = (xVector.*xVelVec + yVector.*yVelVec + zVector.*zVelVec)./magVector;
end
```

Once simulations are loaded using the loadPreviousSimulations program, the animation can be viewed using this plotting program. Various sections within this program allow for different viewing methods but will require other sections to be commented out prior to running.

Listing C.4: Plot Simulation Results: plottingMethods

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                PLOT PREVIOUS MODELS                                %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Plot simulation for gif creation
clf

timeRun=11;     %Enter length of plot

TimeOdeArray = size(Q,1);

fig = figure;
for i=1:1:timeRun
    Qrelevant = Q(:,3:size(Q,2))';
    time = Q(:,1);
    pressure = Q(:,2);
```

```
    X = Qrelevant(                          1:1*(N^2+4*(N-1)*(M-1)),i);
    Y = Qrelevant(1*(N^2+4*(N-1)*(M-1)) +1:2*(N^2+4*(N-1)*(M-1)),i);
    Z = Qrelevant(2*(N^2+4*(N-1)*(M-1)) +1:3*(N^2+4*(N-1)*(M-1)),i);

    Qrelevant = [X;Y;Z];
    [ x_FT,y_FT,z_FT, x_F1,y_F1,z_F1, x_F2,y_F2,z_F2, x_F3,y_F3,z_F3, x_F4,y_F4,z_F4] =
        buildBedCoordMesh(N,M,Qrelevant);

    shp = alphaShape(X,Y,Z,0.5);
%     plot(shp)
    f = @(i) plot3(X,Y,Z,'ro');
    annotation('textbox', [0, 0.5, 0, 0], 'string', sprintf('Time: %0.1f\nPressure: %0.1f',time(i)
        ,0));
    xlim([-0.329 0.198])
    ylim([-0.241 0.286])
    zlim([-0.043 0.274])
    view([67.952 31.712])
    grid on
    axis equal
    axis([-0.25 0.25 -0.25 0.25 0 0.3])
    drawnow
    hold off
    fanimator(f)
    pause(0.05)
    frame = getframe(fig);
    im{i} = frame2im(frame);

end
close;

filename = 'fileName.gif'; % Specify the output file name
for idx = 1:timeRun
    [A,map] = rgb2ind(im{idx},256);
    if idx == 1
        imwrite(A,map,filename,'gif','LoopCount',Inf,'DelayTime',1);
    else
        imwrite(A,map,filename,'gif','WriteMode','append','DelayTime',1);
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Plot Springs

clf


stVal = 700;
[magVector, xVector, yVector, zVector, stretch, stretchP] = getSnapShotSpringProperties(N,M,LoSt,
    springCntST, Q(stVal,3:size(Q,2))');
    Qrelevant = Q(:,3:size(Q,2))';
    X = Qrelevant(                          1:1*(N^2+4*(N-1)*(M-1)),stVal);
    Y = Qrelevant(1*(N^2+4*(N-1)*(M-1)) +1:2*(N^2+4*(N-1)*(M-1)),stVal);
    Z = Qrelevant(2*(N^2+4*(N-1)*(M-1)) +1:3*(N^2+4*(N-1)*(M-1)),stVal);

    X0 = Qrelevant(                          1:1*(N^2+4*(N-1)*(M-1)),1);
    Y0 = Qrelevant(1*(N^2+4*(N-1)*(M-1)) +1:2*(N^2+4*(N-1)*(M-1)),1);
    Z0 = Qrelevant(2*(N^2+4*(N-1)*(M-1)) +1:3*(N^2+4*(N-1)*(M-1)),1);

    Qrelevant = [X;Y;Z];

    for i=1:size(xVector,1)
       Xp = [X(springCntST(i,2)), X(springCntST(i,1))];
       Yp = [Y(springCntST(i,2)), Y(springCntST(i,1))];
       Zp = [Z(springCntST(i,2)), Z(springCntST(i,1))];
       if (stretch(i)<.005*LoSt(i) && stretch(i)>0)
        plot3(Xp,Yp,Zp,'Color','c')
       elseif (stretch(i)>.005*LoSt(i) && stretch(i)<.02*LoSt(i))
        plot3(Xp,Yp,Zp,'Color','r')
       elseif (stretch(i)>.02*LoSt(i))
        plot3(Xp,Yp,Zp,'Color','g')
       elseif (stretch(i)<0)
        plot3(Xp,Yp,Zp,'Color','b')
       else
        shp = alphaShape(X,Y,Z,0.5);
        %plot(shp)
        plot3(Xp,Yp,Zp,'k')
       end
       hold on
       axis equal
       view([135 45])
    end
    area=1;
    Vo=1;
    DT = delaunayTriangulation(X0,Y0,Z0);
```

```
    Con = freeBoundary(DT);
    [pVector, pressure, Vol] = calcPressure2(X,Y,Z,pressure,area,Vo,Con)


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                              PLOT top Surface                                %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clf

i=101;

Qrelevant = Q(:,3:size(Q,2))';
time = Q(:,1);
pressure = Q(:,2);
X = Qrelevant(                          1:1*(N^2+4*(N-1)*(M-1)),i);
Y = Qrelevant(1*(N^2+4*(N-1)*(M-1)) +1:2*(N^2+4*(N-1)*(M-1)),i);
Z = Qrelevant(2*(N^2+4*(N-1)*(M-1)) +1:3*(N^2+4*(N-1)*(M-1)),i);

xTop = reshape(X(1:N*M),N,M);
yTop = reshape(Y(1:N*M),N,M);
zTop = reshape(Z(1:N*M),N,M)-.123647;

s=surf(xTop, yTop, zTop,'FaceColor','none','EdgeColor','r');
axis equal
grid off
hold on
xlim([-0.15 0.15])
ylim([-0.15 0.15])
zlim([-0.03 0.05])      %zlim([-0.01 0.05])
view([0 90])            %view([45 30])



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                          Overlay Data and SythData                           %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear
clc
clf

N = 10;
M = 10;
pressure = 500;
TFINAL = 70;

Kcount = 4;  %1-6
KDirectory = [400 600 800 1000 1200 1400];

K = KDirectory(Kcount);

directory = strcat(num2str(N),'x',num2str(M),'_press_',num2str(pressure),'_t',num2str(TFINAL),'_k'
    ,num2str(K));

fLoad = strcat("FileLocation\",directory,".mat");
fSaveName1 = strcat("FileLocation\",directory,".txt");

params = load(fLoad);
pressure = params.pressure;
mass = params.mass;

FID = fopen(fSaveName1,'r');
numBods = (N^2 + 4*(N-1)*(M-1))*6+3;
lengthQ = TFINAL*10;%- 1;
formatSpec = '%f';
Q0 = fscanf(FID,formatSpec);
Qmodel = reshape(Q0,numBods,[])';

name = "kpa_132_0204";
i=1;

fname3 = strcat(name,"-",num2str(i));
fid3 = strcat("FileLocation\",fname3,'.mat');
load(fid3)
xMat = xMat/.4*.28  + 0.05056;
yMat = yMat/.4*.28  - 0.015081;
zMat = zMat + 0.538;

sampleTime=699;
Qrelevant = Qmodel(:,3:size(Qmodel,2))';
time = Qmodel(:,1);
pressure = Qmodel(:,2);
X = Qrelevant(                          1:1*(N^2+4*(N-1)*(M-1)),sampleTime);
Y = Qrelevant(1*(N^2+4*(N-1)*(M-1)) +1:2*(N^2+4*(N-1)*(M-1)),sampleTime);
Z = Qrelevant(2*(N^2+4*(N-1)*(M-1)) +1:3*(N^2+4*(N-1)*(M-1)),sampleTime);
```

```
187
188 xTop = reshape(X(1:N*M),N,M);
189 yTop = reshape(Y(1:N*M),N,M);
190 zTop = reshape(Z(1:N*M),N,M)-.123647;
191 plot3(xMat,yMat,zMat,'r.')
192 hold on
193 axis equal
194 s=surf(xTop, yTop, zTop,'FaceColor','flat','EdgeColor','r');
195 axis equal
196
197 %-------------------------------------------------------------------------
198
199 function [magVector, xVector, yVector, zVector, stretch, stretchP] = getSnapShotSpringProperties(N
      ,M,LoSt, springCntST, Q)
200     [ x, y, z, Dx, Dy, Dz, ~] = buildBedCoords(N,M,Q);
201     [magVector, xVector, yVector, zVector, stretch, stretchP] = vectorBuilder(LoSt, springCntST, x
          ,y,z,Dx,Dy,Dz);
202 end
203
204 %-------------------------------------------------------------------------
205
206 function [ x, y, z, Dx, Dy, Dz, eCheck ] =      buildBedCoords(N,M,Q)
207     numBod = (N^2 + 4*(N-1)*(M-1));
208
209     xpop = 1;
210     ypop = numBod;
211     x = Q(   xpop  :  ypop   );
212
213     xpop = 1+1*numBod;
214     ypop = 2*numBod;
215     y = Q(   xpop  :  ypop   );
216
217     xpop = 1+2*numBod;
218     ypop = 3*numBod;
219     z = Q(   xpop  :  ypop   );
220
221     xpop = 1+3*numBod;
222     ypop = 4*numBod;
223     Dx= Q(   xpop  :  ypop   );
224
225     xpop = 1+4*numBod;
226     ypop = 5*numBod;
227     Dy= Q(   xpop  :  ypop   );
228
229     xpop = 1+5*numBod;
230     ypop = 6*numBod;
231     Dz= Q(   xpop  :  ypop   );
232
233 %     eCheck = 0;
234     eCheck = Q(ypop+1);
235 end
236
237 %-------------------------------------------------------------------------
238
239 function [magVector, xVector, yVector, zVector, stretch, stretchP] =       vectorBuilder(Lo,
      springCnt,x,y,z,Dx,Dy,Dz)
240     length=size(springCnt,1);
241     xVector = zeros(length,1);
242     yVector = zeros(length,1);
243     zVector = zeros(length,1);
244     xVelVec = zeros(length,1);
245     yVelVec = zeros(length,1);
246     zVelVec = zeros(length,1);
247
248     for i=1:length
249        xVector(i,1) = x(springCnt(i,2)) - x(springCnt(i,1));
250        yVector(i,1) = y(springCnt(i,2)) - y(springCnt(i,1));
251        zVector(i,1) = z(springCnt(i,2)) - z(springCnt(i,1));
252        xVelVec(i,1) = Dx(springCnt(i,2)) - Dx(springCnt(i,1));
253        yVelVec(i,1) = Dy(springCnt(i,2)) - Dy(springCnt(i,1));
254        zVelVec(i,1) = Dz(springCnt(i,2)) - Dz(springCnt(i,1));
255     end
256
257     magVector = vecnorm([xVector yVector zVector],2,2);
258
259     stretch = magVector-Lo;
260     %calculate initial Lo for each spring prior to running code
261     stretchP = (xVector.*xVelVec + yVector.*yVelVec + zVector.*zVelVec)./magVector;
262 end
263
264 %-------------------------------------------------------------------------
```

