A FINITE ELEMENT ANALYSIS ON THE VISCOELASTICITY OF POSTMENOPAUSAL COMPACT BONE UTILIZING A COMPLEX COLLAGEN D-SPACING MODEL

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

Austin Cleary Cummings

June 2015

COMMITTEE MEMBERSHIP

TITLE:	A Finite Element Analysis on the Viscoelasticity of Postmenopausal Compact Bone Utilizing a Complex Collagen D-spacing Model
AUTHOR:	Austin Cleary Cummings
DATE SUBMITTED:	June 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:	David Clague, Ph.D. Associate Professor of Biomedical Engineering

ABSTRACT

A Finite Element Analysis on the Viscoelasticity of Postmenopausal Compact Bone Utilizing a Complex Collagen D-spacing Model

Austin Cleary Cummings

The nanoscale dimension known as D-spacing describes the staggering of collagen molecules, which are fundamental to the biphasic makeup of bone tissue. This dimension was long assumed to be constant, but recent studies have shown that the periodicity of collagen is variable. Given that the arrangement of collagen molecules is closely related to the degree of bone mineralization, recent studies have begun to look at D-spacing as a potential factor in the ongoing effort to battle postmenopausal osteoporosis. The theoretical models presented by previous studies have only opted to model a single collagen-hydroxyapatite period, so the creation of an intricate computational approach that more exhaustively models a network of collagen and mineral is well-warranted.

Sheep present an excellent opportunity to examine metabolic disorders, as their bone structure similar to that of the human skeleton. Six Rambouillet-cross ewes were used for the purpose of gathering experimental data. Three ewes underwent a sham surgery (controls), while an ovariectomy (OVX) was performed on the remaining three sheep. Each sheep was sacrificed after 12 months and their radius and ulna were harvested for atomic force microscopy and mechanical testing. Each sheep bone produced up to 25 beam samples that were available for analysis, and two were randomly selected from each test sheep. The cranial anatomical sector was selected for testing as it replicates the tensile loading condition characteristically experienced by collagen molecules and its exclusive examination removes any unintended variation due to bone section.

Experimental D-spacing measurements were used in a finite element software, Abaqus, to create the "Complex Model": a large-scale, 2-D staggered array representation of collagen and hydroxyapatite periodicity. D-spacings intrinsic variability was mimicked through a Gaussian distribution that randomly determined periodic lengths based on provided experimental data. The model was generated with these random conditions for 2 x 100 units. Safeguards were implemented to ensure appropriate ratios of collagen to hydroxyapatite throughout the randomization. Collagen was assigned viscoelastic material properties originally developed by Dr. Frank Richter and modified by Miguel Mendoza. Hydroxyapatite was modeled as an elastic isotropic material. Four models were created using randomized D-spacings from control sheep and four separate models were created based on OVX sheep. Tangent delta–a damping characteristic–was recorded to evaluate bone viscoelasticity across four test frequencies: 1, 3, 9, and 15 Hz.

Results strongly suggest that the Complex Model matches experimental findings more accurately than previous computational approaches. These results indicate the complicated network of many collagen units is an essential parameter of adequate modeling. A repeated measures analysis of variance was performed to examine the differences between control and OVX sheep. After adjusting for all other predictors, at the 1% significance level, after

adjusting for all other variables, there is not enough evidence to convince this study that the Surgical Treatment alone has a significant impact on output tangent delta. This finding leads this study to conclude that OVX is fully accounted for within the Complex Model through the inclusion of its D-spacing, and the answers to bone's complicated mechanical properties during estrogen loss may lie in how OVX changes collagen viscoelasticity.

Significant interactions were found between the Model Type and the Test Frequency. A Tukey-Kramer pairwise comparison was performed between Complex and Experimental data, which determined the Complex Model did not behave statistically differently from experimental findings at 15 Hz. This result suggests the Complex Model may begin to be validated to experimental results in a statistically meaningfully way that is a first for this style of FEA approach.

The flexibility implemented in the randomization of the Complex Model welcomes refinement primarily in modeling viscoelasticity and fine-tuning the representation of mineralization. Through adjusting these material characteristics, the Complex Model may become an even more powerful tool in examining bone viscoelasticity and metabolic disorders.

Keywords: D-spacing, Menopause, Compact Bone, Viscoelasticity, Biomechanics, Finite Element Analysis

ACKNOWLEDGMENTS

Thank you to Dr. Scott Hazelwood and Dr. Clifford Les for their support, kindness, and guidance through many of my higher-education courses and projects. I would also like to thank Cal Poly's Graduate Education program for offering financial support through my appointment of a Graduate Assistant, allowing me access to funding that made much of this computational research feasible.

I would like to thank my family and friends for their constant encouragement and warmth.

And thank you to California Polytechnic State University for providing me with an excellent and safe academic environment. It has all been a wonderful little adventure.

Go Mustangs.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 Introduction

1.1 Background

Bone is a complex biomaterial that constantly adapts to the rigorous mechanical loading it is subjected to during daily usage. On its most fundamental level, bone may be considered a two-phase material, consisting of collagen and mineral [1]. The ultimate composition of bone may be attributed to multiple finely-tuned biological systems, so any small alterations to its chemical makeup may have large mechanical repercussions. The relationship between these two materials is linked to a dimension called "D-spacing." D-spacing describes the distance between neighboring collagen molecules [2]. As this dimension on the nanoscale may persist for hundreds of aligned collagen molecules [3], the intricacies of its patterning are of great interest in the field of bone research. Recent studies have reveled this dimension is not constant, and its variability warrants analysis.

Mineral (hydroxapatite) fills in the space between collagen molecules, and the degree of mineralization ultimately dictates the mechanical rigidity of bone [4]. Because mineralization plays such an important role in the biomechanical properties of the material, the dimensions of collagen molecules and spaces between them are critical areas of interest as they may impact the amount of mineral that may be laid within the bone matrix. Furthermore, the alignment of collagen alone determines collagen's potential packing factor. This packing factor expresses the volume of collagen molecules that may occupy a single microfibril, together creating an interwoven web of this strong, rope-like substance [5]. As a large packing factor is associated with the material's ability to manage incoming energy [6], the spacing between collagen molecules has a profound impact on the biomechanics of the skeleton.

Because bone possesses a heterogeneous makeup, its mechanical response to loading is non-linear. This nonlinear response is described as being "viscoelastic," and its properties are difficult to accurately quantify [7]. Thankfully, the "tangent delta" represents a simple way to express viscoelasticity. Tangent delta is directly related to the time delay (phase shift) between loading a structure and observing its response. The implications of this

phase shift are huge, as it describes a material's ability to dampen incoming energy. When a material possesses a small tangent delta, it's ability to mitigate oscillatory stresses is compromised [8].

By measuring the tangent delta of collagen molecules at various D-spacings, conclusions may begin to be drawn regarding how viscoelastic mechanical properties vary with serious degenerative diseases such as osteoporosis. If a link can be determined between variable postmenopausal D-spacing and bone viscoelasticity, a connection may begin to be drawn regarding the implications of estrogen loss and nanoscale collagen arrangements. Connecting these biomechanical dots may help contribute to uncovering the material underpinnings of bone disease.

In order to gain a strong understanding for the important role of D-spacing in the viscoelasticity of bone, it is first imperative to familiarize oneself with the basics of bone and the mechanisms that control its form: modeling and remodeling.

1.2 Bone Structure

Bone is a living, changing biomaterial that serves a number of critical roles within the human body. These roles include providing structural support, establishing a framework for muscles to act upon, offering protection from impact, storing vital minerals such as Ca^{2+} (a key messenger molecule), and housing bone marrow to produce blood cells [2]. Bone is constantly being renewed and replaced throughout an individual's lifetime. As this tissue is highly specialized, the architecture of bone differs throughout a body based on the biomechanical role a specific bone structure must perform. As a result, bone is classified into different categories based on its structure: *cortical* and *trabecular*.

The defining characteristic between bone types (seen in figure 1) is the degree of porosity. Apparent density– mass divided by bulk volume–is used to quantify this parameter. Cortical bone, known for being dense and sturdy, has an apparent density of about 1.8 g/cm^3. The far more porous trabecular bone possesses an apparent density of 0.1 to 1.0 g/cm^3. This metric is then transformed into a *relative density*, a ratio that compares the apparent density of a specimen to that of solid cortical bone. If a bone specimen has a relative density greater than that of 0.7, it is considered cortical bone [2]. Though porosity

is reflective of skeletal architecture, bone of both types is comprised primarily of collagen and mineral. The exact composition of bone is explored in greater detail in section 1.5.

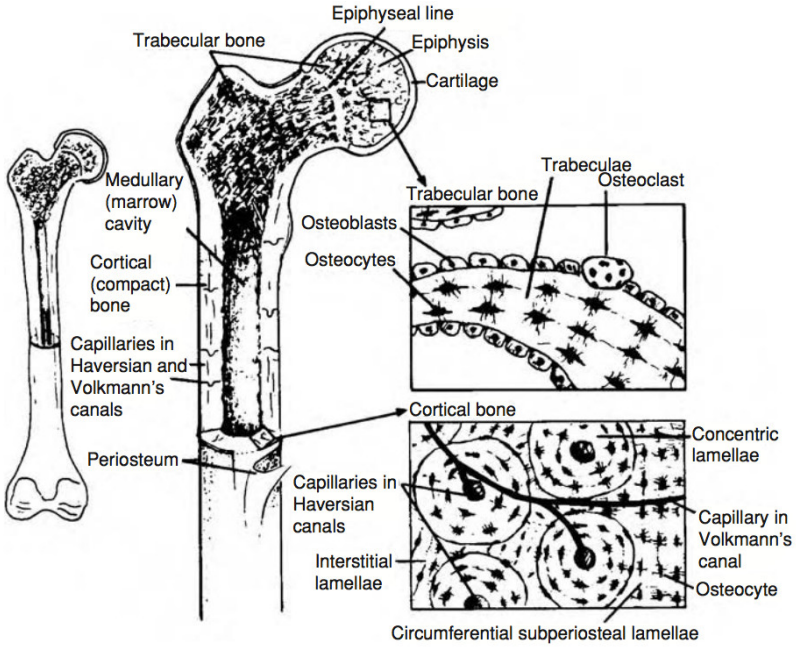

Figure 1: Bone is an adaptive and intricate material. The two main types of bone are cortical (compact) and trabecular (spongy) [2].

Cortical bone–commonly called *compact* bone–is found within the shafts of long bones. It is commonly referred to as *compact bone* given its relatively small porosity and high strength. Cortical bone is laid down in organized, parallel layers known as *lamellae* of only 5 micrometers thick. This highly organized bone structure is known as *lamellar bone* and is formed slowly from both collagen and mineral. Within a single layer, collagen and mineral exist in parallel. Between layers, however, the orientation of lamellae shift. This bone type often draws comparison to "twisted plywood" given its layering of highly organized collagen fibrils in differing angles (figure 2).

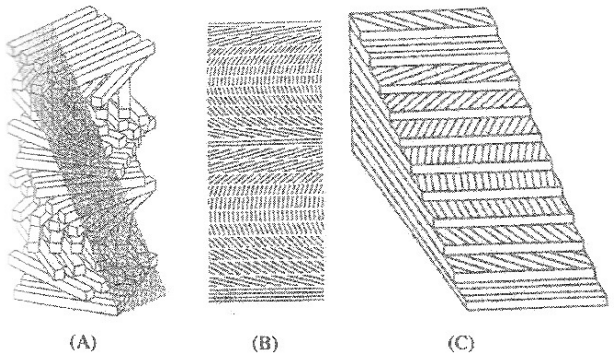

Figure 2: The organized, layered structure lamellar bone draws comparison to plywood. As each layer contains collagen fibrils oriented in different directions, the description of being "twisted" is often applied. Theoretical models (A, B, C) illustrate this layering effect [7].

The bone matrix is composed of a network of interconnected channels; this system's complexity serves an important role in the nourishment of the very bone cells responsible for the creation and upkeep of these networks. *Osteons*, also known as *secondary bone*, are comprised of both the canal that facilitates this nourishment and the surrounding circumferential lamellaer bone. The *Haversian canal* that runs through the center of an osteon houses a blood vessel. The *cement line* is a uniquely mineralized junction between an osteon and the interstitial bone, and its presence is used as a visual indicator for defining these circumferential bone features. The existence of cement lines is notable, as some believe the mineralization of this feature contributes greatly to bone's non-linear response to loading [9]. Each of these important features of bone tissue may be seen in figure 3 [7].

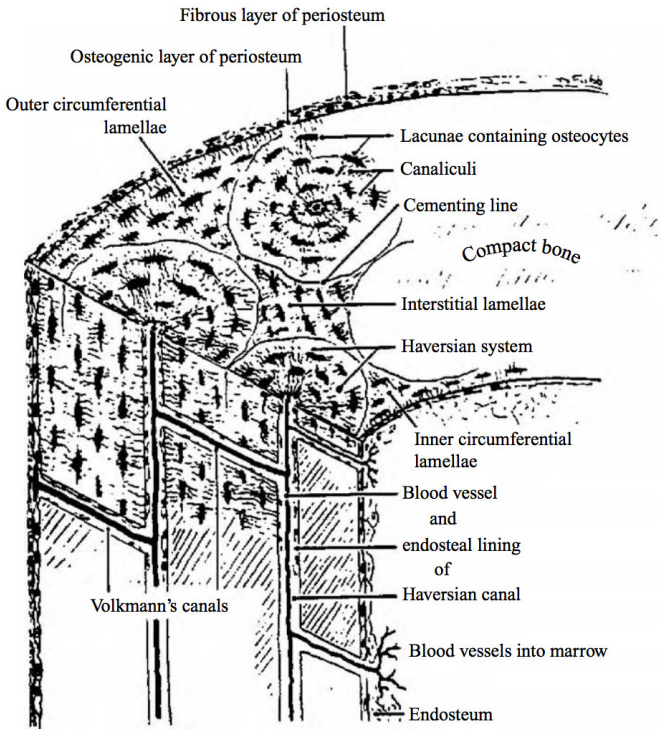

Figure 3: Bone is made up of a dense network of connecting pathways that allow nourishment and permit bone to continually adapt [2].

Trabecular bone (also known as *cancellous* or *spongy* bone) is also a lamellar structure, but the bone matrix is created in a three-dimensional framework. The bone matrix exists in the form of 200 μm diameter struts known as *trabeculae* [7]. Because it is highly porous, trabecular bone sacrifices a degree of strength that cortical bone boasts, but the trade-off comes in the form of a lighter overall weight that affords this bone type unique functions. Wolff's Law stipulates that bone adapts to the loads placed upon it [10]. Though the presence of these unique bone types is also partially genetic [11], the highly adaptive nature of bone personifies this well accepted theory.

On a physiological level, both cortical and trabecular bone possess an anatomy that accommodates a specific role in the biomechanics of the human body. Cortical bone's uniformity and deliberate patterning of osteons gives the bone strength and stiffness with respect to the stress axis [12]. As the architectural makeup of the long bone shaft, cortical bone in the femur supports the fluctuating body weight of an individual throughout his/her lifetime. The anisotropy of osteon orientation guarantees stiffness and strength through

the stress axis of bone, well outperforming trabecular bone in tension and compression. Conversely, trabecular bone's three-dimensional porous structure creates a web-like matrix of tissue that permits the bone to support principal stresses from many directions [13]. As such, the hip and wrist rely on trabecular bone's versatility in transmitting forces at a variety of angles to facilitate human movement. The light-weight matrix of trabeculae allows the bone to undergo large compressive strains when this matrix collapse, providing an effective safeguard against peak stresses [2].

Of course, the forces experienced by an individual change considerably from initial development through their adult years. To adequately respond to this, bone needs to be able to model and remodel to preserve the mechanical integrity of the human frame.

1.3 Bone Modeling and Remodeling

Because the primary role of the skeletal system is to transmit forces while shielding vital organs, bone maintenance is of the utmost importance. Thankfully, this important role is facilitated thanks to the unique adaption capability of the tissue.

Bone models and remodels over differing durations within the lifetime of an individual: bone modeling is the process of bone altering its size and mass and is thus most apparent during one's early to adolescent years [2]. Bone remodeling refers to a renewal process to existing bone and it occurs throughout an individual's lifetime [13]. On the most basic classification, bone modeling is responsible for the sculpture of bones and remodeling is responsible for renewal of bone. This distinction is important, as a human relies on their bone to adequately remodel throughout his/her life to continuously adapt to cracks and flaws in the bone structure [7]. When this mechanism of upkeep is compromised (or in a state of imbalance such as in the case of osteoporosis), a huge risk is presented to the mechanical framework of an individual.

Bone remodeling is an important topic for analysis because it occurs throughout one's life and thus dictates the long-term structure of skeletal tissue. It is estimated that 10-15% of all bone in the human body is replaced each year [2]. Mathematical models are constantly being developed to better hypothesize as to the long-term effect bone remodeling poses on the mechanical structure of an individual [7].

Though bone modeling is accomplished through the individual actions of *osteoblasts* (bone cells that synthesize bone) and *osteoclasts* (bone cells that remove bone), the remodeling process is sequential and achieved through an intricate combination of both bone cells [2].

1.4 Remodeling and Basic Multicellular Units

The synchronization for bone remodeling is accomplished via basic multicellular units (BMUs). BMUs represent an intermediary organization of both osteoblasts and osteoclasts that replace old bone with new bone in the form of discrete packets. BMUs follow a distinct pattern known as the *A-R-F sequence*. The A-R-F sequence represents stages of BMU activity in the form of *Activation*, *Resorption*, and *Formation* (though there are smaller intermediate steps in the sequence) [7]. In response to a microcrack (small fractures that occur naturally throughout an individual's life), osteoblasts and osteoclasts work in unison to alter, replenish, and repair bone [2]. Together, this combined sequence may last about 200 days as bone remodels across these distinct stages [7]. An image of a BMU and the functions it performs are on display in figure 4.

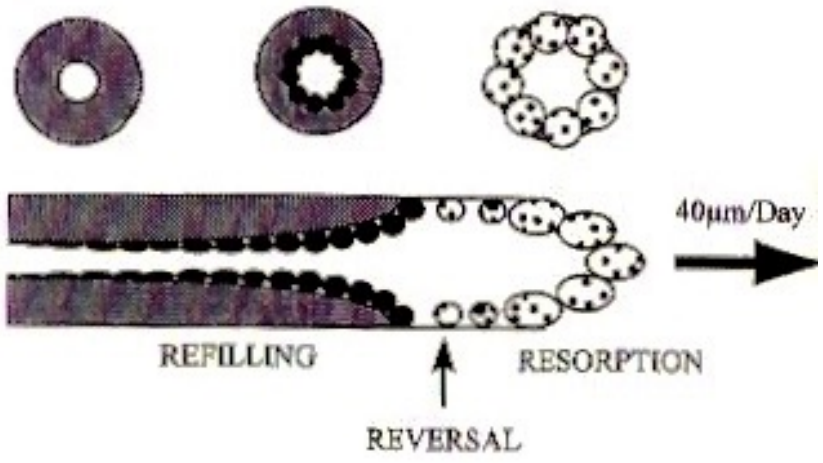

Figure 4: The osteonal BMU is a combination of both osteoclasts and osteoblasts. Large, multinucleated osteoclasts are pictured on the right whereas the small, mononucleated osteoblasts are seen on the left [7]. The unison of bone cells create a highly regimented renewal process of existing bone, including fracture repair.

Activation refers to the period of time where osteoclasts are produced and a resorption

plane space is visible. Activation takes about three days. The *Resorption* stage is marked by newly formed osteoclasts tunneling longitudinally though bone at a rate of 40 μm per day. *Reversal* signals the transition from osteoclast to osteoblast activity. In a completed osteon, Reversal may be identified by the cement line that coincides with the location of the bone surface during this period. *Formation* marks the time where the osteoblasts begin to refill the space created by the previously tunneling osteons with new bone. The process of new bone deposition is slow, averaging 1-2 μm per day. The Formation stage may last up three months. The aforementioned Haversian canal in an osteon is indicative of the passageway that remains to allow the movement of nourishment to the BMUs during this process [7].

Following Formation comes another critical stage: *Mineralization*. As previously mentioned, mineral is a key ingredient to the chemical makeup of bone. Bone mineral provides a stiff and brittle material that contributes to bone's overall toughness [4]. This mineral is deposited into the organic bone matrix (known as *osteoid*) in the Mineralization step following the Formation stage. Within this step, mineral is laid between collagen molecules. This process begins following a delay known as *mineralization lag time*, which may last 10 days. Once the mineralization process has begun, 60% of mineral is laid within a few days, known as primary mineralization. The remainder of the mineral is added more gradually and may take up to 6 months to become fully distributed. This mineral addition plays a vital role in the mechanical rigidity of bone [7]. This important stage of the A-R-F sequence is discussed further in section 1.6.

The final step of the A-R-F sequence is *Quiescence*. Quiescence is marked by the completion of the tunneling and refilling processes. During this final step, osteoclasts vacate the area and osteoblasts may become *osteocytes* (inactive osteoblasts buried in the bone matrix) or *bone lining cells* (former osteoblasts that now rest on the surface of the bone, such as around the Haversian canal) [7].

1.5 Composition of Bone

Most simply, bone may be considered a two-phase composite biomaterial composed of an organic collagen phase that is reinforced by a mineral phase. Combined, these two ingredients provide the structural rigidity and strength for which it is known [2, 1]. The collagen

phase is largely type I collagen, and the mineral phase is primarily crystalline *hydroxyapatite*, ($[Ca_3(PO_4)_2]_3Ca(OH)_2$) [14]. Of overall bone composition, hydroxyapatite accounts for 70% of total bone mass whereas collagen is responsible for 18%. Two percent of bone mass is non-collagenous protein and proteoglycans. Water makes up the remaining 10% [2]. Type I collagen is found in the extracellular matrix (ECM), where collagen plays a key role in the mechanical properties of bone [3]. Though type I collagen is not exclusive to bone and may be found in other non-cartilaginous tissue such as skin (dermis) and tendon, its potential relationship to bone embrittlement makes it a highly attractive avenue for study [15, 16].

Type I collagen is the dominant form of collagen found within bone (approximately 90%) [1]. This collagen type is significant for its high tensile strength, possessing a tangent Young's modulus of approximately 2-5 GPa [2, 17]. Collagen molecules form large bundles with neighboring units in regions of the body that experience loading. Practically, a *bundle* is just a grouping of parallel fibrils cross-linked together, which comprise a lamella layer in bone. These bundles are described as weaving together into a strong, "rope-like" grouping that is fundamental to the mechanical role of compact bone [2].

Collagen's stiffness and strength are a result of its hierarchical structure, as seen in figure 5. Cross-linked fibrils are the culmination of many smaller units. *Tropocollagen* is the most basic unit of collagen. The composition of this unit on the nanometer scale can be described as a *triple helix*. A triple helix is formed as the result of three polypeptide chains (α chains) wound together. This tight winding is stabilized via hydrogen bonds, creating tropocollagen of 280-300 nm in length [2].

Tropocollagen molecules are cross-linked to neighboring tropocollagens forming long strands. These molecules are offset from neighboring units by roughly 67 nm, a dimension described as their *D-spacing*. A combination of at least five tropocollagen molecules form a microfibril, which in tandem make up a collagen fiber [18]. Because the biomechanics of collagen are dependent on many levels of structure, each parameter is critical. It is theorized that the non-linear behavior of bone may be attributed to the complex molecular interactions of collagen within the tissue [2, 6]. As D-spacing is reflective of the staggering of tropocollagen (the fundamental building block of collagen) this parameter is potentially

critical in determining the long term mechanical role of this important bone ingredient.

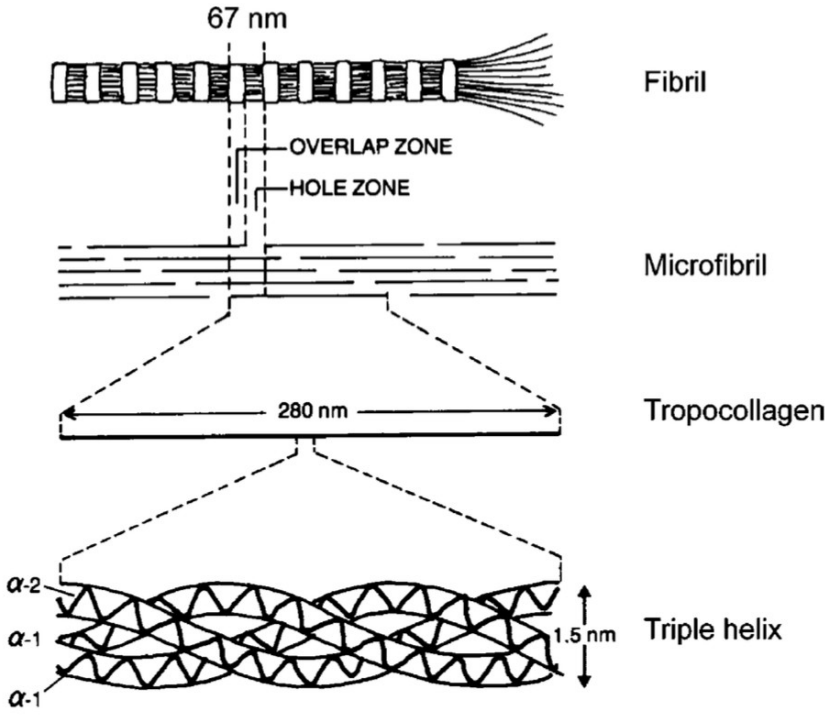

Figure 5: The hierarchy of collagen shows that larger collagen complexes are made up of many smaller units. Neighboring collagen molecules are separated by approximately 67 nm (D-spacing) [2].

1.6 D-spacing

D-spacing can be defined as the spacing between neighboring collagen molecules [19]. This dimension, also referred to as the *periodicity* can be identified as the 67 nm offset in figure 6. Because the length of collagen molecules (about 280 nm [2]) is not a multiple integer of this 67 nm staggering, neighboring molecules exhibit both overlap and gap regions [20].

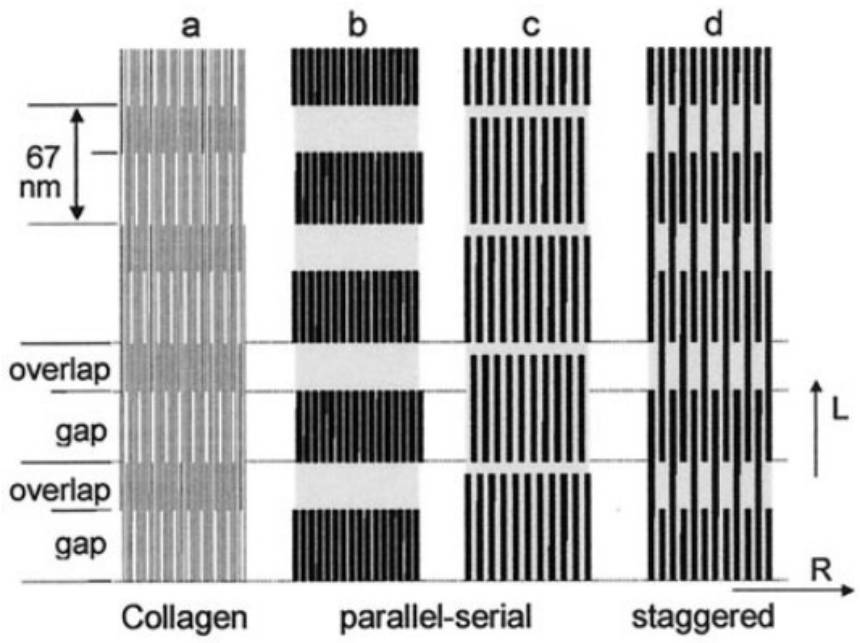

Figure 6: D-spacing describes the degree of separation between collagen molecules. This non-constant 67 nm separation results in collagen molecules expressing regions of gaps and overlaps. The Hodge and Petruska model (a) underwent multiple iterations (b-c) before landing on the now popular staggered array model, a model that accounts for these periods of mismatch (d) [20].

Since the discovery of D-spacing in 1942, little emphasis was initially placed on prioritizing the investigation into the repercussions of changes to this parameter [21]. Advancements in microscope technologies and an increased scientific interest in the field of biomechanics, however, have gradually shifted focus back to this pivotal dimension [22]. Specifically, work by Hodge and Petruska is accredited with renewing a focus on this parameter. In 1963, Hodge and Petruska utilized electron microscopy and phosphotungstic acid (PTA) to identify light and dark regions in tropocollagen's periodicity [4]. Through negative staining, Hodge and Petruska noticed definitive *gap and overlap* regions by their dark or light contrasts. They noted that the tropocollagen molecules were offset by a "quarter-stagger" (i.e. the D-spacing), that could be used to quantify this characteristic spacing. This discovery encouraged Hodge and Petruska to develop a number of theoretical models to better illustrate this nanoscale parameter.

Since their work applying a definitive model to this characteristic collagen spacing [4,

19], research in the field has continuously adapted and improved upon their initial design. Their seminal model can be seen in figure 6a, which has later been represented as the staggered array model 6d [20].

D-spacing plays a critical role in bone mechanics for two reasons: D-spacing has a large impact on collagen distribution and D-spacing is innately linked to the degree of mineralization. D-spacing controls collagen distribution as it is reflective of the alignment of a network of collagen molecules [4]. The aforementioned collagen hierarchy permits spacing between its subunits as a means of crimping under loading [7], implying collagen's mechanical role may be facilitated by its staggering.

Inter- and intrafibril collagen orientation has been proven to be linked with the energy required to cause bone failure and fracture [23], prompting research to deem it plays an important role in bone fragility. The alignment found within collagen fibrils is thus hugely important, as the patterning of tropocollagen is reflective of the packing density and mechanical role of collagen [24]. For this reason, the alignment of collagen molecules is pivotal in dictating overall mechanical properties and warrants modeling.

It was previously assumed that this staggered D-spacing was roughly constant in humans at about 67.0 nm [2, 3, 25], while earliest recordings in 1942 pinned the dimension at 64.4 nm [21]. Variability in D-spacing measurements were previously attributed to X-ray diffraction or the collagen hydration level [26, 27]. This understanding was later broadened, however. D-spacing is a variable dimension that is highly dependent on the relationship a collagen molecule shares with neighboring units. Within a single bundle, D-spacing may vary by 1 nm, but a full range of 10 nm may be observed in differing bundles [3]. The exact reasoning for the 10 nm range between bundles layers is still unknown; some research has pointed to variable strain rates controlling collagen spacing [28, 29] while other findings seem to point to mechanics in the collagen cross-linking [25, 30, 31, 32]. Extensive research, however, accomplished with use of Atomic Force Microscopy and the Fast Fourier Transform across 1710 fibrils in 2012 did not directly observe strain rate or cross-linkings impact on bundle variability [3]. Recent findings have also suggested that genetics play a large role in D-spacing [1], so perhaps part of the answer lies in our genetic code.

Regardless of the exact means of variation, collagen molecule investigation has all

pointed to a common truth: collagen molecule orientation is not constant and warrants close examination [3, 18]. Figure 7 illustrates the variability of this parameter within a collagen fibril.

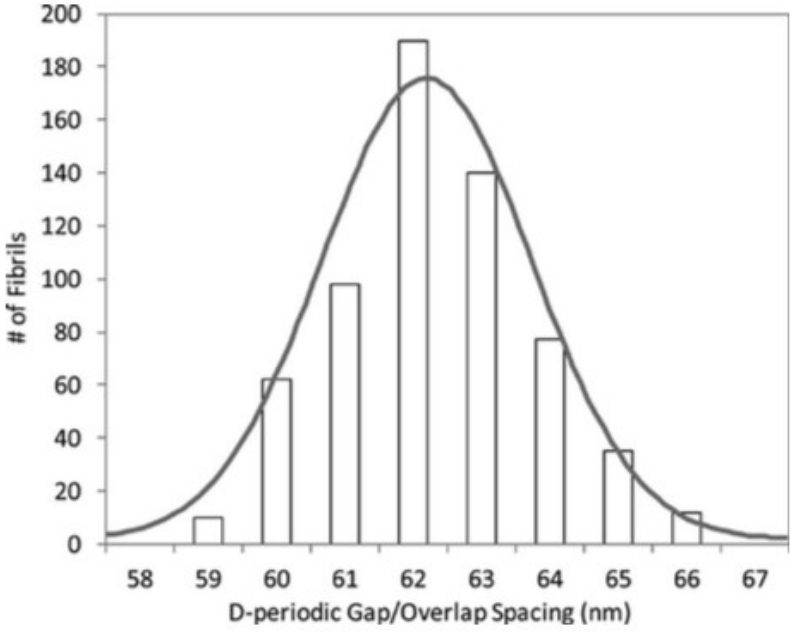

Figure 7: A Gaussian function fits a typical D-space distribution from ovine dermis. The organization of collagen molecules varies significantly within a tissue [3].

