DEVELOPMENT OF A PHYSIOLOGIC IN-VITRO TESTING METHODOLOGY FOR ASSESSMENT OF CERVICAL SPINE KINEMATICS

by

Kevin Michael Bell

BS in Mathematics, Westminster College, 2002

BS in Bioengineering, University of Pittsburgh, 2002

MS in Mechanical Engineering, University of Pittsburgh, 2006

Submitted to the Graduate Faculty of

the Swanson School of Engineering in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy in Bioengineering

University of Pittsburgh

2013

DEVELOPMENT OF A PHYSIOLOGIC IN-VITRO TESTING METHODOLOGY FOR ASSESSMENT OF CERVICAL SPINE KINEMATICS

Kevin Michael Bell, PhD

University of Pittsburgh, 2013

In-vitro biomechanical testing has been critical in the design and evaluation of spinal surgical instrumentation, however determination of realistic physiologic loading levels has proven difficult outside of the in-vivo setting. Unconstrained pure moment testing combined with the hybrid testing method is currently the gold standard test protocol for evaluation of motion preservation technology and adjacent level effects. Pure moment testing is well suited for making relative comparisons between treatments, but is currently not based on or representative of in-vivo spine motion, bringing the clinical relevance into question.

The human cervical spine supports substantial compressive load in-vivo arising from muscle forces and the weight of the head. However, traditional in-vitro testing methods rarely include compressive loads, especially in investigations of multi-segment cervical spine constructs. Therefore, a systematic comparison of standard pure moment testing without compressive loading versus published and novel compressive loading techniques (follower load, axial load, and combined load) was performed. To achieve a pure moment test, a robot/UFS testing system was programmed with hybrid control, which combined load and displacement control to overcome the limitations of either control methodology alone. A follower load system was developed with actively controlled linear actuators and integrated into the robot/UFS testing system's control algorithm. Thorough investigation of the integrated system ensured that the pure moment assumption was upheld and enabled characterization of the kinetics resulting from the application of follower load. In contrast, axial load was applied perpendicular to superior most vertebral body using the robot end-effector; it did not maintain the pure moment assumption resulting in alterations of the segmental motion patterns.

The pure moment testing protocol without compression or follower load was not able to replicate the typical in-vivo segmental motion patterns throughout the entire motion path. Axial load or a combination of axial and follower load was necessary to mimic the in-vivo segmental contributions at the extremes of the extension-flexion motion path. It is hypothesized that dynamically altering the compressive loading throughout the motion path is necessary to mimic the segmental contribution patterns exhibited in-vivo—a novel concept that will be explored in future investigations.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

PREFACE

First and foremost I would like to thank my family for their tireless and relentless support of me in every possible way. My beautiful wife Stephanie has made many personal sacrifices enabling me to continue to push forward and was always there to encourage me when I was ready to hang it up. I love you Stephanie and wouldn't want to share this journey with anyone else. I also want to thank my son Ethan for asking me (almost daily), "Are you a doctor yet?" – The day is finally here Ethan; I am sorry I made you wait so long! I also want to thank my parents, Paul and Marilyn Bell, for raising me to value the pursuit of knowledge and faith in God.

I would also like to thank my advisor, Dr. Scott Tashman, for all his help and guidance; he knew exactly when to pat me on the back or (more often) when to kick me in the butt! I would like to thank my co-advisor Dr. Gwen Sowa for being a faithful mentoring presence in my life since she entered the Ferguson Laboratory. I want to thank Dr. Richard Debski for being an inspiring teacher since my days as an undergraduate. I want to thank Dr. James Kang for his gracious financial support and his empowering leadership style of "Letting the Horses Run". Finally, I want to thank Dr. Lars Gilbertson for instilling a passion for research and Spine Biomechanics in me and for pushing me to continue to pursue excellence and humility. Finally, I want to thank Bill Anderst who graciously granted me direct access to the in-vivo dataset that was critical to the success of this project. Bill was also available to answer numerous questions and review manuscripts along the way.

None of this would have been possible without the Ferguson Laboratory and co-workers and friends like Rob Hartman – although he refers to me as his mentor, I am sure he has taught me more than I will ever teach him – I also could not let him get his PhD before me, so thanks for the motivation! Paulo Coelho and I co-managed the Ferguson Lab, and I wouldn't have lasted as long as I have without his levity and stress relieving antics (laboratory baseball!). Finally, I want to thank Lou Duerring for being my mother in the lab – her constant support and enduring presence exemplifies the Ferguson Laboratory.

1.0 INTRODUCTION

The goal of motion preservation hardware in spine surgery is stabilization of the pathologic joint while preserving/restoring normal physiologic motion at the operated as well as the adjacent levels. In-vitro biomechanical testing has been critical in the design and evaluation of motion preservation devices[1, 2] however, traditional testing methodologies are currently not based on (or representative of) physiologic motion, bringing the clinical relevance into question[3, 4]. The objective of this project is to identify and verify the appropriate in-vitro loading conditions that would replicate the in-vivo kinematics (and kinetics) of the cervical spine, with the overall goal of improving the biofidelity of the experimental platform. This study relies of on the availability of accurate 3D, dynamic, muscle driven, in-vivo, kinematics and a testing system capable of reproducing the kinematics in-vitro. Access to in-vivo kinematics of unprecedented accuracy is the driving force behind the proposed methods, allowing for an iterative experimentation/modeling approach that would not otherwise be possible. It is also imperative that the in-vitro testing apparatus is sufficiently reproducible yet flexible in configuration, making the robot/UFS testing system ideal for this application. Direct integration/comparison of the two technologies will enable clinically relevant physiologic investigations of primary and adjacent level effects of motion preservation technology in the cervical spine.

1.1 SPECIFIC AIMS

Implement the design of experiments methodology to evaluate novel in-vitro compressive loading methods (axial load, follower load, combined axial/follower load) with the goal of identifying the appropriate in-vitro loading conditions that would replicate recorded (Biodynamics Laboratory, University of Pittsburgh) / published (follower load (Miura et al. 2002), axial load (DiAngelo and Foley 2004)) in-vivo global and segmental kinematics (extension-flexion range of motion (ROM) and temporal sequencing of vertebral movement).

1.1.1 Combined Axial/Follower Load

H1a: Increasing magnitude of follower load will result in a uniform moment variation across the spine construct leading to overall stiffening (stability) of the spinal construct.

H1b: Increasing magnitude of axial load will result in a non-uniform moment variation across the spine construct leading to increased ROM of middle segments.

H1c: Combining axial compression with follower load will have a synergistic effect enabling (+/- 10%) agreement with recorded / published extension-flexion range of motion.

1.1.2 Dynamic Follower Load

H2a: Dynamic alteration of the follower load magnitude will enable global and segmental temporal sequencing of vertebral movement to be adjusted throughout the motion path resulting in (+/- 10%) agreement with recorded data.

1.1.3 Secondary Aim

Due to limited availability of in-vivo kinetic data the primary aim will be to validate the system based on kinematics; however the kinetic data (intradiscal pressure, load/displacement curve parameters) will also be assessed and reported.

1.2 BACKGROUND

Anterior cervical decompression and fusion (ACDF) is a well-established surgical treatment option for compressive pathologies associated with radiculopathy and/or myelopathy. This procedure allows surgeons direct access to the anterior aspect of the neural structures permitting thorough decompression, and when combined with a bone graft and plating instrumentation, results in excellent fusion rates and improvement of pain scores and neurologic status[5]. However, recent reports have shown the fusion procedure to have negative long term effects, with as high as 25% of patients showing pathology at adjacent segments within 10 years[6-10]. It is theorized and that the fusion procedure leads to a stress concentration at the instrumented level and leads to subsequent motion redistribution and resultant increased stresses at the adjacent levels [4, 11, 12].

The incidence of accelerated adjacent segment degeneration has given rise to motion preservation technology as a possible alternative. The goal of motion preservation hardware in spine surgery is stabilization of the pathologic (degenerated) joint while preserving/restoring normal physiologic motion at the operated as well as the adjacent levels[1, 13, 14].

1.3 SIGNIFICANCE

In-vitro biomechanical testing has been critical in the design and evaluation of both spinal fusion and motion preservation devices. However, there is a fundamental flaw/limitation in the typical methods used to evaluate motion preservation devices. As stated above the primary goal of motion preservation is to preserve physiologic motion at the operated and adjacent levels and therefore a biomechanical testing methodology that aims to replicate physiologic motion is required [4, 15].

1.3.1 Broader Significance

The methods established and data collected in this project are a prerequisite to clinically relevant physiologic investigations of primary and adjacent level effects of motion preservation technology in the cervical spine. The methodology will also serve as the foundation for future in-vitro experimentation where physiologic kinematics/kinetics are critical to the clinical relevance of the data such as determination of native biomechanical properties, establishment/validation of computational models, and as a loading scheme for mechanobiology experiments.

1.4 CRITICAL BARRIERS TO PROGRESS IN THE FIELD

1.4.1 Status and Limitations of Current In-vitro Testing Methodologies

Current cadaver testing methodologies are invaluable tools for comparative surgical instrumentation studies however, determination of realistic physiologic loading levels has proven difficult outside of the in-vivo setting. Unconstrained pure moment testing combined with the hybrid testing method is currently the gold standard test protocol for evaluation of motion preservation technology and adjacent level effects. Pure moment testing was specifically designed to apply uniform loading at each cross section throughout the length of a spinal construct, whereby permitting irregularities to be identified[3]. This test methodology is well suited for making relative comparisons between treatments, but the in-vitro testing is currently not based on, or representative of in-vivo motion / loads bringing the clinical relevance into question.

Additionally, the traditional in-vitro testing methods rarely include compressive loads, especially in multi-segment constructs, even though the human cervical spine supports substantial compressive load in-vivo arising from muscle forces and the weight of the head [16]. Various methods of modeling physiologic loading have been reported in the literature including axial forces produced with inclined loading plates, eccentric axial forces application, follower load, as well as attempts to individually apply/model muscle forces in-vitro [3, 4, 15-25]. Follower load has emerged as the most accepted method and is now commonly used in many investigations of multi-level cervical spine biomechanics. The advantage of the follower load methodology is that by aligning the loading cables with the center of rotation the specimen is able to support physiologic levels of axial compressive preload while preserving the pure moment assumption.

As referenced in Aim 1, Miura et al. 2002[21] and DiAngelo and Foley 2004[15] published articles directly aimed at determining the most appropriate loading mechanism to produce physiologic motion patterns. Miura et al. presented pure moment testing combined with follower load and through adjusting moment targets was able to achieve ~20% agreement with segmental range of motion reported in literature, however typical segmental motion patterns were not observed with this technique[21]. DiAngelo and Foley utilized an eccentric axial compressive method, violating the pure moment assumption, in attempt to mimic the weight of the head. DiAngelo and Foley were able to show reasonable agreement with the segmental motion patterns, but the magnitudes dramatically underestimated the average in-vivo segmental kinematics[15] due to instability issues with the axial loading method. Authors have also investigated the effect of including simulated muscle loading on kinetics (intradiscal pressure and load displacement curve) and shown some agreement/improvement but the in-vivo kinetic data available for comparison is very limited making these assessments difficult[22, 26].

1.4.2 Status and Limitations of Current In-vivo Testing Methodologies

Numerous methodologies of measurement of in-vivo motion have been utilized resulting in valuable information about cervical spine structure and function. However, not until recently has it been possible to make these measurements with any degree of fidelity.

Currently available methodologies include: measurement of spine movement using skin-based measurement systems, single plane radiographs, Video-fluoroscopy (a popular alternative to static radiographs), roentgen stereophotogrammetric analysis (RSA), tracking of bone pins inserted into L3 and L4 vertebrae[27], reconstructions of static computed tomography (CT), and 3D measurements obtained by hand digitizing biplane static radiographs [28, 29]. Each of these methods comes with associated advantages and disadvantages related to ease of collection, accuracy, level of detail, etc. but in the end none of them are capable of recording 3D, dynamic, muscle driven, in-vivo kinematics.

The biplane x-ray system available in the Biodynamics lab addresses the aforementioned limitations and is capable of recording 3D, dynamic, muscle driven, in-vivo kinematics[30]. However, the biplane x-ray system is not capable of determining kinetic information and therefore cannot be used independently to understand the mechanical factors related to maintenance of spine health.

1.5 OVERVIEW

This study relies of on the availability of accurate 3D, dynamic, muscle driven, in-vivo, kinematics and a testing system capable of reproducing the kinematic in-vitro. Access to in-vivo kinematics of unprecedented precision (average precision of 0.44 mm in translation and 1.1 degrees in rotation[31]) is the driving force behind the proposed methods, allowing for an iterative experimentation/modeling approach that would not otherwise be possible. It is also imperative that the in-vitro testing apparatus is sufficiently reproducible yet flexible in configuration. Direct integration/comparison of the two technologies has potential to revolutionize the field of spine biomechanics.

1.5.1 Chapter 2: Cervical Spine Functional Spinal Unit Efficacy

The robot/UFS testing system (previously described [32, 33]) is potentially ideal for this application because the system has been shown to be able to achieve a pure moment through active minimization of off-axis forces and moments using a hybrid control algorithm. However, the robot/UFS testing system has only been previously described for testing of lumbar spine functional spinal units (FSU) without any compressive preload. Therefore, prior to direct comparison of the in-vivo and in-vitro segmental kinematics it was necessary to develop the methodology and validate the efficacy of performing multi-level cervical spine biomechanical testing in the presence of physiologic axial compressive preloads. In chapter 2.0 , a study is described wherein the robot/UFS testing system is used to investigate cervical spine FSUs and the hybrid control algorithm is compared with displacement control.

1.5.2 Chapters 3 & 4: Multi-Segment Testing with Follower Load

The capabilities of the system are then extended to multi-segment testing in the presence of follower compressive loading. Follower load was chosen because it upholds the pure moment assumption and is commonly described in literature. Despite its common usage for investigation of surgical instrumentation, very little work has been performed to explore the effects of follower load on cervical spine kinetics. Therefore, chapter 3.0 and chapter 4.0 validate the efficacy of integrating follower loading with a robot/UFS testing system and report the resultant kinetics (Secondary Aim, Section 1.1.3). Successful development and validation was necessary prior to exploration of the objective of this project (chapter 5.0 and chapter 6.0).

1.5.3 Chapter 5: Comparison of In-Vitro and In-Vivo Kinematics

In chapter 5.0 , a systematic comparison of standard pure moment with no compressive loading versus published and novel compressive loading techniques (follower load - FL, axial load - AL, and combined load - CL) is performed. It is hypothesized (hypothesis H1a) that follower load will preserve the pure moment assumption and not alter the segmental contributions; therefore, the effect of applying a follower compressive load on the segmental contributions is compared to pure moment testing with no compressive load. Conversely, axial load violates the pure moment assumption and it is hypothesized (hypothesis H1b) that the effect of applying an axial compressive load will have a non-uniform influence on the segmental contribution compared to the no compression state. Finally, it is hypothesized that combining follower load and axial load will have a synergistic effect enabling (+/- 10%) agreement with in-vivo segmental kinematics.

1.5.4 Chapter 6: Optimization of Compressive Loading Parameters

A model for dynamically altering the compressive loading throughout the motion path is explored in chapter 6.0 through optimization desirability function throughout the extension-flexion motion path. It is hypothesized (hypothesis H2a) that dynamic alteration of the compressive load magnitude will enable segmental temporal sequencing of vertebral movement to be adjusted throughout the motion path resulting in (+/- 10%) agreement with in-vivo segmental kinematics.

1.6 OBJECTIVE

The objective of this project is to identify and verify the appropriate in-vitro loading conditions that would replicate the in-vivo kinematics and kinetics of the cervical spine, with the overall goal of improving the biofidelity of the experimental platform.

2.0 IN-VITRO CERVICAL SPINE TESTING USING A ROBOT/UFS TESTING SYSTEM [34]

2.1 INTRODUCTION

In-vitro biomechanical testing the human cadaver cervical spine is widely used as a repeatable platform to quantify three dimensional motion of the spine in response to loads. Traditionally, kinetic parameters of the spine have been obtained through biomechanical tests based on either the flexibility method (load control) or the stiffness method (displacement control) [35]. However, the inherent limitations of the two leading control algorithms have been detailed in the long-standing controversy in spine biomechanical testing, "load control vs. displacement control" [36].

In displacement control experiments, displacements are applied and the resulting loads are measured [37, 38]. In load control experiments, loads (i.e., forces and moments) are applied individually [39, 40] or in combination[41] to the free end of a spinal specimen and the resulting unconstrained three-dimensional displacements (i.e., translations and rotations) are measured. From a control perspective, it is apparent that displacement control is less appropriate than load control in high stiffness regions such as the "elastic zone" (EZ) where small changes in applied displacement can produce large changes in load. For example, large, "unphysiologic" coupled loads can result from displacement control tests when rotational displacements are prescribed about a fixed axis that is not the specimen’s preferred axis of rotation [42]. On the other hand, load control is less appropriate than displacement control in low stiffness regions of the load-displacement curve such as the “neutral zone” (NZ) where little change in applied load can produce large changes in displacement. For example, when closed-loop load control tests are performed, low stiffness of a specimen within the NZ puts high demand on the response characteristics of the control system, for efficient minimization, requiring the testing machine to respond to load control commands with large displacement steps potentially resulting in overshoot of the load targets [43]. Thus, the two leading control algorithms for in-vitro spine biomechanical testing—load control and displacement control— are both limited in their lack of adaptation to changes in the non-linear load-displacement response of a spine specimen—pointing to the need for a sufficiently sophisticated control algorithm that is able to perform a flexibility test by governing the application of loads/motions to a spine specimen in an adaptive manner.

Goel et al. describe the requirements of flexibility tests, emphasizing the necessity of applying pure moments to specimens permitted to move in an unconstrained manner [20]. Experimental designs that comply with these expectations are typically composed of adaptive loading mechanisms such as pulleys and cables, orthogonal stepper motors mounted on linear bearings, robotic arms, and Stewart platforms [44, 45]. The unconstrained path is a function of the testing system’s ability to maintain a pure moment and to permit the natural motion path of the passive subsystem throughout a spinal segment’s range of motion (ROM) [4]. If such testing protocols are properly executed, then motion should be uninhibited along and about each orthogonal axis. As a corollary, non-primary moments and forces should be minimal.