This program, DOTSVgit along with the necessary Intel Realsense SDK, allows for a PC running MATLAB to connect over serial port to a Realsense Camera. It

performs the image processing necessary to extract the positions of each marking in space relative to the camera. The input parameters required by the user are provided at the beginning of the program.

Listing C.5: Intel RealSense Matlab Image Processing: DOTSVgit

```
%Coded by Cristian Almendariz, Adapted from Ayush Thapa & Intel GitHub
%https://github.com/IntelRealSense/librealsense/tree/master/wrappers/matlab
%This function returns the spatial coordinates of the markings on the air
%bladder.
%Final version 7/1/2020

%inputs
skipFrames = 50; %Discards first XX number of frames taken from the intel realsense to allow for
     the camera to properly adjust to lighting.
threshold = 160;
x = [203  , 530  , 533, 205];
y = [41   , 32   , 363, 360];
fileName = 'kpa-110-0121-1.txt';



val = true;
while(val)
    num = DOTS_Vgiter(skipFrames, threshold, x, y, fileName);
    if num == 1024
        val = false;
    end
    val = false;
end

function num = DOTS_Vgiter(skipFrames, threshold, x, y, fileName)
    % Make Pipeline object to manage streaming
    pipe = realsense.pipeline();

    % Start streaming on an arbitrary camera with default settings
    profile = pipe.start();
    align_to = realsense.stream.color;
    alignedFs = realsense.align(align_to);

    % Get streaming device's name
    dev = profile.get_device();
    name = dev.get_info(realsense.camera_info.name);

    % Get frames. We discard the first couple to allow
    % the camera time to settle
    %while(true)
    for i = 1:skipFrames
        fs = pipe.wait_for_frames();
    end

    % Select depth frame
    %depth = fs.get_depth_frame();

    %align the depth frames to the color stream
    aligned_frames = alignedFs.process(fs);
    depth = aligned_frames.get_depth_frame();

    % get depth image parameters
    depthSensor = dev.first('depth_sensor');
    depthScale = depthSensor.get_depth_scale();
    depthWidth = depth.get_width();
    depthHeight = depth.get_height();

    % retrieve UINT16 depth vector
    depthVector = depth.get_data();

    % reshape vector, and scale to depth in meters
    depthMap = double(transpose(reshape(depthVector, [depthWidth,depthHeight]))) .* depthScale;



    %Select rgb frame
    color = fs.get_color_frame();
    colordata=color.get_data();
    img = permute(reshape(colordata',[3,color.get_width(),color.get_height()]),[3 2 1]);
    %subplot(2,1,1);
    %imshow(depthMap)
    %impixelinfo;
    %title('Depth Map (m)');
    %subplot(2,1,2);
    %imshow(img)
```

```
    %impixelinfo;
    %title('Color Map');

    %end
    % Stop streaming
    pipe.stop();

    % original method for determining XY pixel positions
    %[centers,radii] = imfindcircles(img,[3 5],'ObjectPolarity','dark','Sensitivity',0.98,'Method','twostage');

    % replacement of imfindcircles, locate XY pose, return 'centers' and
    % 'radii'

    %[centers,radii] =

    img2 = img;
    grayImg = rgb2gray(img2);
    %grayImg = imgaussfilt(grayImg,1);

    grayImg = grayImg < threshold;
    imshow(grayImg);
    %grayImg= bwconncomp(double(grayImg));

    BW = double(grayImg);
    CC = bwconncomp(BW);
    BW2 = poly2mask(x,y,480,640);

    BW2 = BW2.*BW;
    figure(1)
    imshow(BW2);

%     return;
%      numPixels = cellfun(@numel,CC.PixelIdxList);
%
%      sorted = sort(numPixels);
%
%      for i=1:3
%          idx = find(numPixels==sorted(i));
%          BW(CC.PixelIdxList{idx}) = 0;
%
%      end
%
%     figure
%     imshow(BW)
%
%     %fprintf('hello');
%     return;
%     %figure(1)

    [centers,radii] = imfindcircles(BW2,[3 5],'ObjectPolarity','bright','Sensitivity',0.98,'Method','twostage');
    viscircles(centers,radii);
    title(sprintf("dots on the bed %s", name));

%     return;


    cfg = realsense.config();
    cfg.enable_stream(realsense.stream.depth);
    pipe = realsense.pipeline();
    profile = pipe.start(cfg);
    depth_stream = profile.get_stream(realsense.stream.depth);
    depth_stream = depth_stream.as('video_stream_profile');
    intrinsics1 = depth_stream.get_intrinsics();



    %getting X and Y co-ordinates
    %depthpoint must be in loop for multiple circles

    for i=1:length(radii)
        %depth_point =rs.rs2_deproject_pixel_to_point(intrinsics1, [centers(i,1),centers(i,2)], PixDist(i));
        PixDist = depth.get_distance(round(centers(i,1)),round(centers(i,2)));
        x=((centers(i,1))-intrinsics1.ppx)/intrinsics1.fx;
        y=((centers(i,2))-intrinsics1.ppy)/intrinsics1.fy;
        X(i)=PixDist*x;
        Y(i)=PixDist*y;
        Z(i)=-PixDist;
    end
    figure(2)
    plot3(X,Y,Z,'r.')
    axis([-0.3 0.2 -0.5 0.5 -1 1]);
    title('Position of the dots with respect to camera frame')
    xlabel('X-axis')
```

```
    ylabel('Y-axis')
    zlabel('Z-axis')
    grid on

    fid=fopen(fileName,'wt');
    fprintf(fid, '%f      %f      %f    \n', [X Y Z]');
    fclose(fid);true

    num = length(radii);
end
```

This program takes the output from DOTSVgit a text file with the positions of each air bladder marking in [x,y,z] and converts it to a matlab data file.