D-spacing is also intricately linked to the amount of spacing between collagen molecules and has important mechanical repercussions as it influences the degree of mineralization. Because mineralization fills the space between collagen molecules (known as *mineral nucleation sites* [4]), collagen molecule orientation and the mineralization are entwined. Initially linked with water between gaps of collagen, mineralization occurs after an interval called *mineralization lag time* and ensures bone has sufficiently remodeled before mineral content is laid down. Figure 8 shows that the room for mineralization is dependent on the D-spacing of the collagen matrix. It has been found that bone that remodels more frequently has a lower *volumetric mineralization* (mineral per unit volume), indicating that remodeling is associated with the degree of mineralization [7].

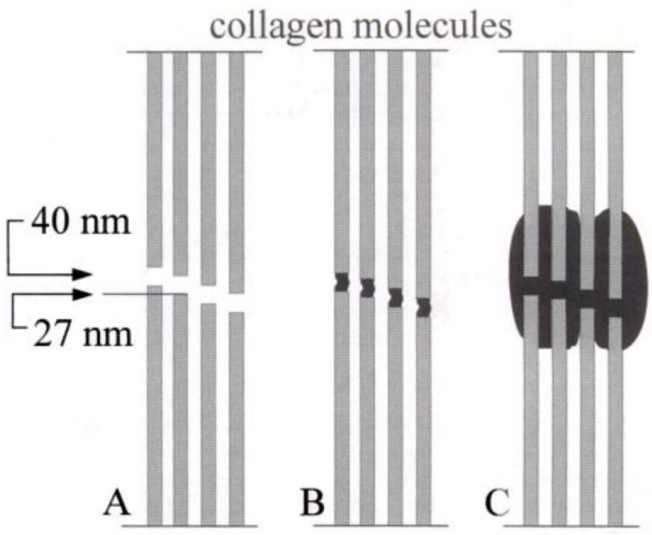

Figure 8: The degree of mineralization of the collagen matrix is dependent on the orientation of the collagen molecules. Here, a 40 nanometer gap between collagen molecules is offset by 27 nm (for a combined 67 nm spacing). This gap permits mineral crystal formation (seen as black) The layout of these collagen molecules and the space permitting mineralization plays a key role in the overall mechanical rigidity of the adapting bone [7].

Mineralization has a direct relationship with the stiffness and strength of bone. Researchers Vose and Kubala found a correlation between the bending strength of human femurs and the degree of mineralization. In their 1959 study, they concluded that even minute changes in the percentage of mineralization could have big repercussions on mechanical properties: a change from 63% to 71% mineralization could dramatically impact the bending strength by a factor of 3.7 [33]. In 1969, Currey expanded on this theory by determining the substantial spikes in stiffness observed at high bone mineral composition were attributed to the hydroxyapatite mineral crystals fusing together when they came into contact [34]. Other research has concluded that the presence of mineral in this matrix may increase the elastic modulus of this structure by a factor of 400 [20]. Though the orientation of mineral crystals does have some affect on the toughness of hydroxyapatite, the material may be considered largely isotropic [35]. Because tiny alterations to the hydroxyapatite content of bone may hold big mechanical implications, study is warranted on how changes to collagen orientation may have a meaningful effect on the growth of bone.

1.7 Postmenopausal Osteoporosis

Osteoporosis is the most common type of metabolic bone disorder, affecting over 200 million individuals worldwide [36]. The disorder is so prevalent that nearly half of the female population over the age of fifty will experience osteoporosis as a direct result of menopause. The condition is marked by a dramatic loss in bone density (figure 9) [37].

This decrease in bone density leaves individuals at risk of fracture at low trauma levels. Bone mineral density (BMD) serves as a metric to classify an individual's state of osteoporosis. This tool permits distinction between the severity of bone brittleness, and a small decrease in bone density can be described as *osteopenia*. To track BMD, patients may be evaluated via dual-energy X-ray absorptiometry [38, 39], though this effort is often in vain; bone brittleness is an exceptionally difficult disorder to track as patients may not realize they are experiencing skeletal weakening until fracture occurs [40]. Furthermore, the mechanical underpinnings of osteoporosis may be more closely associated with the makeup of existing bone (i.e. the nanoscale alignment of collagen), which is largely unaccounted for in current BMD assessment techniques [41]. It is estimated that 8.9 million fractures occur annually worldwide, bringing the lifetime risk of wrist, hip, or vertebral fracture to 30-40% [42].

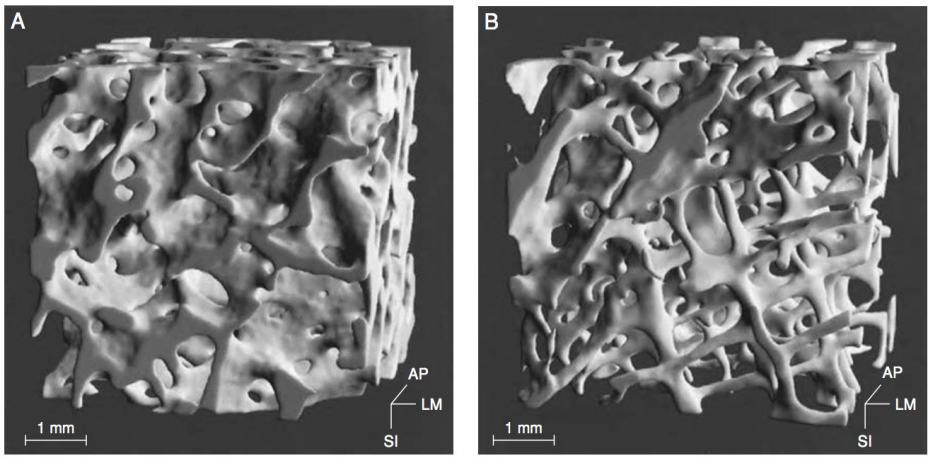

Figure 9: The comparison between A) a healthy 37 year-old man's pelvic bone trabeculae and B) a 73-year-old woman with osteoporosis illustrates the significant loss of density and overall thinning of bone in the diseased patient [2].

Though the effects of osteoporosis on skeletal structural integrity are well-documented,

it is still very unclear as to what causes this highly prevalent weakening [2]. Popular theories speculate an overabundance of bone resorption by osteoclasts signifies the onset of osteoporosis. However, conflicting theories suggest that a potential defect in the osteoblast's ability to properly lay down new bone may also be the root of decreased bone density [37]. Though a single cause is unknown, confirmed risk factors for the disease include smoking, alcohol use, inflammatory disease, excessive inactivity, low vitamin D, genetics, and estrogen loss [40, 43].

It is well established that osteoporosis is especially common in women who have undergone menopause. Menopause is marked by the period in a woman's life when her ovaries no longer produce eggs. There is a significant drop in estrogen during this time [44]. Estrogen plays a critical role in bone development, serving as a signaling factor to monitor bone modeling and remodeling. It is thought that estrogen may inhibit osteoclast activity, implying the hormone's absence permits uncontrolled bone resorption [43]. Though estrogen's responsibility as an important signaling hormone is not exclusive to the female gender, the significant decrease in estrogen associated with menopause leaves women at great risk [45]. In postmenopausal women (or men with defects in their estrogen receptors), bone resorption either outpaces the addition of new bone, or bone is placed inefficiently. In either case, one thing is clear: with declined estrogen levels, bones have shown to remodel rapidly, resulting in a weakened state [46].

Because the exact mechanisms of osteoporosis are unknown, treatment options vary wildly in their effectiveness. Given osteoporosis's strong connection with estrogen loss, postmenopausal hormone-replacement therapy was an attractive option for many years [47]. Though short-term hormone treatment seemed to reduce hip fracture by 33% [48], long-term usage correlated with measurable bone loss. Additionally, long-term hormone therapy has been associated with breast cancer and cardiovascular disease [43]. Alternately, selective estrogen-receptor modulators such as raloxifene may serve as an estrogen substitute, but this approach is relatively new and untested [49].

Bisphosphonates are the most common approach to treating postmenopausal osteoporosis [43]. These agents block the attachment of osteoclasts to the bone matrix and enhance programmed cell death. Bisphosphonates have proven to increase bone mass considerably

and show few adverse side effects [50]. Increased bone mass, however, does not always correlate with increased bone strength, making bisphosphonates an imperfect solution to this common disorder. Similarly, fluoride treatment has been successful at encouraging bone mass, but long-term fracture risk is still a major threat [51].

There is a strong precedent for investigating D-spacing's relationship to osteoporosis [52]. Figure 10 from Wallace et al. was utilized to examine the patterns of D-spacing in test mice possessing an altered allele that left them with a defective collagen protein structure. This histogram illustrates that though both healthy and defective mice possess a similar range of D-spacing values, mice with estrogen depletion exhibit a greater variability in their bone's collagen periodicity [1].

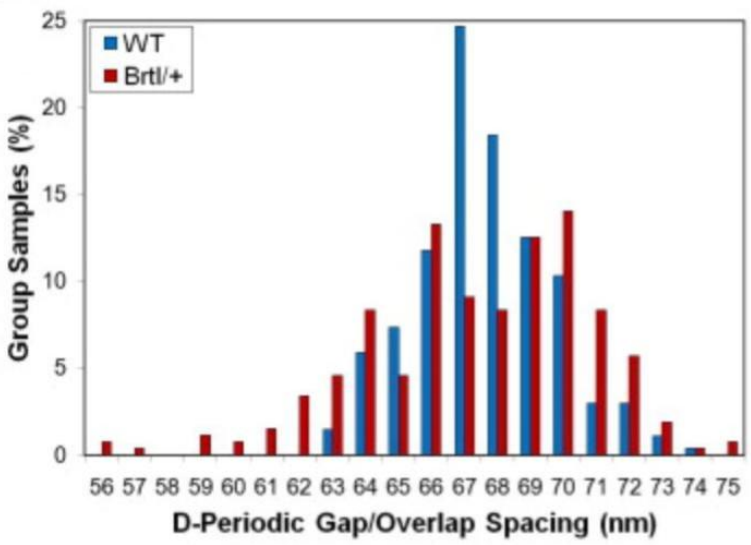

Figure 10: Though control mice (WT) and mice genetically prone to Osteogenesis Imperfecta (Brtl/+) have a similar range of D-spacing values, Brtl/+ express an unusual degree of variability in their collagen periodicity [1].

Osteoporosis is an especially difficult disease to address because bone is a naturally living tissue. As bone remodels throughout the life of an organism, performing relevant research that is reflective of this ongoing adaption is a challenge. For this reason, many researchers from past decades have turned to experimenting with animal models.

1.8 Animal Models for Bone Research

1.8.1 Overview of Animal Models

Because bone is a naturally adapting tissue, it is difficult to track patterns of bone modeling and remodeling in living subjects. Removing bone samples at a single time-point (i.e. at specimen death) limits researchers from observing the bigger picture of long-term bone remodeling. Furthermore, obvious ethical considerations eliminate the possibility of utilizing humans as an experimental model, so research instead targets animals as a means for experimentation. Because random factors can be more easily controlled in animal samples (i.e. controlling diet, environment, etc), studies have focused on animal models for the field of skeletal tissue mechanics.

Though researchers have examined animal bone as a means of bettering the understanding human bone tissue architecture for centuries, the use of animals for biomechanics testing has been popularized more recently [21]. Goodship et al. elected to use pigs in 1979 as a way to observe how a living host controls the modeling and remodeling of bone tissue [7]. In the experiment, Goodship removed a portion of the central ulnar diaphysis from maturing pigs to examine the long-term effect of both modeling and remodeling on the radius and ulna. By utilizing pigs, Goodship was offered the opportunity to gather experimental data that supported longstanding theories from nearly a century earlier that bone adapts to normalize peak strains [10]. The experiment indicated that three months post-surgery, the bone tissue of growing pigs was deposited in a pattern to maximize the damaged bone's cross-sectional area, thus minimizing peak stresses [53].

Pigs are just a single example of an experimental subject used in an effort to better understanding the growth of bone tissue. Researchers must carefully select appropriate animal models to observe specific biomechanical functions. In 1978, Uhthoff and Jaworski studied the effect of remodeling on canine bone tissue in an environment of low mechanical stimulus. Avian models were selected by Lanyon and Rubin in 1984, as birds exclusively load their wings through the act of flapping. Isolated from unforeseen mechanical variables, birds present a unique way to observe patterns of bone adaption with little external influence [54]. Even rats have presented an opportunity to observe the relationship between peak

strains and bone formation rate, as Turner and Forwood selected rodents for the focus of their 1994 study [55].

There are two important limitations that must be circumvented to appropriately use animal models as a proxy for human bone tissue development. The first major difference lies within bone modeling in the animal kingdom. Though section 1.2 outlines the carefully organized lamellar structures that forms bone in humans, animal bone is often organized in a separate fashion.

This separate path comes in the development of an alternate bone type, *plexiform bone*. Plexiform bone is common in prey animals, as the organism maximizes bone deposition in a short time frame. This rapid approach grants the host animal sufficient mechanical support at a young age in order to flee in the presence of a predator [7]. Unlike lamellar bone, plexiform bone formation is exceedingly rapid: a trabecular network is created on the surface of bone and gaps are quickly filled in to generate a block structure with high fatigue resistance (figure 11). The second major difference encountered through animal models is that–perhaps obviously–human bone growth facilitates human usage. Small animals do not experience loading to nearly the same magnitude as humans, and the weight of land animals is commonly distributed across four limbs, not two. Because loading is strongly associated with bone remodeling [10], animals of excessively light weights may not experience remodeling to the same magnitude as do humans, making their selection inappropriate to draw human conclusions.

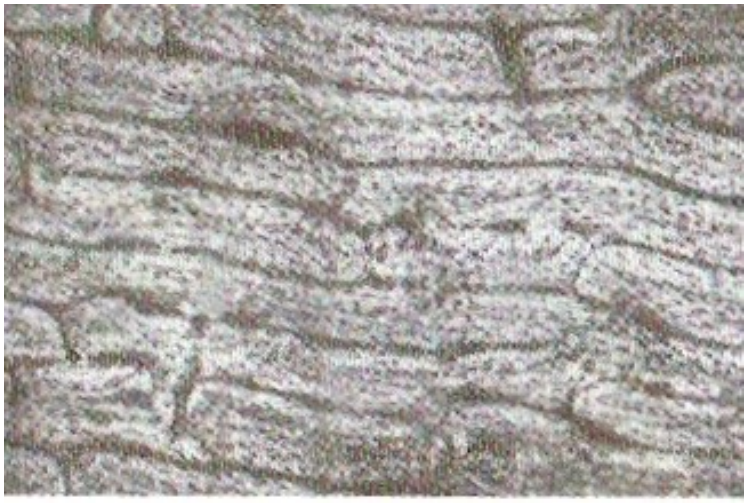

Figure 11: Plexiform bone is common in prey animals and is unlike lamellar bone. It forms rapidly; gaps are quickly filled to give the animal large fatigue resistance if it must be prepared to flee from predators [7].

Another limitation when it comes to modeling osteoporosis lies in the exclusivity of menopause to female humans. Because the intense estrogen loss characteristic of menopause is a high osteoporosis risk factor for middle-aged women, mimicking estrogen depletion becomes a key factor in adequately representing humans through any animal approximation.

Given these limitations, the selection of the appropriate animal model is an important decision in the foundation of applicable bone research.

1.9 Sheep as a Solution

Sheep present a great opportunity to circumvent many animal model limitations. As an animal model, sheep are an ideal candidate: sheep are docile, fairly inexpensive, simply obtained, easy to handle and house, and–perhaps most critically–they ovulate and poses a hormonal pattern similar to women [56]. By limiting experimentation to mature sheep, studies are able to isolate the effects of remodeling, assuming that the bone modeling stages of early life bone growth have ceased. In their 1982 study, Lanyon et al. examined the effects of ulnar osteoectomy on skeletally mature sheep, finding that new bone formation occurred on surfaces adjacent to the removed bone [57].

Sheep present an excellent avenue for experimentation; the size and weight of sheep are a closer counterpart for humans than rodent or porcine models, and their use is well-

documented [3, 52, 58]. Due partially to their weight, sheep also develop bone that is similar to human compact bone (figure 12). A major challenge posed by sheep models, however, is the fact that they ovulate spontaneously and do not undergo natural menopause [56]. To circumvent this, an overiectomy may be performed to remove the ovaries from mature sheep, incurring estrogen loss with the goal of simulating menopause. Sheep also undergo mineral density loss post ovariectomy, meaning they may be an appropriate model for studying postmenopausal osteoporosis [59, 60].

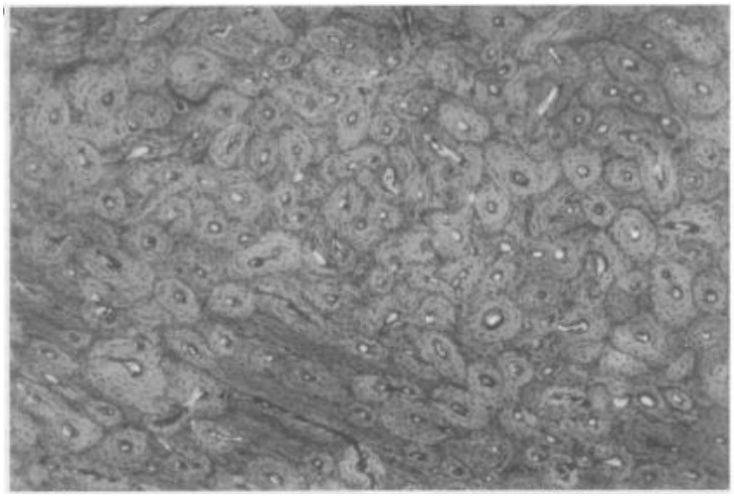

Figure 12: In addition to being an attractive model based on cost effectiveness, abbundance, ands weight, sheep posses a Haversian bone structure similiar to compact bone found in humans (figure 3) [56].

As a model for D-spacing, the ovine model is well established as being an appropriate proxy for human collagen distribution [52, 3]. D-spacing differences lie primarily in differing bundle levels (approximately 76%) and are independent of species and tissue type [3]. Figure 13 illustrates the noticeable similarity between human and ovine collagen molecule orientation, regardless of tissue type.

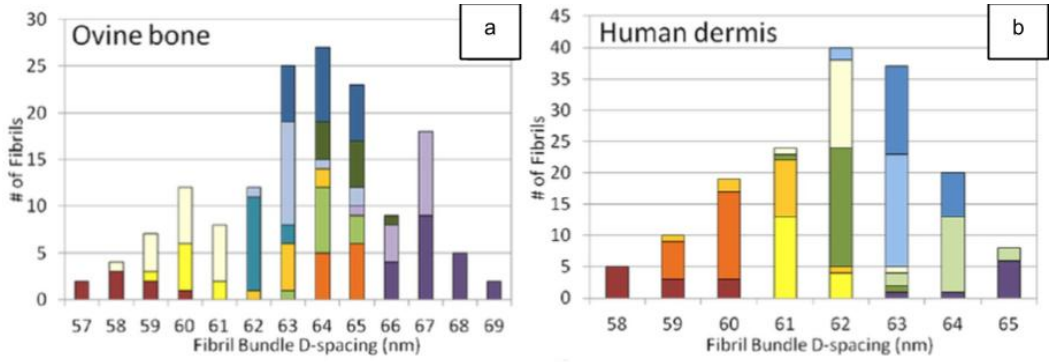

Figure 13: a) Sheep D-spacing displays a variability remarkably similar to that of b) human (Redrawn from Fang et al.) [3].

Ovariectomized skeletally mature sheep present the opportunity to study the effects of collagen orientation and the degree of mineralization on bone growth. Though collagen size and placement in the mineralization phase of the A-R-F sequence controls the amount of mineralization, it is unknown to what magnitude D-spacing (especially the combined impact of many collagen molecules) impacts material properties. Because the composition of bone contains both a tensile collagen phase and stiff mineral phase, the material properties of bone are complex and necessitate detailed description.

1.10 Viscoelascity

1.10.1 Rheological Models

Though dense, mineralized hydroxyapatite may be simply modeled as a linear material [35], collagen is *viscoelastic* [7, 28, 29, 61]. Viscoelasticity describes a relationship where both load-displacement and stress-strain relationships are rate and history dependent. Figure 14 illustrates how the relationship between stress and strain changes based on repeated loading. Collagen may also exhibit *creep* (deformation change under constant loading), *stress relaxation* (stress decreases at constant deformation), and *hysteresis* (elastic energy loss during load-deformation cycle). For these reasons, modeling viscoelasticity is critical to ensuring an accurate finite element model [7].

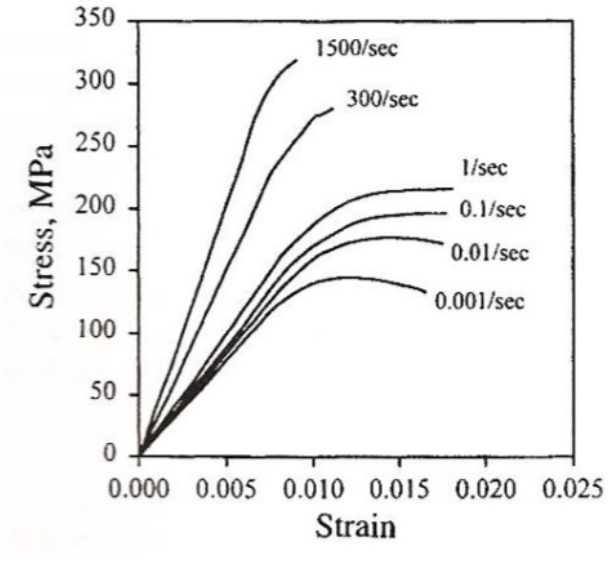

Figure 14: The viscoelastic behavior of bone means the material becomes stiffer at high frequencies [62].

Viscoelastic properties are represented by rheological models in the form of a spring and dashpot. The spring models elastic behavior, whereas the dashpot is responsible for representing viscous properties (where strain rate is linearly proportional to applied stress). Figure 15 illustrates how these elements appear in rheolgoical form [7]. Both units may be combined in series or in parallel to simulate their composite effects. By adding complexity to a rheological model, the user may more closely represent his/her desired viscoelastic material. Each combination of rheological elements may be represented in corresponding mathematical formulas that allow users to approach otherwise complex viscoelastic properties simply [7].

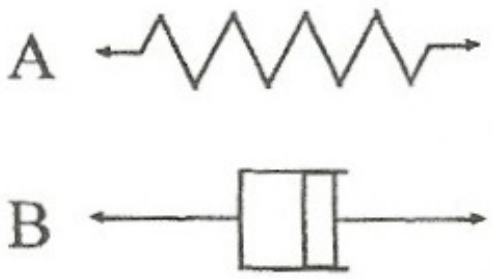

Figure 15: (A) The spring and (B) the dashpot present simpler units for rheological models. The combination of these elements may be manipulated to represent complex viscoelastic material properties [7].

It has been thoroughly shown that viscoelastic properties of cortical and trabecular bone have an effect on the material's overall strength and stiffness [63, 13]. On a physiological level, bone stiffens in response to strain rate as a mechanism to prevent fracture in the event of sudden, traumatic impact [7]. For this reasons, viscoelasticity presents an important parameter for analysis as it helps to more closely approximate the characteristic time-dependent responses of bone. As research in orthopedic biomechanics has evolved in the past decades, so too has an acceptance of the complex time-dependent properties of bone tissue [64].

What was initially believed to be an exclusive relationship between fatigue damage and bone strength was later accepted to be a relationship reflective of creep damage [64]. This revelation pushed the development of new models furthering the connection between ultimate tensile strength in human bone and the strain rate [65, 66]. Similarly, research on the compressive strength and fracture mode of porcine mandibles was determined to be heavily dependent upon strain rate and microstructure (such as the orientation of lamellae, collagen fibrils, and mineral content) [12]. Fracture mode may be controlled by strain rate, as high work at a low rate is associated with fibrous fracture surfaces, whereas low work at a high strain rate is related to resultant cleavage surface [67, 68].

Though it is established that bone expresses viscoelastic behavior, it is unclear as to why this material response exists [69]. Bone is a two-phase composite biomaterial, so a non-linear response of both tensile collagen and stiff mineralization is not completely surprising.

It has been suggested that collagen specifically may be responsible for presence of a creep response [70]. Moisture content may influence these responses [71], and the staggering of type I collagen molecules influences the position of water present prior to mineralization [7]. The cement line interface of Haversian canals may also influence strain rate dependent responses, as bone material varies greatly around these formations [9]. The heterogeneity of bone's viscous collagen and elastic hydroxyapatite is also a likely contributor to the material's non-linear responses [72].

Realistically, bone viscoelasticity is likely a combination of many sources. Similar to the case with ongoing osteoporosis research, the precise reason for these behaviors is likely an amalgam of many separate factors. Whatever the exact mechanism, it is well established that bone is viscoelastic. Modeling D-spacing and observing its impact on creep and stess relaxation responses may uncover another piece of this time-dependent puzzle.

Viscoelasticity is a key parameter for modeling and analysis as it has a proven relationship with long-term ovariectomy. A study by Les et al. utilized dynamic mechanical analysis (DMA) to measure viscoelastic properties in compact bone in sheep with three year ovariectomy [41]. DMA presents a great way to examine bone's material responses at differing frequencies, and this methodology is implemented frequently [69, 70, 71]. DMA was modeled with low-amplitude 3-point bending. In this study, Les et al. found that the viscoelastic storage modulus significantly decreased at high test frequencies in the animals with their ovaries removed [41]. Ultimately, this study determined that patients suffering estrogen loss may experience dramatic changes in viscoelasticity (figure 16).

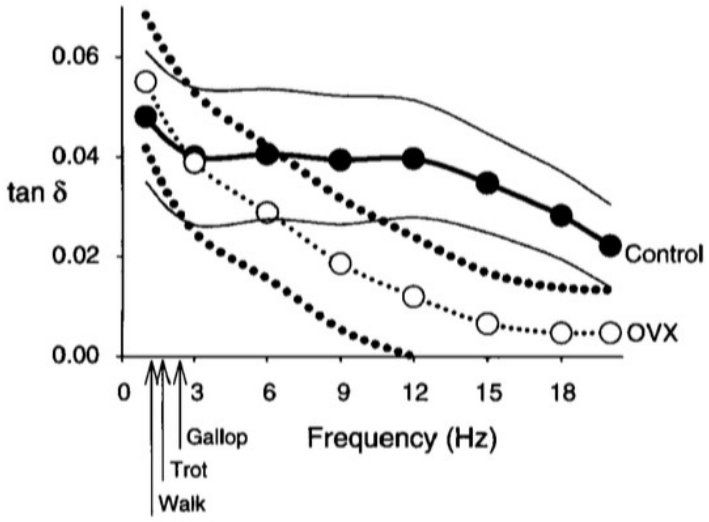

Figure 16: A long-term ovariectomy model exhibits noticeable changes to tangent delta at high test frequencies [41].

Because viscoelastic relationships are intrinsically complex, analysis of changing material properties can be difficult. Luckily, the solution to overcoming time-dependent material property's innate intricacy comes in the form of analysis of the tangent delta.

1.11 Tangent Delta

Measurement of complex time-dependent viscoelastic properties may be best facilitated through analysis of *tangent delta* (tan δ, otherwise known as the *loss tangent*). Tangent delta is obtained through the examination of a phase shift, δ, at a single frequency. Specifically, tangent delta is the phase shift between the sinusoidal loading and the sinusoidal response components [41]. An example of this shift can be see in figure 17 [73]. Tangent delta may be viewed as the conversion of mechanical energy into another form, such as heat energy, though that conversion is only one example to explain the phase shift [74].

Simply, tangent delta is representative of a material's ability to dampen and dissipate vibrational energy [8]. An ideally elastic material would experience a tangent delta of zero. Cortical bone has been shown to typically express a tangent delta of between 0.01 and 0.04 [61, 69, 75], though this number may just be a small snapshot of a larger range of phase shifts across many frequencies. Because *toughness* is characteristic of a material's ability to absorb energy without failing, bone with a low tangent delta may be unable to

dampen incoming rate-dependent energies. Bone that is less tough is at risk for fracture [41, 46]. This correlation between the fracture mechanics of energy at a crack tip and a material's ability to dampen energy is well-established in literature [8]. Similarly, bone strength/toughness and viscoelasticity hold a proven correlation [12, 63, 66, 76]

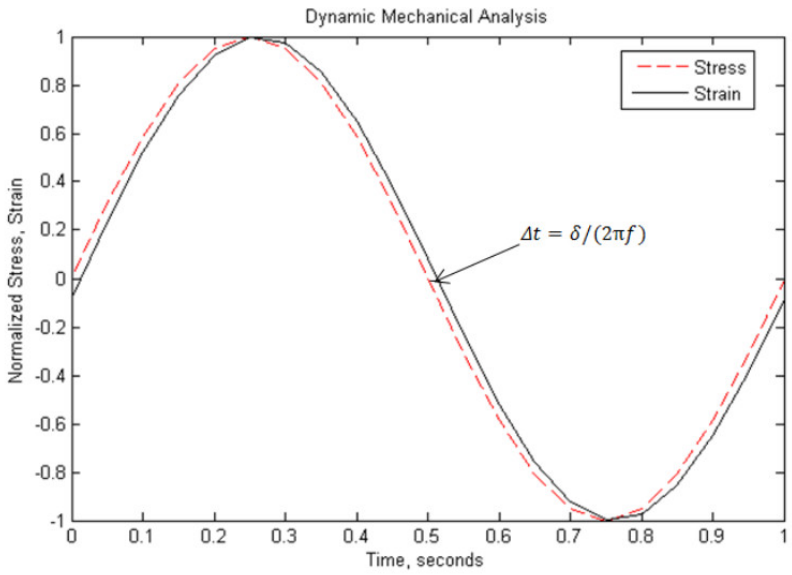

Figure 17: The shift between the loading condition (stress) and response (strain) is illustrated by Δt, a measurement directly related to tangent delta [73].

Tangent delta may be simply defined as a ratio of the *loss modulus* over the *storage modulus*[41, 74]. The storage modulus is also known as the real component of an elastic modulus, and it is roughly equivalent to the material's Young's modulus under non-oscillatory conditions. For this reason, $E_{storage}$ is strongly indicative of a material's stiffness. The loss modulus is representative of the imaginary, dynamic modulus and is a reflection of a material's damping ability [77]. This relationship is derived as the following:

When an oscillating stress, σ, is applied to a viscoelastic material at a frequency ω, the resulting stress is equivalent to

$$\sigma = \sigma_o cos(\omega t) \tag{1}$$

The resulting strain, ε, is defined as

$$\varepsilon = \varepsilon_o cos(\omega t - \delta) \tag{2}$$

where δ is the phase angle between σ and ε. A complex modulus of the material may be expressed as E^*

$$E^* = E_{storage} + iE_{loss} \tag{3}$$

where $E_{storage}$ is the storage modulus

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

and E_{loss} represents the loss modulus.

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

Tangent delta, tan δ, is ultimately then equal to

$$tan\ \delta = \frac{E_{loss}}{E_{storage}} \tag{6}$$

and is representative to the degree of dampening that a material expresses while experiencing a time-dependent, oscillatory stress. Previous work by Les et al. linked estrogen depletion with a significant decline in tangent delta [41], though further work is warranted to determine D-spacing's specific role in bone's viscoelastic properties. By modeling the periodicity of collagen molecules for sheep experiencing estrogen loss, this study aims to examine this relationship.

1.12 Purpose

Tangent delta presents an excellent opportunity to quantify a material's viscoelastic response without exceedingly complex mathematical endeavors. By combining the rheological elements of the spring and dashpot into an orientation representative of both the elastic and viscous behavior of collagen, a computational model may be assigned appropri-

ate viscoelastic properties. Sheep present an excellent proxy for observing patterns of bone formation in humans, and ovariectomy may be performed on animal subjects to simulate the intense estrogen loss associated with postmenopausal osteoporosis.