An alternative method to achieve the specified flexibility test lies in traditional robotic hybrid control. Hybrid control methods, combining aspects of load control and displacement control, are readily found in the classical robotics literature [46]. Industrial robots are inherently displacement controlled devices and are designed for relatively simple pick-and-place operations such as spot-welding, spray painting, etc. However, in recent years robots have begun to be utilized increasingly for assembly tasks such as part-mating which requires high precision manipulation. Hybrid control gives the manipulator the ability to measure and respond to contact forces—extending the effective precision of a manipulation and enabling precise control despite the uncertainties and variations of the work environment. Previously, hybrid control methods adapted from literature have been successfully applied to the multi-DOF (degree-of-freedom) biomechanical testing of musculoskeletal joints such as the knee using a robot/UFS (universal force-moment sensor) testing system [47-50]—suggesting the possibility that hybrid control approaches might be appropriate for the spine as well.

In the present paper, the alternative hybrid control method was directly compared to traditional displacement control using a robot/UFS testing system to test intact cervical motion segments operating under both methods. This study design was chosen in order to confirm or deny the efficacy of utilizing an inherently displacement controlled robotic manipulator for the highly uncertain task of spine flexibility testing. The results were also analyzed in an effort to corroborate the emerging opinions regarding flexibility and stiffness testing and to directly delineate the differences between hybrid control and displacement control.

2.2 MATERIALS AND METHODS

2.2.1 Device Description

The experimental platform consisted of specimen, robotic manipulator, and robotic controller. The serial linkage robotic manipulator (Staubli RX90, Staubli Inc., Duncan, SC) was equipped with an on-board six-axis load cell (UFS Model 90M38A-150, JR3 Inc., Woodland, CA) and custom specimen-mounting fixtures (Figure 1A). Clinical pedicle screws (three per vertebra) were used to secure spinal specimens within the mounting fixtures. Upon insertion the pedicle screw was manually tested for rigidity and augmented with bone cement if necessary to ensure sufficient fixation was achieved. Following testing the rigidity of the pedicle screw fixation was also manually confirmed to ensure loosening did not occur during the testing procedure.

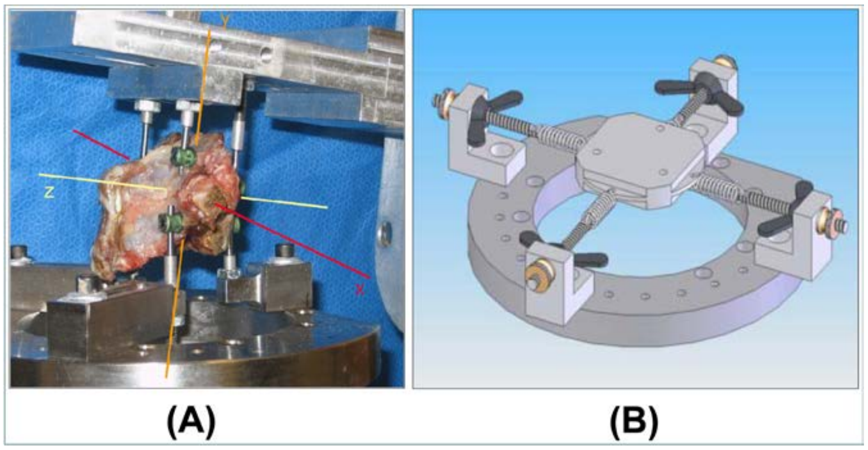


Figure 1. (A) Human cervical functional spinal unit (FSU), (B) Rigid-body-spring model

The robot was controlled via a custom-built PC-based control program written in MATLAB software (Appendix A.1). Prior to in-vitro cadaveric testing, preliminary testing of the

PC-based controller was performed using a rigid-body-spring model (Figure 1B) which was custom designed to mimic the stiffness and range of motion of a spinal segment. The hybrid control algorithm uploaded onto the PC-based controller contained an iterative “displacement control” loop (Appendix A.1.1) with embedded “load control” loop (Appendix A.1.2) to minimize undesired coupled forces/moments induced by motions applied during displacement control (Figure 2).

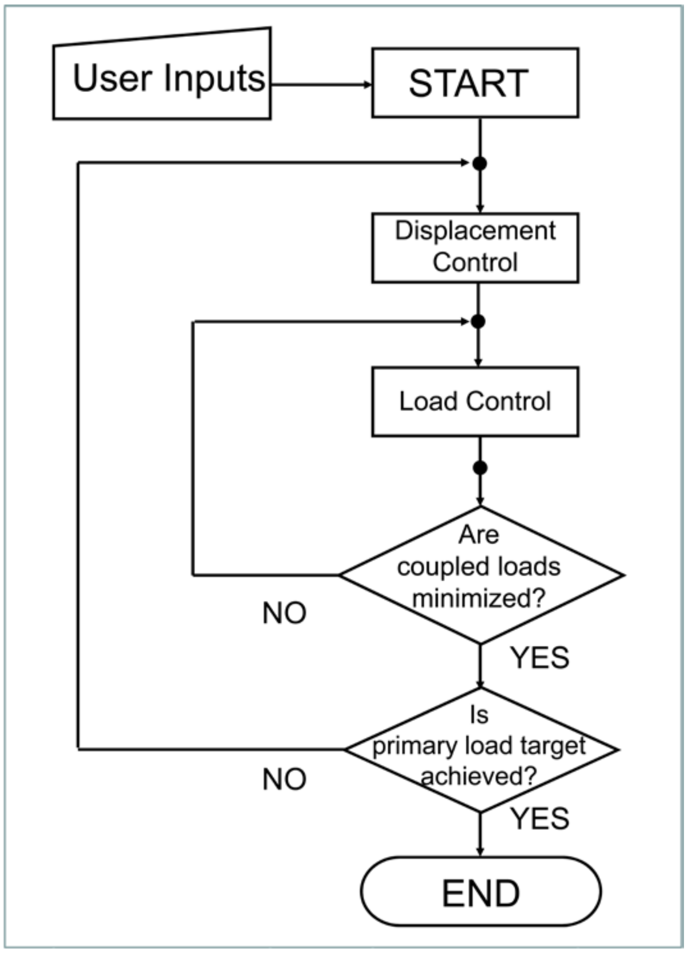


Figure 2. Flow chart of hybrid control algorithm

2.2.2 Development of Hybrid Control Algorithm

Two different displacement control modules (basic displacement control–BDC and adaptive displacement control–ADC) and three different load control modules (stiffness-based, PID-proportional/integral/derivative, and fuzzy logic) were implemented and compared (Figure 3).

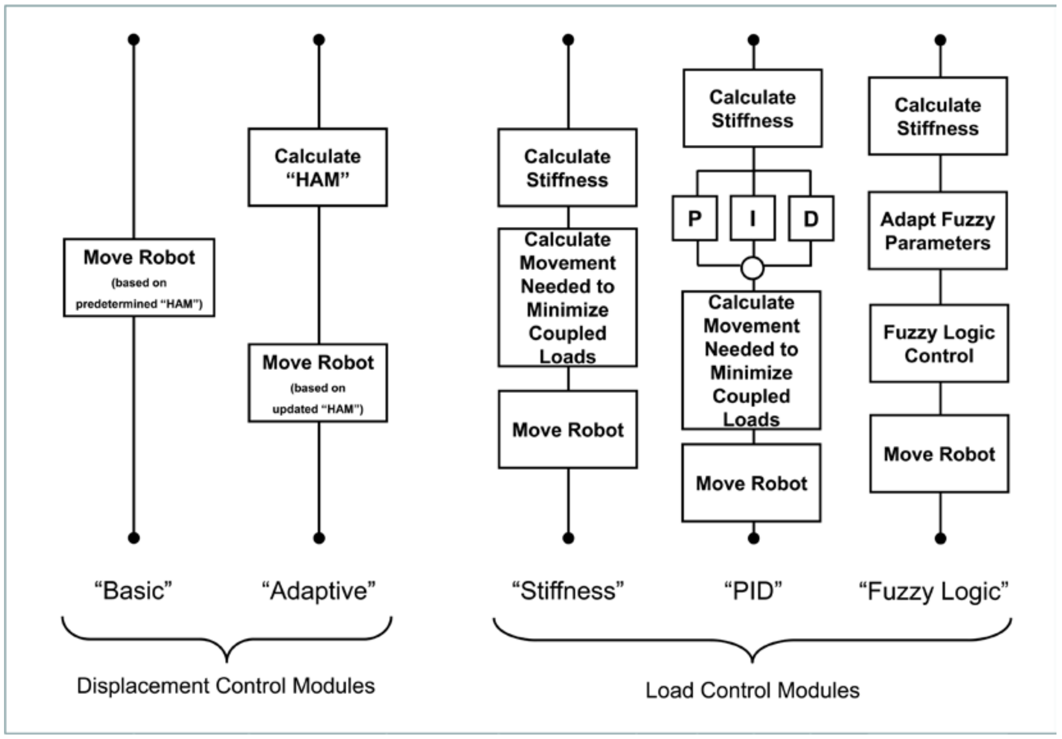


Figure 3. Flow chart of BDC and ADC modules

The stiffness-based load control algorithm minimized coupled forces and moments using data from previous steps, calculating stiffness and inverting the diagonal stiffness to find the displacements needed to minimize the coupled loads. To avoid overshoot and oscillations, step-size limitations were imposed. The PID controller used the same concept, but regulated the error output to systematically step back to a force-minimized position in small increments. The basic concept of fuzzy logic load control is that the “states” of the system are used to derive an output. In the present application, the difference in the measured loads and targeted loads (“error”), and the rate of change of the “error” were used to prescribe movements of the robot end-effector to minimize the coupled forces/moments. Generally, a number of iterations within the fuzzy logic load control module were required to minimize the coupled loads within an acceptable range. Comparison of the performance of the three load control modules was performed by experiments with the physical model, examining the ability of each module to minimize undesired coupled loads without “overshoot”. Direct comparison of the three load control modules was performed in the superior/inferior degree of freedom with a consistent starting load of -33 N and the resulting minimization response was recorded and plotted versus the iteration step number.

The BDC module instructed the robot to incrementally rotate the superior vertebra of the specimen about a user-specified axis of rotation (AOR). The location of the AOR was selected based on the mean location of the instantaneous axis of rotation of typical cervical spine motion segments reported in literature [51]. This was accomplished by creating a local coordinate system aligned with the specimens anatomy with the origin defined as a point on the midline at the posterior 1/3 of the vertebral body’s depth in the anterior posterior direction and superior edge of the inferior vertebral body (Figure 1A). This axis was kept unchanged throughout the test. For the ADC module the user-specified AOR was also utilized for the first displacement step—which usually resulted in undesired coupled forces/moments to be minimized by the embedded load control loop as described above. After the incremental rotation step the fuzzy logic load control module was implemented to minimize off axis forces and moments below the .5 N and .25Nm threshold respectively (no compressive loading). Additionally, after load minimization, the actual axis of rotation was calculated based on the current and previous spinal position, replacing the initial user-specified axis for the next displacement control iteration. In this way, the ADC algorithm was able to adapt to the specimen's preferred axis of rotation.

2.2.3 Comparison of BDC and "Hybrid Control"

For direct comparison of BDC and hybrid control, six fresh frozen human cervical specimens were cleaned of musculature, producing osteoligamentous structures that were sectioned into functional spinal units (FSUs: C4-C5 (n=6) and C6-C7 (n=6)). Each FSU (n=12) was subjected to extension-flexion bending (FE), lateral bending (LB) and axial rotation (AR) testing for each control method. The specimens were preconditioned with three cycles of BDC loading and the BDC and hybrid control tests were performed during the fourth or fifth cycles in a pairwise randomized order. All testing, including preconditioning, was performed to a load target of ±2.5 Nm with 10N compressive preload [25]. BDC was achieved on the robot/UFS testing system by silencing the load control inner loop and rotating about a rigid (non-updating) axis until the load target was achieved. Conversely, ADC and fuzzy logic load control were combined to perform hybrid control, wherein the helical axis of motion (HAM) was calculated and all off-axis forces and moments were minimized at each step in the quasi-static movement. In BDC the system operated at a rate of 0.35 ± 0.0053 degrees/second and whereas the active minimization process of the hybrid control algorithm decreased the rate to 0.067 ± 0.0014 degrees/second. The testing rates for BDC and hybrid control were consistent for all degrees of freedom.

The quality of motion is described by off-axis load measurements and the HAM. Off-axis forces and moments were quantified as root mean squared error from minimization target (zero for all non-primary axis except fy=10N) throughout the motion path. A single (overall) HAM was calculated based on the (+/-) maximum rotation positions for each motion path. The change in HAM was also quantified as the change in orientation of the overall HAM and the point through which the HAM passed though the primary plane of motion. The quantity of motion is displayed as the range of motion (ROM) and the width of the NZ exhibited for each primary motion. The width of the NZ was determined using the method presented by Smit et al., wherein the boundaries of the NZ were determined based on the second derivative of a double sigmoidal curve fit to the moment vs. rotation[52, 53]. Results were summarized and expressed in bar graphs representing median ± standard deviations and a non-parametric Wilcoxon signed-rank test for paired design was used to determine significance (*$p<0.05$).

2.3 RESULTS

2.3.1 Hybrid Control Algorithm Performance

Comparison of the performance of the three load control modules by experiments with the rigid-body-spring model showed that the fuzzy logic controller minimized undesired coupled loads without the characteristic "overshoot" of the stiffness controller (Figure 4). The response of the PID and the fuzzy logic controllers were very similar, although the PID controller did exhibit minimal overshoot. The slight overshoot exhibited by the PID controller could have been reduced/eliminated with further tuning, however the fuzzy logic controller offers the practical advantage of approximate logic (as opposed to exact) enabling it to adapt to the specimen's stiffness without the need for specimen-specific tuning. However, a potential drawback of the fuzzy logic controller (and also the PID controller) was the increased number of iterations needed to minimize the forces; the stiffness-based load control module achieved force minimization in approximately half the number of iterations. Considering the relative advantages and disadvantages of the three load control systems the fuzzy logic load control module was selected and used for all subsequent testing in this study.

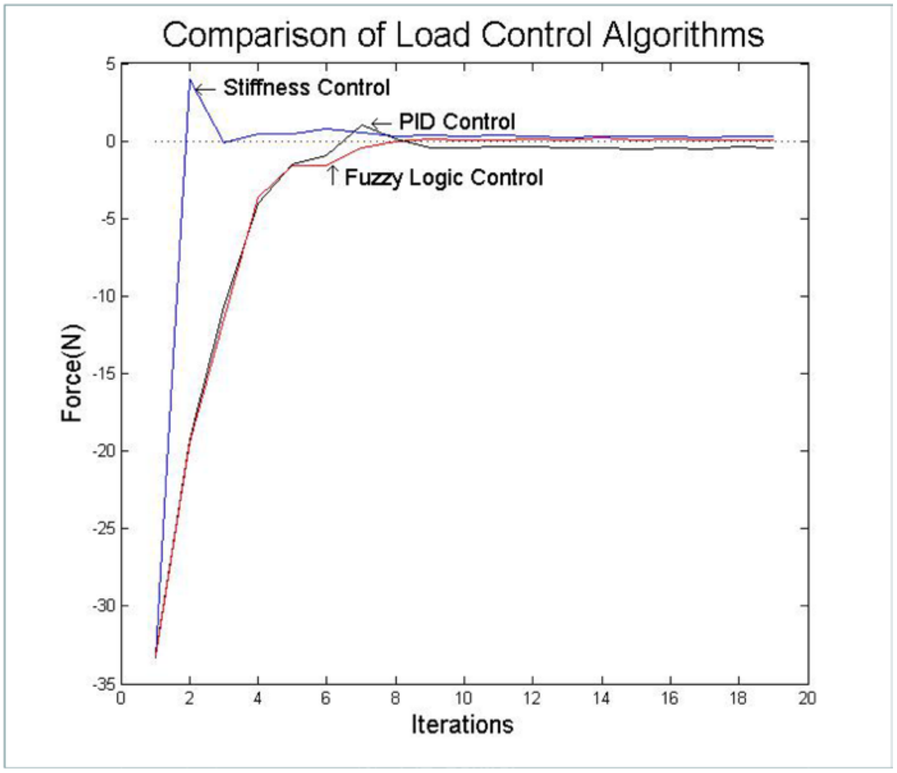


Figure 4. Comparison of load control algorithms

Comparison of the performance of the BDC and ADC modules using the rigid-body-spring model demonstrated that the ADC module automatically adapted to the specimen's natural axis of motion—as evidenced by the smaller coupled forces to be minimized by the ADC after the initial move (Figure 5). The BDC module failed to adapt—hence each applied rotation repeatedly produced a characteristic, large force spike.

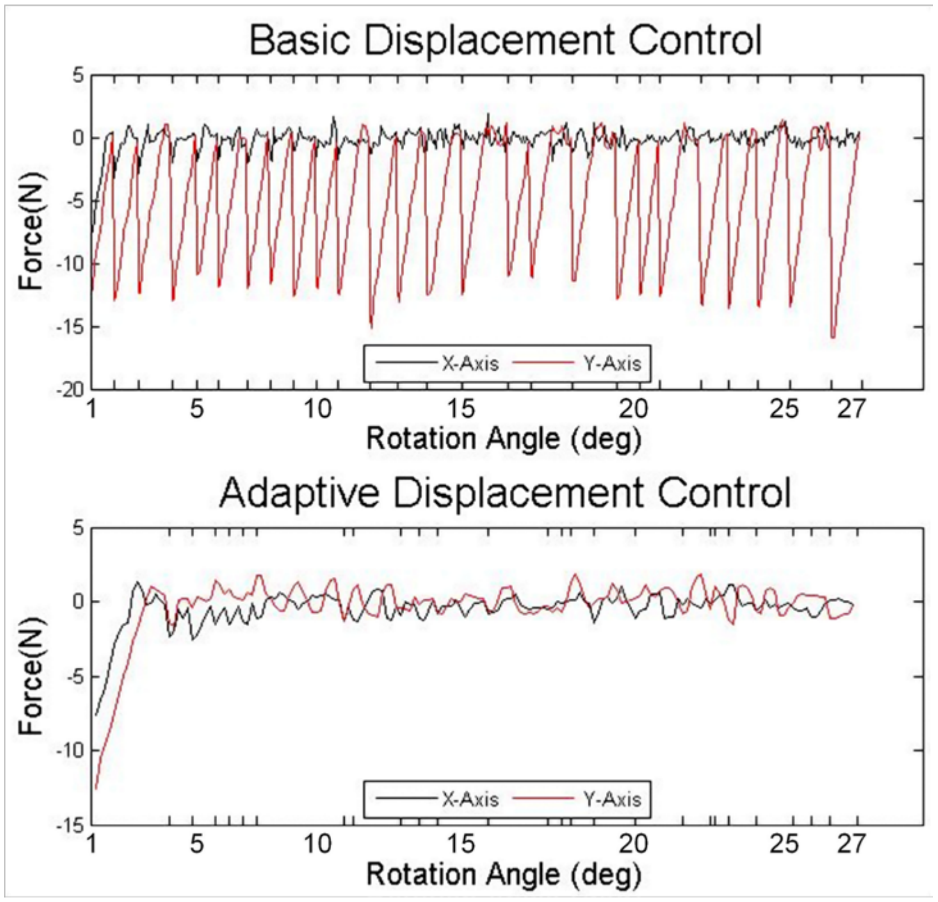


Figure 5. Performance comparison of the BDC and ADC modules

2.3.2 Comparison of BDC and "Hybrid Control"

The hybrid control algorithm (with ADC module and fuzzy logic load control module) enabled the robot/UFS testing system to apply pure moments to an FSU in extension-flexion, lateral bending, or axial rotation in an unconstrained manner through active control of secondary translational/rotational DOF—successfully minimizing coupled forces/moments. These parameters can be observed qualitatively in Figure 6 wherein representative raw data from one FE experiment is displayed and are quantified in Figure 7. The characteristic nonlinear S-shaped curves of the primary moment-rotation responses were consistent with previous reports of the FSU having a region of low stiffness (NZ) bounded by regions of increasing stiffness (EZ).

