MODELING VISCOELASTIC BEHAVIOR IN COMPACT BONE THROUGH A
DISTRIBUTION OF COLLAGEN D-SPACING:
A FINITE ELEMENT ANALYSIS

A Thesis
presented to
the Faculty of California Polytechnic State University,
San Luis Obispo

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in Biomedical Engineering

by
Christopher Ha
November 2015

© 2015

Christopher Ha

ALL RIGHTS RESERVED

COMMITTEE MEMBERSHIP

TITLE: Modeling Viscoelastic Behavior in Compact Bone
through a Distribution of Collagen D-Spacing:
A Finite Element Analysis

AUTHOR: Christopher Ha

DATE SUBMITTED: November 2015

COMMITTEE CHAIR: Scott Hazelwood, Ph.D.
Professor of Biomedical Engineering

COMMITTEE MEMBER: Clifford Les, DVM, Ph.D.
Adjunct Professor of Biomedical Engineering

COMMITTEE MEMBER: Dan Walsh, Ph.D.
Adjunct Professor of Biomedical Engineering

ABSTRACT

Modeling Viscoelastic Behavior in Compact Bone through a Distribution of Collagen D-Spacing : A Finite Element Analysis

Christopher Ha

Osteoporosis affects nearly 54 million people in the United States. The cost associated with treatment is estimated to be \$19 billion per year and is expected to grow yearly. D-spacing is the staggering of collagen molecules found at the nanoscopic level. Previously thought to have a constant value, recent studies have found that D-spacing has a distribution of values throughout the tissue. As part of an ongoing effort in understanding the mechanisms that are affected by osteoporosis, a finite element model was developed to explore the effects of D-spacing distribution on the viscoelastic material properties of bone tissue. The goal of this computational model was to mimic the viscoelastic properties of different sectors of bone tissue that have been treated under different loading conditions (tension and compression).

An appropriate animal model was required to allow for the development of an accurate computational model. Although they don't exhibit similar hormonal cycles as humans, sheep are an excellent animal model for bone research as they experience Haversian bone remodeling, are docile, relatively inexpensive, and have skeletons similar in size and mechanical properties to humans. For this study, six Rambouillet-cross ewes were either ovariectomized (OVX) or underwent a sham surgery (control). After twelve months post-surgery, the ewes were euthanized and rectangular beam bone samples were collected from different sectors of the ulna/radius bones. Dynamic mechanical analysis was performed on these samples and the viscoelastic property, tangent delta, was measured from each analysis at varying frequencies.

Using experimental measurements, the Composite Model was developed on finite element analysis software, Abaqus. The model was generated through a Python script that uses experimental D-spacing mean and standard deviation data to create a large two-dimensional model composed of two hundred collagen and hydroxyapatite complexes with varying D-spacing lengths. Multiple security measurements were implemented to ensure biological relevance. Collagen was assigned viscoelastic material properties through a user subroutine material property. Four models for each sector of interest (caudal and cranial) were generated. Each model was loaded under appropriate loading conditions and tangent delta was recorded for each test frequency.

Results from the Composite Model matched the experimental data more accurately than previous computational models, suggesting a superior model. The results implied that a large network of collagen and hydroxyapatite complexes in series and parallel are effective at modeling bone under different loading conditions. This computational model shows promise in the bone research field. A lot of flexibility was implemented in the model development process, making refinements easy to be performed. This study provides a stepping-stone in computational tooling on examining the effects of metabolic bone diseases on

viscoelasticity.

Keywords: D-spacing, Compact Bone, Finite Element Analysis, Viscoelasticity, Collagen
Quaternary Structure

ACKNOWLEDGMENTS

I would like to thank Dr. Scott Hazelwood and Dr. Clifford Les for all of their support and guidance in my education and various projects. Their vast knowledge in this field is astonishing and I'm grateful that they were helping me along the way. I also want to thank my family and friends for their endless encouragement throughout my life.

TABLE OF CONTENTS

	Page
LIST OF TABLES	ix
LIST OF FIGURES	xii
CHAPTER	
1 Introduction	1
1.1 Purpose	1
1.2 Bone Tissue Background	2
1.2.1 Purpose and Function	2
1.2.2 Compositon	2
1.3 Structure	3
1.3.1 Macro-scale (down to 50 microns)	3
1.3.2 Micro-scale	5
1.3.3 Nano-scale	7
1.4 Modeling and Remodeling	9
1.5 Osteoporosis	13
1.5.1 Post-menopausal Osteoporosis	14
1.5.2 Treatments	14
1.6 Material Properties of Bone	16
1.6.1 Modeling Bone Tissue	17
1.6.2 Viscoelasticity	18
1.6.3 Modeling Viscoelasticity	22
1.7 Animal Models	23
1.7.1 Sheep Models for Osteoporotic Research	23
1.8 Study Objective	25
2 Methods	26
2.1 Experimental Data	26
2.1.1 Sample Preparations	26
2.1.2 Mechanical Testing	26
2.2 Model Evolution	28
2.2.1 Previous Models	28
2.2.2 Siegmund Model	29
2.2.3 Half Unit Cell Model	32
2.3 Custom Viscoelastic Material Definition	33
2.3.1 Parameter Definition	35
2.4 Complex Model	36
2.4.1 Model Builder	37
2.4.2 Loading and Boundary Conditions	38
2.4.3 Material Assignment	39
2.4.4 Element Type	40
2.4.5 Model Validation	40
2.4.6 Mesh Development	40
2.5 Post Processing	43

2.6 Statistical Analysis	43
3 Results	45
4 Discussion	60
5 Conclusion	71
BIBLIOGRAPHY	73
APPENDICES	
APPENDIX A Experimental Data	80
APPENDIX B User Subroutine (UMAT)	83
APPENDIX C Viscoelastic Equations	91
APPENDIX D Loading Condition: Stress Estimation	93
APPENDIX E Input File and Modifications	95
APPENDIX F Composite Model Results	99
APPENDIX G Statistcal Analysis Results	106
APPENDIX H Python Script: Data Retrieval	118
APPENDIX I MATLAB Scripts: Converting data	121
APPENDIX J Python Script: Model Generation	129

LIST OF TABLES

Table	Page
1 Siegmund Model Dimensions. These dimensions describe the details of the model.	31
2 Material Property Definitions of Hydroxyapatite and Collagen as in the Siegmund Model.	31
3 Half Unit Cell Model Sample Dimensions. The model developed by Cal Poly graduate Miguel Mendoza explored modeling a half unit cell at multiple D-spacing values.	33
4 Governing D-spacing means and standard deviations of cranial and caudal sectors that were used to develop the complex model. These values were used for the 'Random.Gaussian' function in the Python script. Values were provided by the University of Michigan, Ann Arbor. 'Control' indicates ovine that were subjected to a sham surgery. These sheep did not have their ovaries removed and acted as a control to the ovariectomized sheep.	38
5 Composite Model Row Lengths. Due to the random function implemented in D-spacing lengths, models had varying top and bottom row lengths. Four sub-models were produced for each model (Control, Cranial and Control, Caudal).	38
6 Summary Table for Convergence Study.	42
7 A Summary Table for Statistical Treatments.	44
8 Summary Table for Control Cranial DMA Testing on Six Rambouillet-cross Ewes. Data were provided by Henry Ford Hospital.	49
9 Summary Table for Control Caudal DMA Testing on Six Rambouillet-cross Ewes. Data were provided by Henry Ford Hospital.	49
10 Summary Table for Control, Cranial Testing. Experimental data were collected via DMA testing on six Rambouillet-Cross Ewes. Composite Model data were obtained through averaging tangent delta values for each test frequency. Reported data from Mendoza's "Normal D-spacing" Half Unit Cell Model were used as comparison [52]. Standard deviation and standard error can't be reported for the Half Unit Cell Model due to a lack of randomization or replicates. A dashpot with a viscosity of 1.25 GPa*s were used in models.	52
11 Summary table of RMSE and R^2 values for cranial Composite and Half Unit Cell models.	53

12	Summary Table for Control, Caudal Testing. Experimental data were collected via DMA testing on six Rambouillet-Cross Ewes. Composite Model data were obtained through averaging tangent delta values for each test frequency. Reported data from Mendoza's "Normal D-spacing" Half Unit Cell Model were used as comparison [52]. Standard deviation and standard error can't be reported for the Half Unit Cell Model due to a lack of randomization or replicates. A dashpot with a viscosity of 1.25 GPa*s were used in models.	54
13	Summary table of RMSE and R^2 values for caudal Composite and Half Unit Cell models.	56
14	Two-Way ANOVA Output from SAS for Control, Cranial Sheep. A 5% overall significance level was utilized to determine the effects of each statistical treatment and their interactions. At a p-value of 0.0284, the interaction between Model Type and Frequency was found to be not significant in explaining the effects of tangent delta. Thus conclusions on the main effects can only be made.	57
15	Two-Way ANOVA Output from SAS for Control, Caudal Sheep. A 5% overall significance level was utilized to determine the effects of each statistical treatment and their interactions. At a p-value of less than 0.0001, the interaction between Model Type and Frequency was found to be significant in explaining the effects of tangent delta. Thus conclusions on purely the main effects can't be made.	58
16	Experimental Mechanical Testing. Data were collected from DMA testing across six test sheep. Testing was performed at Henry Ford Hospital. . . .	81
17	Tangent Delta Means and Standard Deviations for Experimental Data. Testing was performed at Henry Ford Hospital and provided to this current study.	81
18	Experimental D-Spacings. Atomic force microscopy was utilized to record collagen periodicity across 6 test sheep. AFM work was performed by the University of Michigan, Ann Arbor.	82
19	Table of Material Properties in Rheological Elements. The values highlighted in blue represents different parameters for the Standard Linear Solid. Below are the names assigned in the code, their description, and value. . .	96
20	Results from the Control, Cranial Composite Model with a 1.25 GPa*s Dashpot. Four versions of the Control, Cranial model were created and tested ten times before reporting an averaged Tangent delta.	100
21	Results from the Control, Caudal Composite Model with a 1.25 GPa*s Dashpot. Four versions of the Control, Caudal model were created and tested ten times before reporting an averaged Tangent delta.	101

22	Results from the Control, Cranial Composite Model with a 4.00 GPa*s Dashpot. A single version of Control, Cranial was run across all test frequencies ten times to be used as a comparison between the effects of a large viscosity dashpot.	102
----	--	-----

LIST OF FIGURES

Figure	Page
1 Graphical representation of bone in the macroscale(cortical bone) being broken down to the nano scale(collagen molecule) [12].	3
2 Structure of trabecular and cortical bone. Trabecular bone can be seen on the left. Note the dark regions that denote the large pores. Cortical bone can be seen surrounding trabecular bone [7].	4
3 Microradiograph of compact bone. Resorption cavities are seen in large black circles. Haversian canals are seen in smaller black circles [7].	5
4 (A) Three-dimensial representation of helicoidal structure.(B) Visual effect of arches when with a sectional view cut as denoted by shaded plane in A.(C) The arches seen in the sectional view formed by helicoidal lamellae [7].	6
5 Photomicrograph of plexiform bone [7].	7
6 Distribution of D-spacing in ovine compact bone. These ovine have experienced a sham surgery or an ovariectomy(OVX) to induce estrogen depletion replicating the effects of postmenopausal osteoporosis [16].	8
7 Cumulative distribution functions of collagen 1 fibril D-spacing of Sham (control) and OVX (ovariectomy) ovine at different anatomical sites. The distribution of D-spacing seems to become more homogeneous in the bone tissue after inducing estrogen depletion [20].	9
8 Microcrack shown in cortical bone pointed out by black arrows. Under typical loading conditions, bone will develop micro-damage. The micro-damage may propagate into a larger crack if insufficient remodeling is performed to the bone. These can be catastrophic if a large enough load is applied to fracture the bone [7].	10
9 A three dimensional view of cortical bone with an exploded view of one of many osteons. Notable features of osteons are their Haversian canal and cement lines. Haversian canals are in the center and provide vasculature during the remodeling process. They are not filled after remodeling is done. Thus they are accountable for some of the porosity in bone. Cement lines are the interface between the osteon and the surrounding bone. Although their interface is weak in that they may slip between each other, they increase the amount of energy it takes for a crack to propagate through the bone [7].	11

10	Cross-sections of a BMU. Osteoclasts will be at the leading edge (left of cross-section) resorbing bone at a rate of 40 microns per day. Osteoblasts will be at the trailing-end forming bone at a slower rate. They will lay down layers of lamellae at a slower rate of 1 to 2 microns per day. A Haversian canal can be seen at the very right of cross-section. Note how the BMU tunnels through the bone [7].	12
11	Comparison between normal(left) and osteoporotic(right) trabecular bone. The osteoporotic bone has larger porous sites, thus having a less bone mass [28].	13
12	Stress-strain curves of two samples of trabecular bone with varying porosities. Loading rises at the end due to the collapsing of pores [7].	17
13	Depiction of Voigt(left) and Reuss(right) models for a composite material formed by mixing two materials. In the Voigt model, the materials experience the same amount of strain, while the materials in the Reuss model experience the same amount of stress. Both use the volume fraction and elastic modulus of each material used to determine the composite modulus [50].	18
14	Stress-strain curves of bovine femoral cortical bone in compression at varying strain frequencies. The larger the strain rate, a higher modulus can be seen [56].	19
15	Stress and strain response curves during DMA testing. The stress curve applied by the testing can be seen in the dashed red curve. The strain response is seen in black. The tangent delta can be seen as the lag between the stress to the strain curve [55].	20
16	Spring and dashpot elements (top) used in a system to represent viscoelastic material properties. Additionally, some examples of the different schematics are displayed on the bottom portion that they can possibly be configured too [6].	22
17	Images of histological cross-sections of ewe cortical bone at different ages. The left is of a three year old ewe and primarily composed of plexiform bone due to its quickly needed growth. The right is of an older ewe. Note the heavy remodeling on the older ewe that's similar to adult human cortical bone [71].	24
18	Cross section of collected ewe bone samples. Anatomical sections were divided by white lines. Oftentimes, the radius and ulna are fused together. Multiple beams can be created from a single sector. A beam was chosen at random for mechanical testing [79].	27

19	A comparison between the Hodge-Petruska and staggered array models. Note the different arrangement of collagen and hydroxyapatite. The Hodge-Petruska model can be seen on the left and the staggered array model can be seen on the right. The staggered array model portrays the characteristic D-spacing behavior resulting from the overlap and gap regions [83].	29
20	(a) Simplified two-dimensional representation of the staggered array model. Collagen sections are light-grey and hydroxyapatite sections are depicted as black boxes.(b) Periodic arrangement of collagen and hydroxyapatite sections are shown along with the tensile loading condition.(c) Graphic representation of a single unit cell that Siegmund developed shown with its characteristic dimensions. The definitions of these characteristic dimensions and their assumed values are described in table 1.(d) Half unit cell of Siegmund model that was used in their computational model [51].	30
21	(a) Mendoza's Half Unit Cell Model highlighting the hydroxyapatite section in pink and the collagen section in grey. (b) Model shown on Abaqus' global axis. (c) Model and its set boundary and loading conditions [52]. . .	32
22	(Left)Kelvin-Voigt body form of the Standard Linear Solid [90]. (Right) The three-dimensional version of the rheological model where the spring and dashpot portions are replaced by a shear and bulk modulus [52, 90]. . .	34
23	(a)Left end of the Composite Model. The x-symmetry boundary conditions can be seen on the left vertical edge. Additionally, an y-symmetry boundary condition is applied throughout the bottom horizontal edge. (b) A distributed load is applied on the right edge of the Composite Model. The direction of the load is dependent on the model sector. (c) Two materials were assigned to the sections of the model. The salmon colored sections represent the mineralized hydroxyapatite. Green sections were assigned as collagen and their respective material property.	39
24	The node that was probed for mesh convergence is denoted with the red marker.	41
25	After multiple iterations of the test model at different seed sizes, it was determined that a seed size of 0.0005 mm or roughly 1,300,000 degrees of freedom would produce adequate results without wasting computational time/power.	42
26	Tangent delta is calculated between the loading cycle (black line) and the deformation cycle (red dashed line). Time is on the x-axis while the y-axis is the normalized stress/strain.	46
27	The best-fit curve for a single run for a Control, Caud Composite model at 1 Hz from the MATLAB code. The R^2 value of 0.9999 denotes an accurate fit of the data to a sinusoidal curve. Tangent delta for a single run (one of ten) can be seen on the bottom right as well. The MATLAB code was run ten times and all tangent delta values were averaged.	47

28	Comparisons between different dashpot values for a Control Cran replicate in order to view the effects of a more viscous dashpot has on tangent delta. The findings by Mendoza were confirmed that a dashpot with a viscosity 1.25 GPa*s produces data more similar to the experimental values than a dashpot of a viscosity of 4 GPa*s.	48
29	Four replicate Control, Cranial models were developed for this study. All replicates were run at all test frequencies. The average tangent delta for each test frequency will be utilized in statistical analysis.	49
30	Four replicate Control, Caudal models were developed for this study. All replicates were run at all test frequencies. The average tangent delta for each test frequency will be utilized in statistical analysis.	50
31	Plot for all tangent delta values from DMA experimental data, Mendoza's "Normal D-spacing" Half Unit Cell Model results [52], and the Composite Model for Control, Cranial sectors. The tangent delta data from the Composite Model were averaged for each frequency. Standard error bars were applied to each test frequency for experimental and Composite Model results. A dashpot with a viscosity of 1.25 GPa*s were used in models.	51
32	A linear best-fit trend line was used to fit the Composite Model and Experimental tangent delta results. An R^2 of 0.8835 implies a high correlation between experimental and Composite Model results. A dashpot with a viscosity of 1.25 GPa*s were used in models.	53
33	Plot for all tangent delta values from DMA experimental data, Mendoza's "Normal D-spacing" Half Unit Cell Model results [52], and the Composite Model for Control, Caudal sectors. The tangent delta data from the Composite Model were averaged for each frequency. Standard error bars were applied to each test frequency for experimental and Composite Model results. A dashpot with a viscosity of 1.25 GPa*s were used in models.	55
34	A linear best-fit trend line was used to fit the Composite Model and Experimental tangent delta results. An R^2 of 0.8000 implies a high correlation between experimental and Composite Model results. A dashpot with a viscosity of 1.25 GPa*s were used in models.	56
35	Main effect plots were created to view the effects the explanatory variables have on tangent delta. A) Main effect plot for Test Frequency. B) Main effect plot for Model Type. Data points with like markers are not significantly different from one another.	58
36	An interaction plot was developed to view at which test frequency the Composite Model and experimental models differ. The interaction of Model Type and Test Frequency displayed. Data points with like markers are not significantly different from one another.	59

37	The Kelvin-Voigt version of the Standard Linear Solid exhibits both a creep and stress relaxation response [90].	91
38	Experimental data vs. Control, Cranial Composite Model data. A linear line was fitted onto the plot. Plotted Composite Model data are averages for each frequency. $R^2=0.94211$	103
39	Experimental data vs Mendoza data for Control, Cranial. A linear line was fitted onto the plot. $R^2=0.87399$	104
40	Experimental data vs. Control, Caudal Composite Model data. A linear line was fitted onto the plot. Plotted Composite Model data are averages for each frequency $R^2=0.88327$	105
41	Experimental data vs Mendoza data for Control, Caudal. A linear line was fitted onto the plot. $R^2=0.86812$	105

1 Introduction

1.1 Purpose

Bone is a highly adaptive organ that is naturally under mechanical and physiological stress through everyday activity. The shape and composition of bone is highly efficient in supporting and protecting organs. However, like many other organs, if the surrounding chemical milieu is slightly altered, the multiple parameters of bone can also be altered and may cause devastating effects. These changes can include viscoelastic properties, which can result in bone diseases such as osteoporosis [1].

Nearly 54 million people in the United States are affected by osteoporosis. It has been estimated that about \$19 billion dollars are exhausted each year for treatment and this number is projected to grow to \$25.3 billion by 2025 [2]. Unfortunately, a majority of individuals are unaware of having osteoporosis until they experience a fracture related to the disease. This is because there may not be considerable symptoms when osteoporosis is developing in an individual [2, 3].

Bone tissue has a unique material property that allows it withstand different loading conditions. This material property is described as viscoelastic, where it exhibits both viscous and elastic characteristics under deformation. Therefore, they demonstrate a time-dependent strain. However, understanding the viscoelastic material properties of bone have been an ongoing effort. The staggering of collagen molecules also known as D-spacing has been a relatively recent point of interest in understanding viscoelasticity in skeletal tissue [4]. The use of computational models can contribute in understanding how a varying distribution in D-spacing can affect viscoelasticity. This can then provide a link between D-spacing and osteoporosis. For this specific study, a computational model of different sectors of ewe bone ulna/radius will be developed. Since bone sectors are loaded differently, their distribution of D-spacing will slightly vary sector to sector. This study will look into the loading condition (tensile vs compression) effects on viscoelastic properties. A greater understanding can lead to more efficient time and money put into developing better diagnostics tools and therapeutic-preventative medicine. However, before any kind of progress

can be made, a firm grasp in bone structure and function must be understood.

1.2 Bone Tissue Background

1.2.1 Purpose and Function

Bones serve many functions throughout the body. They act as a calcium reservoir for the rest of the body, store blood cells that produce bone marrow, protect vital organs, supply a grounds for our movements, and provide structural support [5, 6]. Bone is particularly known for its mechanical functions. What makes bones especially well fitted for mechanical loading is their calcified structure, adaptive shape, and viscoelastic properties. These material properties will allow the bone to deform accordingly.

Bone is an adaptive biomaterial. It will constantly adjust to its physiologic and mechanical environment [7]. Bones are often experiencing mechanical loading due to everyday activities. In response to changes in mechanical stimulation, bone will alter its shape/architecture accordingly. The magnitude of these loads due to body weight is significantly larger than one should suspect. This adaptive nature is due to bone modeling and remodeling, which will be discussed in section 1.4.

1.2.2 Composititon

Like many composite materials, knowing the composition of bone may provide an understanding for how the viscoelastic bone material behaves. There are three notable constituents in bone: collagen, water, and hydroxyapatite. Collagen is a structural protein that gives bone flexibility and tensile strength [7]. There are 28 identified types of collagen [8]. The collagen in bone is largely composed of type 1 collagen, which is also found in tendons, ligaments and skin. Hydroxyapatite accounts for the stiffness and compressive strength in bone. The hydroxyapatite is embedded into the collagen matrix. They are a calcium version that are in a shape of rods or plates 400 angstroms long [9, 10, 11]. There are some other accountable constitutes in bone, like proteoglycans and noncollagenous proteins; however, they play a lesser role in the mechanical properties of bone.

1.3 Structure

Bone tissue can be categorized as a composite material. Because of this, the structure largely contributes to the material properties of bone. There are many hierarchies in bone demonstrated in figure 1. Each hierarchy can attribute to the viscoelastic properties of bone. This study will focus on the nanoscale as a main contributor to these properties.

Figure 1: Graphical representation of bone in the macroscale(cortical bone) being broken down to the nano scale(collagen molecule) [12].

However, an understanding of the macro and micro scales of bone structure should be known as well.

1.3.1 Macro-scale (down to 50 microns)

Figure 2: Structure of trabecular and cortical bone. Trabecular bone can be seen on the left. Note the dark regions that denote the large pores. Cortical bone can be seen surrounding trabecular bone [7].

In the macro scale, there are two different classes of bone types (figure 2) typically categorized by their porosities. If the bone has a porosity of 75-95%, they are considered trabecular or cancellous bone [7]. The pores of this bone are filled with red or yellow marrow depending on the anatomic site [6]. Red bone marrow is composed of blood vessels, nerves, and other cells [7], while yellow bone marrow contains adipose tissue and primitive blood cells [6]. The bone matrix of cancellous bone is in the form of struts or plates called trabeculae that are about 200 microns thick. They often seem to be arranged randomly. But sometimes they can be seen organized into orthogonal arrays.

The second type of bone tissue is compact or cortical bone, which typically has a porosity of 5 to 10%. Compact bone is often found around shafts of long bone and acts as a shell around spongy bone. Their porosity categorized by three different kinds of cavities or canals, two of which are observed in figure 3. The first is a Haversian canal that is aligned

along the long axis of bone. They are typically 50 microns in diameter and contain capillaries and nerves [7]. Volkmann's canals are shorter than Haversian canals, as they act as a connector between Haversian canals and outer surfaces of the bone. They too contain blood vessels and nerves. Last of the pores in compact bone are the resorption cavities. These cavities are temporary pores created by osteoclasts in the process of remodeling. They are much larger than the two canals and measure about 200 microns in diameter.

Figure 3: Microradiograph of compact bone. Resorption cavities are seen in large black circles. Haversian canals are seen in smaller black circles [7].

It is to be noted that since bone is an adaptive biomaterial, its porosity is subject to change over time. The body will naturally add, remove, and replace bone when needed by means of modeling and remodeling. A more in-depth discussion of modeling and remodeling will be discussed in a section 1.4.

1.3.2 Micro-scale

There are four primary types of bone tissue in the micro scale. The first two types, *lamellar* and *woven* bone, may be found in compact and trabecular bone. *Lamellar* bone consists of multiple highly organized parallel layers or lamellae that are slowly formed at around 1 micron per day [13]. Each layer is about 5 microns thick and comprised of an anisotropic matrix of mineral crystals and collagen fibers. There are two different types of structures

that are used to describe the orientation of how these layers are formed, which is often described as plywood-like [14].

The structure corresponds with the classical view of lamellar structure; each layer is laid down 90° from the previous. In other words, as a new lamella is starting it will change its direction to 90° of the lamella that is adjacent to it. The second type is referred to as the helicoidal plywood structure seen in figure 4. In this type, the fibers are continuously changing their direction so that there are no individual lamellae. It still describes a lamellar structure because as the orientation does a 180° cycle, the orientation repeats itself.

Figure 4: (A) Three-dimensional representation of helicoidal structure.(B) Visual effect of arches when with a sectional view cut as denoted by shaded plane in A.(C) The arches seen in the sectional view formed by helicoidal lamellae [7].

The second type of bone tissue that may be found in trabecular and compact bone is *woven* bone. Although lamellar bone is formed slowly and organized, woven bone is the opposite. It is formed quickly and poorly organized at sometimes more than 4 microns per day [15]. Collagen fibers and mineral crystals in woven bone are randomly arranged, which may cause the bone to be more mineralized than lamellar bone. In short, *woven* bone can be made more quickly and is weaker than *lamellar* bone.

The last two bone tissue types are categories of compact bone called *plexiform* and *Haversian* bone. Plexiform, or laminar, or fibrolamellar bone, seen in figure 5 is often found in large mammals. When bone needs to quickly increase in size faster than *lamellar* bone can be formed, woven bone must be produced instead. However, as discussed before, the

mechanical properties of woven bone is inferior to that of *lamellar* bone. To mend this, the *woven* bone is partially replaced with *plexiform* bone. Essentially, the lack of mechanical strength provided by *woven* bone is replaced with more advantageous *lamellar* bone [15].

Figure 5: Photomicrograph of plexiform bone [7].

The last of the bone tissue types is *Haversian* bone, which is a form of lamellar bone. *Haversian* bone is the result of the remodeling process. In remodeling, secondary osteons are formed that consist of cylindrical lamellae that surround Haversian canals as discussed previously.

1.3.3 Nano-scale

At the lowest scale, bone can be seen as a collagen matrix embedded with hydroxyapatite mineral crystals that provide strength. The individual collagen fibrils are composed of tropocollagen molecules [5]. When aligned, these molecules create gapped regions in the collagen network. In the early stages of mineralization, these gapped regions are filled with water and then replaced with mineralized crystal [7]. The crystal then binds to surrounding collagen molecules. This process essentially pushes the water out. As a result of this action, the tropocollagen molecules are displaced and offset by approximately 67 nm. This offset result is called D-spacing or D-period [5, 15]. Although 67 nm is an accepted value commonly used to define D-spacing, recent studies have found that D-spacing is not a constant value, but more of a distribution of values throughout the tissue [16, 17, 18]. A sample of this distribution can be seen in figure 6. In this current study, the distribution of

varying D-spacing values will be applied into a finite element model.

Figure 6: Distribution of D-spacing in ovine compact bone. These ovine have experienced a sham surgery or an ovariectomy(OVX) to induce estrogen depletion replicating the effects of postmenopausal osteoporosis [16].

First characterized in 1942 [19], D-spacing has been severely neglected by researchers since it was first found [4]. However, due to advancements in microscope technology, there has been an increase in focus of this dimensional characteristic. Additionally, a study found that this distribution is altered due to diseases such as postmenopausal osteoporosis [4]. Specifically, the effects of long-term estrogen depletion causes a decrease in overall D-spacing. Additionally, induced postmenopausal osteoporosis produces a more homogeneous material as seen in 7 [20], thus suggesting a biological significance in D-spacing. For this study, D-spacing is assumed to be a large contributor to the viscoelastic properties of bone tissue because the length of the offset is associated with collagen distribution and the amount of mineralization. The relationship between collagen and viscoelasticity will be discussed in section 1.6.

Figure 7: Cumulative distribution functions of collagen 1 fibril D-spacing of Sham (control) and OVX (ovariectomy) ovine at different anatomical sites. The distribution of D-spacing seems to become more homogeneous in the bone tissue after inducing estrogen depletion [20].

1.4 Modeling and Remodeling

As stated before, bone is a highly adaptive material that is constantly changing to its mechanical and chemical environment. There are two primary methods that this is done: modeling and remodeling.

Bone architecture is dependent on loading conditions and constructed by means of modeling. Since the loading conditions vary among individuals, each skeleton is essentially customized. However, the rate of modeling is greatly reduced after skeletal maturity is achieved. In modeling, bone is absorbed/removed by osteoclasts and formed by osteoblasts. Although forming and removing bone may be done at the same time, osteoclasts and osteoblasts work independently from one another in that osteoblasts will form bone where it's needed and osteoclasts will remove bone where it's needed.

Unlike modeling, remodeling occurs throughout one's lifetime. Because bone is often under load, microscopic damage commonly forms. Although they are not initially dangerous, damage can propagate into microcracks (figure 8) and possibly cause fatigue fractures. To prevent this from occurring, remodeling is done to remove portions of older bone and replaces it with newly formed bone.

Figure 8: Microcrack shown in cortical bone pointed out by black arrows. Under typical loading conditions, bone will develop micro-damage. The microdamage may propagate into a larger crack if insufficient remodeling is performed to the bone. These can be catastrophic if a large enough load is applied to fracture the bone [7].

Remodeling is performed by basic multicellular units (BMUs). Remodeling allows for new bone to be formed in replacement for damaged bone. The newly formed secondary bone comes in the form of an osteon as seen in figure 9. Although remodeling replaces damaged bone, remodeling does decrease the mechanical properties and increase the ductility of bone [7, 21]. This is because the porosity increases due to the Haversian canals and a mineralization lag time makes the bone vulnerable at times. However, without remodeling it has been suggested that human bone will fracture in 3 to 5 years [7]. A notable feature in osteons is their cement lines where the osteon and the surrounding bone interface with one another. This interface may arrest cracks if they were to occur, requiring more energy for cracks to propagate.

Figure 9: A three dimensional view of cortical bone with an exploded view of one of many osteons. Notable features of osteons are their Haversian canal and cement lines. Haversian canals are in the center and provide vasculature during the remodeling process. They are not filled after remodeling is done. Thus they are accountable for some of the porosity in bone. Cement lines are the interface between the osteon and the surrounding bone. Although their interface is weak in that they may slip between each other, they increase the amount of energy it takes for a crack to propagate through the bone [7].

Instead of having osteoclasts and osteoblasts work independently from one another, they work together in a BMU (figure 10) during remodeling. Remodeling can be broken down into osteonal phases known as the *ARF* sequence [7]. The *ARF* sequence consists of phases of activation, resorption, and formation with some other intermediate stages in-between. The first phase is *activation*, where osteoclasts are formed by a chemical or mechanical signal. Next, *resorption* occurs where the osteoclasts resorb the damaged bone in a tunneling fashion. The osteoclasts resorb bone at a rate of 40 microns per day at a diameter of 200 microns. After the osteoclasts would pass a particular point in the bone, osteoblasts would be recruited (*Reversal*). They would then begin *formation* by filling the tunnel with newly formed bone in layers of lamellae at a significantly slower pace of about 1 to 2 microns per day. However, the tunnel is not completely filled. Haversian canals, as mentioned in section 1.3.1, are left to provide a passageway for the vascular supply to be maintained

throughout the osteon.

Figure 10: Cross-sections of a BMU. Osteoclasts will be at the leading edge (left of cross-section) resorbing bone at a rate of 40 microns per day. Osteoblasts will be at the trailing-end forming bone at a slower rate. They will lay down layers of lamellae at a slower rate of 1 to 2 microns per day. A Haversian canal can be seen at the very right of cross-section. Note how the BMU tunnels through the bone [7].

Following the formation phase, the *mineralization* stage is initialized. Before any kind of mineral is deposited, a delay known as the mineralization lag time which normally last around 10 days [7]. Then, mineral is deposited into the organic bone matrix (osteoid). However, only 60% of the total mineral is laid initially. The remaining mineral is laid over the course of about six months. The mineral in this step contributes to the stiffness of the bone [7, 22]. The last of the *ARF* sequence is the *quiescence* phase. In this phase, osteoclasts retreat and osteoblasts may differentiate into osteocytes, bone lining cells, or disappear by adipose. The resulting feature is newly formed bone composed as an osteon. The total reforming process takes about 200 days for a section of bone.

1.5 Osteoporosis

Osteoporosis is the most common metabolic bone disorders that affects over 200 million individuals worldwide [23]. There are typically no symptoms in the early stages of bone loss. However, as bone loss accumulates, individuals may experience some symptoms. These symptoms include: backaches and loss of height. Osteoporosis increases the susceptibility of fracture [24]. About 50% of women and 20% of men older than 50 years will experience a fracture in their remaining lifetime due to osteoporosis [25]. In white women, the risk of a hip fracture is much greater than developing breast cancer [26].

Osteoporosis is characterized by low bone mass [7](figure 11). This is caused by a disruption of bone microarchitecture, loss of trabecular bone, decreased bone thickness, and increase cortical bone porosity [27]. It is highly accepted that the decrease in bone mass is due to an imbalance between bone resorption and formation in remodeling [27]. However, the underlying mechanisms for osteoporosis are less understood.

Figure 11: Comparison between normal(left) and osteoporotic(right) trabecular bone. The osteoporotic bone has larger porous sites, thus having a less bone mass [28].

There are methods to evaluate individuals for osteoporosis. Some risk factors are advanced age, history of osteoporotic fracture, low body weight, family history, etc. [27, 29, 30]. The evaluation usually screens for bone mineral density (BMD). To measure BMD, imaging the bone is done through dual energy x-ray absorptiometry (DXA) [31]. The DXA

is compared to the mean value for a young adult reference population and expresses the difference as a standard deviation producing a T-score [29, 32, 33]. From there, the T-score can be used to determine whether or not an individual has osteopenia, osteoporosis, etc.

1.5.1 Post-menopausal Osteoporosis

During menopause, the ovaries stop functioning as an endocrine organ resulting in a decrease of estrogen and progesterone. A decrease in these hormones will cause an imbalance in bone metabolism [34, 35]. The loss of estrogen increases the amount of remodeling over time where more bone is resorbed than formed [34] especially in the first two years of menopause [35].

It has been firmly established that estrogen loss is the main contributor to continual bone loss during menopause. When estrogen is depleted, an increase of osteoclast formation occurs by cytokines. This isn't entirely understood but some osteoclastogenic factors including RANKL, M-CSF, and tumor necrosis factor alpha (TNF- α) play large roles in bone loss post-menopause. Specifically, they increase the proliferation, lifespan, and capacity of osteoclasts, thus greatly increasing the resorption of bone [36].

1.5.2 Treatments

Despite the available treatment options for osteoporosis, many individuals believe that avoiding osteoporosis from developing in the first place is the best plan of action. There are many ways an individual can decrease their likelihood of developing osteoporosis. One option is to develop a lifestyle that naturally increase one's bone mass. Although bone mass is largely determined by genetics, individuals can further increase their bone mass with plenty of exercise, proper calcium and vitamin D intake, limited alcohol consumption, etc. [37]. However, this becomes difficult because bone mass growth is largely more influential in young adults, thus making this approach limited for a relatively short amount of time. It is more likely pharmaceuticals are used to treat older subjects.

Treatments for osteoporosis can be categorized into two categories: anti-resorptive and anabolic. The effects of both treatments are the same: to increase the user's BMD. As stated previously, increasing one's BMD will decrease the chances of fracture. Anti-resorptive

agents will reduce the amount of resorption done in remodeling, thus improving the balancing act between resorption and formation. On the other hand, anabolic agents will increase the rate of formation causing it to essentially catch-up to the increased rate of resorption. The kind of treatment an individual is prescribed is dependent on a variety of factors such as age, gender, severity of bone loss, history of blood clots, etc.

Bisphosphonates like Alendronate and Zoledronate are commonly used anti-resorptive agents. Commonly considered the standard for osteoporosis prevention and treatment; they reduce the amount of resorption done by either inhibiting prenylation or inducing osteoclast apoptosis [38]. There are some long-term effects generated by bisphosphonates if used in large doses. When used, bisphosphonates will accumulate in bone tissue over a long period of time. This would mean that their effects on bone resorption would continue long after treatment has stopped. Additionally, they have the ability of counteracting the effects of anabolics, prohibiting the user to switch treatment options. There have also been uncommon events of gastrointestinal tract problems and esophagitis due to bisphosphonate usage. Long-term use of bisphosphonates have been linked with an increase of atypical stress fractures in the femoral shaft [39, 40].

Other examples of anti-resorptive agents are selective estrogen receptor modulators (SERMs) and denosumab. SERMs inhibit bone resorption by the same mechanism as estrogen. The most widely used SERM, raloxifene has shown great results in reducing the vertebral fractures [41]. Additionally, SERMs have been found to reduce the risk of invasive breast cancer to their users [42]. However, there are some major safety concerns in the use of raloxifene therapy. There have been associations with an increased risk of venous thromboembolism, pulmonary embolism, and fatal stroke with the usage of raloxifene [42, 43]. A relatively new anti-resorptive agent, denosumab is a fully human monoclonal antibody that inhibits RANKL. RANKL is a protein that is required for osteoclast formation, function, and survival [44]. Thus, reducing the amount of resorption performed during remodeling. Denosumab shows great promise in the reduction of fractures of postmenopausal women with osteoporosis [45]. However, due to a lack of data, long-term side effects of denosumab are unknown. Infections of the skin and urinary tract, dermatitis, eczema, and rashes are common reactions for patients using denosumab [45].

Although anti-resorptive agents are primarily recommended, anabolic agents, such as teriparatide, are an alternative for osteoporosis treatment. Teriparatide is the 1-34 N-terminal fragment of parathyroid hormone. It will bind to receptors on osteoblasts, stimulating bone formation [46]. Some disadvantages found in teriparatide recipients were limb pain, headaches, and dizziness. There is also a study of combining teriparatide with raloxifene, which netted a net effect of bone formation. However, the treatment is expensive and it's best used as treatment for patients with a high risk of fracture or if other treatments are ineffective [46].

1.6 Material Properties of Bone

As stated before, bone can be considered as a composite material at multitudes of hierarchical scales. At the highest scale, there are two significant types of bone: cortical and trabecular. Due to their differences in structure, trabecular bone has been found to be significantly weaker than cortical bone. There are multiple factors that contribute to the mechanical properties of bones. Factors, such as porosity, collagen fiber orientation, and percent mineralization, contribute to the properties of bone. Remember, bone is continuously altering its structure depending on its mechanical environment. Thus the mechanical properties can significantly change every millimeter and throughout its lifetime. This is especially true for trabecular bone as shown in figure 12. This makes bone a difficult material to which to assign universal material properties. However, for this study, there will be a larger emphasis on cortical bone rather than trabecular.

Figure 12: Stress-strain curves of two samples of trabecular bone with varying porosities. Loading rises at the end due to the collapsing of pores [7].

There are many studies that have concluded that remodeling reduces the strength of cortical bone in all modes of loading. This is due to more than just the decrease in bone mineral density and mineralization [7, 47, 48]. There have been suggestions that the cement line between the secondary osteons and primary bone may be an addition factor to the cause of decrease in mechanical strength. Burr et al. (1988) observed ground substance in the cement line, suggesting that it may be more compliant and viscoelastic than other components of the bone [49]. This biomaterial further becomes hard to quantify in that it's viscoelastic.

1.6.1 Modeling Bone Tissue

Compact bone is a complex composite biomaterial. With many layers of lamellae, mineralized collagen network, osteons, and more, there have been many attempts in simplifying the complexities of bone. One of the earliest being bone simply treated as a two-phase composite of hydroxyapatite and collagen based on their volume fractions [7]. These models were known as the Voigt and Ruess models seen in figure 13. However, the downfall of these models was that they were very inaccurate since they didn't include many of the other factors of bone properties such as its structure. The Voigt model provided an upper bound of the elastic modulus and the Ruess model provided a lower bound, the range between the

two are so large that they were almost practically useless [7, 15].

Figure 13: Depiction of Voigt(left) and Reuss(right) models for a composite material formed by mixing two materials. In the Voigt model, the materials experience the same amount of strain, while the materials in the Reuss model experience the same amount of stress. Both use the volume fraction and elastic modulus of each material used to determine the composite modulus [50].

Although still not perfect, modeling bone tissue has come a long way. There has been an effort of modeling bone at the molecular level such as the staggered array model. With the computational work done by Siegmund et al. [51] and Mendoza [52], modeling such models using finite element analysis shows great promise in understanding the mechanical behaviors of bone.

1.6.2 Viscoelasticity

Viscoelasticity is a property in a material that exhibits both viscous and elastic characteristics when strained. In other words, a viscoelastic material's properties will vary depending on factors like its loading condition and strain rates. An experiment testing bovine femoral cortical bone in tension over a variety of strain rates found that the bone would be stronger, stiffer, and more brittle at faster strain rates [53]. Figure 14 displays the findings of the mentioned experiment. Additionally, viscoelastic materials may experience creep, stress relaxation, and hysteresis [6, 7, 15, 54, 55]. A creep response will be displayed when the material is under a constant load. The material will gradually deform to a stable displacement. On the other hand, stress relaxation will occur under a constant deformation where the stress will decrease. Lastly, hysteresis can be observed when the material is put under a

load-deformation cycle where a loss of energy can be seen.

Figure 14: Stress-strain curves of bovine femoral cortical bone in compression at varying strain frequencies. The larger the strain rate, a higher modulus can be seen [56].

One technique used to analyze viscoelasticity is dynamic mechanical analysis (DMA). A DMA will apply a load in a sinusoidal fashion and the resulting strain curve will also follow a sinusoidal formation. However, there will be a slight lag in the strain curve, which can be seen in figure 15. The lag comes from a phase shift in the material between the sinusoidal loading and the corresponding sinusoidal response. This parameter is known as the tangent of delta, commonly known as tangent delta ($\tan \delta$) [54, 57, 55]. In short, tangent delta is a measurement of the material's ability to dampen and dissipate vibration energy [58].

Figure 15: Stress and strain response curves during DMA testing. The stress curve applied by the testing can be seen in the dashed red curve. The strain response is seen in black. The tangent delta can be seen as the lag between the stress to the strain curve [55].

A material that is perfectly elastic would have a tangent delta of zero. Typical tangent delta values for metals are $10E-3$ and 0.1 to 1 for plastics [59]. For cortical bone, tangent delta values are typically between 0.01 to 0.04 [8, 60, 61].

Commonly, tangent delta is expressed as the ratio of *storage* to *loss modulus* (eqn. 6) [54, 62]. The *storage modulus* ($E_{storage}$) can be described as the materials ability to store energy that is applied to it for later use. The *storage modulus* is almost equivalent to the elastic modulus in a non-oscillatory loading condition and can be expressed as the material's stiffness while *loss modulus* (E_{loss}) is the materials ability to dissipate energy. The larger the loss modulus, the greater the material's ability of dampening vibrational energy [63]. The following is a derivation of how tangent delta is related to the storage and loss modulus.

The stress (σ) and strain (ϵ) behaviors are a function of time in the form of a sinusoidal waveform as seen below. Where δ is the phase angle between σ and ϵ

$$\sigma = \sigma_o \cos(\omega t) \quad (1)$$

$$\varepsilon = \varepsilon_o \cos(\omega t - \delta) \quad (2)$$

A complex modulus (E^*) can be expressed in terms of $E_{storage}$ and E_{loss} as seen in equations 3 - 5.

$$E^* = E_{storage} + iE_{loss} \quad (3)$$

$$E_{storage} = \frac{\sigma_o}{\varepsilon_o} \cos(\delta) \quad (4)$$

$$E_{loss} = \frac{\sigma_o}{\varepsilon_o} \sin(\delta) \quad (5)$$

Then tangent delta can be formed as:

$$\tan \delta = \frac{E_{loss}}{E_{storage}} \quad (6)$$

There has been ongoing research on determining the exact mechanism that causes bone to exhibit a viscoelastic characteristic. Specifically in cortical bone, features of osteons, such as the Haversian canal and cement lines, may contribute to these properties [7, 64]. Loss in viscoelastic properties has been observed in bone with a decrease in collagen [57]. Collagen is a definite contributor to viscoelasticity in bone. Lastly, water has been shown to be a large factor as well. Particularly, the interaction between water and the organic matrix, which is primarily composed of collagen [65, 66]. This is consistent with the observed properties of tendon, ligament, and cartilage as they exhibit viscoelastic properties and are composed of water and collagen [5, 6, 7]. Realistically, a combination of all of these characteristics contributes to bone's unique properties. This study will determine if D-spacing variance seen in collagen fibrils is an additional factor and a better-defined explanation of bone mechanical properties.

1.6.3 Modeling Viscoelasticity

There are two primary methods in modeling viscoelasticity. The first method is to collect experimental data and develop a function to fit the data. This would be simplistic if modeling for two variables. However, this can be difficult since viscoelastic behaviors can be a function of multiple variables. Variables, such as temperature, strain rate, loading, etc., can cause difficulty in developing a function that would be accurate. This is especially difficult in skeletal tissue. The issue with this approach is that the modeling can end up either too simplistic or too complex for an individual's needs.

An alternative method would be use a system of springs and dashpots. Springs are to represent the elastic portions and dashpots for the viscous portion (figure 16). There are multiple amounts of arrangements that can be formed to model viscoelasticity. The amount of each component, their elastic and viscous values, and their arrangement affect the viscoelastic behavior [5, 6, 55]. Corresponding equations for each system can be derived. This method is the best suited for this analysis. However, due to the complexity of biomaterials, there can also be improvements in making the arrangements more complex.

Figure 16: Spring and dashpot elements (top) used in a system to represent viscoelastic material properties. Additionally, some examples of the different schematics are displayed on the bottom portion that they can possibly be configured too [6].

1.7 Animal Models

The use of animals for research has dated back to ancient Greece [67]. Animal models may provide an understanding for biological phenomena in hopes of being able to relate back to the human species. This allows for more feasible and relatively ethical studies since there is a reduced risk of harming a human. The strategy is to find an animal that is a biological equivalence to humans. This will allow more relevant testing to be done by working on a living organism, though there are some other considerations when determining an appropriate animal. These considerations include: cost, ease of handling/use, ethical implications, etc. [68]. Additionally, the shorter lifecycles of animals allows for a greater turnover and expedites testing. However, there are some drawbacks of using animal models for research. Although it is not guaranteed that findings will affect humans in a similar action, animal models provide a great basis in research.

A large amount of animal testing is done in osteoporotic research. Small animals, like rats, have been used in bisphosphonate testing and other metabolic bone disease treatments due to their inexpensiveness and their quick turnover [46, 69]. However, they aren't the most ideal in bone related research due to their small size and weight. Their low weight doesn't produce a large enough normal force to produce a lot of damage in their bones. Because of this, their remodeling characteristics in trabecular and cortical bone don't accurately represent human bone tissue [69, 70]. Thus, they are a poor representation for humans in terms of bone research. Although typically more costly, other larger animal species, such as dogs, cats, pigs, sheep, and non-human primates, have been used in osteoporosis research to work around the small size of rats [71]. However, sheep are promising models for osteoporotic research for a variety of reasons.

1.7.1 Sheep Models for Osteoporotic Research

The female sheep or ewes particularly show promise in the postmenopausal osteoporosis research. Some reasonings on what makes sheep more advantageous are that they experience Haversian bone remodeling (figure 17) [72], are docile and easy to handle, relatively inexpensive, and have skeletons similar in size and mechanical properties to humans mak-

ing them easier to perform surgical procedures [69, 72, 73]. Additionally, the size of their bones are ideal in investigating structures, fractures, and fracture healing of metaphyseal bone, an area largely related to osteoporotic fractures [74].

Figure 17: Images of histological cross-sections of ewe cortical bone at different ages. The left is of a three year old ewe and primarily composes of plexiform bone due to its quickly needed growth. The right is of an older ewe. Note the heavy remodeling on the older ewe that's similar adult human cortical bone [71].

However, there are some limitations in using ewes as models. They don't naturally attain as much bone loss during their adult stages. The differences in their gastrointestinal system and phosphorous metabolism compared to humans make them a disadvantage for osteoporotic drug therapy [46].

Unlike humans, most mammals experience lifelong oestrous cycles [75, 76]. Since most mammals don't experience spontaneous menopause, a surge of bone-loss due to estrogen deficiency doesn't naturally occur [72], thus requiring some kind of mechanism to reduce estrogen. To induce menopause on sheep, a procedure known as ovariectomy (OVX) is performed where ovaries are surgically removed. Many studies have stated that this procedure has caused ewes to become osteopenic [69, 77, 78, 79, 80]. Additionally, sheep are predominantly seasonal breeders with higher estrogen levels in the autumn and winter and low estrogen levels in the spring and summer. This can cause some trouble since the seasons may influence bone mass. Some breeds, such as Merinos that are abundant in Spain, breed year-around, making seasons less influential to bone mass [81]. However, for this specific study, Rambouillet-cross ewes will be used due to their availability.

1.8 Study Objective

The purpose of this study is to create a computational model of collagen fibrils in ovine cortical bone samples that are undergoing a three point bending DMA test through finite element analysis. This two-dimensional computational model (Composite Model) includes viscoelastic collagen and elastic hydroxyapatite elements. Additionally, a variance of D-spacing and their interactions will be implemented within the model. This is to determine if a distribution of D-spacing along with the viscoelastic material sections will have a sufficient effect on the total viscoelastic properties of cortical bone. Specifically, the tangent delta from the models will be measured and will represent the viscoelastic properties.

In other words, this study is to determine if including a variance in D-spacing in existing models (Mendoza and Siegmund et al. [51, 52]) will more accurately portray viscoelastic characteristics in bone than models that don't include the distribution. If results are positive, the hopes for this model are to become a foundation in developing osteoporotic skeletal tissue finite element models. The computational models results (tangent delta) will be compared to already collected data from ovine models with the same D-spacing distribution.