Utilizing these foundations, the finite element analysis (FEA) program Abaqus may be manipulated to observe the effects of estrogen depletion on a complex collagen D-spacing model. Because viscoelasticity is so fundamental to the mechanical response of bone, its inclusion in computational analysis is absolutely critical. With recent research pointing to a wide degree of D-spacing within collagen fibrils, examining the relationship collagen molecules have with their neighbors has massive potential ramifications in packing factor and mineralization control. Cortical bone especially undergoes strict patterning and uniformity in 2-D layers, so slight alterations to the periodicity may hold large ramifications, and mature sheep present a bone structure similar to that of human cortical bone.

Combining all of these concepts (viscoelastic parameters, a variety of D-spacing, complex relationships across many parallel molecules, etc), a finite element analysis must be performed. Through utilizing FEA, a determination can be drawn as to which parameters best predict experimental findings on bone viscoelasticity. Once validated, an FEA model may serve as a way to computationally test and develop new theories on the mechanical implications of D-spacing.

Thus, the purpose of this study is both to determine if complex collagen modeling influences the theoretical tangent delta of cortical bone and to observe if estrogen depletion has an effect on bone viscoelasticity as it relates to D-spacing. Because D-spacing is a dimension that exists between many collagen molecules, it is imperative that a computational model be created to reflect the important relationship collagen molecules have with their neighbors. To accomplish this, this study will model 200 subunits that each represent the relationship between collagen and hydroxyapatite. Additionally, a Gaussian distribution will be employed to approximate the established variability of the D-spacing parameter. Through implementation of a coding safeguard named a “spacer,” a computational model will be created with random variability while maintaining a tight biological relevancy.

For the purpose of observing the ramifications of estrogen depletion on compact bone viscoelasticity, this study will model both the D-spacing of sheep that underwent a sham

surgery (statistical control sheep) and the D-spacing of sheep that experienced an ovariectomy. Experimental data will be provided through the usage of both atomic force microscopy and dynamic mechanical analyzer. Through the creation and validation of FEA models, conclusions may be drawn on whether D-spacing plays an important role in the ongoing initiative to treat osteoporosis.

By examining the portion of sheep cortical bone that experiences tensile loading (the cranial region), a finite element program may be created to mimic the staggered array model pioneered by Hodge and Petruska. To ensure this FEA model builds upon existing efforts to computationally express D-spacing, this study will generate a complex model based on the foundation of computational research conducted by both Siegmund et al. [78] and Miguel Mendoza [52].

The hypothesis of this study is that modeling a complex, variable array of collagen molecules will yield results closer to experimental findings than previous FEA D-spacing endeavors, and that effects of long-term ovariectomy will yield significantly different tangent delta data from the viscoelasticity expressed by control sheep.

2 Methods

2.1 Model Basis

2.1.1 Siegmund Model

Because bone is a complex biomaterial, a complex computational model must be generated to accurately approximate the behavior of this material. It has been well established that a relevant approximation must include the following parameters: accurate geometry, oscillatory loading, and viscoelastic behavior. Furthermore, because it is established that the arrangement of D-spacing is not constant, the size and variability of each molecule demands accurate representation [3].

To address these complexities, a finite element model was created to closely approximate the arrangement of collagen molecules. This study's computational model is based on a theoretical model of D-spacing created by Siegmund et al. in 2008 [78]. This model by Siegmund was an evolution of a mathematical model generated by Jager and Fratzl in 2000 [20] that translated the Hodge and Petruska model (figure 6) [19] into a simpler periodicity model (figure 18b). The Jager and Fratzl model expressed the periodicity as collagen and mineral as mineralized collagen fibrils that may be represented as rectangular units with distinct collagen and hydroxyapatite regions.

The research by Siegmund et al. [78] observed that the degree of collagen cross-linking has an important impact on the energy absorption capability (tangent delta) of compact bone, a conclusion that was drawn through the creation of the theoretical model generated in figure 18. Though this model made by Siegmund is not especially complex, it lays a critical foundation for later FEA approaches and the basis of this current study.

The Siegmund Model [78] breaks down a mineralized collagen fibril into a discrete period of collagen molecules. Figure 18c examines this periodicity in the form of a unit cell. The unit cell is made up of both collagen molecules and mineralized regions. For the purpose of analysis, Siegmund simplified the computational model down further by modeling only a half of that unit cell in figure 18d. This extra simplification measure allows for the employment of boundary conditions along the normally shared edge. This boundary

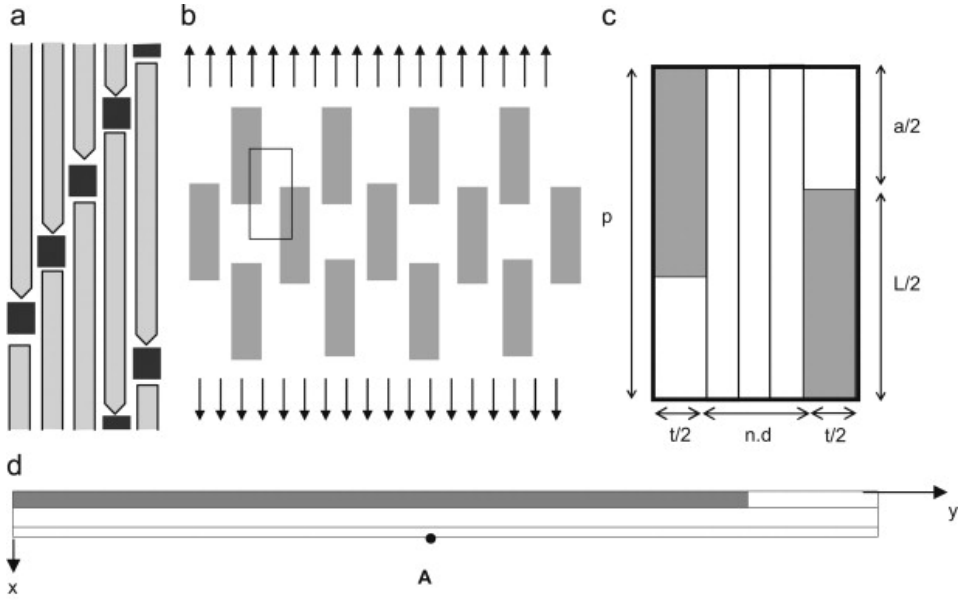

Figure 18: Siegmund et al. translated the staggered array model [4] into a computational form: a) this two-dimensional model represents the staggered array model for the D-spacing relationship between collagen and mineral, b) simplified model expresses the periodicity of collagen (white) and mineral (gray), c) a unit cell represents the basic geometry of the model, d) the half unit cell with actual geometric proportions utilized in the generation of a computational model [78]. Model dimensions may be viewed in table 1.

condition capitalizes on the unit cell's symmetry will experience equal and opposite forces, but the complexity of many unit cells in series remains wholly unaccounted for. This basic model underwent uniaxial loading to express the collagen molecules in tension.

The theoretical model geometry of figure 18 can be defined as mineral platelet width t, platelet length L, collagen helix diameter d, and number of collagen helixes n. The D-spacing (also commonly referred to as the periodicity) of this figure was determined as:

$$p = \frac{L+a}{2} \tag{7}$$

which was set to a constant 67 nm by Siegmund et al. based on the prevailing theory of the time [2, 20]. Mineral volume fraction, V_V^m, is based upon the mineral platelet length L, the thickness t, the distance between short faces of the mineral platelets a, and the distance between the long faces of the mineral platelets b. Dimensions for the Siegmund model can be seen in table 1. The mineral volume fraction is defined as

Table 1: Siegmund Model Dimensions. These dimensions describe figure 18 [78].

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
Collagen Thickness	d	1.5 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

$$V_V^m = \frac{L*t}{(L+a)(3*d+t)} \tag{8}$$

V_V^m is assumed to be a constant 0.30 based on the work of Currey [13, 20, 79] and Fritsch et al. [80]. Hydroxyapatite was modeled as a simple elastic isotropic solid with a modulus of E^m =100 GPa and a Poisson's ratio of $v^m = 0.28$ [35, 81]. The complex viscoelastic properties of collagen were hugely simplified down to modeling the molecules as representative of a homogeneous collagen triple helix in a wet environment: E^c=5 GPa and v^c=0.2 with a shear modulus of G^c=50 GPa [82].

The research conducted by Siegmund et al. is also notable because it incorporates the impact of collagen cross-linkages, finding their inclusion in a computational model is essential. Though this research generated a complex relationship to explain the contacts between adjacent materials, the key takeaway is that the linking between collagen molecules and hydroxyapatite must be included to ensure biological relevancy [78].

This Siegmund et al. model is an essential transformation of the Hodge and Petruska staggered array model into a computational form, but there are numerous shortcomings that warrant further refinement. A single half unit cell of a set periodicity is explored, though figure 18b shows a full periodicity of many collagen-hydroxyapatite complexes. The interactions between a greater number of units will no doubt better resemble the biologic function of these parallel molecules, an orientation that is so fundamental to bone composition. Though a strong foundation, this mechanical setup requires fine-tuning. To adequately simulate the material properties of a composite material such as bone, viscoelasticity must

Table 2: Mendoza Model Sample Dimensions. The model developed by Cal Poly graduate Miguel Mendoza explored modeling a half unit cell at multiple D-spacing values [52].

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

be modeled to obtain accurate deformation data.

2.2 Mendoza Model

Mendoza, in his 2013 California Polytechnic State University, San Luis Obispo MS thesis, expanded upon the Siegmund Model in a number of meaningful ways. Completed in 2013, the Mendoza Model addresses many shortcomings of the Siegmund et al. predecessor; Mendoza's computational model takes into account multiple D-spacing dimensions and–most critically–viscoelasticity [52]. Geometrically, Mendoza elected to also express a single half unit cell for FEA analysis. Dimensions from the Siegmund Model were scaled based on the periodicity. A sample of the core dimensions can be seen in table 2. Mendoza's model of the half unit cell may be seen in figure 19.

To reflect the current understanding of D-spacing existing over a range of values (figure 7), Mendoza took the Siegmund Model half unit cell and ran tests across a number of similar setups. Tests were run at a set period individually and applied to a single half unit cell. By iterating with a single dimension change at a time, Mendoza hoped to find which parameters from the Siegmund Model best corroborated experimental findings. These dimensions of interest can be viewed in table 2.

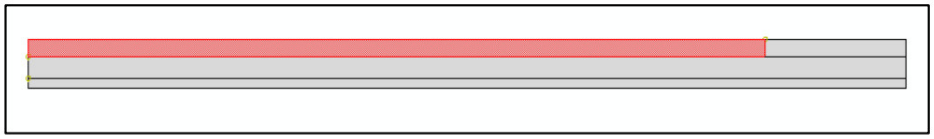

Figure 19: The model by Cal Poly graduate Miguel Mendoza is similar to the periodicity of figure 18, though the implementation of viscoelastic parameters brings the model closer to reality. The red portion is hydroxyapatite, while gray is collagen [52].

The biggest boon from Mendoza research was not from testing variable D-spacings, however, but the inclusion of viscoelastic elements that more closely brought the Siegmund

Model into agreement with actual bone tissue. Abaqus, the finite element software utilized in this study, allows for the input of user-generated commands in the form of a subroutine. These commands are written in coding language Python and can be manipulate to perform a variety of tasks within the finite element software.

A user subroutine was made in the finite element program, Abaqus, to implement the use of a rheological model known as the *Standard Linear Solid*. This user subroutine was originally developed by Frank Richter as part of his 2006 PhD at the Technical University of Berlin [83]. This Abaqus subroutine was then adapted by Miguel Mendoza to express certain rheological elements chosen in his study [84]. The flexibility of this subroutine makes it an attractive subroutine for implementation in this study. The fundamental equations that drive it are displayed in section 2.7.

Mendoza's refinement of the Siegmund Model through the inclusion of viscoelastic material properties is an essential step towards the development of a more comprehensive computational model. The next evolution of the Mendoza Model is clear: implementing a more complex model that incorporates many more collagen molecules and better reflects the true molecule-to-molecule variability in the D-spacing parameter.

Development of an accurate finite element model is impossible without validation against experimental data. In trying to develop a complex computational model to adequately model the viscoelastic properties of a composite bone structure, it is imperative to have a non-theoretical basis to assess the overall accuracy of a finite element model (FEM). Once a model has been confirmed to be accurate by simulating the same test environment that experimental samples underwent, we then have the justification to move forward and simulate new testing parameters computationally.

The following sections outline the process of obtaining and testing Rambouillet-cross ewes, as well as describing the critical steps taken to develop a finite element model to better present the inherit complexity and variability in collagen D-spacing. Specimen preparation and testing for the purposes of collecting experimental data were provided for this current study by Colorado State University and Henry Ford Hospital.

2.3 Experimental Data

2.3.1 Specimen Preparation (Provided By Colorado State University)

All animal testing conducted for the purpose of gathering experimental data was not performed at California Polytechnic State University, San Luis Obispo. Specimen preparation was performed at Colorado State University, while Henry Ford Hospital performed mechanical testing on bone samples. The University of Michigan, Ann Arbor supplied measurement data on experimental D-spacings from the test samples. These data were provided to this current study to aid in the development of biologically relevant finite element models.

Under local Institutional Animal Care and Use Committee (IACUC) approval, six Rambouillet-cross ewes from were obtained from Colorado State University's College of Veterinary Medicine and Biomedical Sciences for the purpose of obtaining experimental data. The selection of this specific sheep type was based on availability. The ewes lived in dry-lot conditions at an altitude of 1524 m at 41° degrees North latitude. The sheep were fed grass-hay and alfalfa on an ad libitum basis, and each was at a total weight of between 150 and 200 lbs. In accordance with Colorado State University's IACUC, the sheep underwent anesthesia before surgery. Once anesthetized, ewes were subjected to one of two procedures: either an ovariectomy (OVX)(to incur estrogen depletion) or a sham surgery to control variability within the experimental units. As such, three sheep underwent the OVX surgery and three received the sham surgery treatment to become control sheep.

These surgeries were performed in August of 2001 and were sacrificed 12 months later in August of 2002. The decision was made to sacrifice these sheep at 12 months in hopes of gaining a clearer picture of the long-term effects for estrogen depletion and its mechanical repercussions. The left radius and ulna (often fused together in adult sheep) were harvested from each of the six sacrificed Rambouillet-cross ewes for the purposes of this current study. These bone samples were then stored at $-20°$ Celsius in calcium-buffered saline solution.

2.4 Testing (Provided By Henry Ford Hospital and the University of Michigan, Ann Arbor)

Ewe bone samples were divided into six distinct anatomical sectors (craniomedial, cranial, craniolateral, caudomedial, caudal, and caudolateral) (figure 20) [58]. For the purpose of this study, only the cranial (occasionally abbreviated to "Cran") sector was examined. This decision was based upon previous findings that suggested a large amount of viscoelastic variation existed between bone regions [52]. By isolating a single sector of the bone area, comparisons can be observed more easily between OVX and control sheep viscoelasticity.

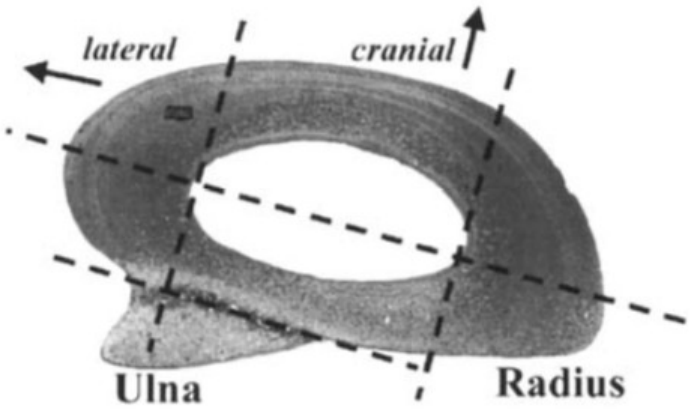

Figure 20: The radius and ulna of the adult ewe are often fused together. This cross-section may be divided up into six distinct anatomical sectors, denoted by the dashed lines. Multiple beams may be made from a single sector, and a random beam was selected for testing [41].

These samples were each machined into small cortical bone beams. Each radius/ulna was divided into a total of up to 25 beams, each classified by anatomical section. Random beams were selected for testing from applicable anatomical test sector, cranial. Beams were machined to dimensions of 1.75 mm x 1.75 mm x 19.0 mm and stored in a 0.9% saline solution at -20° Celsius. This machining was accomplished via Exact Technologies Inc. equipment at Henry Ford Hospital. Dynamic mechanical testing was performed on the cortical beams. Drying and ashing was performed on the distal ends of the samples to make a determination as to the density of the bone. Microradiographs were created for the purpose of histology measurements on cross-sections at 100 μm thickness. Atomic force microscopy (AFM) was utilized to measure a number of essential parameters such as

means and standard deviations of D-spacing for samples of each surgical treatment. These AFM measurements were accomplished by the Department of Chemistry at the University of Michigan, Ann Arbor.

A dynamic mechanical analyzer (DMA 7e, PerkinElmer) was used at Henry Ford Hospital to measure the viscoelastic properties of the bone samples. Data were collected based on the orientation of the cortical bone beam samples undergoing three point bending; in vivo stresses were determined to be compressive or tensile based on the location of the bone sample; the caudal sector is indicative of compression, whereas the cranial sector experiences tensile forces. As collagen molecules are modeled in tension in figure 18b, the cranial sector is an appropriate choice.

Three point bending was performed with a sample placed on an apparatus with outer support separated by a 15 mm distance. Testing was conductive at 37° Celsius. Static loading was performed at 550 mN, while dynamic loading took place at 500 mN. The testing was non-destructive. The gathered information was used to determine the bone's viscoelastic properties in terms of the tangent delta. Tangent delta data were recorded at increments of 0.2 Hz from 1-20 Hz. For the purposes of this study, 1, 3, 9, and 15 Hz were utilized.

2.5 Model Development

2.5.1 Complex Model

The models created by Siegmund et al. and Mendoza each represent collagen periodicity in a single half unit cell of about 67 nm. Because ample research shows that periodicity patterns extend for a full 40 μm [3], there is a huge opportunity to extend the length and overall complexity of the computational model. Ultimately, the goal of a more complex model is determining whether a more involved computational model may accurately match experimental data. If a more complex model can accomplish a greater level of realism, the finite element model will then be validated and can be confidently experimented with to obtain new data and insight into the implications of D-spacing on bone viscoelasticity.

There were two initial considerations taken into account in generating a model of a

greater complexity: the periodicity and the number of unit cells. Periodicity, p, was set at a constant value in the Siegmund Model of 67 nm. The Mendoza Model expanded on this while still modeling only one half unit cell, creating separate models to individually represent multiple sample periodicities: 61 (Low D-spacing), 67 ("Normal" D-spacing), and 73 nm (High D-spacing).

Though modeling three different D-spacings separately on a half unit cell is a commendable advancement toward representing the staggering of D-spacing, the half unit cell inadequately represents the important system of parallel collagen molecules. Because the presence of many molecules is theorized to have a large impact on the packing factor and overall viscoelasticity of bone [24], the system of fibrils is a key component to the overall bone tissue. Furthermore, research has revealed that D-spacing varies by 1 nm within a bundle and 10 nm between bundles [3]. Modeling a half unit cell prevents any exploration into that research. By modeling a full unit cell (i.e. two rows of half unit cells) in series with many other collagen-hydroxyapatite complexes, a better understanding may be developed as to the viscous implications of complex D-spacings.

For these reasons, periodicity in this current study's model–to be referred to simply as Complex Model–more accurately represents these variabilities. Because validation of the computational model is of a high concern, two different models were selected in an effort to determine which is the most suitable proxy for experimental data. Additionally, because the implications of estrogen depletion are of great interest in osteoporosis research [58], both a control and OVX model were created. The cranial bone sector from figure 20 was selected as there is precedent for research on this section in tension [52, 58]. By keeping the bone section consistent, distinctions between surgical treatments will be more evident in analysis. Experimental data collected via AFM is summarized in table 3. This dataset serves as the basis for this complex model's geometry.

Table 3: Complex Model D-spacing Gathered from Experimental AFM Testing. These parameters were used in a random distribution (Gaussian) to create a model which more authentically represents biologic variability. These data were provided by the University of Michigan, Ann Arbor.

Model (Surgical Treatment, Bone Sector)	D-space Mean (μm)	D-space St. Deviation (μm)
Control, Cranial	0.06842	0.00130
OVX, Cranial	0.06694	0.00097

Control sheep and OVX sheep each have variable D-spacing measurements, represented by a mean and standard deviation. This information represents an array of potential periodicities. Creating models set at the mean measurement would discount this inherit variability. For this reason, an extensive Python script was created to represent the complex arrangement of random periodicities that can be found utilizing the data in table 3. This arrangement of collagen-hydroxyapatite complexes alternates between unit cells, a fact that had to be recreated within the Python coding. This Python script may be viewed in appendix K. The standard deviations found in the experimental data collection echo previous findings that suggest collagen D-spacing within a single beam tends to differ by about 1 nm [3].

The number of unit cells was also an important parameter for computation. Opposed to the current approach of modeling a single collagen-hydroxyapatite complex, the Complex Model was made to represent 2 x 100 half unit cells. The selection of 100 units was based off of an observation that D-spacing patterns persist experimentally for about 40 μm in a single bundle [3]. For this reason, modeling collagen molecules of 67 nm length 100 times would remain comfortably biologically relevant without exceeding this threshold for D-spacing homogeneity. Additionally, we are confident that many molecules are found aligned in parallel (illustrated by figures 7, 8, 18), and this is an attractive parameter for modeling in the pursuit of biological relevance. That said, computational power presents another important limiting factor; a theoretical model more dense than 2 x 100 would be impractical to test given the significant run time of this complex model alone; runs took about 8-10 hours per model per frequency.

Each singular half unit cell in the 2 x 100 model has a random periodicity value based on table 3. The randomization is based on a Gaussian distribution [3]. For example, a

Table 4: Complex Model Row Lengths. Implementation of Gaussian randomization means that the top and bottom row will be of differing lengths. Four versions exist for both core models (Control, Cranial and OVX, Cranial).

Model (Treatment, Sector Version)	Row Lengths (μm)	
	Top Row, L_1	Bottom Row, L_2
Control, Cranial 1	6.85573	6.86152
Control, Cranial 2	6.86835	6.85995
Control, Cranial 3	6.84309	6.84149
Control, Cranial 4	6.81504	6.85025
OVX, Cranial 1	6.69219	6.68633
OVX, Cranial 2	6.69962	6.69186
OVX, Cranial 3	6.68794	6.67918
OVX, Cranial 4	6.70369	6.70805

single half unit within the Control, Cranial model may have D-spacing of 68.4 nm, while a neighboring half unit may be randomly generated as having a D-spacing of 69.2 nm. Other dimensions initially outlined in the Siegmund Model (i.e. a, d, L, etc) [78] were scaled based on the periodicity of a particular unit cell. Similarly, the ratio of mineralization, V_V^m, per half unit cell was maintained at 0.30. Because changes in periodicity do not impact the width of the model, each half unit cell has an overall thickness of 3.5 nm [78]. This style of random D-spacing assignment persists for all 2 x 100 half units, ultimately generating a model more similar to Hodge and Petruska's foundation [4].

This Gaussian randomization parameter means that each model is unique, though each unit cell is restricted to the tight periodicity standard deviation highlighted in table 3. To observe the variability intrinsic to his randomization function, four versions of both models (Control, Cranial and OXV, Cranial) were generated.

The Gaussian randomization for each unit cell creates an overall length for both the top row and the bottom row for the 2 x 100 model. Because each of the 100 half units per row are of a random periodicity, the top and bottom rows are not of an equal length. The dimensions for all the created models (8 total, 4 of versions of both core models) can be seen in table 4.

2.6 Spacer

Given the method in which the Python code generates a model in Abaqus, extra consideration had to be given to the overall geometry of the Complex Model. The randomly generated row lengths outlined in table 4 imply that the top and bottom rows will not be of equal lengths. Because the Python code draws the model as a rectangle and then fills in the geometry with the appropriate collagen-hydroxyapatitie complexes (proportionate to the D-spacing of each half unit cell), rows will ultimately not align perfectly for mechanical assignment.

The variable top and bottom row lengths brings about a small challenge. The finite element software applies the loading conditions at the right end of the completed model. With uneven row lengths, however, there will not be a consistent edge on which to apply the load. Sharp changes in geometry yield stress concentrations and will give unexpected results [85]. Because the model is created with a sinusoidal pressure applied along the thickness of the model's free end, avoiding these mechanical variables is critical to approximate biologic function as closely as possible.

For example, consider a simple 2 x 2 half unit cell model utilizing the experimental data for "Control, Cranial." On the top row, one half unit cell may have a D-spacing of 68 nm while the neighbor has a D-spacing of 69 nm. On the bottom row, both half unit cells may have a period of 68 nm. This means the longest row length (top) will be 137 nm. Thus, when the model is drawn in Abaqus, a rectangle of 137 nm x 3.5 nm is created and partitioned for each of the 4 half unit cells. This means that for the shorter, bottom row, 1 nm of free space would exist at the terminus of the final bottom row half unit cell. Making sure this free space remains biologically relevant is an additional challenge that can be overcame with use of a *spacer*.

Illustrated in blue in figure 21, the spacer completes the geometry of the shorter row. This small space guarantees an edge for uniform application of loading conditions on the right edge.

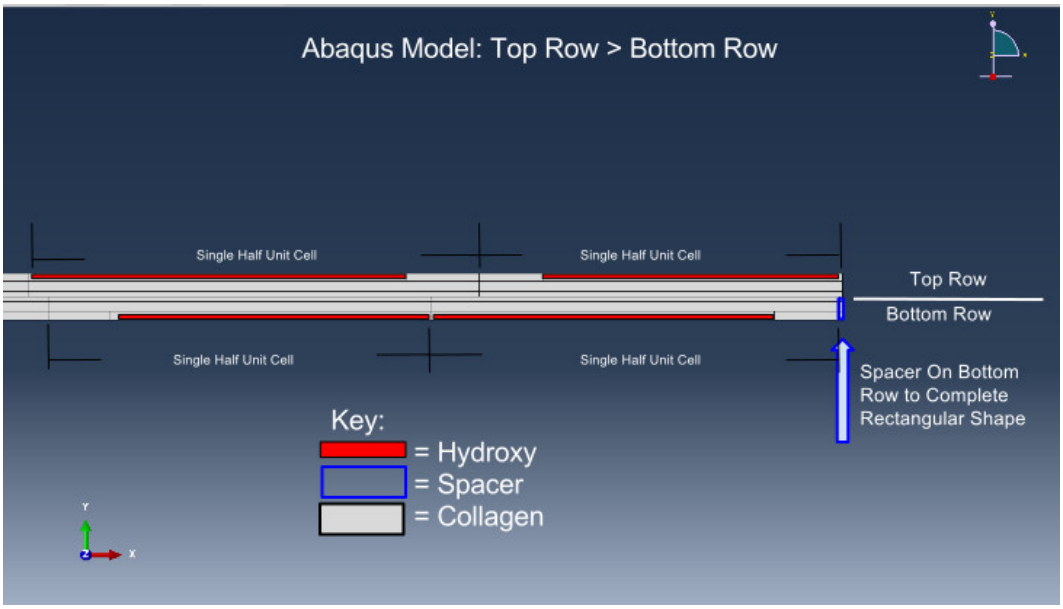

Figure 21: Shown in blue, the spacer completes the geometry between the randomly generated lengths of the top row and bottom row to create a rectangle–a geometry essential for applying uniform loading in Abaqus.

The spacer must also be assigned material properties for the Abaqus simulation to run. This presents a new question: what material should this small spacer unit be assigned? Because the half unit cell is predominately collagen (approximately 70%), the entire area of the spacer is given collagen assignment. To ensure that this does not meaningfully influence the results and maintain model validity, two safeguards were written into the Python script.

First, if a spacer is so large that its inclusion changes the combined collagen/hydroxyapatite ratio (of both the single neighboring half unit cell and spacer) by more than 5%, Abaqus will reject the model and the Python script will attempt to redraw a biologically valid model. This process continues until a valid model is drawn. The distinction of 5% is arbitrary, but ensures that no collagen-hydroxyapatite complex possesses a completely unrealistic material composition. Keep in mind that this 5% is based on the combined material properties of the fully collagen spacer and the neighboring half cell only. A change of a full 5% in only one half unit cell would change the material ratio of the entire 2 x 100 model negligibly. Because this change is so small and the $V_V^m = 30\%$ parameter is only an estimation, there is little worry that the spacer meaningfully skews the results.

The second safeguard the spacer function may perform is to completely swap the re-

mainder area with another half unit cell. This backup plan only activates if the randomly generated spacer has a length greater than one standard deviation under the model's mean periodicity. For example, a Control, Cranial model with a spacer length of 67.5 nm would be swapped with a half unit cell possessing a period of 67.1 nm ($mean - 1 * St.D = 68.4\ nm - 1.3\ nm$). The remaining 0.4 nm (67.5 nm - 67.1 nm) length would serve as the new, smaller spacer, and would be assigned collagen material properties so long as the aforementioned 5% material tolerance remains unviolated. Though this method of adding additional units to the shorter row was never used for the eight models created in this study, the function exists to ensure accuracy in any future research utilizing the Complex Model.

Through the inclusion of this spacer security measure in the Python code, the heavily randomized model creation is ensured a perfect rectangular geometry. With this safeguard, users may be confident in Abaqus's ability to apply loads and boundary conditions without any major mechanical variables.

2.7 Materials

Mineralized hydroxyapatite can be simply modeled as an elastic isotropic solid, $E^m = 100$ GPa and $v^m = 0.28$ [2, 35, 81]. Collagen, however, calls for a more in-depth endeavor in accurately modeling viscoelasticity.

The first simplifications come in the form of modeling in only two dimensions, modeling plane strain [78], and assuming the materials are isotropic. By electing to model in 2-D, the simplifications to the stiffness matrix may be seen in figure 22.

a)
$$\begin{bmatrix}\sigma_{xx}\\ \sigma_{yy}\\ \sigma_{zz}\\ \sigma_{yz}\\ \sigma_{zx}\\ \sigma_{xy}\end{bmatrix} = \frac{E}{(1+\nu)(1-2\nu)}\begin{bmatrix}1-\nu & \nu & \nu & 0 & 0 & 0\\ \nu & 1-\nu & \nu & 0 & 0 & 0\\ \nu & \nu & 1-\nu & 0 & 0 & 0\\ 0 & 0 & 0 & 1-2\nu & 0 & 0\\ 0 & 0 & 0 & 0 & 1-2\nu & 0\\ 0 & 0 & 0 & 0 & 0 & 1-2\nu\end{bmatrix}\begin{bmatrix}\varepsilon_{xx}\\ \varepsilon_{yy}\\ 0\\ 0\\ 0\\ \varepsilon_{xy}\end{bmatrix}$$

b)
$$\begin{bmatrix}\sigma_{xx}\\ \sigma_{yy}\\ \sigma_{xy}\end{bmatrix} = \frac{E}{(1+\nu)(1-2\nu)}\begin{bmatrix}1-\nu & \nu & 0\\ \nu & 1-\nu & 0\\ 0 & 0 & 1-2\nu\end{bmatrix}\begin{bmatrix}\varepsilon_{xx}\\ \varepsilon_{yy}\\ \varepsilon_{xy}\end{bmatrix}$$

Figure 22: a) The 3-D stiffness matrix for plane strain is commonly selected for material assumed to be isotropic. This simplification reduces further b) when considered in only 2-dimensions [86].

The material definition of collagen is based on the viscoelastic system of equations utilized in the Mendoza Model and developed by Richter [52, 83]. These equations were adapted into the appropriate rheological form [84]. This user-defined material subroutine (UMAT) expresses the viscoelasticity parameters of the solid linear model. This UMAT may be viewed in appendix H.

Because bone tissue may experience both stress relaxation and creep, rheological elements were selected to mimic that important material behavior. Specifically, the Kelvin-Voigt version of the Standard Linear Solid was chosen (figure 23).