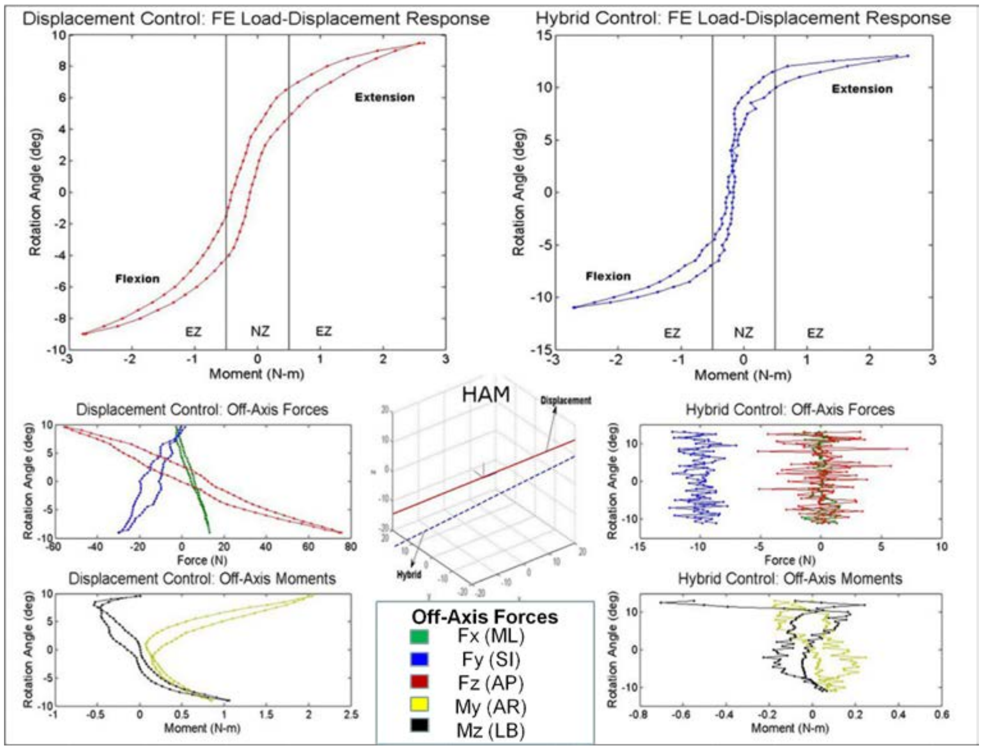


Figure 6. Representative Raw Data (n=1)

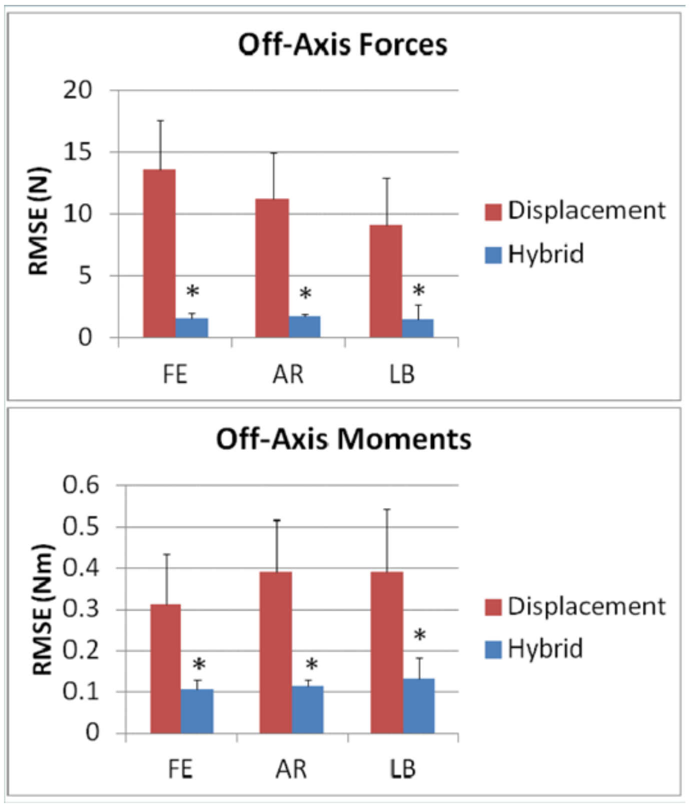


Figure 7. Off-axis forces and moments

The adaptation of the hybrid control system to the specimen specific parameters can be seen in Figure 8 wherein the HAM shifts by an average of 6.30 ± 4.23mm and 12.82 ± 10.65 degrees as opposed to BDC in which the axes were fixed. Also of note is that the smallest change in axes was observed during FE (5.73 ± 3.00mm / 4.91 ± 3.40 degrees) and the largest change was observed during AR (7.70 ± 4.06mm / 22.21 ± 7.34 degrees).

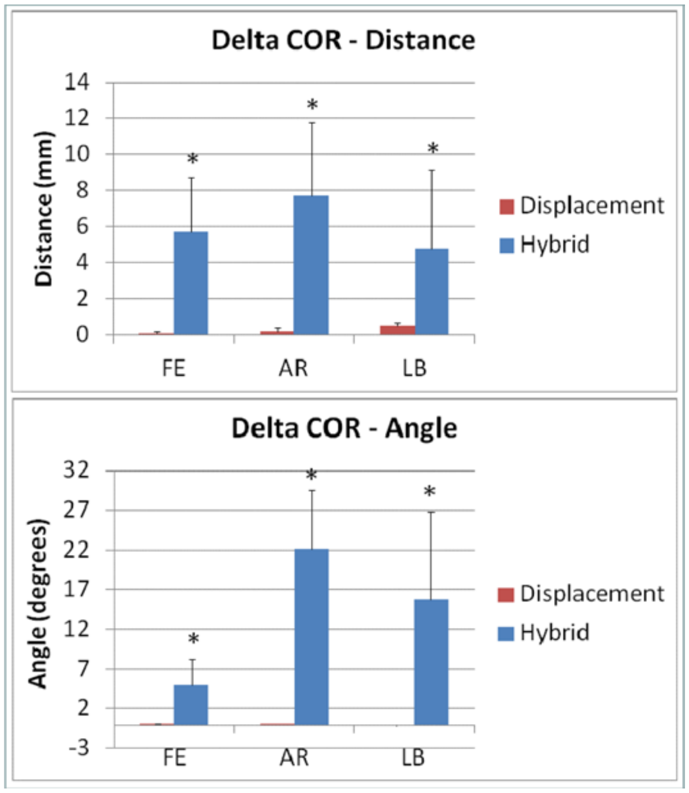


Figure 8. Change in center of rotation

To assess the quantity of motion, the ROM data for each primary motion from each specimen were averaged together. Median rotation extrema for BDC and hybrid tests are compared in Figure 9. For all primary rotations, hybrid control ROM (FE = 18.17 ± 6.11 degrees, AR = 14.52 ± 6.78 degrees, LB = 10.77 ± 4.28 degrees) was significantly larger than BDC ROM (FE = 17.50 ± 6.20 degrees, AR = 10.25 ± 4.19 degrees, LB = 9.00 ± 3.46 degrees). Similar results were found for the width of the NZ, with hybrid control resulting in a wider NZ than BDC for both AR (hybrid = 5.38 ± 3.41 degrees, BDC = 3.25 ± 1.42 degrees) and LB (hybrid = 4.90 ± 2.19 degrees, BDC = 4.00 ± 1.71 degrees) however no difference was detected for FE (hybrid = 6.93 ± 4.06 degrees, BDC = 7.25 ± 2.77 degrees).

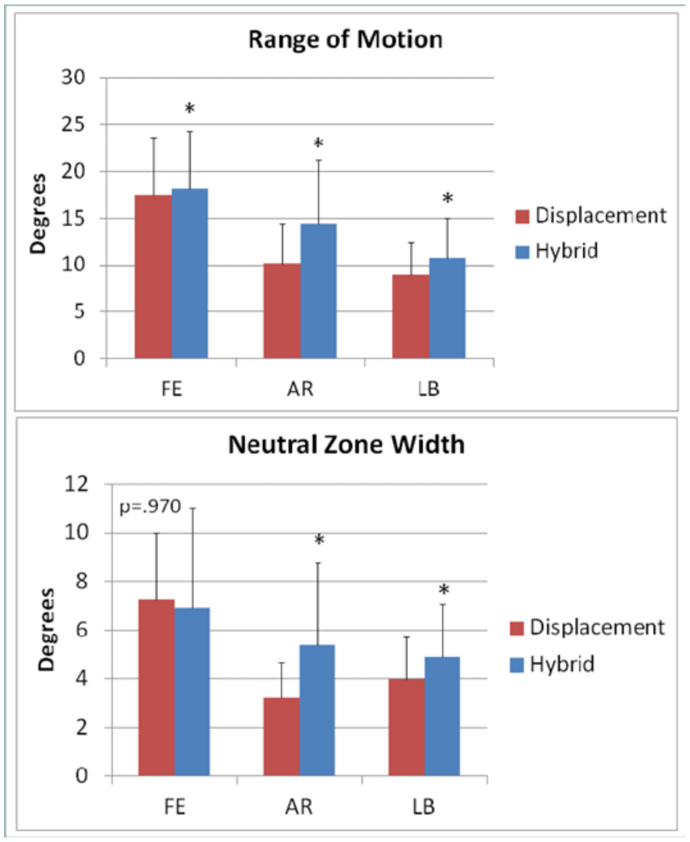


Figure 9. Range of motion (ROM) and width of the neutral zone (NZ)

2.4 DISCUSSION

In the present study, biomechanical testing of human cervical FSUs using a robot/UFS testing system in a hybrid control mode successfully elicited nonlinear kinetic behavior from the specimens similar to that reported by others in studies using traditional control methods. In particular, the characteristic “S-shaped” curves of the moment-rotation response obtained in this study were consistent with previous reports of the FSU having a region of low stiffness (NZ) bounded by regions of increasing stiffness (EZ) [35]. Notably, coupled forces and moments were successfully minimized by the hybrid control algorithm, thus the hybrid control algorithm was able to successfully perform a “flexibility test”—previously the province of load control approaches. Notwithstanding the observation that the hybrid control algorithm of the present study successfully minimized coupled loads without “overshoot”, future direct comparison would be needed to determine whether the hybrid control flexibility test is superior in performance to a traditional “closed-loop” load control flexibility test.

The present study offers a direct comparison between “hybrid control” and “displacement control”. The high magnitude of the off-axis loads in BDC confirm Panjabi’s caveat regarding the non-physiologic and injurious loads that may result from an improper, rigid AOR estimation. This is confirmed in the HAM plots which show that the orientation and the location of the HAM changes significantly when the systems actively adjusts to determine the specimen’s preferred motion path. Minimization of off-axis loads and differences in the HAM present in hybrid control suggest that the hybrid control’s larger ROM and larger/more defined NZ results from actively seeking the specimen’s preferred motion path and updating its AOR. A limitation of this study is that the loading rates were not consistent between the BDC and the hybrid control algorithm. The fact that the hybrid control algorithm operated at a slower testing rate could have contributed to the increased ROM resulting from this algorithm due to the effects of viscoelastic creep. However, it should also be noted that the ROM data of the hybrid control falls into the range of reported ROM from flexibility tests of subaxial cervical motion [15, 21].

One of the primary advantages and future directions of the robot/UFS testing system presented herein is the control systems ability to store the exact kinematics determined during the hybrid control mode. The stored kinematics can be used to drive the displacement control module (with the load control module silenced). Replaying the exact kinematics for the specimen in the intact state would result in a unique situation wherein hybrid control and displacement control are functionally equivalent. A broad implication of this implementation (together with the above observation that the hybrid control algorithm was able to perform a flexibility test which is usually only associated with load control), is that equivalent spinal function may be attainable by the neural control system using different control strategies—in other words, normal spinal functioning may not depend upon a single control scheme. Additionally, replaying the exact kinematics for the specimen in an altered state would enable determination of the removed structures in-situ forces based on the principle of superposition [49, 50, 54]. Replaying exact kinematics in an altered stated [55] is similar in concept to the hybrid method presented by Panjabi [4] and should be explored further in future work. Some of the potential criticisms/barriers of this future work with the methodology as presented here are the quasi-static nature (0.067 ± 0.0014 degrees/second) of the hybrid algorithm which could affect the determined kinetics due to the viscoelastic properties of the biologic material as well as the limited stiffness of the robot end-effector which may result in end-effector deformation /altered kinetics in response to the cut [56].

The single FSU model with 10N of axial compressive load was chosen for this study in order to minimize the variability and focus on the control algorithm rather than the specimen characteristics. However, these characteristics are also a limitation of this study in regards to the current trends in literature [20, 25], wherein most studies have included the entire subaxial cervical spine and many include follower loading in order to simulate physiologic compression [16, 57]. In- house studies (unpublished) are ongoing which utilize the methodology presented here for full cervical spine testing in the presence of follower loading.

Although the data and analysis presented in this study lends support to the hybrid control algorithm over BDC, it cannot be concluded that either control method is more or less physiologic as data was not directly compared to in-vivo kinematics. Additionally, the current work did not include neural and muscular contributions. Therefore, future work should compare the in-vitro testing methodologies to in-vivo kinematics as well as attempt to include neural and/or muscular contributions so as to more thoroughly assess the efficacy of the hybrid control algorithm.

3.0 FOLLOWER LOAD: MOMENT-ROTATION PARAMETERS

3.1 BACKGROUND

In-vivo experimentation and modeling has shown that the spine is able withstand very high levels of load while maintaining stability. However, in-vitro experiments performed by numerous groups have shown that cadaveric specimen tested outside of the body buckle at loads far below the loads experienced in-vivo. Panjabi et al. reported the cervical spine critical (buckling) load as approximately 11 N[58]. As a result most studies investigating multi-segment cervical spine have not included a compressive preload[59].

Patwardhan et al. developed the “follower load” (FL) method of applying compressive preload to a multi-segment lumbar spine specimen without buckling[57, 60], and adapted follower load for application to the cervical spine[16]. The follower load concept was based on mathematical modeling of muscle activation in the spine that established that the internal force vector of the spine runs tangent to the curvature of the spine through segmental centers of rotation[57]. Patwardhan et al. (2000) demonstrated that follower load application significantly increased the stability and reduced the flexibility of the cervical spine when compared to the hypermobility observed in response to a compressive vertical load[16]. Stability was defined based on the change in C2 sagittal tilt (lordosis angle) and flexibility was determined by performing a regression analysis of the C2 sagittal tilt as a function of the compressive loading magnitudes.

An alternative methodology for assessing a specimen’s flexibility and stability is through rotational flexibility testing and associated analyses of the moment-rotation curves. Typical parameters extracted from moment-rotation curves include range of motion (ROM), neutral zone width, neutral zone stiffness and hysteresis. Investigation of the effect of compressive preload on the moment-rotation parameters has been explored primarily in lumbar spine function spinal units, wherein physiological axial compressive preloads has been shown to increase neutral zone stiffness and hysteresis[19, 41, 61-63]. However, variability in testing methods and inconsistent definitions of the moment-rotation parameters have led to conflicting results that shed little insight into the influence of follower preload on cervical spine biomechanics. Very little data exists on the effect of follower load on intact cervical spine moment-rotation parameters—with only a few publications reporting range of motion (ROM) changes resulting from follower load application[2, 64]. Therefore, the goal of this study is to implement a novel actively controlled follower load system within a robot/UFS testing system and utilize this system to explore the effect of follower load on multi-segment cervical spine moment-rotation parameters.

3.2 METHODS

3.2.1 Protocol

N = 12 fresh-frozen human (C3-C7) cervical cadaveric specimen (Mean Age=51.8+/-7.3) were pre-screened with computer tomography (CT) scan, dissected preserving osteoligamentous structures, and mounted in a robot/UFS testing system (Figure 10). The robot/UFS testing system consists of a serial linkage robotic manipulator (Staubli RX90, Staubli Inc., Duncan, SC) with an on-board six-axis load cell (UFS Model 90M38A-150, JR3 Inc., Woodland, CA) and custom specimen-mounting fixtures[32, 33, 55]. Four clinical lateral mass screws are used to secure the specimens to the mounting fixtures (one in each pedicle and two in the anterior portion of the vertebral body). After mounting, specimens were wrapped in 0.9% saline soaked gauze and periodically sprayed with saline in order to prevent dehydration of the specimen. The robot was controlled via a custom-built PC-based control program written in MATLAB software. The hybrid control algorithm uploaded onto the PC-based controller contained an iterative displacement control loop with embedded load control loop to minimize undesired coupled forces/moments and was controlled to a pure moment target of 2.0 Nm for flexion and extension (FE). Due to the quasi-static nature of the hybrid control algorithm the system operated at a rate of 0.067 ± 0.0014 degrees/second. Two consecutive full extension-flexion loops were performed with the data from the second cycle being presented to account for preconditioning[19].