Listing C.6: Convert Data for Matlab: txt2MatInterpreter

```
clear
clc

fname = "kpa_430_0209-1";

fSave = strcat('FileLocation\',fname,'.mat');
fileID = fopen(strcat("FileLocation\",fname,".txt"));
formatSpec = '%f';

A = zeros(1016,3);
Af = fscanf(fileID,formatSpec);
fclose('all');

lenAf = length(Af);

if length(Af)~=3048
    fprintf("Invalid number of dots\n")
end

sel = 0;
N = 32;

Af = reshape(Af,[],3);
lenAf = length(Af);

if lenAf>1024
    lenAf = 1024;
end

A(1:lenAf,:) = Af(1:lenAf,:);

xo = Af(:,1);
yo = Af(:,2);
zo = Af(:,3);

plot3(xo,yo,zo,'ro')
axis equal
view([0 90])

zMin = -0.6;
zMax = -0.4;
xMin = -1;
xMax = 1;
yMin = -1;
yMax = 1;

j=1;
for i=1:length(A)

    x = A(i,1);
    y = A(i,2);
    z = A(i,3);


    if z>zMin && z<zMax
        if x>xMin && x<xMax
            if y>yMin && y<yMax
                Ap(j,1) = x;
                Ap(j,2) = y;
                Ap(j,3) = z;
                j=j+1;
            end
        end
    end
```

```
65
66  end
67
68  % plot3(Ap(:,1),Ap(:,2),Ap(:,3),'ro')
69  % axis equal
70  % view([90 90])
71
72  %--------------- Rotate Points ---------------%
73
74  % theta = -pi/20;
75  % rotX = [1 , 0 , 0 ; 0 , cos(theta) , -sin(theta) ; 0 , sin(theta) , cos(theta)];
76  % rotY = [cos(theta) , 0 , sin(theta); 0 , 1 , 0 ; -sin(theta) , 0 , cos(theta)];
77  % rotZ = [cos(theta) , -sin(theta), 0 ; sin(theta) , cos(theta) , 0 ; 0 , 0 , 1];
78
79  % Ap = Ap';
80  % Ap = rotX*Ap;
81  % Ap = rotY*Ap;
82  % Ap = rotZ*Ap;
83  % Ap = Ap';
84
85  %---------------------------------------------%
86
87  x = Ap(:,1);
88  y = Ap(:,2);
89  z = Ap(:,3);
90
91  clf
92  plot3(x,y,z,'ro')
93  view([0 90])
94  axis equal
95
96  Ap2 = sortrows(Ap);
97
98  Ap3(:,1) = Ap2(:,2);
99  Ap3(:,2) = Ap2(:,1);
100 Ap3(:,3) = Ap2(:,3);
101
102 sortMaster = zeros(1024,3);
103 for i=1:32
104     sortMaster( (i-1)*32+1 : (i-1)*32+32,: ) = sortrows( Ap3( (i-1)*32+1 : (i-1)*32+32,: ) );
105 end
106
107 ApMaster(:,1) = sortMaster(:,2);
108 ApMaster(:,2) = sortMaster(:,1);
109 ApMaster(:,3) = sortMaster(:,3);
110
111 % clf
112 % for i=1:lenAf
113 %     plot3(ApMaster(i,1), ApMaster(i,2), ApMaster(i,3),'ro')
114 %     axis equal
115 %     view([0 90])
116 %     hold on
117 % end
118
119 xMat = reshape(ApMaster(:,1),N,[]);
120 yMat = reshape(ApMaster(:,2),N,[]);
121 zMat = reshape(ApMaster(:,3),N,[]);
122
123 % plot3(xMat,yMat,zMat,'r-')
124 % axis equal
125 % view([0 90])
126
127 dx = zeros(lenAf,1);
128 dy = zeros(lenAf,1);
129 dz = zeros(lenAf,1);
130
131 Q = [x, y, z, dx, dy, dz];
132 Q = reshape(Q,1,[]);
133
134 save(fSave,'Q','xMat','yMat','zMat')
```

comparePoints is utilized to plot the surfaces measured from each air bladder experimental data set.

Listing C.7: Surface Plot Comparing Data Samples: comparePoints

```
1  clear
2  clc
3
4  name = "kpa_101_0204";
```

```

for i=1:5
    fname = strcat(name,"-",num2str(i));

    fid = strcat('FileLocation\',fname,'.mat');

    load(fid)

    switch i
        case 1
            xMat1 = xMat;
            yMat1 = yMat;
            zMat1 = zMat;
        case 2
            xMat2 = xMat;
            yMat2 = yMat;
            zMat2 = zMat;
        case 3
            xMat3 = xMat;
            yMat3 = yMat;
            zMat3 = zMat;
        case 4
            xMat4 = xMat;
            yMat4 = yMat;
            zMat4 = zMat;
        case 5
            xMat5 = xMat;
            yMat5 = yMat;
            zMat5 = zMat;
    end

end

xMatAvg = (xMat1 + xMat2 + xMat3 + xMat4 + xMat5)/5;
yMatAvg = (yMat1 + yMat2 + yMat3 + yMat4 + yMat5)/5;
zMatAvg = (zMat1 + zMat2 + zMat3 + zMat4 + zMat5)/5;

xMatVar = ((xMatAvg - xMat1)^2 + (xMatAvg - xMat2)^2 + (xMatAvg - xMat3)^2 + (xMatAvg - xMat4)^2 +
      (xMatAvg - xMat5)^2)/4;
yMatVar = ((yMatAvg - yMat1)^2 + (yMatAvg - yMat2)^2 + (yMatAvg - yMat3)^2 + (yMatAvg - yMat4)^2 +
      (yMatAvg - yMat5)^2)/4;
zMatVar = ((zMatAvg - zMat1)^2 + (zMatAvg - zMat2)^2 + (zMatAvg - zMat3)^2 + (zMatAvg - zMat4)^2 +
      (zMatAvg - zMat5)^2)/4;

matVar = sqrt(xMatVar.^2 + yMatVar.^2 + zMatVar.^2);

heatmap(matVar)
```

Plots the springs taken from the data samples as a surface, displaying the amount of tension or compression compared to the intial sample on a red-blue color scale.

Listing C.8: Plots Relative Spring Strain From Data: strainCalculation

```
clear
clc

kPa1 = [101,105,107,1072,110,112,117];
kPa2 = 132;

KPAs = [110,105,107,1072,110,112,117,132];

xMatAvgs = zeros(32,32,8);
yMatAvgs = zeros(32,32,8);
zMatAvgs = zeros(32,32,8);
magVector = zeros(1984,8);

for i=1:7
    [xMatAvg, yMatAvg, zMatAvg] = bulkAvgCal(kPa1(i));
    xMatAvgs(:,:,i) = xMatAvg;
    yMatAvgs(:,:,i) = yMatAvg;
    zMatAvgs(:,:,i) = zMatAvg;
    magVector(:,i) = calSpringMag(xMatAvg, yMatAvg, zMatAvg);
end

[xMatAvg, yMatAvg, zMatAvg] = AvgCal132(kPa2);
xMatAvgs(:,:,8) = xMatAvg;
yMatAvgs(:,:,8) = yMatAvg;
zMatAvgs(:,:,8) = zMatAvg;
```

```
magVector(:,8) = calSpringMag(xMatAvg, yMatAvg, zMatAvg);

magVector = magVector-magVector(:,1);

% for i=1:8
%     plot3(xMatAvgs(:,:,i),yMatAvgs(:,:,i),zMatAvgs(:,:,i),'r.')
%     axis equal
%     view([30 30])
%     pause(0.5)
% end

clf

for i=1:8
    sName = strcat('C:\Users\Cristian\OneDrive\UTA-Cristian-PC\Fall2019\Thesis\Autlev\Scaling\data
        \0204\',num2str(KPAs(i)),'.fig');
    colorVect32 = colorGen32(magVector(:,i));
    plotRow2(xMatAvgs(:,:,i),yMatAvgs(:,:,i),zMatAvgs(:,:,i),colorVect32)
%     s=surf(xMatAvgs(:,:,i),yMatAvgs(:,:,i),zMatAvgs(:,:,i))
    axis equal
    hold off
    savefig(sName)
end


function plotRow2(xMat, yMat, zMat, colorVec)
    size0=size(xMat);
    rows = size0(1);
    cols = size0(2);

    %handle rows first
    k=1;
    for i=1:cols
        for j=1:(rows-1)
            Xs = [xMat(i,j), xMat(i,j+1)];
            Ys = [yMat(i,j), yMat(i,j+1)];
            Zs = [zMat(i,j), zMat(i,j+1)];
            plot3(Xs,Ys,Zs,'Color',colorVec(k,:))
            hold on
            axis equal
            k=k+1;
        end
    end

    for i=1:(cols)
        for j=1:(rows-1)
            Xs = [xMat(j,i), xMat(j+1,i)];
            Ys = [yMat(j,i), yMat(j+1,i)];
            Zs = [zMat(j,i), zMat(j+1,i)];
            plot3(Xs,Ys,Zs,'Color',colorVec(k,:))
            hold on
            axis equal
            k=k+1;
        end
    end

end

function plotRow(xMat, yMat, zMat)
    wid = size(xMat);
    for i=1:wid
        plot3(xMat(i,:), yMat(i,:), zMat(i,:))
        hold on
        axis equal
        plot3(xMat(:,i), yMat(:,i), zMat(:,i))
    end
end

function colorVect32 = colorGen32(magVector)
    magMax = max(magVector);
    magMin = min(magVector);

    lenVec = length(magVector);
    colorVect32 = zeros(lenVec,3);

    for i=1:lenVec
        if magVector(i)>0
            red1 = magVector(i)/magMax;
            green1 = (1-red1);
            blue1 = (1-red1);
           colorVect32(i,:) = [1,green1,blue1];
        elseif magVector(i)<0
            red1 = (1-blue1);
            blue1=magVector(i)/magMin;
            green1=(1-blue1);
           colorVect32(i,:) = [red1,green1,1];
        else
```

```
            colorVect32(i,:)=[0.86,0.86,0.86];
        end
    end

end

function [xMatAvg, yMatAvg, zMatAvg] = bulkAvgCal(kPa)
        for i=1:5
            name = strcat("kpa_",num2str(kPa),"_0204");
            fname = strcat(name,"-",num2str(i));
            fid = strcat('FileLocation\',fname,'.mat');
            load(fid)
            switch i
                case 1
                    xMat1 = xMat;
                    yMat1 = yMat;
                    zMat1 = zMat;
                case 2
                    xMat2 = xMat;
                    yMat2 = yMat;
                    zMat2 = zMat;
                case 3
                    xMat3 = xMat;
                    yMat3 = yMat;
                    zMat3 = zMat;
                case 4
                    xMat4 = xMat;
                    yMat4 = yMat;
                    zMat4 = zMat;
                case 5
                    xMat5 = xMat;
                    yMat5 = yMat;
                    zMat5 = zMat;
            end
        end
        xMatAvg = (xMat1 + xMat2 + xMat3 + xMat4 + xMat5)/5;
        yMatAvg = (yMat1 + yMat2 + yMat3 + yMat4 + yMat5)/5;
        zMatAvg = (zMat1 + zMat2 + zMat3 + zMat4 + zMat5)/5;
end

function [xMatAvg, yMatAvg, zMatAvg] = AvgCal132(kPa)
        for i=1:2
            name = strcat("kpa_",num2str(kPa),"_0204");
            fname = strcat(name,"-",num2str(i));
            fid = strcat('FileLocation\',fname,'.mat');
            load(fid)
            switch i
                case 1
                    xMat1 = xMat;
                    yMat1 = yMat;
                    zMat1 = zMat;
                case 2
                    xMat2 = xMat;
                    yMat2 = yMat;
                    zMat2 = zMat;
            end
        end
        xMatAvg = (xMat1 + xMat2)/2;
        yMatAvg = (yMat1 + yMat2)/2;
        zMatAvg = (zMat1 + zMat2)/2;
end
```

Fits the experimental data collected from the air bladder prototype to a surface function where r=0.99. This was initially used as a component of the spring k-solver program.