For this study, two sectors of the radius/ulna of ovine will be modeled. The reasoning behind choosing the two sectors for this study is to see if the model can generate an appropriate response for different bone sectors. Different bone sector will have varying mechanical properties due to their loading conditions. Caudal sector will be subjected to compressive loads and cranial models will be naturally put under tensile loads. This is due to the shape of the ulna/radius causing a small moment arm when weight (or a portion of it) is applied. Additionally, it has been observed that cranial and caudal sectors have a different distribution of D-spacing; thus a differences in collagen and mineralized sectors. This study hopes to determine that differences in distribution accounts for some of their difference in mechanical properties.

The hypothesis to this study is that results from this computation model will better represent viscoelastic properties than previous finite element models that didn't include a variance of D-spacing. An additional hypothesis is that the varying distributions of D-spacing for each sector will account for the differences in their viscoelastic properties.

2 Methods

2.1 Experimental Data

2.1.1 Sample Preparations

Sample preparation was performed at Colorado State University and mechanical testing was done at Henry Ford Hospital. The University of Michigan, Ann Arbor conducted and provided D-spacing data for the test samples that allowed a comparison of the results of this computational analysis. None of the animal testing was performed at California Polytechnic State University, San Luis Obispo.

Under local Institutional Animal Care and Use Committee (IACUC) approval, a total of six Rambouillet-cross ewes were obtained by Colorado State University's College of Veterinary Medicine and Biomedical Sciences. The ewes were anesthetized and either ovariectomized to induce estrogen depletion in order to simulate postmenopausal osteoporosis or underwent a sham surgery as a control. After twelve months post-surgery, the ewes were sacrificed and the left radius and ulna were collected. The collected bones were wrapped in saline-soaked paper towels and stored in Ziplock bags at -20° Celsius until testing.

2.1.2 Mechanical Testing

The generated Composite Models used for this study were based off of ovine bone samples that had undergone a three point bending experiment done at Henry Ford Hospital. The bones were divided into six anatomical sectors seen in figure 18. Each radius and ulna sample was divided into 25 beams and beams were taken at random to be tested for each anatomical sector. Beams were machined into 1.75mm x 1.75 mm x 19mm pieces. Of the six sectors, only data obtained from the cranial and caudal sectors of the control group was used for this analysis (a total of twelve). The reasoning behind choosing the two sectors for this study is to see if the model can generate an appropriate response for different bone sectors. Before viscoelastic tests were done, the beams were thawed and placed in a 0.9% saline solution at 37° Celsius.

Figure 18: Cross section of collected ewe bone samples. Anatomical sectors were divided by white lines. Oftentimes, the radius and ulna are fused together. Multiple beams can be created from a single sector. A beam was chosen at random for mechanical testing [79].

Viscoelastic measurements were acquired using a dynamic mechanical analyzer (DMA 7e, Perkin-Elmer). Testing was done at 37° Celsius to be consistent with normal human body temperature. The loading was done with a 550mN static load and dynamic load applied was 500mN for each sample at varying frequencies between 1-20 Hz in increments of 0.2 Hz. However, for this specific study only 1, 3, 9, and 15 Hz will be simulated to compare results on what was done previous by Mendoza [52]. It is to be noted that testing was non-destructive to the samples. Beams were loaded onto a three-point bending fixture and subjected to bending. A table of the data used for this analysis can be seen in appendix A. The beam sectors were assumed to be a homogeneous material due to their small size in order estimate the stresses which can be seen in appendix B.

The primary goal of this model is to determine if a variation in the distribution of collagen D-spacing affects the viscoelastic properties of skeletal tissue for different bone sectors. To accurately determine this, a model that takes nanoscale D-spacing behavior into account is required to observe the effects on the time dependent properties of the structure. Atomic force microscopy (AFM) was conducted by the Department of Chemistry at University of Michigan, Ann Arbor to evaluate morphology of collagen bundle structure in cortical bone

samples. The means and standard deviations of D-spacing in the six different bone sectors from the AFM evaluation were utilized in this study.

2.2 Model Evolution

2.2.1 Previous Models

Human tissue is complex in a way that it is difficult for today's technology to be able to mimic their exact characteristics. Bone is no exception to being a complex biomaterial. Because of this, a basic computational model will not be enough to produce realistic results. It has been established that a good estimation needs the following parameters: precise geometry, oscillatory loading, and viscoelastic behavior [51, 52]. Lastly, it has been accepted that the distribution of D-spacing is not consistent [4]. The finite element model developed in this study was used to attempt to accurately represent compact bone tissue material properties using the mentioned parameters. However, there were a handful of other models that essentially were the backbone for this model that have already included many of these parameters.

This study's model was aimed to be the next step in an evolutionary chain that started with the Hodge-Petruska model [82], which as undergone multiple iterations to later become the staggered array model. Their work helped generate a mathematical periodicity model produced by Jager and Fratzl in 2000, which expressed the periodicity as collagen and mineral as mineralized collagen fibrils [83]. The Jager-Fratzl model allowed for many benefits to be incorporated into the component model. One of which is that it includes the overlap and gap regions caused by the collagen D-spacing behavior. These fibrils can be represented as rectangular units with collagen and hydroxyapatite sections. Figure 19 shows side-by-side comparison of the Hodge-Petruska and the staggered array model.

Figure 19: A comparison between the Hodge-Petruska and staggered array models. Note the different arrangement of collagen and hydroxyapatite. The Hodge-Petruska model can be seen on the left and the staggered array model can be seen on the right. The staggered array model portrays the characteristic D-spacing behavior resulting from the overlap and gap regions [83].

2.2.2 Siegmund Model

Jager and Fratzl's work then contributed to the research done by Siegmund et al. [51]. Siegmund et al. observed that collagen cross-linking had an influence on the energy absorption done by compact bone. Although the work seems relatively simple, it developed a large portion of the foundation for future finite element work, including this current study.

This Siegmund model breaks down the periodic model by staggered array model into a single period of collagen molecules. The simple period unit cell includes sections of mineralized hydroxyapatite and collagen molecules sections. To further simplify the model, Siegmund's model halves the single cell. This simplification was utilized because the symmetrical nature of the unit cells, which would reduce the computation power and time needed to run an analysis. However, halving the cell requires an implementation of another boundary condition at the shared edge between the two halves. A breakdown of Siegmund's model can be seen in figure 20. The model underwent a uniaxial loading condition to represent the collagen fiber in tension. The dimensions of the model can be define by D-spacing,

equation 7, and mineral volume fraction, equation 8. Both models can be described with platelet length L , the distance between the mineral platelets a , mineral platelet thickness t , and the total thickness of the three collagen triple helices between the mineral $n.d$.

Figure 20: (a) Simplified two-dimensional representation of the staggered array model. Collagen sections are light-grey and hydroxyapatite sections are depicted as black boxes.(b) Periodic arrangement of collagen and hydroxyapatite sections are shown along with the tensile loading condition.(c) Graphic representation of a single unit cell that Siegmund developed shown with its characteristic dimensions. The definitions of these characteristic dimensions and their assumed values are described in table 1.(d) Half unit cell of Siegmund model that was used in their computational model [51].

Each of these variables were determined by assuming a platelet length/periodicity of 67 nm because it was believed that it was constant at the time [5, 83] and a mineral volume fraction of 0.3 based on the work by Currey [15, 83, 84] and Fritsch et al [85]. Other assumptions included: mineral plate width is 2.5 nm [9], the number of collagen domains between the mineral crystals is 3, the width of collagen is 1.5 nm [5], and the periodic unit length is 67 nm [5, 83]. Table 1 provides a quick reference to these dimensions. The following equations express periodic unit length and mineral volume fraction as a function for all variables in the Siegmund's unit cell model:

$$p = \frac{L+a}{2} \quad (7)$$

$$V_V^m = \frac{L * t}{(L + a)(n.d + t)} \quad (8)$$

Table 1: Siegmund Model Dimensions. These dimensions describe the details of the model.

Dimensions	Variable	Value
Dimension of Half Unit Cell Model	N	1 x 1
Periodicity (D-spacing)	p	67 nm
Number of Collagen Helices	n	3
n * Collagen Thickness	n.d	4.5 nm
Short Collagen Length	a	20.1 nm
Fraction of Mineralization	V_V^m	0.3
Mineral Thickness	t	2.5 nm
Mineral Length	L	113.9 nm

Simple linear elastic material properties were used in the Siegmund model. Table 2 displays the elastic mechanical properties of the model. Hydroxyapatite was represented as a simple elastic isotropic solid. Although collagen has complex viscoelastic properties, the Siegmund model represented the collagen as a homogeneous collagen triple helix in a wet environment with a shear modulus of $G^c=50$ GPa [86].

Table 2: Material Property Definitions of Hydroxyapatite and Collagen as in the Siegmund Model.

Material	Elastic Modulus (E)	Poisson's Ratio (ν)
Hydroxyapatite	100 GPa	0.28
Collagen	5 Gpa	0.20

An important finding from the research done by Siegmund et al. is that interaction between the collagen molecules and hydroxyapatite must be included in a model in order for it to be biologically relevant [51]. What was great about this model is that they were able to essentially rework the Hodge-Petruska model into a computation form. However, there are many ways to improve upon this model. One of which is to include a viscoelastic component into the collagen molecules, which would be addressed in the next model of the evolutionary chain. Siegmund et al.'s model makes a great building block towards more complex models to increase biologic relevance.

2.2.3 Half Unit Cell Model

This current study's model was highly based off the work done by Mendoza and his thesis in 2014 [52]. For his thesis, Mendoza explored the effects of altering D-spacing in the mechanical properties of a collagen and hydroxyapatite model (Half Unit Cell Model) representing a collagen fibril using finite element analysis. The Half Unit Cell Model's geometry was based off of Siegmund's works. However, Mendoza incorporated a viscoelastic material subroutine for collagen sections. Figure 21 depicts the details of his viscoelastic model.

Figure 21: (a) Mendoza's Half Unit Cell Model highlighting the hydroxyapatite section in pink and the collagen section in grey. (b) Model shown on Abaqus' global axis. (c) Model and its set boundary and loading conditions [52].

Mendoza was able to explore the effects of D-Spacing by developing several models with varying period(D-spacing) lengths, mineral volume fractions, collagen lengths, and mineral lengths. Table 3 shows the various dimensions he used for his models in his study. Each model was then analyzed once with four different loading frequencies: 1, 3, 9, and 15 Hz [79, 87].

Table 3: Half Unit Cell Model Sample Dimensions. The model developed by Cal Poly graduate Miguel Mendoza explored modeling a half unit cell at multiple D-spacing values.

Model	Periodic Unit Length, P (nm)	Mineral Volume Fraction
Normal D-spacing	67	0.3
High D-spacing	73	0.3
Low D-spacing	61	0.3

The findings in his study were that altering D-spacing by itself does not significantly alter the viscoelastic properties of the material [52]. Instead, what he found was that when the mineral volume fraction was altered there was a significant change in viscoelastic properties. His overall major findings were that D-spacing may play a significant role in viscoelastic properties of bone only if there is an accompanied change in mineral volume fraction. His findings on the significance of mineral volume fraction to bone's viscoelasticity were incorporated into this study.

Although the findings in Mendoza's research were interesting, there were some limitations to his study. First off, his models had no replicates. This decreases the statistical significance in his study because without replicates it is not possible to estimate the experimental error. Secondly, despite having a simple model, a more accurate representation of the material would be multiple unit-cells interacting in parallel and in series. Both limitations were addressed in this current study.

2.3 Custom Viscoelastic Material Definition

Bone is usually considered a composite material that is primarily composed of hydroxyapatite, collagen, and water. Hydroxyapatite can be treated as an elastic material, while collagen was treated as a viscoelastic material [55]. To mimic the viscoelastic properties of collagen in Abaqus, a user subroutine was developed. A user subroutine allows more complex functions to be used in Abaqus. This specific user subroutine called UMAT(user material) was developed to allow Abaqus to implement the custom complex viscoelastic material properties of collagen to the collagen sections of the model [88]. The UMAT code was based off the simplest rheological model that exhibits both stress relaxation and creep. The UMAT used for this study was originally developed by Richter [52, 89] and used by Mendoza in his thesis [52]. Integrated into the UMAT is the ability for ABAQUS to up-

date the stresses and solution-dependent state variables at each step. The schematics of the UMAT user subroutine can be seen in appendix B.

There are two forms to this model: the Kelvin-Voigt body and the Maxwell body [5, 55]. There are no apparent advantages between the two forms. However, for this specific user-defined material, the Kelvin-Voigt body was modeled, which can be seen in figure 22.

Figure 22: (Left) Kelvin-Voigt body form of the Standard Linear Solid [90]. (Right) The three-dimensional version of the rheological model where the spring and dashpot portions are replaced by a shear and bulk modulus [52, 90].

Since the rheological model is one dimensional, a custom three-dimensional form of the constitutive equation was used to develop the linear viscoelastic material definition [89]. The mentioned three-dimensional form can be written as:

$$\begin{aligned}
 & \left(1 + \frac{G_{Ke}}{G_E}\right) \sigma_{ij} + \left(\frac{K_{Ke}}{K_E} + \frac{G_{Ke}}{G_E}\right) \frac{\sigma_{kk}}{3} \delta_{ij} + \frac{\eta_s}{G_E} \dot{\sigma}_{ij} + \left(\frac{\eta_b}{K_E} + \frac{\eta_s}{G_E}\right) \frac{\dot{\sigma}_{kk}}{3} \eta_{ij} \\
 & = 2G_{Ke} \epsilon_{ij} + (3K_{Ke} - 2G_{Ke}) \frac{\epsilon_{kk}}{3} \delta_{ij} + 2\eta_s \dot{\epsilon}_{ij} + (3\eta_b - 2\eta_s) \frac{\dot{\epsilon}_{kk}}{3} \delta_{ij}
 \end{aligned} \tag{9}$$

The i and j indices indicate tensor components of the stress (σ) and strain (ϵ) tensors. The subscript kk shows the trace of the tensor (i.e. $\sigma_{kk} = \sigma_{xx} + \sigma_{yy} + \sigma_{zz}$) and the δ_{ij} is the Kronecker delta, which is a function of i and j that describes the tensor in binary form. The dot denotes a derivative with respect to time. Variables with the subscript E and Ke correspond with the lone spring and the spring within the Kelvin-Voigt body in the model respectively. The expressions for the bulk modulus, K , and the shear modulus, G , composed of the spring modulus (E) and Poisson's ratio (ν) can be seen in equations 10 and 11 respectively.

By inputting already established material values for the Standard Linear Solid, this

three-dimensional model can improve viscoelastic behavior for computational modeling. The coefficients in the three-dimensional model seen in figure 22 can each be assigned a material property to improve biological relevance.

$$K = \frac{E}{3(1-2\nu)} \quad (10)$$

$$G = \frac{E}{2(1+\nu)} \quad (11)$$

Interestingly, the expressions for the bulk viscosity, η_b , and shear viscosity, η_s , follow a similar form as the bulk modulus and shear modulus. These expressions can be seen below in terms of viscosity (η) and Poisson's ratio where η_1 represents a material property without adequate biological relevance.

$$\eta_b = \frac{\eta_1}{3(1-2\nu)} \quad (12)$$

$$\eta_s = \frac{\eta_1}{2(1+\nu)} \quad (13)$$

The three dimensional formulation can be simplified down to the original one dimensional formulation by simply setting the Poisson's ratio for each of the elements and the perpendicular components of the stresses and strains to zero. This is demonstrated in appendix C.

$$\sigma + \frac{\eta_1}{E_1 + E_2} \dot{\sigma} = \frac{\eta_1}{1 + \frac{E_1}{E_2}} \dot{\epsilon} + \frac{1}{\frac{1}{E_1} + \frac{1}{E_2}} \epsilon \quad (14)$$

2.3.1 Parameter Definition

The user-defined material property for collagen calls for six individual parameters for the springs and dashpot of the Kelvin-Voigt body. A table of the parameters and their corresponding values used can be seen in table 19. The first three are the elastic moduli and the viscosity of the springs and dashpot while the last three parameters are the Poisson's ratios

for the three components. Having these six parameters as part of the material definition causes difficulty in that its hard to establish values for each parameter to produce similar mechanical properties to native collagen in compact bone [61].

To tackle this situation, values of the mechanical properties of pure collagen tissue from literature [6, 7, 15] and the measured tangent delta from the bending testing were used [79, 87]. This would result in a spring stiffness of 2 GPa to be used for collagen. Since Poisson's ratio for collagen tissue is roughly 0.2, the three Poisson's ratios were set to 0.2. The 2GPa modulus was then represented as the *effective modulus* as seen in equation 15. To achieve an effective modulus of 2 GPa, E_1 and E_2 were assumed to be 3 GPa and 6 GPa respectively.

$$E_{effective} = \frac{E_1 E_2}{(E_1 + E_2)} \quad (15)$$

A value of 1.25 GPa*s viscosity was used for the dashpot component. Simulations were performed with multiple values for viscosity. However, 1.25 GPa*s was chosen over the rest because its results were better fit to the experimental data in past studies [52].

These UMAT parameters were input through the ABAQUS GUI when the collagen material was being created.

2.4 Complex Model

This computational model was developed using Mendoza's Half Unit Cell Model. The Composite Model was composed of a total of two hundred (2 rows x 100 columns) half unit cells. The top and bottom rows were oriented in a way that edges of the unit cells are interacting on their edges. Thus, there are technically 50 full unit cells on each row in parallel with each other. This model was able to represent multiple unit cells interacting with each other in parallel and in series.

Due to Abaqus' inability to handle geometric dimensions in the nanoscale, each model dimension was entered in microns. Consequently, the elastic modulus and applied loading condition were adjusted proportionally (hydroxyapatite elastic modulus of 100 GPa was set to 0.1 E12 Pa).

2.4.1 Model Builder

Since an objective of the thesis is to determine if a variance in D-spacing distribution has a significant effect on the viscoelastic properties exerted by the collagen fibrils, a Python script was developed to build a large composite model using two hundred Half Unit Cell Models. Like the Half Unit Cell Model, the Composite Model will also be two-dimensional. The Composite Model incorporates the interactions between unit cell models in series and parallel by producing a final model that included 2 x 100 Half Unit Cell Model. The Python script also randomly generates individual collagen D-spacing dimensions by using the 'Random.Gaussian' function utilizing the mean and standard deviations of the observed collagen D-spacing in the AFM measurements by the University of Michigan, Ann Arbor for the cranial and caudal sectors [4]. These governing dimensions can be seen in table 4. The Python code uses these dimensions to determine the dimensions of the total model. The total model is a single part until the script partitions it into hydroxyapatite and collagen sectors accordingly. Lastly, due to the varying dimensions, each row will naturally offset from one another. This offset nature is unwanted due to a potential of the jagged edge in the model to cause unwanted deformation and stress concentration. To prevent this, a collagen spacer was implemented on the shorter row to give it the same length as the longer. However, a large collagen spacer may have a large influence on how the model behaves. To combat this issue, spacers that were larger than 5% of the previous half unit cell were rejected. The script was rerun until a model that is biologically relevant was produced. Four models for each sector (caudal and cranial) were developed using the Python script. These eight models were put under the same boundary conditions. Their loading conditions were based on the DMA testing. The generated total lengths through the Python script can be seen in table 5. The model builder Python script can be seen in appendix J.

Table 4: Governing D-spacing means and standard deviations of cranial and caudal sectors that were used to develop the complex model. These values were used for the 'Random.Gaussian' function in the Python script. Values were provided by the University of Michigan, Ann Arbor. 'Control' indicates ovine that were subjected to a sham surgery. These sheep did not have their ovaries removed and acted as a control to the ovariectomized sheep.

Model (Surgical Treatment, Bone Sector)	D-space Mean (μm)	D-space St. Deviation (μm)
Control, Cranial	0.0684	0.0013
Control, Caudal	0.0664	0.0017

Table 5: Composite Model Row Lengths. Due to the random function implemented in D-spacing lengths, models had varying top and bottom row lengths. Four sub-models were produced for each model (Control, Cranial and Control, Caudal).

Model (Sector Version)	Row Lengths (μm)	
	Top Row, L_1	Bottom Row, L_2
Cranial 1	6.85573	6.86152
Cranial 2	6.86835	6.85995
Cranial 3	6.84309	6.84149
Cranial 4	6.81504	6.85025
Caudal 1	6.63177	6.63619
Caudal 2	6.64287	6.63401
Caudal 3	6.63001	6.28433
Caudal 4	6.63087	6.63515

2.4.2 Loading and Boundary Conditions

Due to the symmetry and the periodic characteristic of the staggered array model, x-symmetry and y-symmetry boundary conditions were included to reduce total computational time of the analysis. The x-symmetry and y-symmetry boundary conditions were applied at the left vertical edge and the bottom horizontal edge of the model respectively (figure 23). A uniform distributed load with comparative mechanical loading to the experimental testing was applied on the right vertical edge of the Composite Model. The loading underwent 20 cycles of 1, 3, 9, or 15 Hz with each cycle comprised of 20 increments causing a total of 400 increments/step. Based on the maximum pressure of testing, the amplitude of the sinusoidal loading was determined to be 3.36 MPa. These values were chosen based on the conditions of the DMA testing done at Henry Ford Hospital and can be seen in ap-

pendix A. The modeled stress of the bone were determined by considering the geometry of beams, applied forces of the DMA, and the dimensions of the three point bending apparatus. The beam sections were assumed to be a homogeneous material due to their small size to estimate the stresses, which can be seen in appendix D. Figure 23 shows the details of the Composite Model that was developed for the analysis. The load was applied with a sinusoidal loading condition using the periodic amplitude feature in Abaqus. Similarly to Mendoza's models, each Composite Model was tested at increasing frequencies (1, 3, 9, and 15 Hz) [79, 87]. The specifications on the sinusoidal loading condition and how to adjust it for each frequency can be seen in appendix E.

Figure 23: (a) Left end of the Composite Model. The x-symmetry boundary conditions can be seen on the left vertical edge. Additionally, an y-symmetry boundary condition is applied throughout the bottom horizontal edge. (b) A distributed load is applied on the right edge of the Composite Model. The direction of the load is dependent on the model sector. (c) Two materials were assigned to the sections of the model. The salmon colored sections represent the mineralized hydroxyapatite. Green sections were assigned as collagen and their respective material property.

2.4.3 Material Assignment

Each collagen and hydroxyapatite section was assigned material properties by hand. Hydroxyapatite sections were modeled as an elastic isotropic solid, $E=100$ GPa and $\nu=0.28$ [5, 91, 92]. The collagen sections were assigned the UMAT subroutine that was mentioned previously.

2.4.4 Element Type

Because the model is two-dimensional, plane strain element types were chosen for the whole model. Additionally, quadrilateral elements were chosen due to the simplicity of the model's shape. On top of that, quadratic quadrilateral elements were selected over linear quadrilateral elements because the increase in model accuracy outweighs the increased computational run time. Thus, the element type, CPE8, was assigned for all elements of the model.

2.4.5 Model Validation

Before running the model, the user-defined material was tested to ensure it performed as predicted. To test it, the material properties were applied to a one-dimensional truss element. The first three properties were the elastic modulus and viscosity of the springs and dashpot. These values were set to some values that allowed for easy manual hand calculations (1Pa, 1Pa, and 1Pa-s). To further increase easy manual hand calculations, the Poisson's ratios for all three components were set to zero. The test for creep and stress relaxation were applied to the model separately and the corresponding results were recorded.

To compare how well the UMAT performed, the governing equation for the Kelvin-Voigt body form of the Standard Linear Solid was manipulated to solve for creep or stress relaxation. Results of the comparison were promising in that they were almost identical for all time points. The calculations can be seen in appendix C. This provided proof that the user defined viscoelastic material was validated against the one-dimensional Standard Linear Solid. The resulting tangent delta for this test was then plotted against experimental data at different frequencies: 1, 3, 9, and 15 Hz.

2.4.6 Mesh Development

After determining which element type would be most adequate for this analysis, mesh convergence was done. Convergence is done to determine how many elements are needed for the model to converge to a solution without consuming too much computational power. In turn, this will provide the shortest computational run time while providing an accurate

model. This was done by running the same model multiple times at different global seed sizes between 0.3 nm to 3 nm. The global seed size determines the coarseness of the element mesh, which correlates with the degrees of freedom in the model. Each model has a stress concentration at a node on the end of the Composite Model where a hydroxyapatite section interacts with surrounding collagen sections. This node can be seen in figure 24. Each model was probed at this node for its displacement. The displacement was plotted against degrees of freedom. Degrees of freedom was chosen because it correlates with computational time more than other parameters. Seed sizes past 0.5 nm provided a very small difference in deflection while significantly increasing computational time. Thus, a seed size of 0.5 nm (nonadjusted) was determined to be the most optimal for the remaining models, which resulted in approximately 1.3 million degrees of freedom. The results for the mesh convergence can be see in figure 25 and table 6.

Figure 24: The node that was probed for mesh convergence is denoted with the red marker.

Figure 25: After multiple iterations of the test model at different seed sizes, it was determined that a seed size of 0.0005 mm or roughly 1,300,000 degrees of freedom would produce adequate results without wasting computational time/power.

Table 6: Summary Table for Convergence Study.

Seed Size (mm)	Degrees of Freedom	Displacement (nm)
0.00030	3156774	0.909061
0.00035	2297128	0.908986
0.00040	1804672	0.908919
0.00050	1285582	0.908780
0.00060	763272	0.908569
0.00070	591264	0.908392
0.00100	261804	0.907346
0.00300	88282	0.904796

Since the general shape of the Composite Model was simple, the quality of the elements was high. Because of the large length to width ratio of the Composite Model, some of the models had distorted elements near the free end. However, these elements didn't cause any unwanted characteristics in model results. This was verified by comparing their displacements to neighboring non-distorted elements. No fatal errors occurred in any of the models during the analysis.

2.5 Post Processing

Abaqus has the ability to provide a variety of output information, such as stress, strain, strain energy, etc. for the Composite Model. However, only the deflections at nodes at the free end are needed for the analysis to evaluate viscoelastic properties of the model. The nodes of interest were all the nodes at the end of the model. This information is extracted by a Python script through the Abaqus GUI seen in appendix H. The output of this Python script was a folder that contained text files of deflections for all nodes of interest. The text files were ready to be analyzed through MATLAB.

The MATLAB code calculated the tangent delta exhibited by the model. This was done by averaging the deflections among all the nodes at the free end of the model for each time point. The model strain for each increment is then calculated by dividing the average deflection by the total model length (LengthF in model development code). The strain history is then plotted against two separate algorithms to collect information on tangent delta. This was done by using a best-fit algorithm to fit the strain history to a sinusoidal curve. The best-fit algorithm has some random element to it. Thus, this procedure was done ten times for each model to allow for a stronger statistic analysis. The average tangent delta for each model is reported. The Python code that was used to extract the deflection data and the MATLAB code that was used to evaluate the data can be seen in appendix I.

2.6 Statistical Analysis

The 'Random.Gaussian' implementation in the Composite Model allowed for an increase in the strength of the statistical analysis. Although the work done by Mendoza helped as a great building block, he only reported one value for tangent delta for each frequency in his models [52]. This would allow for little to no statistical significance to be drawn. With random factors in the model development and the multiple models generated for both sectors, stronger statistical significance can be drawn with the ten replicates for each model and the four subjects.

After gathering data from all eight models, the statistical analysis was performed with the assistance of Dr. Jonathan Walker of the California Polytechnic State University, San

Luis Obispo Statistics department via Statistical Analysis Software (SAS). A two-way analysis of variance (ANOVA) between experimental-cranial vs. simulation-cranial and experimental-caudal vs. simulation-caudal were performed on the resulting data. Cranial data could not be statistically compared to caudal data because the experimental results are paired and the FEA simulation results are not. With a two-way ANOVA, it could be determined if any of the two different main treatments may explain a change in tangent delta. The two treatments are *Model Type* (Composite or Experimental) and *Test Frequency* (1, 3, 9, or 15 Hz). In addition to these two base treatments, their interactions may explain a change in tangent delta. If the interactions were found to be significant, then the effects on treatment are dependent on another treatment and the resulting data can't be described by just the main effects (Model Type and Test Frequency). A table of all the possible treatments are seen in Table 7.

Table 7: A Summary Table for Statistical Treatments.

Statistical Treatment	Abbreviation	Treatment Levels
Model	Mod	Composite Model or Experimental Data
Test Frequency	Frq	1, 3, 9, or 15 Hz
Interaction between Mod and Frq	Mod*Frq	Effect of Mod depends on Frq

A pairwise comparison was done to compare the significant treatments after the ANOVA was completed. A Tukey-Kramer pairwise comparison was performed at an overall significance level of 5% because it would limit the possibility of a type 1 error to occur. Limiting the chances of type 1 error would be advantageous due to this study's relevance to the medical field where if an error were to occur, a false positive is more undesirable than a false negative.

3 Results

A total of eight biological relevant Composite Models were generated using the Python script (four cranial and four caudal) through computers provided by Dr. Scott Hazelwood. The script was able to randomly generate each model using the 'Random.Gaussian' function in Python via the mean and standard deviation of the data from the experimental AFM on corresponding ewe bone samples provided by the University of Michigan, Ann Arbor. The computers were set up specifically for FEA research and other intense computational work. Each model was run four separate times at different loading frequencies (1, 3, 9, and 15 Hz) making a total of 32 independent input files. Model input files were run through the computer terminal. Average computational times for each input spanned from 8 to 10 hours. Most models were run with a single warning message. Models with this message had distorted elements within the model. Distorted elements are defined in Abaqus as elements with an angle less than 45 degrees or greater than 135 degrees. This can be due to the relatively large shape of the model and the automatic meshing fitting elements along the partitions. However, these warnings are normal and don't affect the model negatively. Also, since there are so many elements in a model (219000 elements) and few distorted elements, this warning doesn't raise any red flags.

Results came in a form of output database files (ODB) when computational work was completed. Since deformation data is only needed, other parameters were omitted from being calculated. This greatly reduced the size of the ODB files from 100 GB to 4 GB. The deformation data was extracted from the ODB files through a Python script that compiled individual text files for each node of interest. MATLAB codes were able to extract this data and calculate tangent delta. Tangent delta was calculated as the phase shift between the loading and deformation curves. The MATLAB output can be seen in figure 26. The code works with a best-fit algorithm on how well the deformation data can fit a sinusoidal function. This output can be seen in figure 27. Since there is some element of randomization in the best-fit algorithm in the MATLAB code, it was run ten times for each set of data.

Figure 26: Tangent delta is calculated between the loading cycle (black line) and the deformation cycle (red dashed line). Time is on the x-axis while the y-axis is the normalized stress/strain.

Figure 27: The best-fit curve for a single run for a Control, Caud Composite model at 1 Hz from the MATLAB code. The R^2 value of 0.9999 denotes an accurate fit of the data to a sinusoidal curve. Tangent delta for a single run (one of ten) can be seen on the bottom right as well. The MATLAB code was run ten times and all tangent delta values were averaged.

To accurately mimic the viscoelastic trend in the data, a rheological model with appropriate parameters was established. Data from Mendoza suggested that a dashpot with a viscosity of 1.25 GPa*s (η_1) best matched the characteristic of the experimental data from the DMA testing [52]. To explore the effects of dashpot viscosity on tangent delta, a model was made with a 4 GPa*s dashpot by simply editing the material properties in the input file. The data from this exploration can be seen alongside DMA experimental data and data from the same model with a dashpot of 1.25 GPa*s in figure 28 for all tested frequencies.

Figure 28: Comparisons between different dashpot values for a Control Cran replicate in order to view the effects of a more viscous dashpot has on tangent delta. The findings by Mendoza were confirmed that a dashpot with a viscosity 1.25 GPa*s produces data more similar to the experimental values than a dashpot of a viscosity of 4 GPa*s.

Although it visually seems that the 4 GPa*s dashpot performed well in high test frequencies, it poorly matches the experimental data at lower rates. The 1.25 GPa*s dashpot performed better at matching the trend of the experimental data along all strain rates confirming Mendoza's findings. A table of all data in this exploration can be seen in appendix F.

To validate computational models, model data needed to be compared to experimental findings from the DMA testing done by Henry Ford Hospital. Experimental data was organized based on specimens, treatments, anatomical sectors, and frequency. For this study, cranial and caudal sectors of the radius/ulna of control sheep (sheep that have undergone a sham surgery) were compared at 1, 3, 9, and 15 Hz (tables 8 and 9). The full set of data

from the DMA testing can be seen in appendix A. In addition, the results were compared to results from Mendoza's findings.

Table 8: Summary Table for Control Cranial DMA Testing on Six Rambouillet-cross Ewes. Data were provided by Henry Ford Hospital.

Model (Control, Cran)	Average/ St. Deviation	Tangent Delta			
		1 Hz	3 Hz	9 Hz	15 Hz
DMA Data	Average	0.07656	0.06437	0.03947	0.01876
DMA Data	St. Deviation	0.00883	0.00561	0.00548	0.00751

Table 9: Summary Table for Control Caudal DMA Testing on Six Rambouillet-cross Ewes. Data were provided by Henry Ford Hospital.

Model (Control, Caud)	Average/ St. Deviation	Tangent Delta			
		1 Hz	3 Hz	9 Hz	15 Hz
DMA Data	Average	0.08535	0.07196	0.04674	0.02337
DMA Data	St. Deviation	0.01001	0.00935	0.00917	0.00909

The results from the four models of each anatomical sector (Cranial and Caudal) can be seen in figures 29 and 30. Since the model geometry was developed by random function based on the observed Gaussian distribution found in the experimental AFM study by University of Michigan, Ann Arbor, results varied among all models.

Figure 29: Four replicate Control, Cranial models were developed for this study. All replicates were run at all test frequencies. The average tangent delta for each test frequency will be utilized in statistical analysis.

Figure 30: Four replicate Control, Caudal models were developed for this study. All replicates were run at all test frequencies. The average tangent delta for each test frequency will be utilized in statistical analysis.

The output tangent delta for each testing frequency varied between each replicate model. However, overall, they exhibited a similar trend throughout all testing frequencies. Between the two anatomical sectors, they produced a similar trend in how the data was scattered.

Data from each anatomical sector model were averaged to make comparison easier. The small spread among data points was utilized as error bars for each frequency. Figure 31 depicts a plot of mean tangent delta values from the experimental DMA testing and the FEA study, in addition to values from Mendoza's thesis [52], for cranial data of control ovine. Comparisons with the Half Unit Cell Model allowed for detection in a difference from each model's outputs, determining if adding the interaction between unit cells and a distribution of D-spacing will impact the viscoelastic behavior.

Figure 31: Plot for all tangent delta values from DMA experimental data, Mendoza's "Normal D-spacing" Half Unit Cell Model results [52], and the Composite Model for Control, Cranial sectors. The tangent delta data from the Composite Model were averaged for each frequency. Standard error bars were applied to each test frequency for experimental and Composite Model results. A dashpot with a viscosity of 1.25 GPa*s were used in models.

Mendoza's tangent delta results stemmed from a "Normal D-spacing" Half Unit Cell Model. In his thesis, he had multiple models of half unit cells with different periodicity lengths: 64, 67, and 70 nm. His "Normal D-spacing" model had a periodicity of 67 nm and was used as comparison to data from this study's models. This decision was based off Mendoza's preference for this particular model when reporting his final results. Additionally, the geometry of the Composite Model was highly based off of this model such as mineral, volumetric ratio of mineralization, etc.

Standard error bars for the experimental data were calculated using the DMA test results. Standard error (SE) was calculated using the following equation:

$$SE = \frac{s}{\sqrt{n}} \quad (16)$$

where n is the number of samples and s is the sample standard deviation. This equation for standard error was used for both the experimental data and Composite Model results. The sample standard deviation data used and the resulting standard error can be seen in table 10.

Table 10: Summary Table for Control, Cranial Testing. Experimental data were collected via DMA testing on six Rambouillet-Cross Ewes. Composite Model data were obtained through averaging tangent delta values for each test frequency. Reported data from Mendoza's "Normal D-spacing" Half Unit Cell Model were used as comparison [52]. Standard deviation and standard error can't be reported for the Half Unit Cell Model due to a lack of randomization or replicates. A dashpot with a viscosity of 1.25 GPa*s were used in models.

Model (Control, Cran)	Average/St. Deviation/ St. Error	Tangent Delta			
		1 Hz	3 Hz	9 Hz	15 Hz
Experimental Data	Average	0.07656	0.0644	0.03947	0.01876
Experimental Data	St. Deviation	0.00883	0.00561	0.00548	0.00751
Experimental Data	St. Error	0.00360	0.00229	0.00224	0.00306
Composite Model	Average	0.06148	0.03941	0.02206	0.01294
Composite Model	St. Deviation	0.00954	0.00061	0.00843	0.00743
Composite Model	St. Error	0.00477	0.00031	0.00422	0.00371
Half Unit Cell Model	Average	0.15441	0.08555	0.03084	0.01855

Because Mendoza only reported a single tangent delta value for each frequency, statistical analyses can't be done between the two models. However, comparisons can still be drawn. Through visual inspection, the behaviors of the Composite Model and the Half Unit Cell Model vary greatly. This is clearly seen in the lower frequencies, where the data points from the Composite Model are substantially closer to the desired experimental values. Additionally, throughout all tested frequencies, the data points provided by the Mendoza model fall outside of 3 standard deviations from the mean tangent delta of the Composite Model. The addition of a distribution of D-spacing and their interactions in a single model seems to have a large impact in viscoelastic modeling.

In another attempt to compare the Composite Model and Half Unit Cell Model, an R^2 value was also obtained by plotting the FEA results against the experimental results (figure 32). A linear best-fit trend line was applied to the plot. An R^2 value was found using all four replicates for each sector and the average tangent deltas for each frequency. An average is needed as a means to sufficiently compare R^2 values to Mendoza's Half Unit Cell Model since there was only one tangent delta value reported per frequency. These R^2 values mean that the variation in computational model is explained by the experimental inputs. A summary of all R^2 values for the cranial Composite and Half Unit Cell Model are shown in table 11.

Figure 32: A linear best-fit trend line was used to fit the Composite Model and Experimental tangent delta results. An R^2 of 0.8835 implies a high correlation between experimental and Composite Model results. A dashpot with a viscosity of 1.25 GPa*s were used in models.

In addition to obtaining R^2 values, the trend line can also be used to obtain a root mean squared error (RMSE) value. RMSE can furthermore be used as a comparison between the two models where statistical options are limited. The RMSE was obtained by using the trend line equation and observed values to find the residual (r), which can be then used in the following equation:

$$RMSE = \sqrt{\frac{\sum_{i=1}^4 r^2}{n}} \quad (17)$$

where n is the number of observed values.

The obtained RMSE from cranial Composite and Half Unit Cell Models can be seen in table 11.

Table 11: Summary table of RMSE and R^2 values for cranial Composite and Half Unit Cell models.

Model	R^2 (Individual Points)	R^2 (Averaged Points)	RMSE
Composite	0.8835	0.942	0.00538
Single Half Unit	0.874	N/A	0.00793

Data obtained from the Control, Caudal models were processed in a similar fashion as the Control, Cranial. Table 12 displays data that was obtained through the Composite Model, experimental DMA done in Henry Ford Hospital, and Mendoza's findings.

Table 12: Summary Table for Control, Caudal Testing. Experimental data were collected via DMA testing on six Rambouillet-Cross Ewes. Composite Model data were obtained through averaging tangent delta values for each test frequency. Reported data from Mendoza's "Normal D-spacing" Half Unit Cell Model were used as comparison [52]. Standard deviation and standard error can't be reported for the Half Unit Cell Model due to a lack of randomization or replicates. A dashpot with a viscosity of 1.25 GPa*s were used in models.

Model (Control, Caud)	Average/St. Deviation/ St. Error	Tangent Delta			
		1 Hz	3 Hz	9 Hz	15 Hz
Experimental Data	Average	0.08535	0.07196	0.04674	0.02336
Experimental Data	St. Deviation	0.01001	0.00935	0.00917	0.00908
Experimental Data	St. Error	0.00408	0.00382	0.00375	0.00371
Composite Model	Average	0.06308	0.03333	0.01783	0.00684
Composite Model	St. Deviation	0.00589	0.00261	0.00864	0.00015
Composite Model	St. Error	0.00295	0.00131	0.00432	0.00007
Half Unit Cell Model	Average	0.15459	0.10201	0.03829	0.02318

The average tangent delta values from the four Control, Caudal results for each frequency were plotted along with the experimental data and Mendoza's results (figure 33). The results of Mendoza's "Normal D-spacing" Half Unit Cell Model were once again displayed in the plot. This was because the Control, Caudal model's geometric features were based off of the "Normal D-spacing" model. The Control, Caudal Composite Model displayed similar trends in tangent delta throughout the tested frequencies to the Control, Cranial Composite Model.

Figure 33: Plot for all tangent delta values from DMA experimental data, Mendoza's "Normal D-spacing" Half Unit Cell Model results [52], and the Composite Model for Control, Caudal sectors. The tangent delta data from the Composite Model were averaged for each frequency. Standard error bars were applied to each test frequency for experimental and Composite Model results. A dashpot with a viscosity of 1.25 GPa*s were used in models.

The R^2 and RMSE values were also obtained in a similar fashion. The experimental data was plotted against the Composite Model's resulting tangent delta as seen in figure 34. The plots and trend line equation that were used to determine the RMSE value for the Control, Caudal Composite Model can be seen in appendix F. A table of all R^2 and RMSE values for caudal Composite and Half Unit Cell Models are shown in figure 13.

Figure 34: A linear best-fit trend line was used to fit the Composite Model and Experimental tangent delta results. An R^2 of 0.8000 implies a high correlation between experimental and Composite Model results. A dashpot with a viscosity of 1.25 GPa*s were used in models.

Table 13: Summary table of RMSE and R^2 values for caudal Composite and Half Unit Cell models.

Model	R^2 (Individual Points)	R^2 (Averaged Points)	RMSE
Composite	0.8000	0.8833	0.00813
Single Half Unit	0.7469	N/A	0.00944

An in-depth statistical analysis can be utilized to compare the Composite Models and their respective sectors of the experiment. This can be done because of the random factors integrated into the Composite Model. Though p-values were calculated by a simple two-sample t-test in excel (seen in figure 32 and 34), a more extensive statistical analysis can be done to examine the explanatory variables in the Composite Model.

A two-way ANOVA analysis was performed with SAS through the Statistics Department of California Polytechnic State University, San Luis Obispo. For each model types, an analysis looked into two statistical treatments: Model type and Frequency and their interactions. Each ANOVA was analyzed at a 5% overall significance level. This would allow for a 1.67% individual significance level to test on the effects. The SAS ANOVA output for Control, Cranial sheep could be seen in table 14. The full SAS output can be seen in

Table 14: Two-Way ANOVA Output from SAS for Control, Cranial Sheep. A 5% overall significance level was utilized to determine the effects of each statistical treatment and their interactions. At a p-value of 0.0284, the interaction between Model Type and Frequency was found to be not significant in explaining the effects of tangent delta. Thus conclusions on the main effects can only be made.

Type 3 Tests of Fixed Effects				
Effect	Num DF	Den DF	F Value	Pr > F
Model	1	8	26.66	0.0009
Freq	3	24	101.67	<.0001
Model*Freq	3	24	3.59	0.0284

appendix G for both Composite Model types.

That statistical output seen in table 14 can be used to draw on information about the Experimental and Composite Models on Control, Cranial sheep. Because the interaction between Model Type and Frequency was not found to be significant ($p = 0.0284 > 0.0167 = \alpha$), conclusions on the main effects can be made. With a p-value of 0.0009 and <0.0001, the following conclusion can be made for Control, Cranial sheep: At a 5% overall significance level, Model Type and Frequency has an effect on the mean tangent delta for all experimental and Composite Model data for Control, Cranial sheep.

A pairwise comparison was done to evaluate which underlying variable causes a difference in tangent delta. Since the interaction was found not to be significant, only the main effects can be used in this pairwise comparison. For this analysis a Tukey-Kramer pairwise comparison was utilized. The reasoning behind this is to reduce the type 1 error by narrowing confidence intervals. Additionally, it compares all parts of the treatments. The main effects plot for tangent delta from this analysis can be seen in figure 35 and the complete results can be seen in appendix G.

Figure 35: Main effect plots were created to view the effects the explanatory variables have on tangent delta. A) Main effect plot for Test Frequency. B) Main effect plot for Model Type. Data points with like markers are not significantly different from one another.

Another ANOVA was performed on Control, Caudal sheep. The SAS output for this ANOVA can be seen in table 15.

Table 15: Two-Way ANOVA Output from SAS for Control, Caudal Sheep. A 5% overall significance level was utilized to determine the effects of each statistical treatment and their interactions. At a p-value of less than 0.0001, the interaction between Model Type and Frequency was found to be significant in explaining the effects of tangent delta. Thus conclusions on purely the main effects can't be made.

Type 3 Tests of Fixed Effects				
Effect	Num DF	Den DF	F Value	Pr > F
Model	1	8	38.51	0.0003
Freq	3	24	150.19	<.0001
Model*Freq	3	24	13.38	<.0001

A different conclusion can be drawn with the ANOVA analysis done for Control, Caudal sheep. Since the interaction between the Model type and Frequency was found significant ($p \leq 0.0001 < 0.0167 = \alpha$), conclusions can't be drawn using the main effects individually. Thus, at a 5% overall significance level, this study is convinced that the effect of Model Type on tangent delta depends on the test frequency for all experimental and Composite Model data for Control, Caudal sheep.

Since the interaction between Model Type and Frequency was found to be significant,

a Tukey-Kramer pairwise comparison was done to determine at which test frequency the Composite Model and experimental models differ. The complete results of this comparison can be seen in appendix G. The resulting interaction plot from the comparison can be seen in figure 36.

Figure 36: An interaction plot was developed to view at which test frequency the Composite Model and experimental models differ. The interaction of Model Type and Test Frequency displayed. Data points with like markers are not significantly different from one another.

4 Discussion

As discussed in section 1.6, there are many parameters that affect the mechanical properties of whole bone. Animal models can be used to measure the influence these parameters have in bone properties. The Rambouillet cross ewes as an animal model were beneficial to this study in that they were docile and readily available. However, the differences in their hormonal pattern compared to humans are a large limitation in this study. This fact can be somewhat disregarded in this study due to the early stages of the research. But as models become more complex, the limitation needs to be addressed by finding a better suited animal model.

The underlying reasoning behind bone's unique mechanical properties is still relatively unknown. Although bone is commonly considered a two-phase biomaterial consisting of collagen and mineral, the tissue contains many more characteristics that may be attributed to the viscoelastic property. There are many theories on why bone exhibits this characteristic. Some researchers believe that the biphasic structure of bone may be the root cause of its properties while others are convinced that cement lines surrounding Haversian bone are the main contributors to bone's viscoelasticity [7, 64].

This study specifically looked into D-spacing distribution and its affect on bone's viscoelasticity. D-spacing describes the dimension between staggered collagen molecules. Even though this parameter was first characterized in 1942, little research has been done on determining their function [22, 19]. When looking into this study, it was believed that the distribution of D-spacing would play a substantial role in bone's viscoelasticity. This was hypothesized because D-spacing dictates the amount of mineralization in a collagen fibril and the packing factor of tropocollagen in a region of a fibril. These two parameters have the ability of determining viscoelasticity in that amount of mineralization corresponds to bone's stiffness and a more dense network of tropocollagen creates a stronger fibril [93].

Computer simulations have been a great solution to costly and time-consuming physical experimentation. However, in order for a computer simulation to be effective, it must be validated against experimental findings or through hand calculations. The experimental data from the DMA testing on sectors of sheep ulna/radius were utilized to validate the

Composite Model developed in Abaqus.

The DMA testing was performed by Henry Ford Hospital and the tangent delta data was collected for a multitude of frequencies. However, for this study, only data from 1, 3, 9, and 15 Hz were used for validating the Composite Model.

Tangent of delta (tangent delta) is the effectiveness of a material's dampening capabilities under loading conditions. It can be represented as the ratio of the loss modulus (E_{loss}) to the stored modulus ($E_{storage}$) as seen below. The loss modulus is the material's ability to dissipate energy while the stored modulus characterized similarly to the material's stiffness (or the material's ability to store energy). The larger the value for tangent delta, the more efficient the material can withstand energy and dissipate it. As tangent delta decreases (seen in figure 29 where frequency is increased), bone's ability to dampen incoming stress will decrease. A perfectly elastic material will have a tangent delta of zero.

$$\tan \delta = \frac{E_{loss}}{E_{storage}} \quad (18)$$

To determine tangent delta for each Composite Model, the phase shift between the applied stress and the strain was measured. This can be done because tangent delta is directly related to the phase shift between the stress and strain response for a given material [55]. To reproduce this viscoelastic characteristic in a computer model, time-dependent material properties are required.

A user subroutine viscoelastic material property (UMAT) was used in the Composite Model. This was originally created by Dr. Frank Richter [89] and later modified by Mendoza for his Master's thesis [52]. The viscoelastic properties were created to imitate the response of rheological elements in the arrangement of the Kelvin-Voigt form of the Standard Linear Solid. This UMAT was assigned to all collagen sections of the model while the hydroxyapatite sections were treated as a perfectly elastic material.

Biological relevance was a main concern for this study. This is especially important when working on a biological based computational model. If the model isn't biologically relevant, data obtained in this study may be useless. However, there were many steps taken to ensure a high biological relevance in this study. The dimensions of each randomized

half unit cell were largely influenced by the AFM ewe study done by the Department of Chemistry at University of Michigan, Ann Arbor. Since the standard deviation used for the Gaussian distribution function was inputted at approximately 1 nm for both models, the within fibril D-spacing variation matches reported values. Additionally, since D-spacing has only been observed within 40 μm at a time, the 100 half unit cells of 67 +/- 1 nm in series easily falls within this range.

A spacer code was also implemented into the model generation Python script to provide an appropriate biological ratio of collagen to hydroxyapatite in the Composite Models. This spacer code was added because of the nature of two offset lengths at the free end in the model generation code. If this offset characteristic is not addressed, the applied load will produce unwanted notches and potential stress concentrations. There are two portions to this spacer addition to the Python script. The first being a function to add small spacer composed collagen onto the shorter length of the model to produce a completely flat surface at the free end of the model. If the offset was large enough to add an extra half unit cell (67 nm), the code will initiate the second portion in continually applying a 67 nm long half unit cell until the offset length is less than 67 nm. Then it would apply a spacer composed of collagen material. Regardless if both portions are initialized, the Python script will always check if the material composition (ratio of collagen to hydroxyapatite) of the final half unit is not altered by more than 5%. If the spacer altered the composition by more than 5%, the generated Composite Model was rejected from use. This safeguard will ensure a tight tolerance in maintaining a high biological relevance.