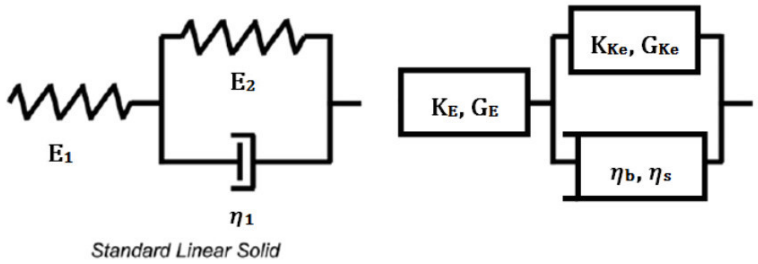

Figure 23: (Left) The one-dimensional Standard Linear Solid possesses an elastic element (spring) in series with a Kelvin-Voigt body. (Right) A 3-D version of the rheological model features the springs and dashpot replaced by shear and bulk moduli/viscosities [52, 87]. This secondary form allows simple assignment of material moduli to each rheological element. This figure has been redrawn to keep the E_1 and E_2 consistent with other included figures.

The Kelvin-Voigt version of the Standard Linear Model was chosen over other rheo-

logical alternatives as it possesses a simpler mathematical form while still adequately expressing viscoelastic behaviors characteristic of bone [87]. As seen in figure 23, the rather abstract spring and dashpots may be simply replaced with elastic viscous coefficients [83].

Rheological models are intrinsically one-dimensional. To create the linear viscoelastic material utilized by this current study, a transformation had to be made utilizing the 3-D form. This 3-D form allows for the input of elastic and viscous material coefficients and may be simplified down to the characteristic one-dimensional form of a rheological model. The three-dimensional equation may be written as:

$$\begin{aligned}(1+\frac{G_{Ke}}{G_E})\sigma_{ij}+(\frac{K_{Ke}}{K_E}+\frac{G_{Ke}}{G_E})\frac{\sigma_{kk}}{3}\delta_i j+\frac{\eta_s}{G_E}\dot{\sigma}_{ij}+(\frac{\eta_b}{K_E}+\frac{\eta_s}{G_E})\frac{\dot{\sigma}_{kk}}{3}\eta_{ij}\\ =2G_{Ke}\varepsilon_{ij}+(3K_{Ke}-2G_{Ke})\frac{\varepsilon_{kk}}{3}\delta_{ij}+2\eta_s\dot{\varepsilon}_{ij}+(3\eta_b-2\eta_s)\frac{\dot{\varepsilon}_{kk}}{3}\delta_{ij}\end{aligned} \tag{9}$$

Indices *i* and *j* indicate the tensor component of stress and strain, and the dot is a derivative with respect to time. The subscript *kk* is representative of the trace of the tensor, which is the sum of each normal component. This *kk* subscript is illustrated in the sample equation:

$$\sigma_{kk}=\sigma_{xx}+\sigma_{yy}+\sigma_{zz}$$

δ_{ij} is the Kronecker delta, a function of two variables (in this case, *i* and *j*) that describes the tensor in a basic binary form [88].

This three-dimensional form allows the input of applicable material coefficients in the Standard Linear Solid. Though rheological elements lack a biologic equivalent, the input of these known material values helps to generate an applicable viscoelastic material property for computational modeling. The coefficients seen in the 3-D version of figure 23 each may be assigned a material characteristic to bring this theoretical setup closer to biologic relevance.

The subscript *E* represents the isolated spring element in the Standard Linear Solid, while *Ke* refers to the spring element within Kelvin-Voigt body. The elastic coefficients in this model are *K* and *G*, where *K* represents the bulk modulus and *G* represents the shear modulus. Each *K* and *G* are reflections of the spring elastic modulus, *E*, and the Poisson's

ratio, v. Selecting an elastic modulus and Poisson's ratio for viscoelastic representation in the computational model is the first step in bringing a finite element analysis closer to biologic equivalence.

$$K = \frac{E}{3(1-2v)} \tag{10}$$

$$G = \frac{E}{2(1+v)} \tag{11}$$

Because there is a spring element both outside the Kelvin-Voigt body and within it, an independent elastic modulus and Poisson's ratio may be applied to each spring individually. E_1 represents the elastic modulus chosen for the lone spring element, and E_2 represents the elastic modulus of the spring found in the Kelvin-Voigt body. These property assignments may be viewed in appendix G. Again, because rheological elements are completely theoretical, this current study decided to utilize a relationship between each spring element known as an *effective modulus* (discussed further in equation 15).

Because this viscoelastic relationship is so fundamental to bone, these material behaviors are an essential piece of the rheological building block. η_b and η_s represent the bulk and shear viscosity, and their form is similar to the bulk and shear moduli:

$$\eta_b = \frac{\eta_1}{3(1-2v)} \tag{12}$$

$$\eta_s = \frac{\eta_1}{2(1+v)} \tag{13}$$

where η_1 is an assigned material property without adequate biologic relevancy. The three-dimensional formula may be simplified further through making the assumption that the Poisson's ratio of the elements and perpendicular components of stress and strains are fixed at zero. This transformation allows for simple calculations of viscoelastic behaviors such as stress relaxation and creep (appendix C).

This simplifies the equation down to a more approachable, one-dimensional form of the Standard Linear Solid:

$$\sigma + \frac{\eta_1}{E_1 + E_2}\dot{\sigma} = \frac{\eta_1}{1 + \frac{E_1}{E_2}}\dot{\varepsilon} + \frac{1}{\frac{1}{E_1} + \frac{1}{E_2}}\varepsilon \tag{14}$$

Though these equations well describe the process of translating the theoretical Standard Linear Solid into more quantifiable elements, additional steps must be taken to make these material definitions relevant in Abaqus. Creating a UMAT requires both a method for updating stresses and solution-dependent state variables at the end of each increment, and implementation of the material Jacobian matrix, $\delta\Delta\sigma/\delta\Delta\varepsilon$ [89, 90]. These requirements were incorporated into the UMAT created by Dr. Frank Richter [83] and modified by Mendoza [52]. The complete viscoelastic UMAT may be seen in appendix H.

While the UMAT is effective for modeling rate dependent behaviors such as creep, there are still six core parameters that must be selected to utilize these equations: two elastic moduli for the spring elements, a dashpot viscosity, and three corresponding Poisson's ratios. The simplification of setting the Poisson's ratios to zero alleviates some of the complexity, but moduli must still be selected to undergo this viscous material definition. Because rheological elements do not have a simple biological equivalent, selecting an appropriate viscosity is imperfect [7].

To effectively simulate compact bone, collagen material properties to be used in the viscous transformations were based on existing literature [2, 7]. As such, the modulus for collagen to be used for spring stiffnesses was 2 GPa, while the Poisson's value was set to 0.2 in the user subroutine [78]. This 2 GPa modulus for spring stiffness was set to the *effective modulus*, equivalent to the following:

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

With an effective modulus for collagen of 2 GPa, E_1 and E_2 were set to 3 GPa and 6 GPa, respectively. This decision was based off of the conditions implemented in the Mendoza Model's application of the Richter UMAT. Though the Siegmund Model employed a modulus of 5 GPa, it did not incorporate viscoelastic parameters. Given the biologic ambiguity of a rheological spring and dashpot, this decision is not an easy one.

The use of the viscoelastic parameters requires a viscosity be assigned to the dashpot in

the Kelvin-Voigt body. This study utilized a viscous dashpot of 1.25 GPa*s. This decision was based on the dashpot's ability to best fit experimental data found in previous studies [52]. Multiple dashpots are expressed graphically in the Results. Further exploration of these concepts can be viewed in the Discussion.

These values were manually entered into the Abaqus input file created for each model. The flexibility of this input file may be viewed in appendix G. This input file was run in tandem with the UMAT originally created by Dr. Frank Richter to model viscoelasticity. These material property assignments are passed into the Richter UMAT when the Abaqus job is run to approximate rate-dependent material behavior.

2.8 Boundary Conditions and Loads

Utilizing the global coordinate system, boundary conditions were applied in the following manner: vertical displacement (y) was set to zero from $(0,0)$ to $(L_F,0)$, where L_F is the length of the longer row (due to the geometric demands explained in section 2.6). The horizontal displacement (x) was constrained to zero from $(0,0)$ to (0, 3.5 nm) (along the left edge). Boundary conditions can be seen as the small, orange triangles in figure 24.

For example, the computational model for "Control, Cranial 1" would be generated with rectangular dimensions of 6.86152 μm x 3.5 nm. The bottom row of Control, Cranial 1 has a length of 6.86152 μm (as seen in table 4), making it longer than this specific computational model's top row. This would designate 6.86152 as Control, Cranial's L_F. A perfect rectangle would be created with these dimensions, and the vertical displacement boundary conditions would be applied to the bottom edge. The horizontal boundary condition would be applied in the same way to the left edge in every mention, as the thickness of the Complex Model is constant.

The constriction of horizontal displacement on the left edge exclusively helps to establish asymmetry in the bounding conditions, simplifying the analysis. By applying a load to the unconstrained end, the equal and opposite force experienced by the left restricted edge simulates the tension that collagen molecules undergo, as seen in figure 18b. Figure 24 conveys the size, shape, and intricacies of the Complex Model.

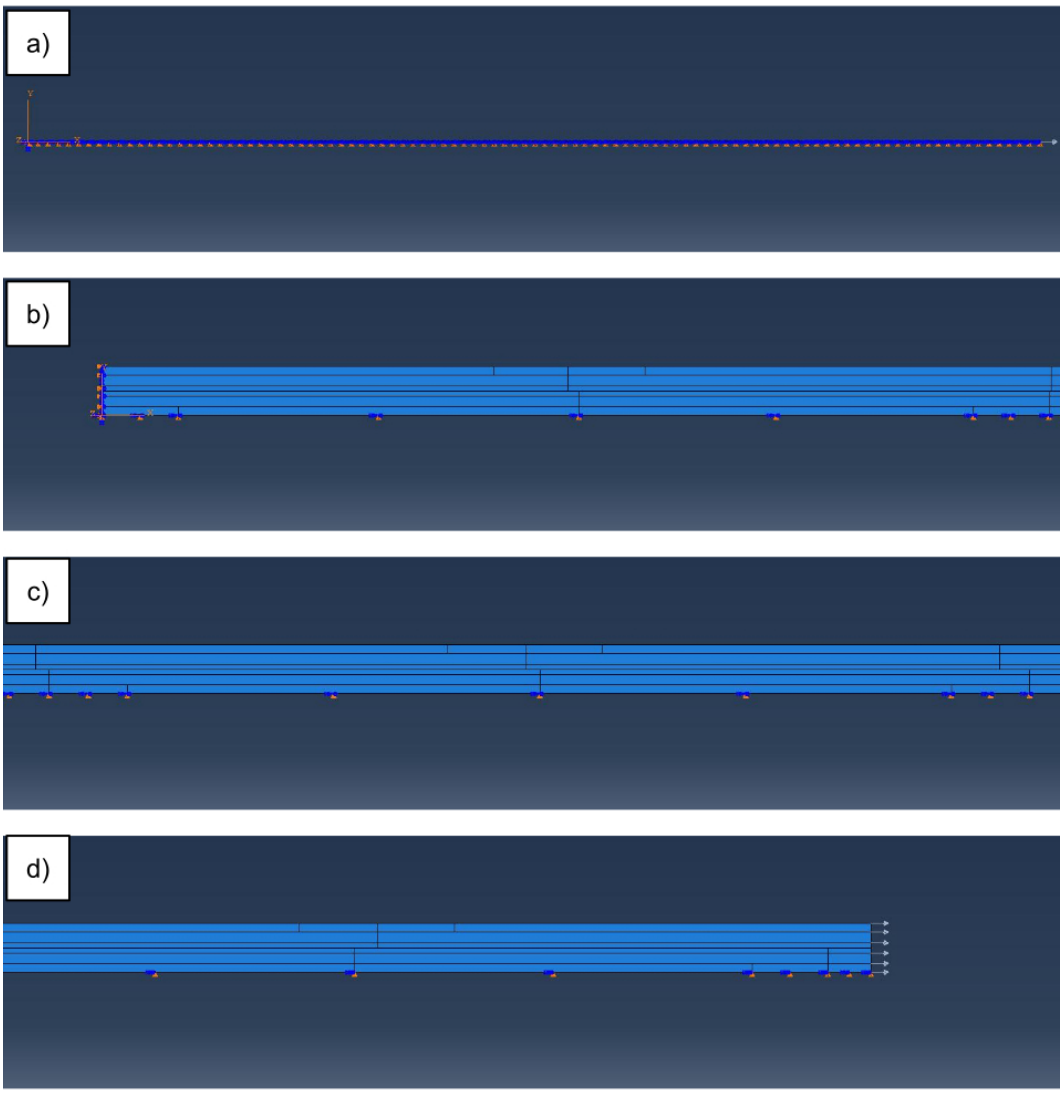

Figure 24: a) The full 2 x 100 Complex Model can be more easily interpreted by examining discrete sections: b) the left end constricts movement in the X-direction by fixing the left-most edge, c) a small portion of the midshaft continues the constrain in the Y-direction on the bottom axis, and d) the right, free end is applied the the sinusoidal pressure load.

This load at the free edge replicates the rate-dependent loading experienced by the test sheep. A uniaxial tension was applied from (L_F,0) to (L_F, 3.5 nm) (seen as the gray arrows in figure 24). This load underwent 20 cycles of either 1, 3, 9, or 15 Hz testing. Each cycle is comprised of 20 individual increments, making a step function of 400 total increments. The increment size may be viewed in appendix G. The amplitude of this sinusoidal waveform was selected at 3.36 MPa. This amplitude is reflective of the maximum pressure. This value was chosen based on the approximated stress experimental bone samples experienced during DMA testing. This derivation may be viewed in appendix B.

Each model was run at the four test frequencies separately. Results were analyzed in the form of tangent delta to be validated against experimental findings. The decision to make four versions of both "Control, Cranial" and "OVX, Cranial" models was made to help create a better understanding for the variance experienced based on the randomization of periodicity that dictates the length of all half unit cells. Creating these replicates also permits statistical implementation. By testing the core two models four different times under the same random generation conditions at each frequency, a clearer picture may be established as to the variability in tangent delta measurements.

Because each collagen-hydroxyapatite half unit cell is drawn as a single piece and then appropriately assigned material properties, the interface between differing materials is perfectly bonded. This decision was made based on the findings of Siegmund et al. that expressed the importance of material interactions in theoretical versions of the staggered array model [78].

2.9 Mesh Convergence

The number of elements included in a finite element analysis dictates the accuracy of the model, as each element presents a number of integration points available to generate composite output data. More elements means a higher accuracy, but comes at the cost of a higher computational time. A mesh convergence operation describes the process of determining the largest element seed size possible while still receiving consistent results.

A plane strain model was selected based on the work of previous studies that determined the plate-shape geometry of hydroxyapatite demands it [78]. Because this analysis takes

place in 2-D, quadrilateral elements were selected to match the rectangular geometry of the model. Control, Cranial 1 was arbitrarily selected as the model used to test convergence.

The node utilized for inspection, node 1140, is identified in figure 25. Analysis for displacement was run at a variety of seed sizes. These displacements were graphed against the degrees of freedom afforded by the degree of seeding. The mesh can be seen to start converging at a seed size of 0.0005 (figure 26). Finer mesh types were also selected, but the computational time was inefficient, especially considering the immense run time and file size for a single datum point in the Complex Model. Additionally, Abaqus could not consistently support a seed size smaller than 0.0005 while applying that level of accuracy to a massive model. As such, Abaqus crashed occasionally, and 0.0005 was selected.

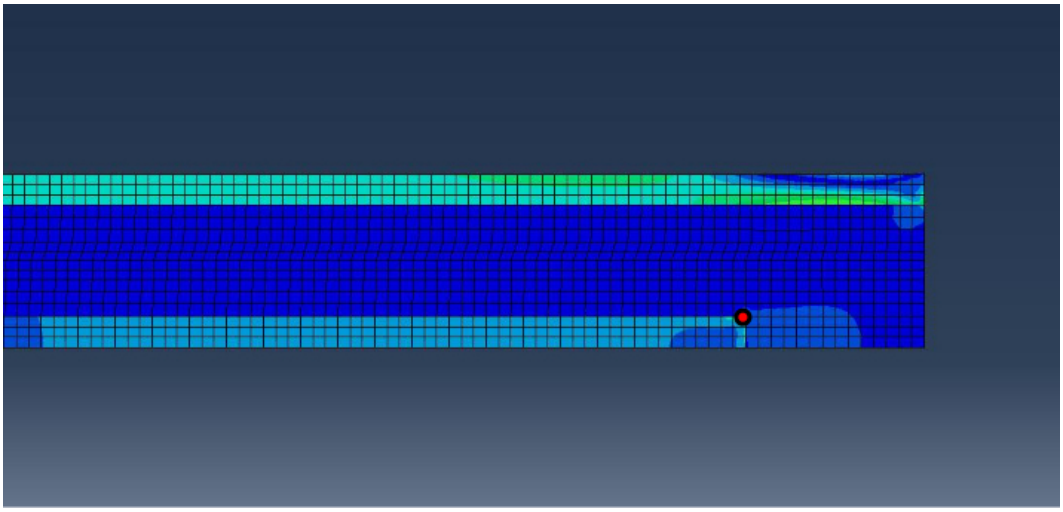

Figure 25: Node 1140, shown in red, at the collagen-hydroxyapatite interface of the model's free end was used to monitor mesh convergence.

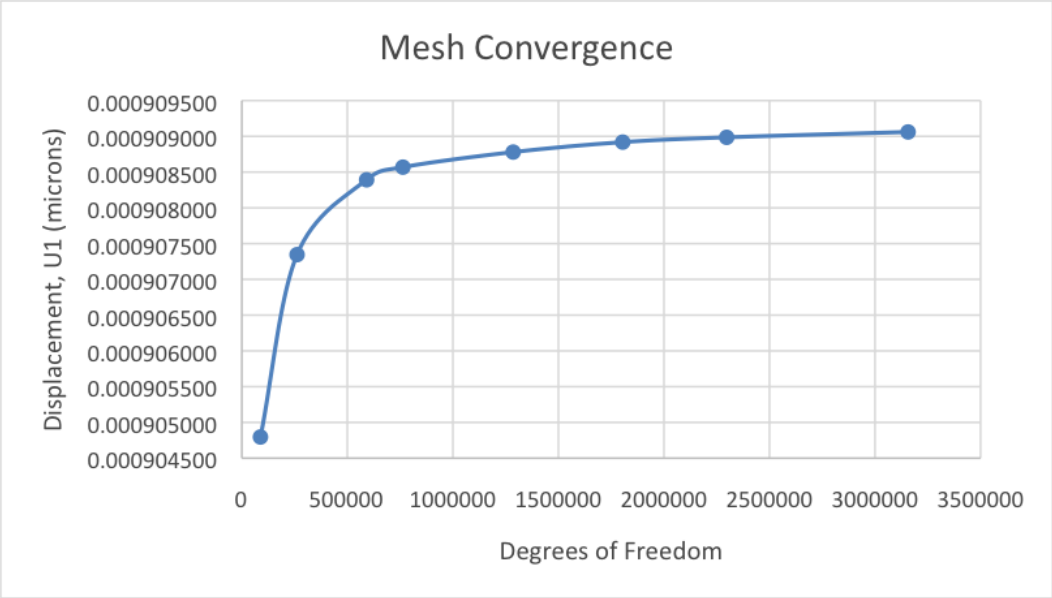

Figure 26: Mesh convergence begins to take place at about 1300000 degrees of freedom, correlating with a seed size of 0.0005. Seed sizes much smaller than 0.0005 were too fine and occasionally crashed this analysis.

2.10 Model Validation

Because the UMAT for collagen viscoelasticity was originally created by Frank Richter [83], validation of the material parameters was critical. Though Mendoza's research was successful at displaying the implementation of the Richter UMAT [52], it was important to confirm these findings personally.

Confirmation of the viscoelastic parameters of collagen were accomplished with simple hand calculations. Because creep and stress relaxation are characteristic properties of bone tissue that must be modeled, a simplified form of these equations was tested by hand. Because this study opted to use the Standard Linear Solid to express these material behaviors, the corresponding creep and stress relaxation equations associated with this specific model were tested. These derivations may be viewed in appendix C.

A single one-dimensional truss element was created in Abaqus and assigned arbitrary values for E_1, E_2, and η_1 (the dashpot viscosity). Example values are 50 GPa, 100 GPa, and 75 GPa*s, respectively. The model was then assigned an appropriate force and boundary condition and observed at varying time increments, t. For the 1 Hz model, this time increment is 0.05 sec. The results of this basic FEA were confirmed by hand calculations using

the same moduli, thus confirming the validity of this UMAT.

2.11 Post-Processing

Data retrieval from the Complex Model was accomplished through a number of steps. Each computational model in Abaqus was set to output displacements for the free end of the computational model (experiencing the sinusoidal tensile pressure, figure 24d). Displacements were gathered at each node along the free edge. Because the sinusoidal force took place over 400 increments, displacement data were recorded for each increment at all applicable nodes. These data points were collected through a Python script (appendix I), and then interpreted through MATLAB code. This MATLAB code may be viewed in appendix J.

MATLAB coding converted the displacements at each node into usable tangent delta data. This process was facilitated through averaging the displacement value at each time increment. Average deflection was translated to a strain by dividing the total displacement by the total model length (dictated by the longest row of a particular model in table 4). Strain was then plotted using two separate algorithms to collect information on the tangent delta. These algorithms accomplished this through determining the phase shift between a sinusoidal stress application and the sinusoidal strain response. A Curve Fitting function plotted this relationship, and minor adjustments were made to the starting amplitude to best fit computational data to a stress-strain function.

2.12 Statistical Analysis

Randomization was critical to both the experimental data collection and model generation for the purpose of statistical analysis. As previously discussed in section 2.4, bone samples were selected for testing based on anatomical sector. Each bone sample may be machined into six separate anatomical sectors–for this analysis, only a single sector (cranial) was examined–which may then be machined into multiple individual test beams (up to 25 per bone). A single beam was selected at random for the sector of interest, and that beam was utilized to collect experimental data.

Randomization was fundamental to this current FEA study. Though Siegmund and Mendoza provided an excellent basis to build upon, neither of their computational models

Table 5: A Summary Table for Statistical Treatments.

Statistical Treatment	Abbreviation	Treatment Levels
Model	Mod	Complex Model or Experimental Data
Surgical Treatment	Trt	Control or OVX Surgery
Test Frequency	Frq	1, 3, 9, or 15 Hz
Interaction between Mod and Trt	Mod*Trt	Effect of Mod depends on Trt
Interaction between Mod and Frq	Mod*Frq	Effect of Mod depends on Frq
Interaction between Trt and Frq	Trt*Frq	Effect of Trt depends on Frq
Interaction between all treatments	Mod*Trt*Freq	Effect of all treatments dependent

incorporated randomized generation parameters. Because recent studies have confirmed the variance in collagen D-spacing even within a single bundle [3], implementing variant model generation parameters is important to the authenticity of this computational analysis. Randomization also permits statistical analysis to more accurately answer the fundamental purposes of this study: does modeling a complex D-spacing arrangement of many collagen molecules significantly impact tangent delta, and does estrogen depletion significantly alter the tangent delta of cortical bone?

The Mendoza Model only reported a single datum point for each test frequency of his Normal D-spacing model, and thus statistical analysis cannot be performed against his results. The Complex Model does, however, implement multiple randomized generation parameters for both Control, Cranial and OVX, Cranial FEA models. This degree of randomness permits the usage of Analysis of Variance (ANOVA) testing to determine if the response variable, tangent delta, is significantly effected by categorical explanatory predictors.

There are three different statistical treatments that may explain tangent delta: the *Model* type (Complex Model or Experimental data), the *Surgical Treatment* (Control or OVX surgery) and the *Test Frequency* (1, 3, 9, or 15 Hz). Each one of these predictors may have an impact on the response variable (tangent delta) individually, or combined as statistical interactions. A statistical interaction means that the effect of at least two treatments together are dependent and non-additive. For example, the combined effect of the Model Type and a specific Test Frequency may have a significant impact on tangent delta separate than the individual combined effects of those two treatments . A summary table for these treatments may be viewed in table 5.

Because the Complex Model was created to theoretically express the variability in experimental D-spacings, input values for the Complex Model exist as averages and standard deviations of all the AFM measurements across the test sheep (three control, three OVX). Additionally, two cranial bone samples from each sheep were collected and averaged. These bone samples were tested at four frequencies, creating a correlation between a specific bone sample and its output at each frequency.

Thus, there is an established correlation between experimental units: the sheep specimen influences the bone sample which influences the tangent delta output at each test frequency. The relationship between these units creates a complicated hierarchical experimental structure. Additionally, test frequencies are not equally staggered: the tangent delta data at 1 Hz are likely more correlated with 3 Hz than data at 3 Hz are correlated with 9 Hz. Due to this unequal spacing, the test frequencies are not equally independent.

To adequately address these statistical complications, a repeated measures analysis of variance (abbreviated to "repeated measures ANOVA") was performed. The repeated measures ANOVA was performed without the assumption that measurements of the same bone sample at different test frequencies are independent of each other.

Repeated measures ANOVA was performed through the utilization of Statistical Analysis Software (SAS). This work was accomplished with the assistance of Professor Johnathan Walker of the Statistics Department at California Polytechnic State University, San Luis Obispo. Once the repeated measures analysis was completed, a pairwise comparison may be implemented to compare significant statistical treatments. A Tukey-Kramer pairwise comparison was selected in an effort to limit Type I error. Because this current study is interested in investigating serious metabolic disorders, limiting false positives by controlling Type I error is critical to the progression of this research.

For this reason, a 1% individual significance level was selected to test each statistical treatment. Because each treatment must also be tested with its interactions, there are eight total tests. Through using a conservative individual significance level, the overall risk of a Type I error across all tests is only 8%.

3 Results

Tangent delta data were collected using each version of the Complex Model. Tangent delta was calculated as the phase shift between the loading and deformation curves. Data were then compared for the purpose of computational model validation to experimental findings conducted via dynamic mechanical analyzer (DMA) on six Rambouillet-cross ewes provided by Henry Ford Hospital. Data were collected at 1, 3, 9, and 15 Hz.

Experimental data were organized based on specimen, treatment, anatomical location, and test frequency. A full list of all experimental data collected may be viewed in appendix A. Mean tangent delta data from DMA testing may be seen in table 6 and table 7. Because this study is interested in comparing the effects of estrogen depletion on viscoelasticity, treatment and test frequency data were utilized for computational model creation. Additionally, all experimental findings utilized for model validation came from the cranial region. This region was selected as this sector experiences tensile forces. The Siegmund Model, which serves as the FEA basis for this current study, displays collagen molecules in uniaxial tensile loading. In an effort to mimic this theoretical setup with experimental data, the cranial region was selected. This also eliminates any sample variability between anatomical sectors.

Table 6: Summary Table for Control, Cranial DMA Testing on Six Rambouilet-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
Experimental Data	AVE	0.07656	0.06437	0.03947	0.01876
Experimental Data	SD	0.00883	0.00561	0.00548	0.00751

Table 7: Summary Table for OVX, Cranial DMA Testing on Six Rambouilet-cross Ewes. Data were provided by Henry Ford Hospital

Model (OVX, Cran)	Average/ St. Deviation	Tangent Delta			
		1 Hz	3 Hz	9 Hz	15 Hz
Experimental Data	AVE	0.07302	0.06217	0.03886	0.01663
Experimental Data	SD	0.03390	0.00396	0.00592	0.00655

The computational model used for this study, the Complex Model of 2 x 100 half unit cells, requires validation to these experimental results. Tangent delta values were obtained computationally (theoretically) by examining the individual displacement of all nodes at the free end of the Complex Model. After having been subjected to the sinusoidal loading condition, displacements were calculated at over 30 individual nodes. These points were retrieved utilizing a Python script. These nodal displacements were then averaged and converted to a strain. This strain was then graphed on a sinusoidal function through the implementation of a curve fitting function in MATLAB.

A curve fitting program was utilized to retrieve tangent delta data. This program calculated tangent delta by measuring the phase shift between stress and strain curves. This MATLAB output may be viewed in figure 27. This program works in tandem with a separate set of code that calculates how well the displacement output from a single model matches the sinusoidal function. This output may be seen in figure 28.

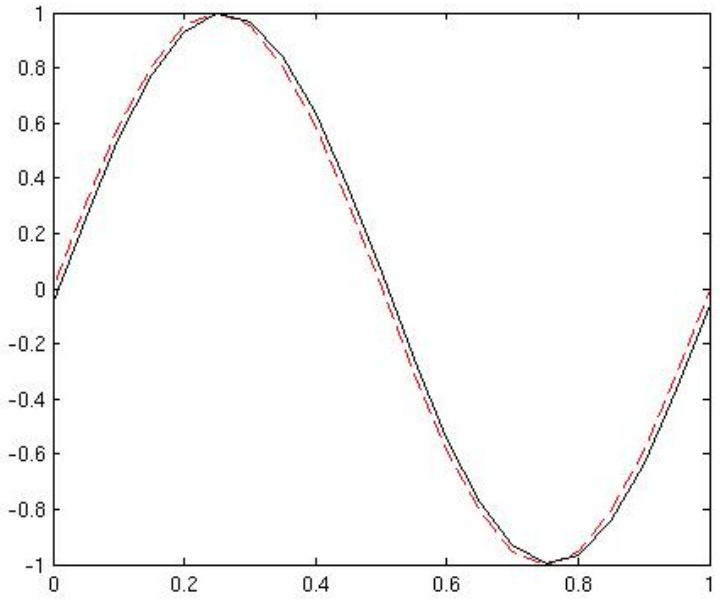

Figure 27: Tangent delta is calculated from the phase shift between the loading cycle (solid black line) and deformation cycle (dashed red line). The x-axis is time (seconds) and the y-axis represents normalized stress/strain.

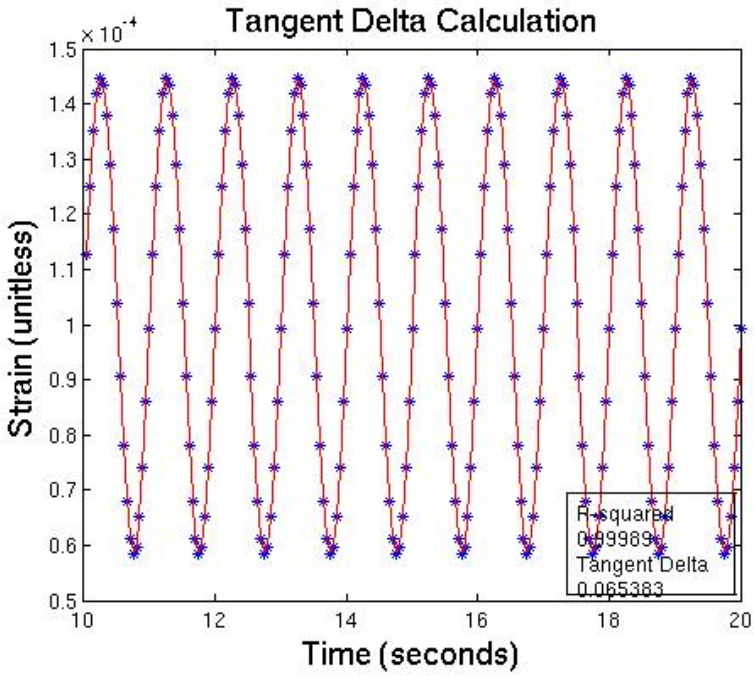

Figure 28: An R^2 of at least 0.99 is confirmed for each model to ensure an accurate tangent delta. This R^2 is reflective of the ability of the MATLAB program to curve fit the output data to a sinusoidal function.

In order to represent this strain rate dependent experimental data with a computational model, an appropriate rheological model had to be established. Findings by Mendoza suggested that a 1.25 GPa*s (η_1) dashpot was best suited for matching similar experimental DMA results from an ovine model [52]. To examine how the dashpot choice affected the accuracy of the computational tangent delta, both a 4 GPa*s and 1.25 GPa*s dashpot were tested and compared across all test frequencies. These results may be viewed graphically in figure 29.