Listing C.9: Surface Plot of Individual Sample: surfaceFit

```
clear
clc
close all

loadMe = "kpa_101_0204-1";
saveLoc = 'FileLocation\';
loadName = strcat(saveLoc,loadMe,'.mat');

N=5;
load(loadName,'Q');

```

```
Np=length(Q)/6;

ql = size(Q);
qStar = ql(1);
ql(1) = qStar;

P = (1:Np)';
P(:,2) = Q(1,       1 : 1*Np);
P(:,3) = Q(1,    Np+1 : 2*Np);
P(:,4) = Q(1, 2*Np+1 : 3*Np);
P(:,5) = Q(1, 3*Np+1 : 4*Np);
P(:,6) = Q(1, 4*Np+1 : 5*Np);
P(:,7) = Q(1, 5*Np+1 : 6*Np);

xA = 0;
yA = 0;
zA = 0.54;

x = (P(:,2) + xA)/.4*.28;
y = (P(:,3) + yA)/.4*.28;
z = (P(:,4) + zA);

plot3(x,y,z,'ro')
axis equal
xlabel("X")
ylabel("Y")

[fitresult, gof] = createFit(x, y, z);

fprintf("STOP");
plot(fitresult)
axis equal

% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% %                                 %
% %     build better data points    %
% %                                 %
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% xS = 0:0.2:0.4;
% yS = 0:0.2:0.4;
% [xS, yS] = meshgrid(xS,yS);
% Cnt = length(xS);
%
% zS(1,1) = getHeight(xS(1,1),yS(1,1));
% zS(1,2) = getHeight(xS(1,2),yS(1,2));
% zS(1,3) = getHeight(xS(1,3),yS(1,3));
% zS(2,1) = getHeight(xS(2,1),yS(2,1));
% zS(2,2) = getHeight(xS(2,2),yS(2,2));
% zS(2,3) = getHeight(xS(2,3),yS(2,3));
% zS(3,1) = getHeight(xS(3,1),yS(3,1));
% zS(3,2) = getHeight(xS(3,2),yS(3,2));
% zS(3,3) = getHeight(xS(3,3),yS(3,3));
%
% bedLength = sqrt((zS(1,1)-zS(1,3))^2 + (xS(1,1)-xS(1,3))^2);
%
% xS = reshape(xS,9,[]);
% yS = reshape(yS,9,[]);
% zS = reshape(zS,9,[]);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                %
%       Build better data points                 %
%       Instructions:                            %
%         To control surface course-ness         %
%         Enter N = #, where the surface         %
%         will be a NxN surface (excluding       %
%         the boundary points).                  %
%                                                %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
nMean = zeros(120,1);
rmMean = zeros(120,1);

maxX = max(x);
minX = min(x);
maxY = max(y);
minY = min(y);
maxZ = max(z);
minZ = min(z);

    zS2 = 0.28/(N+1) + 1;
    xS2 = minX:(maxX-minX)/(N+1):maxX;
    yS2 = minY:(maxY-minY)/(N+1):maxY;

    [xS2, yS2] = meshgrid(xS2,yS2);

    Cnt = length(xS2);
```

```
99
100     for i = 1:length(xS2)
101        for j=1:length(yS2)
102              zS2(i,j) = fitresult(xS2(i,j),yS2(i,j));
103        end
104     end
105
106     xS2 = reshape(xS2,[],1);
107     yS2 = reshape(yS2,[],1);
108     zS2 = reshape(zS2,[],1);
109
110     plot3(xS2,yS2, zS2,'ro')
111     axis equal
112
113     Nstr = num2str(N);
114     fName = strcat('basic',Nstr,'.txt');
115
116     fid = fopen(fName,'wt');
117     for dataPnts = 1:length(xS2)
118          fprintf(fid, '%f\t%f\t%f\n', xS2(dataPnts), yS2(dataPnts), zS2(dataPnts));
119     end
120     fclose(fid);
121     fprintf('Created %0.fx%0.f bed. SUCCESS!!!\n',N,N)
122
123
124     sigma = 0;
125     dx = zeros(length(xS2),1);
126     dy = zeros(length(xS2),1);
127     dz = zeros(length(xS2),1);
128
129
130     Q = [xS2, yS2, zS2, dx, dy, dz];
131     Q = reshape(Q,1,[]);
132
133     fname = strcat(loadMe,"_",num2str(N),".mat");
134     fSave = strcat(saveLoc,fname);
135     save(fSave,'Q')
```

Plots the average positional deviation across N number of samples at a given internal pressure of the air bladder.

Listing C.10: Average Positional Deviation: varianceCalculation

```
1  clear
2  clc
3
4  name = "kpa_110_0204";
5
6  for i=1:5
7      fname = strcat(name,"-",num2str(i));
8
9      fid = strcat('FileLocation\',fname,'.mat');
10
11     load(fid)
12
13     switch i
14         case 1
15             xMat1 = xMat;
16             yMat1 = yMat;
17             zMat1 = zMat;
18         case 2
19             xMat2 = xMat;
20             yMat2 = yMat;
21             zMat2 = zMat;
22         case 3
23             xMat3 = xMat;
24             yMat3 = yMat;
25             zMat3 = zMat;
26         case 4
27             xMat4 = xMat;
28             yMat4 = yMat;
29             zMat4 = zMat;
30         case 5
31             xMat5 = xMat;
32             yMat5 = yMat;
33             zMat5 = zMat;
34     end
35
36 end
37
```

```
xMatAvg = (xMat1 + xMat2 + xMat3 + xMat4 + xMat5)/5;
yMatAvg = (yMat1 + yMat2 + yMat3 + yMat4 + yMat5)/5;
zMatAvg = (zMat1 + zMat2 + zMat3 + zMat4 + zMat5)/5;

xMatVar = ((xMatAvg - xMat1)^2 + (xMatAvg - xMat2)^2 + (xMatAvg - xMat3)^2 + (xMatAvg - xMat4)^2 +
      (xMatAvg - xMat5)^2)/4;
yMatVar = ((yMatAvg - yMat1)^2 + (yMatAvg - yMat2)^2 + (yMatAvg - yMat3)^2 + (yMatAvg - yMat4)^2 +
      (yMatAvg - yMat5)^2)/4;
zMatVar = ((zMatAvg - zMat1)^2 + (zMatAvg - zMat2)^2 + (zMatAvg - zMat3)^2 + (zMatAvg - zMat4)^2 +
      (zMatAvg - zMat5)^2)/4;

matVar = sqrt(xMatVar.^2 + yMatVar.^2 + zMatVar.^2);

heatmap(matVar)
magVector = calSpringMag(xMatAvg,yMatAvg,zMatAvg);
```

Another surface plotting program used to evaulate the data collected from the experimental data.

Listing C.11: Plot Data Collected As a Surface: surfacePlotData

```
clear
clc

KPAs = [101,105,110,132];

for i=1:3
    [xMatAvg, yMatAvg, zMatAvg] = bulkAvgCal(KPAs(i));
    xMatAvgs(:,:,i) = xMatAvg;
    yMatAvgs(:,:,i) = yMatAvg;
    zMatAvgs(:,:,i) = zMatAvg;
end

[xMatAvg, yMatAvg, zMatAvg] = AvgCal132(KPAs(4));
xMatAvgs(:,:,4) = xMatAvg;
yMatAvgs(:,:,4) = yMatAvg;
zMatAvgs(:,:,4) = zMatAvg;

for i=1:4
    clf
    sName = strcat('FileLocation\',num2str(KPAs(i)),'.fig');
    s=surf(xMatAvgs(:,:,i),yMatAvgs(:,:,i),zMatAvgs(:,:,i),'FaceColor','Interp' );
    axis equal
%     hold on
    pause()

end

function [xMatAvg, yMatAvg, zMatAvg] = bulkAvgCal(kPa)

        for i=1:5

            name = strcat("kpa_",num2str(kPa),"_0204");
            fname = strcat(name,"-",num2str(i));
            fid = strcat('FileLocation\',fname,'.mat');

            load(fid)

            switch i
                case 1
                    xMat1 = xMat;
                    yMat1 = yMat;
                    zMat1 = zMat;
                case 2
                    xMat2 = xMat;
                    yMat2 = yMat;
                    zMat2 = zMat;
                case 3
                    xMat3 = xMat;
                    yMat3 = yMat;
                    zMat3 = zMat;
                case 4
                    xMat4 = xMat;
                    yMat4 = yMat;
                    zMat4 = zMat;
                case 5
                    xMat5 = xMat;
                    yMat5 = yMat;
                    zMat5 = zMat;
            end
```

```
        end
        xMatAvg = (xMat1 + xMat2 + xMat3 + xMat4 + xMat5)/5;
        yMatAvg = (yMat1 + yMat2 + yMat3 + yMat4 + yMat5)/5;
        zMatAvg = (zMat1 + zMat2 + zMat3 + zMat4 + zMat5)/5;
end

function [xMatAvg, yMatAvg, zMatAvg] = AvgCal132(kPa)
        for i=1:2
            name = strcat("kpa_",num2str(kPa),"_0204");
            fname = strcat(name,"-",num2str(i));
            fid = strcat('FileLocation\',fname,'.mat');
            load(fid)
            switch i
                case 1
                    xMat1 = xMat;
                    yMat1 = yMat;
                    zMat1 = zMat;
                case 2
                    xMat2 = xMat;
                    yMat2 = yMat;
                    zMat2 = zMat;
            end
        end
        xMatAvg = (xMat1 + xMat2)/2;
        yMatAvg = (yMat1 + yMat2)/2;
        zMatAvg = (zMat1 + zMat2)/2;
end
```

Program used for system identification of collected air bladder experimental data.