Four variant Composite Models were created for each ulna/radius sector (caudal and cranial), making a total of eight models. Models dimensions were randomly determined via a "Normal.Gaussian" function in the model generation Python script using mean and standard deviation data from AFM testing. Each variant model was simulated with their appropriate loading conditions at the four testing frequencies, making a total of 24 simulation jobs. After job completion, deformation data for each node was imported into a MATLAB script and tangent delta was calculated. Since there was a randomization factor incorporated in the best-fit function of the code, the MATLAB script was run ten times for each set of node data and averaged.

The trend throughout the test frequencies expressed by the Composite model correlates with literature findings. In order to protect and support the human body, bone must have the ability to stiffen itself accordingly for different types of impacts. In a slow impact, such as walking, bone will be less stiff and act more like a spring, resulting in a higher tangent delta. However, for a fast impact, like jumping, bone will become stiffer to absorb more energy, causing a lower tangent delta [94]. Observing this trend in the output data helps establish validity in the Composite Model.

Visually, the Half Unit Cell Model performs well at higher test frequencies (9 and 15 Hz) for both bone sectors. However, there is a relatively large difference in tangent delta when compared to experimental data at lower test frequencies (1 and 3 Hz). Mendoza specifically formed his viscoelastic parameters so that his model closely matches the characteristics of the experimental results explaining this large difference in performance between the high and low test frequencies. The Composite Model results are more consistent in its performance throughout the four tested frequencies. Despite the fact that, visually, it may look like the addition of a distribution of D-spacing increases the overall performance of the computational model, a more sophisticated performance analysis is required.

The tangent delta data from the Composite Models was compared to experimental data for their corresponding sectors. Evaluation of model accuracy was done by plotting experimental results against Composite Model results for each model. The plot would produce a trend line resulting in a coefficient of determination (R^2) and a root mean square error (RMSE) value. The R^2 and RMSE values were used to evaluate how well the Composite models represent viscoelasticity of their corresponding cortical bone tissue. Additionally, these values were used to compare against the Half Unit Cell Model. Since Mendoza lacked randomization and only reported single tangent delta values for each testing frequency in his models, there is an absence of statistical significance. However, a similar analysis was performed on his data to obtain an R^2 and RMSE value as a means of comparisons between the Half Unit Cell and Composite models. Tables 11 and 13 display all R^2 and RMSE values obtained from this study.

The R^2 comparison between the Composite Model and the Half Unit Cell Model solidified the improvements made by the Composite Model. Two R^2 values were calculated for

each sector's Composite Model. The first R^2 originates from using all four model variants and the other utilizes the averages for each test frequency. Although averaging removes some variability (and a larger R^2 value), it's a more "one-to-one" comparison to Mendoza's results because he only reported a single value for each test frequency.

Both Composite models generated larger R^2 values than the Half Unit Cell Model. The Control, Cranial Composite Model's R^2 value of 0.942 exceeds Mendoza's R^2 value of 0.874. When comparing the caudal models, the Composite Model produced a R^2 value of .8000 while the Half Unit Cell Model produced a R^2 value of 0.7469. Additionally, R^2 values generated by non-averaged tangent delta values were greater than R^2 values generated with Mendoza's data for both sectors of bone. This further strengthens these Composite Models in modeling bone's viscoelastic properties because the evaluation with the most variability has a stronger correlation with the experimental data.

In addition to comparing R^2 values, a comparison between each model's RMSE values was done to further confirm the improvements by adding a distribution of D-spacing. A smaller RMSE would display a greater accuracy in model result to experimental results. RMSE values were calculated by using averaged values of tangent delta for each test frequency at each sector's Composite Model. The Control, Cranial Composite Model had a smaller RMSE value of 0.00538 compared to 0.00793 calculated using Mendoza's data. Additionally, the Control, Caudal Model's RMSE value of 0.00813 is smaller than Mendoza's 0.0094. With the R^2 and RMSE value comparisons, the confidence in the Composite Model's performance in simulating bone viscoelastic properties has greatly increased.

These results show a lot of promise for the Composite Model. With the inclusion of multiple complexes in a 2 x100 arrangement, the model is capable of modeling different sectors of the ulna/radius bone of the ewes. With how the model building code is written, this brings great promise in modeling different types of skeletal tissue. The model builder is flexible enough that, with enough information, it is relatively easy to generate models of new bone types. This competence is extended in creating fibril sections of bone with metabolic disease, such as osteoporosis. In addition to the high capabilities of this model, the tangent delta values from the Composite model produced provide a better fit in the experimental data when compared to Mendoza's data.

The Composite Model has already shown promise in its ability to be fine-tuned for different bone tissue. A recent Master's thesis by Austin Cummings used the model generation Python script to develop Composite Models of OVX bone tissue [95]. Previous studies have found that the distribution of D-spacing in OVX bone is more homogenous than that of the control bone tissue [20]. Cummings' model was able to produce a more accurate viscoelastic response than previous computational models for experimental OVX bone tissue.

Although the comparisons between the two models are statistically limited due to the lack of variability in the Half Unit Cell Model, the variability in the model variants and randomization factors in model generation allows for a statistical analysis for each Composite Model type. A two-way ANOVA was performed with assistance from Dr. John Walker of the Statistics Department at California Polytechnic State University for Control, Caudal and Control, Cranial models. Due to the complexities of the statistical analysis, Statistical Analysis Software (SAS) was used for the analyses.

There were two core statistical treatments that were compared for the two-way ANOVA: Model Type (Composite and Experimental) and Test Frequency (1, 3, 9, and 15 Hz). In addition, an interaction between the two core treatments was also tested to determine if the effect Test Frequency has on tangent delta is dependent on the type of model. Both analyses were performed at an overall significance level of 5%. This significance level was determined because it provided an individual significance level of 1.67% for each of the three individual tests (Model Type, Test Frequency, and Model Type x Test Frequency) while decreasing the chances of Type 1 error. Reducing the chances of Type 1 error will reduce the probability of false positives. Additionally, comparisons could not be made between the caudal and cranial Composite Models because of the differences in paired and unpaired datasets. Since caudal and cranial data was extracted from the same ulna/radius, the AFM data is paired. The caudal and cranial Composite Models were generated from two separate sets of data, making them unpaired.

Although a proper statistical analysis is not possible in comparing the Caudal and Cranial Composite Models, conclusions can be drawn through visual inspection. Both Composite Models follow a similar trend. However, the Control, Caudal Models had lower

average tangent delta compared to the average tangent delta produced by the Control, Cranial Models when under a 3, 9, and 15 Hz testing frequency. This is not the case when comparing the caudal and cranial sectors from the experimental DMA testing, where the Control, Caudal sectors had a higher average tangent delta values than the Control, Cranial sectors for all test frequencies. This difference in trends shouldn't raise an alarm in the adequacy of the Composite Model. Much of the differences between the tangent deltas in the Composite Models are minute and the tangent deltas may not be significantly different from one another.

Statistical analysis on the Control, Cranial sector has shown that the interaction between Model Type and Test Frequency was not significant, meaning that the Model Type and Test Frequency are independent from one another. Thus, their effects were analyzed individually. The SAS analysis found that there is a significant difference in the effects based on whether the results came from the experimental data or a Composite Model. This seems less than ideal because the goal for this study is to have the Composite Model exhibit similar viscoelastic properties to the experimental data. However, the differences seem to be small enough that the difference in effects can be improved upon in the next model iterations. The SAS analysis also found that the effects on test frequency were all significantly different from one another. This verifies the Composite Model's ability to exhibit variable viscoelasticity under different loading frequencies.

The SAS analysis done on the Control, Caudal sector determined that the interaction between Model Type and Test Frequency was significant. This result only allows conclusions to be drawn on the interaction and not on the main effects from Model Types or Test Frequency. However, since the interaction is found to be significant, it is not possible to draw any kind of conclusions between the Complex Model and the experimental data because the effect that Model Type has on the tangent delta is dependent on the Test Frequency.

Statistical capability was a large limitation in this study. Since there wasn't any kind of variability provided in the previous study, statistical analysis couldn't be performed between the Half Unit Cell Model and the Composite model. However, R^2 and RMSE values allowed for a way to compare each of the model's ability to accurately represent the experimental testing. The data and results of this study will allow future iterations of this model to

perform stronger statistical analyses.

Another limitation in this study is the use of sheep as an animal model for human bone. Although they are currently a better fit than most animals, they fall short in becoming a perfectly accurate representation of human bone. For example, they don't experience the same hormonal patterns as humans [69]. Sheep are seasonal breeders while humans are yearlong breeders (estrous vs. menstrual cycles). However, this downfall doesn't necessarily void the use of sheep as an animal model for this study. They exhibit similar bone structure and loading conditions as humans and their usage is common among many studies [52, 79]. Since this computational model is still relatively new, the usage of sheep doesn't cause immediate problems. Additionally, the model can be easily adjusted for usage of a more representative animal model in later iterations.

Mineral volume fraction was modeled to be consistently 0.30 throughout the whole model. This parameter was chosen to keep consistent with previous studies [51, 52, 83]. Even though this isn't biologically true, it was done to simplify the model and reduce computational time. This can be improved in later iterations by incorporating a randomized function similar to the D-spacing dimensions done in the model generation code. However, the exact relationship is currently unknown making this limitation difficult to alleviate.

Another assumption that isn't biologically accurate is the material property assigned to the mineralized sections of the model. For this model, hydroxyapatite was assumed to be a fully elastic isotropic material with an elastic modulus of 100 GPa and Poisson's ratio of 0.28. These specific parameters for hydroxyapatite were chosen because of the findings indicating that mineral relates more to the stiffness of the material [96] and they were based on the previous studies [15, 51]. The exact value for elastic modulus is still unknown. Research has shown that heavily mineralized whale ear bone has an elastic modulus as low as 34.1 GPa [97], making a large range for an appropriate elastic modulus of hydroxyapatite. Given that it is relatively easy to change the modulus on the Abaqus GUI or the input file, this limitation can be resolved once a more definite modulus value is determined.

Despite the fact that run times for each job took 8-12 hours, they were manageable with access to three computers and a server dedicated for FEA jobs that were provided by Dr. Scott Hazelwood of the Biomedical Engineering Department at California Polytechnic State

University. With 32 variations of jobs to run, it required at least a week to complete all job analyses. The completion time for jobs will exponentially increase as iterations of the model become more complex. Additionally, each job completion will produce 100 GB output database files (ODB). ODB file sizes were decreased down to roughly 4 GB by limiting the field and history outputs in the input file to just recording displacement data. However, with the large amount of files, hard drive space quickly becomes consumed. These limit how complex the models can become. This can be mended by increasing computational power.

The Composite Model replicated the collagen and hydroxyapatite and neighboring complexes to be perfectly bonded. This isn't exactly true. The modeling done by Siegmund et al. was interested in inspecting the significance in contact forces that connected collagen and hydroxyapatite sections [51]. Collagen molecules rely on this relationship as it dictates their tensile strength and packing factor [93]. The research done by Siegmund et al. concluded that including these contact forces is important for accurate modeling.

One of the largest improvements that can be implemented into the model is the refinement of the viscoelastic parameters. Although the exact mechanism responsible for bone's viscoelastic properties is still relatively unknown, there is some research that points to multiple features of bone for its cause. Some researchers suggest the presence of Haversian canals promote the viscoelastic properties [64] while others believe that the cross-linking of collagen molecules is related to this property [51]. Another theory is that the viscoelastic properties are due to bone being seen as a biphasic composite material [98]. This uncertainty and the lack of a proper biological rheological model [7] make it difficult to produce an accurate model.

The user subroutine UMAT, originally developed by Richter, models viscoelasticity based on the Kelvin-Voigt version of the Standard Linear Solid. This material is able to accurately express creep and stress relaxation, crucial viscoelastic responses exhibited by bone tissue. For this version, there are 3 components, two springs (E_1 and E_2) representing the elastic component, and one dashpot (n_1) representing a viscous component shown in figure 22. The key to modifying the Composite Model in order to produce data to fit the experimental dataset may lie in changing the combination of these three parameters.

An interesting observation in datasets seen in figure 36 is an offset in data between the

Composite Model and the experimental data. Most of the tangent delta data points from the Composite Model were not found to be significantly different than the tangent delta value from the succeeding test frequency of the experimental data suggesting that the data can be a better fit by simply shifting it. This shift may be possible by modifying the spring components and/or the dashpot component of the Kelvin-Voigt form of the Standard Linear Solid.

The exact values of the elastic components were determined by using an effective modulus ($E_{effective}$). An $E_{effective}$ of 2 GPa was used since it's a common value for collagen's elasticity [7, 15, 6]. Then E_1 and E_2 can be appropriately determined using the equation below. To achieve an $E_{effective}$ at 2 GPa, E_1 and E_2 were set to 3 GPa and 6 GPa respectively.

$$E_{effective} = \frac{E_1 E_2}{(E_1 + E_2)} \quad (19)$$

The effective modulus needs further refinement since the results of the model don't fit the experimental data exactly. However, determining a correct value may be problematic because the range of values is relatively large. The experimental elastic modulus of type 1 collagen is 5 GPa; however, it may produce worst results. In comparison, there have been some studies that use the elastic modulus of general collagen (not type 1 specific) as low as 200 MPa [99]. As a self-measurement, a single model was run with an effective modulus of 5 GPa and compared to a model of the same geometry with an effective modulus of 2 GPa. The model with a 5 GPa effect modulus produced less favorable results where tangent deltas were further offset below the targeted experimental data. Further iterations of this model can fine-tune the effective modulus to better fit the data. Since there is a wide range of acceptable moduli used in research, starting with a lower modulus than 2 GPa would be a great start to find better fitment since the trend seems to be that a lower effective modulus would produce larger tangent delta values. Altering the modulus of hydroxyapatite may also be an option for fine-tuning.

One concern to be noted with these effective values is that they may be inappropriate for such a small model. The Composite Model only models a small section of a collagen fiber

in the nano and micro scale. Assigning small collagen molecules with these macro-scale properties may be flawed.

The viscosity for the dashpot was determined experimentally. Initially, a viscosity of 4 GPa*s was used based on the Half Unit Cell Model. His goal was to fit tangent delta values at higher frequencies. When a 4GPa*s viscosity dashpot was used for the Composite Model, output tangent delta values underestimated experimental values throughout all test frequencies. Thus, a less viscous dashpot of 1.25 GPa*s was selected because of preliminary findings (figure 28) and was recommended by Mendoza's research [52]. However, a lower viscous dashpot may produce better results. Later iterations may want to perform more experiments to determine a viscosity that produces more accurate results.

Another option is to incorporate a variable dashpot. Including this may provide a much more complex response in the model. Increasing the complexity of the response may cause an increase in the non-linear force-relative velocity relationship of the vicious elements [100].

An additional improvement that can be done is to create a Composite Model by using parameters of individual bone samples rather than averaging data for all the samples. This would create a one-to-one relationship between the Composite Model and the bone sample. This change can be easily done by changing values of the mean and standard deviation at the early sections of the model generation code. Doing this allows for a more accurate model because these models will be more directly correlated with the individual bone samples.

The MATLAB code used to fit the output data to a sinusoidal waveform had a small amount of variability incorporated into it. To ensure a proper fit, initial amplitude values were adjusted so that R^2 values were at least 0.99. However, with the variability in the code, there were some occasional spikes in tangent delta values. The rate of spiked (noticeably larger) values were about 1 in every 10 trials. Spiked values were not omitted in order to keep all data that may provide information on viscoelastic modeling. These values were also included when averaging each model. For future iterations, the Cook's Distance should be analyzed to determine data points of high influence that can be omitted.

5 Conclusion

Bone tissue is a highly complex biomaterial that exhibits a viscoelastic material property difficult to accurately simulate through computer modeling. The underlying mechanisms of bone's behavior is still relatively unknown. This study aimed to look into the effects of multiple hydroxyapatite and collagen complexes on bone tissue's viscoelastic properties.

The results of this study show that the inclusion of a distribution in collagen D-spacing is required to accurately model this behavior through FEA. Previous versions of this model have provided great stepping-stones. Although these models were substantially simpler, the addition of a somewhat randomized factor in the distribution of D-spacing in a computational model has provided more accurate results. Between the two sections of ulna/radius that were modeled, the caudal models produced lower tangent deltas when compared to cranial sections at 3, 9, and 15 Hz. This is not the case for the experimental data, where the caudal sections had a larger tangent delta throughout the all testing frequencies. Although a distribution of D-spacing may not explain the differences in the viscoelastic properties between the different bone sections, it may be a step in the right direction. In addition, the model generation code is capable of producing models that can replicate viscoelastic characteristics of different sectors of bone. These don't mean that the Composite Model is perfect. There are plenty of improvements that can be made to increase the accuracy of the model.

Many different portions of the study may be refined to further increase the accuracy of this model. One to take note of is the viscoelastic component of the model. The parameters of the spring and dashpot elements of the rheological model may need further work to provide more accurate results.

By including a randomization factor to increase biological accuracy in the behavior of collagen D-spacing, the Composite Model has exhibited viscoelastic behavior more accurate to experimental data than its predecessors. The model was set up so that future iterations can easily be created. Many of these can be changed among the various codes in the study. With simple modifications, models of bone can be created to exhibit metabolic bone disease (i.e. postmenopausal osteoporosis). However, some of these changes would

require more research and time to determine appropriate parameter alterations. This model hopes to provide a framework to build upon that can be further refined to create a tool in examining the viscoelastic properties of bone and the effects of its disorders.

BIBLIOGRAPHY

- [1] Maria Felicia Faienza, Annamaria Ventura, Flaviana Marzano, and Luciano Cavallo. Postmenopausal osteoporosis: the role of immune system cells. *Clinical and Developmental Immunology*, 2013, 2013.
- [2] "what is osteoporosis?", Nof.org. National Osteoporosis Foundation. Web. July 2015.
- [3] William J Landis, Karen J Hodgens, James Arena, Min Ja Song, and Bruce F McEwen. Structural relations between collagen and mineral in bone as determined by high voltage electron microscopic tomography. *Microscopy research and technique*, 33(2):192–202, 1996.
- [4] Ming Fang, Elizabeth L Goldstein, A Simon Turner, Clifford M Les, Bradford G Orr, Gary J Fisher, Kathleen B Welch, Edward D Rothman, and Mark M Banaszak Holl. Type i collagen d-spacing in fibril bundles of dermis, tendon, and bone: bridging between nano-and micro-level tissue hierarchy. *ACS nano*, 6(11):9503–9514, 2012.
- [5] C Ross Ethier and Craig A Simmons. *Introductory biomechanics: from cells to organisms*. Cambridge University Press, 2007.
- [6] Donald L Bartel, Dwight T Davy, and Tony M Keaveny. *Orthopaedic biomechanics: mechanics and design in musculoskeletal systems*. Prentice Hall, 2006.
- [7] R Bruce Martin, David Burr, and Neil A Sharkey. *Skeletal tissue mechanics*. Springer Science & Business Media, 2013.
- [8] Sylvie Ricard-Blum. The collagen family. *Cold Spring Harbor perspectives in biology*, 3(1):a004978, 2011.
- [9] Aurelien Gourrier, Wolfgang Wagermaier, Manfred Burghammer, Donna Lammie, Himadri S Gupta, Peter Fratzl, Christian Riekkel, Tim J Wess, and Oskar Paris. Scanning x-ray imaging with small-angle scattering contrast. *Journal of Applied Crystallography*, 40:s78–s82, 2007.
- [10] Stephen Weiner and Wolfie Traub. Organization of hydroxyapatite crystals within collagen fibrils. *FEBS letters*, 206(2):262–266, 1986.
- [11] Stephan Weiner and Wolfie Traub. Bone structure: from angstroms to microns. *The FASEB journal*, 6(3):879–885, 1992.
- [12] Roderic Lakes. Materials with structural hierarchy. *Nature*, 361(6412):511–515, 1993.
- [13] A Boyde. Electron microscopy of the mineralizing front. *Metab. Bone Dis. Relat. Res*, 2(S):69–78, 1980.
- [14] Marie-Madeleine Giraud-Guille. Twisted plywood architecture of collagen fibrils in human compact bone osteons. *Calcified tissue international*, 42(3):167–180, 1988.
- [15] John D Currey. *Bones: structure and mechanics*. Princeton University Press, 2002.

- [16] Joseph M Wallace, Blake Erickson, Clifford M Les, Bradford G Orr, and Mark M Banaszak Holl. Distribution of type i collagen morphologies in bone: relation to estrogen depletion. *Bone*, 46(5):1349–1354, 2010.
- [17] Ming Fang. *Type I Collagen Nanomorphology in Relation to Disease, Tissue Hierarchy, and Fibrillogenesis*. PhD thesis, Henry Ford Hospital, 2013.
- [18] Joseph M Wallace, Qishui Chen, Ming Fang, Blake Erickson, Bradford G Orr, and Mark M Banaszak Holl. Type i collagen exists as a distribution of nanoscale morphologies in teeth, bones, and tendons. *Langmuir*, 26(10):7349–7354, 2010.
- [19] Francis O Schmitt, Cecil E Hall, and Marie A Jakus. Electron microscope investigations of the structure of collagen. *Journal of Cellular and Comparative Physiology*, 20(1):11–33, 1942.
- [20] CM Les, CL Pechey, EM Michels, M Fang, JM MacLeay, AS Turner, and MM Banaszak Holl. Estrogen depletion is associated with a loss in anatomic variability in both bone collagen d-spacing, and in bone material pre-stress. In *Proc Orthop Res Soc*, volume 36, page 2266, 2011.
- [21] J Lawrence Katz. Composite material models for cortical bone. *Mechanical properties of bone*, 45:171–184, 1981.
- [22] Gopalasamudram Narayana Ramachandran. *Aspects of protein structure: proceedings of a symposium held in Madras 14-18 January 1963 and organized by the University of Madras*. Academic Press, 1963.
- [23] Julie T Lin and Joseph M Lane. Osteoporosis: a review. *Clinical orthopaedics and related research*, 425:126–134, 2004.
- [24] Steven R Cummings and L Joseph Melton. Epidemiology and outcomes of osteoporotic fractures. *The Lancet*, 359(9319):1761–1767, 2002.
- [25] US Department of Health, Human Services, et al. Bone health and osteoporosis: a report of the surgeon general, 2004.
- [26] Philip Sambrook and Cyrus Cooper. Osteoporosis. *LANCET*, 367(9527):2010–2018, June 2006.
- [27] Robert Marcus, David Feldman, Dorothy Nelson, and Clifford J Rosen. *Fundamentals of osteoporosis*. Academic Press, 2009.
- [28] "what is osteoporosis?", www.iofbonehealth.org. International Osteoporosis Foundation. Web. July 2015.
- [29] World Health Organization et al. Assessment of fracture risk and its application to screening for postmenopausal osteoporosis: report of a who study group [meeting held in rome from 22 to 25 june 1992]. 1994.
- [30] John Anthony Kanis, World Health Organization, et al. *Assessment of osteoporosis at the primary health care level*. WHO Collaborating Centre for Metabolic Bone Diseases, University of Sheffield Medical School, 2008.

- [31] Pedro Vilela and Teresa Nunes. Osteoporosis. *Neuroradiology*, 53(S1):185–189, August 2011.
- [32] Brian C Lentle and Jerilynn C Prior. Osteoporosis: What a clinician expects to learn from a patient’s bone density examination 1. *Radiology*, 228(3):620–628, 2003.
- [33] Lance G Dasher, Christopher D Newton, and Leon Lenchik. Dual x-ray absorptiometry in today’s clinical practice. *Radiologic Clinics of North America*, 48(3):541–560, 2010.
- [34] F Quereda, N Mendoza, MA Olalla, F Baró, and M Durán. Prophylactic approach for asymptomatic post-menopausal women: Osteoporosis. *Maturitas*, 52:38–45, 2005.
- [35] Robert Lindsay. The menopause and osteoporosis. *Obstetrics & Gynecology*, 87(2):16S–19S, 1996.
- [36] Patrizia D’Amelio, Anastasia Grimaldi, Stefania Di Bella, Stefano ZM Brianza, Maria Angela Cristofaro, Cristina Tamone, Giuliana Giribaldi, Daniela Ulliers, Gian P Pescarmona, and Giancarlo Isaia. Estrogen deficiency increases osteoclastogenesis up-regulating t cells activity: a key mechanism in osteoporosis. *Bone*, 43(1):92–100, 2008.
- [37] Robert Lindsay. Prevention of osteoporosis. *Clinical orthopaedics and related research*, 222:44–59, 1987.
- [38] MR Allen. Skeletal accumulation of bisphosphonates: implications for osteoporosis treatment. *Expert opinion on drug metabolism & toxicology*, 4(11):1371–1378, 2008.
- [39] Clarita V Odvina, Joseph E Zerwekh, D Sudhaker Rao, Naim Maalouf, Frank A Gottschalk, and Charles YC Pak. Severely suppressed bone turnover: a potential complication of alendronate therapy. *The Journal of Clinical Endocrinology & Metabolism*, 90(3):1294–1301, 2005.
- [40] Brett A Lenart, Dean G Lorch, and Joseph M Lane. Atypical fractures of the femoral diaphysis in postmenopausal women taking alendronate. *New England Journal of Medicine*, 358(12):1304–1306, 2008.
- [41] Jian Sheng Chen and Philip N Sambrook. Antiresorptive therapies for osteoporosis: a clinical overview. *Nature Reviews Endocrinology*, 8(2):81–91, 2012.
- [42] Elizabeth Barrett-Connor, Lori Mosca, Peter Collins, Mary Jane Geiger, Deborah Grady, Marcel Kornitzer, Michelle A McNabb, and Nanette K Wenger. Effects of raloxifene on cardiovascular events and breast cancer in postmenopausal women. *New England Journal of Medicine*, 355(2):125–137, 2006.
- [43] Jurga Adomaityte, Maria Farooq, Rehan Qayyum, et al. Effect of raloxifene therapy on venous thromboembolism in postmenopausal women: a meta-analysis. *Thromb Haemost*, 99(2):338–342, 2008.
- [44] Paul D Miller, Michael A Bolognese, E Michael Lewiecki, Michael R McClung, Beiyong Ding, Matthew Austin, Yu Liu, and Javier San Martin. Effect of denosumab

- on bone density and turnover in postmenopausal women with low bone mass after long-term continued, discontinued, and restarting of therapy: a randomized blinded phase 2 clinical trial. *Bone*, 43(2):222–229, 2008.
- [45] Steven R Cummings, Javier San Martin, Michael R McClung, Ethel S Siris, Richard Eastell, Ian R Reid, Pierre Delmas, Holly B Zoog, Matt Austin, Andrea Wang, et al. Denosumab for prevention of fractures in postmenopausal women with osteoporosis. *New England Journal of Medicine*, 361(8):756–765, 2009.
 - [46] Stephanie KA Blick, Sohita Dhillon, and Susan J Keam. Teriparatide. *Drugs*, 68(18):2709–2737, 2008.
 - [47] DR Carter and Wilson C Hayes. Fatigue life of compact bone—i effects of stress amplitude, temperature and density. *Journal of Biomechanics*, 9(1):27–34, 1976.
 - [48] DR Carter and Wilson C Hayes. Compact bone fatigue damage—i. residual strength and stiffness. *Journal of Biomechanics*, 10(5):325–337, 1977.
 - [49] David B Burr, Mitchell B Schaffler, and Richard G Frederickson. Composition of the cement line and its possible mechanical role as a local interface in human compact bone. *Journal of Biomechanics*, 21(11):939–945, 1988.
 - [50] "resin restorative materials", pocketdentistry.com. Pocket Dentistry. Web. Aug 2015.
 - [51] Thomas Siegmund, Matthew R Allen, and David B Burr. Failure of mineralized collagen fibrils: modeling the role of collagen cross-linking. *Journal of biomechanics*, 41(7):1427–1435, 2008.
 - [52] Miguel Angel Mendoza. *The Effects of Variation in Collagen D-spacing on Compact Bone Viscoelasticity: A Finite Element Analysis*. PhD thesis, California Polytechnic State University: San Luis Obispo, August 2013.
 - [53] TM Wright and WC Hayes. Tensile testing of bone over a wide range of strain rates: effects of strain rate, microstructure and density. *Medical and biological engineering*, 14(6):671–680, 1976.
 - [54] David Roylance. Engineering viscoelasticity. *Department of Materials Science and Engineering—Massachusetts Institute of Technology, Cambridge MA*, 2139:1–37, 2001.
 - [55] Roderic S Lakes. *Viscoelastic solids*, volume 9. CRC press, 1998.
 - [56] James H McElhaney. Dynamic response of bone and muscle tissue. *Journal of applied physiology*, 21(4):1231–1236, 1966.
 - [57] JGM Ramaekers. The rheological behaviour of skeletal material originating from several classes of vertebrates. *Netherlands Journal of Zoology*, 29(2):166–176, 1978.
 - [58] Ronald F Gibson, Yu Chen, and Hui Zhao. Improvement of vibration damping capacity and fracture toughness in composite laminates by the use of polymeric interleaves. *Journal of engineering materials and technology*, 123(3):309–314, 2001.

- [59] PM Buechner, D Stone, and RS Lakes. Viscoelastic behavior of superplastic 37 wt% pb 63 wt% sn over a wide range of frequency and time. *Scripta materialia*, 41(5):561–567, 1999.
- [60] Roderic S Lakes, J Lawrence Katz, and Sanford S Sternstein. Viscoelastic properties of wet cortical bone—i. torsional and biaxial studies. *Journal of Biomechanics*, 12(9):657–678, 1979.
- [61] Junro Yamashita, Xiaoe Li, Benjamin R Furman, H Ralph Rawls, Xiaodu Wang, and C Agrawal. Collagen and bone viscoelasticity: a dynamic mechanical analysis. *Journal of biomedical materials research*, 63(1):31–36, 2002.
- [62] Henry W Haslach Jr. *Maximum dissipation non-equilibrium thermodynamics and its geometric structure*. Springer Science & Business Media, 2011.
- [63] Paul Macioce. Viscoelastic damping 101. *Sound and Vibration*, 1, 2003.
- [64] Peter Frasca. Scanning-electron microscopy studies of ‘ground substance’ in the cement lines, resting lines, hypercalcified rings and reversal lines of human cortical bone. *Cells Tissues Organs*, 109(2):115–121, 1981.
- [65] M Fois, A Lamure, MJ Fauran, and C Lacabanne. Study of human cortical bone and demineralized human cortical bone viscoelasticity. *Journal of applied polymer science*, 79(14):2527–2533, 2001.
- [66] Marvin Johnson and J Lawrence Katz. Some new developments in the rheology of bone. *Biorheology. Supplement: the official journal of the International Society of Biorheology*, 1:169–174, 1983.
- [67] BJ Cohen and FM Loew. *Laboratory Anime Medicine: Historical Perspectives in Laboratory Anime Medicin*. Academic Press, 1984.
- [68] Kamran Kaveh, Rashid Ibrahim, Abu Bakar, Mohd Zuki Zakaria, Tengku Ibrahim, and Tengku Azmi. Osteoporosis induction in animal model. *American Journal of Animal and Veterinary Sciences*, 5(2):139–146, 2010.
- [69] MR Zarrinkalam, H Beard, CG Schultz, and RJ Moore. Validation of the sheep as a large animal model for the study of vertebral osteoporosis. *European Spine Journal*, 18(2):244–253, 2009.
- [70] Maria L Hillier and Lynne S Bell. Differentiating human bone from animal bone: a review of histological methods. *Journal of forensic sciences*, 52(2):249–263, 2007.
- [71] E Newman, AS Turner, and JD Wark. The potential of sheep for the study of osteoporosis: current status and comparison with other animal models. *Bone*, 16(4):S277–S284, 1995.
- [72] AS Turner. The sheep as a model for osteoporosis in humans. *The Veterinary Journal*, 163(3):232–239, 2002.
- [73] Ira Van Gieson. *Laboratory notes of technical methods for the nervous system*. publisher not identified, 1889.

- [74] B Lawrence Riggs and L Joseph Melton III. Involutional osteoporosis. *New England journal of medicine*, 314(26):1676–1686, 1986.
- [75] Keiko Sone, Takako Yamamoto-Sawamura, Sachi Kuwahara, Kazutoshi Nishijima, Tamio Ohno, Hiroaki Aoyama, and Shin Tanaka. Changes of estrous cycles with aging in female f344/n rats. *Experimental animals*, 56(2):139–148, 2007.
- [76] Susan Reinwald and David Burr. Review of nonprimate, large animal models for osteoporosis research. *Journal of Bone and Mineral Research*, 23(9):1353–1368, 2008.
- [77] E. M. Michels, C. L. Pechey, J. M. MacLeay, M. J. Mager, A. S. Turner, and C. M. Les. The interaction of estrogen depletion and dietary induced metabolic acidosis on bone time-dependent structural properties: A single branch, or a bundle of sticks?
- [78] CM Les, CA Spence, JL Vance, GT Christopherson, B Patel, AS Turner, GW Divine, and DP Fyhrie. Determinants of ovine compact bone viscoelastic properties: effects of architecture, mineralization, and remodeling. *Bone*, 35(3):729–738, 2004.
- [79] CM Les, JL Vance, GT Christopherson, AS Turner, GW Divine, and DP Fyhrie. Long-term ovariectomy decreases ovine compact bone viscoelasticity. *Journal of orthopaedic research*, 23(4):869–876, 2005.
- [80] Ralf Oheim, Michael Amling, Anita Ignatius, and Pia Pogoda. Large animal model for osteoporosis in humans: the ewe. *Eur Cell Mater*, 24:372–385, 2012.
- [81] SL O’Connell. The sheep as an experimental model for osteoporosis. *Department of Medicine. The University of Melbourne, Melbourne*, 1999.
- [82] John A Petruska and Alan J Hodge. A subunit model for the tropocollagen macromolecule. *Proceedings of the National Academy of Sciences of the United States of America*, 51(5):871, 1964.
- [83] Ingomar Jäger and Peter Fratzl. Mineralized collagen fibrils: a mechanical model with a staggered arrangement of mineral particles. *Biophysical Journal*, 79(4):1737–1746, 2000.
- [84] JD Currey. Effects of differences in mineralization on the mechanical properties of bone. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 304(1121):509–518, 1984.
- [85] Andreas Fritsch and Christian Hellmich. Universal microstructural patterns in cortical and trabecular, extracellular and extravascular bone materials: micromechanics-based prediction of anisotropic elasticity. *Journal of Theoretical Biology*, 244(4):597–620, 2007.
- [86] Markus J Buehler. Atomistic and continuum modeling of mechanical properties of collagen: elasticity, fracture, and self-assembly. *Journal of Materials Research*, 21(08):1947–1961, 2006.
- [87] CM Les, CA Spence, JL Vance, GT Christopherson, B Patel, AS Turner, GW Divine, and DP Fyhrie. Determinants of ovine compact bone viscoelastic properties: effects of architecture, mineralization, and remodeling. *Bone*, 35(3):729–738, 2004.

- [88] Dassault Systemes. Abaqus 6.10 online documentation. *Abaqus User Subroutines Reference Manual*, 2010.
- [89] Frank Richter. Upsetting and viscoelasticity of vitreous sio2: experiments, interpretation and simulation. *Technischen Universität, Berlin*, page 239, 2006.
- [90] Tara N Shepherd, Jingzhou Zhang, Timothy C Ovaert, Ryan K Roeder, and Glen L Niebur. Direct comparison of nanoindentation and macroscopic measurements of bone viscoelasticity. *Journal of the mechanical behavior of biomedical materials*, 4(8):2055–2062, 2011.
- [91] JL Katz and K Ukraincik. On the anisotropic elastic properties of hydroxyapatite. *Journal of biomechanics*, 4(3):221–227, 1971.
- [92] B Viswanath, R Raghavan, U Ramamurty, and N Ravishankar. Mechanical properties and anisotropy in hydroxyapatite single crystals. *Scripta Materialia*, 57(4):361–364, 2007.
- [93] Joseph PRO Orgel, Thomas C Irving, Andrew Miller, and Tim J Wess. Microfibrillar structure of type i collagen in situ. *Proceedings of the National Academy of Sciences*, 103(24):9001–9005, 2006.
- [94] R Schaller and Gilbert Fantozzi. 8.6 damping and toughness. In *Materials Science Forum*, volume 366, pages 615–620. Trans Tech Publ, 2001.
- [95] Austin Cummings. *A Finite Element Analysis on the Viscoelasticity of Post-menopausal Compact Bone Utilizing a Complex Collagen D-spacing Model*. PhD thesis, California Polytechnic State University: San Luis Obispo, July 2015.
- [96] DB Burr. The contribution of the organic matrix to bone’s material properties. *Bone*, 31(1):8–11, 2002.
- [97] John D Currey. The design of mineralised hard tissues for their mechanical functions. *Journal of Experimental Biology*, 202(23):3285–3294, 1999.
- [98] Joseph M Wallace, Bradford G Orr, Joan C Marini, and Mark M Banaszak Holl. Nanoscale morphology of type i collagen is altered in the brtl mouse model of osteogenesis imperfecta. *Journal of structural biology*, 173(1):146–152, 2011.
- [99] Colin A Grant, David J Brockwell, Sheena E Radford, and Neil H Thomson. Tuning the elastic modulus of hydrated collagen fibrils. *Biophysical journal*, 97(11):2985–2992, 2009.
- [100] Ine ABAQUS. Abaqus analysis user’s manual, version 6.7. 2007.
- [101] Jacob Lubliner. *Plasticity theory*. Courier Corporation, 2008.

APPENDICES

A Experimental Data

Under the accordance of local Institutional Animal Care and Use Committee (IACUC) approval at the University of Colorado's College of Veterinary Medicine and Biomedical Sciences, the six adult ewes were raised and sacrificed.

DMA was performed at Henry Ford Hospital. Bone samples were tested between the frequencies of 0 to 20 Hz in increments of 0.2 Hz. This study only used data from 1, 3, 9, and 15 Hz to allow for comparsion to Mendoza's data. The University of Michigan, Ann Arbor utilized atomic force microscopy (AFM) to examine morphology of collagen bundle structure and recorded periodicities utilized in the Complex Model's randomized generation.

AFM data were provided to this study for the purposes of generating a complex computational model. DMA Experimental data were collected with the intent of validating the finite element model.

Table 16: Experimental Mechanical Testing. Data were collected from DMA testing across six test sheep. Testing was performed at Henry Ford Hospital.

Sheep	Specimen	Treatment	Sector	Side	Tangent Delta			
					1 Hz	3 Hz	9 Hz	15 Hz
6	C0615	Control	5	Cd	0.09144	0.077092	0.05042	0.025226
6	C0613	Control	6	Cd	0.08666	0.073777	0.049291	0.028045
6	C0608	Control	2	Cr	0.07762	0.066401	0.045583	0.026755
6	C0603	Control	1	Cr	0.077544	0.068259	0.045227	0.024657
11	C1114	Control	6	Cd	0.085255	0.071874	0.04505	0.018304
11	C1111	Control	5	Cd	0.081187	0.062383	0.033776	0.009397
11	C1120	Control	2	Cr	0.09284	0.07164	0.035124	0.007162
11	C1108	Control	1	Cr	0.07276	0.063211	0.040497	0.021065
22	C2222	Control	5	Cd	0.098628	0.085794	0.060804	0.036278
22	C2219	Control	6	Cd	0.068954	0.060823	0.04108	0.022938
22	C2215	Control	2	Cr	0.068225	0.055921	0.031804	0.012429
22	C2207	Control	1	Cr	0.07035	0.060794	0.03861	0.020517

Table 17: Tangent Delta Means and Standard Deviations for Experimental Data. Testing was performed at Henry Ford Hospital and provided to this current study.

Model	Sector	MEAN/SD	Tangent Delta			
			1 Hz	3 Hz	9 Hz	15 Hz
Experimental Data	Cr	MEAN	0.076557	0.064371	0.039474	0.018764
		ST. DEV	0.008827	0.005615	0.005477	0.007506
	Cd	MEAN	0.085354	0.071957	0.046737	0.023365
		ST. DEV	0.010013	0.009346	0.009174	0.009089

Table 18: Experimental D-Spacings. Atomic force microscopy was utilized to record collagen periodicity across 6 test sheep. AFM work was performed by the University of Michigan, Ann Arbor.

Sample Classification				AFM D-spacing (nm)		
Sheep	Specimen	Sector	Side	Mean	Median	SD
22	C2222	5	Cd	68.00058974	67.995	1.377052261
22	C2219	6		67.05708571	66.683	1.961977978
11	C1114	6		63.14316092	62.48	3.363895822
11	C1111	5		66.66137143	66.296	1.234662662
6	C0615	5		67.12753333	66.666	2.760867286
6	C0613	6		66.13146032	66.235	2.770452019
22	C2215	2	Cr	66.84862857	66.44	2.396897606
22	C2207	1		67.44332432	67.875	1.156853382
11	C1120	2		67.77131429	67.737	2.120829981
11	C1108	1		68.79706452	68.941	0.979417864
6	C0608	2		69.47862069	69.2375	1.351576643
6	C0603	1		70.18066667	70.557	0.81223489

B User Subroutine (UMAT)

The following UMAT was originally created by Dr. Frank Richter at the Technical University of Berlin [89]. The code was then edited by Mendoza for his study and also used for this analysis [52].

```

C
C SDVINI SUBROUTINE TO INITIALIZE AND KEEP TRACK OF STRESS INCREMENTS
C FROM THE PREVIOUS CALCULATION
C
C     SUBROUTINE SDVINI(STATEV,COORDS,NSTATV,NCRDS,NOEL,NPT,
1  LAYER,KSPT)
C
C     INCLUDE 'ABA_PARAM.INC'
C
C     DIMENSION STATEV(NSTATV),COORDS(NCRDS)
C
C STATEV 1, 2, AND 3 CORRESPOND TO DSTRES 1, 2, AND 3 IN THAT ORDER
C WRITE STATEMENTS WERE UTILIZED FOR DEBUGGING PURPOSES
C
C     STATEV(1) = 0.0
C     STATEV(2) = 0.0
C     STATEV(3) = 0.0
C
C     RETURN
C     END
C
C 3D FORMULATION OF THE STANDARD LINEAR SOLID (KELVIN BODY)
C
C
C     SUBROUTINE UMAT(STRESS,STATEV,DDSDDE,SSE,SPD,SCD,
1  RPL,DDSDDT,DRPLDE,DRPLDT,
2  STRAN,DSTRAN,TIME,DTIME,TEMP,DTEMP,PREDEF,DPRED,CMNAME,
3  NDI,NSHR,NTENS,NSTATV,PROPS,NPROPS,COORDS,DROT,PNEWDT,
4  CELENT,DFGRD0,DFGRD1,NOEL,NPT,LAYER,KSPT,KSTEP,KINC)
C
C     INCLUDE 'ABA_PARAM.INC'
C
C     CHARACTER*8 CMNAME
C
C     DIMENSION STRESS(NTENS),STATEV(NSTATV),
1  DDSDE(NTENS,NTENS),
2  DDSDDT(NTENS),DRPLDE(NTENS),

```



```

3 STRAN(NTENS),DSTRAN(NTENS),TIME(2),PREDEF(1),DPRED(1),
4 PROPS(NPROPS),COORDS(3),DROT(3,3),DFGRD0(3,3),DFGRD1(3,3)

  DIMENSION DSTRES(6),D(3,3)

  REAL K_E, G_E,
1 K_Ke, G_Ke,
2 Eta_B, Eta_S

```

C

C ADDITIONAL CONSTANTS ARE LISTED AS FOLLOWS:

C K_E IS THE BULK MODULUS OF THE SPRING

C G_E IS THE SHEAR MODULUS OF THE SPRING

C K_Ke IS THE BULK MODULUS OF THE SPRING IN THE KELVIN BODY

C G_Ke IS THE SHEAR MODULUS OF THE SPRING IN THE KELVIN BODY

C Eta_B IS THE BULK VISCOCITY OF THE DASHPOT IN THE KELVIN BODY

C Eta_S IS THE SHEAR VISCOSITY OF THE DASHPOT IN THE KELVIN BODY

C

C CALCULATE MATERIAL PROPERTIES BASED ON USER DEFINED CONSTANTS

C

```

K_E = PROPS(1)/(3*(1 - 2*PROPS(4)))

```

```

G_E = PROPS(1)/(2*(1 + PROPS(4)))

```

```

K_Ke = PROPS(2)/(3*(1 - 2*PROPS(5)))

```

```

G_Ke = PROPS(2)/(2*(1 + PROPS(5)))

```

```

Eta_B = PROPS(3)/(3*(1 - 2*PROPS(6)))

```

```

Eta_S = PROPS(3)/(2*(1 + PROPS(6)))

```

C

```

C  USER  DEFINED  CONSTANTS  REFER  TO:

C  PROPS(1):  THE  ELASTIC  MODULUS  OF  THE  SPRING

C  PROPS(2):  THE  ELASTIC  MODULUS  OF  THE  SPRING  IN  THE  KELVIN  BODY

C  PROPS(3):  THE  VISCOCITY  OF  THE  DASHPOT  IN  THE  KELVIN  BODY

C  PROPS(4):  POISSONS  RATIO  OF  THE  SPRING

C  PROPS(5):  POISSONS  RATIO  OF  THE  SPRING  IN  THE  KELVIN  BODY

C  PROPS(6):  POISSONS  RATIO  OF  THE  DASHPOT  IN  THE  KELVIN  BODY

C

C  EVALUATE  NEW  STRESS  TENSOR

C

      EV = 0

      DEV = 0

      SV = 0

      DSV = 0

C
C      WRITE(*,*) 'KINC = ',KINC
C      WRITE(*,*) 'KSTEP = ',KSTEP
C

      DO K1=1,NDI

          EV = EV + STRAN(K1)

          DEV = DEV + DSTRAN(K1)

          SV = SV + STRESS(K1)

          DSV = DSV + STATEV(K1)

      END DO

C
C      WRITE(*,*) 'EV = ',EV
C      WRITE(*,*) 'DEV = ',DEV
C      WRITE(*,*) 'SV = ',SV

```

```

C      WRITE(*,*) 'DSV = ',DSV
C
C      EVALUATE DIRECT STRESS COMPONENTS
C
      TERM1A = (6*DTIME*K_E*G_E)/(3*DTIME*K_E*G_E + 2*DTIME*K_E*G_Ke
1          + 4*K_E*Eta_S + DTIME*G_E*K_Ke + 2*G_E*Eta_B)
      TERM2 = (G_Ke + ((2*Eta_S)/DTIME))
      TERM3 = (3*K_Ke - 2*G_Ke)/6 + (3*Eta_B - 2*Eta_S)/(3*DTIME)
      TERM4 = (2*G_Ke)
      TERM5 = (3*K_Ke - 2*G_Ke)/3
      TERM6 = (1+(G_Ke/G_E))
      TERM7 = (K_Ke/K_E - G_Ke/G_E)/3
      TERM8 = (K_Ke/K_E - G_Ke/G_E)/6
1          + (Eta_B/K_E - Eta_S/G_E)/(3*DTIME)
C
      DO K1=1,NDI
          DSTRES(K1) = TERM1A*(TERM2*DSTRAN(K1) + TERM3*DEV
1          + TERM4*STRAN(K1) + TERM5*EV - TERM6*STRESS(K1) - TERM7*SV
2          - TERM8*(DSV - STATEV(K1)))
          STRESS(K1) = STRESS(K1) + DSTRES(K1)
      END DO
C
C      SAVE CURRENT STRESS INCREMENTS FOR THE NEXT STRESS CALCULATION
C
      DO K1 = 1,NDI
          STATEV(K1) = DSTRES(K1)
      END DO
C
C      WRITE(*,*) 'STATEV(1) = ',STATEV(1)
C      WRITE(*,*) 'STATEV(2) = ',STATEV(2)
C      WRITE(*,*) 'STATEV(3) = ',STATEV(3)

```

```
C
C  EVALUATE SHEAR STRESS COMPONENTS
C
```

```
    TERM1B = ((2*DTIME*G_E)/(DTIME*G_E + DTIME*G_Ke + 2*Eta_S))
```

```
    TERM2B = TERM2/2
```

```
    TERM3B = TERM4/2
```

```
    TERM4B = TERM6
```

```
    I1 = NDI
```

```
C
```

```
    DO K1=1,NSHR
```

```
        I1 = I1+1
```

```
        DSTRES(I1) = TERM1B*(TERM2B*DSTRAN(I1) + TERM3B*STRAN(I1)
```

```
1        - TERM4B*STRESS(I1))
```

```
        STRESS(I1) = STRESS(I1)+DSTRES(I1)
```

```
    END DO
```

```
C
```

```
C  CREATE NEW JACOBIAN
```

```
C
```

```
    TERM2C = TERM1A*(6*DTIME*G_Ke + 12*Eta_S + 3*DTIME*K_Ke
1    - 2*DTIME*G_Ke + 6*Eta_B - 4*Eta_s)/(6*DTIME)
```

```
    TERM3C = TERM1A*(3*DTIME*K_Ke - 2*DTIME*G_Ke + 6*Eta_B
1    - 4*Eta_S)/(6*DTIME)
```

```
C
```

```
    DO K1=1,NTENS
```

```
        DO K2=1,NTENS
```

```
            DDSDDDE(K2,K1) = 0
```

```
C            WRITE(*,*) 'K1 = ',K1
```

```

C          WRITE(*,*) 'K2 = ',K2
      END DO
END DO

C
DO K1=1,NDI
      DDSDDDE(K1,K1) = TERM2C
END DO

C
DO K1=2,NDI
      N2 = K1-1
      DO K2=1,N2
            DDSDDDE(K2,K1) = TERM3C
            DDSDDDE(K1,K2) = TERM3C
C          WRITE(*,*) 'K1 = ',K1
C          WRITE(*,*) 'K2 = ',K2
      END DO
END DO

C
TERM4C = TERM1B*(DTIME*G_Ke + 2*Eta_S)/(2*DTIME)
I1 = NDI

C
DO K1=1,NSHR
      I1 = I1+1
      DDSDDDE(I1,I1) = TERM4C
C      WRITE(*,*) 'I1 = ',I1
END DO

```

```

C
C      WRITE(*,*) 'NTENS = ',NTENS
C      WRITE(*,*) 'NDI = ',NDI
C      WRITE(*,*) 'NSHR = ',NSHR
C      WRITE(*,*) 'G_E = ',G_E
C      WRITE(*,*) 'K_E = ',K_E
C      WRITE(*,*) 'G_Ke = ',G_Ke
C      WRITE(*,*) 'K_Ke = ',K_Ke
C      WRITE(*,*) 'Eta_S = ',Eta_S
C      WRITE(*,*) 'Eta_B = ',Eta_B
C      WRITE(*,*) 'DTIME = ',DTIME
C      WRITE(*,*) 'TERM1A = ',TERM1A
C      WRITE(*,*) 'TERM2 = ',TERM2
C      WRITE(*,*) 'TERM3 = ',TERM3
C      WRITE(*,*) 'TERM4 = ',TERM4
C      WRITE(*,*) 'TERM5 = ',TERM5
C      WRITE(*,*) 'TERM6 = ',TERM6
C      WRITE(*,*) 'TERM7 = ',TERM7
C      WRITE(*,*) 'TERM8 = ',TERM8
C      WRITE(*,*) 'TERM1B = ',TERM1B
C      WRITE(*,*) 'TERM2B = ',TERM2B
C      WRITE(*,*) 'TERM3B = ',TERM3B
C      WRITE(*,*) 'TERM4B = ',TERM4B
C      WRITE(*,*) 'TERM2C = ',TERM2C
C      WRITE(*,*) 'TERM3C = ',TERM3C
C      WRITE(*,*) 'TERM4C = ',TERM4C
C

```

```

RETURN

```

```

END

```

C Viscoelastic Equations

The Kelvin-Voigt version of the Standard Linear Solid was selected to model bone viscoelasticity due to its ability in expressing a creep and stress relaxation response. This version (figure 37) is composed of two springs elements and a single dashpot element.