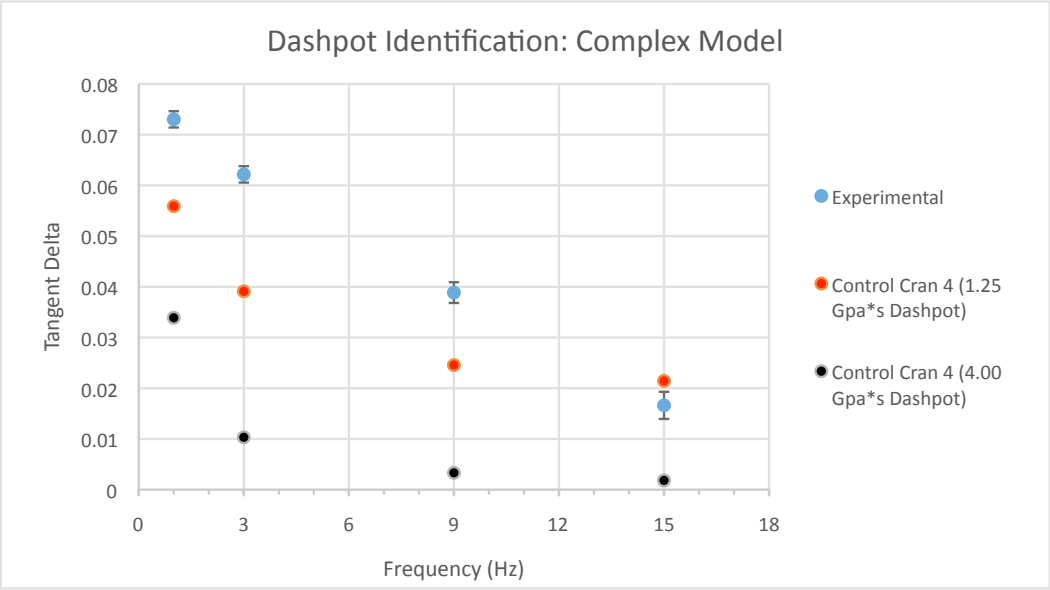

Figure 29: In order to confirm the 1.25 GPa*s dashpot, tangent delta data were graphed utilizing both the 1.25 GPa*s and 4.00 GPa*s dashpot separately. The dashpot with a viscosity of 1.25 GPa*s produced values noticeably closer to experimental findings.

Though the 4 GPa*s dashpot performed adequately at high test frequencies, its selection poorly matched low strain rate experimentation. This confirmed the findings of the Mendoza Model. For this reason, a 1.25 GPa*s was selected to collect all model results. Full tables of all results may be viewed in appendix E.

Four variants of both models (Control and OVX) were randomly generated based on a Gaussian distribution. This randomization utilized the means and standard deviations found via experimental atomic force microscopy on ewe bone samples (provided by the University of Michigan, Ann Arbor). The four Control, Cranial models and four OVX, Cranial models were run individually at each test frequency. Output tangent delta data were recorded from

each test. As seen in figure 30, each model expressed slightly different tangent delta data. The trends between models are very similar, with some points eclipsing results of other variant models.

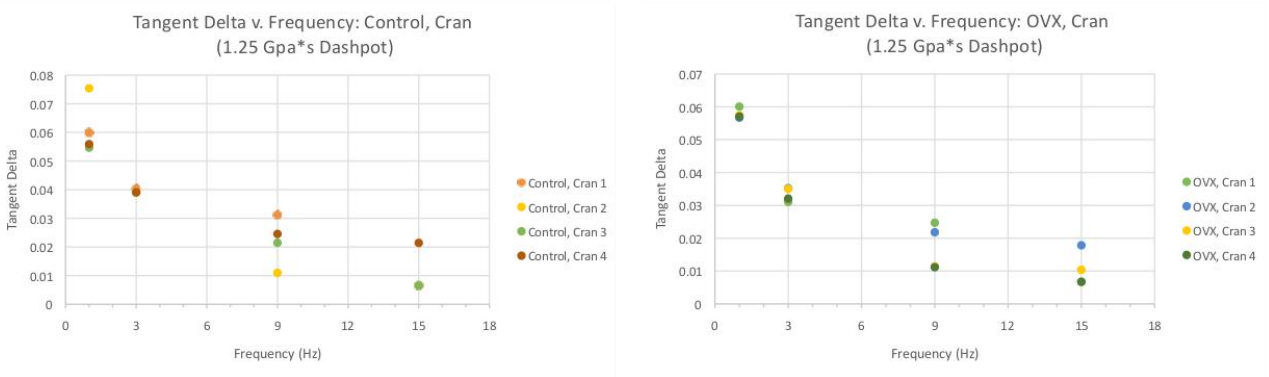

Figure 30: Four variants of both core models (Control, Cranial and OVX, Cranial) were run at all test frequencies. These results were averaged to create figures 31 and 33. Points of overlap may be more easily identified in appendix E. Data utilized a 1.25 GPa*s dashpot.

Tangent delta data from the four variants were averaged to obtain composite data for each Complex Model. The small spread of data discovered with the model variants was utilized to create standard error bars at each of the four test frequencies. Figure 31 displays tangent delta results across three different tests: experimental results, computational results utilizing the Complex Model, and computational results reported from the Mendoza Model [52]. Comparison to the Mendoza Model allows this current study to determine if the Complex Model produces different output data through the inclusion of many randomized collagen-hydroxyapatite complexes as opposed to modeling a single half unit cell.

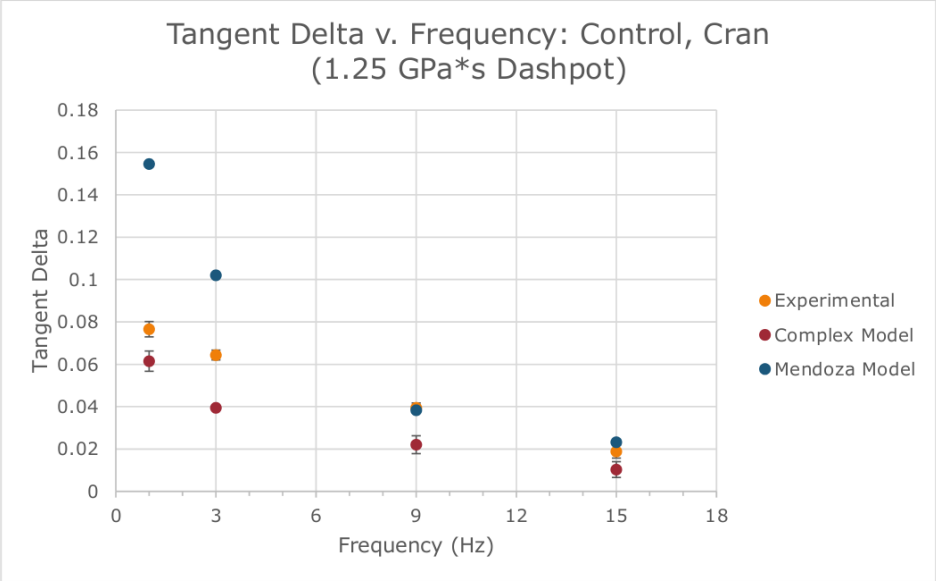

Figure 31: Averaged points from each variant of Control, Cranial were utilized to generate this composite data. Standard error bars were applied to each test frequency. The Mendoza Model utilized for comparison is Mendoza's "Normal D-spacing" finite element model. Data utilized a 1.25 GPa*s dashpot.

Mendoza modeled a single half unit cell at a set periodicity [52]. In his study, Mendoza changed this set periodicity between 64, 67, and 70 nm and tested each half unit individually. His "Normal D-spacing" model was selected for comparison to the Complex Model results. This selection was based on Mendoza's preference for this model across his computational test conditions, such as using this model both for his final results and dashpot identification. Additionally, the Complex Model's dimensions (i.e. mineral thickness, volumetric ratio of mineralization, etc.) were scaled proportionately to Mendoza's Normal D-spacing model, which echoes the original model created by Siegmund et al. [78]. Standard error bars for the experimental results were collected from DMA testing across six separate readings. The estimate for standard error, SE, may be calculated as

$$SE = \frac{s}{\sqrt{n}}$$

where s is the sample standard deviation (SD) and n is the number of samples. A summary table may be viewed in table 8.

Table 8: Summary Table for Control, Cranial Testing. Experimental data were collected via DMA testing on six Rambouillet-Cross Ewes. The reported Complex Model data were obtained through averaging the tangent delta of the four model variants. Reported data from Mendoza's Normal D-spacing model were used for comparison [52]. Because Mendoza did not implement randomization or report variations, his model does not have a standard deviation. Data utilized a 1.25 GPa*s dashpot.

Model (Control, Cran)	Average/ St. Deviation	Tangent Delta			
		1 Hz	3 Hz	9 Hz	15 Hz
Experimental Data	AVE	0.07656	0.0644	0.03947	0.01876
Experimental Data	SD	0.00883	0.00561	0.00548	0.00751
Complex Model	AVE	0.06148	0.03941	0.02206	0.01294
Complex Model	SD	0.00954	0.00061	0.00843	0.00743
Mendoza Model	AVE	0.15459	0.10201	0.03829	0.02318

From visual inspection, the Complex Model's inclusion of 200 half unit cells produces very different results when compared to the modeling of a single periodic half unit in the Mendoza Model. These differences are especially apparent at low frequencies, where the Complex Model appears to be a much stronger fit to experimental findings. Because Mendoza only reported a single tangent delta value at each test frequency in his Normal D-spacing model with constant dimensions for the half unit cell, statistical analysis cannot be performed. However, comparisons can still be appropriately drawn.

Standard deviations were computed through examining the average tangent delta output of each model variant. Comparison between the Complex Model and Mendoza Model suggests that the data varies greatly between each FEA approach. At low frequencies (1 and 3 Hz), the values reported by Mendoza utilizing a single half unit cell fall over three standard deviations outside the average tangent delta of the Complex Model. At the higher frequencies (9 and 15 Hz), Mendoza values still land outside a single standard deviation of the Complex Model's results.

A R^2 value was obtained through plotting computational results against experimental findings (figure 32). A linear line of best fit was applied to this figure. Because model data was collected across four replicates, R^2 values were calculated utilizing the four variants individually as well as a R^2 value utilized averaged tangent deltas at each test frequency. The R^2 value utilizing the points individually was found as 0.883.

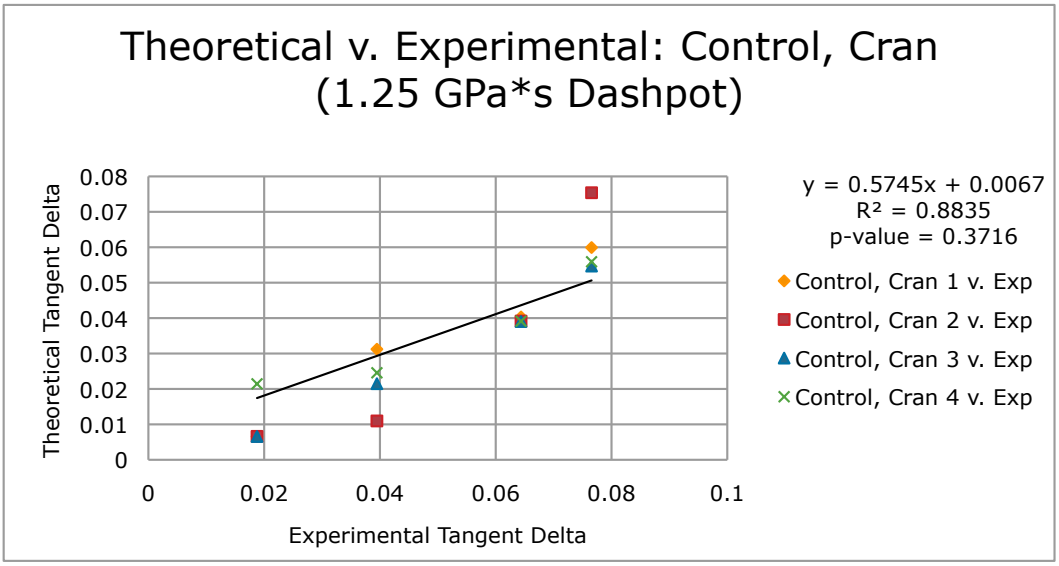

Figure 32: A linear line was fit to the theoretical v. experimental results. The R^2 value of 0.883 shows a high correlation between experimental and each individual computational finding. These data were collected utilizing a 1.25 GPa*s dashpot.

Through averaging, an R^2 value of 0.942 (seen in appendix E) was obtained. Averaging provides an appropriate means of comparison to the Mendoza Model, as that study only reported a single tangent delta at each test frequency. These high correlations found using the Complex Model signify that the variance observed in the computational model is largely explained by the experimental inputs. Both of these R^2 values were higher than that of Mendoza's reported value of 0.874. Though averaging may mask some sample variance, it is a worth performing as a means of comparison against Mendoza's non-variable results. These data were collected utilizing a 1.25 GPa*s dashpot.

A root mean squared error (RMSE) value may also be calculated using the values in table 8. As was the case with R^2, calculating the RMSE provides an extra means of comparison between the Complex and Mendoza models where statistical options are limited. When each model is compared to the experimental data, the Complex Model produces a RMSE of 0.017, while the error is higher in the Mendoza Model with a RMSE of 0.043.

Data were collected on the OVX, Cranial models in the same fashion as Control, Cranial. Each of the four randomized variants of OVX, Cranial were tested separately at each test frequency. This dataset was then collected and averaged to create a composite OVX, Cranial set of data. The trends of this data set may be viewed in figure 33 as "Complex

Table 9: Summary Table for OVX, Cranial Testing. Experimental data were collected via DMA testing on six Rambouillet-Cross Ewes. The reported Complex Model data were obtained through averaging the tangent delta of the four model variants. Reported data from Mendoza's Normal D-spacing model used for comparison [52]. Because Mendoza did not implement randomization or report variations, his model does not have a standard deviation. Data utilized a 1.25 GPa*s dashpot.

Model (OVX, Cran)	Average/ St. Deviation	Tangent Delta			
		1 Hz	3 Hz	9 Hz	15 Hz
Experimental Data	AVE	0.07302	0.06217	0.03886	0.01663
Experimental Data	SD	0.03390	0.00396	0.00592	0.00655
Complex Model	AVE	0.05782	0.03336	0.01729	0.01043
Complex Model	SD	0.00153	0.00210	0.00702	0.00524
Mendoza Model	AVE	0.15459	0.10201	0.03829	0.02318

Model." A summary table may be viewed for this data in table 9.

The OVX, Cranial data collected from the Complex Model exhibit a similar trend to both the experimental findings and the results collected for the Control, Cranial Complex Model. As was the case with the Control, Cranial tests, these data points seem to be a bit conservative, consistently lightly underestimating experimental findings. Because the research of Miguel Mendoza did not create a model exclusively to simulate D-spacing from sheep experiencing estrogen depletion, his "Normal D-spacing"' was again chosen for comparison.

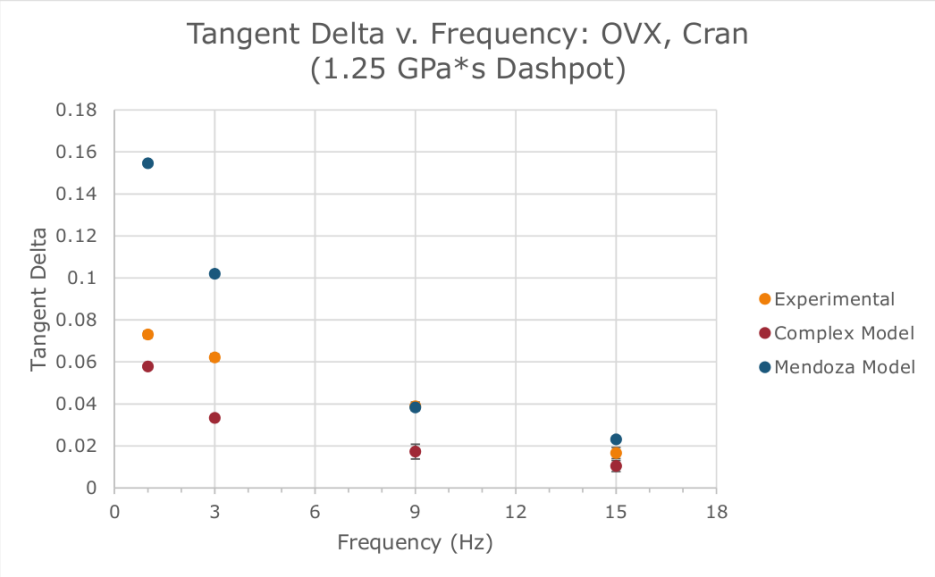

Figure 33: Averaged points from each variant of OVX, Cranial were utilized to generate this composite data. Standard error bars were applied to each test frequency, though they are better viewed numerically in table 9. Mendoza's "Normal D-spacing" model was selected for comparison, as Mendoza's study did not create a model explicitly based on experimental D-spacing recorded from OVX sheep. Data utilized a 1.25 GPa*s dashpot.

Once data were collected, an R^2 value was again calculated to evaluate the correlation between experimental and theoretical data. This may be viewed in figure 34. R^2 values were collected in the same fashion as Control, Cranial; a R^2 value was calculated by plotting all four variants individually versus experimental data, and a R^2 value was found from averaging the computational models for OVX, Cranial (which may be viewed in appendix E).

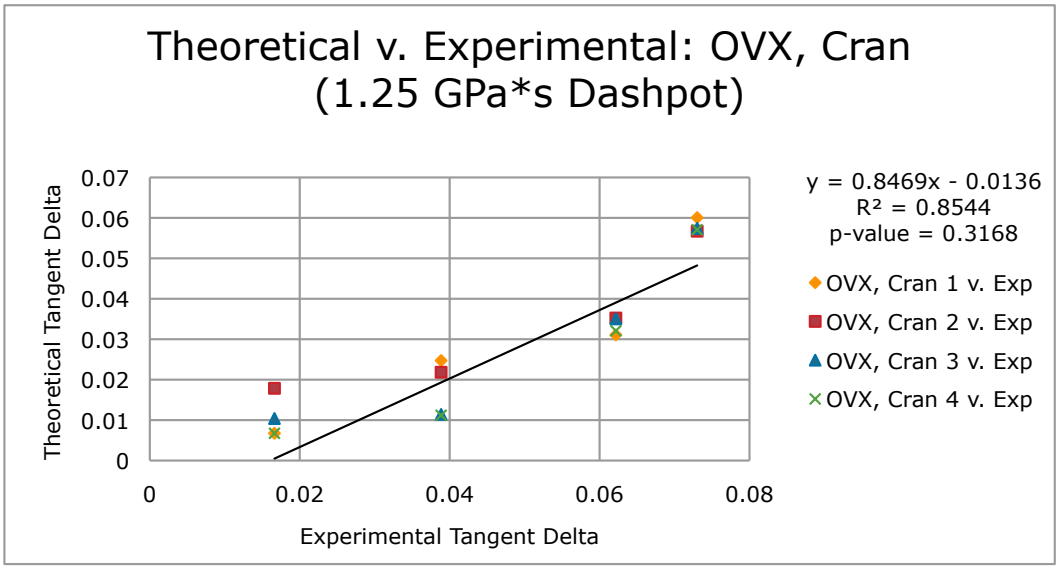

Figure 34: An R^2 value of 0.854 was obtained through plotting individual theoretical OVX, Cranial data against the Experimental data. Because previous FEA research has not produced a computational model based explicitly on estrogen depletion's effect on D-spacing, there is no R^2 value applicable for comparison.

Because extensive randomization parameters were included in Complex Model generation, OVX, Cranial data is well-suited for statistical comparison against the Control, Cranial data. Though both figures 32 and 34 display p-values from a simple two-sample t-test, a more intensive statistical investigation is possible. Given the number of statistical treatments utilized within this study (i.e. Model Type, Surgical Treatment, and Test Frequency), in-depth comparisons may be drawn to examine the implications of all the explanatory variables.

A repeated measures ANOVA analysis was performed with SAS. This analysis examined three core statistical treatments–Model Type (Mod), Surgical Treatment (Trt), and Test Frequency (Frq)–as well as the interactions between each. The critical output is collected in table 10, while the full SAS output may be viewed in appendix F.

Table 10: Repeated Measure ANOVA Output. A 1% individual significance level was utilized to evaluate the effects of multiple statistical treatments.

Repeated Measures ANOVA				
Effect	**Num DF**	**Den DF**	**F Value**	**Pr > F**
Mod	1	4	45.54	0.0025
Trt	1	4	1.25	0.3258
Mod*Trt	1	4	0.08	0.7884
Freq	3	48	290.80	<.0001
Mod*Freq	3	48	16.17	<.0001
Trt*Freq	3	48	0.24	0.8674
Mod*Trt*Freq	3	48	0.56	0.6439

This statistical output may be utilized to draw a number of meaningful conclusions about the Experimental and Complex Models. Because there are no significant interactions involving Surgical Treatment, it is appropriate to examine the statistical main effect of Control v. OVX sheep. With a p-value of 0.3258, the following conclusion may be drawn: after adjusting for all other predictors, at the 1% individual significance level, there is not enough evidence to convince this study that the surgical treatment has an effect on the mean tangent delta for all experimental and theoretical data.

Table 10 does reveal a significant interaction between Model Type and Test Frequency with a p-value less than 0.0001. Based on this output, this study may conclude: after adjusting for all other predictors, at the 1% individual significance level, this study is convinced that the effect of Model Type depends on tangent delta depends on Test Frequency.

Because the interaction between Model Type and Test Frequency is significant, the main effects of the Model Type and Test Frequency may not be interpreted individually. This significant interaction warrants a comparison between each Model at each Test Frequency to determine at which frequencies the Complex Model and experimental results significantly differ. Table 11 illustrates this comparison.

Table 11: Tukey-Kramer Pairwise Comparison of Statistical Interactions.

Tests of Effect Slices						
Effect	**Mod**	**Freq**	**Num DF**	**Den DF**	**F Value**	**Pr > F**
Mod*Freq		1	1	48	21.81	<.0001
Mod*Freq		3	1	48	68.79	<.0001
Mod*Freq		9	1	48	36.16	<.0001
Mod*Freq		15	1	48	5.11	0.0283
Mod*Trt	Complex		1	4	0.75	0.4364
Mod*Trt	Experimental		1	4	0.51	0.5136

The two models for statistical analysis (Complex and Experimental) are significantly different at 1, 3, and 9 Hz. The two models are not significantly different at 15 Hz, however. Given this correlation, it cannot be proven that the two models are statistically different from one another.

A pairwise comparison grants further insight on how each combination of Model Type and Test Frequency compare to one another. The results for this pairwise comparison may be seen in figure 35. Each interaction between Model Type and Test Frequency is given an alphabetical label. Points that share a common letter are not significantly different from one another. For example, the Complex Model at 1 Hz produces a significantly different tangent delta than the Experimental Model at 1 Hz, but the Complex Model at 1 Hz does not have enough evidence to prove it is statically different from the Experimental Model at 3 Hz.

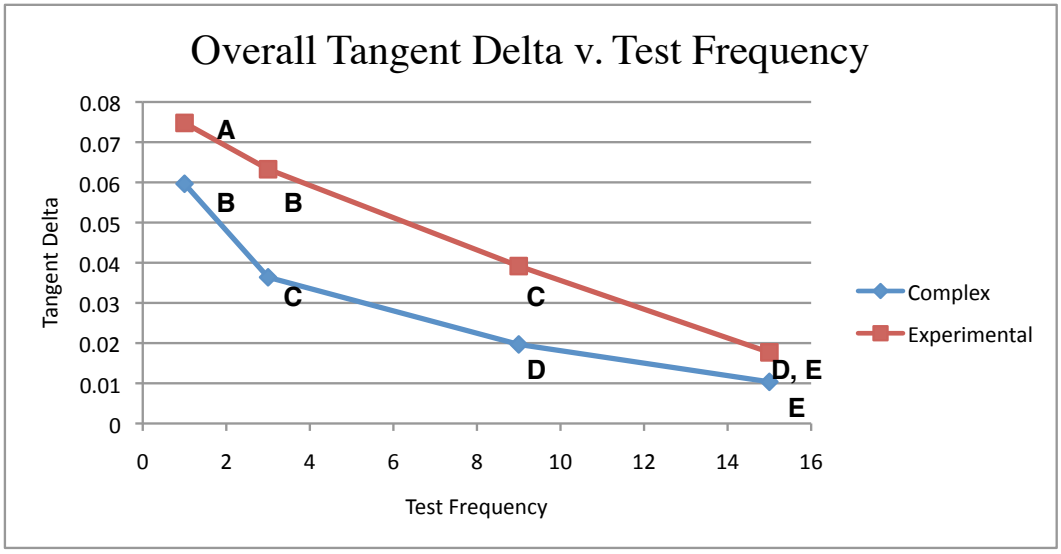

Figure 35: An interaction plot is created from comparing the overall tangent delta output from all eight variants of the Complex Model to Experimental tangent delta across the 12 bone samples. All Complex Model data and all Experimental Model data may be utilized in this comparison as the effect of the Surgical Treatment was not proved to be significant. The significant interaction between Model Type and Test Frequency is on display. Data points with like markers are not significantly different from one another.

All models were run with a single warning message on file, which may be viewed in appendix D. Each model was symptom to a small number of distorted elements (approximately 50 for each model). A distorted element is described in Abaqus as an element with an angle less than 45 degrees or greater than 135 degrees. These elements were typically found along the collagen-hydroxyapatite interface, which is unsurprising given the sharp geometric change between the two materials. Because each model possesses roughly 219000 elements, the effect of a few distorted elements is likely negligible.

4 Discussion

Osteoporosis affects over 200 million people worldwide, making it the most common type of bone disorder. The disorder is so prevalent that more than half the female population over the age of fifty will experience it as a direct result of menopause [37]. Though menopause's characteristic drop in estrogen levels holds a well-established link to a loss in bone density [2], the mechanisms for this change represent a great mystery in the field of biomechanics research. Conflicting theories look to point the blame to differing biological systems that control bone tissue's unique adaption properties. Osteoclasts (which remove bone) may be overactive, or, their counterpart, osteoblasts (which deposit bone) may become defective in the event of massive estrogen loss. Because the exact mechanisms of this disease are largely unknown, long-term treatment is imperfect at best [43].

Bone itself presents many mechanical unknowns, as its response to loading holds its own set of complications. Though bone may be simply considered a two-phase material consisting of collagen and mineral (hydroxyapatite), bone tissue is a complex biomaterial whose structure is the culmination of multiple bone cells performing highly specialized roles. Bone's deformation response to loading is non-linear, as strain rate dramatically affects the stiffness of the material. Similar to the case of osteoporosis research efforts, the root cause of this bone behavior is unknown. Researchers speculate that the biphasic nature of bone may hold a piece to the viscoelastic puzzle [7], whereas other studies point to the presence of cement line structures in Haversian bone [9].

Whatever the root cause of cause of both bone viscoelasticity and debilitating bone disorders such as osteoporosis, one thing is clear: the field of biomechanics is complex and demands extensive new research. The parameter known as D-spacing describes the nanometer distance between staggered collagen molecules, and research into this niche topic is still in its infantile stages [4, 21]. Given bone's biphasic composition, the ratio of the tissue's two main components may hold massive implications for the overall mechanical properties of the material. The staggering between collagen molecules controls the amount of mineral (associated with bone's characteristic stiffness) that may be filled within a collagen fibril. This spacing is doubly critical, as it also determines the potential packing factor of collagen.

The packing factor is reflective of the density of collagen molecules that may fit in a single region, where a dense population of tropocollagen then creates a strong, rope-like network [5].

Changes to this dimension on the nanometer scale may hold the answers to explaining complicated relationships such as osteoporosis and bone viscoelasticity. In an effort to help conceptualize the fundamentals of D-spacing, Hodge and Petruska created the "staggered array" model in the 1960s [19]. By recognizing that the length of tropocollagen (about 280 nm) was not an integer multiple of this spacing between collagen molecules (originally thought to be a constant 67 nm) Hodge and Petruska were able to devise a simple visual that accounted for the regions of collagen gaps and overlaps that they observed experimentally. This two-dimensional model aligned multiple collagen molecules in rows, each separated from their neighbor by a small junction that would ultimately become mineralized [7]. This staggered array model was adapted by Jager and Fratzl in 2000 [20] before being translated into applicable computational modeling by Siegmund et al. [78].

Finite element software, Abaqus, provides an excellent tool for examining the mechanics of a structure. In order to employ this tool, however, a non-theoretical model must validate the findings of the computer simulation. Sheep present a great way of obtaining experimental data for a number of reasons. Though bone formation in the animal kingdom is achieved through slightly different physiological mechanisms, adult female sheep provide a number of solutions to these limitations. In addition to being docile, easily managed, and cost efficient, mature sheep possess a bone structure not unlike Haversian bone found in humans [56]. Though sheep do not naturally undergo menopause, an ovariectomy may be performed to simulate the estrogen depletion experienced by middle aged women [41].

Experimental data provided by the Colorado State University, Henry Ford Hospital, and the University of Michigan, Ann Arbor were used to create computational models to better examine the repercussions of variant D-spacing. Rambouillet-cross ewes were selected based on their availability, and their bone samples were tested for the purpose of model validation at 1, 3, 9, and 15 Hz. The viscoelastic changes experienced by bone were measured through examination of the tangent delta.

Tangent delta represents a damping characteristic and may be expressed as:

$$tan\ \delta = \frac{E_{loss}}{E_{storage}}$$

where E_{loss} is the imaginary loss modulus (representative of a material's ability to dissipative energy) and $E_{storage}$ is the storage modulus (approximately the material's Young's modulus in non-oscillatory conditions and is thus representative of stiffness) [77]. Through application of an oscillating stress to a viscoelastic material and knowledge of a materials' response to linear loading, tangent delta may be simply calculated.

Tangent delta is often called a loss tangent as it represents a material's ability to dampen incoming loading. A perfectly elastic material would possess a tangent delta of zero, as all incoming energy could be stored/returned. A material with tangent delta above zero (such as bone [61, 75]) indicates some conversion of energy to another form (ex. heat). Toughness may be defined as a material's ability to absorb and store energy. Low tangent delta values would indicate a decrease in bone's ability to mitigate incoming energy, and would represent a loss in bone's characteristic toughness [8, 74]. Tangent delta is directly related to the phase shift between a material's stress input and strain response [73]. In order to obtain this value in computational modeling, a structure must exhibit time-dependent material properties. The connection between bone viscoelasticity and its toughness and strength is well established in literature [12, 63, 66, 76].

A UMAT was created to model viscoelastic behavior in Abaqus by Dr. Frank Richter [83]. This user subroutine was adapted by Miguel Mendoza [52]. These viscoelastic parameters were generated to mimic the time-dependent responses of rheological elements in the arrangement of the Kelvin-Voigt form of the Standard Linear Solid. Because this rheological formation is capable of expressing both creep and stress relaxation (both established mechanical responses of bone tissue), it was implemented into the finite element model. To utilize this UMAT, applicable elastic moduli had to be applied to the spring and dashpots in said rheological model. An effective modulus of 2 GPa was ultimately selected for the spring elements, and a dashpot of 1.25 GPa*s was utilized to represent the viscous behavior of bone. Hydroxyapatite was modeled as a simply elastic material.

Because D-spacing is not constant and its variant patterning exists for great lengths

approaching 40 μm, a complicated finite element model was created. This extensive 2-D model built off the aforementioned approaches by researchers in the field [78, 52, 20]. This "Complex Model" possesses a geometry of 2 x 100 half unit cells based on the FEA modeling by Siegmund et al. in 2008 [78]. Each half unit possesses random (Gaussian distribution) periodicity based on experimental findings on test sheep. Four variants were made of each model. Mismatched geometry concerns were addressed by the addition of a spacer that ensured a rectangular model. This FEA model was made to answer two core questions: does modeling a complex arrangement of many collagen molecules impact theoretical tangent delta output, and does bone viscoelasticity significantly change in the case of estrogen depletion as it pertains to the D-spacing of ovariectomized sheep (OVX)?

To evaluate the impact of utilizing a Complex D-spacing model, results are appropriately compared to the Mendoza Model, which serves as the computational forbearer to this current study [52]. A significant limitation of comparison to the Mendoza Model is the fact that Mendoza did not employ randomized elements in his model generation. Additionally, Mendoza reported a single tangent delta value at each test frequency, and each model was created with a set periodicity. The lack of randomization makes any statistical comparison invalid. Because Mendoza's research was not aimed at modeling variability, this limitation is completely justified, but does require this current study to utilize other forms of non-statistical comparisons. Thankfully, the results of his model and that of the Complex Model differ by enough to make effective qualitative comparisons.