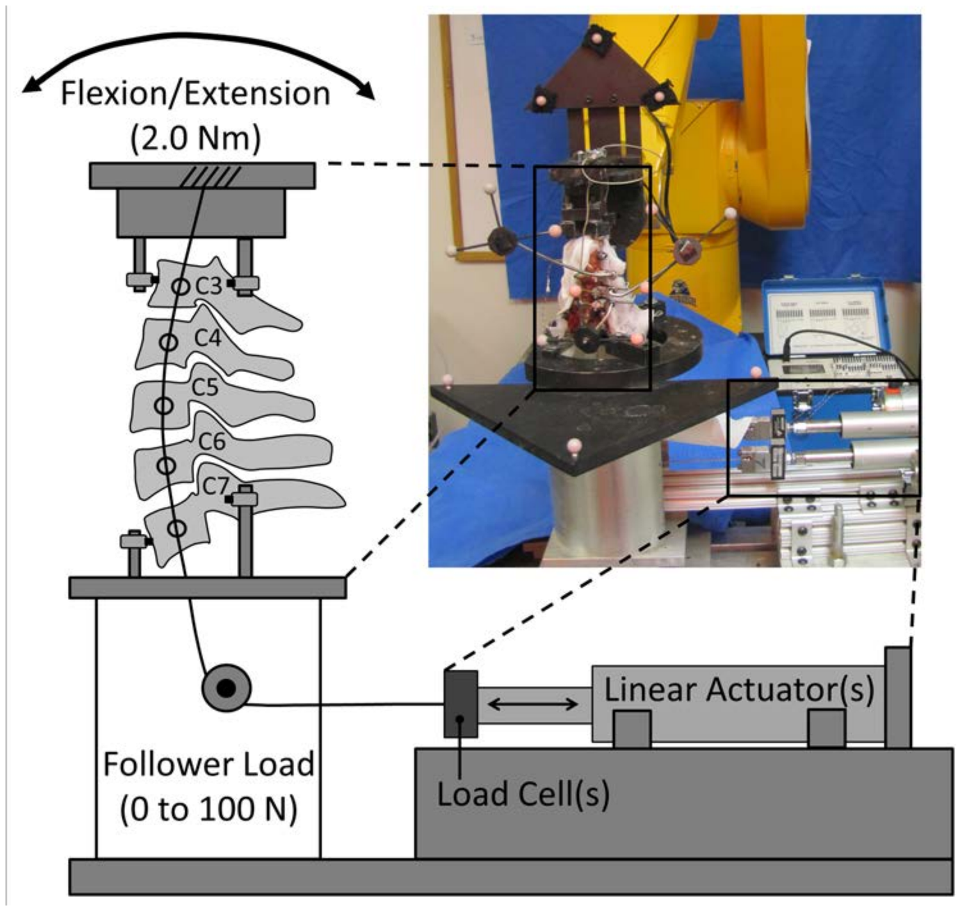


Figure 10. Follower load set-up diagram and image

3.2.2 Follower Load

Follower load application was accomplished by loading the specimen with bilateral cables passing through cable guides inserted into the vertebral bodies and over pulleys attached to the base. Traditionally, follower load is accomplished by load application using pulleys and hanging weights [16]. However, a novel active system was implemented in our laboratory using linear actuators coupled with load cells. Control of the system was integrated with the custom-built PC-based control program written in MATLAB currently used to control the robot/UFS testing system and enabled active control of the loading throughout the motion path (Appendix A.1.3).

The follower load system (Figure 10) consists of two independently controlled 24V servo motor linear actuators (Ultramotion- 3-B.125-DC426_24-4-/4) and compression/tension load cells (Transducer Techniques – MLP-100). A Galil Motion Controller (DMC-4183-BOX8(-16BIT)-D3040-D4040) controlled this system using on-board closed loop Proportional-Integral-Derivative (PID) load control. The ball shank end of a nylon coated 3/64" diameter stainless steel wire rope lanyard was inserted into a mounting plate for the compression/tension load cell. The cable was threaded through a custom designed adjustable cable guide system to enable the follower load cable to interface with the specimen in a manner consistent with the design criterion: a) tangent to the curvature of the spine and b) pass through the specimen's center of rotation (COR). The adjustable cable guide system consisted of a 4-40 stainless steel threaded rod which was bent into a "U" shape and attached to the specimen using clinical lateral mass screws inserted into the pedicles. To accommodate the wire rope a 6-32 ball joint rod end was attached to either end of the 4-40 threaded rod allowing for anterior/posterior adjustment of the follower load path. Optimization of the follower load path to align with the specimen's COR was accomplished through an offline iterative feedback process using the moment output of the testing systems on-board six –axis load cell. With the specimen in the neutral position, 100N of follower load was applied to the specimen and resulting change in moments was recorded. Follower load magnitude of 100N was chosen as it is representative of the most common follower load magnitude presented in literature [64-70]. The position of the cable guide was then adjusted to counteract the moment change and the process was repeated until less than 0.1Nm change in moment was observed. Preliminary testing of the described optimization process was performed ensuring that the application of the follower load did not significantly alter the unloaded curvature of the spine [63].

3.2.3 Mechanical Outcomes

Extension-flexion range of motion (ROM) was defined as the rotational deflection difference between the maximum applied loads in each direction (Appendix A.2.1).

NZ parameters were determined by fitting a double sigmoidal function (Equation 1) to moment-rotation data to define the NZ as the high compliance region demarcated by extrema of the second derivative as described by Smit et al.[52] (Appendix A.2.2).

Equation 1. Formula for bi-sigmoidal curve

$$D = \frac{1}{1 + e^{-(a1+b1*L)}} * c1 + \frac{1}{1 + e^{-(a2+b2*L)}} * c2 + d$$

NZ width was defined as the difference in load between the function's inflection points. NZ stiffness was defined as the inverse of the slope of a linear fit of the function in the NZ. To investigate the difference in loading and unloading, all parameters were determined for the flexion to extension as well as the extension to flexion directions.

Elastic zone stiffness was determined for the loading curves by calculating inverse of the slope of a linear fit to the last five points (~1.5-2.0 Nm) of the flexion and extension loading curves.

Hysteresis (energy dissipation) was defined as the area between the flexion to extension and the extension to flexion curves and was determined by subtracting the integral of the double sigmoidal function (above) for each curve (Appendix A.2.4).

3.2.4 Statistical Analysis

Results were summarized and expressed in bar graphs representing mean ± 95% confidence interval and a non-parametric Wilcoxon signed-rank test for paired design (no compression and follower load) was used to determine significance (*$p<0.05$).

3.3 RESULTS

Figure 11A shows a representative scatter plot (n=1) of the moment-rotation curve fit with the double sigmoidal function. For all experiments the double sigmoidal function provided an excellent fit to the measured moment-rotation curve with average no compression $r^2 = 0.999$ and follower load $r^2 = 0.998$. No differences were observed between the flexion to extension and extension to flexion moment-rotation curves for ROM or NZ parameters as determined from the calculation of the first (Figure 11B) and second derivatives (Figure 11C) to automatically demarcate the NZ (Figure 11D). Therefore, ROM and NZ parameter data from the flexion to extension and extension to flexion moment-rotation curves were averaged together for all remaining comparisons.

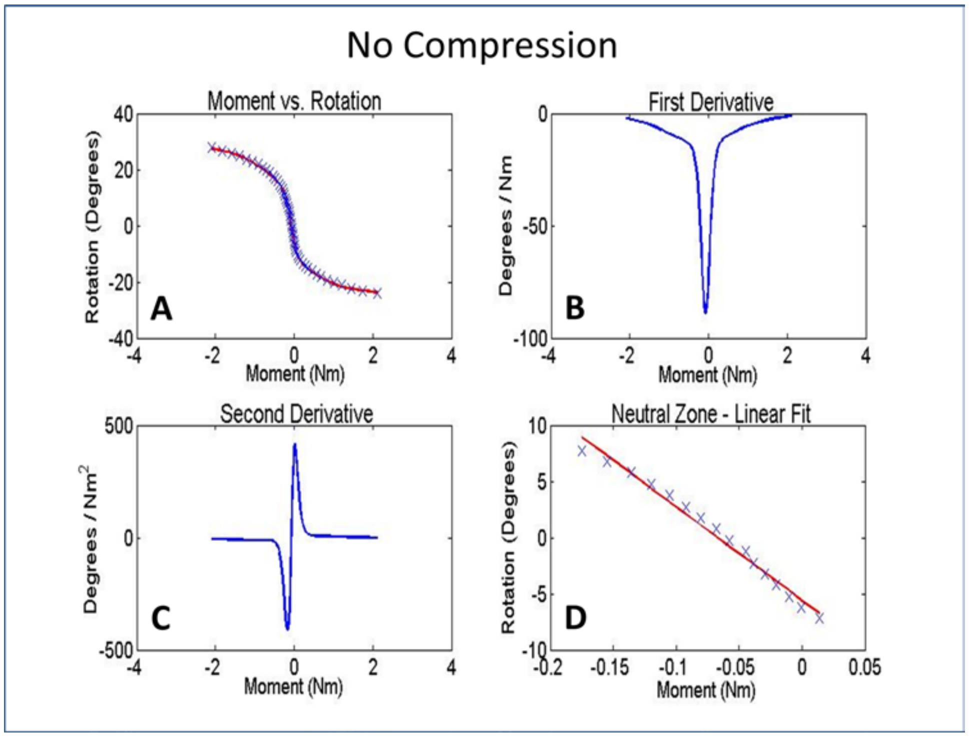


Figure 11. Representative image of neutral zone parameters calculation

Application of follower load had no significant effect on the ROM (Figure 12A, Table 1) at 2.0 Nm. However, the width of the neutral zone (Figure 12B, Table 1) was significantly increased with the application of follower load.

Table 1. Summary of effect of follower load on moment-rotation parameters

	No Compression	Follower Load
Range of Motion (^{o})	48.162 ± 6.848	48.525 ± 7.482
NZ Width - Rotation (^{o})	14.495 ± 2.688	19.842 ± 5.009
NZ Stiffness (Nm/^{o})	-0.016 ± 0.004	-0.030 ± 0.008
EZ Stiffness - Flexion (Nm/^{o})	-0.218 ± 0.022	-0.184 ± 0.022
EZ Stiffness - Extension (Nm/^{o})	-0.215 ± 0.024	-0.209 ± 0.021
Hysteresis (NM*^{o})	9.309 ± 1.188	15.613 ± 3.928

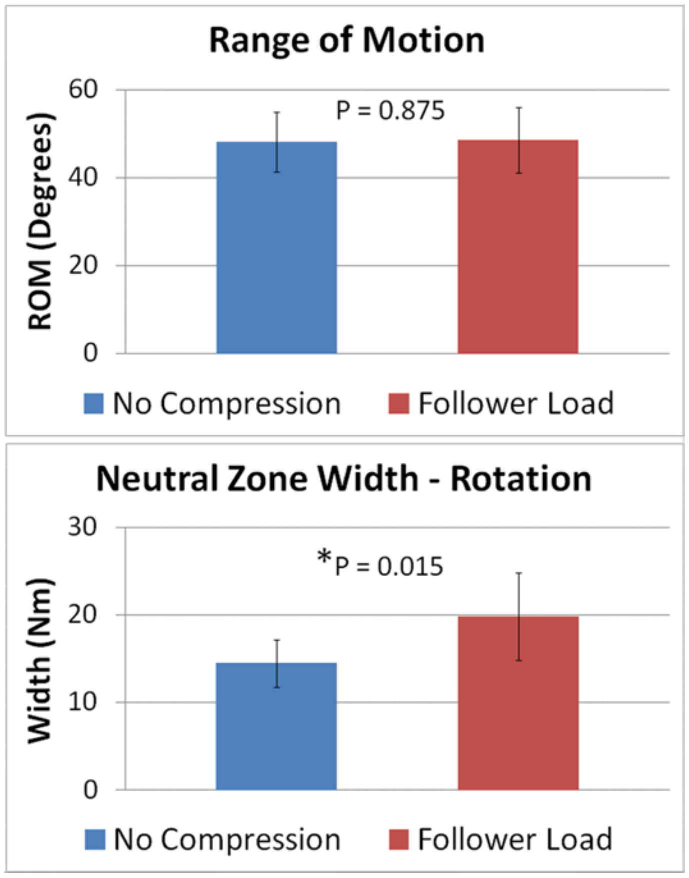


Figure 12. Follower load influence on ROM and NZ width

Similarly, application of follower load also significantly increased the specimen's stiffness within the neutral zone (Figure 13A). However, the elastic zone stiffness was not significantly change with the application of follower load, although a trend of decreased elastic zone stiffness was observed in flexion.

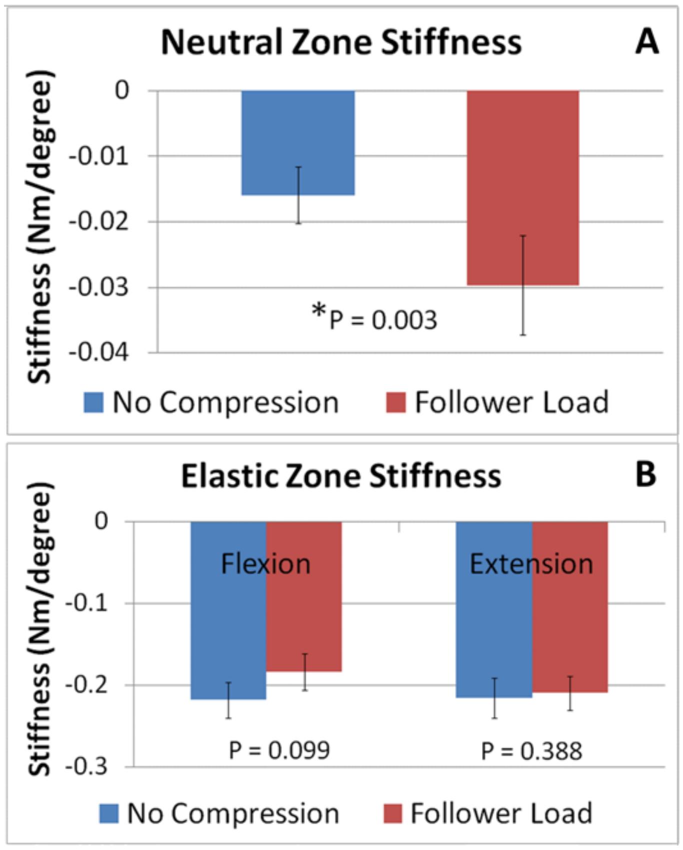


Figure 13. Follower load influence on NZ and EZ stiffness

Figure 14A depicts a representative scatter plot (n=1) showing area bounded by flexion-extension and extension-flexion moment-rotation curves fit with double sigmoidal function. Integration of double sigmoidal fit was performed for quantification of hysteresis (energy dissipation) (Figure 14B). Application of follower load resulted in a significant increase in the hysteresis observed (Figure 14C).

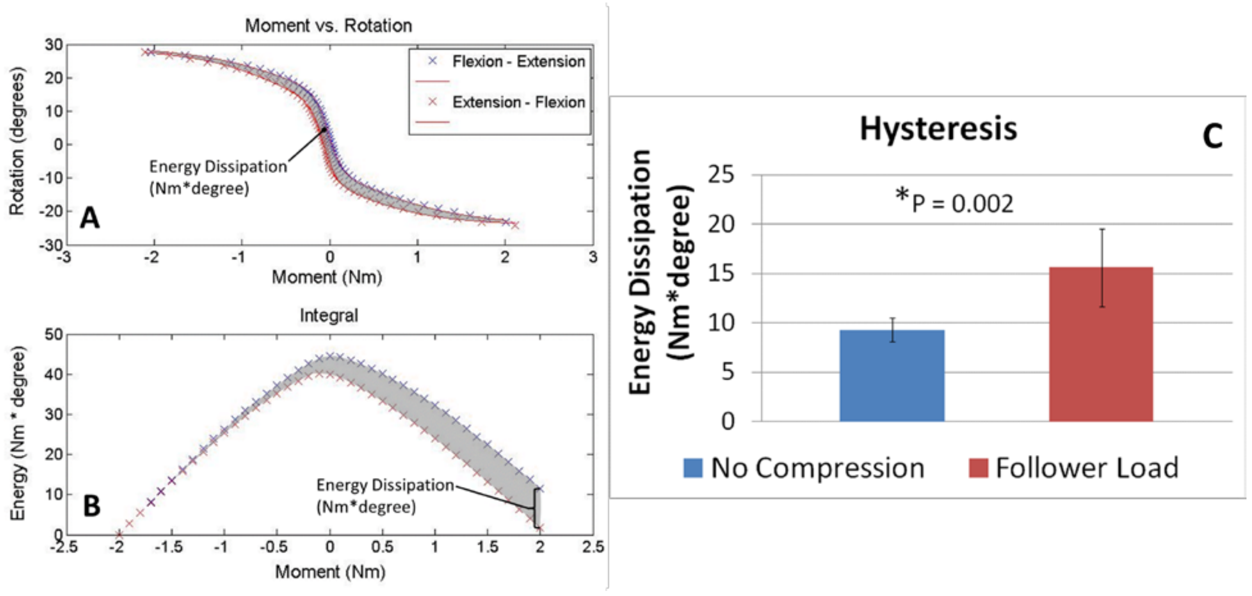


Figure 14. Representative image and bar graph of hysteresis

3.4 DISCUSSION

This is the first study to investigate the influence of follower load application on the moment-rotation parameters in a multi-segment cervical spine. Application of follower load did not appear to effect the quantity of motion (ROM) but did effect the quality (shape of the curve). The width of the neutral zone, the stiffness of the neutral zone and hysteresis were all increased with the application of follower load—consistent with an increased stability of the joint.

Comparable data has been reported Patwardhan et al.[63] in a multi-segment lumbar spine study, wherein it was reported that the neutral zone stiffness significatly increased with increasing levels of follower load. Tawackoli et al.[71] also showed example moment-rotation curves in their study invstigating the influence of axial compressive preload on the thoracolumbar spine. Tawackoli observed, based on the shapes of the curves (not quantified), that the neutral zone width decreased or vanished, the neutral zone stiffness increased and the hysteresis loops were altered with increased compresive load.

The only known study specificly designed to investigate the influence of preload magnitudes of cervical biomechanics is a 2-D finite element study of a functional spinal unit published by Ng et al.[59]. Ng applied 0,50,100, and 150N of preload to C5-C6 whiler rotating the model to 1.8Nm of extension-flexion and showed slight (~4%) increased ROM with 100N preload and (~8%) with 150N preload. The small change in ROM observed with 100N of follower load is consistent with the rusults of the present study.