Figure 37: The Kelvin-Voigt version of the Standard Linear Solid exhibits both a creep and stress relaxation response [90].

One method used to analyze the stress relaxation and creep behaviors of rheological models will be demonstrated below. The governing equation of this rheological model is:

$$\eta E_1 \dot{\epsilon} + E_1 E_2 \epsilon = \eta \dot{\sigma} + (E_1 + E_2) \sigma \quad (20)$$

Creep Response:

A creep response is described by continually deforming after initial deformation when applied under a constant stress (σ_o & $\dot{\sigma} = 0$). When these conditions are applied, the integration of equation 20 is:

$$\epsilon = \frac{\sigma_o}{E_1} + \frac{\sigma_o}{E_2} \left[1 - \exp\left(\frac{-t}{\tau}\right) \right]$$

where τ is the retardation time of the viscoelastic material during creep and expressed as:

$$\tau = \frac{\eta}{E_2}$$

which creates a Standard Linear Solid model for creep compliance:

$$\varepsilon = \frac{1}{E_1} + \frac{1}{E_2} * [1 - \exp(\frac{-t}{\tau})] \sigma_0$$

Stress Relaxation:

A stress relaxation response from a viscoelastic material can be observed when the model is under constant deformation (ε_0 & $\dot{\varepsilon} = 0$). The model will undergo the initial stress. The stress will start decrease over time as the model is being held at a constant deformation. When these conditions are applied, the integration of equation 20 is:

$$\sigma = \frac{E_1 \varepsilon_0}{E_1 + E_2} [E_2 + E_1 * \exp(\frac{-t}{\tau})]$$

where τ is the retardation time of the viscoelastic material during stress relaxation and expressed as:

$$\tau = \frac{\eta}{E_1 + E_2}$$

which creates a Standard Linear Solid model for stress relaxation:

$$\sigma = \frac{E_1}{E_1 + E_2} * [E_2 + E_1 * \exp(\frac{-t}{\tau})] \varepsilon_0$$

D Loading Condition: Stress Estimation

A three point bending Dynamic mechanical analysis (DMA) was done to all experimental bone samples. The Dimensions of bone samples were 15 mm x 1.75 mm x 1.75 mm. Bone samples were held by two supporting beams 15 mm apart. Loading conditions for each analysis include a static load of 550 mN and dynamic load of 500 mN at the midpoint between the two support structures. Using this information, the stress applied on the computation models can be estimated. The maximum bending stress, σ_{Max} , can be calculated the following equation:

$$\sigma_{Max} = \frac{M_{Max}c}{I}$$

where M_{Max} is the maximum moment about the neutral axis, c is the perpendicular distance to the neutral axis, and I is the second moment of area about the neutral axis.

M_{Max} can be determined using the equation below:

$$M_{Max} = F_{Total} * d$$

where

$$F_{Total} = F_{Static} + F_{Dynamic}$$

F_{Static} is the force from the static loading condition. The static force can be represented by the reactionary force from a single support structure. Making an imaginary cut at the middle of the beam can simplify the stress calculations.

$F_{Dynamic}$ is the force from the dynamic loading condition can be written as $F(x, t)$. x can be assumed to be zero since the distance between the where the force is applied and point of interest is zero. Additionally, t is the time when the dynamic force is at its maximum [101]. The force the dynamic load can be simplified to:

$$F_{Dynamic} = F(0, t_{Max}) = \frac{500 * 10^{-3} N}{4}$$

therefore

$$F_{Total} = F_{Static} + F_{Dynamic} = \frac{550 * 10^{-3} N}{2} + \frac{500 * 10^{-3} N}{4} = 400 * 10^{-3} N$$

Then M_{Max} can be solved with the length of the moment arm, d , being 7.5 mm.

$$M_{Max} = F_{Total} * d = 400 * 10^{-3} N * (7.5 * 10^{-3} m) = 3 * 10^{-3} Nm$$

The second moment of area about the neutral axis, I , can be calculated with the equation:

$$I = \frac{1}{12} * b * h^3$$

where b and h are the lengths of the base and height from the cross section of the beam (1.75 mm x 1.75 mm):

$$I = \frac{1}{12} * b * h^3 = \frac{1}{12} * (1.75 * 10^{-3} m) * (1.75 * 10^{-3} m)^3 = 7.82 * 10^{-13} m^4$$

With the perpendicular distance to the neutral axis known, the maximum stress can be calculated:

$$\sigma_{Max} = \frac{M_{Max} c}{I} = \frac{(3 * 10^{-3} Nm) (\frac{1.75 * 10^{-3} m}{2})}{7.816 * 10^{-13} m^4} = 3.36 MPa$$

E Input File and Modifications

An input file is required for an Abaqus job to be run. The input file contains the details of each model from loading conditions, elements, material properties, etc.

This study developed the replicates of all the models by modifying input files created through Abaqus. Editing models through the input files provides for great time saves since specific parameters of the model can be edited through a text document instead of going through the Abaqus GUI. There were two important parameters to change for each model replicate. The first is to create steps for this analysis to be dynamic.

Because of the multiple actions it takes, creating steps through the GUI becomes time consuming and tasking if hundreds of steps are needed. In order to create an accurate curve-fitting plot, a large amount of data points are needed. It was then determined that at least twenty points per cycle would provide enough data to yield an accurate measurement of tangent delta. This was largely based on Mendoza's study. Since the simulation caused the model to be loaded for twenty cycles, a total of 400 points will be required. Thus, 400 steps need to be created for each model.

Since boundary conditions, loads, and controls are only required for the first step, the following steps are identical to each other. Step-2 can then be copied, pasted, and renamed. This continues until 400 steps are created. Once this is done for one model, it can simple be copied onto other models of the same loading frequency.

The next advantage is to be able to alter the frequency of the applied load. This task only requires the modification for one line of the input file seen below and using the "find and replace" feature of the text editor.

The amplitude can easily be adjusted knowing that the angular frequency ($\omega=2\pi*f$) is represented in the highlighted yellow in the code. Where ω represents the angular frequency in radians per seconds and f represents the test frequency. For this example, the number highlighted in yellow is the angular frequency at 1 Hz. Changing the loading condition is simple by replacing the angular frequency that corresponds to the wanted loading frequency.

Since the test frequency is changed, the time it takes for the model to finish needs to be adjusted accordingly too. The number's seen in the above code highlighted in magenta

represent the step time, initial increment, smallest increment, and largest increment respectively. The numbers displayed are at 1 Hz and represent the time it takes to take to complete one cycle (0.05 seconds*20 steps). Thus, these numbers are divided by the desired frequency.

The input file also contains the six parameters for the viscoelastic material property. Seen highlighted in blue, each number represents value of the parameters in the Kelvin-Voigt version of the Standard Linear Solid. A table of the values and their means can be seen in table 19. Values are scaled accordingly due to Abaqus' inability to work with dimensions in the Nano scale. These values work in conjunction with the Richter UMAT file in appendix B seen as "Prop(1), Prop(2), etc".

Table 19: Table of Material Properties in Rheological Elements. The values highlighted in blue represents different parameters for the Standard Linear Solid. Below are the names assigned in the code, their description, and value.

Name	Description	Rheological	Input Value	Actual Value
Prop(1)	Elastic Modulus of Lone Spring Element	E_1	0.003	3 GPa
Prop(2)	Elastic Modulus of Kelvin-Voigt Spring Element	E_2	0.006	6 GPa
Prop(3)	Dashpot Viscosity	η_1	0.00125	1.25 GPa*s
Prop(4)	Poisson's Ratio of Lone Spring Element	ν_{E1}	0.2	0.2
Prop(5)	Poisson's Ratio of Kelvin-Voigt Spring Element	ν_{E2}	0.2	0.2
Prop(6)	Poisson's Ratio of Dashpot	$\nu_{\eta1}$	0.2	0.2

```

*Surface, type=ELEMENT, name=_PickedSurf12, internal
_PickedSurf12_S4, S4
_PickedSurf12_S2, S2
_PickedSurf12_S1, S1
*End Assembly
*Amplitude, name=SINUSOIDAL, time=TOTAL TIME, definition=PERIODIC
1, 6.28319, 0., 0.6875
0., 0.3125
**
** MATERIALS
**
*Material, name=COLLAGEN
*Depvar
3,
*User Material, constants=6
0.003, 0.006, 0.00125, 0.2, 0.2, 0.2
*Material, name=HYDROXYAPATITE
*Elastic
0.1, 0.28
** -----
**
** STEP: APPLY LOAD
**
*Step, name="APPLY LOAD", inc=100000
*Static
0.05, 0.05, 1e-6, 0.05
**
** BOUNDARY CONDITIONS
**
** Name: XSYM Type: Symmetry/Antisymmetry/Encastre
*Boundary
_PickedSet17, XSYMM
** Name: YSYM Type: Symmetry/Antisymmetry/Encastre
*Boundary
_PickedSet13, YSYMM
**
** LOADS
**
** Name: PRESSURE Type: Pressure
*Dload, amplitude=SINUSOIDAL
_PickedSurf12, P, -3.36e-06
**
** CONTROLS
**
*Controls, reset
*Controls, parameters=field, field=displacement
,,,,, 1e-09,,
*Controls, parameters=field, field=hydrostatic fluid pressure
,,,,, 1e-09,,
*Controls, parameters=field, field=rotation
,,,,, 1e-09,,
*Controls, parameters=field, field=electrical potential
,,,,, 1e-09,,
**
** OUTPUT REQUESTS
**
*Restart, write, frequency=0
**
** FIELD OUTPUT: F-Output-1
**
*Output, field
*Node Output
U,
**
** HISTORY OUTPUT: H-Output-1

```

```
**
*Output, history, variable=PRESELECT
*End Step
** -----
**
** STEP: Step-2
**
*Step, name=Step-2, inc=100000
*Static
0.05, 0.05, 1e-6, 0.05
**
** OUTPUT REQUESTS
**
*Restart, write, frequency=0
**
** FIELD OUTPUT: F-Output-1
**
*Output, field
*Node Output
U,
**
** HISTORY OUTPUT: H-Output-1
**
*Output, history, variable=PRESELECT
*End Step
** -----
```

F Composite Model Results

Data was collected from all Composite Models. Abaqus provided an output database file (ODB) for each job run. The ODB file is read by another Python script that extracts deflection data from specific nodes. MATLAB scripts convert this deflection data into tangent delta information. Because there is some randomization in the scripts, the code is run ten times and averaged for each test frequency for each model. The data presented below is by Control, Cranial and Control, Caudal Composite Models at 1, 3, 9, and 15 Hz test frequencies.

The next set of data is from testing the 4 GPa*s dashpot at a Control, Cranial Composite Model. The model was also run at all test frequencies and its results were compared to the same Composite Model with a 1.25 GPa*s dashpot.

Table 20: Results from the Control, Cranial Composite Model with a 1.25 GPa*s Dashpot. Four versions of the Control, Cranial model were created and tested ten times before reporting an averaged Tangent delta.

1.25 GPa*s Dasphot (eta 1)						
Model	Version	Run	Tangent Delta 1 Hz	Tangent Delta 3 Hz	Tangent Delta 9 Hz	Tangent Delta 15 Hz
Control, Cran	1	1	0.054592	0.030171	0.010915	0.006574
		2	0.065483	0.030174	0.043486	0.0065322
		3	0.054619	0.11831	0.01089	0.006533
		4	0.054538	0.030225	0.1084	0.0065987
		5	0.054555	0.030156	0.010871	0.0065356
		6	0.058422	0.030237	0.010926	0.00654
		7	0.08476	0.030193	0.010875	0.0065124
		8	0.063109	0.030184	0.038644	0.0065405
		9	0.054564	0.030224	0.056288	0.0065361
		10	0.054551	0.043347	0.010913	0.006559
		Average	0.0599193	0.0403221	0.0312208	0.00654615
	2	1	0.065588	0.030384	0.010985	0.0065915
		2	0.054963	0.030364	0.010984	0.0065738
		3	0.054966	0.11832	0.010905	0.0066188
		4	0.054954	0.030338	0.010987	0.0065775
		5	0.20775	0.030372	0.010989	0.0066314
		6	0.05803	0.030349	0.011019	0.0066715
		7	0.08476	0.030375	0.010932	0.0066323
		8	0.054894	0.030371	0.010922	0.0066318
		9	0.054912	0.030347	0.010964	0.0066058
		10	0.063138	0.030399	0.010972	0.0066166
		Average	0.0753955	0.0391619	0.0109659	0.0066151
	3	1	0.054696	0.030286	0.04342	0.0066482
		2	0.054969	0.030181	0.010945	0.0065575
		3	0.05467	0.11832	0.010896	0.006595
		4	0.054662	0.030298	0.010903	0.006589
		5	0.054624	0.030188	0.010919	0.0065449
		6	0.054661	0.030207	0.038674	0.0065423
		7	0.054634	0.030193	0.056288	0.0066049
		8	0.054625	0.030242	0.010923	0.0065571
		9	0.054688	0.030248	0.010978	0.006601
		10	0.054636	0.030193	0.010912	0.0065322
		Average	0.0546865	0.0390356	0.0214858	0.00657721
	4	1	0.054846	0.030308	0.04342	0.0066304
		2	0.054885	0.030348	0.010908	0.11832
		3	0.05485	0.11832	0.010905	0.0066166
		4	0.054844	0.030288	0.010886	0.0065644
		5	0.054781	0.030363	0.010925	0.0066131
		6	0.054808	0.030309	0.038674	0.0066055
		7	0.065593	0.03029	0.056288	0.0065349
		8	0.054834	0.030285	0.010905	0.0065705
		9	0.054776	0.030315	0.010913	0.0065808
		10	0.054855	0.030292	0.041858	0.043354
		Average	0.0559072	0.0391118	0.0245682	0.02143902

Table 21: Results from the Control, Caudal Composite Model with a 1.25 GPa*s Dashpot. Four versions of the Control, Caudal model were created and tested ten times before reporting an averaged Tangent delta.

1.25 GPa*s Dasphot (eta 1)						
Model	Version	Run	Tangent Delta 1 Hz	Tangent Delta 3 Hz	Tangent Delta 9 Hz	Tangent Delta 15 Hz
Control, Caud	1	1	0.056128	0.031	0.011172	0.0067195
		2	0.05612	0.030926	0.011139	0.0067842
		3	0.056063	0.030979	0.011205	0.0067213
		4	0.056104	0.030923	0.01114	0.0067586
		5	0.056117	0.038766	0.011151	0.0066674
		6	0.056079	0.05624	0.19713	0.0067594
		7	0.056065	0.030949	0.011215	0.0066994
		8	0.056079	0.030995	0.011165	0.0067656
		9	0.056124	0.041872	0.011162	0.0067634
		10	0.065431	0.031039	0.011157	0.0066946
		Average	0.057031	0.0353689	0.0297636	0.00673334
	2	1	0.056909	0.031334	0.011327	0.0068584
		2	0.05683	0.031326	0.011272	0.0068606
		3	0.056895	0.031412	0.011324	0.0068518
		4	0.056876	0.031324	0.011341	0.0068486
		5	0.05681	0.031391	0.038764	0.0068409
		6	0.15803	0.031383	0.05624	0.0067704
		7	0.056919	0.0314	0.011273	0.0068017
		8	0.056828	0.031383	0.011299	0.0067811
		9	0.056865	0.03138	0.011941	0.0067913
		10	0.056812	0.0314	0.011315	0.006827
		Average	0.0669774	0.0313733	0.0186096	0.00682318
	3	1	0.059123	0.032596	0.011692	0.0070431
		2	0.059123	0.032539	0.011701	0.0070306
		3	0.059121	0.032507	0.011796	0.0070473
		4	0.059154	0.064965	0.011778	0.0070726
		5	0.059123	0.032506	0.011767	0.0070247
		6	0.059151	0.032574	0.011702	0.0070305
		7	0.05913	0.032581	0.01171	0.0070704
		8	0.059149	0.032533	0.011766	0.0070844
		9	0.059161	0.032567	0.011715	0.0071046
		10	0.059122	0.032534	0.011744	0.0070342
		Average	0.0591357	0.0357902	0.0117371	0.00705424
	4	1	0.056328	0.031048	0.011247	0.0067313
		2	0.090764	0.028204	0.011179	0.00677
		3	0.056286	0.031074	0.011238	0.0068052
		4	0.056272	0.031084	0.011223	0.0067739
		5	0.056338	0.031119	0.011182	0.0067118
		6	0.134	0.031121	0.011229	0.0067929
		7	0.056268	0.03108	0.011215	0.0067744
		8	0.056264	0.031119	0.01123	0.0067698
		9	0.056315	0.031124	0.011181	0.0067783
		10	0.072846	0.03106	0.011211	0.0067338
		Average	0.0691681	0.0308033	0.0112135	0.00676414

Table 22: Results from the Control, Cranial Composite Model with a 4.00 GPa*s Dashpot. A single version of Control, Cranial was run across all test frequencies ten times to be used as a comparison between the effects of a large viscosity dashpot.

4.00 GPa*s Dashpot (eta 1)						
Model	Version	Run	Tangent Delta 1 Hz	Tangent Delta 3 Hz	Tangent Delta 9 Hz	Tangent Delta 15 Hz
Control, Cran	4	1	0.028617	0.010299	0.0033548	0.0017822
		2	0.043376	0.01028	0.0032828	0.0018583
		3	0.028662	0.010277	0.0033282	0.0018674
		4	0.028659	0.01083	0.0033545	0.0018387
		5	0.028683	0.010212	0.0033191	0.0018738
		6	0.02866	0.010263	0.0033327	0.0018054
		7	0.038764	0.010279	0.0032937	0.0018174
		8	0.05625	0.0103	0.0033741	0.0017977
		9	0.028669	0.010264	0.0033558	0.001847
		10	0.028624	0.010272	0.0033061	0.0018049
		Average	0.0338964	0.0103276	0.00333018	0.00182928

Figure 38: Experimental data vs. Control, Cranial Composite Model data. A linear line was fitted onto the plot. Plotted Composite Model data are averages for each frequency. $R^2=0.94211$.

Figure 39: Experimental data vs Mendoza data for Control, Cranial. A linear line was fitted onto the plot. $R^2=0.87399$.

Figure 40: Experimental data vs. Control, Caudal Composite Model data. A linear line was fitted onto the plot. Plotted Composite Model data are averages for each frequency $R^2=0.88327$.

Figure 41: Experimental data vs Mendoza data for Control, Caudal. A linear line was fitted onto the plot. $R^2=0.86812$.

G Statistical Analysis Results

Statistical analysis was performed with the assistance of Dr. Jonathan Walker of the California Polytechnic State University, San Luis Obispo Statistics department. Due to the somewhat complex analysis, Statistical Analysis Software (SAS) was used. There were two separate two-way ANOVA analysis performed. The first being for Control, Cranial data and the second being for Control, Caudal data. Highlighted portions were found to be significantly different.

*The SAS System**The Mixed Procedure*

Model Information	
Data Set	HA.CRANIAL
Dependent Variable	TD
Covariance Structure	Spatial Power
Subject Effect	Bone(Model)
Estimation Method	REML
Residual Variance Method	Profile
Fixed Effects SE Method	Model-Based
Degrees of Freedom Method	Between-Within

Class Level Information		
Class	Levels	Values
Model	2	Experimental Model
Freq	4	1 3 9 15
Bone	8	5 6 7 8 9 10 11 12

Dimensions	
Covariance Parameters	2
Columns in X	15
Columns in Z	0
Subjects	10
Max Obs Per Subject	4

Number of Observations	
Number of Observations Read	40
Number of Observations Used	40
Number of Observations Not Used	0

Iteration History			
Iteration	Evaluations	-2 Res Log Like	Criterion
0	1	-212.74363015	
1	2	-217.53532836	0.00124000
2	1	-217.72976154	0.00007420

The SAS System

Iteration History			
Iteration	Evaluations	-2 Res Log Like	Criterion
3	1	-217.74048622	0.00000039
4	1	-217.74053992	0.00000000

Convergence criteria met.

Estimated R Matrix for Bone(Model) 7 Experimental				
Row	Col1	Col2	Col3	Col4
1	0.000053	0.000036	0.000011	3.441E-6
2	0.000036	0.000053	0.000016	5.091E-6
3	0.000011	0.000016	0.000053	0.000016
4	3.441E-6	5.091E-6	0.000016	0.000053

Estimated R Correlation Matrix for Bone(Model) 7 Experimental				
Row	Col1	Col2	Col3	Col4
1	1.0000	0.6758	0.2086	0.06438
2	0.6758	1.0000	0.3087	0.09527
3	0.2086	0.3087	1.0000	0.3087
4	0.06438	0.09527	0.3087	1.0000

Covariance Parameter Estimates		
Cov Parm	Subject	Estimate
SP(POW)	Bone(Model)	0.8221
Residual		0.000053

Fit Statistics	
-2 Res Log Likelihood	-217.7
AIC (smaller is better)	-213.7
AICC (smaller is better)	-213.3
BIC (smaller is better)	-213.1

The SAS System

Null Model Likelihood Ratio Test		
DF	Chi-Square	Pr > ChiSq
1	5.00	0.0254

Type 3 Tests of Fixed Effects				
Effect	Num DF	Den DF	F Value	Pr > F
Model	1	8	26.66	0.0009
Freq	3	24	101.67	<.0001
Model*Freq	3	24	3.59	0.0284

Least Squares Means										
Effect	Model	Freq	Estimate	Standard Error	DF	t Value	Pr > t	Alpha	Lower	Upper
Model	Experimental		0.04979	0.002019	8	24.66	<.0001	0.0167	0.04371	0.05588
Model	Model		0.03331	0.002473	8	13.47	<.0001	0.0167	0.02586	0.04076
Freq		1	0.06902	0.002359	24	29.25	<.0001	0.0167	0.06295	0.07509
Freq		3	0.05189	0.002359	24	21.99	<.0001	0.0167	0.04582	0.05796
Freq		9	0.03077	0.002359	24	13.04	<.0001	0.0167	0.02470	0.03684
Freq		15	0.01453	0.002359	24	6.16	<.0001	0.0167	0.008459	0.02060
Model*Freq	Experimental	1	0.07656	0.002984	24	25.65	<.0001	0.0167	0.06888	0.08423
Model*Freq	Experimental	3	0.06437	0.002984	24	21.57	<.0001	0.0167	0.05669	0.07205
Model*Freq	Experimental	9	0.03947	0.002984	24	13.23	<.0001	0.0167	0.03180	0.04715
Model*Freq	Experimental	15	0.01876	0.002984	24	6.29	<.0001	0.0167	0.01109	0.02644
Model*Freq	Model	1	0.06148	0.003655	24	16.82	<.0001	0.0167	0.05207	0.07088
Model*Freq	Model	3	0.03941	0.003655	24	10.78	<.0001	0.0167	0.03000	0.04881
Model*Freq	Model	9	0.02206	0.003655	24	6.04	<.0001	0.0167	0.01266	0.03146
Model*Freq	Model	15	0.01029	0.003655	24	2.82	0.0096	0.0167	0.000891	0.01970

Differences of Least Squares Means											
Effect	Model	Freq	Model	Freq	Estimate	Standard Error	DF	t Value	Pr > t	Adjustment	Adj P
Model	Experimental		Model		0.01648	0.003192	8	5.16	0.0009	Tukey-Kramer	0.0009
Freq		1		3	0.01713	0.001900	24	9.02	<.0001	Tukey-Kramer	<.0001
Freq		1		9	0.03825	0.002968	24	12.89	<.0001	Tukey-Kramer	<.0001
Freq		1		15	0.05449	0.003227	24	16.88	<.0001	Tukey-Kramer	<.0001

The SAS System

Differences of Least Squares Means											
Effect	Model	Freq	Model	Freq	Estimate	Standard Error	DF	t Value	Pr > t	Adjustment	Adj P
Freq		3		9	0.02112	0.002774	24	7.61	<.0001	Tukey-Kramer	<.0001
Freq		3		15	0.03736	0.003174	24	11.77	<.0001	Tukey-Kramer	<.0001
Freq		9		15	0.01624	0.002774	24	5.85	<.0001	Tukey-Kramer	<.0001
Model*Freq	Experimental	1	Experimental	3	0.01219	0.002403	24	5.07	<.0001	Tukey-Kramer	0.0008
Model*Freq	Experimental	1	Experimental	9	0.03708	0.003755	24	9.88	<.0001	Tukey-Kramer	<.0001
Model*Freq	Experimental	1	Experimental	15	0.05779	0.004082	24	14.16	<.0001	Tukey-Kramer	<.0001
Model*Freq	Experimental	1	Model	1	0.01508	0.004719	24	3.20	0.0039	Tukey-Kramer	0.0641
Model*Freq	Experimental	1	Model	3	0.03715	0.004719	24	7.87	<.0001	Tukey-Kramer	<.0001
Model*Freq	Experimental	1	Model	9	0.05450	0.004719	24	11.55	<.0001	Tukey-Kramer	<.0001
Model*Freq	Experimental	1	Model	15	0.06626	0.004719	24	14.04	<.0001	Tukey-Kramer	<.0001
Model*Freq	Experimental	3	Experimental	9	0.02490	0.003509	24	7.09	<.0001	Tukey-Kramer	<.0001
Model*Freq	Experimental	3	Experimental	15	0.04561	0.004015	24	11.36	<.0001	Tukey-Kramer	<.0001
Model*Freq	Experimental	3	Model	1	0.002894	0.004719	24	0.61	0.5455	Tukey-Kramer	0.9983
Model*Freq	Experimental	3	Model	3	0.02496	0.004719	24	5.29	<.0001	Tukey-Kramer	0.0005
Model*Freq	Experimental	3	Model	9	0.04231	0.004719	24	8.97	<.0001	Tukey-Kramer	<.0001
Model*Freq	Experimental	3	Model	15	0.05408	0.004719	24	11.46	<.0001	Tukey-Kramer	<.0001
Model*Freq	Experimental	9	Experimental	15	0.02071	0.003509	24	5.90	<.0001	Tukey-Kramer	0.0001
Model*Freq	Experimental	9	Model	1	-0.02200	0.004719	24	-4.66	<.0001	Tukey-Kramer	0.0021
Model*Freq	Experimental	9	Model	3	0.000066	0.004719	24	0.01	0.9889	Tukey-Kramer	1.0000
Model*Freq	Experimental	9	Model	9	0.01741	0.004719	24	3.69	0.0011	Tukey-Kramer	0.0215
Model*Freq	Experimental	9	Model	15	0.02918	0.004719	24	6.18	<.0001	Tukey-Kramer	<.0001
Model*Freq	Experimental	15	Model	1	-0.04271	0.004719	24	-9.05	<.0001	Tukey-Kramer	<.0001
Model*Freq	Experimental	15	Model	3	-0.02064	0.004719	24	-4.37	0.0002	Tukey-Kramer	0.0043
Model*Freq	Experimental	15	Model	9	-0.00330	0.004719	24	-0.70	0.4916	Tukey-Kramer	0.9962
Model*Freq	Experimental	15	Model	15	0.008470	0.004719	24	1.79	0.0853	Tukey-Kramer	0.6291
Model*Freq	Model	1	Model	3	0.02207	0.002943	24	7.50	<.0001	Tukey-Kramer	<.0001
Model*Freq	Model	1	Model	9	0.03942	0.004599	24	8.57	<.0001	Tukey-Kramer	<.0001
Model*Freq	Model	1	Model	15	0.05118	0.005000	24	10.24	<.0001	Tukey-Kramer	<.0001
Model*Freq	Model	3	Model	9	0.01735	0.004298	24	4.04	0.0005	Tukey-Kramer	0.0096
Model*Freq	Model	3	Model	15	0.02911	0.004917	24	5.92	<.0001	Tukey-Kramer	<.0001
Model*Freq	Model	9	Model	15	0.01177	0.004298	24	2.74	0.0115	Tukey-Kramer	0.1593

The SAS System

Differences of Least Squares Means									
Effect	Model	Freq	Model	Freq	Alpha	Lower	Upper	Adj Lower	Adj Upper
Model	Experimental		Model		0.0167	0.006858	0.02610	.	.
Freq		1		3	0.0167	0.01224	0.02202	.	.
Freq		1		9	0.0167	0.03061	0.04589	.	.
Freq		1		15	0.0167	0.04618	0.06279	.	.
Freq		3		9	0.0167	0.01398	0.02826	.	.
Freq		3		15	0.0167	0.02919	0.04553	.	.
Freq		9		15	0.0167	0.009100	0.02338	.	.
Model*Freq	Experimental	1	Experimental	3	0.0167	0.006003	0.01837	.	.
Model*Freq	Experimental	1	Experimental	9	0.0167	0.02742	0.04674	.	.
Model*Freq	Experimental	1	Experimental	15	0.0167	0.04729	0.06830	.	.
Model*Freq	Experimental	1	Model	1	0.0167	0.002939	0.02722	.	.
Model*Freq	Experimental	1	Model	3	0.0167	0.02501	0.04929	.	.
Model*Freq	Experimental	1	Model	9	0.0167	0.04236	0.06664	.	.
Model*Freq	Experimental	1	Model	15	0.0167	0.05412	0.07840	.	.
Model*Freq	Experimental	3	Experimental	9	0.0167	0.01587	0.03393	.	.
Model*Freq	Experimental	3	Experimental	15	0.0167	0.03528	0.05594	.	.
Model*Freq	Experimental	3	Model	1	0.0167	-0.00925	0.01503	.	.
Model*Freq	Experimental	3	Model	3	0.0167	0.01282	0.03710	.	.
Model*Freq	Experimental	3	Model	9	0.0167	0.03017	0.05445	.	.
Model*Freq	Experimental	3	Model	15	0.0167	0.04194	0.06622	.	.
Model*Freq	Experimental	9	Experimental	15	0.0167	0.01168	0.02974	.	.
Model*Freq	Experimental	9	Model	1	0.0167	-0.03414	-0.00986	.	.
Model*Freq	Experimental	9	Model	3	0.0167	-0.01207	0.01221	.	.
Model*Freq	Experimental	9	Model	9	0.0167	0.005274	0.02955	.	.
Model*Freq	Experimental	9	Model	15	0.0167	0.01704	0.04132	.	.
Model*Freq	Experimental	15	Model	1	0.0167	-0.05485	-0.03057	.	.
Model*Freq	Experimental	15	Model	3	0.0167	-0.03278	-0.00850	.	.
Model*Freq	Experimental	15	Model	9	0.0167	-0.01544	0.008844	.	.
Model*Freq	Experimental	15	Model	15	0.0167	-0.00367	0.02061	.	.
Model*Freq	Model	1	Model	3	0.0167	0.01450	0.02964	.	.
Model*Freq	Model	1	Model	9	0.0167	0.02759	0.05125	.	.
Model*Freq	Model	1	Model	15	0.0167	0.03832	0.06405	.	.
Model*Freq	Model	3	Model	9	0.0167	0.006290	0.02841	.	.

The SAS System

Differences of Least Squares Means									
Effect	Model	Freq	Model	Freq	Alpha	Lower	Upper	Adj Lower	Adj Upper
Model*Freq	Model	3	Model	15	0.0167	0.01646	0.04176	.	.
Model*Freq	Model	9	Model	15	0.0167	0.000708	0.02282	.	.

*The SAS System**The Mixed Procedure*

Model Information	
Data Set	HA.CAUDAL
Dependent Variable	TD
Covariance Structure	Spatial Power
Subject Effect	Bone(Model)
Estimation Method	REML
Residual Variance Method	Profile
Fixed Effects SE Method	Model-Based
Degrees of Freedom Method	Between-Within

Class Level Information		
Class	Levels	Values
Model	2	Experimental Model
Freq	4	1 3 9 15
Bone	6	1 2 3 4 5 6

Dimensions	
Covariance Parameters	2
Columns in X	15
Columns in Z	0
Subjects	10
Max Obs Per Subject	4

Number of Observations	
Number of Observations Read	40
Number of Observations Used	40
Number of Observations Not Used	0

The SAS System

Iteration History			
Iteration	Evaluations	-2 Res Log Like	Criterion
0	1	-204.37187185	
1	2	-221.57864951	0.00027066
2	1	-221.61873832	0.00000224
3	1	-221.61905407	0.00000000

Convergence criteria met.

Estimated R Matrix for Bone(Model) 1 Experimental				
Row	Col1	Col2	Col3	Col4
1	0.000067	0.000057	0.000035	0.000021
2	0.000057	0.000067	0.000041	0.000025
3	0.000035	0.000041	0.000067	0.000041
4	0.000021	0.000025	0.000041	0.000067

Estimated R Correlation Matrix for Bone(Model) 1 Experimental				
Row	Col1	Col2	Col3	Col4
1	1.0000	0.8476	0.5161	0.3142
2	0.8476	1.0000	0.6089	0.3707
3	0.5161	0.6089	1.0000	0.6089
4	0.3142	0.3707	0.6089	1.0000

Covariance Parameter Estimates		
Cov Parm	Subject	Estimate
SP(POW)	Bone(Model)	0.9206
Residual		0.000067

Fit Statistics	
-2 Res Log Likelihood	-221.6
AIC (smaller is better)	-217.6
AICC (smaller is better)	-217.2
BIC (smaller is better)	-217.0

The SAS System

Null Model Likelihood Ratio Test		
DF	Chi-Square	Pr > ChiSq
1	17.25	<.0001

Type 3 Tests of Fixed Effects				
Effect	Num DF	Den DF	F Value	Pr > F
Model	1	8	38.51	0.0003
Freq	3	24	150.19	<.0001
Model*Freq	3	24	13.38	<.0001

Least Squares Means										
Effect	Model	Freq	Estimate	Standard Error	DF	t Value	Pr > t	Alpha	Lower	Upper
Model	Experimental		0.05685	0.002709	8	20.99	<.0001	0.0167	0.04869	0.06502
Model	Model		0.03027	0.003318	8	9.12	<.0001	0.0167	0.02027	0.04027
Model*Freq	Experimental	1	0.08535	0.003339	24	25.56	<.0001	0.0167	0.07676	0.09394
Model*Freq	Experimental	3	0.07196	0.003339	24	21.55	<.0001	0.0167	0.06337	0.08055
Model*Freq	Experimental	9	0.04674	0.003339	24	14.00	<.0001	0.0167	0.03815	0.05533
Model*Freq	Experimental	15	0.02336	0.003339	24	7.00	<.0001	0.0167	0.01477	0.03196
Model*Freq	Model	1	0.06308	0.004090	24	15.42	<.0001	0.0167	0.05256	0.07360
Model*Freq	Model	3	0.03333	0.004090	24	8.15	<.0001	0.0167	0.02281	0.04386
Model*Freq	Model	9	0.01783	0.004090	24	4.36	0.0002	0.0167	0.007310	0.02835
Model*Freq	Model	15	0.006844	0.004090	24	1.67	0.1072	0.0167	-0.00368	0.01737

Differences of Least Squares Means											
Effect	Model	Freq	Model	Freq	Estimate	Standard Error	DF	t Value	Pr > t	Adjustment	Adj P
Model*Freq	Experimental	1	Experimental	3	0.01340	0.001844	24	7.27	<.0001	Tukey-Kramer	<.0001
Model*Freq	Experimental	1	Experimental	9	0.03862	0.003285	24	11.76	<.0001	Tukey-Kramer	<.0001
Model*Freq	Experimental	1	Experimental	15	0.06199	0.003911	24	15.85	<.0001	Tukey-Kramer	<.0001
Model*Freq	Experimental	1	Model	1	0.02228	0.005280	24	4.22	0.0003	Tukey-Kramer	0.0062
Model*Freq	Experimental	1	Model	3	0.05202	0.005280	24	9.85	<.0001	Tukey-Kramer	<.0001
Model*Freq	Experimental	1	Model	9	0.06752	0.005280	24	12.79	<.0001	Tukey-Kramer	<.0001
Model*Freq	Experimental	1	Model	15	0.07851	0.005280	24	14.87	<.0001	Tukey-Kramer	<.0001

The SAS System

Differences of Least Squares Means											
Effect	Model	Freq	Model	Freq	Estimate	Standard Error	DF	t Value	Pr > t	Adjustment	Adj P
Model*Freq	Experimental	3	Experimental	9	0.02522	0.002953	24	8.54	<.0001	Tukey-Kramer	<.0001
Model*Freq	Experimental	3	Experimental	15	0.04859	0.003746	24	12.97	<.0001	Tukey-Kramer	<.0001
Model*Freq	Experimental	3	Model	1	0.008879	0.005280	24	1.68	0.1056	Tukey-Kramer	0.6982
Model*Freq	Experimental	3	Model	3	0.03862	0.005280	24	7.32	<.0001	Tukey-Kramer	<.0001
Model*Freq	Experimental	3	Model	9	0.05413	0.005280	24	10.25	<.0001	Tukey-Kramer	<.0001
Model*Freq	Experimental	3	Model	15	0.06511	0.005280	24	12.33	<.0001	Tukey-Kramer	<.0001
Model*Freq	Experimental	9	Experimental	15	0.02337	0.002953	24	7.91	<.0001	Tukey-Kramer	<.0001
Model*Freq	Experimental	9	Model	1	-0.01634	0.005280	24	-3.10	0.0049	Tukey-Kramer	0.0790
Model*Freq	Experimental	9	Model	3	0.01340	0.005280	24	2.54	0.0180	Tukey-Kramer	0.2271
Model*Freq	Experimental	9	Model	9	0.02891	0.005280	24	5.48	<.0001	Tukey-Kramer	0.0003
Model*Freq	Experimental	9	Model	15	0.03989	0.005280	24	7.56	<.0001	Tukey-Kramer	<.0001
Model*Freq	Experimental	15	Model	1	-0.03971	0.005280	24	-7.52	<.0001	Tukey-Kramer	<.0001
Model*Freq	Experimental	15	Model	3	-0.00997	0.005280	24	-1.89	0.0711	Tukey-Kramer	0.5710
Model*Freq	Experimental	15	Model	9	0.005534	0.005280	24	1.05	0.3050	Tukey-Kramer	0.9615
Model*Freq	Experimental	15	Model	15	0.01652	0.005280	24	3.13	0.0046	Tukey-Kramer	0.0736
Model*Freq	Model	1	Model	3	0.02974	0.002258	24	13.17	<.0001	Tukey-Kramer	<.0001
Model*Freq	Model	1	Model	9	0.04525	0.004023	24	11.25	<.0001	Tukey-Kramer	<.0001
Model*Freq	Model	1	Model	15	0.05623	0.004789	24	11.74	<.0001	Tukey-Kramer	<.0001
Model*Freq	Model	3	Model	9	0.01550	0.003617	24	4.29	0.0003	Tukey-Kramer	0.0053
Model*Freq	Model	3	Model	15	0.02649	0.004588	24	5.77	<.0001	Tukey-Kramer	0.0001
Model*Freq	Model	9	Model	15	0.01099	0.003617	24	3.04	0.0057	Tukey-Kramer	0.0889

Differences of Least Squares Means									
Effect	Model	Freq	Model	Freq	Alpha	Lower	Upper	Adj Lower	Adj Upper
Model*Freq	Experimental	1	Experimental	3	0.0167	0.008654	0.01814	.	.
Model*Freq	Experimental	1	Experimental	9	0.0167	0.03017	0.04707	.	.
Model*Freq	Experimental	1	Experimental	15	0.0167	0.05193	0.07205	.	.
Model*Freq	Experimental	1	Model	1	0.0167	0.008693	0.03586	.	.
Model*Freq	Experimental	1	Model	3	0.0167	0.03844	0.06560	.	.
Model*Freq	Experimental	1	Model	9	0.0167	0.05394	0.08111	.	.
Model*Freq	Experimental	1	Model	15	0.0167	0.06493	0.09209	.	.
Model*Freq	Experimental	3	Experimental	9	0.0167	0.01762	0.03282	.	.
Model*Freq	Experimental	3	Experimental	15	0.0167	0.03895	0.05823	.	.

The SAS System

Differences of Least Squares Means									
Effect	Model	Freq	Model	Freq	Alpha	Lower	Upper	Adj Lower	Adj Upper
Model*Freq	Experimental	3	Model	1	0.0167	-0.00470	0.02246	.	.
Model*Freq	Experimental	3	Model	3	0.0167	0.02504	0.05221	.	.
Model*Freq	Experimental	3	Model	9	0.0167	0.04054	0.06771	.	.
Model*Freq	Experimental	3	Model	15	0.0167	0.05153	0.07870	.	.
Model*Freq	Experimental	9	Experimental	15	0.0167	0.01577	0.03097	.	.
Model*Freq	Experimental	9	Model	1	0.0167	-0.02992	-0.00276	.	.
Model*Freq	Experimental	9	Model	3	0.0167	-0.00018	0.02699	.	.
Model*Freq	Experimental	9	Model	9	0.0167	0.01532	0.04249	.	.
Model*Freq	Experimental	9	Model	15	0.0167	0.02631	0.05348	.	.
Model*Freq	Experimental	15	Model	1	0.0167	-0.05330	-0.02613	.	.
Model*Freq	Experimental	15	Model	3	0.0167	-0.02355	0.003614	.	.
Model*Freq	Experimental	15	Model	9	0.0167	-0.00805	0.01912	.	.
Model*Freq	Experimental	15	Model	15	0.0167	0.002938	0.03010	.	.
Model*Freq	Model	1	Model	3	0.0167	0.02393	0.03555	.	.
Model*Freq	Model	1	Model	9	0.0167	0.03490	0.05560	.	.
Model*Freq	Model	1	Model	15	0.0167	0.04391	0.06856	.	.
Model*Freq	Model	3	Model	9	0.0167	0.006197	0.02481	.	.
Model*Freq	Model	3	Model	15	0.0167	0.01469	0.03829	.	.
Model*Freq	Model	9	Model	15	0.0167	0.001681	0.02029	.	.

H Python Script: Data Retrieval

The output database file (ODB) provided displacement data for the whole model. However, only data from certain nodes were needed for post-processing. These nodes were located at the right edge of the model. A Python script was used through Abaqus to extract nodal deflection. The script was able to take individual nodes and write their displacement in a text file. Additionally, it would create a folder for all the text files for that specific model. This text files were then used to calculate tangent delta through a MATLAB script.

```

# Python script to write displacements for desired nodes into separate
# files. Each file contains the displacements in the x-direction from
# the last increment of every step.

# Import odb commands
from odbAccess import *
from abaqusConstants import *

# Import serialization commands
import pickle

# Import OS commands
import os

# Open odb file
odb = openOdb('/home/chha/125controlcran41hz/controlcran41hz.odb')

# Create folder for node displacement output (may not work on Windows)
if not os.path.exists('./node_displacement'):
    os.mkdir('./node_displacement')

# Create array with all steps and count the number of steps.
step_list = odb.steps.keys()
numSteps = len(step_list)
last_step = odb.steps.keys()[-1]

# Define which nodes to extract displacements from
# Contolcran41hz Model
node_list =
[1195,675475,96044,675250,96043,675025,1194,680100,97016,679964,97015,679695,
1197,675480,96158,675477,1183,670211,95105,670206,1167,664569,93874,664564,
93873,664559,1151,660075,92898,660081,92897,660086,1154]

# Output displacements for each node
for node_num in node_list:

    # Clear/create displacements array
    displacements = []

    # Write displacements from the last frame of every step to
    # separate files for each node
    for step in step_list:

        last_frame = odb.steps[step].frames[-1]

        # Add the value to displacement array
        displacements.append(last_frame.fieldOutputs['U'].values[node_num -
1].data[0])

```

```
# Wait to write data to file until last step
if step == last_step:

    file_name = './node_displacement/node_' + str(node_num) +
'_disp.txt'
    fid = open(file_name, 'wb')

    for index in range(0, len(displacements)):
        print>>fid, displacements[index]

    print 'Node ', node_num, ' complete'

    fid.close

odb.close
```

I MATLAB Scripts: Converting data

Multiple MATLAB scripts were utilized in retrieving deflection data from nodes of interest and translating it to desired tangent delta data. Specifically, there were three codes that were used in conjunction: TangentDelta.m, Curve-Fit.m, and rsquare.m.

The TangentDelta.m code transforms displacement into strain using the model length (LengthF). The specific test frequency is also used for a time factor calculation and "*intial_amp*" is slightly modified to accurately fit data into a sinusoidal waveform. An R^2 value of 0.99 would portray accurate fitment. Since there is some estimation integrated into the Curve-fit.m, the MATLAB code was rerun ten times for each model and averaged.

%TangentDelta.m

%% MatLab code to acquire data from Abaqus files
%% and determine the tangent delta for each analysis
clear all
close all

%% Save data from Abaqus displacement files to column vectors
filename = 'node_1151_disp.txt';
A1 = importdata(filename);

filename = 'node_1167_disp.txt';
A2 = importdata(filename);

filename = 'node_1183_disp.txt';
A3 = importdata(filename);

filename = 'node_1194_disp.txt';
A4 = importdata(filename);

filename = 'node_1195_disp.txt';
A5 = importdata(filename);

filename = 'node_1197_disp.txt';
A6 = importdata(filename);

filename = 'node_92897_disp.txt';
A7 = importdata(filename);

filename = 'node_92898_disp.txt';
A8 = importdata(filename);

filename = 'node_93873_disp.txt';
A9 = importdata(filename);

filename = 'node_93874_disp.txt';
A10 = importdata(filename);

filename = 'node_95105_disp.txt';
A11 = importdata(filename);

filename = 'node_96043_disp.txt';
A12 = importdata(filename);

filename = 'node_96044_disp.txt';

```
A13 = importdata(filename);

filename = 'node_96158_disp.txt';
A14 = importdata(filename);

filename = 'node_97015_disp.txt';
A15 = importdata(filename);

filename = 'node_97016_disp.txt';
A16 = importdata(filename);

filename = 'node_660075_disp.txt';
A17 = importdata(filename);

filename = 'node_660081_disp.txt';
A18 = importdata(filename);

filename = 'node_660086_disp.txt';
A19 = importdata(filename);

filename = 'node_664559_disp.txt';
A20 = importdata(filename);

filename = 'node_664564_disp.txt';
A21 = importdata(filename);

filename = 'node_664569_disp.txt';
A22 = importdata(filename);

filename = 'node_670206_disp.txt';
A23 = importdata(filename);

filename = 'node_670211_disp.txt';
A24 = importdata(filename);

filename = 'node_675025_disp.txt';
A25 = importdata(filename);

filename = 'node_675250_disp.txt';
A26 = importdata(filename);

filename = 'node_675475_disp.txt';
A27 = importdata(filename);

filename = 'node_675477_disp.txt';
```

```
A28 = importdata(filename);

filename = 'node_675480_disp.txt';
A29 = importdata(filename);

filename = 'node_679695_disp.txt';
A30 = importdata(filename);

filename = 'node_679964_disp.txt';
A31 = importdata(filename);

filename = 'node_680100_disp.txt';
A32 = importdata(filename);

%% Combine all node displacement column vectors into a single array
Disp_Data = cat(2,A1,A2,A3,A4,A5,A6,A7,A8,A9,A10,A11,A12,A13,A14,A15,
A16,A17,A18,A19,A20,A21,A22,A23,A24,A25,A26,A27,A28,A29,A30,A31,A32);

%% Determine an average displacement from all the nodes and save to a
%% single column vector
Disp_Data = transpose(Disp_Data);
Ave_Displacement = squeeze(mean(Disp_Data));
Ave_Displacement = transpose(Ave_Displacement);

%% Calculate the average strain behavior based on periodic unit length
p = 6.8510358;
Data = Ave_Displacement/p; % Divide by the periodic length to get strain

%% Remove data from first 10 cycles
Data_10_cycles = Data(201:length(Data));

%% Initialize frequency, time, and initial amplitude
f = 1; % 1 Hz frequency
% f = 3; % 3 Hz frequency
% f = 9; % 9 Hz frequency
% f = 15; % 15 Hz frequency

% t = 1/(20.*f):1/(20.*f):20/f; % Time for entire data

t = 10/f + 1/(20.*f):1/(20.*f):20/f; % Time for last 10 cycles
t = t(:); % Transpose time to match Data vector

t2 = 0:1/(20.*f):1/f;
```

```
initial_amp = 0.475*2.12e-4;

%% Function file that accepts curve parameters as inputs and then outputs
%% fitting error
%Starting = rand(1,3);
Starting = rand(1,2);
options = optimset('Display','iter');
%Estimates = fminsearch(@CurveFit,Starting,options,t,Data,f,initial_amp);
% Curve fit for entire data

Estimates = fminsearch(@CurveFit,Starting,options,t,Data_10_cycles,f,...
    initial_amp);
% Curve fit for last 10 cycles

%% Calculate curve fit equation and coefficient of determination
strain = Estimates(1)*sin(2.*pi.*f*t - Estimates(2)) + initial_amp;
[r2 rmse] = rsquare(Data_10_cycles,strain); % r^2 value for last 10 cycles

%strain = Estimates(1)*sin(2.*pi.*f*t - Estimates(2)) + initial_amp;
%[r2 rmse] = rsquare(Data,strain); % r^2 value for entire data

%% Normalized stress and strain history for first cycle
norm_stress = sin(2.*pi.*f*t2);
norm_strain = sin(2.*pi.*f*t2 - Estimates(2));

%% Plot the fitted curve over the raw data
fig1 = figure;
plot(t,Data_10_cycles,'*') % Plot last 10 cycles
%plot(t,Data,'*') % Plot entire data
hold on
plot(t,strain,'r')
xlabel('Time (seconds)','FontSize',16)
ylabel('Strain (unitless)','FontSize',16)
title('Tangent Delta Calculation','FontSize',16)
str = {'R-squared',num2str(r2),'Tangent Delta',num2str(Estimates(2))};
annotation('textbox',[.7,.12,.2,.15],'String',str);
set(fig1,'Position',[1 540 500 400])
%% Plot normalized stress and strain for 1 cycle on a separate figure
fig2 = figure;
plot(t2,norm_stress,'--r')
hold on
plot(t2,norm_strain,'k')
set(fig2,'Position',[1 1 500 400])
```

```
function [r2 rmse] = rsquare(y,f,varargin)
% Compute coefficient of determination of data fit model and RMSE
%
% [r2 rmse] = rsquare(y,f)
% [r2 rmse] = rsquare(y,f,c)
%
% RSQUARE computes the coefficient of determination (R-square) value from
% actual data Y and model data F. The code uses a general version of
% R-square, based on comparing the variability of the estimation errors
% with the variability of the original values. RSQUARE also outputs the
% root mean squared error (RMSE) for the user's convenience.
%
% Note: RSQUARE ignores comparisons involving NaN values.
%
% INPUTS
%   Y       : Actual data
%   F       : Model fit
%
% OPTION
%   C       : Constant term in model
%             R-square may be a questionable measure of fit when no
%             constant term is included in the model.
%   [DEFAULT] TRUE : Use traditional R-square computation
%               FALSE : Uses alternate R-square computation for model
%                       without constant term [R2 = 1 - NORM(Y-F)/NORM(Y)]
%
% OUTPUT
%   R2      : Coefficient of determination
%   RMSE    : Root mean squared error
%
% EXAMPLE
%   x = 0:0.1:10;
%   y = 2.*x + 1 + randn(size(x));
%   p = polyfit(x,y,1);
%   f = polyval(p,x);
%   [r2 rmse] = rsquare(y,f);
%   figure; plot(x,y,'b-');
%   hold on; plot(x,f,'r-');
%   title(strcat(['R2 = ' num2str(r2) '; RMSE = ' num2str(rmse)]))
%
% Jered R Wells
% 11/17/11
% jered [dot] wells [at] duke [dot] edu
%
% v1.2 (02/14/2012)
```

```
%
% Thanks to John D'Errico for useful comments and insight which has helped
% to improve this code. His code POLYFITN was consulted in the inclusion of
% the C-option (REF. File ID: #34765).

if isempty(varargin); c = true;
elseif length(varargin)>1; error 'Too many input arguments';
elseif ~islogical(varargin{1}); error 'C must be logical (TRUE||FALSE)'
else c = varargin{1};
end

% Compare inputs
if ~all(size(y)==size(f)); error 'Y and F must be the same size'; end

% Check for NaN
tmp = ~or(isnan(y),isnan(f));
y = y(tmp);
f = f(tmp);

if c; r2 = max(0,1 - sum((y(:)-f(:)).^2)/sum((y(:)-mean(y(:))).^2));
else r2 = 1 - sum((y(:)-f(:)).^2)/sum((y(:)).^2);
    if r2<0
        % http://web.maths.unsw.edu.au/~adelle/Garvan/Assays/GoodnessOfFit.html
        warning('Consider adding a constant term to your model') %#ok<WNTAG>
        r2 = 0;
    end
end
rmse = sqrt(mean((y(:) - f(:)).^2));
```

```
% CurveFit.m
```

```
function sse = CurveFit(params,Input,Actual_Output,f,initial_amp)
amplitude = params(1);
delta = params(2);
```

```
Fitted_Curve = amplitude.*sin((2.*pi.*f)*Input - delta) + initial_amp;
Error_Vector = Fitted_Curve - Actual_Output;
```

```
%% When curvefitting, a typical quantity to minimize
%% is the sum of squares error
sse = sum(Error_Vector.^2);
```

J Python Script: Model Generation

Composite Models were generated using a Python script through the Abaqus GUI. The script used data from the AFM done on the sheep by the University of Michigan, Ann Arbor. Since the distribution of D-spacing exhibited a normal distribution, a "Random.Gaussian" function was used to generate half unit cell lengths using the means and standard deviation for each respective ulna/radius section.