Listing C.12: System Identification: kSolver

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                   RUN K-SOLVER                                 %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clf
clear
clc

N = 7;
M = 7;
pressure = 500;
TFINAL = 70;

Kcount = 4;  %1-6
KDirectory = [400 600 800 1000 1200 1400];

K = KDirectory(Kcount);

directory = strcat(num2str(N),'x',num2str(M),'_press_',num2str(pressure),'_t',num2str(TFINAL),'_k'
    ,num2str(K));
%directory = '';
%   ZeroPressure_60s_Test.mat

fLoad = strcat("FileLocations\",directory,".mat");
fSaveName1 = strcat('FileLocations\',directory,'.txt');

%Load Parameters (LOst, Damp, mass, etc.)
params = load(fLoad);

LoSt = params.LoSt;
LoSh = params.LoSh;
LoBe = params.LoBe;
LoRi = params.LoRi;
pressure = params.pressure;
mass = params.mass;
KSh = params.KSh;
KSt = params.KSt;
KBe = params.KBe;
KRi = params.KRi;
dampSt = params.dampSt;
dampSh = params.dampSh;
dampRI = params.dampRI;
dampBe = params.dampBe;
springCntST = params.springCntST;
springCntSH = params.springCntSH;
springCntRI = params.springCntRI;
springCntBE = params.springCntBE;
```

```
47  springMap_ST = full(params.springMap_ST);
48  springMap_SH = full(params.springMap_SH);
49  springMap_RI = full(params.springMap_RI);
50  springMap_BE = full(params.springMap_BE);
51
52  %Load Q (Time, Pressure, Positions, Velocities)
53  FID = fopen(fSaveName1,'r');
54
55  numBods = (N^2 + 4*(N-1)*(M-1))*6+3;
56
57  lengthQ = TFINAL*10;%- 1;
58  formatSpec = '%f';
59  Q0 = fscanf(FID,formatSpec);
60  Q = reshape(Q0,numBods,[])';
61
62  % Sample Times
63  T1 = 9;
64  T2 = 24;
65  T3 = 39;
66  T4 = 54;
67  T5 = 69;
68
69  xLength = 0.3;
70  yLength = 0.3;
71  zLength = 0.127;
72
73  numBods = N^2+4*(N-1)^2;
74
75  grav =9.81; %m/s^2
76  area = (2*(xLength*zLength)+2*(yLength*zLength)+(yLength*xLength))/numBods;
77  % mass = totalMass / numBods;
78  gVector = repmat([0;0;-grav*mass],numBods,1);
79
80  Masses = repmat(mass,1,3*(N^2+4*(N-1)^2));
81  A = sparse(diag(Masses));
82  clear Masses
83
84  %-------------------------------------------------------------------------
85
86  % Q should be
87  Q0 = Q(1,3:size(Q,2));
88  Q1 = Q(T1*10,3:size(Q,2));
89  Q2 = Q(T2*10,3:size(Q,2));
90  Q3 = Q(T3*10,3:size(Q,2));
91  Q4 = Q(T4*10,3:size(Q,2));
92  Q5 = Q(T5*10,3:size(Q,2));
93
94  pressure1 = Q(T1*10,2);
95  pressure2 = Q(T2*10,2);
96  pressure3 = Q(T3*10,2);
97  pressure4 = Q(T4*10,2);
98  pressure5 = Q(T5*10,2);
99
100 [x, y, z, Dx, Dy, Dz, eCheck] = buildBedCoords(N,M,Q0);
101 DT = delaunayTriangulation(x',y',z');
102 Con = freeBoundary(DT);
103 [~,Vo] = convexHull(DT);
104
105 %USED TO CALC CURRENT LENGTHS
106 LoStC = lengthBuilder(springCntST, N, M, Q0);
107 LoShC = lengthBuilder(springCntSH, N, M, Q0);
108 LoBeC = lengthBuilder(springCntBE, N, M, Q0);
109 LoRiC = lengthBuilder(springCntRI, N, M, Q0);
110
111 [A1, b1] = processQ(Q1 , N, M, springMap_ST, springMap_SH, springMap_BE, springMap_RI, springCntST
        , springCntSH, springCntBE, springCntRI, gVector,  pressure1, Vo, area, LoStC, LoShC, LoBeC,
        LoRiC,Con);
112 [A2, b2] = processQ(Q2 , N, M, springMap_ST, springMap_SH, springMap_BE, springMap_RI, springCntST
        , springCntSH, springCntBE, springCntRI, gVector,  pressure2, Vo, area, LoStC, LoShC, LoBeC,
        LoRiC,Con);
113 [A3, b3] = processQ(Q3 , N, M, springMap_ST, springMap_SH, springMap_BE, springMap_RI, springCntST
        , springCntSH, springCntBE, springCntRI, gVector,  pressure3, Vo, area, LoStC, LoShC, LoBeC,
        LoRiC,Con);
114 [A4, b4] = processQ(Q4 , N, M, springMap_ST, springMap_SH, springMap_BE, springMap_RI, springCntST
        , springCntSH, springCntBE, springCntRI, gVector,  pressure4, Vo, area, LoStC, LoShC, LoBeC,
        LoRiC,Con);
115 [A5, b5] = processQ(Q5 , N, M, springMap_ST, springMap_SH, springMap_BE, springMap_RI, springCntST
        , springCntSH, springCntBE, springCntRI, gVector,  pressure5, Vo, area, LoStC, LoShC, LoBeC,
        LoRiC,Con);
116
117 A1 = full(A1);
118 A2 = full(A2);
119 A3 = full(A3);
120 A4 = full(A4);
121 A5 = full(A5);
122
123 A = [A1 ; A2 ; A3 ; A4 ; A5];%
```

```
b = [b1 ; b2 ; b3 ; b4 ; b5];

%filter Top A Bodies
A1_top2 = A1(1:N*M*3,:);
%filter Top springs only
j=1;
for i=1:size(A1_top2,2)
    bullox = isequal( A1_top2(:,size(A1_top2,2)+1-i) , zeros(size(A1_top2,1),1) );
    if( bullox == 0 )
        A1_top(:,j) = A1_top2(:,size(A1_top2,2)+1-i);
        j=j+1;
    end
end
A1_top = flip(A1_top,1);

%Filter Structural Body and Springs
    % Filter Top
    j=1;
    for i=1:size(springCntST,1)
        if(springCntST(i,2)<=49)
            springCntST2(j,:)=springCntST(i,:);
            j=j+1;
        end
    end
    % Filter front and rear edges
    j=1;
    for i=1:size(springCntST2,1)
        if( springCntST2(i,1)+1~=springCntST2(i,2) || (springCntST2(i,1)>N && springCntST2(i,1)<N
            *(M-1)) )
            springCntST3(j,:)=springCntST2(i,:);
            j=j+1;
        end
    end

    % Filter out left and right edges
    leftEdge(:,1) = (1:N:N*(M-1));
    leftEdge(:,2) = (1+N:N:N*(M));
    rightEdge(:,1) = (N:N:N*(M-1));
    rightEdge(:,2) = (N+N:N:N*(M));

    ij=1;
    for i=1:size(springCntST3,1)
        for j=1:size(leftEdge,1)
            Lcheck = isequal(springCntST3(i,1:2),leftEdge(j,:));
            Rcheck = isequal(springCntST3(i,1:2),rightEdge(j,:));
            if( Lcheck==1 )
                springCntST3(i,3) = 1;
            end
            if( Rcheck==1 )
                springCntST3(i,3) = 1;
            end
        end
    end
    size0 = size(springCntST3,1);
    for i=1:size0
        if(springCntST3(size0+1-i,3)==1 )
            springCntST3(size0+1-i,:)=[];
        end
    end
    springCntST = springCntST3(:,1:2);

j=1;
for i=1:size(springCntSH,1)
    if(springCntSH(i,2)<=49)
        springCntSH2(j,:)=springCntSH(i,:);
        j=j+1;
    end
end