Although numerous authors are now using follower load to apply compressive loading during cervical spine insturmentation testing[2, 64-70], there is little baseline multisegment cervical spine moment-rotation data available for comparison with these findings. Paxinos, et al.[64] published a paper investigating the effectiveness of a wedge graft and lock plate for stabilizing an anterior cervical fusion, wherein they loaded n=8 C3-C7 cervical spine specimen with 150N follower load to a max moment of 1.5Nm. They reported extension-flexion ROM of C5-C6 with no preload (13.3 +/- 4.2 degrees) was not significantly changed with the application of follower load (13.7 +/- 4.1 degrees). The effect of follower load on range of motion was also reported by Puttlitz et al.[2] in an investigation of cervical spine disc replacemnt kinetics. Puttlitz showed a slight (<5%) decrease in in extension-flexion ROM at C4-C5 with the appplication of 44N of follower load. It should be noted that Puttlitz also reported the effective of follower load application of lateral bending and axial rotation and showed larger decrease, although this decrease was still not significant.

It is important to highlight the main limitations of the current study. First, only extension-flexion was investigated. This study design was chosen due to the controvery regarding accurately applying follower load in the lateral bending and axial rotation motions[19, 72-74]. Also, there is discussion in literature regarding proper optimization of the follower load path. Optimization performed in the neutral position has been shown to result in greater loss of ROM than flexed position optimization in the lumbar spine. In this study a novel moment-based follower load optimization was performed in the neutral position, however no loss in ROM was observed.

Despite the sparcity of directly compariable data, in general the application of compressive preload appeared to have little to no influence of the multi-segment range of motion in the cervical spine, which is consistent to the findings of the present study. However, this appears to be contraditory to comparable studies in the lumbar spine. Patwardhan et al.[63] showed a 25% reduction in ROM with application of 1200N of follower load. However, it should be noted that Patwardhan also identified difference between a follower load optimized in the neutral posture compared to a follower load optimized in a flexed posture and demonstrated that reduction in ROM could be decreased to 15% with flexed posture optimization. The noted difference suggests the possibility of an apparatus related artifact[19]. Tawackoli et al.[71] also explored the influence of follower load on multi-segment ROM in human thoracolumbar specimen (T9-L3). Tawackoli also showed ~25% decrease in ROM when follower load exceeded 500N and resolved that future flexibility testing in the lumbar spine apply a minimum of 500N of follower loading.

In addition to extracting ROM from the moment-rotation curve the present study looked at the neutral zone parameters (neutral zone width and neutral zone stiffness) based on the definition presented by Smit et al.[52] and also extended that method to calculate hyseresis. There have been numereous, sometime conflicting, definitions of the neutral zone presented in literature[52, 75-78]. The neutral zone was originally defined by Panjabi[79] as the region on either side of the neutral position where there is little to no resistance to motion. Functionally, Panjabi's calculation of the neutral zone was based on the residual discplacement after the removal of a quasi-static load. Similarly, Wilke et al. defined the neutral zone as the angulation difference at zero load in both directions of motion, a defintion that applicable to dynamic testing[77]. Both the definition by Panjabi and Wilke are based on the loading history of the specimen and are associated with the viscoelastic propeties of the intevertebral disc[52]. Alternatively, Sarver and Elliott[78] and Thompson et al.[76] presented definitions based on the shape of the moment-rotation curve. Similarly, in the definition by Smit et al., a double sigmoidal function is fit to the shape of the curve—providing an objective mothod that is not based on abitrary testing parameters and resulted in an excellent fit for all extension-flexion moment-rotation curves in this study.

The influence of compressive preload on neutral zone parameters and hysteresis has also been more thouroughly explored in lumbar spine motion segments. In a study investigation a new defintion of the neutral zone, used in the present study, Smit et al.[52] showed that the width of the NZ decreased, hysteresis increased and NZ stiffness was unchannged after seven hours of axial compression. However, it is important to point out that width of the neutral zone actually increased if calculated based on the method presented by Wilke et al. and that the specimen were not compressed during testing, just for the seven hours prior to testing. Garner-Morse and Stokes[80] published a paper using a 6-degrees-of-freedom (DOF) hexapod robot and showed that physiological axial compressive preload ranging to 400N increased neutral zone stiffness, linearity and hysteresis. A strong correlation was also observed between hysteresis and neutral zone stiffness for all DOF under all testing conditions. Gardner-Morse comprehensively explored all 6-DOF, however their study was limited to porcine functional spinal units. Additionally, Edwards et al.[41], Janevic et al.[61], and Wilke et al.[81] all showed increased sagittal plane stiffness in the neutral zone with the application preload. The finding that the elastic zone stiffness was not significalty effected by the application of follower load is consistent with the literature[19], although this data is not often presented.

Summarizing the finding in the lumbar spine overall it appears that the neutral zone width, neutral zone stiffness and hysteresis all tend in increase with the application of follower load. These findings are in agreement with the results presented in this study and are consistent with the concept of follower acting to stabilize the cervical spine, proposed in the original publication presenting the concept of follower load application for the cervical spine. Proposed physiologic mechanisms for this stiffening include facet contact[82] and disc pressurization[61]. While evidence is scarce for its effect in cervical spine segments, follower load appears to have a similar stiffening effect in cervical spine segments absent a consistent reduction in ROM. It is reasonable to infer similar mechanisms underlie this stiffening. In the present study, follower load application increased the stiffness but extended the width of the neutral zone without reducing global ROM. We surmise that disc stiffening along the cervical spine account for the increased stiffness in the low-stiffness region. Further, it is possible that follower load extends the low-stiffness region by increasing ligament laxity, delaying ligament fiber recruitment which marks the transition to the high stiffness region. This study also demonstrated increased hysteresis with follower load application. It is likely that the added energy from pure compression coupled with extension-flexion leads to greater energy dissipation.

Conceptually, follower load application approximates the role of in-vivo musculature in stabilizing the spinal column in support of axial compression. Within this framework, the present study highlights a potential benefit of musculature. Follower load broadens the region of low stiffness and confers greater stability within that low stiffness region. If musculature plays a similar role, then it may act to avoid the high loading of the high stiffness region and increase protective stability in the ROM that corresponds to most daily, functional movement of the subaxial cervical spine.

4.0 FOLLOWER LOAD: INTRADISCAL PRESSURE

4.1 BACKGROUND

Several authors have recorded in-vivo intradiscal pressure (IDP) in the lumbar spine and have demonstrated several concepts; disc pressures increase with application of compressive load, IDP is posturally dependent and that IDP depends on the degeneration grade of the disc[83-87]. While significant literature exists for the lumbar spine, Hattori et al.[88] was the only group to perform an in-vivo study investigating IDP in the cervical spine. Similarly, to the lumbar spine findings Hattori reported that IDP increased approximately 1.4 times when the subject went from the supine to seated position. IDP also generally increased with extension-flexion, with the highest pressures being observed in extension and the next highest being observed in flexion. Degeneration also affected variability and patterns observed at flexion and extension with some subjects exhibiting highest pressure in flexion with the neutral position not always being the lowest.

Given ethical conflicts and technical barriers to confirming Dr. Hattori’s in-vivo study, investigators have opted to perform in-vitro investigations of IDP. The most directly comparable study in literature to the Hattori study was published by Pospiech et al.[26, 89], wherein they simulated muscle loading on a multi-segment (C2-C7) human cervical spine model and showed an increase in load with simulated muscle loading and reported finding characteristic IDP-rotation curves. Pospiech et al. also projected a dependence on degeneration grade, but this data was not reported.

The most appropriate method of applying compressive loading to the cervical spine is still under debate in literature [15, 16, 21, 90]. As a result most studies investigating multi-segment models of the cervical spine have not included a compressive preload[59]. Further complicating this matter, in-vitro experiments performed by various groups have shown that cadaveric specimen being tested outside of the body buckle at loads far below the loads experienced in-vivo. Panjabi et al. reported the cervical spine critical (buckling) load as only 11 N[58]. However, mathematical modeling of muscle activation in the spine has shown that the internal force vector of the spine runs tangent to the curvature of the spine through segmental centers of rotation[57]. Based on this premise, Patwardhan et al. developed the "follower load" (FL) method of applying compressive preload to a multiple segment lumbar spine specimen without buckling[57], which has also been adapted and validated for the cervical spine[16]. Numerous authors are now using follower loading in the cervical spine for instrumentation testing [2, 64-70], although little data exists on the effect of follower loading on normal cervical spine biomechanics.

Therefore, the current study combined simulated muscle loading (Follower Load, FL) with a robot/UFS testing system capable to applying pure moments to a multi-segment cervical spine to more closely replicate Hattori's in-vivo methodology. Finally, in an attempt to probe the reported IDP dependence on disc degeneration the relationship between disc height, which is often used as a marker of disc degeneration, and IDP was explored.

4.2 METHODS

4.2.1 Protocol

N = 12 fresh-frozen human (C3-C7) cervical cadaveric specimen (Mean Age=51.8+/-7.3) were pre-screened with computer tomography (CT) scan, dissected (preserving osteoligamentous structures), and mounted in a robot/UFS testing system (Figure 15A). The robot/UFS testing system[32, 33, 55] consists of a serial linkage robotic manipulator (Staubli RX90, Staubli Inc., Duncan, SC) with an on-board six-axis load cell (UFS Model 90M38A-150, JR3 Inc., Woodland, CA) and custom specimen-mounting fixtures. Four clinical lateral mass screws are used to secure the specimens to the mounting fixtures (one in each pedicle and two in the anterior portion of the vertebral body). After mounting specimens were wrapped in 0.9% saline soaked gauze and periodically sprayed with saline in order to prevent dehydration of the specimen. The robot was controlled via a custom-built PC-based control program written in MATLAB software. The hybrid control algorithm uploaded onto the PC-based controller contained an iterative displacement control loop with embedded load control loop to minimize undesired coupled forces/moments and was controlled to a pure moment target of 2.0 Nm for flexion and extension (FE). Due to the quasi-static nature of the hybrid control algorithm the system operated at a rate of 0.067 ± 0.0014 degrees/second. Two consecutive full extension-flexion loops were performed with the data from the second cycle being presented to account for preconditioning [19].

4.2.2 Follower Load

Follower load application is accomplished by loading the specimen with bilateral cables passing through cable guides inserted into the vertebral bodies and over pulleys attached to the base. Traditionally, follower load is accomplished by load application using pulleys and hanging weights. However, a novel active system was implemented in our laboratory using linear actuators coupled with load cells. Control of the system was integrated with the custom-built PC-based control program written in MATLAB currently used to control the robot/UFS testing system and enabled active control of the loading throughout the motion path.

The follower load system (Figure 15A) consists of two independently controlled 24V servo motor linear actuators (Ultramotion- 3-B.125-DC426_24-4-/4) and compression/tension load cells (Transducer Techniques – MLP-100). This system was controlled with a Galil Motion Controller (DMC-4183-BOX8(-16BIT)-D3040-D4040) using on-board closed loop Proportional-Integral-Derivative (PID) load control. The ball shank end of a nylon coated 3/64" diameter stainless steel wire rope lanyard was inserted into a mounting plate for the compression/tension load cell. The cable was threaded through an adjustable custom designed cable guide system to enable the follower load cable to interface with the specimen in a manner consistent with the design criterion: a) tangent to the curvature of the spine, b) pass through the specimen's center of rotation (COR). The adjustable cable guide system consisted of a 4-40 stainless steel threaded rod which was bent into a "U" shape and attached to the specimen using clinical lateral mass screws inserted into the pedicles. To accommodate the wire rope a 6-32 ball joint rod end was attached to both ends of the 4-40 threaded rod which allowed for anterior/posterior adjustment of the follower load path. Optimization of the follower load path to align with the specimen's COR was accomplished through an offline iterative feedback process using the moment output of the testing systems on-board six –axis load cell. With the specimen in the neutral position, 100N of follower load was applied to the specimen and resulting change in moments was recorded. The position of the cable guide was then adjusted to counteract the moment change and the process was repeated until less than 0.1Nm change in moment was observed. Preliminary testing of the described optimization process was performed in order to ensure that the previously described maintenance of segmental curvature angle criteria was upheld [63].

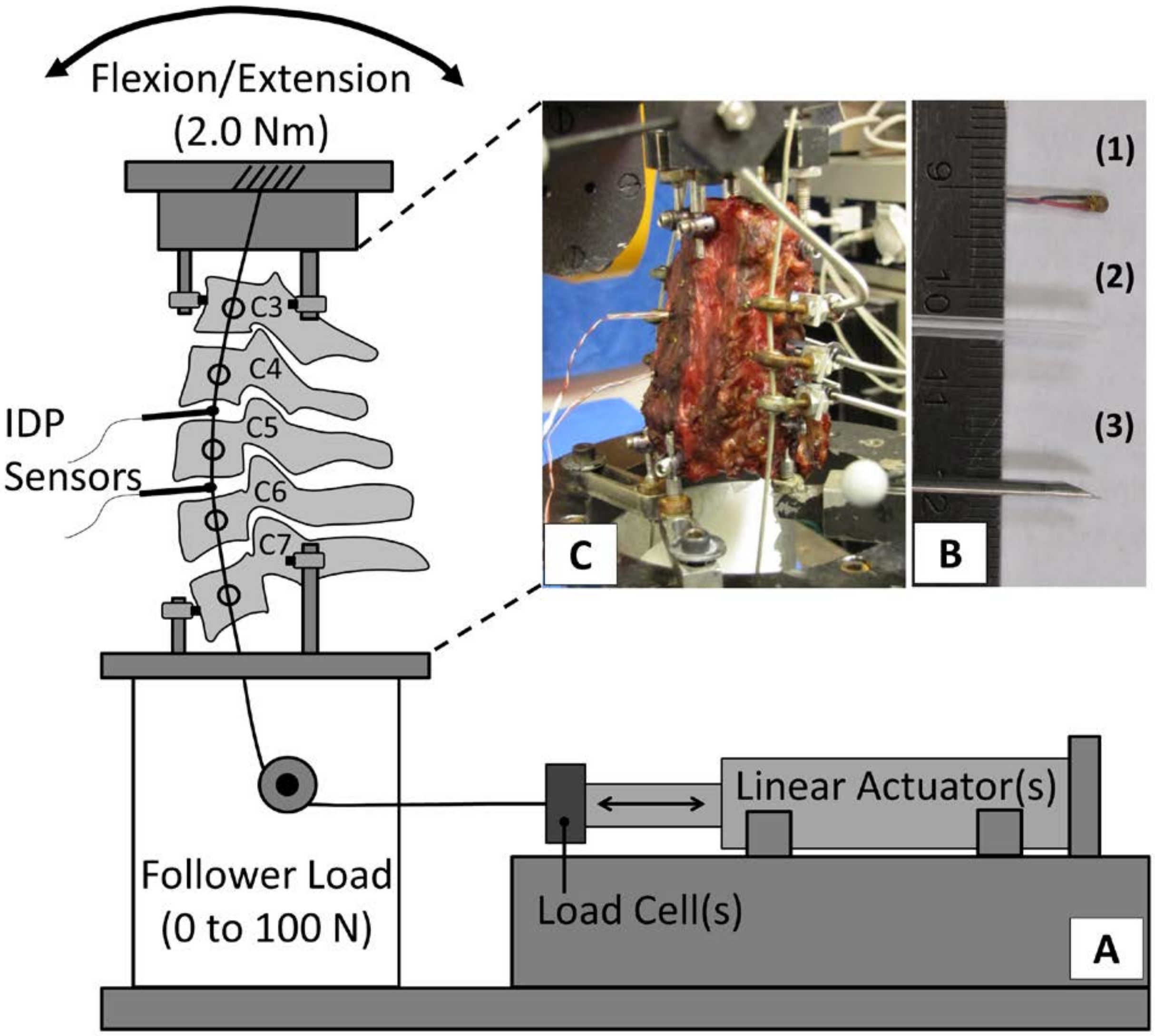


Figure 15. Schematic of testing system indicating placement of IDP sensors

4.2.3 Intradiscal Pressure

Intradiscal Pressure (IDP) was recorded with a 500 PSI miniature pressure transducer (Precision Measurement Company, Model 060) coupled with a strain indicator and recorder (Vishay, Model P3, Figure 15B-1). To prevent damage/cracking at the sensor terminal a thin plastic strip was adhered to the non-sensing side of the IDP sensor using a malleable epoxy resin (Loctite 1166731) and then inserted into an 18 gauge 2 inch long PTFE dispensing needle (Figure 15B-2). Prior to testing, the IDP sensors were calibrated in a lab controlled set-up composed of a manual hydraulic pump and analog pressure gage and reliability was determined across three trials using interclass correlation coefficient (ICC) analysis. The model 060 pressure transducer measuring 1.5mmx.3mm was inserted through a tunnel created by a 14 gauge needle (Figure 15B-3) and secured to the annulus with surgical sutures to prevent movement of the sensor (Figure 15C). IDP was recorded continuously throughout the testing procedure and IDP was analyzed at the neutral (FE moment = 0) position (IDP – neutral position), as well as the change in IDP with increasing flexion (IDP – flexion delta), and change in IDP with increasing extension (IDP – extension delta) (Figure 16) (Appendix A.4.3).

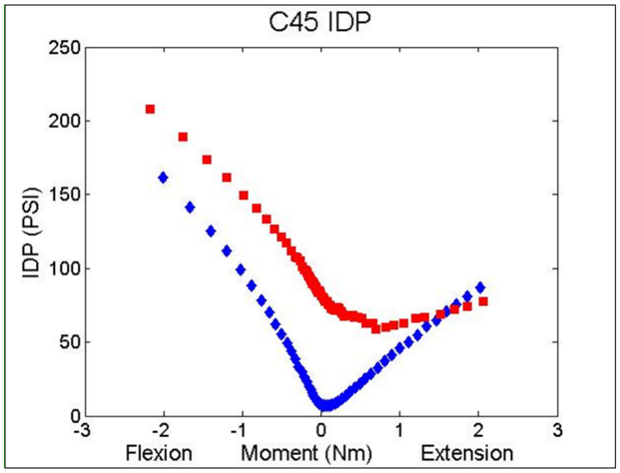


Figure 16. Representative scatter plot of C4-C5 IDP change with FE

4.2.4 Disc Height

Disc height measurements were calculated using a custom written MATLAB program based on the mid-sagittal slice of the CT scan (Lightspeed Plus 16 CT, GE Medical Systems) at a voxel size of 0.29x0.29x.625 mm). The mid-sagittal CT slice was identified and uploaded into MATLAB, converted to grayscale and the edge detection thresholding function (MATLAB, edge.m) was used to consistently display the vertebral body edges for labeling (Figure 17). The anterior, middle and posterior portions of the superior and interior bodies were identified. Disc height (DH) was calculated at the midpoint of the intervertebral disc based on the distance between the midpoints of adjacent vertebral bodies (Figure 17). Similarly, vertebral body height (VBH) was calculated as the distance between the midpoints of vertebral bodies. In order to account for the anatomical differences between specimen relative disc height (RDH) was calculated by dividing DH by the inferior adjacent VBH.