A constant mineralization parameter is applied to each half unit cell. The constant 0.84 value allows each half unit cell to the 30% volumetric mineralization discovered by Mendoza [52].

Since the Composite Model is composed of 200 half unit cells (2 x 100), there will likely be an offset in model rows. This offset nature is unwanted due to a potential of the jagged edge in the model to cause unwanted deformation and stress concentration. To prevent this from occurring, a collagen spacer is applied to the shorter of the two rows. If the spacer is larger than 5% of the previous half unit cell, it was deemed "biologically invalid". The model generation code was rerun until a "biologically valid" model was created.

The Python script additionally creates the materials and their respective properties. It also creates the x-symmetry boundary condition. However, due to the varying lengths of each created model, it is unable to accurately assign the y-symmetry boundary condition and the model's loading conditions. Material assignment was done by hand through the Abaqus GUI.

```

# coding=utf-8
# Do not delete the following import lines
from abaqus import *
from abaqusConstants import *
import __main__

def TWOXONEHUNDREDFINAL():
    import section
    import regionToolset
    import displayGroupMdbToolset as dgm
    import part
    import material
    import assembly
    import step
    import interaction
    import load
    import mesh
    import job
    import sketch
    import visualization
    import xyPlot
    import displayGroupOdbToolset as dgo
    import connectorBehavior
    import random

#Dspacing mean and standard dev (Control Cran)
#   DSmean=0.06841994
#   DSstdev=0.001281148

#Dspacing mean and standard dev (Control Caud)
DSmean=0.066353
DSstdev=0.001688634

#Random dspacing
ds1 =random.gauss (DSmean,DSstdev)
ds2 =random.gauss (DSmean,DSstdev)
ds3 =random.gauss (DSmean,DSstdev)
ds4 =random.gauss (DSmean,DSstdev)
ds5 =random.gauss (DSmean,DSstdev)
ds6 =random.gauss (DSmean,DSstdev)
ds7 =random.gauss (DSmean,DSstdev)
ds8 =random.gauss (DSmean,DSstdev)
ds9 =random.gauss (DSmean,DSstdev)
ds10 =random.gauss (DSmean,DSstdev)
ds11 =random.gauss (DSmean,DSstdev)
ds12 =random.gauss (DSmean,DSstdev)
ds13 =random.gauss (DSmean,DSstdev)
ds14 =random.gauss (DSmean,DSstdev)

```

```
ds15 =random.gauss (DSmean,DSstdev)
ds16 =random.gauss (DSmean,DSstdev)
ds17 =random.gauss (DSmean,DSstdev)
ds18 =random.gauss (DSmean,DSstdev)
ds19 =random.gauss (DSmean,DSstdev)
ds20 =random.gauss (DSmean,DSstdev)
ds21 =random.gauss (DSmean,DSstdev)
ds22 =random.gauss (DSmean,DSstdev)
ds23 =random.gauss (DSmean,DSstdev)
ds24 =random.gauss (DSmean,DSstdev)
ds25 =random.gauss (DSmean,DSstdev)
ds26 =random.gauss (DSmean,DSstdev)
ds27 =random.gauss (DSmean,DSstdev)
ds28 =random.gauss (DSmean,DSstdev)
ds29 =random.gauss (DSmean,DSstdev)
ds30 =random.gauss (DSmean,DSstdev)
ds31 =random.gauss (DSmean,DSstdev)
ds32 =random.gauss (DSmean,DSstdev)
ds33 =random.gauss (DSmean,DSstdev)
ds34 =random.gauss (DSmean,DSstdev)
ds35 =random.gauss (DSmean,DSstdev)
ds36 =random.gauss (DSmean,DSstdev)
ds37 =random.gauss (DSmean,DSstdev)
ds38 =random.gauss (DSmean,DSstdev)
ds39 =random.gauss (DSmean,DSstdev)
ds40 =random.gauss (DSmean,DSstdev)
ds41 =random.gauss (DSmean,DSstdev)
ds42 =random.gauss (DSmean,DSstdev)
ds43 =random.gauss (DSmean,DSstdev)
ds44 =random.gauss (DSmean,DSstdev)
ds45 =random.gauss (DSmean,DSstdev)
ds46 =random.gauss (DSmean,DSstdev)
ds47 =random.gauss (DSmean,DSstdev)
ds48 =random.gauss (DSmean,DSstdev)
ds49 =random.gauss (DSmean,DSstdev)
ds50 =random.gauss (DSmean,DSstdev)
ds51 =random.gauss (DSmean,DSstdev)
ds52 =random.gauss (DSmean,DSstdev)
ds53 =random.gauss (DSmean,DSstdev)
ds54 =random.gauss (DSmean,DSstdev)
ds55 =random.gauss (DSmean,DSstdev)
ds56 =random.gauss (DSmean,DSstdev)
ds57 =random.gauss (DSmean,DSstdev)
ds58 =random.gauss (DSmean,DSstdev)
ds59 =random.gauss (DSmean,DSstdev)
ds60 =random.gauss (DSmean,DSstdev)
ds61 =random.gauss (DSmean,DSstdev)
ds62 =random.gauss (DSmean,DSstdev)
```

```
ds63 =random.gauss (DSmean,DSstdev)
ds64 =random.gauss (DSmean,DSstdev)
ds65 =random.gauss (DSmean,DSstdev)
ds66 =random.gauss (DSmean,DSstdev)
ds67 =random.gauss (DSmean,DSstdev)
ds68 =random.gauss (DSmean,DSstdev)
ds69 =random.gauss (DSmean,DSstdev)
ds70 =random.gauss (DSmean,DSstdev)
ds71 =random.gauss (DSmean,DSstdev)
ds72 =random.gauss (DSmean,DSstdev)
ds73 =random.gauss (DSmean,DSstdev)
ds74 =random.gauss (DSmean,DSstdev)
ds75 =random.gauss (DSmean,DSstdev)
ds76 =random.gauss (DSmean,DSstdev)
ds77 =random.gauss (DSmean,DSstdev)
ds78 =random.gauss (DSmean,DSstdev)
ds79 =random.gauss (DSmean,DSstdev)
ds80 =random.gauss (DSmean,DSstdev)
ds81 =random.gauss (DSmean,DSstdev)
ds82 =random.gauss (DSmean,DSstdev)
ds83 =random.gauss (DSmean,DSstdev)
ds84 =random.gauss (DSmean,DSstdev)
ds85 =random.gauss (DSmean,DSstdev)
ds86 =random.gauss (DSmean,DSstdev)
ds87 =random.gauss (DSmean,DSstdev)
ds88 =random.gauss (DSmean,DSstdev)
ds89 =random.gauss (DSmean,DSstdev)
ds90 =random.gauss (DSmean,DSstdev)
ds91 =random.gauss (DSmean,DSstdev)
ds92 =random.gauss (DSmean,DSstdev)
ds93 =random.gauss (DSmean,DSstdev)
ds94 =random.gauss (DSmean,DSstdev)
ds95 =random.gauss (DSmean,DSstdev)
ds96 =random.gauss (DSmean,DSstdev)
ds97 =random.gauss (DSmean,DSstdev)
ds98 =random.gauss (DSmean,DSstdev)
ds99 =random.gauss (DSmean,DSstdev)
ds100 =random.gauss (DSmean,DSstdev)
ds101 =random.gauss (DSmean,DSstdev)
ds102 =random.gauss (DSmean,DSstdev)
ds103 =random.gauss (DSmean,DSstdev)
ds104 =random.gauss (DSmean,DSstdev)
ds105 =random.gauss (DSmean,DSstdev)
ds106 =random.gauss (DSmean,DSstdev)
ds107 =random.gauss (DSmean,DSstdev)
ds108 =random.gauss (DSmean,DSstdev)
ds109 =random.gauss (DSmean,DSstdev)
ds110 =random.gauss (DSmean,DSstdev)
```



```
ds111 =random.gauss (DSmean,DSstdev)
ds112 =random.gauss (DSmean,DSstdev)
ds113 =random.gauss (DSmean,DSstdev)
ds114 =random.gauss (DSmean,DSstdev)
ds115 =random.gauss (DSmean,DSstdev)
ds116 =random.gauss (DSmean,DSstdev)
ds117 =random.gauss (DSmean,DSstdev)
ds118 =random.gauss (DSmean,DSstdev)
ds119 =random.gauss (DSmean,DSstdev)
ds120 =random.gauss (DSmean,DSstdev)
ds121 =random.gauss (DSmean,DSstdev)
ds122 =random.gauss (DSmean,DSstdev)
ds123 =random.gauss (DSmean,DSstdev)
ds124 =random.gauss (DSmean,DSstdev)
ds125 =random.gauss (DSmean,DSstdev)
ds126 =random.gauss (DSmean,DSstdev)
ds127 =random.gauss (DSmean,DSstdev)
ds128 =random.gauss (DSmean,DSstdev)
ds129 =random.gauss (DSmean,DSstdev)
ds130 =random.gauss (DSmean,DSstdev)
ds131 =random.gauss (DSmean,DSstdev)
ds132 =random.gauss (DSmean,DSstdev)
ds133 =random.gauss (DSmean,DSstdev)
ds134 =random.gauss (DSmean,DSstdev)
ds135 =random.gauss (DSmean,DSstdev)
ds136 =random.gauss (DSmean,DSstdev)
ds137 =random.gauss (DSmean,DSstdev)
ds138 =random.gauss (DSmean,DSstdev)
ds139 =random.gauss (DSmean,DSstdev)
ds140 =random.gauss (DSmean,DSstdev)
ds141 =random.gauss (DSmean,DSstdev)
ds142 =random.gauss (DSmean,DSstdev)
ds143 =random.gauss (DSmean,DSstdev)
ds144 =random.gauss (DSmean,DSstdev)
ds145 =random.gauss (DSmean,DSstdev)
ds146 =random.gauss (DSmean,DSstdev)
ds147 =random.gauss (DSmean,DSstdev)
ds148 =random.gauss (DSmean,DSstdev)
ds149 =random.gauss (DSmean,DSstdev)
ds150 =random.gauss (DSmean,DSstdev)
ds151 =random.gauss (DSmean,DSstdev)
ds152 =random.gauss (DSmean,DSstdev)
ds153 =random.gauss (DSmean,DSstdev)
ds154 =random.gauss (DSmean,DSstdev)
ds155 =random.gauss (DSmean,DSstdev)
ds156 =random.gauss (DSmean,DSstdev)
ds157 =random.gauss (DSmean,DSstdev)
ds158 =random.gauss (DSmean,DSstdev)
```

```
ds159 =random.gauss (DSmean,DSstdev)
ds160 =random.gauss (DSmean,DSstdev)
ds161 =random.gauss (DSmean,DSstdev)
ds162 =random.gauss (DSmean,DSstdev)
ds163 =random.gauss (DSmean,DSstdev)
ds164 =random.gauss (DSmean,DSstdev)
ds165 =random.gauss (DSmean,DSstdev)
ds166 =random.gauss (DSmean,DSstdev)
ds167 =random.gauss (DSmean,DSstdev)
ds168 =random.gauss (DSmean,DSstdev)
ds169 =random.gauss (DSmean,DSstdev)
ds170 =random.gauss (DSmean,DSstdev)
ds171 =random.gauss (DSmean,DSstdev)
ds172 =random.gauss (DSmean,DSstdev)
ds173 =random.gauss (DSmean,DSstdev)
ds174 =random.gauss (DSmean,DSstdev)
ds175 =random.gauss (DSmean,DSstdev)
ds176 =random.gauss (DSmean,DSstdev)
ds177 =random.gauss (DSmean,DSstdev)
ds178 =random.gauss (DSmean,DSstdev)
ds179 =random.gauss (DSmean,DSstdev)
ds180 =random.gauss (DSmean,DSstdev)
ds181 =random.gauss (DSmean,DSstdev)
ds182 =random.gauss (DSmean,DSstdev)
ds183 =random.gauss (DSmean,DSstdev)
ds184 =random.gauss (DSmean,DSstdev)
ds185 =random.gauss (DSmean,DSstdev)
ds186 =random.gauss (DSmean,DSstdev)
ds187 =random.gauss (DSmean,DSstdev)
ds188 =random.gauss (DSmean,DSstdev)
ds189 =random.gauss (DSmean,DSstdev)
ds190 =random.gauss (DSmean,DSstdev)
ds191 =random.gauss (DSmean,DSstdev)
ds192 =random.gauss (DSmean,DSstdev)
ds193 =random.gauss (DSmean,DSstdev)
ds194 =random.gauss (DSmean,DSstdev)
ds195 =random.gauss (DSmean,DSstdev)
ds196 =random.gauss (DSmean,DSstdev)
ds197 =random.gauss (DSmean,DSstdev)
ds198 =random.gauss (DSmean,DSstdev)
ds199 =random.gauss (DSmean,DSstdev)
ds200 =random.gauss (DSmean,DSstdev)
y1=.84*ds1
y2=.84*ds2
y3=.84*ds3
y4=.84*ds4
y5=.84*ds5
y6=.84*ds6
```

y7=.84*ds7
y8=.84*ds8
y9=.84*ds9
y10=.84*ds10
y11=.84*ds11
y12=.84*ds12
y13=.84*ds13
y14=.84*ds14
y15=.84*ds15
y16=.84*ds16
y17=.84*ds17
y18=.84*ds18
y19=.84*ds19
y20=.84*ds20
y21=.84*ds21
y22=.84*ds22
y23=.84*ds23
y24=.84*ds24
y25=.84*ds25
y26=.84*ds26
y27=.84*ds27
y28=.84*ds28
y29=.84*ds29
y30=.84*ds30
y31=.84*ds31
y32=.84*ds32
y33=.84*ds33
y34=.84*ds34
y35=.84*ds35
y36=.84*ds36
y37=.84*ds37
y38=.84*ds38
y39=.84*ds39
y40=.84*ds40
y41=.84*ds41
y42=.84*ds42
y43=.84*ds43
y44=.84*ds44
y45=.84*ds45
y46=.84*ds46
y47=.84*ds47
y48=.84*ds48
y49=.84*ds49
y50=.84*ds50
y51=.84*ds51
y52=.84*ds52
y53=.84*ds53
y54=.84*ds54

y55=.84*ds55
y56=.84*ds56
y57=.84*ds57
y58=.84*ds58
y59=.84*ds59
y60=.84*ds60
y61=.84*ds61
y62=.84*ds62
y63=.84*ds63
y64=.84*ds64
y65=.84*ds65
y66=.84*ds66
y67=.84*ds67
y68=.84*ds68
y69=.84*ds69
y70=.84*ds70
y71=.84*ds71
y72=.84*ds72
y73=.84*ds73
y74=.84*ds74
y75=.84*ds75
y76=.84*ds76
y77=.84*ds77
y78=.84*ds78
y79=.84*ds79
y80=.84*ds80
y81=.84*ds81
y82=.84*ds82
y83=.84*ds83
y84=.84*ds84
y85=.84*ds85
y86=.84*ds86
y87=.84*ds87
y88=.84*ds88
y89=.84*ds89
y90=.84*ds90
y91=.84*ds91
y92=.84*ds92
y93=.84*ds93
y94=.84*ds94
y95=.84*ds95
y96=.84*ds96
y97=.84*ds97
y98=.84*ds98
y99=.84*ds99
y100=.84*ds100
y101=.84*ds101
y102=.84*ds102

y103=.84*ds103
y104=.84*ds104
y105=.84*ds105
y106=.84*ds106
y107=.84*ds107
y108=.84*ds108
y109=.84*ds109
y110=.84*ds110
y111=.84*ds111
y112=.84*ds112
y113=.84*ds113
y114=.84*ds114
y115=.84*ds115
y116=.84*ds116
y117=.84*ds117
y118=.84*ds118
y119=.84*ds119
y120=.84*ds120
y121=.84*ds121
y122=.84*ds122
y123=.84*ds123
y124=.84*ds124
y125=.84*ds125
y126=.84*ds126
y127=.84*ds127
y128=.84*ds128
y129=.84*ds129
y130=.84*ds130
y131=.84*ds131
y132=.84*ds132
y133=.84*ds133
y134=.84*ds134
y135=.84*ds135
y136=.84*ds136
y137=.84*ds137
y138=.84*ds138
y139=.84*ds139
y140=.84*ds140
y141=.84*ds141
y142=.84*ds142
y143=.84*ds143
y144=.84*ds144
y145=.84*ds145
y146=.84*ds146
y147=.84*ds147
y148=.84*ds148
y149=.84*ds149
y150=.84*ds150

y151=.84*ds151
y152=.84*ds152
y153=.84*ds153
y154=.84*ds154
y155=.84*ds155
y156=.84*ds156
y157=.84*ds157
y158=.84*ds158
y159=.84*ds159
y160=.84*ds160
y161=.84*ds161
y162=.84*ds162
y163=.84*ds163
y164=.84*ds164
y165=.84*ds165
y166=.84*ds166
y167=.84*ds167
y168=.84*ds168
y169=.84*ds169
y170=.84*ds170
y171=.84*ds171
y172=.84*ds172
y173=.84*ds173
y174=.84*ds174
y175=.84*ds175
y176=.84*ds176
y177=.84*ds177
y178=.84*ds178
y179=.84*ds179
y180=.84*ds180
y181=.84*ds181
y182=.84*ds182
y183=.84*ds183
y184=.84*ds184
y185=.84*ds185
y186=.84*ds186
y187=.84*ds187
y188=.84*ds188
y189=.84*ds189
y190=.84*ds190
y191=.84*ds191
y192=.84*ds192
y193=.84*ds193
y194=.84*ds194
y195=.84*ds195
y196=.84*ds196
y197=.84*ds197
y198=.84*ds198

y199=.84*ds199
y200=.84*ds200

x1=ds1-y1
x2=ds2-y2
x3=ds3-y3
x4=ds4-y4
x5=ds5-y5
x6=ds6-y6
x7=ds7-y7
x8=ds8-y8
x9=ds9-y9
x10=ds10-y10
x11=ds11-y11
x12=ds12-y12
x13=ds13-y13
x14=ds14-y14
x15=ds15-y15
x16=ds16-y16
x17=ds17-y17
x18=ds18-y18
x19=ds19-y19
x20=ds20-y20
x21=ds21-y21
x22=ds22-y22
x23=ds23-y23
x24=ds24-y24
x25=ds25-y25
x26=ds26-y26
x27=ds27-y27
x28=ds28-y28
x29=ds29-y29
x30=ds30-y30
x31=ds31-y31
x32=ds32-y32
x33=ds33-y33
x34=ds34-y34
x35=ds35-y35
x36=ds36-y36
x37=ds37-y37
x38=ds38-y38
x39=ds39-y39
x40=ds40-y40
x41=ds41-y41
x42=ds42-y42
x43=ds43-y43
x44=ds44-y44
x45=ds45-y45

x46=ds46-y46
x47=ds47-y47
x48=ds48-y48
x49=ds49-y49
x50=ds50-y50
x51=ds51-y51
x52=ds52-y52
x53=ds53-y53
x54=ds54-y54
x55=ds55-y55
x56=ds56-y56
x57=ds57-y57
x58=ds58-y58
x59=ds59-y59
x60=ds60-y60
x61=ds61-y61
x62=ds62-y62
x63=ds63-y63
x64=ds64-y64
x65=ds65-y65
x66=ds66-y66
x67=ds67-y67
x68=ds68-y68
x69=ds69-y69
x70=ds70-y70
x71=ds71-y71
x72=ds72-y72
x73=ds73-y73
x74=ds74-y74
x75=ds75-y75
x76=ds76-y76
x77=ds77-y77
x78=ds78-y78
x79=ds79-y79
x80=ds80-y80
x81=ds81-y81
x82=ds82-y82
x83=ds83-y83
x84=ds84-y84
x85=ds85-y85
x86=ds86-y86
x87=ds87-y87
x88=ds88-y88
x89=ds89-y89
x90=ds90-y90
x91=ds91-y91
x92=ds92-y92
x93=ds93-y93

x94=ds94-y94
x95=ds95-y95
x96=ds96-y96
x97=ds97-y97
x98=ds98-y98
x99=ds99-y99
x100=ds100-y100
x101=ds101-y101
x102=ds102-y102
x103=ds103-y103
x104=ds104-y104
x105=ds105-y105
x106=ds106-y106
x107=ds107-y107
x108=ds108-y108
x109=ds109-y109
x100=ds100-y100
x111=ds111-y111
x112=ds112-y112
x113=ds113-y113
x114=ds114-y114
x115=ds115-y115
x116=ds116-y116
x117=ds117-y117
x118=ds118-y118
x119=ds119-y119
x120=ds120-y120
x121=ds121-y121
x122=ds122-y122
x123=ds123-y123
x124=ds124-y124
x125=ds125-y125
x126=ds126-y126
x127=ds127-y127
x128=ds128-y128
x129=ds129-y129
x130=ds130-y130
x131=ds131-y131
x132=ds132-y132
x133=ds133-y133
x134=ds134-y134
x135=ds135-y135
x136=ds136-y136
x137=ds137-y137
x138=ds138-y138
x139=ds139-y139
x140=ds140-y140
x141=ds141-y141

x142=ds142-y142
x143=ds143-y143
x144=ds144-y144
x145=ds145-y145
x146=ds146-y146
x147=ds147-y147
x148=ds148-y148
x149=ds149-y149
x150=ds150-y150
x151=ds151-y151
x152=ds152-y152
x153=ds153-y153
x154=ds154-y154
x155=ds155-y155
x156=ds156-y156
x157=ds157-y157
x158=ds158-y158
x159=ds159-y159
x160=ds160-y160
x161=ds161-y161
x162=ds162-y162
x163=ds163-y163
x164=ds164-y164
x165=ds165-y165
x166=ds166-y166
x167=ds167-y167
x168=ds168-y168
x169=ds169-y169
x170=ds170-y170
x171=ds171-y171
x172=ds172-y172
x173=ds173-y173
x174=ds174-y174
x175=ds175-y175
x176=ds176-y176
x177=ds177-y177
x178=ds178-y178
x179=ds179-y179
x180=ds180-y180
x181=ds181-y181
x182=ds182-y182
x183=ds183-y183
x184=ds184-y184
x185=ds185-y185
x186=ds186-y186
x187=ds187-y187
x188=ds188-y188
x189=ds189-y189

```
x190=ds190-y190
x191=ds191-y191
x192=ds192-y192
x193=ds193-y193
x194=ds194-y194
x195=ds195-y195
x196=ds196-y196
x197=ds197-y197
x198=ds198-y198
x199=ds199-y199
x200=ds200-y200
```

```
set1=ds1+ds2+ds3+ds4+ds5+ds6+ds7+ds8+ds9+ds10
set2=set1+ds11+ds12+ds13+ds14+ds15+ds16+ds17+ds18+ds19+ds20
set3=set2+ds21+ds22+ds23+ds24+ds25+ds26+ds27+ds28+ds29+ds30
set4=set3+ds31+ds32+ds33+ds34+ds35+ds36+ds37+ds38+ds39+ds40
set5=set4+ds41+ds42+ds43+ds44+ds45+ds46+ds47+ds48+ds49+ds50
set6=set5+ds51+ds52+ds53+ds54+ds55+ds56+ds57+ds58+ds59+ds60
set7=set6+ds61+ds62+ds63+ds64+ds65+ds66+ds67+ds68+ds69+ds70
set8=set7+ds71+ds72+ds73+ds74+ds75+ds76+ds77+ds78+ds79+ds80
set9=set8+ds81+ds82+ds83+ds84+ds85+ds86+ds87+ds88+ds89+ds90
set10=set9+ds91+ds92+ds93+ds94+ds95+ds96+ds97+ds98+ds99+ds100
```

```
set11=ds101+ds102+ds103+ds104+ds105+ds106+ds107+ds108+ds109+ds110
set12=set11+ds111+ds112+ds113+ds114+ds115+ds116+ds117+ds118+ds119+ds120
set13=set12+ds121+ds122+ds123+ds124+ds125+ds126+ds127+ds128+ds129+ds130
set14=set13+ds131+ds132+ds133+ds134+ds135+ds136+ds137+ds138+ds139+ds140
set15=set14+ds141+ds142+ds143+ds144+ds145+ds146+ds147+ds148+ds149+ds150
set16=set15+ds151+ds152+ds153+ds154+ds155+ds156+ds157+ds158+ds159+ds160
set17=set16+ds161+ds162+ds163+ds164+ds165+ds166+ds167+ds168+ds169+ds170
set18=set17+ds171+ds172+ds173+ds174+ds175+ds176+ds177+ds178+ds179+ds180
set19=set18+ds181+ds182+ds183+ds184+ds185+ds186+ds187+ds188+ds189+ds190
set20=set19+ds191+ds192+ds193+ds194+ds195+ds196+ds197+ds198+ds199+ds200
```

```
# print ('dspace1 =',ds1)
# print ('dspace2 =',ds2)
# print ('dspace3 =',ds3)
# print ('dspace4 =',ds4)
```

```
#Model lengths
```

```
Length1= sum(ds1,ds2,ds3,ds4,ds5,ds6,ds7,ds8,ds9,ds10,
             ds11,ds12,ds13,ds14,ds15,ds16,ds17,ds18,ds19,ds20,
             ds21,ds22,ds23,ds24,ds25,ds26,ds27,ds28,ds29,ds30,
             ds31,ds32,ds33,ds34,ds35,ds36,ds37,ds38,ds39,ds40,
             ds41,ds42,ds43,ds44,ds45,ds46,ds47,ds48,ds49,ds50,
             ds51,ds52,ds53,ds54,ds55,ds56,ds57,ds58,ds59,ds60,
             ds61,ds62,ds63,ds64,ds65,ds66,ds67,ds68,ds69,ds70,
```

```

        ds71,ds72,ds73,ds74,ds75,ds76,ds77,ds78,ds79,ds80,
        ds81,ds82,ds83,ds84,ds85,ds86,ds87,ds88,ds89,ds90,
        ds91,ds92,ds93,ds94,ds95,ds96,ds97,ds98,ds99,ds100)
Length2= sum(ds101,ds102,ds103,ds104,ds105,ds106,ds107,ds108,ds109,ds110,
        ds111,ds112,ds113,ds114,ds115,ds116,ds117,ds118,ds119,ds120,
        ds121,ds122,ds123,ds124,ds125,ds126,ds127,ds128,ds129,ds130,
        ds131,ds132,ds133,ds134,ds135,ds136,ds137,ds138,ds139,ds140,
        ds141,ds142,ds143,ds144,ds145,ds146,ds147,ds148,ds149,ds150,
        ds151,ds152,ds153,ds154,ds155,ds156,ds157,ds158,ds159,ds160,
        ds161,ds162,ds163,ds164,ds165,ds166,ds167,ds168,ds169,ds170,
        ds171,ds172,ds173,ds174,ds175,ds176,ds177,ds178,ds179,ds180,
        ds181,ds182,ds183,ds184,ds185,ds186,ds187,ds188,ds189,ds190,
        ds191,ds192,ds193,ds194,ds195,ds196,ds197,ds198,ds199,ds200)
LengthF= max(Length1, Length2)
# LengthFh=LengthF/2
print ('Length1 =',Length1)
print ('Length2 =',Length2)
print ('LengthF =',LengthF)
LengthDiff=Length1-Length2
print ('Difference in Row Length =',LengthDiff)

#LengthF=Length1 (ie. The Top Row is Larger; Bottom Row Has Spacer)

if LengthF == Length1:
    s1 = mdb.models['Model-1'].ConstrainedSketch(name='__profile__',
        sheetSize=200.0)
    g, v, d, c = s1.geometry, s1.vertices, s1.dimensions, s1.constraints
    s1.setPrimaryObject(option=STANDALONE)
    s1.rectangle(point1=(0.0, 0.0), point2=(LengthF, 0.007))
    p = mdb.models['Model-1'].Part(name='Composite Bone',
        dimensionality=TWO_D_PLANAR, type=DEFORMABLE_BODY)
    p = mdb.models['Model-1'].parts['Composite Bone']
    p.BaseShell(sketch=s1)
    s1.unsetPrimaryObject()
    p = mdb.models['Model-1'].parts['Composite Bone']
    session.viewports['Viewport: 1'].setValues(displayedObject=p)
    del mdb.models['Model-1'].sketches['__profile__']
    f, e1, d2 = p.faces, p.edges, p.datums
    t = p.MakeSketchTransform(sketchPlane=f[0], sketchPlaneSide=SIDE1,
origin=(
    0.0, 0.0, 0.0))
    s = mdb.models['Model-1'].ConstrainedSketch(name='__profile__',
        sheetSize=0.268, gridSpacing=0.006, transform=t)
    g, v, d, c = s.geometry, s.vertices, s.dimensions, s.constraints
    s.sketchOptions.setValues(decimalPlaces=3)
    s.setPrimaryObject(option=SUPERIMPOSE)
    p = mdb.models['Model-1'].parts['Composite Bone']

```

```

p.projectReferencesOntoSketch(sketch=s, filter=COPLANAR_EDGES)
s.Line(point1=(0, 0.00125), point2=(LengthF, 0.00125))
s.HorizontalConstraint(entity=g[6], addUndoState=False)
s.Line(point1=(0, 0.00275), point2=(LengthF, 0.00275))
s.HorizontalConstraint(entity=g[7], addUndoState=False)
s.Line(point1=(0, 0.0035), point2=(LengthF, 0.0035))
s.HorizontalConstraint(entity=g[8], addUndoState=False)
s.Line(point1=(0, 0.00425), point2=(LengthF, 0.00425))
s.HorizontalConstraint(entity=g[9], addUndoState=False)
s.Line(point1=(0, 0.00575), point2=(LengthF, 0.00575))
s.HorizontalConstraint(entity=g[10], addUndoState=False)

```

#Top Row, Dspace 1-10

```

s.Line(point1=(y1, 0.007), point2=(y1, 0.00575))

s.Line(point1=(ds1, 0.007), point2=(ds1, 0.0035))

s.Line(point1=(ds1+x2, 0.007), point2=(ds1+x2, 0.00575))

s.Line(point1=(ds1+ds2, 0.007), point2=(ds1+ds2, 0.0035))

s.Line(point1=(ds1+ds2+y3, 0.007), point2=(ds1+ds2+y3, 0.00575))

s.Line(point1=(ds1+ds2+ds3, 0.007), point2=(ds1+ds2+ds3, 0.0035))

s.Line(point1=(ds1+ds2+ds3+x4, 0.007), point2=(ds1+ds2+ds3+x4, 0.00575))

s.Line(point1=(ds1+ds2+ds3+ds4, 0.007), point2=(ds1+ds2+ds3+ds4, 0.0035))

s.Line(point1=(ds1+ds2+ds3+ds4+y5, 0.007), point2=(ds1+ds2+ds3+ds4+y5,
0.00575))

s.Line(point1=(ds1+ds2+ds3+ds4+ds5, 0.007), point2=(ds1+ds2+ds3+ds4+ds5,
0.0035))

s.Line(point1=(ds1+ds2+ds3+ds4+ds5+x6, 0.007),
point2=(ds1+ds2+ds3+ds4+ds5+x6, 0.00575))

s.Line(point1=(ds1+ds2+ds3+ds4+ds5+ds6, 0.007),
point2=(ds1+ds2+ds3+ds4+ds5+ds6, 0.0035))

s.Line(point1=(ds1+ds2+ds3+ds4+ds5+ds6+y7, 0.007),
point2=(ds1+ds2+ds3+ds4+ds5+ds6+y7, 0.00575))

s.Line(point1=(ds1+ds2+ds3+ds4+ds5+ds6+ds7, 0.007),
point2=(ds1+ds2+ds3+ds4+ds5+ds6+ds7, 0.0035))

```

```

s.Line(point1=(ds1+ds2+ds3+ds4+ds5+ds6+ds7+x8, 0.007),
point2=(ds1+ds2+ds3+ds4+ds5+ds6+ds7+x8, 0.00575))

s.Line(point1=(ds1+ds2+ds3+ds4+ds5+ds6+ds7+ds8, 0.007),
point2=(ds1+ds2+ds3+ds4+ds5+ds6+ds7+ds8, 0.0035))

s.Line(point1=(ds1+ds2+ds3+ds4+ds5+ds6+ds7+ds8+y9, 0.007),
point2=(ds1+ds2+ds3+ds4+ds5+ds6+ds7+ds8+y9, 0.00575))

s.Line(point1=(ds1+ds2+ds3+ds4+ds5+ds6+ds7+ds8+ds9, 0.007),
point2=(ds1+ds2+ds3+ds4+ds5+ds6+ds7+ds8+ds9, 0.0035))

s.Line(point1=(ds1+ds2+ds3+ds4+ds5+ds6+ds7+ds8+ds9+x10, 0.007),
point2=(ds1+ds2+ds3+ds4+ds5+ds6+ds7+ds8+ds9+x10, 0.00575))

s.Line(point1=(set1, 0.007), point2=(set1, 0.0035))

```

#Top Row, Dspace 11-20

```

s.Line(point1=(set1+y11, 0.007), point2=(set1+y11, 0.00575))

s.Line(point1=(set1+ds11, 0.007), point2=(set1+ds11, 0.0035))

s.Line(point1=(set1+ds11+x12, 0.007), point2=(set1+ds11+x12, 0.00575))

s.Line(point1=(set1+ds11+ds12, 0.007), point2=(set1+ds11+ds12, 0.0035))

s.Line(point1=(set1+ds11+ds12+y13, 0.007), point2=(set1+ds11+ds12+y13,
0.00575))

s.Line(point1=(set1+ds11+ds12+ds13, 0.007), point2=(set1+ds11+ds12+ds13,
0.0035))

s.Line(point1=(set1+ds11+ds12+ds13+x14, 0.007),
point2=(set1+ds11+ds12+ds13+x14, 0.00575))

s.Line(point1=(set1+ds11+ds12+ds13+ds14, 0.007),
point2=(set1+ds11+ds12+ds13+ds14, 0.0035))

s.Line(point1=(set1+ds11+ds12+ds13+ds14+y15, 0.007),
point2=(set1+ds11+ds12+ds13+ds14+y15, 0.00575))

s.Line(point1=(set1+ds11+ds12+ds13+ds14+ds15, 0.007),
point2=(set1+ds11+ds12+ds13+ds14+ds15, 0.0035))

s.Line(point1=(set1+ds11+ds12+ds13+ds14+ds15+x16, 0.007),
point2=(set1+ds11+ds12+ds13+ds14+ds15+x16, 0.00575))

```

```

s.Line(point1=(set1+ds11+ds12+ds13+ds14+ds15+ds16, 0.007),
point2=(set1+ds11+ds12+ds13+ds14+ds15+ds16, 0.0035))

s.Line(point1=(set1+ds11+ds12+ds13+ds14+ds15+ds16+y17, 0.007),
point2=(set1+ds11+ds12+ds13+ds14+ds15+ds16+y17, 0.00575))

s.Line(point1=(set1+ds11+ds12+ds13+ds14+ds15+ds16+ds17, 0.007),
point2=(set1+ds11+ds12+ds13+ds14+ds15+ds16+ds17, 0.0035))

s.Line(point1=(set1+ds11+ds12+ds13+ds14+ds15+ds16+ds17+x18, 0.007),
point2=(set1+ds11+ds12+ds13+ds14+ds15+ds16+ds17+x18, 0.00575))

s.Line(point1=(set1+ds11+ds12+ds13+ds14+ds15+ds16+ds17+ds18, 0.007),
point2=(set1+ds11+ds12+ds13+ds14+ds15+ds16+ds17+ds18, 0.0035))

s.Line(point1=(set1+ds11+ds12+ds13+ds14+ds15+ds16+ds17+ds18+y19, 0.007),
point2=(set1+ds11+ds12+ds13+ds14+ds15+ds16+ds17+ds18+y19, 0.00575))

s.Line(point1=(set1+ds11+ds12+ds13+ds14+ds15+ds16+ds17+ds18+ds19, 0.007),
point2=(set1+ds11+ds12+ds13+ds14+ds15+ds16+ds17+ds18+ds19, 0.0035))

s.Line(point1=(set1+ds11+ds12+ds13+ds14+ds15+ds16+ds17+ds18+ds19+x20,
0.007), point2=(set1+ds11+ds12+ds13+ds14+ds15+ds16+ds17+ds18+ds19+x20,
0.00575))

```

#Top Row, Dspace 21-30

```

s.Line(point1=(set2, 0.007), point2=(set2, 0.0035))

s.Line(point1=(set2+y21, 0.007), point2=(set2+y21, 0.00575))

s.Line(point1=(set2+ds21, 0.007), point2=(set2+ds21, 0.0035))

s.Line(point1=(set2+ds21+x22, 0.007), point2=(set2+ds21+x22, 0.00575))

s.Line(point1=(set2+ds21+ds22, 0.007), point2=(set2+ds21+ds22, 0.0035))

s.Line(point1=(set2+ds21+ds22+y23, 0.007), point2=(set2+ds21+ds22+y23,
0.00575))

s.Line(point1=(set2+ds21+ds22+ds23, 0.007), point2=(set2+ds21+ds22+ds23,
0.0035))

s.Line(point1=(set2+ds21+ds22+ds23+x24, 0.007),
point2=(set2+ds21+ds22+ds23+x24, 0.00575))

s.Line(point1=(set2+ds21+ds22+ds23+ds24, 0.007),
point2=(set2+ds21+ds22+ds23+ds24, 0.0035))

```

```

s.Line(point1=(set2+ds21+ds22+ds23+ds24+y25, 0.007),
point2=(set2+ds21+ds22+ds23+ds24+y25, 0.00575))

s.Line(point1=(set2+ds21+ds22+ds23+ds24+ds25, 0.007),
point2=(set2+ds21+ds22+ds23+ds24+ds25, 0.0035))

s.Line(point1=(set2+ds21+ds22+ds23+ds24+ds25+x26, 0.007),
point2=(set2+ds21+ds22+ds23+ds24+ds25+x26, 0.00575))

s.Line(point1=(set2+ds21+ds22+ds23+ds24+ds25+ds26, 0.007),
point2=(set2+ds21+ds22+ds23+ds24+ds25+ds26, 0.0035))

s.Line(point1=(set2+ds21+ds22+ds23+ds24+ds25+ds26+y27, 0.007),
point2=(set2+ds21+ds22+ds23+ds24+ds25+ds26+y27, 0.00575))

s.Line(point1=(set2+ds21+ds22+ds23+ds24+ds25+ds26+ds27, 0.007),
point2=(set2+ds21+ds22+ds23+ds24+ds25+ds26+ds27, 0.0035))

s.Line(point1=(set2+ds21+ds22+ds23+ds24+ds25+ds26+ds27+x28, 0.007),
point2=(set2+ds21+ds22+ds23+ds24+ds25+ds26+ds27+x28, 0.00575))

s.Line(point1=(set2+ds21+ds22+ds23+ds24+ds25+ds26+ds27+ds28, 0.007),
point2=(set2+ds21+ds22+ds23+ds24+ds25+ds26+ds27+ds28, 0.0035))

s.Line(point1=(set2+ds21+ds22+ds23+ds24+ds25+ds26+ds27+ds28+y29, 0.007),
point2=(set2+ds21+ds22+ds23+ds24+ds25+ds26+ds27+ds28+y29, 0.00575))

s.Line(point1=(set2+ds21+ds22+ds23+ds24+ds25+ds26+ds27+ds28+ds29,
0.007), point2=(set2+ds21+ds22+ds23+ds24+ds25+ds26+ds27+ds28+ds29, 0.0035))

s.Line(point1=(set2+ds21+ds22+ds23+ds24+ds25+ds26+ds27+ds28+ds29+x30,
0.007), point2=(set2+ds21+ds22+ds23+ds24+ds25+ds26+ds27+ds28+ds29+x30,
0.00575))

```

#Top Row, Dspace 31-40

```

s.Line(point1=(set3, 0.007), point2=(set3, 0.0035))

s.Line(point1=(set3+y31, 0.007), point2=(set3+y31, 0.00575))

s.Line(point1=(set3+ds31, 0.007), point2=(set3+ds31, 0.0035))

s.Line(point1=(set3+ds31+x32, 0.007), point2=(set3+ds31+x32, 0.00575))

s.Line(point1=(set3+ds31+ds32, 0.007), point2=(set3+ds31+ds32, 0.0035))

s.Line(point1=(set3+ds31+ds32+y33, 0.007), point2=(set3+ds31+ds32+y33,

```


0.00575))

s.Line(point1=(set3+ds31+ds32+ds33, 0.007), point2=(set3+ds31+ds32+ds33, 0.0035))

s.Line(point1=(set3+ds31+ds32+ds33+x34, 0.007), point2=(set3+ds31+ds32+ds33+x34, 0.00575))

s.Line(point1=(set3+ds31+ds32+ds33+ds34, 0.007), point2=(set3+ds31+ds32+ds33+ds34, 0.0035))

s.Line(point1=(set3+ds31+ds32+ds33+ds34+y35, 0.007), point2=(set3+ds31+ds32+ds33+ds34+y35, 0.00575))

s.Line(point1=(set3+ds31+ds32+ds33+ds34+ds35, 0.007), point2=(set3+ds31+ds32+ds33+ds34+ds35, 0.0035))

s.Line(point1=(set3+ds31+ds32+ds33+ds34+ds35+x36, 0.007), point2=(set3+ds31+ds32+ds33+ds34+ds35+x36, 0.00575))

s.Line(point1=(set3+ds31+ds32+ds33+ds34+ds35+ds36, 0.007), point2=(set3+ds31+ds32+ds33+ds34+ds35+ds36, 0.0035))

s.Line(point1=(set3+ds31+ds32+ds33+ds34+ds35+ds36+y37, 0.007), point2=(set3+ds31+ds32+ds33+ds34+ds35+ds36+y37, 0.00575))

s.Line(point1=(set3+ds31+ds32+ds33+ds34+ds35+ds36+ds37, 0.007), point2=(set3+ds31+ds32+ds33+ds34+ds35+ds36+ds37, 0.0035))

s.Line(point1=(set3+ds31+ds32+ds33+ds34+ds35+ds36+ds37+x38, 0.007), point2=(set3+ds31+ds32+ds33+ds34+ds35+ds36+ds37+x38, 0.00575))

s.Line(point1=(set3+ds31+ds32+ds33+ds34+ds35+ds36+ds37+ds38, 0.007), point2=(set3+ds31+ds32+ds33+ds34+ds35+ds36+ds37+ds38, 0.0035))

s.Line(point1=(set3+ds31+ds32+ds33+ds34+ds35+ds36+ds37+ds38+y39, 0.007), point2=(set3+ds31+ds32+ds33+ds34+ds35+ds36+ds37+ds38+y39, 0.00575))

s.Line(point1=(set3+ds31+ds32+ds33+ds34+ds35+ds36+ds37+ds38+ds39, 0.007), point2=(set3+ds31+ds32+ds33+ds34+ds35+ds36+ds37+ds38+ds39, 0.0035))

s.Line(point1=(set3+ds31+ds32+ds33+ds34+ds35+ds36+ds37+ds38+ds39+x40, 0.007), point2=(set3+ds31+ds32+ds33+ds34+ds35+ds36+ds37+ds38+ds39+x40, 0.00575))

#Top Row, Dspace 41-50

s.Line(point1=(set4, 0.007), point2=(set4, 0.0035))

```

s.Line(point1=(set4+y41, 0.007), point2=(set4+y41, 0.00575))

s.Line(point1=(set4+ds41, 0.007), point2=(set4+ds41, 0.0035))

s.Line(point1=(set4+ds41+x42, 0.007), point2=(set4+ds41+x42, 0.00575))

s.Line(point1=(set4+ds41+ds42, 0.007), point2=(set4+ds41+ds42, 0.0035))

s.Line(point1=(set4+ds41+ds42+y43, 0.007), point2=(set4+ds41+ds42+y43,
0.00575))

s.Line(point1=(set4+ds41+ds42+ds43, 0.007), point2=(set4+ds41+ds42+ds43,
0.0035))

s.Line(point1=(set4+ds41+ds42+ds43+x44, 0.007),
point2=(set4+ds41+ds42+ds43+x44, 0.00575))

s.Line(point1=(set4+ds41+ds42+ds43+ds44, 0.007),
point2=(set4+ds41+ds42+ds43+ds44, 0.0035))

s.Line(point1=(set4+ds41+ds42+ds43+ds44+y45, 0.007),
point2=(set4+ds41+ds42+ds43+ds44+y45, 0.00575))

s.Line(point1=(set4+ds41+ds42+ds43+ds44+ds45, 0.007),
point2=(set4+ds41+ds42+ds43+ds44+ds45, 0.0035))

s.Line(point1=(set4+ds41+ds42+ds43+ds44+ds45+x46, 0.007),
point2=(set4+ds41+ds42+ds43+ds44+ds45+x46, 0.00575))

s.Line(point1=(set4+ds41+ds42+ds43+ds44+ds45+ds46, 0.007),
point2=(set4+ds41+ds42+ds43+ds44+ds45+ds46, 0.0035))

s.Line(point1=(set4+ds41+ds42+ds43+ds44+ds45+ds46+y47, 0.007),
point2=(set4+ds41+ds42+ds43+ds44+ds45+ds46+y47, 0.00575))

s.Line(point1=(set4+ds41+ds42+ds43+ds44+ds45+ds46+ds47, 0.007),
point2=(set4+ds41+ds42+ds43+ds44+ds45+ds46+ds47, 0.0035))

s.Line(point1=(set4+ds41+ds42+ds43+ds44+ds45+ds46+ds47+x48, 0.007),
point2=(set4+ds41+ds42+ds43+ds44+ds45+ds46+ds47+x48, 0.00575))

s.Line(point1=(set4+ds41+ds42+ds43+ds44+ds45+ds46+ds47+ds48, 0.007),
point2=(set4+ds41+ds42+ds43+ds44+ds45+ds46+ds47+ds48, 0.0035))

s.Line(point1=(set4+ds41+ds42+ds43+ds44+ds45+ds46+ds47+ds48+y49, 0.007),
point2=(set4+ds41+ds42+ds43+ds44+ds45+ds46+ds47+ds48+y49, 0.00575))

```

```
s.Line(point1=(set4+ds41+ds42+ds43+ds44+ds45+ds46+ds47+ds48+ds49, 0.007),  
point2=(set4+ds41+ds42+ds43+ds44+ds45+ds46+ds47+ds48+ds49, 0.0035))
```

```
s.Line(point1=(set4+ds41+ds42+ds43+ds44+ds45+ds46+ds47+ds48+ds49+x50,  
0.007), point2=(set4+ds41+ds42+ds43+ds44+ds45+ds46+ds47+ds48+ds49+x50,  
0.00575))
```