Preliminary results from this analysis strongly suggest a large difference in tangent delta is obtained when modeling the relationship of many collagen-hydroxyapatite units when compared to expressing only a single half unit cell. The decision to utilize the same dashpot advocated by Mendoza's Model (as well as the decision to isolate a common anatomical sector, cranial) helps to make these comparisons clear. As seen in table 8, values examining Control, Cranial sheep appear to be very different between the Mendoza and Complex Model. At the low test frequencies of 1 and 3 Hz (frequencies the Mendoza Model reportedly had difficulty accurately computationally expressing), the reported values for the Mendoza Model fall well outside three standard deviations from those of the Complex Model. The Mendoza Model over-approximated tangent delta at low test frequencies,

while the Complex Model more consistently estimated these tangent deltas just underneath experimental values, producing stronger R^2 and RMSE values.

Control, Cranial comparisons between the Mendoza and Complex models show a greater similarity at high test frequencies. Mendoza specifically chose to model his viscoelastic material properties in an effort to match experimental high test frequencies. For this reason, it is unsurprising that his values were closer to the Complex Model tests of high frequency loading. That said, the Mendoza Model still expresses values that fall outside of one standard deviation from the Complex Model at test frequencies of 9 and 15 Hz, reiterating the difference that exists when modeling many half unit cells.

The general trends expressed by the Complex Model and experimental results corroborate previous literature findings. Because bone provides essential structural support and protection for the organs of the human body, it must be able to stiffen quickly in the event of traumatic impact [62]. A high stiffness value would result in a low tangent delta (i.e. a variable loss modulus to high storage modulus) [91]. Though oscillatory energy dissipation and toughness are two different topics, correlations have been drawn in literature between a material's ability to dissipate energy without fail and the energy at a crack tip in fracture mechanics [8]. Observing low values of tangent delta at high frequencies illustrates this viscoelastic function of the bone tissue, and helps to further establish confidence in this current study's computational results.

Mendoza elected to calculate an R^2 value as a means of comparing his computational findings to experimental results. R^2 statistically expresses the amount of variability in the response variable explained by a predictor. Though Mendoza's models were focused on simulating viscoelasticity and less concerned with randomized parameters, R^2 still presents an extra means for basic comparison.

There are two ways to produce an R^2 value that is appropriate for comparison to Mendoza's Control, Cranial results. Because he only reported a single tangent delta value at each test frequency, this setup may be replicated for this current study for comparison. By averaging the Complex Model's Control, Cranial output at each test frequency, a single value at each frequency may be fitted to the experimental findings. In doing this, the Complex Model excels with a R^2 value of 0.942 to Mendoza's 0.874. Averaging data points does

also "average-out" some of the variability. For this reason, a high R^2 value is not completely surprising. Because Mendoza reported only a single value, however, this transformation to the data seems very appropriate.

If the Complex Model's data points are not averaged, the R^2 value for the Complex Model shifts slightly. By fitting a linear line through each Control, Cranial variant, a R^2 value of 0.883 is obtained. A comparison of RMSE values for the Control, Cranial data produced a value of 0.017 for the Complex Model, while the Mendoza Model produced a higher error value of 0.043. This value is still an improvement over previous correlations. The fact that a higher R^2 value was discovered and RMSE was lowered only furthers this study's confidence that finite element modeling a complex relationship of collagen molecules is critical to relevant results.

This trend observed in these Control, Cranial comparisons is very promising in developing a narrative for the importance of more holistically modeling a collagen fibril. The tangent delta output is simply a reflection of bone viscoelasticity; bone's mechanical properties are well-documented to correlate with the presence of a dense type I collagen population [2, 5]. The network of linking collagen molecules creates a material framework that makes bone mechanically robust. Because this current study utilized perfect contact relationships between half cell units (as deemed important by Siegmund et al. [78]), the interactivity between collagen molecules is reflective within this complex computational model. The observed trend of the Complex Model's consistently conservative tangent delta output seems to confirm this theory; the presence of many neighboring collagen-hydroxyapatite complexes appropriately exhibit collagen's characteristic linking and subsequently decrease the ratio of loss modulus to storage modulus through a plausible increase in stiffness.

This study is also confident in the biologic relevance of the obtained data. The spacer code was implemented as a means of ensuring an appropriate biologic ratio of collagen to hydroxyapatite. This addition to the Python script also ensures that loads are applied across a surface free of notches and potential stress concentrations [85]. No model utilized in this study employed the spacer code's secondary safeguard of adding an extra half unit cell to complete the rectangular geometry, but the presence of this backup makes the Complex Model flexible for future studies. Though the models used in this study did not end up

utilizing this feature, the code still checked the material composition of the final half units to ensure the 70% ratio of collagen to hydroxyapatite was not altered by more than 5%. This tight biologic constraint ensured that any randomly generated model maintained a relevant biologic makeup [78].

The dimensions of each randomized half unit cell is highly encouraging from a biologic standpoint as well. Because the standard deviation employed in the Gaussian distribution (selected due to D-spacing research precedent [41]) was input at approximately 1 nm, the within bundle periodicity variation recently reported is confirmed [3]. Furthermore, because D-spacing dimensions have been observed to persist for 40 μm at a time [3], the 100 half unit cells in series at roughly 67 +/- 1 nm apiece fall comfortably within this biologic range.

Though statistical analysis cannot be performed due to a lack of previously reported variable data, these findings very appropriately suggest that the inclusion of the complex collagen arrangement employed in this current study yields very different viscoelastic results. These data, coupled with previous literature findings on the important role of a network of collagen molecules in dictating bone's mechanical properties, strongly supports the usage of the Complex Model as the basis for future D-spacing research.

Though comparison to Mendoza's model is statistically limited, the extensive randomization employed in the generation of the Complex Model permits statistical analysis to be drawn within versions of this randomized computational framework. A core purpose of this current study was to determine if tangent delta values changed significantly between OVX and control D-spacing models. Because four randomly generated models were created based on both Control, Cranial and OVX, Cranial experimental data, these eight models provide an excellent opportunity to draw meaningful comparisons between healthy subjects and subjects experiencing simulated menopause.

A repeated measures ANOVA was performed on the data collected form the Complex and Experimental Models. This style of analysis was necessitated based on the complexity of the experimental units utilized in this analysis. Because the sheep specimen influences the bone samples which then influence the tangent delta retrieved at frequency tests, a repeated measures ANOVA provides a means to parse through this hierarchical experimental structure. Additionally, the nonuniform staggering of test frequencies means the effects of

test frequency are not equally independent and statistical analysis must be more sophisticated.

Statistical comparisons were accomplished through use of Statistical Analysis Software (SAS), with assistance from Professor John Walker of California Polytechnic State University's Statistics Department. Three core statistical treatments were compared: Model Type (Complex or Experimental), Surgical Treatment (Control or OVX) and Test Frequency (1, 3, 9, or 15 Hz). Interactions were also tested to see if the effect of a treatment was dependent on the presence of another treatment in a non-additive manner.

Statistical analysis was able to prove a significant interaction between Model Type and Test Frequency. The combination of both of these factors has a significant, non-additive impact on mean tangent delta. The Complex Model is significantly different from experimental results at 1, 3, and 9 Hz, though there is not enough evidence to suggest that the models are significantly different at the highest test frequency, 15 Hz. SAS analysis confirmed that tangent delta varies with the model at each frequency. This confirms that the Complex Model was successful at expressing meaningfully variable viscoelasticity at each test frequency. Furthermore, as the statistical analysis revealed a significant interaction between Model Type and Test Frequency, it is not possible to claim that the theoretical model is significantly different than experimental findings, which is a critical step towards model verification.

SAS output revealed that at the 1% significance level, after adjusting for all other variables, there is not enough evidence to convince this study that the Surgical Treatment alone has a significant impact on output tangent delta. This result was initially surprising as previous literature has reported a correlation between OVX sheep and viscoelasticity [41]. Similarly, D-spacing has recently been shown to be significantly associated with estrogen loss [92].

The finding by this current study that OVX's link to viscoelasticity is not significant may be adequately explained from a different angle, however. The lack of significance in OVX surgical treatment implies either that the Complex Model is not a strong predictor of experimental values, or the effect of OVX on D-spacing lies within a change to the collagen viscoelasticity. Because the Complex Model has been shown statistically to not produce

results that are significantly different from Experimental data (particularly at 15 Hz), it is unlikely that the model is an entirely inadequate predictor.

What is more likely is that OVX has a significant impact on viscoelasticity based solely on an expression of collagen-hydroxyapatite periods. The lack of significance and the strong prediction capability of the Complex Model gives this study confidence that OVX is fully accounted for in the Complex Model by including its D-spacing. This leads the current study to believe that OVX is linked to viscoelasticity as it has a significant effect on the D-spacing itself.

Because OVX has been shown to cause a change in viscoelasticity [41], this means the significance of estrogen loss may also be through a change to *OVX collagen viscoelasticity*. This would mean the next step for examination may lie in addressing the collagen-mineral material representation for OVX D-spacings specifically. These changes may be addressed in the form of changing the effective modulus of type I collagen, or collagen's viscous representation. This indication that the Complex Model accounts for the full effects of OVX D-spacing is highly exciting for future postmenopausal research.

The Mendoza Model served as a heavily influential basis for the Complex Model. Because a core purpose of this study is to improve upon the foundation of the work established by Miguel Mendoza, areas that remain open to further fine-tuning are of great interest. In addition to having created a model to extensively express D-spacing variability in this current study, another form of refinement lies in its statistical flexibility. The lack of randomization, replicates, and repeated trials are each shortcomings of previous FEA D-spacing research [78, 52]. This current study has addressed these disadvantages by implementing many avenues to observe variability in the results. In addition to having created four replicates for both Control, Cranial and OVX, Cranial models, each replicate was tested at each frequency a total of ten times (appendix E). Because the MATLAB program utilized to obtain a tangent delta value also has a small degree of intrinsic variability, this expansive dataset ensures that further iterations of the Complex Model are not limited in their statistical ability to prove significant changes.

Though this statistical limitation was carefully addressed in the current study, there is still much room for the Complex Model to grow. Refinement is essential in the ongoing ef-

fort to better understand the driving factors in bone viscoelasticity and osteoporosis. While huge steps have been taken to show the sizable impact of complex computational modeling, future work is warranted.

There are a number of limitations experienced by the current state of the Complex Model. Among the more obvious limitations is the utilization of sheep as a proxy for human bone development. Sheep do not experience the same hormonal pattern as humans, so their usage is imperfect on that basis alone [56]. Ovariectomy helps to simulate menopause, but sheep are still not completely representative of human bone remodeling. However, because the computational modeling of D-spacing is still in its infantile stages, sheep present an excellent substitute for these exploratory efforts. Sheep usage is well-documented [52, 41], and their familiar bone structure and appropriate weight makes ewes the appropriate animal model choice while the FEA representation of D-spacing continues to evolve.

A simplifying assumption made for the purposes of this study was that bone mineral volume fraction remains constant. Though OVX D-spacing parameters were employed in the appropriate Gaussian distribution, the specific percent mineralization for any one period was not implemented into the current model's generation. If specific mineral volume fractions were to be included in the already extensive Python script, an extra randomization parameter may have to be devised to pair specific D-spacings to a range of percent mineralization–should such a relationship be determined. A 0.30 V_V^m value was utilized, in-keeping with the selection made by previous studies [78, 52, 20].

Additionally, hydroxyapatite was modeled simply as an elastic isotropic material with an elastic modulus of 100 GPa and Poisson's ratio of 0.28. This selection was made based on findings indicating that collagen is likely the major factor in dictating viscoelasticity, whereas mineral has a more direct relationship with stiffness [23]. That said, this elastic material simplification is flexible within the Python script, and it may be changed to express viscoelastic behaviors if deemed appropriate.

The selection of a 100 GPa modulus for mineralized hydroxyapatite was made based on previous studies [78, 13], but this may be an extreme estimate of mineral stiffness. Research of heavily mineralized fin whale ear bone has been experimentally found to be as low as 34.1 GPa [93]. Given this modulus range, experimentation with this mechanical property is

welcome.

The presence of mineral in the collagen-hydroxyapatite complex has been proven to have a great impact on the mechanical properties of bone [20, 34]. Because tropocollagen and mineral laid in nucleation zones are of diameters in the nanoscale, experimentally isolating and testing these components is incredebly challenging. The volumetric mineralization selected in the work by Jager and Fratzl [20] as well as Siegmund et al. [78] was noted to be essentially an average of mineralization experimentally measured in the gap region of the collagen-hydroxyapatite network. Manipulating the Complex Model further to authentically model mineralization using the provided OVX experimental data presents a potentially unprecedented way to examine these nano structures.

The decision to keep this parameter consistent was made in an effort to make the main iteration of this current model the arrangement of a great many randomized collagen molecules. Exploring the variable effects of percent mineralization in D-spacing with a modified version of the Complex Model is an extremely exciting avenue for future research as mineral volume fraction may hold a correlation with osteoporosis. Because this model did not tamper with volume fraction or mineral's mechanical expression, this may explain the inability to statistically prove a correlation between OVX D-spacing and tangent delta. Meaningful findings linking these parameters may still be masked in this current version of the Complex Model, but each advancement of this study's FEA approach brings established theoretical models closer to experimental relevancy. Recognizing these shortcomings is the first step to better adapting the Complex Model to accurately incorporate mineralization.

Another important limitation comes in the form of computer processing time. This study was fortunate to have had been provided access by Dr. Scott Hazelwood to computers specifically designed for intense computational analysis. A server was also utilized to run extra FEA jobs. Even with access to all these options, the Complex Model takes a long time to complete analysis. Each model variant took between 8-10 hours to run to completion for each test frequency. This time commitment was ultimately manageable, but a limitation is presented in the Complex Model's flexibility for future adaption, specifically in employing contact forces.

The FEA D-spacing research conducted by Siegmund et al. was particularly interested

in examining the significance of the contact forces that connect collagen and hydroxyapatite units [78]. The cross-linking of these materials is of great scientific interest; collagen molecules specifically rely on this relationship as it pertains to their packing factor and tensile strength [5]. Siegmund's research concluded that modeling these contact forces is critical in accurate theoretical results. This current study perfectly bonded collagen to hydroxyapatite. Neighboring complexes too were perfectly bound to one another.

Siegmund et al. [78] reported differing stress-strain relationships based on the degree of collagen cross-linking. The implementation of a more intricate system of material interactivity may more accurately express the biologic implications of cross-linking within theoretical D-spacing. However, if this relationship were added in addition to the current Complex Model, it is likely that the ensuing FEA model would have to be reduced from 2 x 100 half unit cells down to a more computationally manageable dimension.

The most welcome refinement of the Complex Model comes in the form of computationally representing viscoelastic parameters. Because bone's complex relationship with time-dependent stresses and strains is still being investigated [8, 64], accurately modeling viscoelasticity is not an easy task. The root cause of bone viscoelasticity is generally unknown. Some research pointing to the presence of Haversian canals greatly influencing the structural makeup of bone [9]. Perhaps the simple fact that bone is a two-phase composite biomaterial may be the root cause of strain rate dependent responses [1], while this material property could be a connection to collagen molecule cross-linkages [78]. Estrogen depletion is shown to have a relationship with viscoelasticity, but the mechanism for this is unclear [41]. Combine this general lack of knowledge with the lack of biologically equivalent rheological models [7], and computationally expressing mechanical reactions to oscillatory forces is immensely challenging.

This uncertainty was taken into account during this investigation into collagen molecule D-spacing, but limitations exist. The UMAT utilized for this study [83] was robust and was verified to accurately express both creep and stress relaxation (critical viscoelastic responses exhibited by bone tissue) [7]. However, selecting the correct moduli for these behaviors is intrinsically ambiguous and ultimately warrants more analysis.

Because this UMAT models viscoelasticity based on the Kelvin-Voigt version of the

Standard Linear Solid, moduli/viscosities must be user-selected as to best fit the experimental data. This means that elastic coefficients must be chosen for the two spring elements (E_1 and E_2) in the Standard Linear Solid, and one viscous element must be assigned a viscosity (η_1). A simplified version of the rheological model employed in this UMAT can be viewed in figure 36.

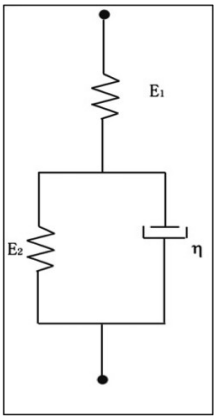

Figure 36: The viscoelastic relationship employed in the Complex Model is the Kelvin-Voigt version of the Standard Linear Solid. The UMAT must be assigned a value for each simplified element on display: E_1, E_2, and η_1. Selecting an appropriate value for each is an ongoing challenge.

The methodology employed in this study was to select a value for the modulus of collagen and work backwards. A common value of collagen's elastic modulus is 2 GPa, as the elastic material is far less stiff than its hydroxyapatite counterpart [2]. For the purpose of this study, this value was set as the effective modulus ($E_{effective}$). Values for the spring constants, E_1 and E_2 were appropriately selected based on this following equation:

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

To keep this relationship valid, E_1 was set to 3 GPa and E_2 was set at 6 GPa. These are the same values the Mendoza Model utilized to create this effective modulus. To compare the impact of computational modeling utilizing many collagen-hydroxyapatite complexes opposed to just a single half unit cell, these same values for E_1 and E_2 were selected and tested. Because each spring element in figure 36 is an abstract representation for biphasic bone composition, these assignments can undoubtedly be swapped and altered.

Because the ultimate goal of this research is to more closely match experimental findings, this processes requires further refinement. Though a modulus of 5 GPa may be the experimental elastic modulus of type I collagen [78], this does not mean it is a proper fit for a viscoelastic effective modulus. The same apprehension surrounds the more general choice of 2 GPa elastic modulus of collagen (not type I specific). Collagen itself can express a wide range of elastic moduli, with some studies placing the material in the low 200 MPa range [94]. As mentioned, accurately examining the effect of OVX sheep on D-spacing may lie within a refinement of these properties. Furthermore, the computationally modeled geometry is that of many collagen molecules collected into a small sub-section of a collagen fibril. These structures exist on the nano and micro level, and assigning a modulus to them based on test findings is imperfect at best.

The answer to more accurately matching the data to the Complex Model likely lies in a combination of these three parameters: E_1, E_2, and η_1. The dashpot's η_1 was experimented upon in this study. When the Complex Model was run at 4 GPa*s, tangent delta data were noticeably lower and less similar to the experimental findings. The higher value made the material stiffer and less representative of bone's viscoelastic behavior. The dashpot viscosity of 1.25 GPa*s was selected based on preliminary findings (figure 29) and was a recommended viscosity in Mendoza's research [52]. However, iterating further with lower values for viscosity may yield even more accurate results in future versions of the Complex Model.

An effective modulus of 5 GPa was also tested, and E_1 and E_2 were scaled accordingly. Similar to the results of utilizing a larger dashpot viscosity, 5 GPa unfavorably lowered tangent delta data away from experimental findings. For this reason, it is theorized that an effective modulus lower than 2 GPa may create favorable data. An effective modulus of 1 GPa is an obvious starting place, with spring values decreased to 1.5 GPa and 3 GPa, respectively. Additionally, as collagen has even been shown to express a range of moduli [18, 94], there is sufficient precedent for lowering the effective modulus to better match experimental findings. Similarly, altering the modulus of hydroxyapatite provides yet another option for fine-tuning.

An important observed statistical trend is that Complex Model values are not signifi-

cantly different from Experimental Model's output value at the "next" test frequency. This pattern exists for all test points, until 15 Hz where both models produce similar results that are not statistically different. For example, the fact that the Complex Model at 1 Hz is not statistically different from the Experimental data at 3 Hz implies that better fitting the data may be a matter of shifting the data slightly. A small change to the effective modulus or dashpot may easily accomplish this translation, which is highly encouraging for future iterations of the Complex Model.

This said, the experimental findings of DMA testing on adult ewes is not a procedure executed with flawless accuracy. As seen in figures 31 and 33, experimental data, too, exhibit similarly sized standard error bars. This is encouraging, as it implies that the computational model is able to calculate viscoelastic material properties with a similar consistency to that of experimental measurements. In any case, a greater sample size of experimental and computational models will only assist in adequately detecting variability.

Should lowering the moduli not be the solution, a Maxwell body may provide an alternate approach to rheological representation. Selecting the Kelvin-Voigt model was based solely on its ability to express creep and stress relaxation, and the Maxwell rheological form could also be experimented with in an effort to more accurately target experimental tangent delta [52]. The UMAT employed in this study was originally conceived using Maxwell parameters, so returning to this form may ease troubleshooting [83].

Other options in modeling viscoelasticity include input of a variable dashpot. The introduction of a variable dashpot may provide a more sophisticated way to approach the viscous properties of the rheological model. Increasingly complex dashpots may increase the non-linear force-relative velocity relationship of the viscous elements [95]. Because it has been statistically proven in this study that the Complex Model is significantly different from experimental data at lower frequencies, implementing more non-linear mechanisms may better fit the theoretical data.

The process on display in figures 27 and 28 revolves around the implementation of a curve fitting program and calculation of tangent delta. There is a small amount of variability when utilizing this program. This study ensured that the Complex Model's output data were matched to the curve fitting program accurately. Initial amplitudes modified for each model

to ensure a R^2 value of at least 0.99. Even with this safeguard, there are some unexpected computational results.

Through examination of the computational model results in appendix E it may be revealed that data occasionally experience small value spikes. This tendency may be attributed to the curve fitting function, which calculates the phase shift utilizing a light degree of estimation. For the purposes of this study, these data points were not thrown out in favor of gathering each piece of information that may reveal clues to more properly modeling bone viscoelasticity. To ensure that the recorded tangent delta data were fairly consistent, results were collected for each model version ten times per test frequency. These spikes were very infrequent (about one datum point per set of ten trials expressed this abnormality). These results were then averaged, and any strange patterns were adequately normalized. Again, the exact nature of these spikes is difficult to qualify biologically, and experimental data certainly have their own set of inconsistencies. In future research where viscoelastic parameters may continue to be approximated more finely, a Cook's Distance may be employed to remove points that are influential outliers.

The selection of a 1.25 GPa*s dashpot and 2 GPa effective modulus in the research by Mendoza was likely a reflection of his model's tendency to both overestimate and underestimate tangent deltas. Statistical analysis confirms this theory, confirming that the Complex Model expresses significantly lower output. Because data collected with the Complex Model exclusively underestimate the theoretical tangent delta, the solution may simply be to lower both moduli further. This consistency in the Complex Model is highly encouraging, as the Complex Model's conservative calculations of this viscoelastic parameter may make refinement a simple process.

Creation of the Complex Model marks many refinements over previous theoretical models. The inclusion of extensive random elements and the representation of many collagen-hydroxyapatite complexes all culminate in a finite element model more apt to accurately model the patterning of bone composition on the nanoscale. The Complex Model was created to account for the D-spacing variability expressed by many experimental bone samples. Once the model's viscoelastic response is fine-tuned through the means suggested in this study, it will be possible to create an accurate Complex Model to represent individual bone

samples.

Creating a one-to-one relationship of theoretical model to individual bone sample will allow for a greatly simplified statistical analysis without the necessity of a repeated measures ANOVA. A one-to-one creation of a randomized Complex Model to a single experimental bone sample will allow users to examine the effect of treatments much more clearly. Because of the flexibility in the Complex Model's code, inserting experimental D-spacing values of individual samples is simple. The mineralization parameters may be changed similarly, allowing FEA research to be conducted utilizing a model that authentically represents the treatment effects on OVX sheep. By implementing changes to mineralization representation and rheological assignment, future users may use the Complex Model to gain new, more accurate insight into the mechanical implications of D-spacing.

5 Conclusion

Bone is a complex biomaterial that is constantly repaired and renewed throughout an individual's life. Its intricate structure is a culmination of multiple physiological systems that carefully maintain the material's upkeep and mechanical integrity. When these systems are thrown off-balance, bone tissue is susceptible to dangerous metabolic disorders such as osteoporosis. Because the exact mechanisms of osteoporosis are unknown, the effectiveness of treatment options is greatly limited.

The material properties of bone are notoriously difficult to accurately quantify, as bone's viscoelasticity sees the tissue stiffen as a result of rapid loading. Bone is fundamentally a biphasic material consisting primarily of type I collagen and mineral hydroxyapatite. By distilling the arrangement of these two components down to their periodicity on the nanometer scale, questions regarding these complicated material properties may begin to be answered.

The results of this study indicate that computationally modeling a complex staggered array of collagen molecules is vital to the accuracy of FEA D-spacing research. Though the simpler computational modeling of past studies has established a great framework to build upon, randomly generating computational models based on the biologic variability of collagen arrangements yields results truer to experimental findings. Through theoretically expressing the complex relationship of hundreds of collagen-mineral complexes, a more accurate representation of bone's fundamental building blocks may be generated.

After adjusting for all other predictors, at the 1% individual significance level, there is not enough evidence to suggest that surgical treatment type has a significant effect on mean tangent delta output. However, the Complex Model was statistically capable of expressing variable tangent delta at differing test frequencies, confirming its adequacy for future research and refinement. The Complex Model was not proven to be statistically different from experimental findings based on Model Type alone, which is highly encouraging as to the validity of this theoretical foundation. Additionally, the statistical trend that illustrates a highly consistent offset in the Complex and Experimental Model outputs at low frequencies welcomes further refinement through potentially simple rheological adjustments. The lack

of significance in the OVX findings coupled with the Complex Model's validated predictive ability points to OVX having an effect on D-spacing's material properties beyond just the expression of variable periodicities, which is a highly exciting avenue for future research.

Tangent delta data retrieved from the use of this Complex Model shows a great deal of consistency. That said, the rheological elements employed to represent bone's complex strain rate dependent behavior are intrinsically theoretical and warrant further refinement. By selectively iterating with lower moduli and viscosities for the spring and dashpot elements, experimental data may be matched even more closely while utilizing this extensive D-spacing model.

This study has made great efforts to generate a complex, randomly generated, biologically relevant representation of D-spacing. The current state of the Complex Model appears more accurate than its theoretical predecessors, and the flexibility of its model generation makes iteration possible. By utilizing the expansive periodicities and randomizations implemented within the Complex Model code, future users may create models to represent specific bone samples, thus painting a clearer picture into the intricacies of bone mechanics with less statistical rigor. With further refinement, the Complex Model will provide an increasingly powerful tool into better examining bone's complicated relationships with viscoelasticity and all-too-prevalent metabolic disorders.

BIBLIOGRAPHY

[1] J M Wallace, B G Orr, J C Marini, and M M B Banaszak Holl. Nanoscale morphology of type 1 collagen is altered in the brtl mouse model of osteogenesis imperfecta. *Journal of Structural Biology*, 173(1):146–152, 2011.

[2] C R Ethier and C Simmons. *Biomechanics: From Cells to Organisms*. Cambridge, 2007.

[3] M Fang, E L Goldstein, A S Turner, C M Les, Bradford G Orr, G J Fisher, Kathleen B Welch, Edward D Rothman, and Mark M Banaszak Holl. Type 1 collagen d-spacing in fibril bundles of dermins, tendon, and bone: Bridging between nano- and micro-level tissue hierarchy. *ACS nano*, 6(11):9503–9514, 2012.

[4] A J Hodge and J A Petruska. *Aspects of protein structure; proceedings of a symposium held in Madras 14-18 January 1963 and organized by the University of Madras. Recent Studies with the Electron Microscope on Ordered Aggregates of the Tropocollagen Molecule*. Academic Press, 1963.

[5] J P R O Orgel, T C Irving, A Miller, and T J Wess. Microfibrillar structure of type i collagen in situ. *PNAS*, 2005.

[6] F Cacho, P J Elbischger, and JF Rodriguez. A constitutive model for fibrous tissues considering collagen fiber crimp. *International Journal of Non-Linear Mechanics*, 42(2):391–402, 2007.

[7] B R Martin, D B Burr, and N A Sharkey. *Skeletal Tissue Mechanics*. Springer, 1998.

[8] R F Gibson, Y Chen, and H Zhao. Improvement of vibration damping capacity and fracture toughness in composite laminates by the use of polymeric interleaves. *Journal of Engineering Materials and Technology*, 2000.

[9] P 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*, 1981.

[10] R B Greer. Wolff's law. *Orthopaedic Review*, 22(10), 1993.

[11] G Karsenty, H M Kronenberg, and C Settembre. Genetic control of bone formation. *Annual Review of Cell and Developmental Biology*, 25(1):629–648, 2009.

[12] D M Robertson and D C Smith. Compressive strength of mandibular bone as a function of microstructure and strain rate. *Journal of Biomechanics*, 11(10-12):455–471, 1978.

[13] J D Currey. *Bone: Structure and Mechanics*. Princeton University Press, 2002.

[14] R J Havlik. Hydroxyapatite. *Plastic and Reconstructive Surgery*, 110(4), 2002.

[15] B Burton, A Gaspar, D Josey, J Tupy, M D Grynpas, and TL Willett. Bone embrittlement and collagen modifications due to high-dose gamma-irradiation sterilization. *Bone*, 61:71–81, 2014.

[16] C A Miles and N C Avery. Thermal stabilization of collagen in skin and decalcified bone. *Physical Biology*, 8(2):26002–26014, 2011.

[17] M P E Wenger, L Bozec, M A Horton, and P Mesquida. Mechanical properties of collagen fibrils. *Biophysical Journal*, 93(4):1255–1263, 2007.

[18] A Gautieri, S Vesentini, A Redaelli, and M J Buhler. Hierarchical structure and nanomechanics of collagen microfibrils from the atomistic scale up. *Nano Letters*, 11(2):757–766, 2011.

[19] J A Petruska and A J Hodge. A subunit model for the tropocollagen macromolecule. *Proceedings of the National Academy of Sciences in the United States of America*, 51(5):871–876, 1964.

[20] I Jager and P Fratzl. Mineralized collagen fibrils: a mechanical model with a staggered arrangement of mineral particle. *Biophysical Journal*, 79:1737–1746, 2000.

[21] F O Schmitt, C E Hall, and M A Jakus. Electron microscope investigations of the structure of collagen. *Journal of Cellular and Comparative Physiology*, 20(1):11–33, 1942.

[22] R H Stinson and P R Sweeny. Skin collagen has an unusual d-spacing. *Biochimica Et Biophysica Acta*, 1980.

[23] D B Burr. The contribution of the organic matrix to bone's material properties. *Bone*, 31(1):8–11, 2001.

[24] J Orgel, T C Irving, A Miller, and Wess T J. Microfibrillar structure of type i collagen in situ. *Proc. Natl. Acad. Science*, pages 9001–9005, 2006.

[25] J P R O Orgel, J D San Antonio, and O Antipova. Molecular and structural mapping of collagen fibril interactions. *Connective Tissue Research*, 52:2–17, 2011.

[26] B Brodsky, E F Eikenberry, and K Cassiday. An unusual collagen periodicity in skin. *Biochimica Et Biophysica Acta*, 1980.

[27] R H Stinson and P R Sweeny. Skin collagen has an unusual d-spacing. *Biochimica Et Biophysica Acta*, 1980.

[28] N Sasaki, N Shukunami, N Matsushima, and Y Izumi. Time-resolved x-ray diffraction from tendon collagen during creep using synchroton radiation. *Journal of Biomechanics*, 32:285–292, 1999.

[29] R Puxkandl, I Kizak, O Paris, J Keckes, W Tesch, S Bernstorff, P Purslow, and P Fretzl. Viscoelastic properties of collagen: Synchrotron radiation investigations and structural model. *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences*, 357(2418):191–197, 2002.

[30] D R Eyre, M A Paz, and P A Gallop. Crosslinking in collagen and elastin. *A. Rev. Biochem*, 53:717–748, 1984.

[31] P F Davison. The contribution of labile crosslinks to the tensile behavior of tendons. *Connective Tissue Research*, 18:293–305, 1989.

[32] A J Bailey, R G Paul, and L Knott. Mechanisms of maturation and ageing of collagen. *Mechanisms Ageing Dev.*, 106:1–56, 1998.