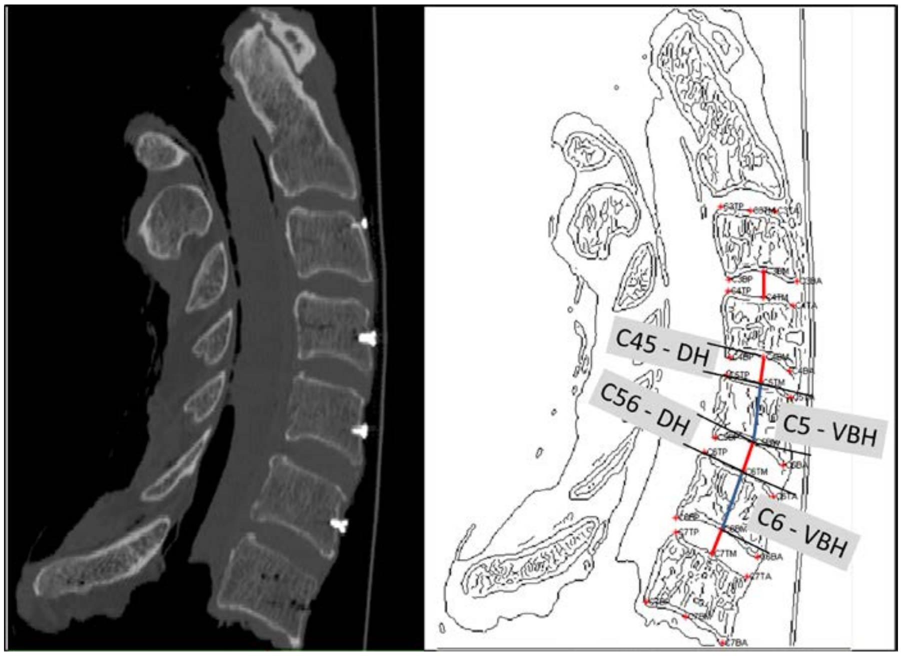


Figure 17. Representative image detailing disc height calculation from CT scan

4.2.5 Analysis

Results were summarized and expressed in bar graphs representing mean ± 95% confidence interval and a non-parametric Wilcoxon signed-rank test for paired design (no compression and follower load) was used to determine significance (*p<0.05). The correlation (Pearson) between RDH and IDP was also investigated with no compression and follower with significance set at *p<0.05. The following scale was used for interpretation of the Pearson correlation data: weak (0.0-0.2), mild (0.2-0.4), moderate (0.4-0.6), moderate/strong (0.6-0.8), and strong (0.8-1.0).

4.3 RESULTS

The three cycle reliability of the pressure transducers in the calibration set-up was determined to be ICC = 1.000 using the interclass correlation coefficient analysis (absolute agreement), indicating that he sensors are highly reliably and appropriate for use in this investigation. No difference was observed between the RDH measurements at C4-C5 and C5-C6 (Table 2) and the trends observed for the IDP outcome measures were consistent therefore, C4-C5 and C5-C6 datasets were combined (for analysis) for the purpose of investigating the correlation between RDH and IDP.

Table 2. Summary of IDP and disc height measurements

	C45			C56	
	No Compression	100N Follower Load		No Compression	100N Follower Load
IDP - Neutral Position (PSI)	16.678 ± 6.917	76.705 ± 14.347		29.498 ± 8.268	75.907 ± 14.083
IDP - Flexion Delta (PSI)	135.896 ± 38.753	146.879 ± 45.175		70.044 ± 32.130	79.551 ± 36.256
IDP - Extension Delta (PSI)	63.849 ± 34.960	35.163 ± 27.076		50.169 ± 28.035	22.464 ± 25.005
Disc Height (mm)	5.961 ± 0.606			5.725 ± 0.825	
Relative Disc Height	0.453 ± 0.050			0.442 ± 0.070	

C4-C5 IDP measured in the neutral position was significantly increased (*p=0.002) with application of follower load (Figure 18A, Table 2). This difference was also observed at C5-C6 with the neutral position IDP significantly increasing (*p=0.002). Analysis of the Pearson Correlation of RDH showed a mild correlation with IDP-neutral position without compression, but this was did not reach statistical significance (0.254, p=0.253). Application of follower load in the neutral position increased the correlation (0.369, p=0.091, Figure 18B).

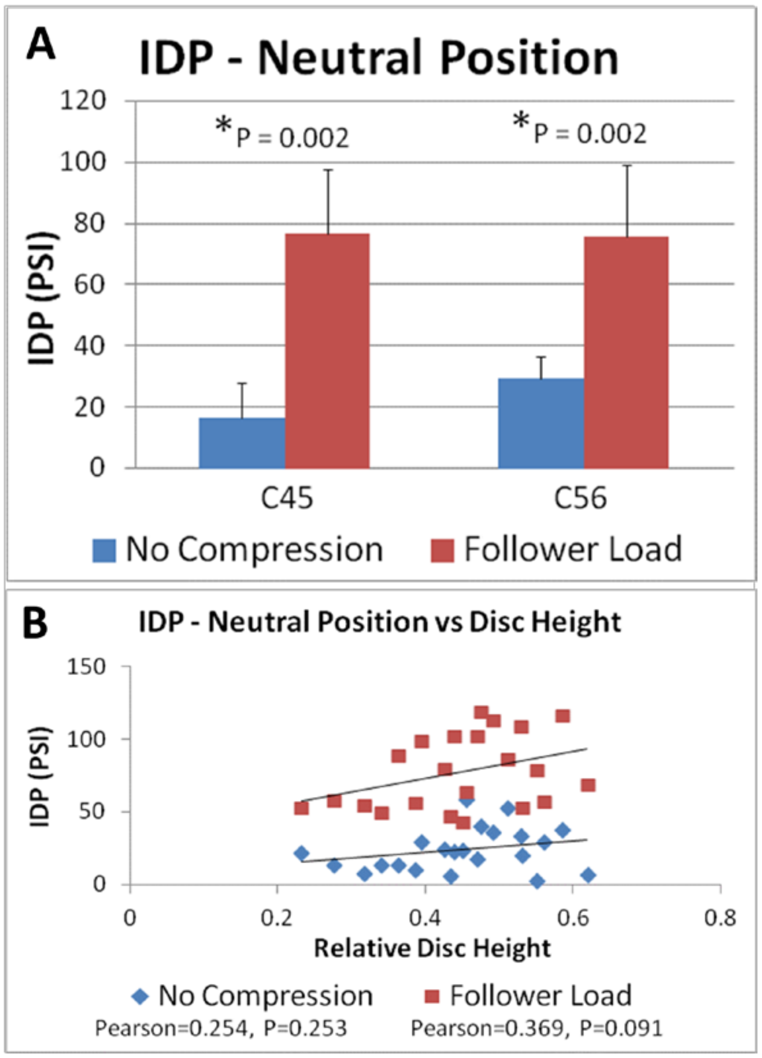


Figure 18. Change in IDP-neutral position with application of FL

The change in IDP with increasing flexion (IDP – flexion delta) was not significantly affected by the application of follower load at C4-C5 (p=0.099) or C5-C6 (p=0.638) (Figure 19A, Table 2). Analysis of the Pearson Correlation of RDH also showed a mild, not significant correlation with IDP-flexion delta with no compression (0.224, p=0.316). The correlation between IDP-flexion delta and RDH actually decreased with the application of follower load from mild to weak (0.133, p=0.556) (Figure 19B).

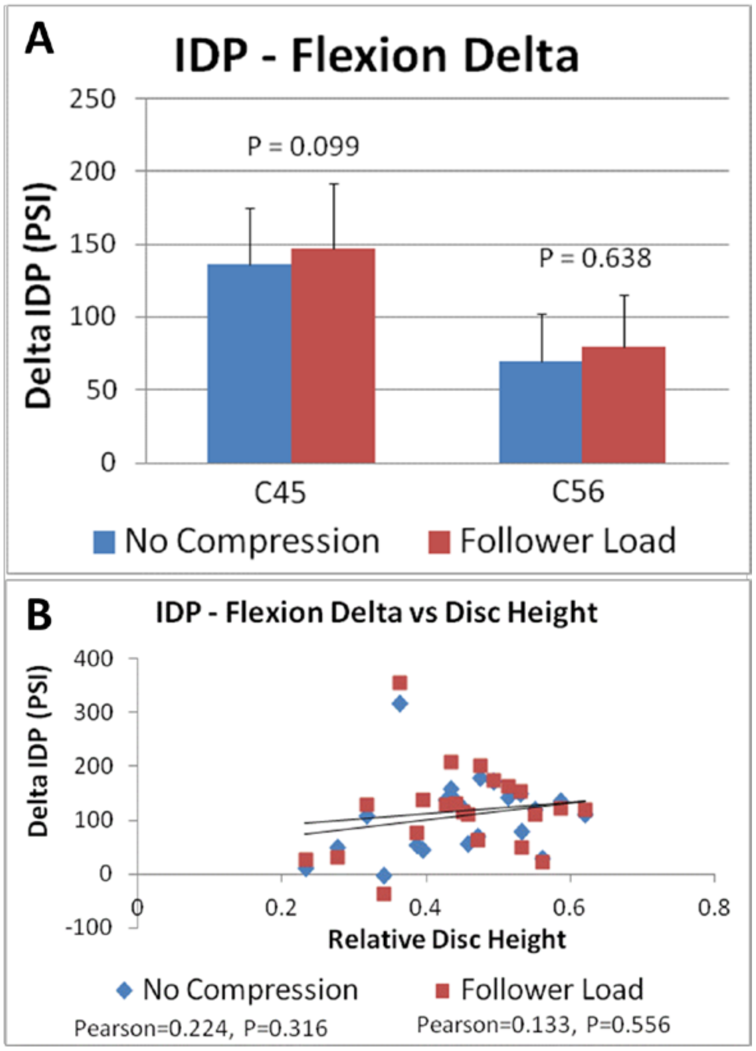


Figure 19. Change in IDP-flexion delta with application of FL

The change in IDP with increasing extension (IDP – extension delta) was significantly reduced by the application of follower load at C4-C5 (*p=0.010) and C5-C6 (*p=0.005) (Figure 20A, Table 2). Analysis of the Pearson Correlation of RDH also showed a moderate, significant correlation with IDP-extension delta with no compression (0.474, *p=0.026) and with the application of follower load (0.453,*p=0.034) (Figure 20B).

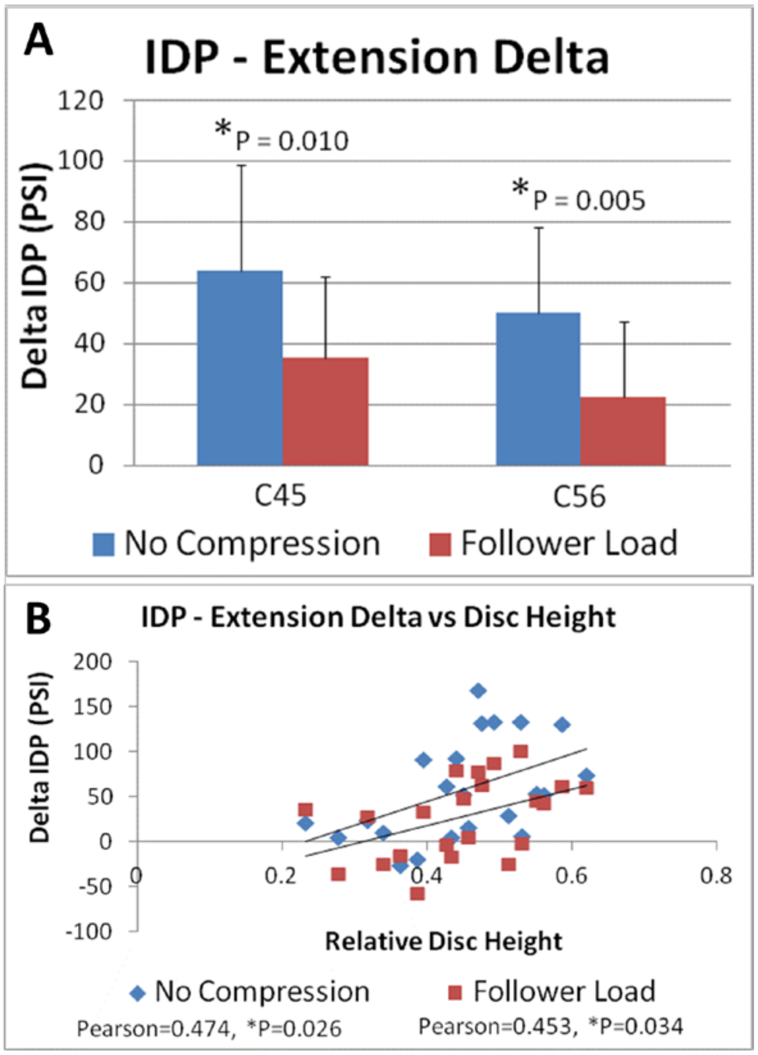


Figure 20. Change in IDP-extension delta with application of FL

4.4 DISCUSSION

The intradiscal pressure of the lumbar disk has been well studied[83-87], however limited literature exists in regards to the cervical spine. Hattori et al. performed the only (known) in-vivo study investigating IDP in the cervical spine[88]. Given ethical conflicts and technical barriers to recreating / confirming Dr. Hattori's in-vivo study, investigators have opted to perform in-vitro investigations of IDP with simulated in-vivo loading conditions. However, the most appropriate method of applying compressive loading to a multi-segment cervical spine is still under debate in literature—although the follower load method of applying compressive preload to multi-segment cervical spine seems to have become the standard[16].

This study is the first to investigate in the influence of follower load application on intradiscal pressures in a multi-segment cervical spine at levels multiple levels, C4-C5 and C5-C6. Additionally, this study is the first to attempt to correlate the intradiscal pressures with intervertebral disk height which several authors have referenced as a marker of degeneration[91-95]. However, the use of disc height alone is also a limitation of the study. The fact that MRI degeneration grade was not assessed in this study made it difficult to compare the results of this study to previous literature.

In comparing the present study the only available in-vivo study Hatorri et al.[88] found that the IDP of normal discs in the supine position was the lowest of all positions tested (44.95±5.83 PSI). Patients in the sitting position with their neck in the neutral position had an increase of their intradiscal pressure of approximately 1.4 times the supine pressure (63.58±7.40 PSI).

The present study was also able to show an increase in IDP with the application of 100N of follower load of approximately 4.6 times at C4-C5 and 2.6 times at C5-C6. The larger increases in pressure compared the in-vivo results can be explained by the fact the the in-vivo study a ‘no compression’ state does not exist. Even in the supine position, basal muscle activation exists and therefore the change observed when going to the seated position only reflects a change in muscle activation and application of the weight of the head.

It is also important to correlate the relationship between in-vivo and in-vitro intradiscal pressures measurements of the cervical spine as future testing will likely rely mainly on in-vitro testing and standard pressure measurements should be established upon which to base future testing. The IDP method in the present study was based on the proven techniques as described by Cripton et al. with the use of a miniature pressure inducer[96]. The reliability of the this method was confirmed in a laboratory controlled set-up and proved to be a reliable system (ICC=1.000). This system was adopted as to not restrict or alter any segmental movement due to contact with the complex bony anatomy which can occur with rigid needle pressure sensor systems.

Pospiech et al.[26, 89] confirmed the findings of Hatorri in an in-vitro setting, which is the first in-vitro evidence to match Hatorri's study. Pospiech confirm similar variability in intradiscal pressure but also identified the important role that the stimulation of muscle forces in raising intradiscal pressures. Pospiech tested specifically C3-C4 and C5-C6 with the C3-C4 segment showed an increase of pressure with muscle loading to be 1.13x nonloaded conditions, where as C5-C6 incured the majority of the load with increased pressure by 2.79x. Our findings are in agreement with this previous literature with a significant increase in pressure at both the C4-C5 and C5-C6 discs with application of the follower load but our results suggest a more dramatic change in the more cranial segment.

Hattori et al. also observed that IDP changes (generally increases) with FE, with the highest pressures being observed in extension and the next highest being observed in flexion. This finding contradicts our results in that we observed the highest IDP in flexion not extension.

However, Hattori also reported that degeneration affected the variability and patterns observed at flexion and extension with some subjects exhibiting highest pressure in flexion with the neutral position not always being the lowest. In the present study disc height relative to the inferior adjacent vertebral body height was recorded and correlated with IDP. The only significant correlation that was observed was change in IDP that occurred with extension (IDP – extension delta). The moderate correlation was positive indicating that with low RDH (modeling a degenerative state) that decreases or small increases in IDP occurred with FE. It is expected that this finding is consistent with an off loading of the disc due to bony contact at the facet joints. These results help explain the variability that Hattori observed with degeneration and why IDP – flexion delta recorded the largest increases in IDP in the present study.

The concept of off-loading due to bony contact also helps to explain why only weak correlations between RDH and the change in IDP with flexion (IDP – flexion delta). Due to the anatomy of the cervical spine flexion would not result in any bony contact posteriorly whereby not having any effect on the IDP. In contrast, in the neutral position compressive loading could result in off-loading by bony contact which is consistent with the mild correlation observed between RDH and IDP that was increased with the application of follower load.

Wu et al.[97] also investigated the influence of FE on IDP in an in-vitro setting in the presense of 100N of compression. However, the test set-up used in Wu's study varied dramatically from the set-up used in the current study. Most notably the compressive load was applied vertically as to simulate the weight of the head (not in the follower load manner). Also the axial compression was applied at fixed angular positions ranging from 20 degrees of flexion to 20 degress of extension and not recorded continuously during pure moment application. Wu found the highest pressure at 20 degrees of flexion and consistently decreasing pressure with the lowest pressue being observed at 20 degrees of extension. Although this finding is apparently consistent with our IDP – flexion delta, it is expected that these results are reflective of the curvature of the spine coupled with the purely vertical method of applying axial compression resulting in less load passing through the discs in extension.