#Top Row, Dspace 51-60

```
s.Line(point1=(set5, 0.007), point2=(set5, 0.0035))
```

```
s.Line(point1=(set5+y51, 0.007), point2=(set5+y51, 0.00575))
```

```
s.Line(point1=(set5+ds51, 0.007), point2=(set5+ds51, 0.0035))
```

```
s.Line(point1=(set5+ds51+x52, 0.007), point2=(set5+ds51+x52, 0.00575))
```

```
s.Line(point1=(set5+ds51+ds52, 0.007), point2=(set5+ds51+ds52, 0.0035))
```

```
s.Line(point1=(set5+ds51+ds52+y53, 0.007), point2=(set5+ds51+ds52+y53,  
0.00575))
```

```
s.Line(point1=(set5+ds51+ds52+ds53, 0.007), point2=(set5+ds51+ds52+ds53,  
0.0035))
```

```
s.Line(point1=(set5+ds51+ds52+ds53+x54, 0.007),  
point2=(set5+ds51+ds52+ds53+x54, 0.00575))
```

```
s.Line(point1=(set5+ds51+ds52+ds53+ds54, 0.007),  
point2=(set5+ds51+ds52+ds53+ds54, 0.0035))
```

```
s.Line(point1=(set5+ds51+ds52+ds53+ds54+y55, 0.007),  
point2=(set5+ds51+ds52+ds53+ds54+y55, 0.00575))
```

```
s.Line(point1=(set5+ds51+ds52+ds53+ds54+ds55, 0.007),  
point2=(set5+ds51+ds52+ds53+ds54+ds55, 0.0035))
```

```
s.Line(point1=(set5+ds51+ds52+ds53+ds54+ds55+x56, 0.007),  
point2=(set5+ds51+ds52+ds53+ds54+ds55+x56, 0.00575))
```

```
s.Line(point1=(set5+ds51+ds52+ds53+ds54+ds55+ds56, 0.007),  
point2=(set5+ds51+ds52+ds53+ds54+ds55+ds56, 0.0035))
```

```
s.Line(point1=(set5+ds51+ds52+ds53+ds54+ds55+ds56+y57, 0.007),  
point2=(set5+ds51+ds52+ds53+ds54+ds55+ds56+y57, 0.00575))
```

```
s.Line(point1=(set5+ds51+ds52+ds53+ds54+ds55+ds56+ds57, 0.007),  
point2=(set5+ds51+ds52+ds53+ds54+ds55+ds56+ds57, 0.0035))
```

```

s.Line(point1=(set5+ds51+ds52+ds53+ds54+ds55+ds56+ds57+x58, 0.007),
point2=(set5+ds51+ds52+ds53+ds54+ds55+ds56+ds57+x58, 0.00575))

s.Line(point1=(set5+ds51+ds52+ds53+ds54+ds55+ds56+ds57+ds58, 0.007),
point2=(set5+ds51+ds52+ds53+ds54+ds55+ds56+ds57+ds58, 0.0035))

s.Line(point1=(set5+ds51+ds52+ds53+ds54+ds55+ds56+ds57+ds58+y59, 0.007),
point2=(set5+ds51+ds52+ds53+ds54+ds55+ds56+ds57+ds58+y59, 0.00575))

s.Line(point1=(set5+ds51+ds52+ds53+ds54+ds55+ds56+ds57+ds58+ds59, 0.007),
point2=(set5+ds51+ds52+ds53+ds54+ds55+ds56+ds57+ds58+ds59, 0.0035))

s.Line(point1=(set5+ds51+ds52+ds53+ds54+ds55+ds56+ds57+ds58+ds59+x60,
0.007), point2=(set5+ds51+ds52+ds53+ds54+ds55+ds56+ds57+ds58+ds59+x60,
0.00575))

```

#Top Row, Dspace 61-70

```

s.Line(point1=(set6, 0.007), point2=(set6, 0.0035))

s.Line(point1=(set6+y61, 0.007), point2=(set6+y61, 0.00575))

s.Line(point1=(set6+ds61, 0.007), point2=(set6+ds61, 0.0035))

s.Line(point1=(set6+ds61+x62, 0.007), point2=(set6+ds61+x62, 0.00575))

s.Line(point1=(set6+ds61+ds62, 0.007), point2=(set6+ds61+ds62, 0.0035))

s.Line(point1=(set6+ds61+ds62+y63, 0.007), point2=(set6+ds61+ds62+y63,
0.00575))

s.Line(point1=(set6+ds61+ds62+ds63, 0.007), point2=(set6+ds61+ds62+ds63,
0.0035))

s.Line(point1=(set6+ds61+ds62+ds63+x64, 0.007),
point2=(set6+ds61+ds62+ds63+x64, 0.00575))

s.Line(point1=(set6+ds61+ds62+ds63+ds64, 0.007),
point2=(set6+ds61+ds62+ds63+ds64, 0.0035))

s.Line(point1=(set6+ds61+ds62+ds63+ds64+y65, 0.007),
point2=(set6+ds61+ds62+ds63+ds64+y65, 0.00575))

s.Line(point1=(set6+ds61+ds62+ds63+ds64+ds65, 0.007),
point2=(set6+ds61+ds62+ds63+ds64+ds65, 0.0035))

s.Line(point1=(set6+ds61+ds62+ds63+ds64+ds65+x66, 0.007),

```

```

point2=(set6+ds61+ds62+ds63+ds64+ds65+x66, 0.00575))

    s.Line(point1=(set6+ds61+ds62+ds63+ds64+ds65+ds66, 0.007),
point2=(set6+ds61+ds62+ds63+ds64+ds65+ds66, 0.0035))

    s.Line(point1=(set6+ds61+ds62+ds63+ds64+ds65+ds66+y67, 0.007),
point2=(set6+ds61+ds62+ds63+ds64+ds65+ds66+y67, 0.00575))

    s.Line(point1=(set6+ds61+ds62+ds63+ds64+ds65+ds66+ds67, 0.007),
point2=(set6+ds61+ds62+ds63+ds64+ds65+ds66+ds67, 0.0035))

    s.Line(point1=(set6+ds61+ds62+ds63+ds64+ds65+ds66+ds67+x68, 0.007),
point2=(set6+ds61+ds62+ds63+ds64+ds65+ds66+ds67+x68, 0.00575))

    s.Line(point1=(set6+ds61+ds62+ds63+ds64+ds65+ds66+ds67+ds68, 0.007),
point2=(set6+ds61+ds62+ds63+ds64+ds65+ds66+ds67+ds68, 0.0035))

    s.Line(point1=(set6+ds61+ds62+ds63+ds64+ds65+ds66+ds67+ds68+y69, 0.007),
point2=(set6+ds61+ds62+ds63+ds64+ds65+ds66+ds67+ds68+y69, 0.00575))

    s.Line(point1=(set6+ds61+ds62+ds63+ds64+ds65+ds66+ds67+ds68+ds69, 0.007),
point2=(set6+ds61+ds62+ds63+ds64+ds65+ds66+ds67+ds68+ds69, 0.0035))

    s.Line(point1=(set6+ds61+ds62+ds63+ds64+ds65+ds66+ds67+ds68+ds69+x70,
0.007), point2=(set6+ds61+ds62+ds63+ds64+ds65+ds66+ds67+ds68+ds69+x70,
0.00575))

```

#Top Row, Dspace 71-80

```

    s.Line(point1=(set7, 0.007), point2=(set7, 0.0035))

    s.Line(point1=(set7+y71, 0.007), point2=(set7+y71, 0.00575))

    s.Line(point1=(set7+ds71, 0.007), point2=(set7+ds71, 0.0035))

    s.Line(point1=(set7+ds71+x72, 0.007), point2=(set7+ds71+x72, 0.00575))

    s.Line(point1=(set7+ds71+ds72, 0.007), point2=(set7+ds71+ds72, 0.0035))

    s.Line(point1=(set7+ds71+ds72+y73, 0.007), point2=(set7+ds71+ds72+y73,
0.00575))

    s.Line(point1=(set7+ds71+ds72+ds73, 0.007), point2=(set7+ds71+ds72+ds73,
0.0035))

    s.Line(point1=(set7+ds71+ds72+ds73+x74, 0.007),
point2=(set7+ds71+ds72+ds73+x74, 0.00575))

```

```

s.Line(point1=(set7+ds71+ds72+ds73+ds74, 0.007),
point2=(set7+ds71+ds72+ds73+ds74, 0.0035))

s.Line(point1=(set7+ds71+ds72+ds73+ds74+y75, 0.007),
point2=(set7+ds71+ds72+ds73+ds74+y75, 0.00575))

s.Line(point1=(set7+ds71+ds72+ds73+ds74+ds75, 0.007),
point2=(set7+ds71+ds72+ds73+ds74+ds75, 0.0035))

s.Line(point1=(set7+ds71+ds72+ds73+ds74+ds75+x76, 0.007),
point2=(set7+ds71+ds72+ds73+ds74+ds75+x76, 0.00575))

s.Line(point1=(set7+ds71+ds72+ds73+ds74+ds75+ds76, 0.007),
point2=(set7+ds71+ds72+ds73+ds74+ds75+ds76, 0.0035))

s.Line(point1=(set7+ds71+ds72+ds73+ds74+ds75+ds76+y77, 0.007),
point2=(set7+ds71+ds72+ds73+ds74+ds75+ds76+y77, 0.00575))

s.Line(point1=(set7+ds71+ds72+ds73+ds74+ds75+ds76+ds77, 0.007),
point2=(set7+ds71+ds72+ds73+ds74+ds75+ds76+ds77, 0.0035))

s.Line(point1=(set7+ds71+ds72+ds73+ds74+ds75+ds76+ds77+x78, 0.007),
point2=(set7+ds71+ds72+ds73+ds74+ds75+ds76+ds77+x78, 0.00575))

s.Line(point1=(set7+ds71+ds72+ds73+ds74+ds75+ds76+ds77+ds78, 0.007),
point2=(set7+ds71+ds72+ds73+ds74+ds75+ds76+ds77+ds78, 0.0035))

s.Line(point1=(set7+ds71+ds72+ds73+ds74+ds75+ds76+ds77+ds78+y79, 0.007),
point2=(set7+ds71+ds72+ds73+ds74+ds75+ds76+ds77+ds78+y79, 0.00575))

s.Line(point1=(set7+ds71+ds72+ds73+ds74+ds75+ds76+ds77+ds78+ds79, 0.007),
point2=(set7+ds71+ds72+ds73+ds74+ds75+ds76+ds77+ds78+ds79, 0.0035))

s.Line(point1=(set7+ds71+ds72+ds73+ds74+ds75+ds76+ds77+ds78+ds79+x80,
0.007), point2=(set7+ds71+ds72+ds73+ds74+ds75+ds76+ds77+ds78+ds79+x80,
0.00575))

```

#Top Row, Dspace 81-90

```

s.Line(point1=(set8, 0.007), point2=(set8, 0.0035))

s.Line(point1=(set8+y81, 0.007), point2=(set8+y81, 0.00575))

s.Line(point1=(set8+ds81, 0.007), point2=(set8+ds81, 0.0035))

s.Line(point1=(set8+ds81+x82, 0.007), point2=(set8+ds81+x82, 0.00575))

s.Line(point1=(set8+ds81+ds82, 0.007), point2=(set8+ds81+ds82, 0.0035))

```

```

s.Line(point1=(set8+ds81+ds82+y83, 0.007), point2=(set8+ds81+ds82+y83,
0.00575))

s.Line(point1=(set8+ds81+ds82+ds83, 0.007), point2=(set8+ds81+ds82+ds83,
0.0035))

s.Line(point1=(set8+ds81+ds82+ds83+x84, 0.007),
point2=(set8+ds81+ds82+ds83+x84, 0.00575))

s.Line(point1=(set8+ds81+ds82+ds83+ds84, 0.007),
point2=(set8+ds81+ds82+ds83+ds84, 0.0035))

s.Line(point1=(set8+ds81+ds82+ds83+ds84+y85, 0.007),
point2=(set8+ds81+ds82+ds83+ds84+y85, 0.00575))

s.Line(point1=(set8+ds81+ds82+ds83+ds84+ds85, 0.007),
point2=(set8+ds81+ds82+ds83+ds84+ds85, 0.0035))

s.Line(point1=(set8+ds81+ds82+ds83+ds84+ds85+x86, 0.007),
point2=(set8+ds81+ds82+ds83+ds84+ds85+x86, 0.00575))

s.Line(point1=(set8+ds81+ds82+ds83+ds84+ds85+ds86, 0.007),
point2=(set8+ds81+ds82+ds83+ds84+ds85+ds86, 0.0035))

s.Line(point1=(set8+ds81+ds82+ds83+ds84+ds85+ds86+y87, 0.007),
point2=(set8+ds81+ds82+ds83+ds84+ds85+ds86+y87, 0.00575))

s.Line(point1=(set8+ds81+ds82+ds83+ds84+ds85+ds86+ds87, 0.007),
point2=(set8+ds81+ds82+ds83+ds84+ds85+ds86+ds87, 0.0035))

s.Line(point1=(set8+ds81+ds82+ds83+ds84+ds85+ds86+ds87+x88, 0.007),
point2=(set8+ds81+ds82+ds83+ds84+ds85+ds86+ds87+x88, 0.00575))

s.Line(point1=(set8+ds81+ds82+ds83+ds84+ds85+ds86+ds87+ds88, 0.007),
point2=(set8+ds81+ds82+ds83+ds84+ds85+ds86+ds87+ds88, 0.0035))

s.Line(point1=(set8+ds81+ds82+ds83+ds84+ds85+ds86+ds87+ds88+y89, 0.007),
point2=(set8+ds81+ds82+ds83+ds84+ds85+ds86+ds87+ds88+y89, 0.00575))

s.Line(point1=(set8+ds81+ds82+ds83+ds84+ds85+ds86+ds87+ds88+ds89, 0.007),
point2=(set8+ds81+ds82+ds83+ds84+ds85+ds86+ds87+ds88+ds89, 0.0035))

s.Line(point1=(set8+ds81+ds82+ds83+ds84+ds85+ds86+ds87+ds88+ds89+x90,
0.007), point2=(set8+ds81+ds82+ds83+ds84+ds85+ds86+ds87+ds88+ds89+x90,
0.00575))

```

#Top Row, Dspace 91-100

```

s.Line(point1=(set9, 0.007), point2=(set9, 0.0035))

s.Line(point1=(set9+y91, 0.007), point2=(set9+y91, 0.00575))

s.Line(point1=(set9+ds91, 0.007), point2=(set9+ds91, 0.0035))

s.Line(point1=(set9+ds91+x92, 0.007), point2=(set9+ds91+x92, 0.00575))

s.Line(point1=(set9+ds91+ds92, 0.007), point2=(set9+ds91+ds92, 0.0035))

s.Line(point1=(set9+ds91+ds92+y93, 0.007), point2=(set9+ds91+ds92+y93,
0.00575))

s.Line(point1=(set9+ds91+ds92+ds93, 0.007), point2=(set9+ds91+ds92+ds93,
0.0035))

s.Line(point1=(set9+ds91+ds92+ds93+x94, 0.007),
point2=(set9+ds91+ds92+ds93+x94, 0.00575))

s.Line(point1=(set9+ds91+ds92+ds93+ds94, 0.007),
point2=(set9+ds91+ds92+ds93+ds94, 0.0035))

s.Line(point1=(set9+ds91+ds92+ds93+ds94+y95, 0.007),
point2=(set9+ds91+ds92+ds93+ds94+y95, 0.00575))

s.Line(point1=(set9+ds91+ds92+ds93+ds94+ds95, 0.007),
point2=(set9+ds91+ds92+ds93+ds94+ds95, 0.0035))

s.Line(point1=(set9+ds91+ds92+ds93+ds94+ds95+x96, 0.007),
point2=(set9+ds91+ds92+ds93+ds94+ds95+x96, 0.00575))

s.Line(point1=(set9+ds91+ds92+ds93+ds94+ds95+ds96, 0.007),
point2=(set9+ds91+ds92+ds93+ds94+ds95+ds96, 0.0035))

s.Line(point1=(set9+ds91+ds92+ds93+ds94+ds95+ds96+y97, 0.007),
point2=(set9+ds91+ds92+ds93+ds94+ds95+ds96+y97, 0.00575))

s.Line(point1=(set9+ds91+ds92+ds93+ds94+ds95+ds96+ds97, 0.007),
point2=(set9+ds91+ds92+ds93+ds94+ds95+ds96+ds97, 0.0035))

s.Line(point1=(set9+ds91+ds92+ds93+ds94+ds95+ds96+ds97+x98, 0.007),
point2=(set9+ds91+ds92+ds93+ds94+ds95+ds96+ds97+x98, 0.00575))

s.Line(point1=(set9+ds91+ds92+ds93+ds94+ds95+ds96+ds97+ds98, 0.007),
point2=(set9+ds91+ds92+ds93+ds94+ds95+ds96+ds97+ds98, 0.0035))

s.Line(point1=(set9+ds91+ds92+ds93+ds94+ds95+ds96+ds97+ds98+y99, 0.007),

```



```

point2=(set9+ds91+ds92+ds93+ds94+ds95+ds96+ds97+ds98+y99, 0.00575))

s.Line(point1=(set9+ds91+ds92+ds93+ds94+ds95+ds96+ds97+ds98+ds99, 0.007),
point2=(set9+ds91+ds92+ds93+ds94+ds95+ds96+ds97+ds98+ds99, 0.0035))

s.Line(point1=(set9+ds91+ds92+ds93+ds94+ds95+ds96+ds97+ds98+ds99+x100,
0.007), point2=(set9+ds91+ds92+ds93+ds94+ds95+ds96+ds97+ds98+ds99+x100,
0.00575))

```

#Bottom Row, Dspace 1-10

```

s.Line(point1=(x101, 0.0), point2=(x101, 0.00125))

s.Line(point1=(ds101, 0.0), point2=(ds101, 0.0035))

s.Line(point1=(ds101+y102, 0), point2=(ds101+y102, 0.00125))

s.Line(point1=(ds101+ds102, 0.0), point2=(ds101+ds102, 0.0035))

s.Line(point1=(ds101+ds102+x103, 0.0), point2=(ds101+ds102+x103,
0.00125))

s.Line(point1=(ds101+ds102+ds103, 0.0), point2=(ds101+ds102+ds103,
0.0035))

s.Line(point1=(ds101+ds102+ds103+y104, 0.0),
point2=(ds101+ds102+ds103+y104, 0.00125))

s.Line(point1=(ds101+ds102+ds103+ds104, 0.0),
point2=(ds101+ds102+ds103+ds104, 0.0035))

s.Line(point1=(ds101+ds102+ds103+ds104+x105, 0.0),
point2=(ds101+ds102+ds103+ds104+x105, 0.00125))

s.Line(point1=(ds101+ds102+ds103+ds104+ds105, 0.0),
point2=(ds101+ds102+ds103+ds104+ds105, 0.0035))

s.Line(point1=(ds101+ds102+ds103+ds104+ds105+y106, 0.0),
point2=(ds101+ds102+ds103+ds104+ds105+y106, 0.00125))

s.Line(point1=(ds101+ds102+ds103+ds104+ds105+ds106, 0.0),
point2=(ds101+ds102+ds103+ds104+ds105+ds106, 0.0035))

s.Line(point1=(ds101+ds102+ds103+ds104+ds105+ds106+x107, 0.0),
point2=(ds101+ds102+ds103+ds104+ds105+ds106+x107, 0.00125))

s.Line(point1=(ds101+ds102+ds103+ds104+ds105+ds106+ds107, 0.0),
point2=(ds101+ds102+ds103+ds104+ds105+ds106+ds107, 0.0035))

```

```

        s.Line(point1=(ds101+ds102+ds103+ds104+ds105+ds106+ds107+y108, 0.0),
point2=(ds101+ds102+ds103+ds104+ds105+ds106+ds107+y108, 0.00125))

        s.Line(point1=(ds101+ds102+ds103+ds104+ds105+ds106+ds107+ds108, 0.0),
point2=(ds101+ds102+ds103+ds104+ds105+ds106+ds107+ds108, 0.0035))

        s.Line(point1=(ds101+ds102+ds103+ds104+ds105+ds106+ds107+ds108+x109,
0.0), point2=(ds101+ds102+ds103+ds104+ds105+ds106+ds107+ds108+x109, 0.00125))

        s.Line(point1=(ds101+ds102+ds103+ds104+ds105+ds106+ds107+ds108+ds109,
0.0), point2=(ds101+ds102+ds103+ds104+ds105+ds106+ds107+ds108+ds109, 0.0035))

s.Line(point1=(ds101+ds102+ds103+ds104+ds105+ds106+ds107+ds108+ds109+y110,
0.0), point2=(ds101+ds102+ds103+ds104+ds105+ds106+ds107+ds108+ds109+y110,
0.00125))

        s.Line(point1=(set11, 0.0), point2=(set11, 0.0035))

#Bottom Row, Dspace 11-20

        s.Line(point1=(set11+x111, 0.0), point2=(set11+x111, 0.00125))

        s.Line(point1=(set11+ds111, 0.0), point2=(set11+ds111, 0.0035))

        s.Line(point1=(set11+ds111+y112, 0), point2=(set11+ds111+y112, 0.00125))

        s.Line(point1=(set11+ds111+ds112, 0.0), point2=(set11+ds111+ds112,
0.0035))

        s.Line(point1=(set11+ds111+ds112+x113, 0.0),
point2=(set11+ds111+ds112+x113, 0.00125))

        s.Line(point1=(set11+ds111+ds112+ds113, 0.0),
point2=(set11+ds111+ds112+ds113, 0.0035))

        s.Line(point1=(set11+ds111+ds112+ds113+y114, 0.0),
point2=(set11+ds111+ds112+ds113+y114, 0.00125))

        s.Line(point1=(set11+ds111+ds112+ds113+ds114, 0.0),
point2=(set11+ds111+ds112+ds113+ds114, 0.0035))

        s.Line(point1=(set11+ds111+ds112+ds113+ds114+x115, 0.0),
point2=(set11+ds111+ds112+ds113+ds114+x115, 0.00125))

        s.Line(point1=(set11+ds111+ds112+ds113+ds114+ds115, 0.0),
point2=(set11+ds111+ds112+ds113+ds114+ds115, 0.0035))

```

```

        s.Line(point1=(set11+ds111+ds112+ds113+ds114+ds115+y116, 0.0),
point2=(set11+ds111+ds112+ds113+ds114+ds115+y116, 0.00125))

        s.Line(point1=(set11+ds111+ds112+ds113+ds114+ds115+ds116, 0.0),
point2=(set11+ds111+ds112+ds113+ds114+ds115+ds116, 0.0035))

        s.Line(point1=(set11+ds111+ds112+ds113+ds114+ds115+ds116+x117, 0.0),
point2=(set11+ds111+ds112+ds113+ds114+ds115+ds116+x117, 0.00125))

        s.Line(point1=(set11+ds111+ds112+ds113+ds114+ds115+ds116+ds117, 0.0),
point2=(set11+ds111+ds112+ds113+ds114+ds115+ds116+ds117, 0.0035))

        s.Line(point1=(set11+ds111+ds112+ds113+ds114+ds115+ds116+ds117+y118,
0.0), point2=(set11+ds111+ds112+ds113+ds114+ds115+ds116+ds117+y118, 0.00125))

        s.Line(point1=(set11+ds111+ds112+ds113+ds114+ds115+ds116+ds117+ds118,
0.0), point2=(set11+ds111+ds112+ds113+ds114+ds115+ds116+ds117+ds118, 0.0035))

s.Line(point1=(set11+ds111+ds112+ds113+ds114+ds115+ds116+ds117+ds118+x119,
0.0), point2=(set11+ds111+ds112+ds113+ds114+ds115+ds116+ds117+ds118+x119,
0.00125))

s.Line(point1=(set11+ds111+ds112+ds113+ds114+ds115+ds116+ds117+ds118+ds119,
0.0), point2=(set11+ds111+ds112+ds113+ds114+ds115+ds116+ds117+ds118+ds119,
0.0035))

s.Line(point1=(set11+ds111+ds112+ds113+ds114+ds115+ds116+ds117+ds118+ds119+y120
, 0.0),
point2=(set11+ds111+ds112+ds113+ds114+ds115+ds116+ds117+ds118+ds119+y120,
0.00125))

        s.Line(point1=(set12, 0.0), point2=(set12, 0.0035))

#Bottom Row, Dspace 21-30

        s.Line(point1=(set12+x121, 0.0), point2=(set12+x121, 0.00125))

        s.Line(point1=(set12+ds121, 0.0), point2=(set12+ds121, 0.0035))

        s.Line(point1=(set12+ds121+y122, 0), point2=(set12+ds121+y122, 0.00125))

        s.Line(point1=(set12+ds121+ds122, 0.0), point2=(set12+ds121+ds122,
0.0035))

```

```

        s.Line(point1=(set12+ds121+ds122+x123, 0.0),
point2=(set12+ds121+ds122+x123, 0.00125))

        s.Line(point1=(set12+ds121+ds122+ds123, 0.0),
point2=(set12+ds121+ds122+ds123, 0.0035))

        s.Line(point1=(set12+ds121+ds122+ds123+y124, 0.0),
point2=(set12+ds121+ds122+ds123+y124, 0.00125))

        s.Line(point1=(set12+ds121+ds122+ds123+ds124, 0.0),
point2=(set12+ds121+ds122+ds123+ds124, 0.0035))

        s.Line(point1=(set12+ds121+ds122+ds123+ds124+x125, 0.0),
point2=(set12+ds121+ds122+ds123+ds124+x125, 0.00125))

        s.Line(point1=(set12+ds121+ds122+ds123+ds124+ds125, 0.0),
point2=(set12+ds121+ds122+ds123+ds124+ds125, 0.0035))

        s.Line(point1=(set12+ds121+ds122+ds123+ds124+ds125+y126, 0.0),
point2=(set12+ds121+ds122+ds123+ds124+ds125+y126, 0.00125))

        s.Line(point1=(set12+ds121+ds122+ds123+ds124+ds125+ds126, 0.0),
point2=(set12+ds121+ds122+ds123+ds124+ds125+ds126, 0.0035))

        s.Line(point1=(set12+ds121+ds122+ds123+ds124+ds125+ds126+x127, 0.0),
point2=(set12+ds121+ds122+ds123+ds124+ds125+ds126+x127, 0.00125))

        s.Line(point1=(set12+ds121+ds122+ds123+ds124+ds125+ds126+ds127, 0.0),
point2=(set12+ds121+ds122+ds123+ds124+ds125+ds126+ds127, 0.0035))

        s.Line(point1=(set12+ds121+ds122+ds123+ds124+ds125+ds126+ds127+y128,
0.0), point2=(set12+ds121+ds122+ds123+ds124+ds125+ds126+ds127+y128, 0.00125))

        s.Line(point1=(set12+ds121+ds122+ds123+ds124+ds125+ds126+ds127+ds128,
0.0), point2=(set12+ds121+ds122+ds123+ds124+ds125+ds126+ds127+ds128, 0.0035))

s.Line(point1=(set12+ds121+ds122+ds123+ds124+ds125+ds126+ds127+ds128+x129,
0.0), point2=(set12+ds121+ds122+ds123+ds124+ds125+ds126+ds127+ds128+x129,
0.00125))

s.Line(point1=(set12+ds121+ds122+ds123+ds124+ds125+ds126+ds127+ds128+ds129,
0.0), point2=(set12+ds121+ds122+ds123+ds124+ds125+ds126+ds127+ds128+ds129,
0.0035))

s.Line(point1=(set12+ds121+ds122+ds123+ds124+ds125+ds126+ds127+ds128+ds129+y130

```

```
, 0.0),
point2=(set12+ds121+ds122+ds123+ds124+ds125+ds126+ds127+ds128+ds129+y130,
0.00125))

s.Line(point1=(set13, 0.0), point2=(set13, 0.0035))

#Bottom Row, Dspace 31-40

s.Line(point1=(set13+x131, 0.0), point2=(set13+x131, 0.00125))

s.Line(point1=(set13+ds131, 0.0), point2=(set13+ds131, 0.0035))

s.Line(point1=(set13+ds131+y132, 0), point2=(set13+ds131+y132, 0.00125))

s.Line(point1=(set13+ds131+ds132, 0.0), point2=(set13+ds131+ds132,
0.0035))

s.Line(point1=(set13+ds131+ds132+x133, 0.0),
point2=(set13+ds131+ds132+x133, 0.00125))

s.Line(point1=(set13+ds131+ds132+ds133, 0.0),
point2=(set13+ds131+ds132+ds133, 0.0035))

s.Line(point1=(set13+ds131+ds132+ds133+y134, 0.0),
point2=(set13+ds131+ds132+ds133+y134, 0.00125))

s.Line(point1=(set13+ds131+ds132+ds133+ds134, 0.0),
point2=(set13+ds131+ds132+ds133+ds134, 0.0035))

s.Line(point1=(set13+ds131+ds132+ds133+ds134+x135, 0.0),
point2=(set13+ds131+ds132+ds133+ds134+x135, 0.00125))

s.Line(point1=(set13+ds131+ds132+ds133+ds134+ds135, 0.0),
point2=(set13+ds131+ds132+ds133+ds134+ds135, 0.0035))

s.Line(point1=(set13+ds131+ds132+ds133+ds134+ds135+y136, 0.0),
point2=(set13+ds131+ds132+ds133+ds134+ds135+y136, 0.00125))

s.Line(point1=(set13+ds131+ds132+ds133+ds134+ds135+ds136, 0.0),
point2=(set13+ds131+ds132+ds133+ds134+ds135+ds136, 0.0035))

s.Line(point1=(set13+ds131+ds132+ds133+ds134+ds135+ds136+x137, 0.0),
point2=(set13+ds131+ds132+ds133+ds134+ds135+ds136+x137, 0.00125))

s.Line(point1=(set13+ds131+ds132+ds133+ds134+ds135+ds136+ds137, 0.0),
point2=(set13+ds131+ds132+ds133+ds134+ds135+ds136+ds137, 0.0035))

s.Line(point1=(set13+ds131+ds132+ds133+ds134+ds135+ds136+ds137+y138,
```

```
0.0), point2=(set13+ds131+ds132+ds133+ds134+ds135+ds136+ds137+y138, 0.00125))
```

```
s.Line(point1=(set13+ds131+ds132+ds133+ds134+ds135+ds136+ds137+ds138,  
0.0), point2=(set13+ds131+ds132+ds133+ds134+ds135+ds136+ds137+ds138, 0.0035))
```

```
s.Line(point1=(set13+ds131+ds132+ds133+ds134+ds135+ds136+ds137+ds138+x139,  
0.0), point2=(set13+ds131+ds132+ds133+ds134+ds135+ds136+ds137+ds138+x139,  
0.00125))
```

```
s.Line(point1=(set13+ds131+ds132+ds133+ds134+ds135+ds136+ds137+ds138+ds139,  
0.0), point2=(set13+ds131+ds132+ds133+ds134+ds135+ds136+ds137+ds138+ds139,  
0.0035))
```

```
s.Line(point1=(set13+ds131+ds132+ds133+ds134+ds135+ds136+ds137+ds138+ds139+y140  
, 0.0),  
point2=(set13+ds131+ds132+ds133+ds134+ds135+ds136+ds137+ds138+ds139+y140,  
0.00125))
```

```
s.Line(point1=(set14, 0.0), point2=(set14, 0.0035))
```

```
#Bottom Row, Dspace 41-50
```

```
s.Line(point1=(set14+x141, 0.0), point2=(set14+x141, 0.00125))
```

```
s.Line(point1=(set14+ds141, 0.0), point2=(set14+ds141, 0.0035))
```

```
s.Line(point1=(set14+ds141+y142, 0), point2=(set14+ds141+y142, 0.00125))
```

```
s.Line(point1=(set14+ds141+ds142, 0.0), point2=(set14+ds141+ds142,  
0.0035))
```

```
s.Line(point1=(set14+ds141+ds142+x143, 0.0),  
point2=(set14+ds141+ds142+x143, 0.00125))
```

```
s.Line(point1=(set14+ds141+ds142+ds143, 0.0),  
point2=(set14+ds141+ds142+ds143, 0.0035))
```

```
s.Line(point1=(set14+ds141+ds142+ds143+y144, 0.0),  
point2=(set14+ds141+ds142+ds143+y144, 0.00125))
```

```
s.Line(point1=(set14+ds141+ds142+ds143+ds144, 0.0),  
point2=(set14+ds141+ds142+ds143+ds144, 0.0035))
```

```
s.Line(point1=(set14+ds141+ds142+ds143+ds144+x145, 0.0),  
point2=(set14+ds141+ds142+ds143+ds144+x145, 0.00125))
```

```

        s.Line(point1=(set14+ds141+ds142+ds143+ds144+ds145, 0.0),
point2=(set14+ds141+ds142+ds143+ds144+ds145, 0.0035))

        s.Line(point1=(set14+ds141+ds142+ds143+ds144+ds145+y146, 0.0),
point2=(set14+ds141+ds142+ds143+ds144+ds145+y146, 0.00125))

        s.Line(point1=(set14+ds141+ds142+ds143+ds144+ds145+ds146, 0.0),
point2=(set14+ds141+ds142+ds143+ds144+ds145+ds146, 0.0035))

        s.Line(point1=(set14+ds141+ds142+ds143+ds144+ds145+ds146+x147, 0.0),
point2=(set14+ds141+ds142+ds143+ds144+ds145+ds146+x147, 0.00125))

        s.Line(point1=(set14+ds141+ds142+ds143+ds144+ds145+ds146+ds147, 0.0),
point2=(set14+ds141+ds142+ds143+ds144+ds145+ds146+ds147, 0.0035))

        s.Line(point1=(set14+ds141+ds142+ds143+ds144+ds145+ds146+ds147+y148,
0.0), point2=(set14+ds141+ds142+ds143+ds144+ds145+ds146+ds147+y148, 0.00125))

        s.Line(point1=(set14+ds141+ds142+ds143+ds144+ds145+ds146+ds147+ds148,
0.0), point2=(set14+ds141+ds142+ds143+ds144+ds145+ds146+ds147+ds148, 0.0035))

s.Line(point1=(set14+ds141+ds142+ds143+ds144+ds145+ds146+ds147+ds148+x149,
0.0), point2=(set14+ds141+ds142+ds143+ds144+ds145+ds146+ds147+ds148+x149,
0.00125))

s.Line(point1=(set14+ds141+ds142+ds143+ds144+ds145+ds146+ds147+ds148+ds149,
0.0), point2=(set14+ds141+ds142+ds143+ds144+ds145+ds146+ds147+ds148+ds149,
0.0035))

s.Line(point1=(set14+ds141+ds142+ds143+ds144+ds145+ds146+ds147+ds148+ds149+y150
, 0.0),
point2=(set14+ds141+ds142+ds143+ds144+ds145+ds146+ds147+ds148+ds149+y150,
0.00125))

        s.Line(point1=(set15, 0.0), point2=(set15, 0.0035))

#Bottom Row, Dspace 51-60

        s.Line(point1=(set15+x151, 0.0), point2=(set15+x151, 0.00125))

        s.Line(point1=(set15+ds151, 0.0), point2=(set15+ds151, 0.0035))

        s.Line(point1=(set15+ds151+y152, 0), point2=(set15+ds151+y152, 0.00125))

```

```

s.Line(point1=(set15+ds151+ds152, 0.0), point2=(set15+ds151+ds152,
0.0035))

s.Line(point1=(set15+ds151+ds152+x153, 0.0),
point2=(set15+ds151+ds152+x153, 0.00125))

s.Line(point1=(set15+ds151+ds152+ds153, 0.0),
point2=(set15+ds151+ds152+ds153, 0.0035))

s.Line(point1=(set15+ds151+ds152+ds153+y154, 0.0),
point2=(set15+ds151+ds152+ds153+y154, 0.00125))

s.Line(point1=(set15+ds151+ds152+ds153+ds154, 0.0),
point2=(set15+ds151+ds152+ds153+ds154, 0.0035))

s.Line(point1=(set15+ds151+ds152+ds153+ds154+x155, 0.0),
point2=(set15+ds151+ds152+ds153+ds154+x155, 0.00125))

s.Line(point1=(set15+ds151+ds152+ds153+ds154+ds155, 0.0),
point2=(set15+ds151+ds152+ds153+ds154+ds155, 0.0035))

s.Line(point1=(set15+ds151+ds152+ds153+ds154+ds155+y156, 0.0),
point2=(set15+ds151+ds152+ds153+ds154+ds155+y156, 0.00125))

s.Line(point1=(set15+ds151+ds152+ds153+ds154+ds155+ds156, 0.0),
point2=(set15+ds151+ds152+ds153+ds154+ds155+ds156, 0.0035))

s.Line(point1=(set15+ds151+ds152+ds153+ds154+ds155+ds156+x157, 0.0),
point2=(set15+ds151+ds152+ds153+ds154+ds155+ds156+x157, 0.00125))

s.Line(point1=(set15+ds151+ds152+ds153+ds154+ds155+ds156+ds157, 0.0),
point2=(set15+ds151+ds152+ds153+ds154+ds155+ds156+ds157, 0.0035))

s.Line(point1=(set15+ds151+ds152+ds153+ds154+ds155+ds156+ds157+y158,
0.0), point2=(set15+ds151+ds152+ds153+ds154+ds155+ds156+ds157+y158, 0.00125))

s.Line(point1=(set15+ds151+ds152+ds153+ds154+ds155+ds156+ds157+ds158,
0.0), point2=(set15+ds151+ds152+ds153+ds154+ds155+ds156+ds157+ds158, 0.0035))

s.Line(point1=(set15+ds151+ds152+ds153+ds154+ds155+ds156+ds157+ds158+x159,
0.0), point2=(set15+ds151+ds152+ds153+ds154+ds155+ds156+ds157+ds158+x159,
0.00125))

s.Line(point1=(set15+ds151+ds152+ds153+ds154+ds155+ds156+ds157+ds158+ds159,
0.0), point2=(set15+ds151+ds152+ds153+ds154+ds155+ds156+ds157+ds158+ds159,
0.0035))

```



```
s.Line(point1=(set15+ds151+ds152+ds153+ds154+ds155+ds156+ds157+ds158+ds159+y160, 0.0),
point2=(set15+ds151+ds152+ds153+ds154+ds155+ds156+ds157+ds158+ds159+y160, 0.00125))
```

```
s.Line(point1=(set16, 0.0), point2=(set16, 0.0035))
```

#Bottom Row, Dspace 61–70

```
s.Line(point1=(set16+x161, 0.0), point2=(set16+x161, 0.00125))
```

```
s.Line(point1=(set16+ds161, 0.0), point2=(set16+ds161, 0.0035))
```

```
s.Line(point1=(set16+ds161+y162, 0), point2=(set16+ds161+y162, 0.00125))
```

```
s.Line(point1=(set16+ds161+ds162, 0.0), point2=(set16+ds161+ds162, 0.0035))
```

```
s.Line(point1=(set16+ds161+ds162+x163, 0.0),
point2=(set16+ds161+ds162+x163, 0.00125))
```

```
s.Line(point1=(set16+ds161+ds162+ds163, 0.0),
point2=(set16+ds161+ds162+ds163, 0.0035))
```

```
s.Line(point1=(set16+ds161+ds162+ds163+y164, 0.0),
point2=(set16+ds161+ds162+ds163+y164, 0.00125))
```

```
s.Line(point1=(set16+ds161+ds162+ds163+ds164, 0.0),
point2=(set16+ds161+ds162+ds163+ds164, 0.0035))
```

```
s.Line(point1=(set16+ds161+ds162+ds163+ds164+x165, 0.0),
point2=(set16+ds161+ds162+ds163+ds164+x165, 0.00125))
```

```
s.Line(point1=(set16+ds161+ds162+ds163+ds164+ds165, 0.0),
point2=(set16+ds161+ds162+ds163+ds164+ds165, 0.0035))
```

```
s.Line(point1=(set16+ds161+ds162+ds163+ds164+ds165+y166, 0.0),
point2=(set16+ds161+ds162+ds163+ds164+ds165+y166, 0.00125))
```

```
s.Line(point1=(set16+ds161+ds162+ds163+ds164+ds165+ds166, 0.0),
point2=(set16+ds161+ds162+ds163+ds164+ds165+ds166, 0.0035))
```

```
s.Line(point1=(set16+ds161+ds162+ds163+ds164+ds165+ds166+x167, 0.0),
point2=(set16+ds161+ds162+ds163+ds164+ds165+ds166+x167, 0.00125))
```

```
s.Line(point1=(set16+ds161+ds162+ds163+ds164+ds165+ds166+ds167, 0.0),
```

```

point2=(set16+ds161+ds162+ds163+ds164+ds165+ds166+ds167, 0.0035))

    s.Line(point1=(set16+ds161+ds162+ds163+ds164+ds165+ds166+ds167+y168,0.0),
point2=(set16+ds161+ds162+ds163+ds164+ds165+ds166+ds167+y168, 0.00125))

    s.Line(point1=(set16+ds161+ds162+ds163+ds164+ds165+ds166+ds167+ds168,
0.0), point2=(set16+ds161+ds162+ds163+ds164+ds165+ds166+ds167+ds168, 0.0035))

s.Line(point1=(set16+ds161+ds162+ds163+ds164+ds165+ds166+ds167+ds168+x169,
0.0), point2=(set16+ds161+ds162+ds163+ds164+ds165+ds166+ds167+ds168+x169,
0.00125))

s.Line(point1=(set16+ds161+ds162+ds163+ds164+ds165+ds166+ds167+ds168+ds169,
0.0), point2=(set16+ds161+ds162+ds163+ds164+ds165+ds166+ds167+ds168+ds169,
0.0035))

s.Line(point1=(set16+ds161+ds162+ds163+ds164+ds165+ds166+ds167+ds168+ds169+y170
, 0.0),
point2=(set16+ds161+ds162+ds163+ds164+ds165+ds166+ds167+ds168+ds169+y170,
0.00125))

    s.Line(point1=(set17, 0.0), point2=(set17, 0.0035))

#Bottom Row, Dspace 71-80

    s.Line(point1=(set17+x171, 0.0), point2=(set17+x171, 0.00125))

    s.Line(point1=(set17+ds171, 0.0), point2=(set17+ds171, 0.0035))

    s.Line(point1=(set17+ds171+y172, 0), point2=(set17+ds171+y172, 0.00125))

    s.Line(point1=(set17+ds171+ds172, 0.0), point2=(set17+ds171+ds172,
0.0035))

    s.Line(point1=(set17+ds171+ds172+x173, 0.0),
point2=(set17+ds171+ds172+x173, 0.00125))

    s.Line(point1=(set17+ds171+ds172+ds173, 0.0),
point2=(set17+ds171+ds172+ds173, 0.0035))

    s.Line(point1=(set17+ds171+ds172+ds173+y174, 0.0),
point2=(set17+ds171+ds172+ds173+y174, 0.00125))

    s.Line(point1=(set17+ds171+ds172+ds173+ds174, 0.0),
point2=(set17+ds171+ds172+ds173+ds174, 0.0035))

```

```

        s.Line(point1=(set17+ds171+ds172+ds173+ds174+x175, 0.0),
point2=(set17+ds171+ds172+ds173+ds174+x175, 0.00125))

        s.Line(point1=(set17+ds171+ds172+ds173+ds174+ds175,
0.0),point2=(set17+ds171+ds172+ds173+ds174+ds175, 0.0035))

        s.Line(point1=(set17+ds171+ds172+ds173+ds174+ds175+y176, 0.0),
point2=(set17+ds171+ds172+ds173+ds174+ds175+y176, 0.00125))

        s.Line(point1=(set17+ds171+ds172+ds173+ds174+ds175+ds176, 0.0),
point2=(set17+ds171+ds172+ds173+ds174+ds175+ds176, 0.0035))

        s.Line(point1=(set17+ds171+ds172+ds173+ds174+ds175+ds176+x177, 0.0),
point2=(set17+ds171+ds172+ds173+ds174+ds175+ds176+x177, 0.00125))

        s.Line(point1=(set17+ds171+ds172+ds173+ds174+ds175+ds176+ds177, 0.0),
point2=(set17+ds171+ds172+ds173+ds174+ds175+ds176+ds177, 0.0035))

        s.Line(point1=(set17+ds171+ds172+ds173+ds174+ds175+ds176+ds177+y178,
0.0), point2=(set17+ds171+ds172+ds173+ds174+ds175+ds176+ds177+y178, 0.00125))

        s.Line(point1=(set17+ds171+ds172+ds173+ds174+ds175+ds176+ds177+ds178,
0.0), point2=(set17+ds171+ds172+ds173+ds174+ds175+ds176+ds177+ds178, 0.0035))

s.Line(point1=(set17+ds171+ds172+ds173+ds174+ds175+ds176+ds177+ds178+x179,
0.0), point2=(set17+ds171+ds172+ds173+ds174+ds175+ds176+ds177+ds178+x179,
0.00125))

s.Line(point1=(set17+ds171+ds172+ds173+ds174+ds175+ds176+ds177+ds178+ds179,
0.0), point2=(set17+ds171+ds172+ds173+ds174+ds175+ds176+ds177+ds178+ds179,
0.0035))

s.Line(point1=(set17+ds171+ds172+ds173+ds174+ds175+ds176+ds177+ds178+ds179+y180
, 0.0),
point2=(set17+ds171+ds172+ds173+ds174+ds175+ds176+ds177+ds178+ds179+y180,
0.00125))

        s.Line(point1=(set18, 0.0), point2=(set18, 0.0035))

#Bottom Row, Dspace 81-90

        s.Line(point1=(set18+x181, 0.0), point2=(set18+x181, 0.00125))

        s.Line(point1=(set18+ds181, 0.0), point2=(set18+ds181, 0.0035))

```

```

        s.Line(point1=(set18+ds181+y182, 0.0), point2=(set18+ds181+y182,
0.00125))

        s.Line(point1=(set18+ds181+ds182, 0.0), point2=(set18+ds181+ds182,
0.0035))

        s.Line(point1=(set18+ds181+ds182+x183, 0.0),
point2=(set18+ds181+ds182+x183, 0.00125))

        s.Line(point1=(set18+ds181+ds182+ds183, 0.0),
point2=(set18+ds181+ds182+ds183, 0.0035))

        s.Line(point1=(set18+ds181+ds182+ds183+y184, 0.0),
point2=(set18+ds181+ds182+ds183+y184, 0.00125))

        s.Line(point1=(set18+ds181+ds182+ds183+ds184, 0.0),
point2=(set18+ds181+ds182+ds183+ds184, 0.0035))

        s.Line(point1=(set18+ds181+ds182+ds183+ds184+x185, 0.0),
point2=(set18+ds181+ds182+ds183+ds184+x185, 0.00125))

        s.Line(point1=(set18+ds181+ds182+ds183+ds184+ds185, 0.0),
point2=(set18+ds181+ds182+ds183+ds184+ds185, 0.0035))

        s.Line(point1=(set18+ds181+ds182+ds183+ds184+ds185+y186, 0.0),
point2=(set18+ds181+ds182+ds183+ds184+ds185+y186, 0.00125))

        s.Line(point1=(set18+ds181+ds182+ds183+ds184+ds185+ds186, 0.0),
point2=(set18+ds181+ds182+ds183+ds184+ds185+ds186, 0.0035))

        s.Line(point1=(set18+ds181+ds182+ds183+ds184+ds185+ds186+x187, 0.0),
point2=(set18+ds181+ds182+ds183+ds184+ds185+ds186+x187, 0.00125))

        s.Line(point1=(set18+ds181+ds182+ds183+ds184+ds185+ds186+ds187, 0.0),
point2=(set18+ds181+ds182+ds183+ds184+ds185+ds186+ds187, 0.0035))

        s.Line(point1=(set18+ds181+ds182+ds183+ds184+ds185+ds186+ds187+y188,
0.0), point2=(set18+ds181+ds182+ds183+ds184+ds185+ds186+ds187+y188, 0.00125))

        s.Line(point1=(set18+ds181+ds182+ds183+ds184+ds185+ds186+ds187+ds188,
0.0), point2=(set18+ds181+ds182+ds183+ds184+ds185+ds186+ds187+ds188, 0.0035))

s.Line(point1=(set18+ds181+ds182+ds183+ds184+ds185+ds186+ds187+ds188+x189,
0.0), point2=(set18+ds181+ds182+ds183+ds184+ds185+ds186+ds187+ds188+x189,
0.00125))

```

```
s.Line(point1=(set18+ds181+ds182+ds183+ds184+ds185+ds186+ds187+ds188+ds189,  
0.0), point2=(set18+ds181+ds182+ds183+ds184+ds185+ds186+ds187+ds188+ds189,  
0.0035))
```

```
s.Line(point1=(set18+ds181+ds182+ds183+ds184+ds185+ds186+ds187+ds188+ds189+y190,  
0.0),  
point2=(set18+ds181+ds182+ds183+ds184+ds185+ds186+ds187+ds188+ds189+y190,  
0.00125))
```

```
s.Line(point1=(set19, 0.0), point2=(set19, 0.0035))
```

```
#Bottom Row, Dspace 91-100
```

```
s.Line(point1=(set19+x191, 0.0), point2=(set19+x191, 0.00125))
```

```
s.Line(point1=(set19+ds191, 0.0), point2=(set19+ds191, 0.0035))
```

```
s.Line(point1=(set19+ds191+y192, 0), point2=(set19+ds191+y192, 0.00125))
```

```
s.Line(point1=(set19+ds191+ds192, 0.0), point2=(set19+ds191+ds192,  
0.0035))
```

```
s.Line(point1=(set19+ds191+ds192+x193, 0.0),  
point2=(set19+ds191+ds192+x193, 0.00125))
```

```
s.Line(point1=(set19+ds191+ds192+ds193, 0.0),  
point2=(set19+ds191+ds192+ds193, 0.0035))
```

```
s.Line(point1=(set19+ds191+ds192+ds193+y194, 0.0),  
point2=(set19+ds191+ds192+ds193+y194, 0.00125))
```

```
s.Line(point1=(set19+ds191+ds192+ds193+ds194, 0.0),  
point2=(set19+ds191+ds192+ds193+ds194, 0.0035))
```

```
s.Line(point1=(set19+ds191+ds192+ds193+ds194+x195, 0.0),  
point2=(set19+ds191+ds192+ds193+ds194+x195, 0.00125))
```

```
s.Line(point1=(set19+ds191+ds192+ds193+ds194+ds195, 0.0),  
point2=(set19+ds191+ds192+ds193+ds194+ds195, 0.0035))
```

```
s.Line(point1=(set19+ds191+ds192+ds193+ds194+ds195+y196, 0.0),  
point2=(set19+ds191+ds192+ds193+ds194+ds195+y196, 0.00125))
```

```
s.Line(point1=(set19+ds191+ds192+ds193+ds194+ds195+ds196, 0.0),  
point2=(set19+ds191+ds192+ds193+ds194+ds195+ds196, 0.0035))
```

```

s.Line(point1=(set19+ds191+ds192+ds193+ds194+ds195+ds196+x197, 0.0),
point2=(set19+ds191+ds192+ds193+ds194+ds195+ds196+x197, 0.00125))

s.Line(point1=(set19+ds191+ds192+ds193+ds194+ds195+ds196+ds197, 0.0),
point2=(set19+ds191+ds192+ds193+ds194+ds195+ds196+ds197, 0.0035))

s.Line(point1=(set19+ds191+ds192+ds193+ds194+ds195+ds196+ds197+y198,
0.0), point2=(set19+ds191+ds192+ds193+ds194+ds195+ds196+ds197+y198, 0.00125))

s.Line(point1=(set19+ds191+ds192+ds193+ds194+ds195+ds196+ds197+ds198,
0.0), point2=(set19+ds191+ds192+ds193+ds194+ds195+ds196+ds197+ds198, 0.0035))

s.Line(point1=(set19+ds191+ds192+ds193+ds194+ds195+ds196+ds197+ds198+x199,
0.0), point2=(set19+ds191+ds192+ds193+ds194+ds195+ds196+ds197+ds198+x199,
0.00125))

s.Line(point1=(set19+ds191+ds192+ds193+ds194+ds195+ds196+ds197+ds198+ds199,
0.0), point2=(set19+ds191+ds192+ds193+ds194+ds195+ds196+ds197+ds198+ds199,
0.0035))

s.Line(point1=(set19+ds191+ds192+ds193+ds194+ds195+ds196+ds197+ds198+ds199+y200,
0.0),
point2=(set19+ds191+ds192+ds193+ds194+ds195+ds196+ds197+ds198+ds199+y200,
0.00125))

s.Line(point1=(set20, 0.0), point2=(set20, 0.0035))

```

#SPACER SUBSTITUTION

```

print 'Top Row larger than Bottom Row'
print ('Dspace200 =', ds200)
LengthS=LengthF-(set20)
spacer=DSmean-1*DSstdev
hys=0.84*spacer
LengthS2=LengthS-spacer
LengthS3=LengthS2-spacer
LengthS4=LengthS3-spacer
LengthS5=LengthS4-spacer
LengthS6=LengthS5-spacer
LengthS7=LengthS6-spacer
LengthS8=LengthS7-spacer

```

```

#Spacer Remainder work--is the model still biologically valid?
spacerremainder=LengthF-(set20)
NewArea=(ds200+spacerremainder)*(0.0035)

```

```

NewRatio=(ds200*0.84)*(1.25E-3)/(NewArea)
if NewRatio < .25:
    print 'REJECT MODEL: Currently Biologically Invalid'
if NewRatio >= .25:
    print '*MODEL IS NOW VALID; Good for Analysis*'

if LengthS > spacer:
    print 'Added FIRST spacer to bottom row'
#creates boundary line for new DSpace in spacer substitution case
s.Line(point1=(set20+spacer, 0.0), point2=(set20+spacer, 0.0035))
#creates hydrox line for new DSpace in spacer substitution case
s.Line(point1=(set20+spacer-hys, 0.0), point2=(set20+spacer-hys,
0.00125))
#Spacer Remainder work--is the model still biologically valid?
spacerremainder=LengthF-(set20+spacer)
NewArea=(spacer+spacerremainder)*(0.0035)
NewRatio=(spacer*0.84)*(1.25E-3)/(NewArea)
if NewRatio < .25:
    print 'REJECT MODEL: Currently Biologically Invalid'
if NewRatio >= .25:
    print '*MODEL IS NOW VALID; Good for Analysis*'

if LengthS2 > spacer:
    print 'Added SECOND spacer to bottom row'
#creates boundary line for new DSpace in SECOND spacer substitution case
s.Line(point1=(set20+spacer+spacer, 0.0), point2=(set20+spacer+spacer,
0.0035))
#creates hydrox line for new DSpace in SECOND spacer substitution case
s.Line(point1=(set20+spacer+hys, 0.0), point2=(set20+spacer+hys,
0.00125))
#Spacer Remainder work--is the model still biologically valid?
spacerremainder=LengthF-(set20+spacer+spacer)
NewArea=(spacer+spacerremainder)*(0.0035)
NewRatio=(spacer*0.84)*(1.25E-3)/(NewArea)
if NewRatio < .25:
    print 'REJECT MODEL: Currently Biologically Invalid'
if NewRatio >= .25357:
    print '*MODEL IS NOW VALID; Good for Analysis*'

if LengthS3 > spacer:
    print 'Added THIRD spacer to bottom row'
#creates boundary line for new DSpace in THIRD spacer substitution case
s.Line(point1=(set20+spacer+spacer+spacer, 0.0),
point2=(set20+spacer+spacer+spacer, 0.0035))
#creates hydrox line for new DSpace in THIRD spacer substitution case
s.Line(point1=(set20+spacer+spacer+spacer-hys, 0.0),
point2=(set20+spacer+spacer+spacer-hys, 0.00125))
#Spacer Remainder work--is the model still biologically valid?