[33] G Vose and A Kubala. Bone strength–its relationship to x-ray-determined ash content. *Human Biology*, 31:261–270, 1985.

[34] J D Currey. The relationship between stiffness and the mineral content of bone. *Journal of Biomechanics*, 1969.

[35] J Katz and K Ukraincik. On the anisotropic elastic properties of hydroxyapatite. *Journal of Biomechanics*, 4:221–227, 1971.

[36] J Lin and J Lane. Osteoporosis - a review. *Clinical Orthopaedics and Related Research*, 2004.

[37] Osteoporosis. PubMed Health. Web, April 2014.

[38] W D Leslie and J T Schousboe. A review of osteoporosis diagnosis and treatment ï£ijoptions in new and recently updated guidelines on case finding around the world. *Current Osteoporosis Reports*, 9(129-40), 2011.

[39] P Vilela and T Nunes. Osteoporosis. *Neuroradiology*, 53(1):185–189, 2011.

[40] Clinical review: Osteoporosis. *GP*, 2012.

[41] C M Les, J L Vance, G T Christopher, A S Turner, G W Divine, and D P Fyhrie. Long-term ovariectomy decreases ovine compact bone viscoelasticity. *Journal of Orthopaedic Research*, 23:869–876, 2005.

[42] J A Kanis. Assessment of osteoporosis at the primary health care leve. *Collaborating Centre for Metabolic Bone Diseases*, 2007.

[43] C J Rosen. Postmenopausal osteoporosis. *The New England Journal of Medicine*, 2005.

[44] Menopause. PubMed Health. Web, April 2014.

[45] B L Riggs, S Khosla, and L J Melton. A unitary model for involutional osteoporosis: estrogen deficiency causes both type i and type ii osteoporosis in postmenopausal women and contributes to bone loss in aging men. *Journal of Bone and Mineral Research*, 13(5):763–777, 1998.

[46] L C Raisz. Physiology and pathophysiology of bone remodeling. *Clinical Chemistry*, 45(8):1353–1358, 1999.

[47] G Wells, P Tugwell, and B Shea. Meta-analysis of the efficacy of hormone replacement therapy in treating and preventing osteoporosis in postmenopausal women. *Endocrine Review*, 2002.

[48] J E Rossouw and G L Anderson. Risks and benefits of estrogen plus progestin in healthy postmenopausal women. *Journal of the American Medical Association*, 288(3):321–333, 2002.

[49] B L Riggs. Selective estrogen-receptor modulators. *New England Journal of Medicine*, 2003.

[50] A Cranney, G Guyatt, L Griffith, G Wells, P Tugwell, and C Rosen. Meta-analyses of therapies for postmenopausal osteoporosis. *Endocrine Review*, 2002.

[51] B L Riggs, S F Hodgson, and W M O'Fallon. Effect of fluoride treatment on the fracture rate in postmenopausal women with osteoporosis. *New England Journal of Medicine*, 322(12):802–809, 1990.

[52] M Mendoza. The effects of variation in collagen d-spacing on compact bone viscoelasticity: a finite element analysis. Master's thesis, California Polytechnic State University, San Luis Obispo, 2013.

[53] A E Goodship, L E Lanyon, and H McFie. Functional adaption of bone to increased stress. *Journal of Bone and Joint Surgery*, 61:539–546, 1979.

[54] L E Lanyon and C T Rubin. Static vs dynamic loads as an influence on bone remodeling. *Journal of Biomechanics*, 17:897–905, 1984.

[55] M R Forwood and C H Turner. The response of rat tibiae to incremental bouts of mechanical loading: a quantum concept for bone formation. *Bone*, 29:1309–1317, 1994.

[56] E Newman. The potential of sheep for the study of osteopenia - current status and comparison with other animal-models. *Bone*, 13(4):214–222, 1995.

[57] L E Lanyon, A E Goodship, C J Pye, and J H KacFie. Mechanically adaptive bone remodeling. *Journal of Biomechanics*, 15:141–154, 1982.

[58] J Calcagno and S Hazelwood. Seasonal and anatomical variation in compact bone remodeling in the adult sheep: A thesis. Master's thesis, Cal Poly San Luis Obispo, 2011.

[59] A S Turner. The sheep as a model for osteoporosis in humans. *The Veterinary Journal*, 163(3):232–239, 2002.

[60] B I Newton. The ovariectomized sheep as a model for human bone loss. *Journal of Comparative Pathology*, 130(4):323–326, 2004.

[61] R S Lakes, J L Katz, and S S Sternstein. Viscoelastic properties of wetcortical bone. i. torsional and biaxial studies. *Journal of Biomechanics*, pages 657–678, 1979.

[62] J McElhaney. Dynamic response of bone and mussle tissue. *J. Appl. Physiol.*, 21(4):1231–1236, 1966.

[63] D R Carter and W C Hayes. The compressive behavior of bone as a two-phase poroous structure. *Journal of Bone and Joint Surgery*, 59:954–962, 1977.

[64] D R Carter and W C Hayes. Compact bone fatigue damage: a microscopic examination. *Clin Orthop*, 127:265–274, 1977.

[65] D R Carter and W E Caler. Cycle-dependent and time-dependent bone fracture with repeated loading. *Journal of Biomechanics Engineering*, 105(166-170), 1983.

[66] J D Currey. Strain rate and mineral content in fracture models of bone. *Journal of Orthopaedic Research*, 6(1):32–38, 1988.

[67] D R Margel-Robertson. *Studies of fracture in bone*. PhD thesis, Stanford University, 1973.

[68] K Piekarski. Fracture of bone. *Journal of Applied Physics*, 41(1):215–223, 1970.

[69] J Yamashita, L Xiaeo, and R Benjamin. Collagen and bone viscoelasticity: A dynamic mechanical analysis. *Journal of Biomedical Materials Research*, 2002.

[70] S M Bowman, L J Gibson, W C Hayes, and T A McMahon. Resultsfrom demineralized bone creep tests suggest that collagen isresponsible for the creep behavior of bone. *Journal of Biomechanical Engineering*, 1999.

[71] J Yamashita, B R Furman, H R Rawls, X Wang, and C M Agrawal. The use of dynamic mechanical analysis to assess the viscoelas-tic properties of human cortical bone. *Journal of Biomedical Materials Research*, 58(1):47–53, 2001.

[72] E Garner, R Lakes, T Lee, C Swan, and R Brand. Viscoelastic dissipation in compact bone: Implications for stress-induced fluid flow in bone. *ASME*, 2000.

[73] R S Lakes. *Viscoelastic Solids*. Boca Raton: CRC, 1999.

[74] H Haslach. *Maximum Dissipation Non-Equilibrium Thermodynamics and Its Geometric Structure*. New York: Springer, 2011.

[75] J G M Ramaekers. The dynamic shear modulus and damping ofcompact bovine metacarpal bone in dependence on the topog-raphy along the bone shaft. *Netherlands Journal of Zoology*, 29(2):151–165, 1978.

[76] Y N Yeni, G T Christopher, and D P Fyhrie. Post-yield energy absorption of cancellous bone is predictable from viscoelastic properties of undamaged bone (abstract). *Proc Orthopedic Research Society*, 27:87, 2002.

[77] Paul Macioce. Viscoelastic damping 101. Roush Industries, Web.

[78] T Siegmund, M R Allen, and D B Burr. Failure of mineralized collagen fibrils: Modeling the role of collagen cross-linking. *Journal of Biomechanics*, 41(7):1427–1435, 2008.

[79] J D Currey. Effects of differences in mineralization on the mechanical properties of bone. *Philosophical Transactions of the Royal Society of London Series B, Biological Sciences*, 304:509–518, 1984.

[80] A Fritsch and C Hellmich. Universal microstructural patterns in cortical and trabecular, extracellular bone materials, micromechanics- based prediction of anisotropic elasticity. *Journal of Theoretical Biology*, 244:597–620, 2007.

[81] B Viswanath, R Raghavan, N Ramamurty, and N Ravishankar. Mechanical properties and anisotropy in hydroxyapatite single crystals. *Scripta Materialia*, 57:361–364, 2007.

[82] M J Buhler. Atomistic and continuum modeling of mechanical properties of collagen: elasticity, fracture, and self-assembly. *Journal of Materials Reserarch*, 21:1947–1961, 2006.

[83] F Richter. *Upsetting and viscoelasticity of vitreous SiO2: experiments, interpretation and simulation*. PhD thesis, The Technical University of Berlin, 2006.

[84] F Richter. Viscoelasticity umat. iMechanica. Web.

[85] J E Shigley and C R Mischke. *Mechanical Engineering Design*. New York: McGraw-hill, 2011.

[86] Stress-stain material laws. University of Colorado Boulder. Web, 2015.

[87] T N Shepherd, J Zhang, T C Ovaert, R K Roeder, and G 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.

[88] Kronecker delta function and levi-civita (epsilon) symbol. Ohio State. Web.

[89] Abaqus user subroutines reference manual (6.10). University of Calgary. Web., 2012.

[90] How to make umats for soils in abaqus. Universidad de los Andes. Web, December 2008.

[91] R Schaller and G Fantozzi. 8.6 damping and toughness, 2001.

[92] C M Les, C L Pechey, E M Michels, M Fang, J M MacLeay, A S Turner, and M M B Banaszak Holl. Estrogen depletion is associated with a loss in anatomic variabilitiy in both bone collagen d-spacing, and in bone material pre-stress. In *ORS 2011 Annual Meeting*, 2011.

[93] J D Currey. The design of mineralised hard tissues for their mechanical functions. *The Journal of Experimental Biology*, 202:3285–3294, 1999.

[94] C A Grant, D J Brockwell, S E Radford, and N H Thomson. Tuning the elastic modulus of hydrated collagen fibrils. *Biophysical Journal*, 97(11):2985–2992, 2009.

[95] 26.2.1 dashpots. Abaqus Analysis User's Manual 6.7. Web.

[96] J Lubliner. *Plasticity Theory*. Courier Corporation, 2008.

[97] Polymer engineering: Viscoelasticity. University of Nottingham. Web.

APPENDIX

A Experimental Data

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

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

Dynamic mechanical analysis was performed at Henry Ford Hospital. Bone samples were tested 1, 3, 9, and 15 Hz. The University of Michigan, Ann Arbor utilized atomic force microscopy (AFM) to examine bone samples and record periodicities utilized in the Complex Model's randomized generation.

B Loading Condition

The Complex Model was subjected to sinusoidal loading to examine the rate-dependent mechanical response of bone loading. This load was applied as a pressure along the free end of the computational model. The amplitude of this pressure was chosen to mimic the stress conditions experienced by the experimental bone samples. Experimental data was provided by Henry Ford Hospital, which utilized a dynamic mechanical analyzer (DMA) to test bone beams at a variety of frequencies.

Bone from sacrificed ewes was machined into beams of 1.75 mm x 1.75 mm x 19.0 mm. DMA testing was accomplished with a static load of 550E-3 N and dynamic loading of 500E-3 N. Beam samples were suspended by two supports separated by a 15 mm distance.

Because the FEA model requires an input pressure for sinusoidal loading, maximum stress must be estimated from this experimental setup for computational validation. The input stress for the Complex Model echoed that of Mendoza's for the purpose of ultimately comparing output data based on model complexity. This maximum input stress may be estimated simply with basic beam bending equations [85]. Basic beam bending assumptions

Table 12: 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	Tangent Delta 3 Hz	Tangent Delta 9 Hz	Tangent Delta 15 Hz
5	C0514	OVX	2	Cr	0.076028	0.064128	0.037049	0.013405
5	C0503	OVX	1	Cr	0.076469	0.065369	0.041262	0.01760
6	C0608	Control	2	Cr	0.077620	0.066401	0.045583	0.026755
6	C0603	Control	1	Cr	0.077544	0.068259	0.045227	0.024657
8	C0816	OVX	1	Cr	0.066467	0.055985	0.035628	0.014889
8	C0802	OVX	2	Cr	0.073447	0.066112	0.048028	0.029362
11	C1120	Control	2	Cr	0.092840	0.071640	0.035124	0.007162
11	C1108	Control	1	Cr	0.072760	0.063211	0.040497	0.021065
18	C1811	OVX	2	Cr	0.070244	0.058924	0.034877	0.012325
18	C1809	OVX	1	Cr	0.075466	0.062502	0.036316	0.012177
22	C2215	Control	2	Cr	0.068225	0.055921	0.031804	0.012429
22	C2207	Control	1	Cr	0.070350	0.060794	0.038610	0.020517

Table 13: Tangent Delta Means and Standard Deviations for Experimental Data. DMA testing was performed by Henry Ford Hospital and provided to this current study.

			Tangent Delta at Each Frequency (Hz)			
Model	Treatment		1	3	9	15
Experimental Data	Control	Mean	0.0765565	0.064371	0.039474167	0.018764167
		SD	0.008827304	0.005614603	0.005476519	0.007505861
	OVX	Mean	0.073020167	0.06217	0.03886	0.016626333
		SD	0.003943524	0.003962517	0.005017641	0.006554976

Table 14: 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	Treatment	Sector	Side	Mean	Median	SD
5	C0514	OVX	2	Cr	67.65167442	67.606	0.786636869
5	C0503	OVX	1	Cr	67.79390385	68.1305	1.771534063
6	C0608	Control	2	Cr	69.47862069	69.2375	1.351576643
6	C0603	Control	1	Cr	70.18066667	70.557	0.81223489
8	C0816	OVX	1	Cr	65.62297778	64.716	2.480279317
8	C0802	OVX	2	Cr	67.59310802	67.798	1.45955533
11	C1120	Control	2	Cr	67.77131429	67.737	2.120829981
11	C1108	Control	1	Cr	68.79706452	68.941	0.979417864
18	C1811	OVX	2	Cr	65.82201538	65.842	1.652481426
18	C1809	OVX	1	Cr	67.15122	67.5815	2.005968032
22	C2215	Control	2	Cr	66.84862857	66.44	2.396897606
22	C2207	Control	1	Cr	67.44332432	67.875	1.156853382

may be made to reach the applied pressure of 3.36 MPa utilized both in the Mendoza Model and this current study.

Beam bending stress, σ_{Max}, may be derived through the following equations:

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

where y is the distance to the neutral axis, and

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

where

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

F_{Static} is representative of the static loading condition, while $F_{Dynamic}$ is the dynamic loading condition. F_{Static} may be represented as a reaction force produced by a single support structure at 7.5 mm from the center of the beam. Making a cut at the center of the beam allows simplified beam stress calculations.

A dynamic loading condition may be written as $F(x,t)$. If x is assumed to be zero (i.e. the force is applied to the center of a beam and distance measurements are taken from the center) and t is the time at which the dynamic force is at a maximum [96], the Dynamic load may be simplified to:

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

therefore

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

and

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

and the bending moment of inertia, I is equal to:

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

where b and h represent the base and height of the square beam cross-section of 1.75 mm x 1.75 mm:

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

therefore

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

This calculated stress was input as the pressure applied in the FEA model. This pressure exhibited a sinusoidal waveform based on the test frequency. The increments for the sinusoidal wave function are explained in detail in appendix H.

C Viscoelastic Equations

The Kelvin-Voigt version of the Standard Linear Solid (SLS) (figure 37) was selected to model bone viscoelasticity, as the rheological model may express both stress relaxation and creep.

Expressions for stress relaxation and creep may be derived through a number of simplified assumptions [97]. These equations were used to validate the finite element model. The governing equation of this rheological model is given by:

$$\eta * E_1 \dot{\varepsilon} + E_1 E_2 \varepsilon = \eta \dot{\sigma} + (E_1 + E_2) \sigma$$

C.1 SLS Model Solution for Creep

Assuming a constant stress, σ_0, integration of the SLS governing equations yields:

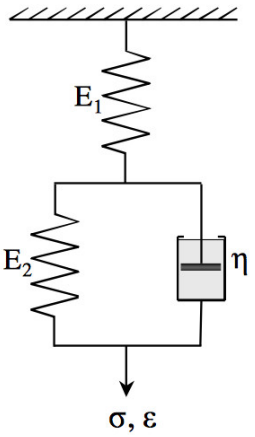

Figure 37: The Kelvin-Voigt version of the SLS expresses both stress relaxation and creep [97].

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

where the retardation time of the viscoelastic material, τ, during creep is equal to

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

which creates a SLS model for creep compliance of

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

C.2 SLS Model Solution for Stress Relaxation

At a constant strain, ε_0, integration of the SLS governing equation yields:

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

where the retardation time of the viscoelastic material, τ, during stress relaxation is equal to

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

which creates a SLS model for stress relaxation of

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

D Model Warnings

This computational model was created with only a single warning message on file. Each model experienced some degree of distorted elements. On average, there were approximately 50 distorted elements per model (eight models in total). A distorted element is described in Abaqus as an element with an angle less than 45 degrees or greater than 135 degrees. With a CPE8 seed size of 0.0005, each model had about 219000 elements. For this reason, 50 distorted elements was fairly negligible.

The location of the distorted elements was variable for each model. Most distorted elements were found towards the free end that experienced the sinusoidal loading conditions. As elements at this end underwent the most deformation, it was not a surprise that distortion was apparent here. Similarly unsurprisingly, the distorted elements were all found at the interface between collagen and hydroxyapatite. Because this border is of sharp geometric conditions with varying elastic moduli, it is to be expected that Abaqus would encounter mechanical warnings here. Furthermore, when probed (as also revealed by examining the key in figures 38 and 39), all distorted elements experienced the same horizontal displacement as their neighbors.

For these reasons, this study is confident that the small number of distorted elements did not impact the results.

E Complex Model Results

The following data were collected from utilizing the Complex Model. Abaqus displacement outputs were converted via a series of MATLAB scripts, and tangent delta data were collected. Data are presented at four test frequencies: 1, 3, 9, and 15 Hz. A single model, Control, Cranial 4, was run with a 4.00 GPa*s dashpot as a means of comparison to the 1.25 GPa*s dashpot, which fit the experimental data more closely.

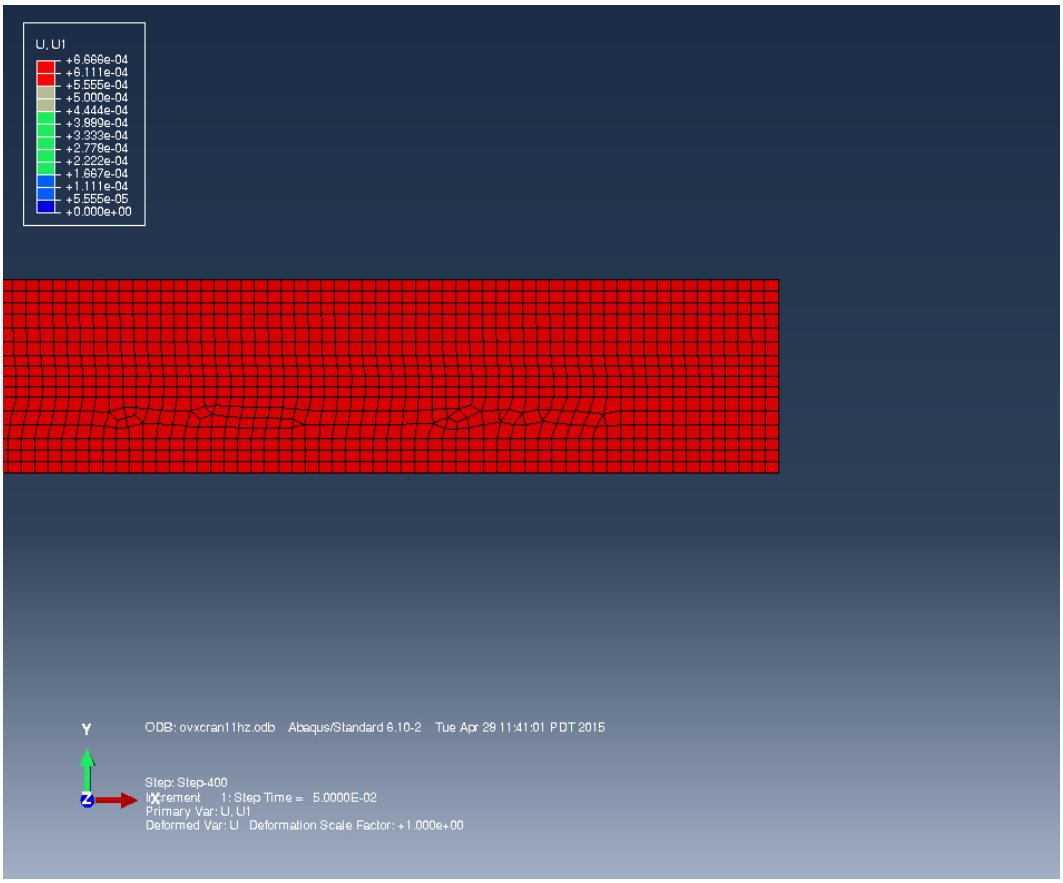

Figure 38: The finite element model's free end was a common location for distorted elements, as seen in OVX, Cranial 1.

Both models (Control, Cranial and OVX, Cranial) were randomly generated four times each in Abaqus using an extensive Python macro. Tangent delta data were collected at the four test frequencies for each of these model versions. Because there is some variability in the curve fitting function that helps to output the tangent delta, data were recorded 10 times at each frequency for each model version. These results were then averaged at each test frequency.

Table 15: Results Utilizing the Complex Model with a 1.25 GPa*s Dashpot. Four versions of the Control, Cranial model were created and tested.

1.25 GPas 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
		Ave	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
		Ave	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
		Ave	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
		Ave	0.0559072	0.0391118	0.0245682	0.02143902

Table 16: Results Utilizing the Complex Model with a 4.00 GPa*s Dashpot. A single version of Control, Cranial was run across all test frequencies and used as a basis of comparison.

4.00 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	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
		Ave	0.0338964	0.0103276	0.00333018	0.00182928

Table 17: Results Utilizing the Complex Model with a 1.25 GPa*s Dashpot. Four versions of the OVX, Cranial model were created and tested.

1.25 GPas Dasphot (eta 1)						
Model	Version	Run	Tangent Delta 1 Hz	Tangent Delta 3 Hz	Tangent Delta 9 Hz	Tangent Delta 15 Hz
OVX, Cran	1	1	0.05635	0.031077	0.043376	0.0067454
		2	0.056326	0.031094	0.011214	0.0066602
		3	0.056269	0.031055	0.011158	0.0067548
		4	0.056261	0.031102	0.011176	0.0067808
		5	0.056314	0.031055	0.011193	0.006768
		6	0.056294	0.031035	0.038681	0.006722
		7	0.056324	0.031085	0.056235	0.0067341
		8	0.065383	0.031048	0.01118	0.0067685
		9	0.084808	0.031046	0.011204	0.0067572
		10	0.056366	0.031094	0.04187	0.006728
		Ave	0.0600695	0.0310691	0.0247287	0.0067419
	2	1	0.055679	0.030763	0.01107	0.0067022
		2	0.055683	0.038824	0.01107	0.0067216
		3	0.055721	0.05624	0.11839	0.11838
		4	0.055721	0.030801	0.011086	0.0066407
		5	0.055767	0.030714	0.011114	0.0067081
		6	0.055745	0.041941	0.011073	0.0067047
		7	0.055753	0.030764	0.011088	0.0066763
		8	0.055761	0.030817	0.01113	0.006648
		9	0.055766	0.030732	0.011117	0.0067175
		10	0.065508	0.030816	0.011133	0.006725
		Ave	0.0567104	0.0352412	0.0218271	0.01786241
	3	1	0.057399	0.031686	0.011379	0.0068808
		2	0.057489	0.031689	0.011418	0.041858
		3	0.057429	0.031702	0.011333	0.0068867
		4	0.057384	0.031682	0.011365	0.0068973
		5	0.057437	0.031625	0.011455	0.0069002
		6	0.057456	0.065338	0.011455	0.0068946
		7	0.057374	0.031668	0.011387	0.0069259
		8	0.057388	0.031627	0.011379	0.0068896
		9	0.057432	0.031652	0.011411	0.0069194
		10	0.057417	0.03163	0.011439	0.0068886
		Ave	0.0574205	0.0350299	0.0114021	0.01039411
	4	1	0.056174	0.041862	0.011218	0.0067607
		2	0.056159	0.030986	0.011203	0.0067619
		3	0.05616	0.030972	0.011175	0.0067046
		4	0.056117	0.030963	0.011181	0.0067083
		5	0.056105	0.030972	0.011253	0.0067298
		6	0.056143	0.030979	0.011205	0.0067638
		7	0.056187	0.030965	0.011147	0.0067491
		8	0.065339	0.031035	0.011191	0.0067657
		9	0.05618	0.031056	0.011203	0.0067055
		10	0.056137	0.031014	0.01113	0.0067166
		Ave	0.0570701	0.0320804	0.0111906	0.0067366

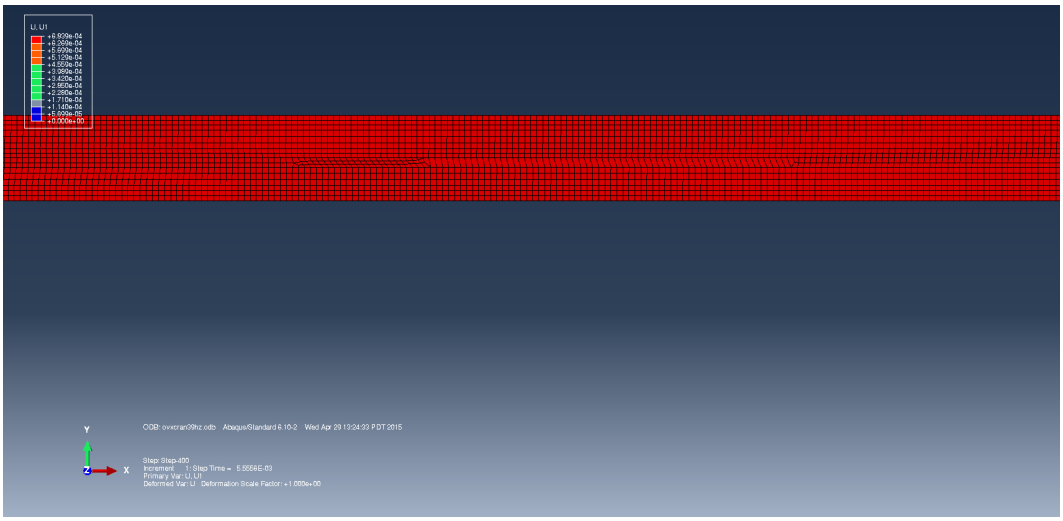

Figure 39: The midsection of the OVX, Cranial 3 model exhibits some distorted elements, but there is no noticeable coloration change in the contour plot, signifying the distortion has little to-no impact on the results.

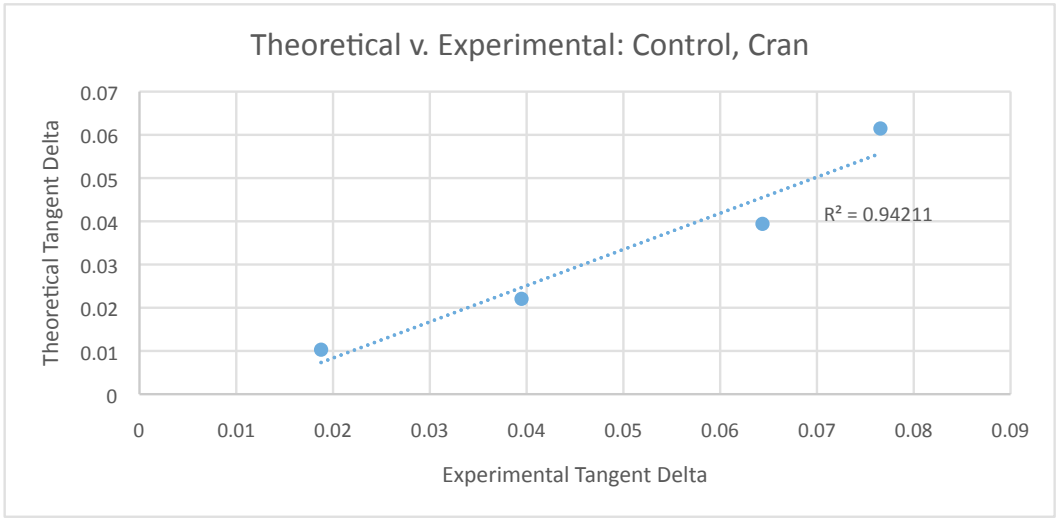

Figure 40: A linear line was fit to data of theoretical v. experimental results. These results were obtained through averaging the tangent delta at each test frequency between the four randomized model replicates. The R^2 value of 0.942 shows a high correlation between experimental and computational findings. The Mendoza Model had previously reported a R^2 value of 0.874 [52], signifying that the Complex Model may be a stronger fit to the experimental data.

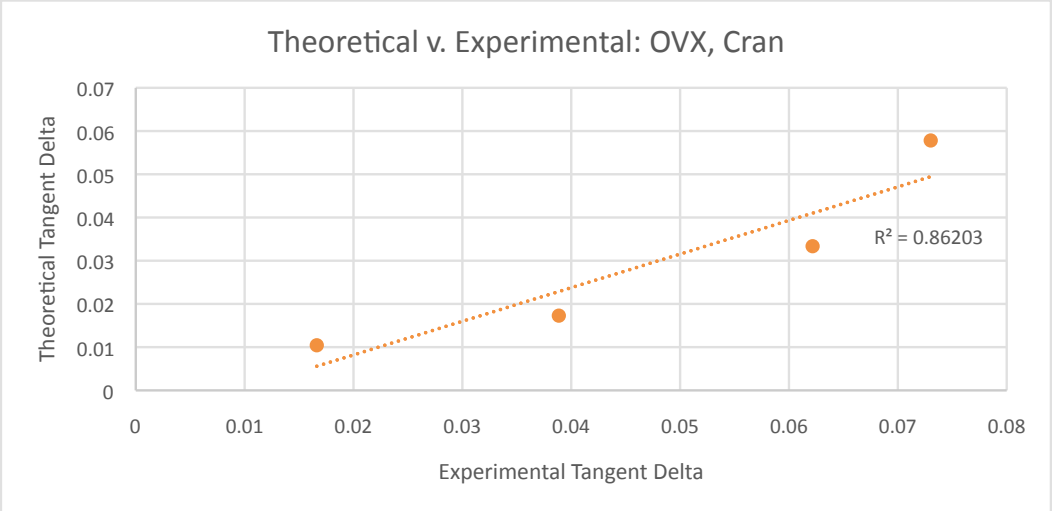

Figure 41: An R^2 value of 0.86203 was obtained when plotting averaged theoretical OVX, Cranial data against the experimental findings. Because previous FEA research has not produced a computational model based explicitly on estrogen depletion's effect on D-spacing, there is no R^2 value applicable for comparison.

Table 18: Standard Deviations for OVX, Cranial Testing. Because the Mendoza Model did not explicitly create a single half unit cell FEA model with OVX dimensions, his data are not applicable for comparison. However, OVX, Cranial data are useful for comparison with Control, Cranial data via statistical analysis. All data were collected with a 1.25 GPa*s dashpot.

Model	Average/ St. Deviation	Tangent Delta			
		1 Hz	3 Hz	9 Hz	15 Hz
Complex Model	Average	0.05782	0.03336	0.01729	0.01043
Complex Model	St. Deviation	0.00153	0.00210	0.0072	0.00524

F Statistical Output

Statistical analysis was completed with the assistance of Professor John Walker through the California Polytechnic State University's Statistics Counseling Service. A repeated measures analysis of variance was selected to explore the relationship across three statistical treatment types: Test Frequency (1, 3, 9, and 15 Hz), Model Type (Complex Model or OVX), and Surgical Treatment (Control or OVX).

Analysis was completed through use of Statistical Analysis Software (SAS). The use of this sophisticated software was necessitated by the multiple levels of experimental units (i.e. sheep donor, bone sector, and the tangent delta measurement at each frequency). Because Mendoza [52] did not provide any repeat testing or output variability in his FEA study, statistical comparisons could not be made between the Mendoza and Complex models.