%top left
    nullNodes(1,1) = 2;
    nullNodes(1,2) = 2+(N-1);
%top right
    nullNodes(2,1) = (N-1);
    nullNodes(2,2) = (N)+(N);
%bottom left
    nullNodes(3,1) = N*(M-2)+1;
    nullNodes(3,2) = N*(M-1)+2;
%bottom right
    nullNodes(4,1) = N*(M-1);
    nullNodes(4,2) = (N)*(N)-1;

    for i=1:size(springCntSH2,1)
        for j=1:size(nullNodes,1)
            Nullcheck = isequal(springCntSH2(i,1:2),nullNodes(j,:));
            if( Nullcheck==1 )
                springCntSH2(i,3) = 1;
```

```
            end
        end
    end

size0 = size(springCntSH2,1);
for i=1:size0
    if(springCntSH2(size0+1-i,3)==1 )
        springCntSH2(size0+1-i,:)=[];
    end
end
springCntSH = springCntSH2(:,1:2);

% Try top surface only
% Show the first N*M bodies are the complete top surface
xTop = x(1:N*M);
yTop = y(1:N*M);
zTop = z(1:N*M);

for i=1:length(x)
    plot3(xTop,yTop,zTop,'k*')
    hold on
%     pause(0.1);
end

% Xtop not edges
topMatrix = reshape([1:N*M],N,M)';
topVector = reshape(topMatrix(2:N-1,2:M-1)',(N-2)*(M-2),[]);

LoSt_top = lengthBuilder(springCntST, N, M, Q0);
LoSh_top = lengthBuilder(springCntSH, N, M, Q0);

[springMap_ST] = springMapBuilder_top(N, M, springCntST);
[springMap_SH] = springMapBuilder_top(N, M, springCntSH);

[A1_top, b1_top] = processQ_top(Q1 , N, M, springMap_ST, springMap_SH, springCntST, springCntSH,
    gVector,  pressure1, Vo, area, LoSt_top, LoSh_top, Con);

A1_top = full(A1_top);
for i=1:size(topVector,1)
    for j=1:3
        topVector_expanded(3*(i-1)+j,1) = 3*(topVector(i)-1)+j;
    end
end

j=1;
for i=1:size(topVector_expanded,1)
    A1_topM(j,:) = A1_top(topVector_expanded(i),:);
    j=j+1;
end

ka = repmat(1000,128,1);
BAns = A1_topM*ka;


% SVD Compression
%     [U,S,V]=svd(A);
%
%     G=S';
%     for i=1:size(S,2)
%         G(i,i)=1/S(i,i);
%     end
%
%     r1 = b-A*V*G*U'*b;
%     e1 = sqrt(r1'*r1);
%
%     Ks2 = V*G*U'*b;
%     % Ks3 = V*inv(S)*U'*b;
%     A_2 = zeros(size(A,1),size(A,2));
%     for i=1:rank(A)
%         Ek=U(:,i)*V(:,i)';
%         A_2 = A_2 + S(i,i)*Ek;
%     end

% % LSQ Norm Min
%     Ks_LSQMN = lsqminnorm(A,b);
%     mean(Ks_LSQMN)
%     mean(rmoutliers(Ks_LSQMN,'mean'))

% % Truncated SVD
%     [U,S,V]=svd(A);
%     K_truncatedSVD = zeros(size(V,1),1);
%     for i=1:rank(A)
%         K_truncatedSVD = K_truncatedSVD + V(:,i)*(U(:,i)'*b/S(i,i));
%     end
%     mean(K_truncatedSVD)
%     mean(rmoutliers(K_truncatedSVD,'mean'))

```

```
%% Remove dependent rows and unused springs
%     [Xsub,idx]=licols(A');
%     Xsub=Xsub';
%
%     [Xsub, Abasisi, Asub]= getLinearIndependent(A');
%     Xsub=Xsub';
%
%     bnew2(1,1)=b(1,1);
%     for i=2:size(Asub,2)
%         bnew2(i,1)=b(cell2mat(Asub(1,i)),1);
%     end
%
%     cnt4=0;
%     j=1;
%     for i=1:size(Xsub,2)
%         if isequal(Xsub(:,i),zeros(size(Xsub,1),1))
%         else
%            Anew2(:,j) = Xsub(:,i);
%            j=j+1;
%         end
%     end
%
%     K_rmDRC = Anew2\bnew2;
%     mean(K_rmDRC)

%% % Tikhonov Regularization
%     lambda = 1e-6;
%     b_0 = zeros(size(A,2));
%     b_0(2:size(b_0,1),1:size(b_0,2)-1) = diag(-1*ones(size(A,2)-1,1));
%     BigB = diag(ones(size(A,2),1)) + b_0;
%
%     K_TReg = inv(A'*A+lambda^2*BigB'*BigB)*A'*b;
%
%     mean(K_TReg)
%     mean(rmoutliers(K_TReg,'median'))
%     %Check to see if lambda is correct

%% Biconjugate Gradient
%     A_star = A'*A;
%     maxIt = 10000;
%     tol = 1e-8;
%     K_BigC = bicg(A_star,A'*b, tol, maxIt);
%     mean(K_BigC)
%     mean(rmoutliers(K_BigC,'mean'))

%% Landweber Iteration
%     [U,S,V]=svd(A);
%     n=100;
%     lambda = 100;
%     K_LW = zeros(size(V,1),1);
%     for i=1:1000
%         K_LW = K_LW + A'*(b-A*K_LW);
%     end
%
%
%     for i=1:1042
%         K_LW = K_LW + V(:,i)*( ( 1-(1-S(i,i)^2)^i )/S(i,i)  )*(U(:,i)'*b);
%     end
%     mean(K_LW)
%     mean(rmoutliers(K_LW,'mean'))
    % doesnt work?

%% Least Sqaure - Preconditioned
%     tol = 1e-12;
%     maxit = 1000;
%     % x is the computed solution to A*x = b.
%     % fl is a flag indicating whether the algorithm converged.
%     % rr is the relative residual of the computed answer x.
%     % it is the iteration number when x was computed.
%     % rv is a vector of the residual history for     bAx    .
%     % lsrv is a vector of the least squares residual history.
%     setup = struct('type','ilutp','droptol',1e-6);
%     [L,U] = ilu(sparse(Anew2),setup);
%     [x,fl,rr,it,rv,lsrv] = lsqr(Anew2,bnew2,tol,maxit,L,U);
%     fl
%     it
%     rr

%% Landweber Iteration
%     [U,S,V]=svd(A);
%     n=100;
%     lambda = 100;
%     K_LW = repmat(100,size(V,1),1);
%
%     for i=1:1042
%         K_LW = K_LW + V(:,i)*(1-S(i,i)^2)^i*(V(:,i)'*K_LW);
%     end
```

```
%      mean(K_LW)
%      mean(rmoutliers(K_LW,'mean'))
%
%      lambda = 1e-6;
%      b_0 = zeros(size(A,2));
%      b_0(2:size(b_0,1),1:size(b_0,2)-1) = diag(-1*ones(size(A,2)-1,1));
%      BigB = diag(ones(size(A,2),1)) + b_0;
%
%      K_TReg = inv(A'*A+lambda^2*BigB'*BigB)*A'*b;
%
%      mean(K_TReg)
%      mean(rmoutliers(K_TReg,'median'))
    %Check to see if lambda is correct


% Ks_A_2 = lsqminnorm(A_2,b);
% Ks_A = lsqminnorm(A,b);
% w1=mean(rmoutliers(Ks_A_2,'median'));
% w2=mean(rmoutliers(Ks_A,'median'));
% fprintf("%.0f %.0f",w1,w2);


return;
%-------------------------------------------------------------------------

function [springMap] = springMapBuilder_top(N, M,springCnt)
    springMap = zeros(N*M,size(springCnt,1));
    for i=1:size(springCnt)
        springMap(springCnt(i,1),i)=1;
        springMap(springCnt(i,2),i)=-1;
    end
%     springMap=reshape([springMap , springMap , springMap]',size(springMap,2),[])';
end

%-------------------------------------------------------------------------

function [ x, y, z, Dx, Dy, Dz, eCheck ] =       buildBedCoords(N,M,Q)
    numBod = (N^2 + 4*(N-1)*(M-1));

    xpop = 1;
    ypop = numBod;
    x = Q(    xpop   :   ypop    );

    xpop = 1+1*numBod;
    ypop = 2*numBod;
    y = Q(    xpop   :   ypop    );

    xpop = 1+2*numBod;
    ypop = 3*numBod;
    z = Q(    xpop   :   ypop    );

    xpop = 1+3*numBod;
    ypop = 4*numBod;
    Dx= Q(    xpop   :   ypop    );

    xpop = 1+4*numBod;
    ypop = 5*numBod;
    Dy= Q(    xpop   :   ypop    );

    xpop = 1+5*numBod;
    ypop = 6*numBod;
    Dz= Q(    xpop   :   ypop    );