```

```

spacerremainder=LengthF-(set20+spacer+spacer+spacer)
NewArea=(spacer+spacerremainder)*(0.0035)
NewRatio=(spacer*0.84)*(1.25E-3)/(NewArea)
if NewRatio < .25:
    print 'REJECT MODEL: Currently Biologically Invalid'
if NewRatio >= .25357:
    print '*MODEL IS NOW VALID; Good for Analysis*'

if LengthS4 > spacer:
    print 'Added FOURTH spacer to bottom row'
    #creates boundary line for new DSpace in FOURTH spacer substitution case
    s.Line(point1=(set20+spacer+spacer+spacer, 0.0),
point2=(set20+spacer+spacer+spacer, 0.0035))
    #creates hydrox line for new DSpace in FOURTH spacer substitution case
    s.Line(point1=(set20+spacer+spacer+spacer+hys, 0.0),
point2=(set20+spacer+spacer+spacer+hys, 0.00125))
    #Spacer Remainder work--is the model still biologically valid?
    spacerremainder=LengthF-(set20+spacer+spacer+spacer+spacer)
    NewArea=(spacer+spacerremainder)*(0.0035)
    NewRatio=(spacer*0.84)*(1.25E-3)/(NewArea)
    if NewRatio < .25:
        print 'REJECT MODEL: Currently Biologically Invalid'
    if NewRatio >= .25357:
        print '*MODEL IS NOW VALID; Good for Analysis*'

if LengthS5 > spacer:
    print 'Added FIFTH spacer to bottom row'
    #creates boundary line for new DSpace in FIFTH spacer substitution case
    s.Line(point1=(set20+spacer+spacer+spacer+spacer, 0.0),
point2=(set20+spacer+spacer+spacer+spacer, 0.0035))
    #creates hydrox line for new DSpace in FIFTH spacer substitution case
    s.Line(point1=(set20+spacer+spacer+spacer+spacer+spacer-hys, 0.0),
point2=(set20+spacer+spacer+spacer+spacer+spacer-hys, 0.00125))
    #Spacer Remainder work--is the model still biologically valid?
    spacerremainder=LengthF-(set20+spacer+spacer+spacer+spacer+spacer)
    NewArea=(spacer+spacerremainder)*(0.0035)
    NewRatio=(spacer*0.84)*(1.25E-3)/(NewArea)
    if NewRatio < .25:
        print 'REJECT MODEL: Currently Biologically Invalid'
    if NewRatio >= .25:
        print '*MODEL IS NOW VALID; Good for Analysis*'

if LengthS6 > spacer:
    print 'Added SIXTH spacer to bottom row'
    #creates boundary line for new DSpace in SIXTH spacer substitution case
    s.Line(point1=(set20+spacer+spacer+spacer+spacer+spacer, 0.0),
point2=(set20+spacer+spacer+spacer+spacer+spacer, 0.0035))
    #creates hydrox line for new DSpace in SIXTH spacer substitution case

```



```

s.Line(point1=(set20+spacer+spacer+spacer+spacer+spacer+hys, 0.0),
point2=(set20+spacer+spacer+spacer+spacer+spacer+hys, 0.00125))
#Spacer Remainder work--is the model still biologically valid?

spacerremainder=LengthF-(set20+spacer+spacer+spacer+spacer+spacer+spacer)
NewArea=(spacer+spacerremainder)*(0.0035)
NewRatio=(spacer*0.84)*(1.25E-3)/(NewArea)
if NewRatio < .25:
    print 'REJECT MODEL: Currently Biologically Invalid'
if NewRatio >= .25:
    print '*MODEL IS NOW VALID; Good for Analysis*'

if LengthS7 > spacer:
    print 'Added SEVENTH spacer to bottom row'
#creates boundary line for new DSpace in SEVENTH spacer substitution case
s.Line(point1=(set20+spacer+spacer+spacer+spacer+spacer+spacer+spacer, 0.0),
point2=(set20+spacer+spacer+spacer+spacer+spacer+spacer+spacer, 0.0035))
#creates hydrox line for new DSpace in SEVENTH spacer substitution case

s.Line(point1=(set20+spacer+spacer+spacer+spacer+spacer+spacer+spacer+spacer-hys,
0.0), point2=(set20+spacer+spacer+spacer+spacer+spacer+spacer+spacer+spacer-hys,
0.00125))
#Spacer Remainder work--is the model still biologically valid?

spacerremainder=LengthF-(set20+spacer+spacer+spacer+spacer+spacer+spacer+spacer+spacer
)
NewArea=(spacer+spacerremainder)*(0.0035)
NewRatio=(spacer*0.84)*(1.25E-3)/(NewArea)
if NewRatio < .25:
    print 'REJECT MODEL: Currently Biologically Invalid'
if NewRatio >= .25:
    print '*MODEL IS NOW VALID; Good for Analysis*'

if LengthS8 > spacer:
    print 'Added EIGHTH spacer to bottom row NO WAY IS THIS FOR REAL?!'
#creates boundary line for new DSpace in EIGHTH spacer substitution case
s.Line(point1=(set20+spacer+spacer+spacer+spacer+spacer+spacer+spacer+spacer,
0.0), point2=(set20+spacer+spacer+spacer+spacer+spacer+spacer+spacer+spacer, 0.0035))
#creates hydrox line for new DSpace in EIGHTH spacer substitution case

s.Line(point1=(set20+spacer+spacer+spacer+spacer+spacer+spacer+spacer+spacer+hys,
0.0), point2=(set20+spacer+spacer+spacer+spacer+spacer+spacer+spacer+spacer+hys,
0.00125))
#Spacer Remainder work--is the model still biologically valid?

spacerremainder=LengthF-(set20+spacer+spacer+spacer+spacer+spacer+spacer+spacer+spacer
+spacer)
NewArea=(spacer+spacerremainder)*(0.0035)

```

```

NewRatio=(spacer*0.84)*(1.25E-3)/(NewArea)
if NewRatio < .25:
    print 'REJECT MODEL: Currently Biologically Invalid'
if NewRatio >= .25:
    print '*MODEL IS NOW VALID; Good for Analysis*'

p = mdb.models['Model-1'].parts['Composite Bone']
f = p.faces
pickedFaces = f.getSequenceFromMask(mask=('[#1 ]', ), )
e, d1 = p.edges, p.datums
p.PartitionFaceBySketch(faces=pickedFaces, sketch=s)
s.unsetPrimaryObject()
del mdb.models['Model-1'].sketches['__profile__']

```

#MATERIAL CREATION

```

mdb.models['Model-1'].Material(name='COLLAGEN')
mdb.models['Model-1'].materials['COLLAGEN'].Depvar(n=3)

mdb.models['Model-1'].materials['COLLAGEN'].UserMaterial(mechanicalConstants=(
    0.003, 0.006, 0.004, 0.2, 0.2, 0.2))
mdb.models['Model-1'].Material(name='HYDROXYAPATITE')
mdb.models['Model-1'].materials['HYDROXYAPATITE'].Elastic(table=((0.1,
0.28),
    ))
mdb.models['Model-1'].HomogeneousSolidSection(name='COLLAGEN SECTION',
    material='COLLAGEN', thickness=None)
mdb.models['Model-1'].HomogeneousSolidSection(name='HYDROXYAPATITE
SECTION',
    material='HYDROXYAPATITE', thickness=None)

```

#SECTION ASSINGMENT

```

p = mdb.models['Model-1'].parts['Composite Bone']
region = p.sets['COLLAGEN SET']
p = mdb.models['Model-1'].parts['Composite Bone']
p.SectionAssignment(region=region, sectionName='COLLAGEN SECTION',
offset=0.0,
    offsetType=MIDDLE_SURFACE, offsetField='',
    thicknessAssignment=FROM_SECTION)
p = mdb.models['Model-1'].parts['Composite Bone']
region = p.sets['HYDROXYAPATITE SET']
p = mdb.models['Model-1'].parts['Composite Bone']
p.SectionAssignment(region=region, sectionName='HYDROXYAPATITE SECTION',
    offset=0.0, offsetType=MIDDLE_SURFACE, offsetField='',
    thicknessAssignment=FROM_SECTION)

```

#INSTANCE SET AND XSYM MAKER

```

session.viewports['Viewport: 1'].assemblyDisplay.setValues(loads=OFF,
bcs=OFF,

```

```

        predefinedFields=OFF, connectors=OFF)
a = mdb.models['Model-1'].rootAssembly
a.DatumCsysByDefault(CARTESIAN)
p = mdb.models['Model-1'].parts['Composite Bone']
a.Instance(name='Composite Bone-1', part=p, dependent=ON)
session.viewports['Viewport: 1'].assemblyDisplay.setValues(
    adaptiveMeshConstraints=ON)
mdb.models['Model-1'].StaticStep(name='APPLY LOAD', previous='Initial',
    timePeriod=0.05, maxNumInc=100000, initialInc=0.05, minInc=1e-10,
    maxInc=0.05)
session.viewports['Viewport: 1'].assemblyDisplay.setValues(
    step='APPLY LOAD')
session.viewports['Viewport: 1'].assemblyDisplay.setValues(loads=ON,
bcs=ON,
    predefinedFields=ON, connectors=ON, adaptiveMeshConstraints=OFF)
mdb.models['Model-1'].PeriodicAmplitude(name='SINUSOIDAL',
timeSpan=TOTAL,
    frequency=6.28319, start=0.0, a_0=0.6875, data=((0.0, 0.3125), ))
session.viewports['Viewport: 1'].view.setValues(nearPlane=12.8063,
    farPlane=12.8435, width=0.201952, height=0.110266,
viewOffsetX=3.20047,
    viewOffsetY=-0.00229728)
session.viewports['Viewport: 1'].view.setValues(nearPlane=12.8134,
    farPlane=12.8364, width=0.1195, height=0.065247, viewOffsetX=3.20271,
    viewOffsetY=-0.00322843)
a = mdb.models['Model-1'].rootAssembly
s1 = a.instances['Composite Bone-1'].edges
side1Edges1 = s1.getSequenceFromMask(mask=(
    '#0:58 #4000000 #8000 #20000 #80000240 ]', ), )
region = regionToolset.Region(side1Edges=side1Edges1)
mdb.models['Model-1'].Pressure(name='PRESSURE', createStepName='APPLY
LOAD',
    region=region, distributionType=UNIFORM, field='',
magnitude=-3.36e-06,
    amplitude='SINUSOIDAL')
session.viewports['Viewport: 1'].view.fitView()
session.viewports['Viewport: 1'].view.setValues(nearPlane=12.4455,
    farPlane=13.2043, width=3.94297, height=2.15286,
viewOffsetX=-1.23672,
    viewOffsetY=-0.023348)
session.viewports['Viewport: 1'].view.setValues(nearPlane=12.8161,
    farPlane=12.8338, width=0.096555, height=0.0527191,
    viewOffsetX=-3.20778, viewOffsetY=0.000928941)
a = mdb.models['Model-1'].rootAssembly
e1 = a.instances['Composite Bone-1'].edges
edges1 = e1.getSequenceFromMask(mask=('[#42008020 #80004 ]', ), )
region = regionToolset.Region(edges=edges1)
mdb.models['Model-1'].XsymmBC(name='XSYM', createStepName='APPLY LOAD',

```

```
region=region)
```

```
#LengthF=Length2 (ie. The Bottom Row is Larger; Top Row Has Spacer)
```

```
elif LengthF==Length2:
    s1 = mdb.models['Model-1'].ConstrainedSketch(name='__profile__',
        sheetSize=200.0)
    g, v, d, c = s1.geometry, s1.vertices, s1.dimensions, s1.constraints
    s1.setPrimaryObject(option=STANDALONE)
    s1.rectangle(point1=(0.0, 0.0), point2=(LengthF, 0.007))
    p = mdb.models['Model-1'].Part(name='Composite Bone',
        dimensionality=TWO_D_PLANAR, type=DEFORMABLE_BODY)
    p = mdb.models['Model-1'].parts['Composite Bone']
    p.BaseShell(sketch=s1)
    s1.unsetPrimaryObject()
    p = mdb.models['Model-1'].parts['Composite Bone']
    session.viewports['Viewport: 1'].setValues(displayedObject=p)
    del mdb.models['Model-1'].sketches['__profile__']
    f, e1, d2 = p.faces, p.edges, p.datums
    t = p.MakeSketchTransform(sketchPlane=f[0], sketchPlaneSide=SIDE1,
origin=(
    0.0, 0.0, 0.0))
    s = mdb.models['Model-1'].ConstrainedSketch(name='__profile__',
        sheetSize=0.268, gridSpacing=0.006, transform=t)
    g, v, d, c = s.geometry, s.vertices, s.dimensions, s.constraints
    s.sketchOptions.setValues(decimalPlaces=3)
    s.setPrimaryObject(option=SUPERIMPOSE)
    p = mdb.models['Model-1'].parts['Composite Bone']
    p.projectReferencesOntoSketch(sketch=s, filter=COPLANAR_EDGES)
    s.Line(point1=(0, 0.00125), point2=(LengthF, 0.00125))
    s.HorizontalConstraint(entity=g[6], addUndoState=False)
    s.Line(point1=(0, 0.00275), point2=(LengthF, 0.00275))
    s.HorizontalConstraint(entity=g[7], addUndoState=False)
    s.Line(point1=(0, 0.0035), point2=(LengthF, 0.0035))
    s.HorizontalConstraint(entity=g[8], addUndoState=False)
    s.Line(point1=(0, 0.00425), point2=(LengthF, 0.00425))
    s.HorizontalConstraint(entity=g[9], addUndoState=False)
    s.Line(point1=(0, 0.00575), point2=(LengthF, 0.00575))
    s.HorizontalConstraint(entity=g[10], addUndoState=False)
```

```
#Top Row, Dspace 1-10
```

```
s.Line(point1=(y1, 0.007), point2=(y1, 0.00575))

s.Line(point1=(ds1, 0.007), point2=(ds1, 0.0035))

s.Line(point1=(ds1+x2, 0.007), point2=(ds1+x2, 0.00575))
```

```

s.Line(point1=(ds1+ds2, 0.007), point2=(ds1+ds2, 0.0035))

s.Line(point1=(ds1+ds2+y3, 0.007), point2=(ds1+ds2+y3, 0.00575))

s.Line(point1=(ds1+ds2+ds3, 0.007), point2=(ds1+ds2+ds3, 0.0035))

s.Line(point1=(ds1+ds2+ds3+x4, 0.007), point2=(ds1+ds2+ds3+x4, 0.00575))

s.Line(point1=(ds1+ds2+ds3+ds4, 0.007), point2=(ds1+ds2+ds3+ds4, 0.0035))

s.Line(point1=(ds1+ds2+ds3+ds4+y5, 0.007), point2=(ds1+ds2+ds3+ds4+y5,
0.00575))

s.Line(point1=(ds1+ds2+ds3+ds4+ds5, 0.007), point2=(ds1+ds2+ds3+ds4+ds5,
0.0035))

s.Line(point1=(ds1+ds2+ds3+ds4+ds5+x6, 0.007),
point2=(ds1+ds2+ds3+ds4+ds5+x6, 0.00575))

s.Line(point1=(ds1+ds2+ds3+ds4+ds5+ds6, 0.007),
point2=(ds1+ds2+ds3+ds4+ds5+ds6, 0.0035))

s.Line(point1=(ds1+ds2+ds3+ds4+ds5+ds6+y7, 0.007),
point2=(ds1+ds2+ds3+ds4+ds5+ds6+y7, 0.00575))

s.Line(point1=(ds1+ds2+ds3+ds4+ds5+ds6+ds7, 0.007),
point2=(ds1+ds2+ds3+ds4+ds5+ds6+ds7, 0.0035))

s.Line(point1=(ds1+ds2+ds3+ds4+ds5+ds6+ds7+x8, 0.007),
point2=(ds1+ds2+ds3+ds4+ds5+ds6+ds7+x8, 0.00575))

s.Line(point1=(ds1+ds2+ds3+ds4+ds5+ds6+ds7+ds8, 0.007),
point2=(ds1+ds2+ds3+ds4+ds5+ds6+ds7+ds8, 0.0035))

s.Line(point1=(ds1+ds2+ds3+ds4+ds5+ds6+ds7+ds8+y9, 0.007),
point2=(ds1+ds2+ds3+ds4+ds5+ds6+ds7+ds8+y9, 0.00575))

s.Line(point1=(ds1+ds2+ds3+ds4+ds5+ds6+ds7+ds8+ds9, 0.007),
point2=(ds1+ds2+ds3+ds4+ds5+ds6+ds7+ds8+ds9, 0.0035))

s.Line(point1=(ds1+ds2+ds3+ds4+ds5+ds6+ds7+ds8+ds9+x10, 0.007),
point2=(ds1+ds2+ds3+ds4+ds5+ds6+ds7+ds8+ds9+x10, 0.00575))

s.Line(point1=(set1, 0.007), point2=(set1, 0.0035))

```

#Top Row, Dspace 11–20

```

s.Line(point1=(set1+y11, 0.007), point2=(set1+y11, 0.00575))

s.Line(point1=(set1+ds11, 0.007), point2=(set1+ds11, 0.0035))

s.Line(point1=(set1+ds11+x12, 0.007), point2=(set1+ds11+x12, 0.00575))

s.Line(point1=(set1+ds11+ds12, 0.007), point2=(set1+ds11+ds12, 0.0035))

s.Line(point1=(set1+ds11+ds12+y13, 0.007), point2=(set1+ds11+ds12+y13,
0.00575))

s.Line(point1=(set1+ds11+ds12+ds13, 0.007), point2=(set1+ds11+ds12+ds13,
0.0035))

s.Line(point1=(set1+ds11+ds12+ds13+x14, 0.007),
point2=(set1+ds11+ds12+ds13+x14, 0.00575))

s.Line(point1=(set1+ds11+ds12+ds13+ds14, 0.007),
point2=(set1+ds11+ds12+ds13+ds14, 0.0035))

s.Line(point1=(set1+ds11+ds12+ds13+ds14+y15, 0.007),
point2=(set1+ds11+ds12+ds13+ds14+y15, 0.00575))

s.Line(point1=(set1+ds11+ds12+ds13+ds14+ds15, 0.007),
point2=(set1+ds11+ds12+ds13+ds14+ds15, 0.0035))

s.Line(point1=(set1+ds11+ds12+ds13+ds14+ds15+x16, 0.007),
point2=(set1+ds11+ds12+ds13+ds14+ds15+x16, 0.00575))

s.Line(point1=(set1+ds11+ds12+ds13+ds14+ds15+ds16, 0.007),
point2=(set1+ds11+ds12+ds13+ds14+ds15+ds16, 0.0035))

s.Line(point1=(set1+ds11+ds12+ds13+ds14+ds15+ds16+y17, 0.007),
point2=(set1+ds11+ds12+ds13+ds14+ds15+ds16+y17, 0.00575))

s.Line(point1=(set1+ds11+ds12+ds13+ds14+ds15+ds16+ds17, 0.007),
point2=(set1+ds11+ds12+ds13+ds14+ds15+ds16+ds17, 0.0035))

s.Line(point1=(set1+ds11+ds12+ds13+ds14+ds15+ds16+ds17+x18, 0.007),
point2=(set1+ds11+ds12+ds13+ds14+ds15+ds16+ds17+x18, 0.00575))

s.Line(point1=(set1+ds11+ds12+ds13+ds14+ds15+ds16+ds17+ds18, 0.007),
point2=(set1+ds11+ds12+ds13+ds14+ds15+ds16+ds17+ds18, 0.0035))

s.Line(point1=(set1+ds11+ds12+ds13+ds14+ds15+ds16+ds17+ds18+y19, 0.007),
point2=(set1+ds11+ds12+ds13+ds14+ds15+ds16+ds17+ds18+y19, 0.00575))

s.Line(point1=(set1+ds11+ds12+ds13+ds14+ds15+ds16+ds17+ds18+ds19, 0.007),

```

```
point2=(set1+ds11+ds12+ds13+ds14+ds15+ds16+ds17+ds18+ds19, 0.0035))
```

```
    s.Line(point1=(set1+ds11+ds12+ds13+ds14+ds15+ds16+ds17+ds18+ds19+x20,  
0.007), point2=(set1+ds11+ds12+ds13+ds14+ds15+ds16+ds17+ds18+ds19+x20,  
0.00575))
```

#Top Row, Dspace 21-30

```
    s.Line(point1=(set2, 0.007), point2=(set2, 0.0035))
```

```
    s.Line(point1=(set2+y21, 0.007), point2=(set2+y21, 0.00575))
```

```
    s.Line(point1=(set2+ds21, 0.007), point2=(set2+ds21, 0.0035))
```

```
    s.Line(point1=(set2+ds21+x22, 0.007), point2=(set2+ds21+x22, 0.00575))
```

```
    s.Line(point1=(set2+ds21+ds22, 0.007), point2=(set2+ds21+ds22, 0.0035))
```

```
    s.Line(point1=(set2+ds21+ds22+y23, 0.007), point2=(set2+ds21+ds22+y23,  
0.00575))
```

```
    s.Line(point1=(set2+ds21+ds22+ds23, 0.007), point2=(set2+ds21+ds22+ds23,  
0.0035))
```

```
    s.Line(point1=(set2+ds21+ds22+ds23+x24, 0.007),  
point2=(set2+ds21+ds22+ds23+x24, 0.00575))
```

```
    s.Line(point1=(set2+ds21+ds22+ds23+ds24, 0.007),  
point2=(set2+ds21+ds22+ds23+ds24, 0.0035))
```

```
    s.Line(point1=(set2+ds21+ds22+ds23+ds24+y25, 0.007),  
point2=(set2+ds21+ds22+ds23+ds24+y25, 0.00575))
```

```
    s.Line(point1=(set2+ds21+ds22+ds23+ds24+ds25, 0.007),  
point2=(set2+ds21+ds22+ds23+ds24+ds25, 0.0035))
```

```
    s.Line(point1=(set2+ds21+ds22+ds23+ds24+ds25+x26, 0.007),  
point2=(set2+ds21+ds22+ds23+ds24+ds25+x26, 0.00575))
```

```
    s.Line(point1=(set2+ds21+ds22+ds23+ds24+ds25+ds26, 0.007),  
point2=(set2+ds21+ds22+ds23+ds24+ds25+ds26, 0.0035))
```

```
    s.Line(point1=(set2+ds21+ds22+ds23+ds24+ds25+ds26+y27, 0.007),  
point2=(set2+ds21+ds22+ds23+ds24+ds25+ds26+y27, 0.00575))
```

```
    s.Line(point1=(set2+ds21+ds22+ds23+ds24+ds25+ds26+ds27, 0.007),  
point2=(set2+ds21+ds22+ds23+ds24+ds25+ds26+ds27, 0.0035))
```

```

s.Line(point1=(set2+ds21+ds22+ds23+ds24+ds25+ds26+ds27+x28, 0.007),
point2=(set2+ds21+ds22+ds23+ds24+ds25+ds26+ds27+x28, 0.00575))

s.Line(point1=(set2+ds21+ds22+ds23+ds24+ds25+ds26+ds27+ds28, 0.007),
point2=(set2+ds21+ds22+ds23+ds24+ds25+ds26+ds27+ds28, 0.0035))

s.Line(point1=(set2+ds21+ds22+ds23+ds24+ds25+ds26+ds27+ds28+y29, 0.007),
point2=(set2+ds21+ds22+ds23+ds24+ds25+ds26+ds27+ds28+y29, 0.00575))

s.Line(point1=(set2+ds21+ds22+ds23+ds24+ds25+ds26+ds27+ds28+ds29,
0.007), point2=(set2+ds21+ds22+ds23+ds24+ds25+ds26+ds27+ds28+ds29, 0.0035))

s.Line(point1=(set2+ds21+ds22+ds23+ds24+ds25+ds26+ds27+ds28+ds29+x30,
0.007), point2=(set2+ds21+ds22+ds23+ds24+ds25+ds26+ds27+ds28+ds29+x30,
0.00575))

```

#Top Row, Dspace 31-40

```

s.Line(point1=(set3, 0.007), point2=(set3, 0.0035))

s.Line(point1=(set3+y31, 0.007), point2=(set3+y31, 0.00575))

s.Line(point1=(set3+ds31, 0.007), point2=(set3+ds31, 0.0035))

s.Line(point1=(set3+ds31+x32, 0.007), point2=(set3+ds31+x32, 0.00575))

s.Line(point1=(set3+ds31+ds32, 0.007), point2=(set3+ds31+ds32, 0.0035))

s.Line(point1=(set3+ds31+ds32+y33, 0.007), point2=(set3+ds31+ds32+y33,
0.00575))

s.Line(point1=(set3+ds31+ds32+ds33, 0.007), point2=(set3+ds31+ds32+ds33,
0.0035))

s.Line(point1=(set3+ds31+ds32+ds33+x34, 0.007),
point2=(set3+ds31+ds32+ds33+x34, 0.00575))

s.Line(point1=(set3+ds31+ds32+ds33+ds34, 0.007),
point2=(set3+ds31+ds32+ds33+ds34, 0.0035))

s.Line(point1=(set3+ds31+ds32+ds33+ds34+y35, 0.007),
point2=(set3+ds31+ds32+ds33+ds34+y35, 0.00575))

s.Line(point1=(set3+ds31+ds32+ds33+ds34+ds35, 0.007),
point2=(set3+ds31+ds32+ds33+ds34+ds35, 0.0035))

s.Line(point1=(set3+ds31+ds32+ds33+ds34+ds35+x36, 0.007),
point2=(set3+ds31+ds32+ds33+ds34+ds35+x36, 0.00575))

```



```

s.Line(point1=(set3+ds31+ds32+ds33+ds34+ds35+ds36, 0.007),
point2=(set3+ds31+ds32+ds33+ds34+ds35+ds36, 0.0035))

s.Line(point1=(set3+ds31+ds32+ds33+ds34+ds35+ds36+y37, 0.007),
point2=(set3+ds31+ds32+ds33+ds34+ds35+ds36+y37, 0.00575))

s.Line(point1=(set3+ds31+ds32+ds33+ds34+ds35+ds36+ds37, 0.007),
point2=(set3+ds31+ds32+ds33+ds34+ds35+ds36+ds37, 0.0035))

s.Line(point1=(set3+ds31+ds32+ds33+ds34+ds35+ds36+ds37+x38, 0.007),
point2=(set3+ds31+ds32+ds33+ds34+ds35+ds36+ds37+x38, 0.00575))

s.Line(point1=(set3+ds31+ds32+ds33+ds34+ds35+ds36+ds37+ds38, 0.007),
point2=(set3+ds31+ds32+ds33+ds34+ds35+ds36+ds37+ds38, 0.0035))

s.Line(point1=(set3+ds31+ds32+ds33+ds34+ds35+ds36+ds37+ds38+y39, 0.007),
point2=(set3+ds31+ds32+ds33+ds34+ds35+ds36+ds37+ds38+y39, 0.00575))

s.Line(point1=(set3+ds31+ds32+ds33+ds34+ds35+ds36+ds37+ds38+ds39, 0.007),
point2=(set3+ds31+ds32+ds33+ds34+ds35+ds36+ds37+ds38+ds39, 0.0035))

s.Line(point1=(set3+ds31+ds32+ds33+ds34+ds35+ds36+ds37+ds38+ds39+x40,
0.007), point2=(set3+ds31+ds32+ds33+ds34+ds35+ds36+ds37+ds38+ds39+x40,
0.00575))

```

#Top Row, Dspace 41-50

```

s.Line(point1=(set4, 0.007), point2=(set4, 0.0035))

s.Line(point1=(set4+y41, 0.007), point2=(set4+y41, 0.00575))

s.Line(point1=(set4+ds41, 0.007), point2=(set4+ds41, 0.0035))

s.Line(point1=(set4+ds41+x42, 0.007), point2=(set4+ds41+x42, 0.00575))

s.Line(point1=(set4+ds41+ds42, 0.007), point2=(set4+ds41+ds42, 0.0035))

s.Line(point1=(set4+ds41+ds42+y43, 0.007), point2=(set4+ds41+ds42+y43,
0.00575))

s.Line(point1=(set4+ds41+ds42+ds43, 0.007), point2=(set4+ds41+ds42+ds43,
0.0035))

s.Line(point1=(set4+ds41+ds42+ds43+x44, 0.007),
point2=(set4+ds41+ds42+ds43+x44, 0.00575))

s.Line(point1=(set4+ds41+ds42+ds43+ds44, 0.007),

```

```

point2=(set4+ds41+ds42+ds43+ds44, 0.0035))

    s.Line(point1=(set4+ds41+ds42+ds43+ds44+y45, 0.007),
point2=(set4+ds41+ds42+ds43+ds44+y45, 0.00575))

    s.Line(point1=(set4+ds41+ds42+ds43+ds44+ds45, 0.007),
point2=(set4+ds41+ds42+ds43+ds44+ds45, 0.0035))

    s.Line(point1=(set4+ds41+ds42+ds43+ds44+ds45+x46, 0.007),
point2=(set4+ds41+ds42+ds43+ds44+ds45+x46, 0.00575))

    s.Line(point1=(set4+ds41+ds42+ds43+ds44+ds45+ds46, 0.007),
point2=(set4+ds41+ds42+ds43+ds44+ds45+ds46, 0.0035))

    s.Line(point1=(set4+ds41+ds42+ds43+ds44+ds45+ds46+y47, 0.007),
point2=(set4+ds41+ds42+ds43+ds44+ds45+ds46+y47, 0.00575))

    s.Line(point1=(set4+ds41+ds42+ds43+ds44+ds45+ds46+ds47, 0.007),
point2=(set4+ds41+ds42+ds43+ds44+ds45+ds46+ds47, 0.0035))

    s.Line(point1=(set4+ds41+ds42+ds43+ds44+ds45+ds46+ds47+x48, 0.007),
point2=(set4+ds41+ds42+ds43+ds44+ds45+ds46+ds47+x48, 0.00575))

    s.Line(point1=(set4+ds41+ds42+ds43+ds44+ds45+ds46+ds47+ds48, 0.007),
point2=(set4+ds41+ds42+ds43+ds44+ds45+ds46+ds47+ds48, 0.0035))

    s.Line(point1=(set4+ds41+ds42+ds43+ds44+ds45+ds46+ds47+ds48+y49, 0.007),
point2=(set4+ds41+ds42+ds43+ds44+ds45+ds46+ds47+ds48+y49, 0.00575))

    s.Line(point1=(set4+ds41+ds42+ds43+ds44+ds45+ds46+ds47+ds48+ds49, 0.007),
point2=(set4+ds41+ds42+ds43+ds44+ds45+ds46+ds47+ds48+ds49, 0.0035))

    s.Line(point1=(set4+ds41+ds42+ds43+ds44+ds45+ds46+ds47+ds48+ds49+x50,
0.007), point2=(set4+ds41+ds42+ds43+ds44+ds45+ds46+ds47+ds48+ds49+x50,
0.00575))

```

#Top Row, Dspace 51-60

```

s.Line(point1=(set5, 0.007), point2=(set5, 0.0035))

s.Line(point1=(set5+y51, 0.007), point2=(set5+y51, 0.00575))

s.Line(point1=(set5+ds51, 0.007), point2=(set5+ds51, 0.0035))

s.Line(point1=(set5+ds51+x52, 0.007), point2=(set5+ds51+x52, 0.00575))

s.Line(point1=(set5+ds51+ds52, 0.007), point2=(set5+ds51+ds52, 0.0035))

```

```

        s.Line(point1=(set5+ds51+ds52+y53, 0.007), point2=(set5+ds51+ds52+y53,
0.00575))

        s.Line(point1=(set5+ds51+ds52+ds53, 0.007), point2=(set5+ds51+ds52+ds53,
0.0035))

        s.Line(point1=(set5+ds51+ds52+ds53+x54, 0.007),
point2=(set5+ds51+ds52+ds53+x54, 0.00575))

        s.Line(point1=(set5+ds51+ds52+ds53+ds54, 0.007),
point2=(set5+ds51+ds52+ds53+ds54, 0.0035))

        s.Line(point1=(set5+ds51+ds52+ds53+ds54+y55, 0.007),
point2=(set5+ds51+ds52+ds53+ds54+y55, 0.00575))

        s.Line(point1=(set5+ds51+ds52+ds53+ds54+ds55, 0.007),
point2=(set5+ds51+ds52+ds53+ds54+ds55, 0.0035))

        s.Line(point1=(set5+ds51+ds52+ds53+ds54+ds55+x56, 0.007),
point2=(set5+ds51+ds52+ds53+ds54+ds55+x56, 0.00575))

        s.Line(point1=(set5+ds51+ds52+ds53+ds54+ds55+ds56, 0.007),
point2=(set5+ds51+ds52+ds53+ds54+ds55+ds56, 0.0035))

        s.Line(point1=(set5+ds51+ds52+ds53+ds54+ds55+ds56+y57, 0.007),
point2=(set5+ds51+ds52+ds53+ds54+ds55+ds56+y57, 0.00575))

        s.Line(point1=(set5+ds51+ds52+ds53+ds54+ds55+ds56+ds57, 0.007),
point2=(set5+ds51+ds52+ds53+ds54+ds55+ds56+ds57, 0.0035))

        s.Line(point1=(set5+ds51+ds52+ds53+ds54+ds55+ds56+ds57+x58, 0.007),
point2=(set5+ds51+ds52+ds53+ds54+ds55+ds56+ds57+x58, 0.00575))

        s.Line(point1=(set5+ds51+ds52+ds53+ds54+ds55+ds56+ds57+ds58, 0.007),
point2=(set5+ds51+ds52+ds53+ds54+ds55+ds56+ds57+ds58, 0.0035))

        s.Line(point1=(set5+ds51+ds52+ds53+ds54+ds55+ds56+ds57+ds58+y59, 0.007),
point2=(set5+ds51+ds52+ds53+ds54+ds55+ds56+ds57+ds58+y59, 0.00575))

        s.Line(point1=(set5+ds51+ds52+ds53+ds54+ds55+ds56+ds57+ds58+ds59, 0.007),
point2=(set5+ds51+ds52+ds53+ds54+ds55+ds56+ds57+ds58+ds59, 0.0035))

        s.Line(point1=(set5+ds51+ds52+ds53+ds54+ds55+ds56+ds57+ds58+ds59+x60,
0.007), point2=(set5+ds51+ds52+ds53+ds54+ds55+ds56+ds57+ds58+ds59+x60,
0.00575))
#Top Row, Dspace 61-70

        s.Line(point1=(set6, 0.007), point2=(set6, 0.0035))

```

```

s.Line(point1=(set6+y61, 0.007), point2=(set6+y61, 0.00575))

s.Line(point1=(set6+ds61, 0.007), point2=(set6+ds61, 0.0035))

s.Line(point1=(set6+ds61+x62, 0.007), point2=(set6+ds61+x62, 0.00575))

s.Line(point1=(set6+ds61+ds62, 0.007), point2=(set6+ds61+ds62, 0.0035))

s.Line(point1=(set6+ds61+ds62+y63, 0.007), point2=(set6+ds61+ds62+y63,
0.00575))

s.Line(point1=(set6+ds61+ds62+ds63, 0.007), point2=(set6+ds61+ds62+ds63,
0.0035))

s.Line(point1=(set6+ds61+ds62+ds63+x64, 0.007),
point2=(set6+ds61+ds62+ds63+x64, 0.00575))

s.Line(point1=(set6+ds61+ds62+ds63+ds64, 0.007),
point2=(set6+ds61+ds62+ds63+ds64, 0.0035))

s.Line(point1=(set6+ds61+ds62+ds63+ds64+y65, 0.007),
point2=(set6+ds61+ds62+ds63+ds64+y65, 0.00575))

s.Line(point1=(set6+ds61+ds62+ds63+ds64+ds65, 0.007),
point2=(set6+ds61+ds62+ds63+ds64+ds65, 0.0035))

s.Line(point1=(set6+ds61+ds62+ds63+ds64+ds65+x66, 0.007),
point2=(set6+ds61+ds62+ds63+ds64+ds65+x66, 0.00575))

s.Line(point1=(set6+ds61+ds62+ds63+ds64+ds65+ds66, 0.007),
point2=(set6+ds61+ds62+ds63+ds64+ds65+ds66, 0.0035))

s.Line(point1=(set6+ds61+ds62+ds63+ds64+ds65+ds66+y67, 0.007),
point2=(set6+ds61+ds62+ds63+ds64+ds65+ds66+y67, 0.00575))

s.Line(point1=(set6+ds61+ds62+ds63+ds64+ds65+ds66+ds67, 0.007),
point2=(set6+ds61+ds62+ds63+ds64+ds65+ds66+ds67, 0.0035))

s.Line(point1=(set6+ds61+ds62+ds63+ds64+ds65+ds66+ds67+x68, 0.007),
point2=(set6+ds61+ds62+ds63+ds64+ds65+ds66+ds67+x68, 0.00575))

s.Line(point1=(set6+ds61+ds62+ds63+ds64+ds65+ds66+ds67+ds68, 0.007),
point2=(set6+ds61+ds62+ds63+ds64+ds65+ds66+ds67+ds68, 0.0035))

s.Line(point1=(set6+ds61+ds62+ds63+ds64+ds65+ds66+ds67+ds68+y69, 0.007),
point2=(set6+ds61+ds62+ds63+ds64+ds65+ds66+ds67+ds68+y69, 0.00575))

```

```
s.Line(point1=(set6+ds61+ds62+ds63+ds64+ds65+ds66+ds67+ds68+ds69, 0.007),  
point2=(set6+ds61+ds62+ds63+ds64+ds65+ds66+ds67+ds68+ds69, 0.0035))
```

```
s.Line(point1=(set6+ds61+ds62+ds63+ds64+ds65+ds66+ds67+ds68+ds69+x70,  
0.007), point2=(set6+ds61+ds62+ds63+ds64+ds65+ds66+ds67+ds68+ds69+x70,  
0.00575))
```

#Top Row, Dspace 71-80

```
s.Line(point1=(set7, 0.007), point2=(set7, 0.0035))
```

```
s.Line(point1=(set7+y71, 0.007), point2=(set7+y71, 0.00575))
```

```
s.Line(point1=(set7+ds71, 0.007), point2=(set7+ds71, 0.0035))
```

```
s.Line(point1=(set7+ds71+x72, 0.007), point2=(set7+ds71+x72, 0.00575))
```

```
s.Line(point1=(set7+ds71+ds72, 0.007), point2=(set7+ds71+ds72, 0.0035))
```

```
s.Line(point1=(set7+ds71+ds72+y73, 0.007), point2=(set7+ds71+ds72+y73,  
0.00575))
```

```
s.Line(point1=(set7+ds71+ds72+ds73, 0.007), point2=(set7+ds71+ds72+ds73,  
0.0035))
```

```
s.Line(point1=(set7+ds71+ds72+ds73+x74, 0.007),  
point2=(set7+ds71+ds72+ds73+x74, 0.00575))
```

```
s.Line(point1=(set7+ds71+ds72+ds73+ds74, 0.007),  
point2=(set7+ds71+ds72+ds73+ds74, 0.0035))
```

```
s.Line(point1=(set7+ds71+ds72+ds73+ds74+y75, 0.007),  
point2=(set7+ds71+ds72+ds73+ds74+y75, 0.00575))
```

```
s.Line(point1=(set7+ds71+ds72+ds73+ds74+ds75, 0.007),  
point2=(set7+ds71+ds72+ds73+ds74+ds75, 0.0035))
```

```
s.Line(point1=(set7+ds71+ds72+ds73+ds74+ds75+x76, 0.007),  
point2=(set7+ds71+ds72+ds73+ds74+ds75+x76, 0.00575))
```

```
s.Line(point1=(set7+ds71+ds72+ds73+ds74+ds75+ds76, 0.007),  
point2=(set7+ds71+ds72+ds73+ds74+ds75+ds76, 0.0035))
```

```
s.Line(point1=(set7+ds71+ds72+ds73+ds74+ds75+ds76+y77, 0.007),  
point2=(set7+ds71+ds72+ds73+ds74+ds75+ds76+y77, 0.00575))
```

```
s.Line(point1=(set7+ds71+ds72+ds73+ds74+ds75+ds76+ds77, 0.007),  
point2=(set7+ds71+ds72+ds73+ds74+ds75+ds76+ds77, 0.0035))
```

```

s.Line(point1=(set7+ds71+ds72+ds73+ds74+ds75+ds76+ds77+x78, 0.007),
point2=(set7+ds71+ds72+ds73+ds74+ds75+ds76+ds77+x78, 0.00575))

s.Line(point1=(set7+ds71+ds72+ds73+ds74+ds75+ds76+ds77+ds78, 0.007),
point2=(set7+ds71+ds72+ds73+ds74+ds75+ds76+ds77+ds78, 0.0035))

s.Line(point1=(set7+ds71+ds72+ds73+ds74+ds75+ds76+ds77+ds78+y79, 0.007),
point2=(set7+ds71+ds72+ds73+ds74+ds75+ds76+ds77+ds78+y79, 0.00575))

s.Line(point1=(set7+ds71+ds72+ds73+ds74+ds75+ds76+ds77+ds78+ds79, 0.007),
point2=(set7+ds71+ds72+ds73+ds74+ds75+ds76+ds77+ds78+ds79, 0.0035))

s.Line(point1=(set7+ds71+ds72+ds73+ds74+ds75+ds76+ds77+ds78+ds79+x80,
0.007), point2=(set7+ds71+ds72+ds73+ds74+ds75+ds76+ds77+ds78+ds79+x80,
0.00575))

```

#Top Row, Dspace 81-90

```

s.Line(point1=(set8, 0.007), point2=(set8, 0.0035))

s.Line(point1=(set8+y81, 0.007), point2=(set8+y81, 0.00575))

s.Line(point1=(set8+ds81, 0.007), point2=(set8+ds81, 0.0035))

s.Line(point1=(set8+ds81+x82, 0.007), point2=(set8+ds81+x82, 0.00575))

s.Line(point1=(set8+ds81+ds82, 0.007), point2=(set8+ds81+ds82, 0.0035))

s.Line(point1=(set8+ds81+ds82+y83, 0.007), point2=(set8+ds81+ds82+y83,
0.00575))

s.Line(point1=(set8+ds81+ds82+ds83, 0.007), point2=(set8+ds81+ds82+ds83,
0.0035))

s.Line(point1=(set8+ds81+ds82+ds83+x84, 0.007),
point2=(set8+ds81+ds82+ds83+x84, 0.00575))

s.Line(point1=(set8+ds81+ds82+ds83+ds84, 0.007),
point2=(set8+ds81+ds82+ds83+ds84, 0.0035))

s.Line(point1=(set8+ds81+ds82+ds83+ds84+y85, 0.007),
point2=(set8+ds81+ds82+ds83+ds84+y85, 0.00575))

s.Line(point1=(set8+ds81+ds82+ds83+ds84+ds85, 0.007),
point2=(set8+ds81+ds82+ds83+ds84+ds85, 0.0035))

s.Line(point1=(set8+ds81+ds82+ds83+ds84+ds85+x86, 0.007),

```

```

point2=(set8+ds81+ds82+ds83+ds84+ds85+x86, 0.00575))

    s.Line(point1=(set8+ds81+ds82+ds83+ds84+ds85+ds86, 0.007),
point2=(set8+ds81+ds82+ds83+ds84+ds85+ds86, 0.0035))

    s.Line(point1=(set8+ds81+ds82+ds83+ds84+ds85+ds86+y87, 0.007),
point2=(set8+ds81+ds82+ds83+ds84+ds85+ds86+y87, 0.00575))

    s.Line(point1=(set8+ds81+ds82+ds83+ds84+ds85+ds86+ds87, 0.007),
point2=(set8+ds81+ds82+ds83+ds84+ds85+ds86+ds87, 0.0035))

    s.Line(point1=(set8+ds81+ds82+ds83+ds84+ds85+ds86+ds87+x88, 0.007),
point2=(set8+ds81+ds82+ds83+ds84+ds85+ds86+ds87+x88, 0.00575))

    s.Line(point1=(set8+ds81+ds82+ds83+ds84+ds85+ds86+ds87+ds88, 0.007),
point2=(set8+ds81+ds82+ds83+ds84+ds85+ds86+ds87+ds88, 0.0035))

    s.Line(point1=(set8+ds81+ds82+ds83+ds84+ds85+ds86+ds87+ds88+y89, 0.007),
point2=(set8+ds81+ds82+ds83+ds84+ds85+ds86+ds87+ds88+y89, 0.00575))

    s.Line(point1=(set8+ds81+ds82+ds83+ds84+ds85+ds86+ds87+ds88+ds89, 0.007),
point2=(set8+ds81+ds82+ds83+ds84+ds85+ds86+ds87+ds88+ds89, 0.0035))

    s.Line(point1=(set8+ds81+ds82+ds83+ds84+ds85+ds86+ds87+ds88+ds89+x90,
0.007), point2=(set8+ds81+ds82+ds83+ds84+ds85+ds86+ds87+ds88+ds89+x90,
0.00575))

```

#Top Row, Dspace 91-100

```

    s.Line(point1=(set9, 0.007), point2=(set9, 0.0035))

    s.Line(point1=(set9+y91, 0.007), point2=(set9+y91, 0.00575))

    s.Line(point1=(set9+ds91, 0.007), point2=(set9+ds91, 0.0035))

    s.Line(point1=(set9+ds91+x92, 0.007), point2=(set9+ds91+x92, 0.00575))

    s.Line(point1=(set9+ds91+ds92, 0.007), point2=(set9+ds91+ds92, 0.0035))

    s.Line(point1=(set9+ds91+ds92+y93, 0.007), point2=(set9+ds91+ds92+y93,
0.00575))

    s.Line(point1=(set9+ds91+ds92+ds93, 0.007), point2=(set9+ds91+ds92+ds93,
0.0035))

    s.Line(point1=(set9+ds91+ds92+ds93+x94, 0.007),
point2=(set9+ds91+ds92+ds93+x94, 0.00575))