The Mixed Procedure

Model Information	
Data Set	CUMMINGS.BONES
Dependent Variable	TD
Covariance Structures	Variance Components, Spatial Power
Subject Effects	Sheep2(Mod*Trt), Bone(Sheep2)
Estimation Method	REML
Residual Variance Method	Profile
Fixed Effects SE Method	Model-Based
Degrees of Freedom Method	Containment

Class Level Information		
Class	**Levels**	**Values**
Mod	2	Complex Experimental
Trt	2	CONTROL OVX
Freq	4	1 3 9 15
Bone	12	1 2 3 4 5 6 7 8 9 10 11 12
Sheep2	8	1 2 3 4 5 6 7 8

Dimensions	
Covariance Parameters	4
Columns in X	45
Columns in Z	168
Subjects	1
Max Obs Per Subject	80

Number of Observations	
Number of Observations Read	80
Number of Observations Used	80
Number of Observations Not Used	0

Iteration History			
Iteration	**Evaluations**	**-2 Res Log Like**	**Criterion**
0	1	-445.72190642	
1	4	-458.27471372	0.00662706
2	3	-458.80408575	.
3	1	-459.29407388	0.00007900

4	1	-459.31779798	0.00000035
5	1	-459.31790029	0.00000000

Convergence criteria met.

Estimated R Matrix for Bone(Sheep2) 1 1				
Row	**Col1**	**Col2**	**Col3**	**Col4**
1	0.000037	0.000027	9.498E-6	3.394E-6
2	0.000027	0.000037	0.000013	4.783E-6
3	9.498E-6	0.000013	0.000037	0.000013
4	3.394E-6	4.783E-6	0.000013	0.000037

Estimated R Correlation Matrix for Bone(Sheep2) 1 1				
Row	**Col1**	**Col2**	**Col3**	**Col4**
1	1.0000	0.7096	0.2536	0.09060
2	0.7096	1.0000	0.3573	0.1277
3	0.2536	0.3573	1.0000	0.3573
4	0.09060	0.1277	0.3573	1.0000

Covariance Parameter Estimates		
Cov Parm	**Subject**	**Estimate**
Sheep2(Mod*Trt)		4.058E-6
Bone(Sheep2)	Sheep2(Mod*Trt)	0
SP(POW)	Bone(Sheep2)	0.8424
Residual		0.000037

Fit Statistics	
-2 Res Log Likelihood	-459.3
AIC (smaller is better)	-453.3
AICC (smaller is better)	-452.9
BIC (smaller is better)	-453.1

Type 3 Tests of Fixed Effects				
Effect	**Num DF**	**Den DF**	**F Value**	**Pr > F**
Mod	1	4	45.54	0.0025
Trt	1	4	1.25	0.3258
Mod*Trt	1	4	0.08	0.7884
Freq	3	48	290.80	<.0001

Mod*Freq	3	48	16.17	<.0001
Trt*Freq	3	48	0.24	0.8674
Mod*Trt*Freq	3	48	0.56	0.6439

Least Squares Means

Effect	Mod	Trt	Freq	Estimate	Standard Error	DF	t Value	Pr > |t|	Alpha	Lower	Upper
Mod*Freq	Complex		1	0.05965	0.002591	48	23.02	<.0001	0.01	0.05270	0.06660
Mod*Freq	Complex		3	0.03638	0.002591	48	14.04	<.0001	0.01	0.02943	0.04333
Mod*Freq	Complex		9	0.01967	0.002591	48	7.59	<.0001	0.01	0.01272	0.02662
Mod*Freq	Complex		15	0.01036	0.002591	48	4.00	0.0002	0.01	0.003415	0.01731
Mod*Freq	Experimental		1	0.07479	0.001949	48	38.38	<.0001	0.01	0.06956	0.08002
Mod*Freq	Experimental		3	0.06327	0.001949	48	32.47	<.0001	0.01	0.05804	0.06850
Mod*Freq	Experimental		9	0.03917	0.001949	48	20.10	<.0001	0.01	0.03394	0.04439
Mod*Freq	Experimental		15	0.01770	0.001949	48	9.08	<.0001	0.01	0.01247	0.02292
Mod*Trt	Complex	CONTROL		0.03331	0.002936	4	11.35	0.0003	0.01	0.01979	0.04683
Mod*Trt	Complex	OVX		0.02972	0.002936	4	10.12	0.0005	0.01	0.01621	0.04324
Mod*Trt	Experimental	CONTROL		0.04979	0.002096	4	23.76	<.0001	0.01	0.04014	0.05944
Mod*Trt	Experimental	OVX		0.04767	0.002096	4	22.74	<.0001	0.01	0.03802	0.05732

Differences of Least Squares Means

Effect	Mod	Trt	Freq	Mod	Trt	Freq	Estimate	Standard Error	DF	t Value	Pr > |t|
Mod*Freq	Complex		1	Complex		3	0.02327	0.001649	48	14.11	<.0001
Mod*Freq	Complex		1	Complex		9	0.03997	0.002644	48	15.12	<.0001
Mod*Freq	Complex		1	Complex		15	0.04928	0.002918	48	16.89	<.0001
Mod*Freq	Complex		1	Experimental		1	-0.01514	0.003242	48	-4.67	<.0001
Mod*Freq	Complex		1	Experimental		3	-0.00362	0.003242	48	-1.12	0.2693

Mod*Freq	Complex		1	Experimental		9	0.02048	0.003242	48	6.32	<.0001
Mod*Freq	Complex		1	Experimental		15	0.04195	0.003242	48	12.94	<.0001
Mod*Freq	Complex		3	Complex		9	0.01671	0.002453	48	6.81	<.0001
Mod*Freq	Complex		3	Complex		15	0.02602	0.002858	48	9.10	<.0001
Mod*Freq	Complex		3	Experimental		1	-0.03841	0.003242	48	-11.85	<.0001
Mod*Freq	Complex		3	Experimental		3	-0.02689	0.003242	48	-8.29	<.0001
Mod*Freq	Complex		3	Experimental		9	-0.00279	0.003242	48	-0.86	0.3945
Mod*Freq	Complex		3	Experimental		15	0.01869	0.003242	48	5.76	<.0001
Mod*Freq	Complex		9	Complex		15	0.009310	0.002453	48	3.79	0.0004
Mod*Freq	Complex		9	Experimental		1	-0.05511	0.003242	48	-17.00	<.0001
Mod*Freq	Complex		9	Experimental		3	-0.04360	0.003242	48	-13.45	<.0001
Mod*Freq	Complex		9	Experimental		9	-0.01949	0.003242	48	-6.01	<.0001
Mod*Freq	Complex		9	Experimental		15	0.001978	0.003242	48	0.61	0.5446
Mod*Freq	Complex		15	Experimental		1	-0.06442	0.003242	48	-19.87	<.0001
Mod*Freq	Complex		15	Experimental		3	-0.05291	0.003242	48	-16.32	<.0001
Mod*Freq	Complex		15	Experimental		9	-0.02880	0.003242	48	-8.88	<.0001
Mod*Freq	Complex		15	Experimental		15	-0.00733	0.003242	48	-2.26	0.0283
Mod*Freq	Experimental		1	Experimental		3	0.01152	0.001347	48	8.55	<.0001
Mod*Freq	Experimental		1	Experimental		9	0.03562	0.002159	48	16.50	<.0001
Mod*Freq	Experimental		1	Experimental		15	0.05709	0.002383	48	23.96	<.0001
Mod*Freq	Experimental		3	Experimental		9	0.02410	0.002003	48	12.03	<.0001
Mod*Freq	Experimental		3	Experimental		15	0.04558	0.002334	48	19.53	<.0001
Mod*Freq	Experimental		9	Experimental		15	0.02147	0.002003	48	10.72	<.0001
Mod*Trt	Complex	CONTROL		Complex	OVX		0.003586	0.004152	4	0.86	0.4364

Mod*Trt	Complex	CONTROL		Experimental	CONTROL		-0.01648	0.003607	4	-4.57	0.0103
Mod*Trt	Complex	CONTROL		Experimental	OVX		-0.01436	0.003607	4	-3.98	0.0164
Mod*Trt	Complex	OVX		Experimental	CONTROL		-0.02007	0.003607	4	-5.56	0.0051
Mod*Trt	Complex	OVX		Experimental	OVX		-0.01795	0.003607	4	-4.97	0.0076
Mod*Trt	Experimental	CONTROL		Experimental	OVX		0.002122	0.002964	4	0.72	0.5136

Differences of Least Squares Means

Effect	Mod	Trt	Freq	Mod	Trt	Freq	Adjustment	Adj P	Alpha
Mod*Freq	Complex		1	Complex		3	Tukey-Kramer	<.0001	0.01
Mod*Freq	Complex		1	Complex		9	Tukey-Kramer	<.0001	0.01
Mod*Freq	Complex		1	Complex		15	Tukey-Kramer	<.0001	0.01
Mod*Freq	Complex		1	Experimental		1	Tukey-Kramer	0.0006	0.01
Mod*Freq	Complex		1	Experimental		3	Tukey-Kramer	0.9497	0.01
Mod*Freq	Complex		1	Experimental		9	Tukey-Kramer	<.0001	0.01
Mod*Freq	Complex		1	Experimental		15	Tukey-Kramer	<.0001	0.01
Mod*Freq	Complex		3	Complex		9	Tukey-Kramer	<.0001	0.01
Mod*Freq	Complex		3	Complex		15	Tukey-Kramer	<.0001	0.01
Mod*Freq	Complex		3	Experimental		1	Tukey-Kramer	<.0001	0.01
Mod*Freq	Complex		3	Experimental		3	Tukey-Kramer	<.0001	0.01
Mod*Freq	Complex		3	Experimental		9	Tukey-Kramer	0.9883	0.01
Mod*Freq	Complex		3	Experimental		15	Tukey-Kramer	<.0001	0.01
Mod*Freq	Complex		9	Complex		15	Tukey-Kramer	0.0091	0.01
Mod*Freq	Complex		9	Experimental		1	Tukey-Kramer	<.0001	0.01
Mod*Freq	Complex		9	Experimental		3	Tukey-Kramer	<.0001	0.01
Mod*Freq	Complex		9	Experimental		9	Tukey-Kramer	<.0001	0.01

Mod*Freq	Complex		9	Experimental		15	Tukey-Kramer	0.9986	0.01
Mod*Freq	Complex		15	Experimental		1	Tukey-Kramer	<.0001	0.01
Mod*Freq	Complex		15	Experimental		3	Tukey-Kramer	<.0001	0.01
Mod*Freq	Complex		15	Experimental		9	Tukey-Kramer	<.0001	0.01
Mod*Freq	Complex		15	Experimental		15	Tukey-Kramer	0.3355	0.01
Mod*Freq	Experimental		1	Experimental		3	Tukey-Kramer	<.0001	0.01
Mod*Freq	Experimental		1	Experimental		9	Tukey-Kramer	<.0001	0.01
Mod*Freq	Experimental		1	Experimental		15	Tukey-Kramer	<.0001	0.01
Mod*Freq	Experimental		3	Experimental		9	Tukey-Kramer	<.0001	0.01
Mod*Freq	Experimental		3	Experimental		15	Tukey-Kramer	<.0001	0.01
Mod*Freq	Experimental		9	Experimental		15	Tukey-Kramer	<.0001	0.01
Mod*Trt	Complex	CONTROL		Complex	OVX		Tukey-Kramer	0.8235	0.01
Mod*Trt	Complex	CONTROL		Experimental	CONTROL		Tukey-Kramer	0.0342	0.01
Mod*Trt	Complex	CONTROL		Experimental	OVX		Tukey-Kramer	0.0537	0.01
Mod*Trt	Complex	OVX		Experimental	CONTROL		Tukey-Kramer	0.0174	0.01
Mod*Trt	Complex	OVX		Experimental	OVX		Tukey-Kramer	0.0256	0.01
Mod*Trt	Experimental	CONTROL		Experimental	OVX		Tukey-Kramer	0.8862	0.01

Differences of Least Squares Means

Effect	Mod	Trt	Freq	Mod	Trt	Freq	Lower	Upper	Adj Lower	Adj Upper
Mod*Freq	Complex		1	Complex		3	0.01884	0.02769	.	.
Mod*Freq	Complex		1	Complex		9	0.03288	0.04707	.	.
Mod*Freq	Complex		1	Complex		15	0.04146	0.05711	.	.
Mod*Freq	Complex		1	Experimental		1	-0.02384	-0.00645	.	.
Mod*Freq	Complex		1	Experimental		3	-0.01232	0.005072	.	.

Mod*Freq	Complex		1	Experimental		9	0.01178	0.02918	.	.
Mod*Freq	Complex		1	Experimental		15	0.03326	0.05065	.	.
Mod*Freq	Complex		3	Complex		9	0.01013	0.02329	.	.
Mod*Freq	Complex		3	Complex		15	0.01835	0.03368	.	.
Mod*Freq	Complex		3	Experimental		1	-0.04710	-0.02971	.	.
Mod*Freq	Complex		3	Experimental		3	-0.03558	-0.01819	.	.
Mod*Freq	Complex		3	Experimental		9	-0.01148	0.005910	.	.
Mod*Freq	Complex		3	Experimental		15	0.009991	0.02738	.	.
Mod*Freq	Complex		9	Complex		15	0.002729	0.01589	.	.
Mod*Freq	Complex		9	Experimental		1	-0.06381	-0.04642	.	.
Mod*Freq	Complex		9	Experimental		3	-0.05229	-0.03490	.	.
Mod*Freq	Complex		9	Experimental		9	-0.02819	-0.01080	.	.
Mod*Freq	Complex		9	Experimental		15	-0.00672	0.01067	.	.
Mod*Freq	Complex		15	Experimental		1	-0.07312	-0.05573	.	.
Mod*Freq	Complex		15	Experimental		3	-0.06160	-0.04421	.	.
Mod*Freq	Complex		15	Experimental		9	-0.03750	-0.02011	.	.
Mod*Freq	Complex		15	Experimental		15	-0.01603	0.001364	.	.
Mod*Freq	Experimental		1	Experimental		3	0.007906	0.01513	.	.
Mod*Freq	Experimental		1	Experimental		9	0.02983	0.04141	.	.
Mod*Freq	Experimental		1	Experimental		15	0.05070	0.06348	.	.
Mod*Freq	Experimental		3	Experimental		9	0.01873	0.02948	.	.
Mod*Freq	Experimental		3	Experimental		15	0.03932	0.05183	.	.
Mod*Freq	Experimental		9	Experimental		15	0.01610	0.02684	.	.
Mod*Trt	Complex	CONTROL		Complex	OVX		-0.01553	0.02270	.	.

Mod*Trt	Complex	CONTROL		Experimental	CONTROL		-0.03309	0.000127	.	.
Mod*Trt	Complex	CONTROL		Experimental	OVX		-0.03097	0.002249	.	.
Mod*Trt	Complex	OVX		Experimental	CONTROL		-0.03668	-0.00346	.	.
Mod*Trt	Complex	OVX		Experimental	OVX		-0.03455	-0.00134	.	.
Mod*Trt	Experimental	CONTROL		Experimental	OVX		-0.01153	0.01577	.	.

Tests of Effect Slices

Effect	Mod	Freq	Num DF	Den DF	F Value	Pr > F
Mod*Freq		1	1	48	21.81	<.0001
Mod*Freq		3	1	48	68.79	<.0001
Mod*Freq		9	1	48	36.16	<.0001
Mod*Freq		15	1	48	5.11	0.0283
Mod*Trt	Complex		1	4	0.75	0.4364
Mod*Trt	Experimental		1	4	0.51	0.5136

G Abaqus Job: Input File

Once the full Complex Model has been generated via the Python script in appendix K and appropriate material properties, loads, boundary conditions, element selection, and seed sizes have been applied, an Abaqus job is created as an input file. This input file describes all the elements (approximately 219000), and establishes the loads and material properties.

The loads and material properties must be adjusted by the user after the input file is generated. The relevant pieces of this input file that are flexible to user selection are included within this appendix. These portions that warrant user augmentation to reflect the desired test conditions are highlighted. These values are the same as chosen by Mendoza in his model generation [52], though additional experimentation with these constants were explored in this current study (such as changing the effective modulus and dashpot viscosity).

The green highlight represents the area for user input of angular frequency. Angular frequency is simply:

$$\omega = 2\pi f$$

Adjusting solely the value of angular frequency in the portion highlighted in green is the first step in correctly altering the test loading conditions, as it changes the frequency of the applied sinusoidal pressure. These changes may be seen in table 19.

Table 19: Input File: Green Highlight Key. The input file must be manipulated by adjusting the angular frequency to reflect the test frequency.

Desired Test Frequency	Angular Frequency (radians per second)
1 Hz	6.28319
3 Hz	18.84956
9 Hz	56.54867
15 Hz	94.24778

Adjusting the test frequency also requires altering the step increment (pink highlight). The Complex Model undergoes 20 load cycles. Each cycle is completed through 20 steps. Together, this creates 400 data points that may be used in calculating the nodal displacement and resulting tangent delta.

Each step must represent the amount of time to complete a single cycle. At a test frequency of 1 Hz, for example, each step increment would be 0.05 seconds, as 20 increments would correlate with one cycle per second. Table 20 illustrates input values necessary to create each test frequency.

These step increments must be applied to each of the 400 steps in the 20 cycle loading condition. "Find and Replace With" feature of most text editors allows this to be accomplished simply.

Finally, the yellow highlight emphasizes the material properties utilized in this current study. This line of the Python script greatly alters the viscoelastic behavior. Table 21 clarifies how each value corresponds with the material property assigned in the Kelvin-Voigt version of the Standard Linear Solid. As previously mentioned, all values are appropriately scaled for the model dimensions, which are μm x μm. As such, an elastic modulus of 0.003 is equivalent to 3 GPa. This transformation may also be viewed in table 21.

Table 20: Input File: Pink Highlight Key. The time increment for each step is emphasized in the pink highlighted region. Values for the "Smallest Increment" were chosen to reflect Mendoza's selection [52]. Changes to the "Smallest increment" parameter did not appear to have any large repercussions.

Desired Test Frequency	Step Time	Initial Increment	Smallest Increment	Largest Increment
1 Hz	0.0500	0.0500	1.0E-06	0.0500
3 Hz	0.0166	0.0166	1.0E-10	0.0166
9 Hz	0.0055	0.0055	1.0E-10	0.0055
15 Hz	0.0033	0.0033	1.0E-10	0.0033

Table 21: Input File: Yellow Highlight. Terms highlighted in yellow dictate the material properties assigned to the rheolohical elements. E_1 refers to the elastic modulus of the lone spring element while E_2 represents the spring element within the Kelvin-Voigt body (i.e. the spring in parallel with the dashpot). The dashpot viscosity is represented by η_1. Poisson's ratios correspond with the rheological element of the same subscript.

Name	Description	Rheologial Variable	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)	Daspoht Viscosity	η_1	0.00125	1.25 GPa*s
Prop(4)	Poisson's Ratio of Lone Spring Element	v_{E1}	0.2	0.2
Prop(5)	Poisson's Ratio of Kelvin-Voigt Spring Element	v_{E2}	0.2	0.2
Prop(6)	Poisson's Ratio of Dashpot	$v_{\eta 1}$	0.2	0.2

The full input file is run in conjunction with the modified version of the Richter UMAT file (appendix H). The properties described as "Prop(1), Prop(2), etc." are passed into this Richter UMAT file and converted to appropriate material values (i.e. bulk moduli, shear moduli, bulk viscosity, and shear viscosity) outlined in section 2.7.

```
*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
**
** BOUNDARY CONDITIONS
**
** Name: YSYM Type: Symmetry/Antisymmetry/Encastre
*Boundary
_PickedSet33, YSYMM
** ----------------------------------------------------------------
**
** 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
_PickedSet36, XSYMM
**
** LOADS
**
** Name: PRESSURE   Type: Pressure
*Dsload, amplitude=SINUSOIDAL
_PickedSurf32, 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
** ----------------------------------------------------------------
**
** STEP: Step-3
**
*Step, name=Step-3, 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
** ----------------------------------------------------------------
```

H Python Script: Material Properties

The following UMAT was originally created by Dr. Frank Richter at the Technical University of Berlin [83]. This code was then adapted by Mendoza and utilized in this study [52].

This viscoelastic material definition is based off of the Kelvin-Voigt version of the Standard Linear Solid, a model chosen to express both stress relaxation and creep. Other rheological models may express these non-linear mechanical responses, and the selection of the Kelvin-Voigt basis was arbitrary.

An important point of clarification is that though D-spacing is a parameter that dictates geometries on the nano-scale, finite element program Abaqus is limited to inputs in the micron (μm) range. For this reason, a measurement of 67 nm would have to be input at $67E-3$ micrometers. This conversion affected many values within this FEA analysis (i.e. elastic moduli, applied stress), and changes were made accordingly.

```
C
C  SDVINI SUBROUTINE TO INITIALIZE AND KEEP TRACK OF STRESS INCREMENTS
C  FROM THE PREVIOUS CALCULATION
C
      SUBROUTINE SDVINI(STATEV,COORDS,NSTATV,NCRDS,NOEL,NPT,
     1 LAYER,KSPT)
C
      INCLUDE 'ABA_PARAM.INC'
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
      STATEV(1) = 0.0
      STATEV(2) = 0.0
      STATEV(3) = 0.0
C
      RETURN
      END
C
C  3D FORMULATION OF THE STANDARD LINEAR SOLID (KELVIN BODY)
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
      INCLUDE 'ABA_PARAM.INC'
C
      CHARACTER*8 CMNAME
      DIMENSION STRESS(NTENS),STATEV(NSTATV),
     1 DDSDDE(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
            DDSDDE(K2,K1) = 0
C            WRITE(*,*) 'K1 = ',K1
C            WRITE(*,*) 'K2 = ',K2
         END DO
      END DO
C
      DO K1=1,NDI
         DDSDDE(K1,K1) = TERM2C
      END DO
C
      DO K1=2,NDI
         N2 = K1-1
         DO K2=1,N2
            DDSDDE(K2,K1) = TERM3C
            DDSDDE(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
         DDSDDE(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
```

I Python Script: Node Retrieval

This Python script was generated by Mendoza for the purposes of retrieving nodal displacements from a completed Abaqus job [52]. This script was modified to be applicable for the Complex Model.

This Python script was run to retrieve the displacement at each node after the 400 step loading cycle had completed.

These nodes may be easily selected within Abaqus utilizing the built-in Query->Probe Values->Nodes pathway found in the Tools option. Applicable nodes for displacement are located along the unconstrained edge of the Complex Model. Once identified, this selection of approximately 30 points would be input into the respective node retrieval document for the corresponding model (green highlight). The specific nodes for a set model (i.e. the node numbers for model Control, Cranial 1 were not the same for Control, Cranial 2) were to be placed in this script, and the script was activated through the Run Script option in Abaqus. A "node_displacement" file was generated in the designated directory (yellow highlight).

This node_displacement file specific for each model at each test frequency was then processed through the MATLAB code (appendix J).

```
# 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/accummin/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
# NormalDSpacing model
node_list =
[1195,675475,96044,675250,96043,675025,1194,680100,97016,679964,97015,679695,1197,6754
80,96158,675477,1183,670211,95105,670206,1167,664569,93874,664564,93873,664559,1151,66
0075,92898,660081,92897,660086,1154]
# LowDSpacing model
#node_list = [2,3,5,8,32,33,328,559,560,1299,1344,1389,1394,1802,3550,3552,3553,1]
# HighDSpacing model
#node_list = [2,3,5,8,36,37,388,663,664,1545,1598,1651,1656,2142,4226,4228,4229,1]
# LowDSpacingConstMin model
#node_list = [5,8,9,10,536,539,540,550,551,2122,2530,3489,3492,3493,3549,3552,3553,1]
# HighDSpacingConstMin model
#node_list = [5,8,9,10,586,589,590,627,628,2349,2835,4003,4006,4007,4225,4228,4229,1]
# LowDSpacingConstCol model
#node_list = [5,8,9,10,536,539,540,550,551,2122,2530,3489,3492,3493,3549,3552,3553,1]
# HighDSpacingConstCol model
#node_list = [5,8,9,10,586,589,590,627,628,2349,2835,4003,4006,4007,4225,4228,4229,1]

# 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
```

J MATLAB Code: Post-Processing

Post-processing of nodal displacements was accomplished through MATLAB code. Three pieces of code performed this function in tandem: TangentDelta.m, CurveFit.m, rsquare.m.

Each node analyzed for displacement must be input manually into this MATLAB program (yellow highlight). The TangentDelta.m code transforms the nodal displacements to a strain through user input of the model length (pink highlight). The test frequency run for this particular model is included for calculation of the time factor. The "initial_amp" must be altered slightly to accurately match the sinusoidal waves to represent viscoelastic behavior. An R^2 value is reported, and the initial_amp must be altered to receive an R^2 of at least 0.99. Initial_amp is emphasized with a green highlight.

A small amount of estimation matches the sinusoidal loading and deformation curves. Because tangent delta values change slightly between runs of this MATLAB code, output tangent delta was collected 10 times for each model variant before being averaged.

```
%TangentDelta.m

%% MatLab code to aqcuire 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,A
23,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_Disp = squeeze(mean(Disp_Data));
Ave_Disp = transpose(Ave_Disp);

%% Calculate the average strain behavior based on total length
p = 6.8510358;
Data = Ave_Disp/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])
```

```
% 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);
```

```
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));
```

K Python Script: Model Generation

The following Python Script was written to generate the Complex Model. The script utilizing a randomized Gaussian distribution to generate periodic unit lengths based on the mean and standard deviations of D-spacing discovered via AFM on test sheep. These data were provided by the University of Michigan, Ann Arbor.

Random D-spacings are computed (*ds*), and then a constant mineralization parameter is applied to each half unit cell. This 0.84 value brings each half unit to the 30% volumetric mineralization discovered utilized in past FEA studies [78, 52].

Length1 and *Length2* correspond to the top and bottom row lengths, respectively, and are the summation of all the half unit cell periodicities within that row. The longer row length, *LengthF (Lf)*, is then determined. These values are clearly printed to the user in the Abaqus messages window. As discussed in detail in section 2.6, Lf dictates the rectangular geometry of the entire model. A spacer is included to complete the geometry of the shorter row and make the rows an equal length.

An "if" statement divides the code into two sections: one where the top row is longer (i.e. the top row dictates the dimension Lf) and one section where the bottom row is longer. This is important for accurate placement of the spacer on the shorter row, exclusively. This spacer is assigned entirely collagen material properties. This spacer may also be swapped with a collagen-hydroxyapatite complex of a set size if the spacer is larger than one standard deviation under the mean periodic length for that particular model.

To ensure that this added section of collagen doesn't skew the material properties of the final half unit cells by more than 5%, an "if" statement section ensures the model's biological relevance. If this relationship is violated, an error message is displaced and the code must be rerun until a spacer of appropriate size is generated.

```
# 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 (OVX Cran)
#    DSmean=0.06693915
#    DSstdev=0.000968631

#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+ds30
    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+ds80
    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))
#       session.viewports['Viewport: 1'].view.fitView()
      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__']
#       p = mdb.models['Model-1'].parts['Composite Bone']
#       p.DatumPointByCoordinate(coords=(0.0, 0.00125, 0.0))
#       p = mdb.models['Model-1'].parts['Composite Bone']
#       p.DatumPointByCoordinate(coords=(0.0, 0.00275, 0.0))
#       p = mdb.models['Model-1'].parts['Composite Bone']
#       p.DatumPointByCoordinate(coords=(0.0, 0.00425, 0.0))
#       p = mdb.models['Model-1'].parts['Composite Bone']
#       p.DatumPointByCoordinate(coords=(0.0, 0.00575, 0.0))
#       p = mdb.models['Model-1'].parts['Composite Bone']
#       p.DatumPointByCoordinate(coords=(0.05628, 0.007, 0.0))
#       p = mdb.models['Model-1'].parts['Composite Bone']
#       p.DatumPointByCoordinate(coords=(0.067, 0.007, 0.0))
#       p = mdb.models['Model-1'].parts['Composite Bone']
#       p.DatumPointByCoordinate(coords=(0.07772, 0.007, 0.0))
#       p = mdb.models['Model-1'].parts['Composite Bone']
#       p.DatumPointByCoordinate(coords=(0.01072, 0.0, 0.0))
#       p = mdb.models['Model-1'].parts['Composite Bone']
#       p.DatumPointByCoordinate(coords=(0.067, 0.0, 0.0))
#       p = mdb.models['Model-1'].parts['Composite Bone']
#       p.DatumPointByCoordinate(coords=(0.12328, 0.0, 0.0))
#       p = mdb.models['Model-1'].parts['Composite Bone']
#       p.DatumPointByCoordinate(coords=(0.13, 0.0, 0.0))
#       p = mdb.models['Model-1'].parts['Composite Bone']
      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,
```

```
      0.0), 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 spcer 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, 0.0),
        point2=(set20+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-hys,
        0.0), point2=(set20+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
        )
        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 EIGTH spacer to bottom row NO WAY IS THIS FOR REAL?!'
       #creates boundary line for new DSpace in EIGTH 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 EIGTH spacer substitution case
        s.Line(point1=(set20+spacer+spacer+spacer+spacer+spacer+spacer+spacer+hys,
        0.0), point2=(set20+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*'

       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__']
```

```
#SET CREATION
      p = mdb.models['Model-1'].parts['Composite Bone']
      f = p.faces
      faces = f.getSequenceFromMask(mask=('[#7f7b5 ]', ), )
      p.Set(faces=faces, name='COLLAGEN SET')
      p = mdb.models['Model-1'].parts['Composite Bone']
      f = p.faces
      faces = f.getSequenceFromMask(mask=('[#84a ]', ), )
      p.Set(faces=faces, name='HYDROXYAPATITE SET')

#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))
#       session.viewports['Viewport: 1'].view.fitView()
      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__']
#       p = mdb.models['Model-1'].parts['Composite Bone']
#       p.DatumPointByCoordinate(coords=(0.0, 0.00125, 0.0))
#       p = mdb.models['Model-1'].parts['Composite Bone']
#       p.DatumPointByCoordinate(coords=(0.0, 0.00275, 0.0))
```

```
#       p = mdb.models['Model-1'].parts['Composite Bone']
#       p.DatumPointByCoordinate(coords=(0.0, 0.00425, 0.0))
#       p = mdb.models['Model-1'].parts['Composite Bone']
#       p.DatumPointByCoordinate(coords=(0.0, 0.00575, 0.0))
#       p = mdb.models['Model-1'].parts['Composite Bone']
#       p.DatumPointByCoordinate(coords=(0.05628, 0.007, 0.0))
#       p = mdb.models['Model-1'].parts['Composite Bone']
#       p.DatumPointByCoordinate(coords=(0.067, 0.007, 0.0))
#       p = mdb.models['Model-1'].parts['Composite Bone']
#       p.DatumPointByCoordinate(coords=(0.07772, 0.007, 0.0))
#       p = mdb.models['Model-1'].parts['Composite Bone']
#       p.DatumPointByCoordinate(coords=(0.01072, 0.0, 0.0))
#       p = mdb.models['Model-1'].parts['Composite Bone']
#       p.DatumPointByCoordinate(coords=(0.067, 0.0, 0.0))
#       p = mdb.models['Model-1'].parts['Composite Bone']
#       p.DatumPointByCoordinate(coords=(0.12328, 0.0, 0.0))
#       p = mdb.models['Model-1'].parts['Composite Bone']
#       p.DatumPointByCoordinate(coords=(0.13, 0.0, 0.0))
#       p = mdb.models['Model-1'].parts['Composite Bone']
        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,
```

```
        0.0035))

        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-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)
        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, 0.007),
       point2=(set10+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+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
       )
        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 EIGTH spacer to top row AGAINST ALL ODDS WHY SO MANY SPACERS
       GOODNESS'

      #creates boundary line for new DSpace in EIGTH 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 EIGTH 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)
        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__']

#SET CREATION
      p = mdb.models['Model-1'].parts['Composite Bone']
      f = p.faces
      faces = f.getSequenceFromMask(mask=(
          '[#dbe7dbd6 #dbe7dbe7:3 #af97dbe7 #bd7ebebf #bd7ebd7e:7 #f67ebd7e
          #e7dbf3f6',
          ' #e7dbe7db:8 #7dabe7db #defebf ]', ), )
      p.Set(faces=faces, name='COLLAGEN SET')
      p = mdb.models['Model-1'].parts['Composite Bone']
      f = p.faces
      faces = f.getSequenceFromMask(mask=('[#84a ]', ), )
      p.Set(faces=faces, name='HYDROXYAPATITE SET')

#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)
```