Some publications have investigated stress profilometry in the cervical spine FSUs and in addition to showing changing profile across the disc space, were able to explore the compressive loading, postural, and degeneration dependence findings of Hattori[98, 99]. Although these reports have significantly added to the understanding of cervical spine IDP, it is difficult to extrapolate these FSU results for direct comparison with the in-vivo study, due to the limitations of the model and loading methodology. Wigfield et al.[99] attempted to determine maximum IDP related to both cervical segment and moment. Wigfield found maximum peak stress occurred in extension and was located in the posterior region of the disc (C2-C5) while in lower segments, maximum peak stess occurred with flexion and at the anterior annulus (C5-T1). While the location of maxium stress within the disc in both flexion and extension appears logical, the loading system employed by Wigfield was a platform system which does not account for the normal lordosis of the cervical spine instead rather compressed only at the top and bottom portions of the spine. This limitation is avoided with the multi-segment follower load methodology employed in the current study. Another study was published by the same research group, using similar methodology, looking at the influence of degeneration, segment and FE on IDP[98]. Skyzypiec's results confirm the findings of this study that severe degeneration decreases IDP in the neutral position, however IDP always increase with flexion regardless of degeneration grade. Conversly, although extension increases IDP in specimen with mild degeneration reductions in IDP are observed with increasing degeneration grade. The findings of the stress profilometry studies help to reinfornce the finding of the present study, however they also point out an important limitation. Although, a lateral x-ray was used in preliminary testing to confirm the placement of the pressure sensor in the disc, this proecdure was not repeated for each test. Given that placement of the pressure gauge can impact the pressure readings with pressure lowest in the exact middle of the disc but maximum elevation in the anterior annulus with flexion and posterior annulus with extension[98], uncertainty of the sensor placement is potentially a source of variability in the present study.

In summary, the current study combined simulated muscle loading (Follower Load, FL) with a robot/UFS testing system capable to applying pure moments to a multi-segment cervical spine to more closely replicate Hattori's in-vivo methodology.   The effects of adding FL compression approximates the effect of the patient going from the supine to seated position. However, data showed a higher increase in IDP with flexion than extension, whereas Hattori's study showed the opposite effect on average, although this was not consistent and depended on degeneration grade. The high significant correlation between IDP – extension delta and disc height may give insight into the mechanics of IDD and is of great interest to researchers and clinicians.

5.0 COMPARISON OF IN-VIVO AND IN-VITRO KINEMATICS

5.1 BACKGROUND

In-vitro biomechanical testing has been critical in the design and evaluation of surgical instrumentation. Determination of realistic physiologic loading levels for the cervical spine has, however, proven difficult outside of the in-vivo setting. Unconstrained pure moment testing combined with the hybrid testing method is currently the gold standard test protocol for evaluation of motion preservation technology and adjacent level effects. Pure moment testing was specifically designed to apply uniform loading at each cross section throughout the length of a spinal construct, whereby permitting irregularities to be identified [3]. Pure moment testing is well suited for making relative comparisons between treatments, but is currently not based on or representative of in-vivo motion, bringing the clinical relevance into question [4].

Additionally, the human cervical spine supports substantial compressive load in-vivo arising from muscle forces and the weight of the head. However, the traditional in-vitro testing methods rarely include compressive loads; especially in investigations of multi-segment cervical spine constructs. Various methods of modeling physiologic loading have been reported in the literature including axial forces produced with inclined loading plates, eccentric axial forces application, follower load, as well as attempts to individually apply/model muscle forces in-vitro [3, 4, 15-25].

Miura et al. [21] and DiAngelo and Foley [15] published articles directly aimed at determining the most appropriate loading mechanism to produce physiologic motion patterns. Miura et al. presented pure moment testing combined with follower load and through adjusting moment targets combined with the was able to achieve ~20% agreement with segmental range of motion reported in literature, however typical segmental motion patterns were not observed with this technique [21]. DiAngelo and Foley utilized an eccentric axial compressive method in attempt to mimic the weight of the head. DiAngelo and Foley were able to show reasonable agreement with the segmental motion patterns, but the magnitudes dramatically underestimated the average in-vivo segmental kinematics [15].

The objective of this project is to identify and verify the appropriate in-vitro loading conditions that would replicate the in-vivo kinematics of the cervical spine, with the overall goal of improving the biofidelity of the experimental platform. A systematic comparison of standard pure moment with no compressive loading versus published and novel compressive loading techniques (follower load - FL, axial load - AL, and combined load - CL) was performed. It is hypothesized that an optimized follower load, passing through the segmental centers of rotation, will add stability to the system but will not dramatically affect the segmental motion patterns observed throughout the extension-flexion motion path. In contrast, axial load applied perpendicular to superior most vertebral body, will not maintain the pure moment assumption, whereby enabling the segmental motion patterns to be altered.

5.2 METHODS

5.2.1 Protocol

N = 12 fresh-frozen human (C3-C7) cervical cadaveric specimens (51.8 years ± 7.3) were pre-screened with CT and dissected, preserving osteoligamentous structures. Specimens were mounted in a robot/UFS testing system, consisting of a serial linkage robotic manipulator (Staubli RX90, Staubli Inc., Duncan, SC) with an on-board six-axis load cell (UFS Model 90M38A-150, JR3 Inc., Woodland, CA) and custom specimen-mounting fixtures [32, 33, 55]. Four clinical lateral mass screws were used to secure the specimens to the mounting fixtures (one in each pedicle and two in the anterior portion of the vertebral body). After mounting, specimens were wrapped in 0.9% saline soaked gauze and periodically sprayed with saline in order to prevent dehydration. The robot was controlled via MATLAB (Mathworks, Inc.) and operates under adaptive displacement control to a pure moment target of 2.0 Nm for flexion and extension (FE) for each state in a randomized order (no compression (Figure 21A), follower load, axial rotation, combined loading). Due to the quasi-static nature of the adaptive displacement control algorithm the system operated at a rate of 0.067 ± 0.0014 degrees/second. Two consecutive full extension-flexion loops were performed with the data from the second extension-flexion cycle being presented to account for preconditioning [19]. Segmental motion was recorded using a five camera VICON system tracking passive reflective markers rigidly attached as a marker group to each vertebral body. A hand held VICON digitizer was utilized to digitize the anatomical coordinate system (Appendix A.3.1) for each vertebral body relative to the marker group and the Euler angles of C3-C4, C4-C5, C5-C6, and C6-C7 were determined and reported (Appendix A.3 & A.4).

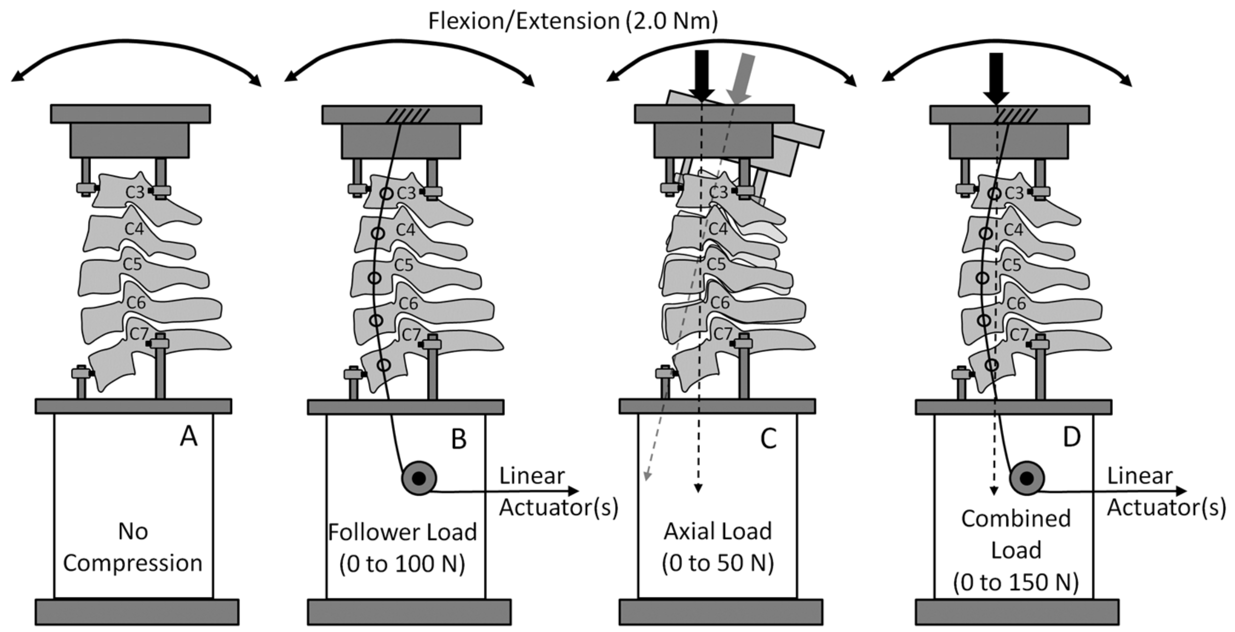


Figure 21. Schematic of the four loading states implemented in this study

5.2.2 Follower Load

Follower load application was accomplished by loading the specimen with bilateral cables passing through cable guides inserted into the vertebral bodies and over pulleys attached to the base (Figure 21B). A novel active system was implemented in our laboratory using linear actuators coupled with load cells. Control of the system was integrated with the custom-built PC-based control program written in MATLAB currently used to control the robot/UFS testing system, and enabled active control of the loading throughout the motion path. The follower load system consists of two independently controlled 24V servo motor linear actuators (Ultramotion-3-B.125-DC426_24-4-/4) and compression/tension load cells (Transducer Techniques – MLP-100). A Galil Motion Controller (DMC-4183-BOX8(-16BIT)-D3040-D4040) controlled this system using on-board closed loop Proportional-Integral-Derivative (PID) load control. The

3/64” diameter stainless steel wire rope lanyard was threaded through a custom designed adjustable cable guide system attached with clinical pedicle screws to enable the follower load cable to interface with the specimen in a manner consistent with the design criterion: a) tangent to the curvature of the spine and b) pass through the specimen’s center of rotation (COR). Optimization of the follower load path to align with the specimen’s COR was accomplished through an offline iterative feedback process using the moment output of the testing system’s on-board six–axis load cell. With the specimen in the neutral position, 100N of follower load was applied to the specimen and resulting change in moments was recorded. The position of the cable guide was then adjusted to counteract the moment change and the process was repeated until less than 0.1Nm change in moment was observed. Preliminary testing of the described optimization process was performed ensuring that the previously described maintenance of segmental curvature angle criteria was upheld [63].

5.2.3 Axial Load

Although less popular than follower load as a method to apply compressive load due to published instability issues with this testing method, some authors believe axial loading to be the most physiologic loading scheme—mimicking head weight [15]. Axial loading can be applied along an axis locally fixed to the specimen or globally fixed to the world coordinate system. Previous reports have shown that cervical spine buckles at very low loads when an axial load is applied globally, therefore for this study the axial load was applied along an axis locally fixed to the specimen (perpendicular to the robot end effector – Figure 21C). The axial load was applied using the robotic arm to a load target of 50N using the adaptive displacement control algorithm enabling the load to be applied purely in the axial direction and be maintained throughout the extension-flexion path.

5.2.4 Combined Load

Simulation of muscle loading and the weight of the head with the follower load and axial compressive load have been independently shown to result in modest improvements in making the in-vitro motion more physiologic [15, 21]. In this study, the two loading schemes will be combined. It is hypothesized that the combination of axial and follower loading will have a synergistic effect, producing more physiologic kinematics (Figure 21D).

5.2.5 In-Vivo Data

The previously reported in-vivo data set [100, 101] utilized in the study was reanalyzed for direct comparison with the current in-vitro data. In-vivo data consisted of N=20 asymptomatic control patients (45.5 years ± 5.8) consented to participate in an IRB approved protocol. Subjects performed continuous, full ROM extension-flexion at a rate of one complete cycle every 3 seconds. Subject-specific bone models of C3-C7 were created from CT scans. A previously validated tracking process determined three-dimensional vertebral position with sub-millimeter accuracy by matching bone models from the CT scan to the biplane X-rays [31].

5.2.6 Data Analysis

Results of the in-vitro testing at full range of motion (ROM) were summarized and expressed in bar graphs representing mean ± 95% confidence intervals. A non-parametric Wilcoxon signed-rank test for paired design was used to identify significant ($p<0.05$) differences between loading modes (no compression versus follower load, axial load and combined load). In-vivo data was uniformly scaled to match the in-vitro data by adjusting for mean differences between in-vitro and in-vivo ROM and normalizing to percent ROM (Appendix A.5). The quality of fit between in-vivo and in-vitro motion was assessed by calculating root mean square error (RMSE) at 20% increments of percent ROM.

5.3 RESULTS

Figure 22 displays the segmental contributions at 100% of the overall extension-flexion motion path. At 100% ROM the segmental distribution for no compressive load state exhibited approximately equal segmental contribution from each level. Adding follower load only had minimal effects on the segmental contribution from each level. However, in comparison to the no compression state, adding the axial load significantly reduced the C6-C7 contribution to the motion path, shifting the majority of the motion toward the middle (C4-C5 and C5-C6) segments. This significant reduction was also observed in the combined loading state.

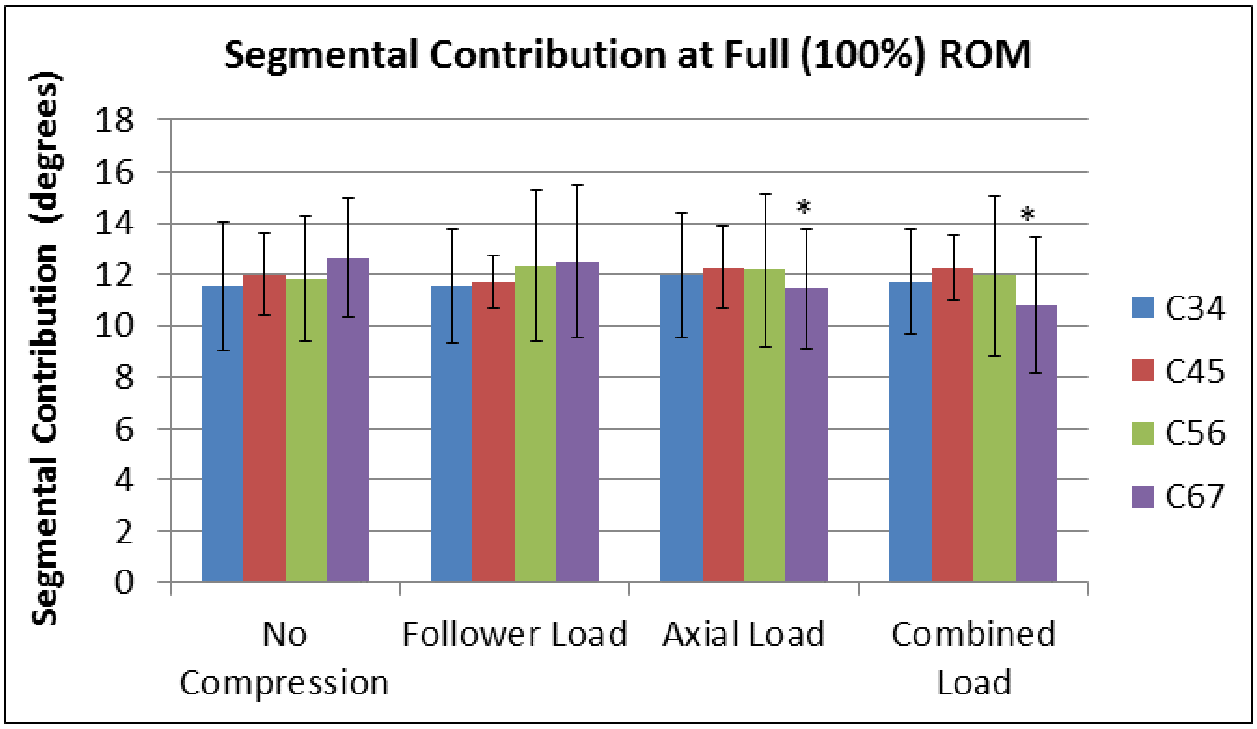


Figure 22. Average segmental kinematics at 100% ROM

The overall extension-flexion ROM was not affected by the application of a compressive load, with all four compressive loading states exhibiting virtually identical ROM (Figure 23A). However, in comparison to the in-vivo ROM, the in-vitro ROM was on average 12.3% smaller than the in-vivo data set, although difference was not statistically significant. Therefore, in order to directly compare the segmental contributions between the in-vivo and in-vitro data the in-vivo segmental rotation was uniformly scaled by a factor of 87.7% (Figure 23B).

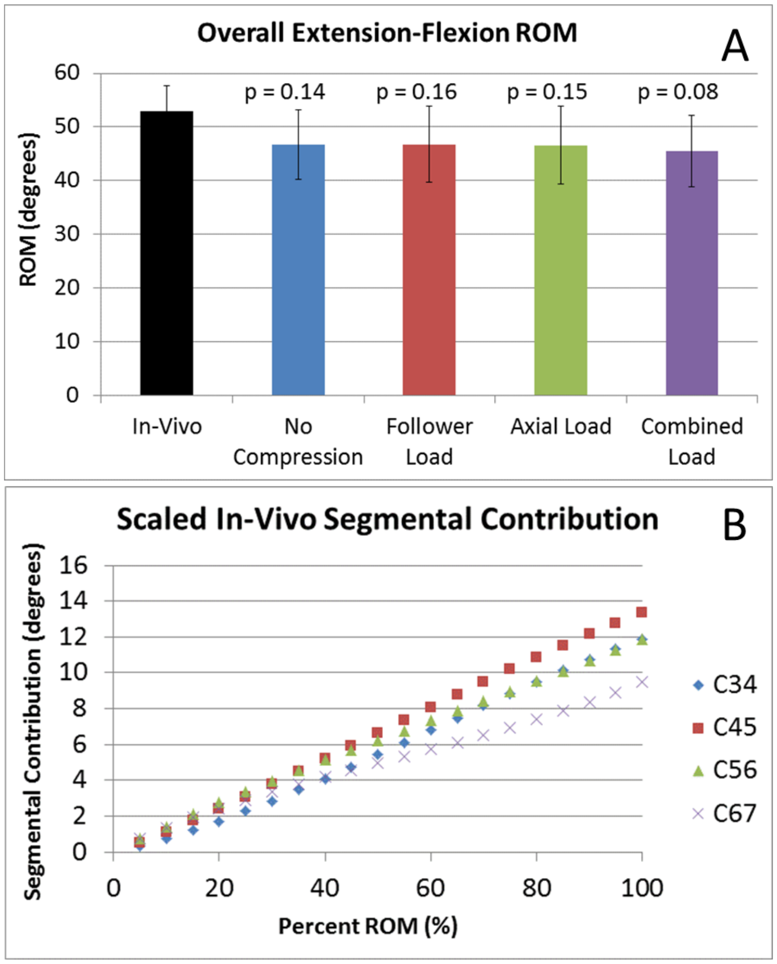


Figure 23. Overall ROM and scaled in-vivo segmental rotation

The scatter plot of the in-vitro segmental contribution plotted versus percent of total ROM (Figure 24) demonstrates clearly that adding compressive load effects the segmental motion distribution. A small change is observed with application of follower load, wherein C3-C4 and C4-C5 appear to contribute more than C5-C6 and C6-C7 throughout middle portion of the extension- flexion curve. This same trend is further magnified with the application of an axial compressive load. The axial compressive loading state and the combined loading curves show a very similar pattern with the upper segments appearing to be recruited from superior to inferior as the motion path progresses.