%     eCheck = 0;
    eCheck = Q(ypop+1);
end

%-------------------------------------------------------------------------

function [pVector, pressure, Vol] = calcPressure2(x,y,z,pressure,area,Vo,Con)
    P = [x,y,z];
    [~,Vol]=convhulln(P);
    TR = triangulation(Con,P);
    V = vertexNormal(TR);

    deltaV =1;% Vo/Vol;
    pressure = pressure*deltaV;
    pForce = pressure*area;

    quiver3(x,y,z,V(:,1),V(:,2),V(:,3))

    pVector = reshape(V',size(V,1)*size(V,2),[]);
    pVector = pVector*pForce;

end

%-------------------------------------------------------------------------
```

```
470
471 function [magVector, xVector, yVector, zVector, stretch, stretchP] =        vectorBuilder(Lo,
     springCnt,x,y,z,Dx,Dy,Dz)
472     length=size(springCnt,1);
473     xVector = zeros(length,1);
474     yVector = zeros(length,1);
475     zVector = zeros(length,1);
476     xVelVec = zeros(length,1);
477     yVelVec = zeros(length,1);
478     zVelVec = zeros(length,1);
479
480     for i=1:length
481        xVector(i,1) = x(springCnt(i,2)) - x(springCnt(i,1));
482        yVector(i,1) = y(springCnt(i,2)) - y(springCnt(i,1));
483        zVector(i,1) = z(springCnt(i,2)) - z(springCnt(i,1));
484        xVelVec(i,1) = Dx(springCnt(i,2)) - Dx(springCnt(i,1));
485        yVelVec(i,1) = Dy(springCnt(i,2)) - Dy(springCnt(i,1));
486        zVelVec(i,1) = Dz(springCnt(i,2)) - Dz(springCnt(i,1));
487     end
488
489     magVector = vecnorm([xVector yVector zVector],2,2);
490
491     stretch = magVector-Lo;
492     %calculate initial Lo for each spring prior to running code
493     stretchP = (xVector.*xVelVec + yVector.*yVelVec + zVector.*zVelVec)./magVector;
494 end
495
496 %-------------------------------------------------------------------------
497
498 function Lo = lengthBuilder(springCnt,N,M,Q)
499      num =  N^2+4*(N-1)*(M-1);
500     x = Q(1:num);
501     y = Q(1+1*num:2*num);
502     z = Q(1+2*num:3*num);
503     length=size(springCnt,1);
504
505     xVector = zeros(length,1);
506     yVector = zeros(length,1);
507     zVector = zeros(length,1);
508
509     for i=1:length
510        xVector(i,1) = x(springCnt(i,2)) - x(springCnt(i,1));
511        yVector(i,1) = y(springCnt(i,2)) - y(springCnt(i,1));
512        zVector(i,1) = z(springCnt(i,2)) - z(springCnt(i,1));
513     end
514     Lo = sqrt(xVector.^2 + yVector.^2 + zVector.^2);
515 end
516
517 %-------------------------------------------------------------------------
518
519 function [Sh1, Sh2] =       springForceBuilder(~,~, xVector, yVector, zVector, magVector,
     springMapMini1, stretch, stretchP)
520
521     xMat1 = sparse(springMapMini1*diag(xVector./magVector));
522     yMat1 = sparse(springMapMini1*diag(yVector./magVector));
523     zMat1 = sparse(springMapMini1*diag(zVector./magVector));
524
525     springMapShear = reshape( [xMat1, yMat1, zMat1]' , length(xVector) , [] )';
526     Sh1 = sparse(springMapShear*diag(stretch));
527     Sh2 = sparse(springMapShear*diag(stretchP));
528
529 end
530
531 %-------------------------------------------------------------------------
532
533 function [A, b] = processQ(Q , N, M, springMap_ST, springMap_SH, springMap_BE, springMap_RI,
     springCntST, springCntSH, springCntBE, springCntRI, gVector,  pressure, Vo, area, LoStC, LoShC
     , LoBeC, LoRiC,Con)
534     [ x, y, z, Dx, Dy, Dz, eCheck] = buildBedCoords(N,M,Q);
535
536     % DT = delaunayTriangulation(x,y,z);
537     % Con = freeBoundary(DT);
538     % [~,Vo] = convexHull(DT);
539
540     x=x';
541     y=y';
542     z=z';
543
544     [pVector] = calcPressure2(x,y,z,pressure,area,Vo,Con);
545
546
547     [magVector_ST, xVector_ST, yVector_ST, zVector_ST, stretch_ST, stretchP_ST] = vectorBuilder(
          LoStC, springCntST, x,y,z,Dx,Dy,Dz);
548     [magVector_SH, xVector_SH, yVector_SH, zVector_SH, stretch_SH, stretchP_SH] = vectorBuilder(
          LoShC, springCntSH, x,y,z,Dx,Dy,Dz);
549     [magVector_BE, xVector_BE, yVector_BE, zVector_BE, stretch_BE, stretchP_BE] = vectorBuilder(
          LoBeC, springCntBE, x,y,z,Dx,Dy,Dz);
```

```
550     [magVector_RI, xVector_RI, yVector_RI, zVector_RI, stretch_RI, stretchP_RI] = vectorBuilder(
            LoRiC, springCntRI, x,y,z,Dx,Dy,Dz);
551
552     [St1, St2] = springForceBuilder(N, M, xVector_ST, yVector_ST, zVector_ST, magVector_ST,
            springMap_ST, stretch_ST, stretchP_ST);
553     [Sh1, Sh2] = springForceBuilder(N, M, xVector_SH, yVector_SH, zVector_SH, magVector_SH,
            springMap_SH, stretch_SH, stretchP_SH);
554     [Sb1, Sb2] = springForceBuilder(N, M, xVector_BE, yVector_BE, zVector_BE, magVector_BE,
            springMap_BE, stretch_BE, stretchP_BE);
555     [Se1, Se2] = springForceBuilder(N, M, xVector_RI, yVector_RI, zVector_RI, magVector_RI,
            springMap_RI, stretch_RI, stretchP_RI);
556
557     [fixForces] = constrainedForcesSolver(N,M,gVector,pVector);
558
559     A = [ St1 Sh1 Sb1 ]; %
560     b = -(gVector + pVector);% + fixForces;
561 end
562
563 %-------------------------------------------------------------------------
564
565 function [A, b] = processQ_top(Q , N, M, springMap_ST, springMap_SH, springCntST, springCntSH,
        gVector,  pressure, Vo, area, LoStC, LoShC,Con)
566     [ x, y, z, Dx, Dy, Dz, eCheck] = buildBedCoords(N,M,Q);
567
568     plot3(x,y,z,'ro')
569
570     % DT = delaunayTriangulation(x,y,z);
571     % Con = freeBoundary(DT);
572     % [~,Vo] = convexHull(DT);
573
574     x=x';
575     y=y';
576     z=z';
577
578     [pVector] = calcPressure2(x,y,z,pressure,area,Vo,Con);
579
580
581     [magVector_ST, xVector_ST, yVector_ST, zVector_ST, stretch_ST, stretchP_ST] = vectorBuilder(
            LoStC, springCntST, x,y,z,Dx,Dy,Dz);
582     [magVector_SH, xVector_SH, yVector_SH, zVector_SH, stretch_SH, stretchP_SH] = vectorBuilder(
            LoShC, springCntSH, x,y,z,Dx,Dy,Dz);
583
584     [St1, St2] = springForceBuilder(N, M, xVector_ST, yVector_ST, zVector_ST, magVector_ST,
            springMap_ST, stretch_ST, stretchP_ST);
585     [Sh1, Sh2] = springForceBuilder(N, M, xVector_SH, yVector_SH, zVector_SH, magVector_SH,
            springMap_SH, stretch_SH, stretchP_SH);
586
587     [fixForces] = constrainedForcesSolver(N,M,gVector,pVector);
588
589     A = [ St1 Sh1 ]; %
590     b = -(gVector + pVector);% + fixForces;
591 end
592
593 %-------------------------------------------------------------------------
594
595 function [fixForces] = constrainedForcesSolver(N,M,gVector,pVector)
596
597     numBods = N^2+4*(N-1)^2;
598
599     fixedForcesX = zeros(numBods,1);
600     fixedForcesY = zeros(numBods,1);
601     fixedForcesZ = zeros(numBods,1);
602
603     charlie = (N^2 + (M-1));
604     delta = (M-1);
605     sigma = (N^2 + 4*(M-1)^2);
606     fixedPoints = charlie:delta:sigma;
607
608     gVector = reshape(gVector,3,[])';
609     pVector = reshape(pVector,3,[])';
610
611     for i=1:length(fixedPoints)
612         fixedForcesX(fixedPoints(i)) = -gVector(fixedPoints(i),1) -pVector(fixedPoints(i),1);
613         fixedForcesY(fixedPoints(i)) = -gVector(fixedPoints(i),2) -pVector(fixedPoints(i),2);
614         fixedForcesZ(fixedPoints(i)) = -gVector(fixedPoints(i),3) -pVector(fixedPoints(i),3);
615     end
616
617     fixForces = reshape([fixedForcesX fixedForcesY fixedForcesZ]',numBods*3,[]);
618
619 end
620
621 %-------------------------------------------------------------------------
622
623 function plotSprings(N, M, Q, springCnt)
624     [ x, y, z, ~, ~, ~, ~] = buildBedCoords(N,M,Q);
625
626     for i=1:length(springCnt)
```