```

```

        s.Line(point1=(set9+ds91+ds92+ds93+ds94, 0.007),
point2=(set9+ds91+ds92+ds93+ds94, 0.0035))

        s.Line(point1=(set9+ds91+ds92+ds93+ds94+y95, 0.007),
point2=(set9+ds91+ds92+ds93+ds94+y95, 0.00575))

        s.Line(point1=(set9+ds91+ds92+ds93+ds94+ds95, 0.007),
point2=(set9+ds91+ds92+ds93+ds94+ds95, 0.0035))

        s.Line(point1=(set9+ds91+ds92+ds93+ds94+ds95+x96, 0.007),
point2=(set9+ds91+ds92+ds93+ds94+ds95+x96, 0.00575))

        s.Line(point1=(set9+ds91+ds92+ds93+ds94+ds95+ds96, 0.007),
point2=(set9+ds91+ds92+ds93+ds94+ds95+ds96, 0.0035))

        s.Line(point1=(set9+ds91+ds92+ds93+ds94+ds95+ds96+y97, 0.007),
point2=(set9+ds91+ds92+ds93+ds94+ds95+ds96+y97, 0.00575))

        s.Line(point1=(set9+ds91+ds92+ds93+ds94+ds95+ds96+ds97, 0.007),
point2=(set9+ds91+ds92+ds93+ds94+ds95+ds96+ds97, 0.0035))

        s.Line(point1=(set9+ds91+ds92+ds93+ds94+ds95+ds96+ds97+x98, 0.007),
point2=(set9+ds91+ds92+ds93+ds94+ds95+ds96+ds97+x98, 0.00575))

        s.Line(point1=(set9+ds91+ds92+ds93+ds94+ds95+ds96+ds97+ds98, 0.007),
point2=(set9+ds91+ds92+ds93+ds94+ds95+ds96+ds97+ds98, 0.0035))

        s.Line(point1=(set9+ds91+ds92+ds93+ds94+ds95+ds96+ds97+ds98+y99, 0.007),
point2=(set9+ds91+ds92+ds93+ds94+ds95+ds96+ds97+ds98+y99, 0.00575))

        s.Line(point1=(set9+ds91+ds92+ds93+ds94+ds95+ds96+ds97+ds98+ds99, 0.007),
point2=(set9+ds91+ds92+ds93+ds94+ds95+ds96+ds97+ds98+ds99, 0.0035))

        s.Line(point1=(set9+ds91+ds92+ds93+ds94+ds95+ds96+ds97+ds98+ds99+x100,
0.007), point2=(set9+ds91+ds92+ds93+ds94+ds95+ds96+ds97+ds98+ds99+x100,
0.00575))

        s.Line(point1=(set10, 0.007), point2=(set10, 0.0035))

#Bottom Row, Dspace 1-10

        s.Line(point1=(x101, 0.0), point2=(x101, 0.00125))

        s.Line(point1=(ds101, 0.0), point2=(ds101, 0.0035))

        s.Line(point1=(ds101+y102, 0), point2=(ds101+y102, 0.00125))

        s.Line(point1=(ds101+ds102, 0.0), point2=(ds101+ds102, 0.0035))

```



```

s.Line(point1=(ds101+ds102+x103, 0.0), point2=(ds101+ds102+x103,
0.00125))

s.Line(point1=(ds101+ds102+ds103, 0.0), point2=(ds101+ds102+ds103,
0.0035))

s.Line(point1=(ds101+ds102+ds103+y104, 0.0),
point2=(ds101+ds102+ds103+y104, 0.00125))

s.Line(point1=(ds101+ds102+ds103+ds104, 0.0),
point2=(ds101+ds102+ds103+ds104, 0.0035))

s.Line(point1=(ds101+ds102+ds103+ds104+x105, 0.0),
point2=(ds101+ds102+ds103+ds104+x105, 0.00125))

s.Line(point1=(ds101+ds102+ds103+ds104+ds105, 0.0),
point2=(ds101+ds102+ds103+ds104+ds105, 0.0035))

s.Line(point1=(ds101+ds102+ds103+ds104+ds105+y106, 0.0),
point2=(ds101+ds102+ds103+ds104+ds105+y106, 0.00125))

s.Line(point1=(ds101+ds102+ds103+ds104+ds105+ds106, 0.0),
point2=(ds101+ds102+ds103+ds104+ds105+ds106, 0.0035))

s.Line(point1=(ds101+ds102+ds103+ds104+ds105+ds106+x107, 0.0),
point2=(ds101+ds102+ds103+ds104+ds105+ds106+x107, 0.00125))

s.Line(point1=(ds101+ds102+ds103+ds104+ds105+ds106+ds107, 0.0),
point2=(ds101+ds102+ds103+ds104+ds105+ds106+ds107, 0.0035))

s.Line(point1=(ds101+ds102+ds103+ds104+ds105+ds106+ds107+y108, 0.0),
point2=(ds101+ds102+ds103+ds104+ds105+ds106+ds107+y108, 0.00125))

s.Line(point1=(ds101+ds102+ds103+ds104+ds105+ds106+ds107+ds108, 0.0),
point2=(ds101+ds102+ds103+ds104+ds105+ds106+ds107+ds108, 0.0035))

s.Line(point1=(ds101+ds102+ds103+ds104+ds105+ds106+ds107+ds108+x109,
0.0), point2=(ds101+ds102+ds103+ds104+ds105+ds106+ds107+ds108+x109, 0.00125))

s.Line(point1=(ds101+ds102+ds103+ds104+ds105+ds106+ds107+ds108+ds109,
0.0), point2=(ds101+ds102+ds103+ds104+ds105+ds106+ds107+ds108+ds109, 0.0035))

s.Line(point1=(ds101+ds102+ds103+ds104+ds105+ds106+ds107+ds108+ds109+y110,
0.0), point2=(ds101+ds102+ds103+ds104+ds105+ds106+ds107+ds108+ds109+y110,
0.00125))

```

```
s.Line(point1=(set11, 0.0), point2=(set11, 0.0035))
```

```
#Bottom Row, Dspace 11-20
```

```
s.Line(point1=(set11+x111, 0.0), point2=(set11+x111, 0.00125))
```

```
s.Line(point1=(set11+ds111, 0.0), point2=(set11+ds111, 0.0035))
```

```
s.Line(point1=(set11+ds111+y112, 0), point2=(set11+ds111+y112, 0.00125))
```

```
s.Line(point1=(set11+ds111+ds112, 0.0), point2=(set11+ds111+ds112,  
0.0035))
```

```
s.Line(point1=(set11+ds111+ds112+x113, 0.0),  
point2=(set11+ds111+ds112+x113, 0.00125))
```

```
s.Line(point1=(set11+ds111+ds112+ds113, 0.0),  
point2=(set11+ds111+ds112+ds113, 0.0035))
```

```
s.Line(point1=(set11+ds111+ds112+ds113+y114, 0.0),  
point2=(set11+ds111+ds112+ds113+y114, 0.00125))
```

```
s.Line(point1=(set11+ds111+ds112+ds113+ds114, 0.0),  
point2=(set11+ds111+ds112+ds113+ds114, 0.0035))
```

```
s.Line(point1=(set11+ds111+ds112+ds113+ds114+x115, 0.0),  
point2=(set11+ds111+ds112+ds113+ds114+x115, 0.00125))
```

```
s.Line(point1=(set11+ds111+ds112+ds113+ds114+ds115, 0.0),  
point2=(set11+ds111+ds112+ds113+ds114+ds115, 0.0035))
```

```
s.Line(point1=(set11+ds111+ds112+ds113+ds114+ds115+y116, 0.0),  
point2=(set11+ds111+ds112+ds113+ds114+ds115+y116, 0.00125))
```

```
s.Line(point1=(set11+ds111+ds112+ds113+ds114+ds115+ds116, 0.0),  
point2=(set11+ds111+ds112+ds113+ds114+ds115+ds116, 0.0035))
```

```
s.Line(point1=(set11+ds111+ds112+ds113+ds114+ds115+ds116+x117, 0.0),  
point2=(set11+ds111+ds112+ds113+ds114+ds115+ds116+x117, 0.00125))
```

```
s.Line(point1=(set11+ds111+ds112+ds113+ds114+ds115+ds116+ds117, 0.0),  
point2=(set11+ds111+ds112+ds113+ds114+ds115+ds116+ds117, 0.0035))
```

```
s.Line(point1=(set11+ds111+ds112+ds113+ds114+ds115+ds116+ds117+y118,  
0.0), point2=(set11+ds111+ds112+ds113+ds114+ds115+ds116+ds117+y118, 0.00125))
```

```
s.Line(point1=(set11+ds111+ds112+ds113+ds114+ds115+ds116+ds117+ds118,  
0.0), point2=(set11+ds111+ds112+ds113+ds114+ds115+ds116+ds117+ds118, 0.0035))
```

```
s.Line(point1=(set11+ds111+ds112+ds113+ds114+ds115+ds116+ds117+ds118+x119,  
0.0), point2=(set11+ds111+ds112+ds113+ds114+ds115+ds116+ds117+ds118+x119,  
0.00125))
```

```
s.Line(point1=(set11+ds111+ds112+ds113+ds114+ds115+ds116+ds117+ds118+ds119,  
0.0), point2=(set11+ds111+ds112+ds113+ds114+ds115+ds116+ds117+ds118+ds119,  
0.0035))
```

```
s.Line(point1=(set11+ds111+ds112+ds113+ds114+ds115+ds116+ds117+ds118+ds119+y120  
, 0.0),  
point2=(set11+ds111+ds112+ds113+ds114+ds115+ds116+ds117+ds118+ds119+y120,  
0.00125))
```

```
s.Line(point1=(set12, 0.0), point2=(set12, 0.0035))
```

#Bottom Row, Dspace 21-30

```
s.Line(point1=(set12+x121, 0.0), point2=(set12+x121, 0.00125))
```

```
s.Line(point1=(set12+ds121, 0.0), point2=(set12+ds121, 0.0035))
```

```
s.Line(point1=(set12+ds121+y122, 0), point2=(set12+ds121+y122, 0.00125))
```

```
s.Line(point1=(set12+ds121+ds122, 0.0), point2=(set12+ds121+ds122,  
0.0035))
```

```
s.Line(point1=(set12+ds121+ds122+x123, 0.0),  
point2=(set12+ds121+ds122+x123, 0.00125))
```

```
s.Line(point1=(set12+ds121+ds122+ds123, 0.0),  
point2=(set12+ds121+ds122+ds123, 0.0035))
```

```
s.Line(point1=(set12+ds121+ds122+ds123+y124, 0.0),  
point2=(set12+ds121+ds122+ds123+y124, 0.00125))
```

```
s.Line(point1=(set12+ds121+ds122+ds123+ds124, 0.0),  
point2=(set12+ds121+ds122+ds123+ds124, 0.0035))
```

```
s.Line(point1=(set12+ds121+ds122+ds123+ds124+x125, 0.0),  
point2=(set12+ds121+ds122+ds123+ds124+x125, 0.00125))
```

```
s.Line(point1=(set12+ds121+ds122+ds123+ds124+ds125, 0.0),  
point2=(set12+ds121+ds122+ds123+ds124+ds125, 0.0035))
```

```

        s.Line(point1=(set12+ds121+ds122+ds123+ds124+ds125+y126, 0.0),
point2=(set12+ds121+ds122+ds123+ds124+ds125+y126, 0.00125))

        s.Line(point1=(set12+ds121+ds122+ds123+ds124+ds125+ds126, 0.0),
point2=(set12+ds121+ds122+ds123+ds124+ds125+ds126, 0.0035))

        s.Line(point1=(set12+ds121+ds122+ds123+ds124+ds125+ds126+x127, 0.0),
point2=(set12+ds121+ds122+ds123+ds124+ds125+ds126+x127, 0.00125))

        s.Line(point1=(set12+ds121+ds122+ds123+ds124+ds125+ds126+ds127, 0.0),
point2=(set12+ds121+ds122+ds123+ds124+ds125+ds126+ds127, 0.0035))

        s.Line(point1=(set12+ds121+ds122+ds123+ds124+ds125+ds126+ds127+y128,
0.0), point2=(set12+ds121+ds122+ds123+ds124+ds125+ds126+ds127+y128, 0.00125))

        s.Line(point1=(set12+ds121+ds122+ds123+ds124+ds125+ds126+ds127+ds128,
0.0), point2=(set12+ds121+ds122+ds123+ds124+ds125+ds126+ds127+ds128, 0.0035))

s.Line(point1=(set12+ds121+ds122+ds123+ds124+ds125+ds126+ds127+ds128+x129,
0.0), point2=(set12+ds121+ds122+ds123+ds124+ds125+ds126+ds127+ds128+x129,
0.00125))

s.Line(point1=(set12+ds121+ds122+ds123+ds124+ds125+ds126+ds127+ds128+ds129,
0.0), point2=(set12+ds121+ds122+ds123+ds124+ds125+ds126+ds127+ds128+ds129,
0.0035))

s.Line(point1=(set12+ds121+ds122+ds123+ds124+ds125+ds126+ds127+ds128+ds129+y130
, 0.0),
point2=(set12+ds121+ds122+ds123+ds124+ds125+ds126+ds127+ds128+ds129+y130,
0.00125))

        s.Line(point1=(set13, 0.0), point2=(set13, 0.0035))

#Bottom Row, Dspace 31-40

        s.Line(point1=(set13+x131, 0.0), point2=(set13+x131, 0.00125))

        s.Line(point1=(set13+ds131, 0.0), point2=(set13+ds131, 0.0035))

        s.Line(point1=(set13+ds131+y132, 0), point2=(set13+ds131+y132, 0.00125))

        s.Line(point1=(set13+ds131+ds132, 0.0), point2=(set13+ds131+ds132,
0.0035))

        s.Line(point1=(set13+ds131+ds132+x133, 0.0),

```

```

point2=(set13+ds131+ds132+x133, 0.00125))

    s.Line(point1=(set13+ds131+ds132+ds133, 0.0),
point2=(set13+ds131+ds132+ds133, 0.0035))

    s.Line(point1=(set13+ds131+ds132+ds133+y134, 0.0),
point2=(set13+ds131+ds132+ds133+y134, 0.00125))

    s.Line(point1=(set13+ds131+ds132+ds133+ds134, 0.0),
point2=(set13+ds131+ds132+ds133+ds134, 0.0035))

    s.Line(point1=(set13+ds131+ds132+ds133+ds134+x135, 0.0),
point2=(set13+ds131+ds132+ds133+ds134+x135, 0.00125))

    s.Line(point1=(set13+ds131+ds132+ds133+ds134+ds135, 0.0),
point2=(set13+ds131+ds132+ds133+ds134+ds135, 0.0035))

    s.Line(point1=(set13+ds131+ds132+ds133+ds134+ds135+y136, 0.0),
point2=(set13+ds131+ds132+ds133+ds134+ds135+y136, 0.00125))

    s.Line(point1=(set13+ds131+ds132+ds133+ds134+ds135+ds136, 0.0),
point2=(set13+ds131+ds132+ds133+ds134+ds135+ds136, 0.0035))

    s.Line(point1=(set13+ds131+ds132+ds133+ds134+ds135+ds136+x137, 0.0),
point2=(set13+ds131+ds132+ds133+ds134+ds135+ds136+x137, 0.00125))

    s.Line(point1=(set13+ds131+ds132+ds133+ds134+ds135+ds136+ds137, 0.0),
point2=(set13+ds131+ds132+ds133+ds134+ds135+ds136+ds137, 0.0035))

    s.Line(point1=(set13+ds131+ds132+ds133+ds134+ds135+ds136+ds137+y138,
0.0), point2=(set13+ds131+ds132+ds133+ds134+ds135+ds136+ds137+y138, 0.00125))

    s.Line(point1=(set13+ds131+ds132+ds133+ds134+ds135+ds136+ds137+ds138,
0.0), point2=(set13+ds131+ds132+ds133+ds134+ds135+ds136+ds137+ds138, 0.0035))

s.Line(point1=(set13+ds131+ds132+ds133+ds134+ds135+ds136+ds137+ds138+x139,
0.0), point2=(set13+ds131+ds132+ds133+ds134+ds135+ds136+ds137+ds138+x139,
0.00125))

s.Line(point1=(set13+ds131+ds132+ds133+ds134+ds135+ds136+ds137+ds138+ds139,
0.0), point2=(set13+ds131+ds132+ds133+ds134+ds135+ds136+ds137+ds138+ds139,
0.0035))

s.Line(point1=(set13+ds131+ds132+ds133+ds134+ds135+ds136+ds137+ds138+ds139+y140
, 0.0),

```

```
point2=(set13+ds131+ds132+ds133+ds134+ds135+ds136+ds137+ds138+ds139+y140,  
0.00125))
```

```
s.Line(point1=(set14, 0.0), point2=(set14, 0.0035))
```

```
#Bottom Row, Dspace 41-50
```

```
s.Line(point1=(set14+x141, 0.0), point2=(set14+x141, 0.00125))
```

```
s.Line(point1=(set14+ds141, 0.0), point2=(set14+ds141, 0.0035))
```

```
s.Line(point1=(set14+ds141+y142, 0), point2=(set14+ds141+y142, 0.00125))
```

```
s.Line(point1=(set14+ds141+ds142, 0.0), point2=(set14+ds141+ds142,  
0.0035))
```

```
s.Line(point1=(set14+ds141+ds142+x143, 0.0),  
point2=(set14+ds141+ds142+x143, 0.00125))
```

```
s.Line(point1=(set14+ds141+ds142+ds143, 0.0),  
point2=(set14+ds141+ds142+ds143, 0.0035))
```

```
s.Line(point1=(set14+ds141+ds142+ds143+y144, 0.0),  
point2=(set14+ds141+ds142+ds143+y144, 0.00125))
```

```
s.Line(point1=(set14+ds141+ds142+ds143+ds144, 0.0),  
point2=(set14+ds141+ds142+ds143+ds144, 0.0035))
```

```
s.Line(point1=(set14+ds141+ds142+ds143+ds144+x145, 0.0),  
point2=(set14+ds141+ds142+ds143+ds144+x145, 0.00125))
```

```
s.Line(point1=(set14+ds141+ds142+ds143+ds144+ds145, 0.0),  
point2=(set14+ds141+ds142+ds143+ds144+ds145, 0.0035))
```

```
s.Line(point1=(set14+ds141+ds142+ds143+ds144+ds145+y146, 0.0),  
point2=(set14+ds141+ds142+ds143+ds144+ds145+y146, 0.00125))
```

```
s.Line(point1=(set14+ds141+ds142+ds143+ds144+ds145+ds146, 0.0),  
point2=(set14+ds141+ds142+ds143+ds144+ds145+ds146, 0.0035))
```

```
s.Line(point1=(set14+ds141+ds142+ds143+ds144+ds145+ds146+x147, 0.0),  
point2=(set14+ds141+ds142+ds143+ds144+ds145+ds146+x147, 0.00125))
```

```
s.Line(point1=(set14+ds141+ds142+ds143+ds144+ds145+ds146+ds147, 0.0),  
point2=(set14+ds141+ds142+ds143+ds144+ds145+ds146+ds147, 0.0035))
```

```
s.Line(point1=(set14+ds141+ds142+ds143+ds144+ds145+ds146+ds147+y148,  
0.0), point2=(set14+ds141+ds142+ds143+ds144+ds145+ds146+ds147+y148, 0.00125))
```

```
s.Line(point1=(set14+ds141+ds142+ds143+ds144+ds145+ds146+ds147+ds148,  
0.0), point2=(set14+ds141+ds142+ds143+ds144+ds145+ds146+ds147+ds148, 0.0035))
```

```
s.Line(point1=(set14+ds141+ds142+ds143+ds144+ds145+ds146+ds147+ds148+x149,  
0.0), point2=(set14+ds141+ds142+ds143+ds144+ds145+ds146+ds147+ds148+x149,  
0.00125))
```

```
s.Line(point1=(set14+ds141+ds142+ds143+ds144+ds145+ds146+ds147+ds148+ds149,  
0.0), point2=(set14+ds141+ds142+ds143+ds144+ds145+ds146+ds147+ds148+ds149,  
0.0035))
```

```
s.Line(point1=(set14+ds141+ds142+ds143+ds144+ds145+ds146+ds147+ds148+ds149+y150,  
0.0),  
point2=(set14+ds141+ds142+ds143+ds144+ds145+ds146+ds147+ds148+ds149+y150,  
0.00125))
```

```
s.Line(point1=(set15, 0.0), point2=(set15, 0.0035))
```

#Bottom Row, Dspace 51-60

```
s.Line(point1=(set15+x151, 0.0), point2=(set15+x151, 0.00125))
```

```
s.Line(point1=(set15+ds151, 0.0), point2=(set15+ds151, 0.0035))
```

```
s.Line(point1=(set15+ds151+y152, 0), point2=(set15+ds151+y152, 0.00125))
```

```
s.Line(point1=(set15+ds151+ds152, 0.0), point2=(set15+ds151+ds152,  
0.0035))
```

```
s.Line(point1=(set15+ds151+ds152+x153, 0.0),  
point2=(set15+ds151+ds152+x153, 0.00125))
```

```
s.Line(point1=(set15+ds151+ds152+ds153, 0.0),  
point2=(set15+ds151+ds152+ds153, 0.0035))
```

```
s.Line(point1=(set15+ds151+ds152+ds153+y154, 0.0),  
point2=(set15+ds151+ds152+ds153+y154, 0.00125))
```

```
s.Line(point1=(set15+ds151+ds152+ds153+ds154, 0.0),  
point2=(set15+ds151+ds152+ds153+ds154, 0.0035))
```

```
s.Line(point1=(set15+ds151+ds152+ds153+ds154+x155, 0.0),  
point2=(set15+ds151+ds152+ds153+ds154+x155, 0.00125))
```

```

        s.Line(point1=(set15+ds151+ds152+ds153+ds154+ds155, 0.0),
point2=(set15+ds151+ds152+ds153+ds154+ds155, 0.0035))

        s.Line(point1=(set15+ds151+ds152+ds153+ds154+ds155+y156, 0.0),
point2=(set15+ds151+ds152+ds153+ds154+ds155+y156, 0.00125))

        s.Line(point1=(set15+ds151+ds152+ds153+ds154+ds155+ds156, 0.0),
point2=(set15+ds151+ds152+ds153+ds154+ds155+ds156, 0.0035))

        s.Line(point1=(set15+ds151+ds152+ds153+ds154+ds155+ds156+x157, 0.0),
point2=(set15+ds151+ds152+ds153+ds154+ds155+ds156+x157, 0.00125))

        s.Line(point1=(set15+ds151+ds152+ds153+ds154+ds155+ds156+ds157, 0.0),
point2=(set15+ds151+ds152+ds153+ds154+ds155+ds156+ds157, 0.0035))

        s.Line(point1=(set15+ds151+ds152+ds153+ds154+ds155+ds156+ds157+y158,
0.0), point2=(set15+ds151+ds152+ds153+ds154+ds155+ds156+ds157+y158, 0.00125))

        s.Line(point1=(set15+ds151+ds152+ds153+ds154+ds155+ds156+ds157+ds158,
0.0), point2=(set15+ds151+ds152+ds153+ds154+ds155+ds156+ds157+ds158, 0.0035))

s.Line(point1=(set15+ds151+ds152+ds153+ds154+ds155+ds156+ds157+ds158+x159,
0.0), point2=(set15+ds151+ds152+ds153+ds154+ds155+ds156+ds157+ds158+x159,
0.00125))

s.Line(point1=(set15+ds151+ds152+ds153+ds154+ds155+ds156+ds157+ds158+ds159,
0.0), point2=(set15+ds151+ds152+ds153+ds154+ds155+ds156+ds157+ds158+ds159,
0.0035))

s.Line(point1=(set15+ds151+ds152+ds153+ds154+ds155+ds156+ds157+ds158+ds159+y160
, 0.0),
point2=(set15+ds151+ds152+ds153+ds154+ds155+ds156+ds157+ds158+ds159+y160,
0.00125))

        s.Line(point1=(set16, 0.0), point2=(set16, 0.0035))

#Bottom Row, Dspace 61–70

        s.Line(point1=(set16+x161, 0.0), point2=(set16+x161, 0.00125))

        s.Line(point1=(set16+ds161, 0.0), point2=(set16+ds161, 0.0035))

        s.Line(point1=(set16+ds161+y162, 0), point2=(set16+ds161+y162, 0.00125))

        s.Line(point1=(set16+ds161+ds162, 0.0), point2=(set16+ds161+ds162,

```


0.0035))

s.Line(point1=(set16+ds161+ds162+x163, 0.0),
point2=(set16+ds161+ds162+x163, 0.00125))

s.Line(point1=(set16+ds161+ds162+ds163, 0.0),
point2=(set16+ds161+ds162+ds163, 0.0035))

s.Line(point1=(set16+ds161+ds162+ds163+y164, 0.0),
point2=(set16+ds161+ds162+ds163+y164, 0.00125))

s.Line(point1=(set16+ds161+ds162+ds163+ds164, 0.0),
point2=(set16+ds161+ds162+ds163+ds164, 0.0035))

s.Line(point1=(set16+ds161+ds162+ds163+ds164+x165, 0.0),
point2=(set16+ds161+ds162+ds163+ds164+x165, 0.00125))

s.Line(point1=(set16+ds161+ds162+ds163+ds164+ds165, 0.0),
point2=(set16+ds161+ds162+ds163+ds164+ds165, 0.0035))

s.Line(point1=(set16+ds161+ds162+ds163+ds164+ds165+y166, 0.0),
point2=(set16+ds161+ds162+ds163+ds164+ds165+y166, 0.00125))

s.Line(point1=(set16+ds161+ds162+ds163+ds164+ds165+ds166, 0.0),
point2=(set16+ds161+ds162+ds163+ds164+ds165+ds166, 0.0035))

s.Line(point1=(set16+ds161+ds162+ds163+ds164+ds165+ds166+x167, 0.0),
point2=(set16+ds161+ds162+ds163+ds164+ds165+ds166+x167, 0.00125))

s.Line(point1=(set16+ds161+ds162+ds163+ds164+ds165+ds166+ds167, 0.0),
point2=(set16+ds161+ds162+ds163+ds164+ds165+ds166+ds167, 0.0035))

s.Line(point1=(set16+ds161+ds162+ds163+ds164+ds165+ds166+ds167+y168, 0.0),
point2=(set16+ds161+ds162+ds163+ds164+ds165+ds166+ds167+y168, 0.00125))

s.Line(point1=(set16+ds161+ds162+ds163+ds164+ds165+ds166+ds167+ds168,
0.0), point2=(set16+ds161+ds162+ds163+ds164+ds165+ds166+ds167+ds168, 0.0035))

s.Line(point1=(set16+ds161+ds162+ds163+ds164+ds165+ds166+ds167+ds168+x169,
0.0), point2=(set16+ds161+ds162+ds163+ds164+ds165+ds166+ds167+ds168+x169,
0.00125))

s.Line(point1=(set16+ds161+ds162+ds163+ds164+ds165+ds166+ds167+ds168+ds169,
0.0), point2=(set16+ds161+ds162+ds163+ds164+ds165+ds166+ds167+ds168+ds169,
0.0035))

```
s.Line(point1=(set16+ds161+ds162+ds163+ds164+ds165+ds166+ds167+ds168+ds169+y170, 0.0),  
point2=(set16+ds161+ds162+ds163+ds164+ds165+ds166+ds167+ds168+ds169+y170, 0.00125))
```

```
s.Line(point1=(set17, 0.0), point2=(set17, 0.0035))
```

```
#Bottom Row, Dspace 71-80
```

```
s.Line(point1=(set17+x171, 0.0), point2=(set17+x171, 0.00125))
```

```
s.Line(point1=(set17+ds171, 0.0), point2=(set17+ds171, 0.0035))
```

```
s.Line(point1=(set17+ds171+y172, 0), point2=(set17+ds171+y172, 0.00125))
```

```
s.Line(point1=(set17+ds171+ds172, 0.0), point2=(set17+ds171+ds172, 0.0035))
```

```
s.Line(point1=(set17+ds171+ds172+x173, 0.0),  
point2=(set17+ds171+ds172+x173, 0.00125))
```

```
s.Line(point1=(set17+ds171+ds172+ds173, 0.0),  
point2=(set17+ds171+ds172+ds173, 0.0035))
```

```
s.Line(point1=(set17+ds171+ds172+ds173+y174, 0.0),  
point2=(set17+ds171+ds172+ds173+y174, 0.00125))
```

```
s.Line(point1=(set17+ds171+ds172+ds173+ds174, 0.0),  
point2=(set17+ds171+ds172+ds173+ds174, 0.0035))
```

```
s.Line(point1=(set17+ds171+ds172+ds173+ds174+x175, 0.0),  
point2=(set17+ds171+ds172+ds173+ds174+x175, 0.00125))
```

```
s.Line(point1=(set17+ds171+ds172+ds173+ds174+ds175, 0.0),point2=(set17+ds171+ds172+ds173+ds174+ds175, 0.0035))
```

```
s.Line(point1=(set17+ds171+ds172+ds173+ds174+ds175+y176, 0.0),  
point2=(set17+ds171+ds172+ds173+ds174+ds175+y176, 0.00125))
```

```
s.Line(point1=(set17+ds171+ds172+ds173+ds174+ds175+ds176, 0.0),  
point2=(set17+ds171+ds172+ds173+ds174+ds175+ds176, 0.0035))
```

```
s.Line(point1=(set17+ds171+ds172+ds173+ds174+ds175+ds176+x177, 0.0),  
point2=(set17+ds171+ds172+ds173+ds174+ds175+ds176+x177, 0.00125))
```

```
s.Line(point1=(set17+ds171+ds172+ds173+ds174+ds175+ds176+ds177, 0.0),  
point2=(set17+ds171+ds172+ds173+ds174+ds175+ds176+ds177, 0.0035))
```

```
s.Line(point1=(set17+ds171+ds172+ds173+ds174+ds175+ds176+ds177+y178,  
0.0), point2=(set17+ds171+ds172+ds173+ds174+ds175+ds176+ds177+y178, 0.00125))
```

```
s.Line(point1=(set17+ds171+ds172+ds173+ds174+ds175+ds176+ds177+ds178,  
0.0), point2=(set17+ds171+ds172+ds173+ds174+ds175+ds176+ds177+ds178, 0.0035))
```

```
s.Line(point1=(set17+ds171+ds172+ds173+ds174+ds175+ds176+ds177+ds178+x179,  
0.0), point2=(set17+ds171+ds172+ds173+ds174+ds175+ds176+ds177+ds178+x179,  
0.00125))
```

```
s.Line(point1=(set17+ds171+ds172+ds173+ds174+ds175+ds176+ds177+ds178+ds179,  
0.0), point2=(set17+ds171+ds172+ds173+ds174+ds175+ds176+ds177+ds178+ds179,  
0.0035))
```

```
s.Line(point1=(set17+ds171+ds172+ds173+ds174+ds175+ds176+ds177+ds178+ds179+y180,  
0.0),  
point2=(set17+ds171+ds172+ds173+ds174+ds175+ds176+ds177+ds178+ds179+y180,  
0.00125))
```

```
s.Line(point1=(set18, 0.0), point2=(set18, 0.0035))
```

#Bottom Row, Dspace 81-90

```
s.Line(point1=(set18+x181, 0.0), point2=(set18+x181, 0.00125))
```

```
s.Line(point1=(set18+ds181, 0.0), point2=(set18+ds181, 0.0035))
```

```
s.Line(point1=(set18+ds181+y182, 0.0), point2=(set18+ds181+y182,  
0.00125))
```

```
s.Line(point1=(set18+ds181+ds182, 0.0), point2=(set18+ds181+ds182,  
0.0035))
```

```
s.Line(point1=(set18+ds181+ds182+x183, 0.0),  
point2=(set18+ds181+ds182+x183, 0.00125))
```

```
s.Line(point1=(set18+ds181+ds182+ds183, 0.0),  
point2=(set18+ds181+ds182+ds183, 0.0035))
```

```
s.Line(point1=(set18+ds181+ds182+ds183+y184, 0.0),  
point2=(set18+ds181+ds182+ds183+y184, 0.00125))
```

```
s.Line(point1=(set18+ds181+ds182+ds183+ds184, 0.0),  
point2=(set18+ds181+ds182+ds183+ds184, 0.0035))
```

```

        s.Line(point1=(set18+ds181+ds182+ds183+ds184+x185, 0.0),
point2=(set18+ds181+ds182+ds183+ds184+x185, 0.00125))

        s.Line(point1=(set18+ds181+ds182+ds183+ds184+ds185, 0.0),
point2=(set18+ds181+ds182+ds183+ds184+ds185, 0.0035))

        s.Line(point1=(set18+ds181+ds182+ds183+ds184+ds185+y186, 0.0),
point2=(set18+ds181+ds182+ds183+ds184+ds185+y186, 0.00125))

        s.Line(point1=(set18+ds181+ds182+ds183+ds184+ds185+ds186, 0.0),
point2=(set18+ds181+ds182+ds183+ds184+ds185+ds186, 0.0035))

        s.Line(point1=(set18+ds181+ds182+ds183+ds184+ds185+ds186+x187, 0.0),
point2=(set18+ds181+ds182+ds183+ds184+ds185+ds186+x187, 0.00125))

        s.Line(point1=(set18+ds181+ds182+ds183+ds184+ds185+ds186+ds187, 0.0),
point2=(set18+ds181+ds182+ds183+ds184+ds185+ds186+ds187, 0.0035))

        s.Line(point1=(set18+ds181+ds182+ds183+ds184+ds185+ds186+ds187+y188,
0.0), point2=(set18+ds181+ds182+ds183+ds184+ds185+ds186+ds187+y188, 0.00125))

        s.Line(point1=(set18+ds181+ds182+ds183+ds184+ds185+ds186+ds187+ds188,
0.0), point2=(set18+ds181+ds182+ds183+ds184+ds185+ds186+ds187+ds188, 0.0035))

s.Line(point1=(set18+ds181+ds182+ds183+ds184+ds185+ds186+ds187+ds188+x189,
0.0), point2=(set18+ds181+ds182+ds183+ds184+ds185+ds186+ds187+ds188+x189,
0.00125))

s.Line(point1=(set18+ds181+ds182+ds183+ds184+ds185+ds186+ds187+ds188+ds189,
0.0), point2=(set18+ds181+ds182+ds183+ds184+ds185+ds186+ds187+ds188+ds189,
0.0035))

s.Line(point1=(set18+ds181+ds182+ds183+ds184+ds185+ds186+ds187+ds188+ds189+y190
, 0.0),
point2=(set18+ds181+ds182+ds183+ds184+ds185+ds186+ds187+ds188+ds189+y190,
0.00125))

        s.Line(point1=(set19, 0.0), point2=(set19, 0.0035))

#Bottom Row, Dspace 91-100

        s.Line(point1=(set19+x191, 0.0), point2=(set19+x191, 0.00125))

        s.Line(point1=(set19+ds191, 0.0), point2=(set19+ds191, 0.0035))

```

```

s.Line(point1=(set19+ds191+y192, 0), point2=(set19+ds191+y192, 0.00125))

s.Line(point1=(set19+ds191+ds192, 0.0), point2=(set19+ds191+ds192,
0.0035))

s.Line(point1=(set19+ds191+ds192+x193, 0.0),
point2=(set19+ds191+ds192+x193, 0.00125))

s.Line(point1=(set19+ds191+ds192+ds193, 0.0),
point2=(set19+ds191+ds192+ds193, 0.0035))

s.Line(point1=(set19+ds191+ds192+ds193+y194, 0.0),
point2=(set19+ds191+ds192+ds193+y194, 0.00125))

s.Line(point1=(set19+ds191+ds192+ds193+ds194, 0.0),
point2=(set19+ds191+ds192+ds193+ds194, 0.0035))

s.Line(point1=(set19+ds191+ds192+ds193+ds194+x195, 0.0),
point2=(set19+ds191+ds192+ds193+ds194+x195, 0.00125))

s.Line(point1=(set19+ds191+ds192+ds193+ds194+ds195, 0.0),
point2=(set19+ds191+ds192+ds193+ds194+ds195, 0.0035))

s.Line(point1=(set19+ds191+ds192+ds193+ds194+ds195+y196, 0.0),
point2=(set19+ds191+ds192+ds193+ds194+ds195+y196, 0.00125))

s.Line(point1=(set19+ds191+ds192+ds193+ds194+ds195+ds196, 0.0),
point2=(set19+ds191+ds192+ds193+ds194+ds195+ds196, 0.0035))

s.Line(point1=(set19+ds191+ds192+ds193+ds194+ds195+ds196+x197, 0.0),
point2=(set19+ds191+ds192+ds193+ds194+ds195+ds196+x197, 0.00125))

s.Line(point1=(set19+ds191+ds192+ds193+ds194+ds195+ds196+ds197, 0.0),
point2=(set19+ds191+ds192+ds193+ds194+ds195+ds196+ds197, 0.0035))

s.Line(point1=(set19+ds191+ds192+ds193+ds194+ds195+ds196+ds197+y198,
0.0), point2=(set19+ds191+ds192+ds193+ds194+ds195+ds196+ds197+y198, 0.00125))

s.Line(point1=(set19+ds191+ds192+ds193+ds194+ds195+ds196+ds197+ds198,
0.0), point2=(set19+ds191+ds192+ds193+ds194+ds195+ds196+ds197+ds198, 0.0035))

s.Line(point1=(set19+ds191+ds192+ds193+ds194+ds195+ds196+ds197+ds198+x199,
0.0), point2=(set19+ds191+ds192+ds193+ds194+ds195+ds196+ds197+ds198+x199,
0.00125))

```

```
s.Line(point1=(set19+ds191+ds192+ds193+ds194+ds195+ds196+ds197+ds198+ds199,
0.0), point2=(set19+ds191+ds192+ds193+ds194+ds195+ds196+ds197+ds198+ds199,
0.0035))
```

```
s.Line(point1=(set19+ds191+ds192+ds193+ds194+ds195+ds196+ds197+ds198+ds199+y200,
0.0),
point2=(set19+ds191+ds192+ds193+ds194+ds195+ds196+ds197+ds198+ds199+y200,
0.00125))
```

```
s.Line(point1=(set20, 0.0), point2=(set20, 0.0035))
```

#SPACER SUBSTITUTION

```
print 'Bottom Row larger than Top Row'
print ('Dspace100 =', ds100)
LengthS=LengthF-(set10)
spacer=DSmean-1*DSstdev
hys=0.84*spacer
LengthS2=LengthS-spacer
LengthS3=LengthS2-spacer
LengthS4=LengthS3-spacer
LengthS5=LengthS4-spacer
LengthS6=LengthS5-spacer
LengthS7=LengthS6-spacer
LengthS8=LengthS7-spacer
```

#Spacer Remainder--is the model still biologically valid?

```
spacerremainder=LengthF-(set10)
NewArea=(ds100+spacerremainder)*(0.0035)
NewRatio=(ds100*0.84)*(1.25E-3)/(NewArea)
if NewRatio < .25:
    print 'REJECT MODEL: Currently Biologically Invalid'
if NewRatio >= .25:
    print '*MODEL IS NOW VALID; Good for Analysis*'
```

```
if LengthS > spacer:
```

```
    print 'Added FIRST spacer to top row'
```

#creates boundary line for new DSpace in spacer substitution case

```
s.Line(point1=(set10+spacer, 0.007), point2=(set10+spacer, 0.0035))
```

#creates hydrox line for new DSpace in spacer substitution case

```
s.Line(point1=(set10+hys, 0.007), point2=(set10+hys, 0.00575))
```

#Spacer Remainder--is the model still biologically valid?

```
spacerremainder=LengthF-(set10+spacer)
NewArea=(spacer+spacerremainder)*(0.0035)
NewRatio=(spacer*0.84)*(1.25E-3)/(NewArea)
if NewRatio < .25:
    print 'REJECT MODEL: Currently Biologically Invalid'
if NewRatio >= .25:
```

```

    print '*MODEL IS NOW VALID; Good for Analysis*'

    if LengthS2 > spacer:
        print 'Added SECOND spacer to top row'
        #creates boundary line for new DSpace in SECOND spacer substitution case
        s.Line(point1=(set10+spacer+spacer, 0.007), point2=(set10+spacer+spacer,
0.0035))
        #creates hydrox line for new DSpace in SECOND spacer substitution case
        s.Line(point1=(set10+spacer+spacer-hys, 0.007),
point2=(set10+spacer+spacer-hys, 0.00575))
        #Spacer Remainder--is the model still biologically valid?
        spacerremainder=LengthF-(set10+spacer+spacer)
        NewArea=(spacer+spacerremainder)*(0.0035)
        NewRatio=(spacer*0.84)*(1.25E-3)/(NewArea)
        if NewRatio < .25:
            print 'REJECT MODEL: Currently Biologically Invalid'
        if NewRatio >= .25:
            print '*MODEL IS NOW VALID; Good for Analysis*'

    if LengthS3 > spacer:
        print 'Added THIRD spacer to top row'
        #creates boundary line for new DSpace in THIRD spacer substitution case
        s.Line(point1=(set10+spacer+spacer+spacer, 0.007),
point2=(set10+spacer+spacer+spacer, 0.0035))
        #creates hydrox line for new DSpace in THIRD spacer substitution case
        s.Line(point1=(set10+spacer+spacer+hys, 0.007),
point2=(set10+spacer+spacer+hys, 0.00575))
        #Spacer Remainder--is the model still biologically valid?
        spacerremainder=LengthF-(set10+spacer+spacer+spacer)
        NewArea=(spacer+spacerremainder)*(0.0035)
        NewRatio=(spacer*0.84)*(1.25E-3)/(NewArea)
        if NewRatio < .25:
            print 'REJECT MODEL: Currently Biologically Invalid'
        if NewRatio >= .25:
            print '*MODEL IS NOW VALID; Good for Analysis*'

    if LengthS4 > spacer:
        print 'Added FOURTH spacer to top row'
        #creates boundary line for new DSpace in FOURTH spacer substitution case
        s.Line(point1=(set10+spacer+spacer+spacer+spacer, 0.007),
point2=(set10+spacer+spacer+spacer+spacer, 0.0035))
        #creates hydrox line for new DSpace in FOURTH spacer substitution case
        s.Line(point1=(set10+spacer+spacer+spacer+spacer-hys, 0.007),
point2=(set10+spacer+spacer+spacer+spacer-hys, 0.00575))
        #Spacer Remainder--is the model still biologically valid?
        spacerremainder=LengthF-(set10+spacer+spacer+spacer+spacer)
        NewArea=(spacer+spacerremainder)*(0.0035)
        NewRatio=(spacer*0.84)*(1.25E-3)/(NewArea)

```

```

if NewRatio < .25:
    print 'REJECT MODEL: Currently Biologically Invalid'
if NewRatio >= .25:
    print '*MODEL IS NOW VALID; Good for Analysis*'

if LengthS5 > spacer:
    print 'Added FIFTH spacer to top row'
    #creates boundary line for new DSpace in FIFTH spacer substitution case
    s.Line(point1=(set10+spacer+spacer+spacer+spacer+spacer, 0.007),
point2=(set10+spacer+spacer+spacer+spacer+spacer, 0.0035))
    #creates hydrox line for new DSpace in FIFTH spacer substitution case
    s.Line(point1=(set10+spacer+spacer+spacer+spacer+hys, 0.007),
point2=(set10+spacer+spacer+spacer+spacer+hys, 0.00575))
    #Spacer Remainder—is the model still biologically valid?
    spacerremainder=LengthF-(set10+spacer+spacer+spacer+spacer+spacer)
    NewArea=(spacer+spacerremainder)*(0.0035)
    NewRatio=(spacer*0.84)*(1.25E-3)/(NewArea)
    if NewRatio < .25:
        print 'REJECT MODEL: Currently Biologically Invalid'
    if NewRatio >= .25:
        print '*MODEL IS NOW VALID; Good for Analysis*'

if LengthS6 > spacer:
    print 'Added SIXTH spacer to top row'
    #creates boundary line for new DSpace in SIXTH spacer substitution case
    s.Line(point1=(set10+spacer+spacer+spacer+spacer+spacer+spacer, 0.007),
point2=(set10+spacer+spacer+spacer+spacer+spacer+spacer, 0.0035))
    #creates hydrox line for new DSpace in SIXTH spacer substitution case
    s.Line(point1=(set10+spacer+spacer+spacer+spacer+spacer+spacer+spacer-hys,
0.007), point2=(set10+spacer+spacer+spacer+spacer+spacer+spacer+spacer-hys, 0.00575))
    #Spacer Remainder—is the model still biologically valid?

spacerremainder=LengthF-(set10+spacer+spacer+spacer+spacer+spacer+spacer)
    NewArea=(spacer+spacerremainder)*(0.0035)
    NewRatio=(spacer*0.84)*(1.25E-3)/(NewArea)
    if NewRatio < .25:
        print 'REJECT MODEL: Currently Biologically Invalid'
    if NewRatio >= .25:
        print '*MODEL IS NOW VALID; Good for Analysis*'

if LengthS7 > spacer:
    print 'Added SEVENTH spacer to top row'
    #creates boundary line for new DSpace in SEVENTH spacer substitution case
    s.Line(point1=(set10+spacer+spacer+spacer+spacer+spacer+spacer+spacer+spacer,
0.007), point2=(set10+spacer+spacer+spacer+spacer+spacer+spacer+spacer+spacer,
0.0035))
    #creates hydrox line for new DSpace in SEVENTH spacer substitution case
    s.Line(point1=(set10+spacer+spacer+spacer+spacer+spacer+spacer+spacer+hys,

```



```

0.007), point2=(set10+spacer+spacer+spacer+spacer+spacer+spacer+hys, 0.00575))
    #Spacer Remainder--is the model still biologically valid?

spacerremainder=LengthF-(set10+spacer+spacer+spacer+spacer+spacer+spacer+spacer+spacer
)
    NewArea=(spacer+spacerremainder)*(0.0035)
    NewRatio=(spacer*0.84)*(1.25E-3)/(NewArea)
    if NewRatio < .25:
        print 'REJECT MODEL: Currently Biologically Invalid'
    if NewRatio >= .25:
        print '*MODEL IS NOW VALID; Good for Analysis*'
    if LengthS8 > spacer:
        print 'Added EIGHT spacer to top row AGAINST ALL ODDS WHY SO MANY
SPACERS GOODNESS'

    #creates boundary line for new DSpace in EIGHT spacer substitution case

s.Line(point1=(set10+spacer+spacer+spacer+spacer+spacer+spacer+spacer+spacer,
0.007), point2=(set10+spacer+spacer+spacer+spacer+spacer+spacer+spacer+spacer,
0.0035))
    #creates hydrox line for new DSpace in EIGHT spacer substitution case

s.Line(point1=(set10+spacer+spacer+spacer+spacer+spacer+spacer+spacer+spacer-
hys, 0.007),
point2=(set10+spacer+spacer+spacer+spacer+spacer+spacer+spacer+spacer-hys,
0.00575))
    #Spacer Remainder--is the model still biologically valid?

spacerremainder=LengthF-(set10+spacer+spacer+spacer+spacer+spacer+spacer+spacer+spacer
+spacer)
    NewArea=(spacer+spacerremainder)*(0.0035)
    NewRatio=(spacer*0.84)*(1.25E-3)/(NewArea)
    if NewRatio < .25:
        print 'REJECT MODEL: Currently Biologically Invalid'
    if NewRatio >= .25:
        print '*MODEL IS NOW VALID; Good for Analysis*'

    p = mdb.models['Model-1'].parts['Composite Bone']
    f = p.faces
    pickedFaces = f.getSequenceFromMask(mask=('[#1 ]', ), )
    e, d1 = p.edges, p.datums
    p.PartitionFaceBySketch(faces=pickedFaces, sketch=s)
    s.unsetPrimaryObject()
    del mdb.models['Model-1'].sketches['__profile__']

#MATERIAL CREATION
mdb.models['Model-1'].Material(name='COLLAGEN')
mdb.models['Model-1'].materials['COLLAGEN'].Depvar(n=3)

```

```

mdb.models['Model-1'].materials['COLLAGEN'].UserMaterial(mechanicalConstants=(
    0.003, 0.006, 0.004, 0.2, 0.2, 0.2))
mdb.models['Model-1'].Material(name='HYDROXYAPATITE')
mdb.models['Model-1'].materials['HYDROXYAPATITE'].Elastic(table=((0.1,
0.28),
    ))
mdb.models['Model-1'].HomogeneousSolidSection(name='COLLAGEN SECTION',
    material='COLLAGEN', thickness=None)
mdb.models['Model-1'].HomogeneousSolidSection(name='HYDROXYAPATITE
SECTION',
    material='HYDROXYAPATITE', thickness=None)

#INSTANCE SET AND XSYM MAKER
session.viewports['Viewport: 1'].assemblyDisplay.setValues(loads=OFF,
bcs=OFF,
    predefinedFields=OFF, connectors=OFF)
a = mdb.models['Model-1'].rootAssembly
a.DatumCsysByDefault(CARTESIAN)
p = mdb.models['Model-1'].parts['Composite Bone']
a.Instance(name='Composite Bone-1', part=p, dependent=ON)
session.viewports['Viewport: 1'].assemblyDisplay.setValues(
    adaptiveMeshConstraints=ON)
mdb.models['Model-1'].StaticStep(name='APPLY LOAD', previous='Initial',
    timePeriod=0.05, maxNumInc=100000, initialInc=0.05, minInc=1e-10,
    maxInc=0.05)
session.viewports['Viewport: 1'].assemblyDisplay.setValues(
    step='APPLY LOAD')
session.viewports['Viewport: 1'].assemblyDisplay.setValues(loads=ON,
bcs=ON,
    predefinedFields=ON, connectors=ON, adaptiveMeshConstraints=OFF)
mdb.models['Model-1'].PeriodicAmplitude(name='SINUSOIDAL', timeSpan=TOTAL,
    frequency=6.28319, start=0.0, a_0=0.6875, data=((0.0, 0.3125), ))
session.viewports['Viewport: 1'].view.setValues(nearPlane=12.8063,
    farPlane=12.8435, width=0.201952, height=0.110266, viewOffsetX=3.20047,
    viewOffsetY=-0.00229728)
session.viewports['Viewport: 1'].view.setValues(nearPlane=12.8134,
    farPlane=12.8364, width=0.1195, height=0.065247, viewOffsetX=3.20271,
    viewOffsetY=-0.00322843)
a = mdb.models['Model-1'].rootAssembly
s1 = a.instances['Composite Bone-1'].edges
side1Edges1 = s1.getSequenceFromMask(mask=(
    '[#0:58 #4000000 #8000 #20000 #80000240 ]', ), )
region = regionToolset.Region(side1Edges=side1Edges1)
mdb.models['Model-1'].Pressure(name='PRESSURE', createStepName='APPLY
LOAD',
    region=region, distributionType=UNIFORM, field='', magnitude=-3.36e-06,
    amplitude='SINUSOIDAL')

```

```
session.viewports['Viewport: 1'].view.fitView()
session.viewports['Viewport: 1'].view.setValues(nearPlane=12.4455,
    farPlane=13.2043, width=3.94297, height=2.15286, viewOffsetX=-1.23672,
    viewOffsetY=-0.023348)
session.viewports['Viewport: 1'].view.setValues(nearPlane=12.8161,
    farPlane=12.8338, width=0.096555, height=0.0527191,
    viewOffsetX=-3.20778, viewOffsetY=0.000928941)
a = mdb.models['Model-1'].rootAssembly
e1 = a.instances['Composite Bone-1'].edges
edges1 = e1.getSequenceFromMask(mask=('[#42008020 #80004 ]', ), )
region = regionToolset.Region(edges=edges1)
mdb.models['Model-1'].XsymmBC(name='XSYM', createStepName='APPLY LOAD',
    region=region)
```