```
627         springLine_X = [x(springCnt(i,1)) x(springCnt(i,2))];
628         springLine_Y = [y(springCnt(i,1)) y(springCnt(i,2))];
629         springLine_Z = [z(springCnt(i,1)) z(springCnt(i,2))];
630
631         plot3(springLine_X,springLine_Y,springLine_Z);
632
633         axis equal
634         hold on
635         pause(0.1);
636     end
637
638 end
```

REFERENCES

[1] J. Weil, "The synthesis of cloth objects," *Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, 1986.

[2] X. Provot, "Deformation constraints in a mass-spring model to describe rigid cloth behavior," *Graphics Interface*, vol. 23(19), 09 1995.

[3] D. Baraff and A. Witkin, "Large steps in cloth simulation," in *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '98. New York, NY, USA: Association for Computing Machinery, 1998, p. 43–54. [Online]. Available: https://doi.org/10.1145/280814.280821

[4] E. English and R. Bridson, "Animating developable surfaces using nonconforming elements," *ACM Transaction on Graphics*, 2008.

[5] G. M. Whitesides, "Soft robotics," *Angewandte Chemie International Edition*, vol. 57, no. 16, pp. 4258–4273, 2018. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/anie.201800907

[6] H. Hu, Q. Tian, and C. Liu, "Soft machines: Challenges to computational dynamics," *Procedia IUTAM*, vol. 20, pp. 10–17, 2017, 24th International Congress of Theoretical and Applied Mechanics. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2210983817300044

[7] S. Kim, C. Laschi, and B. Trimmer, "Soft robotics: a bioinspired evolution in robotics," *Trends in Biotechnology*, vol. 31, no. 5, pp. 287–294, 2013. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167779913000632

[8] B. Caasenbrood, A. Pogromsky, and H. Nijmeijer, "A computational design framework for pressure-driven soft robots through nonlinear topology optimization," in *2020 3rd IEEE International Conference on Soft Robotics (RoboSoft)*, 2020, pp. 633–638.

[9] A. Nealen, M. Müller, R. Keiser, E. Boxerman, and M. Carlson, "Physically based deformable models in computer graphics," *Computer Graphics Forum*, vol. 25, no. 4, pp. 809–836, 2006. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-8659.2006.01000.x

[10] G. Picinbono, H. Delingette, and N. Ayache, "Non-linear anisotropic elasticity for real-time surgery simulation," *Graphical Models*, vol. 65, no. 5, pp. 305–321, 2003, special Issue on SMI 2002. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1524070303000456

[11] K. W. Buffinton, B. B. Wheatley, S. Habibian, J. Shin, B. H. Cenci, and A. E. Christy, "Investigating the mechanics of human-centered soft robotic actuators with finite element analysis," in *2020 3rd IEEE International Conference on Soft Robotics (RoboSoft)*, 2020, pp. 489–496.

[12] W. Huang, X. Huang, C. Majidi, and K. Jawed, "Dynamic simulation of articulated soft robots," *Nature Communications*, vol. 11, 05 2020.

[13] S. Honji and K. Tahara, "Dynamic modeling and joint design of a cable driven soft gripper," in *2020 3rd IEEE International Conference on Soft Robotics (RoboSoft)*, 2020, pp. 593–598.

[14] A. Buso, R. Scharff, E. Doubrovski, J. Wu, C. Wang, and P. Vink, "Soft robotic module for sensing and controlling contact force," in *2020 3rd IEEE International Conference on Soft Robotics (RoboSoft)*, 2020, pp. 70–75.

[15] A. K. Narwal, A. Vaz, and K. Gupta, "Bond graph modeling of dynamics of soft contact interaction of a non-circular rigid body rolling on a soft material,"

Mechanism and Machine Theory, vol. 86, pp. 265–280, 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0094114X14003231

[16] B. K. Johnson, V. Sundaram, M. Naris, E. Acome, K. Ly, N. Correll, C. Keplinger, J. S. Humbert, and M. E. Rentschler, "Identification and control of a nonlinear soft actuator and sensor system," *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 3783–3790, 2020.

[17] H. Banerjee and Z. Tse, "Soft robotics with compliance and adaptation for biomedical applications and forthcoming challenges," *International Journal of Robotics and Automation*, vol. 33, 01 2018.

[18] J. Liang, M. Lin, and V. Koltun, "Differentiable cloth simulation for inverse problems," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32. Curran Associates, Inc., 2019. [Online]. Available: https://proceedings.neurips.cc/paper/2019/file/28f0b864598a1291557bed248a998d4e-Paper.pdf

[19] S.-W. Hsiao and R.-Q. Chen, "A method of drawing cloth patterns with fabric behavior," 01 2005.

[20] A. A. ElBadrawy and E. E. Hemayed, "Speeding up cloth simulation by linearizing the bending function of the physical mass-spring model," in *2011 International Conference on 3D Imaging, Modeling, Processing, Visualization and Transmission*, 2011, pp. 101–107.

[21] T. DeRose, M. Kass, and T. Truong, "Subdivision surfaces in character animation," 1995. [Online]. Available: https://graphics.pixar.com/library/Geri/paper.pdf

[22] M. Muller, M. Teschner, and M. Gross, "Physically-based simulation of objects represented by surface meshes," in *Proceedings Computer Graphics International, 2004.*, 2004, pp. 26–33.

[23] U. Meier, O. López, C. Monserrat, M. Juan, and M. Alcañiz, "Real-time deformable models for surgery simulation: a survey," *Computer Methods and Programs in Biomedicine*, vol. 77, no. 3, pp. 183–197, 2005. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0169260704002093

[24] K.-J. Choi and H.-S. Ko, "Stable but responsive cloth," *ACM Transactions on Graphics*, vol. 21, 07 2002.

[25] D. Eberle, "Better collisions and faster cloth for pixar's coco," 08 2018, pp. 1–2.

[26] E. Basafa, F. Farahmand, and G. Vossoughi, "A non-linear mass-spring model for more realistic and efficient simulation of soft tissues surgery," *Studies in health technology and informatics*, vol. 132, pp. 23–5, 02 2008.

[27] C. Schumacher, E. Knoop, and M. Bacher, "Simulation-ready characterization of soft robotic materials," *IEEE Robotics and Automation Letters*, vol. PP, pp. 1–1, 03 2020.

[28] K. L. Bouman, B. Xiao, P. Battaglia, and W. T. Freeman, "Estimating the material properties of fabric from video," in *2013 IEEE International Conference on Computer Vision*, 2013, pp. 1984–1991.

[29] S. Yang, J. Liang, and M. Lin, "Learning-based cloth material recovery from video," 10 2017, pp. 4393–4403.

[30] T. Runia, K. Gavrilyuk, C. Snoek, and A. Smeulders, "Cloth in the wind: A case study of physical measurement through simulation," 06 2020, pp. 10 495–10 504.

[31] A. Davis*, K. L. Bouman*, J. G. Chen, M. Rubinstein, O. Büyüköztürk, F. Durand, and W. T. Freeman, "Visual vibrometry: Estimating material properties

from small motions in video," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 4, pp. 732–745, 2017.

[32] K. Bauer, K. Rock, M. Nazzal, O. Jones, and W. Qu, "Pressure ulcers in the united states' inpatient population from 2008 to 2012: Results of a retrospective nationwide study," *Ostomy/wound management*, vol. 62, pp. 30–38, 11 2016.

[33] B. J. M. G. M. M. L. M. L. . S. M. Edsberg, L. E., "Revised national pressure ulcer advisory panel pressure injury staging system: Revised pressure injury staging system," *Journal of wound, ostomy, and continence nursing : official publication of The Wound, Ostomy and Continence Nurses Society*, vol. 43, p. 585–597, 6 2016.

[34] R. P. A. Reddy M, Gill S S, "Preventing pressure ulcers: a systematic review," *JAMA*, vol. 296, pp. 974–984, 8 2006.

[35] A. Nelson, J. Lloyd, N. Menzel, and C. Gross, "Preventing nursing back injuries: Redesigning patient handling tasks," *AAOHN journal : official journal of the American Association of Occupational Health Nurses*, vol. 51, pp. 126–34, 04 2003.

[36] R. Yousefi, S. Ostadabbas, M. Faezipour, M. Nourani, V. Ng, L. Tamil, A. Bowling, D. Behan, and M. Pompeo, "A smart bed platform for monitoring amp ulcer prevention," in *2011 4th International Conference on Biomedical Engineering and Informatics (BMEI)*, vol. 3, 2011, pp. 1362–1366.

[37] A. Bowling, *Vector Mechanics: A Systematic Approach*. Aqualan Press, LLC, 2018. [Online]. Available: https://books.google.com/books?id=5_bOuQEACAAJ

[38] P. Parihk, "Smart bed and contact impact of flexible bodies using energy compensation method," Master's thesis, The University of Texas at Arlington, 634 Nedderman Hall, Box 19019, 416 Yates Street, Arlington, TX 76019-0019, 2015.

[39] R. Kuicek, "Pressure ulcer prevention using soft non-grasp manipulation," Master's thesis, The University of Texas at Arlington: Honors College, College Hall, Suite 108, 600 S. West Street, Arlington, TX 76019, 2019.

BIOGRAPHICAL STATEMENT

Cristian Almendariz was born in Arlington, Texas USA, in 1995. He received his Honors B.S. degree and M.S. degree from The University of Texas at Arlington in 2018 and 2021, respectively, both in Mechanical Engineering. Since his early summers between grade school years at TexPREP, his interests have always included Robotics, 3D Printing, and other Engineering Technologies. Other non-technical interests and hobbies include: running, reading, and rock climbing with his beloved wife.