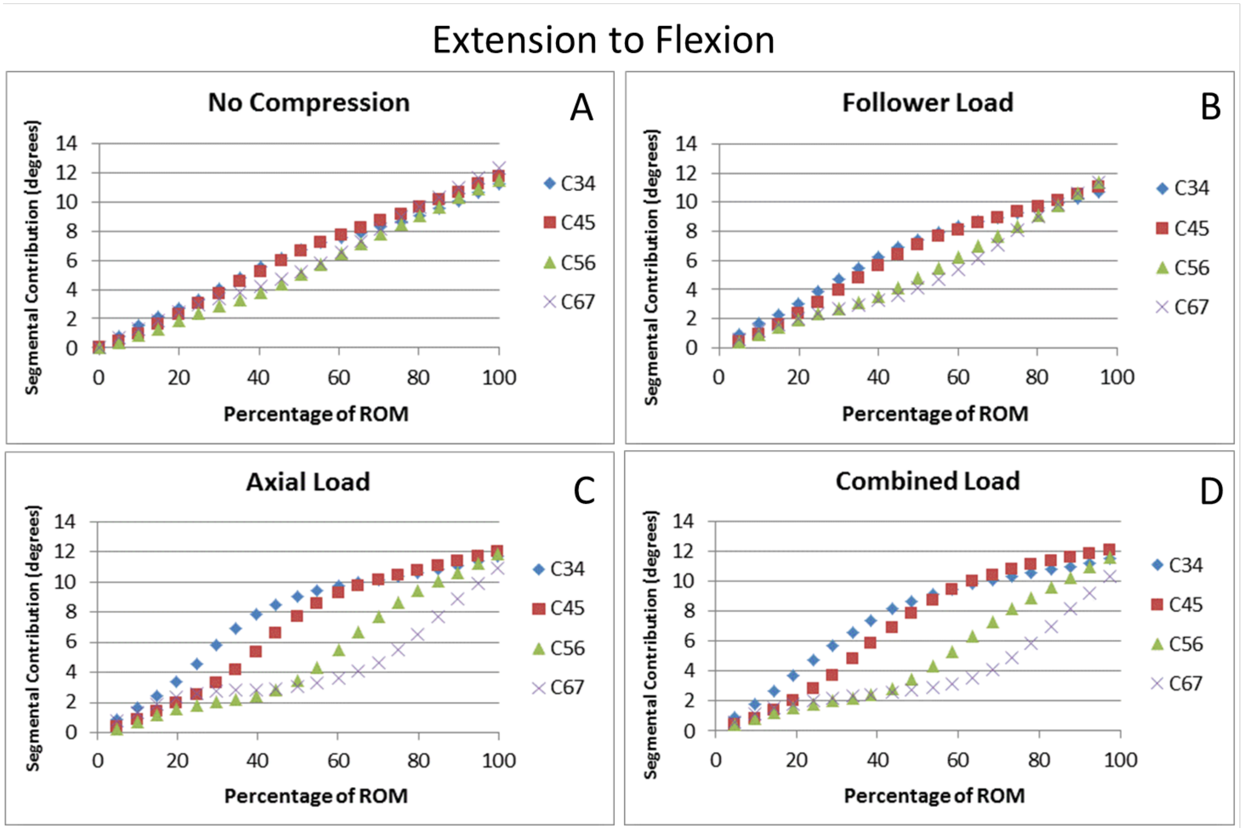


Figure 24. Effects of altered compressive loading on in-vitro segmental contribution

The average RMSE between the in-vivo data and in-vitro data increased with the application of compressive loading. However, the RMSE was dependent on percentage ROM (Figure 25A) and segmental level (Figure 25B). No compression had the lowest RMSE at 20%, 40% and 60% ROM, which corresponds to the extension and neutral region of the overall path of motion. However, the addition of axial and combined loading resulted in the lowest RMSE at 80% and 100% ROM, which corresponds to the flexion region of the motion path (Figure 25, Table 3). No compression and follower load had the lowest RMSE at C3-C4 and C5-C6; however, axial load and combined load had the lowest RMSE at C4-C5 and C6-C7.

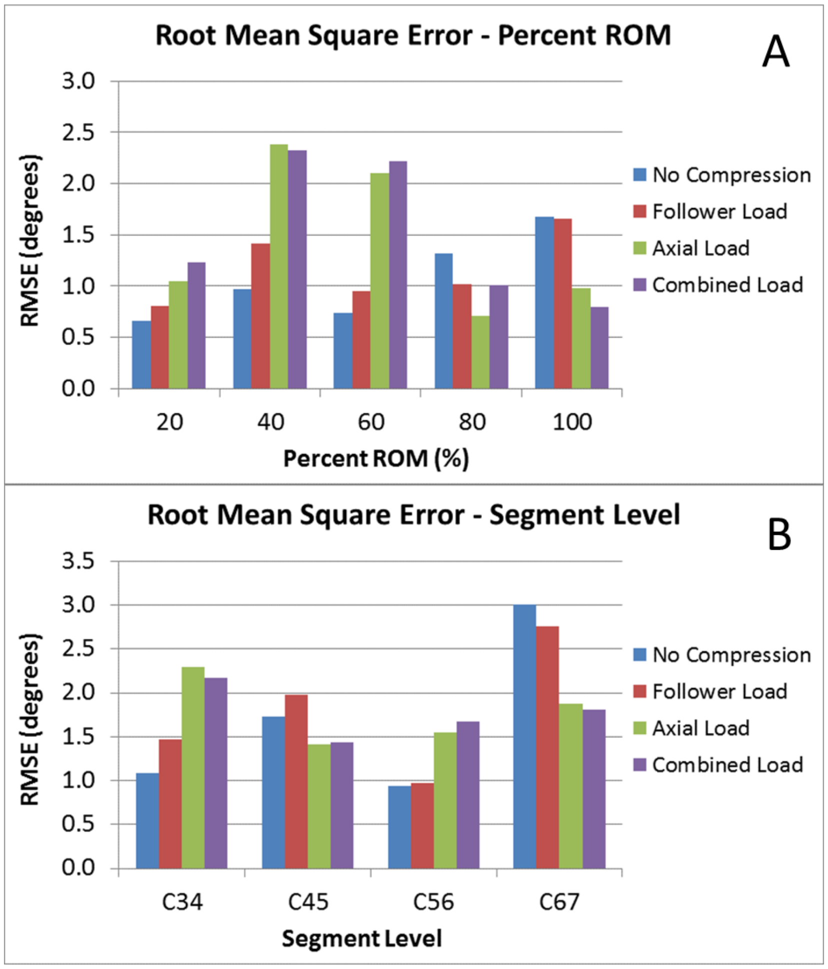
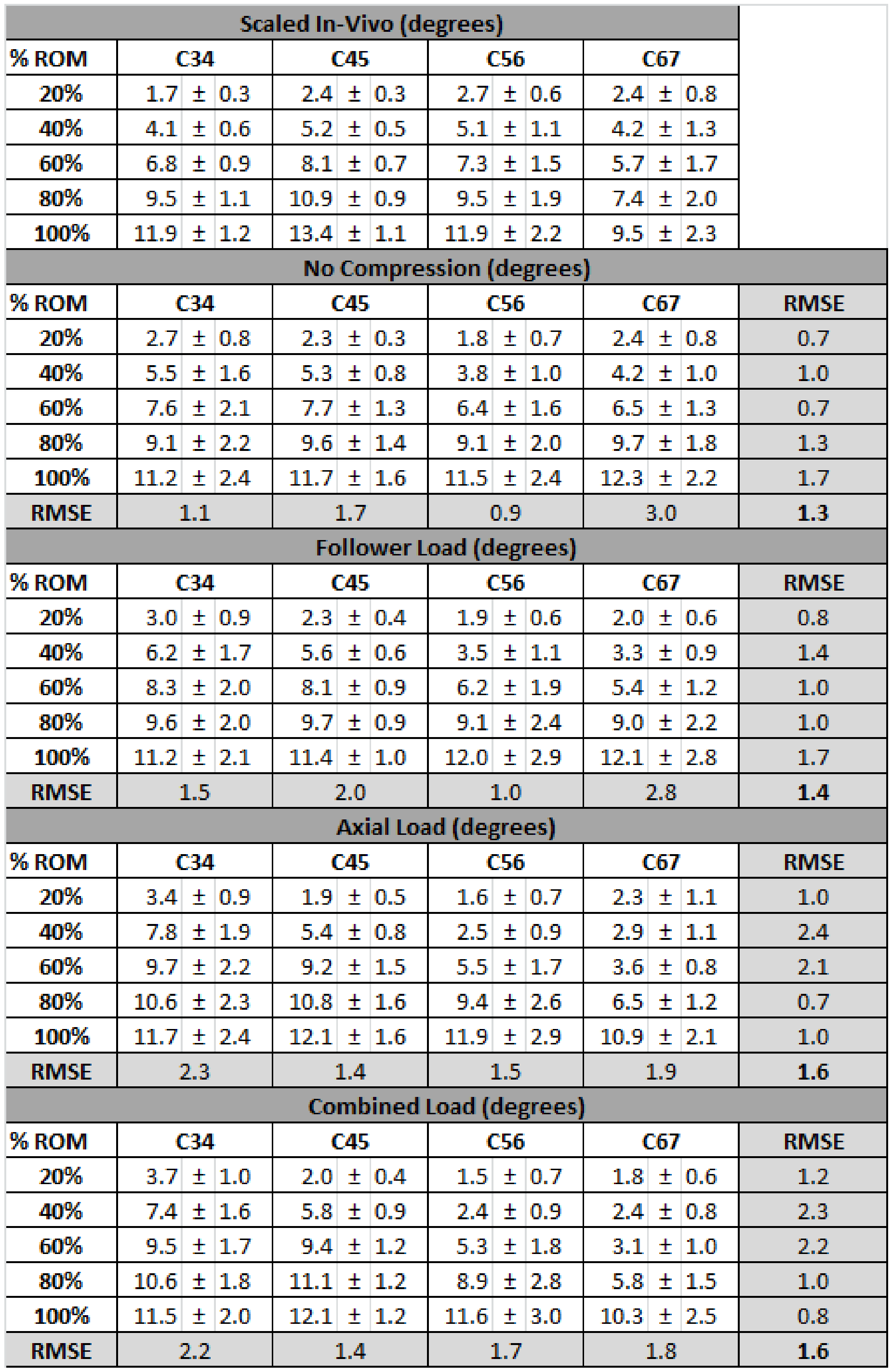


Figure 25. RMS errors between in-vivo and in-vitro motion

Table 3. Raw kinematic data and RMSE comparison



Scaled In-Vivo (degrees)					
% ROM	C34	C45	C56	C67	
20%	1.7 ± 0.3	2.4 ± 0.3	2.7 ± 0.6	2.4 ± 0.8	
40%	4.1 ± 0.6	5.2 ± 0.5	5.1 ± 1.1	4.2 ± 1.3	
60%	6.8 ± 0.9	8.1 ± 0.7	7.3 ± 1.5	5.7 ± 1.7	
80%	9.5 ± 1.1	10.9 ± 0.9	9.5 ± 1.9	7.4 ± 2.0	
100%	11.9 ± 1.2	13.4 ± 1.1	11.9 ± 2.2	9.5 ± 2.3	
No Compression (degrees)					
% ROM	C34	C45	C56	C67	RMSE
20%	2.7 ± 0.8	2.3 ± 0.3	1.8 ± 0.7	2.4 ± 0.8	0.7
40%	5.5 ± 1.6	5.3 ± 0.8	3.8 ± 1.0	4.2 ± 1.0	1.0
60%	7.6 ± 2.1	7.7 ± 1.3	6.4 ± 1.6	6.5 ± 1.3	0.7
80%	9.1 ± 2.2	9.6 ± 1.4	9.1 ± 2.0	9.7 ± 1.8	1.3
100%	11.2 ± 2.4	11.7 ± 1.6	11.5 ± 2.4	12.3 ± 2.2	1.7
RMSE	1.1	1.7	0.9	3.0	**1.3**
Follower Load (degrees)					
% ROM	C34	C45	C56	C67	RMSE
20%	3.0 ± 0.9	2.3 ± 0.4	1.9 ± 0.6	2.0 ± 0.6	0.8
40%	6.2 ± 1.7	5.6 ± 0.6	3.5 ± 1.1	3.3 ± 0.9	1.4
60%	8.3 ± 2.0	8.1 ± 0.9	6.2 ± 1.9	5.4 ± 1.2	1.0
80%	9.6 ± 2.0	9.7 ± 0.9	9.1 ± 2.4	9.0 ± 2.2	1.0
100%	11.2 ± 2.1	11.4 ± 1.0	12.0 ± 2.9	12.1 ± 2.8	1.7
RMSE	1.5	2.0	1.0	2.8	**1.4**
Axial Load (degrees)					
% ROM	C34	C45	C56	C67	RMSE
20%	3.4 ± 0.9	1.9 ± 0.5	1.6 ± 0.7	2.3 ± 1.1	1.0
40%	7.8 ± 1.9	5.4 ± 0.8	2.5 ± 0.9	2.9 ± 1.1	2.4
60%	9.7 ± 2.2	9.2 ± 1.5	5.5 ± 1.7	3.6 ± 0.8	2.1
80%	10.6 ± 2.3	10.8 ± 1.6	9.4 ± 2.6	6.5 ± 1.2	0.7
100%	11.7 ± 2.4	12.1 ± 1.6	11.9 ± 2.9	10.9 ± 2.1	1.0
RMSE	2.3	1.4	1.5	1.9	**1.6**
Combined Load (degrees)					
% ROM	C34	C45	C56	C67	RMSE
20%	3.7 ± 1.0	2.0 ± 0.4	1.5 ± 0.7	1.8 ± 0.6	1.2
40%	7.4 ± 1.6	5.8 ± 0.9	2.4 ± 0.9	2.4 ± 0.8	2.3
60%	9.5 ± 1.7	9.4 ± 1.2	5.3 ± 1.8	3.1 ± 1.0	2.2
80%	10.6 ± 1.8	11.1 ± 1.2	8.9 ± 2.8	5.8 ± 1.5	1.0
100%	11.5 ± 2.0	12.1 ± 1.2	11.6 ± 3.0	10.3 ± 2.5	0.8
RMSE	2.2	1.4	1.7	1.8	**1.6**

5.4 DISCUSSION

This study systematically evaluated standard pure moment with no compressive loading versus published and novel compressive loading techniques (follower load, axial load, and combined load). Consistent with previous in-vitro reports, the pure moment testing protocol with no compression was not able to replicate the typical in-vivo segmental motion pattern reported in literature [15, 21]. Although variability existed from specimen to specimen do to local degenerative differences between specimens, overall applying a pure moment at each segment resulted in approximately equal segmental contribution to the overall extension-flexion motion path. Equal segmental contribution is advantageous for comparative studies; however the absence of physiologic loading magnitude and the fact that the in-vivo segmental motion patterns are not reproduced throughout the entire motion path does limit the clinical relevance of this method. In-vitro tests that fail to accurately reproduce in-vivo kinematics through the entire motion path may lead to incorrect assessments of spinal instrumentation function, especially for motion sparing devices such as total disc replacements. Adding follower load allowed the specimen to be tested at a physiologic loading magnitude, while maintaining the pure moment assumption. However, this combined loading scheme was still not able to fully replicate the in-vivo segmental motion patterns.

Interestingly, the follower load state in the present study reproduced extension-flexion parameters used in the Miura et al. study [21]. Similar trends are observed between the two studies in terms of percent contribution, however the ROM magnitude observed at 2.0Nm in the Miura et al. study was approximately 20% more than the ROM observed in this study. Differences in specimen grade and testing methods could account for some of this disagreement; however in both cases a larger moment target would have been necessary to replicate the magnitude of the reported in-vivo kinematics. Additionally, the Miura et al. study only reported end-range kinematics, which is the predominate trend in literature, however the end-range kinematics are not necessarily representative of the mid-range cervical kinematics and are highly variable [102-105]. High variability in the in-vivo and in-vitro data sets was also a limitation of the present study, although attempts were made to limit the variability by controlling recruitment age and pre-screening subjects and specimens.

The present study is unique in that a direct comparison to continuous cervical kinematics over the entire extension-flexion motion path was possible. As hypothesized, application of an axial load perpendicular to the superior most vertebral body altered the segmental motion patterns. The most dramatic differences were observed in the middle portion of the curve (the neutral zone). Very few authors have reported segmental distribution data at sub-maximal points on the motion curves, making this data difficult to interpret [106]. Although further investigation is necessary to understand this fully, it is theorized that the inherent laxity of the neutral zone combined with the lordotic curvature of the intact cervical spines account for these dramatic shifts in the segmental motion pattern. The changes in segmental motion pattern were less pronounced at full range of motion, axial load and combined loading were able to produce a motion pattern similar to in-vivo at the extremes of the extension-flexion motion path. This observation is consistent with the study published by DiAngelo and Foley [15], which utilized an eccentric axial compressive method in attempt to mimic the weight of the head. DiAngelo and Foley were able to show reasonable agreement with the segmental motion patterns, but the reported magnitudes dramatically underestimated the average in-vivo segmental kinematics [15]. Although, the two methods share similarities, it is theorized that the eccentric axial loading method has limited stability, an issue that is corrected in the present study with the combined loading methodology.

Based on this promising preliminary data, future investigations focused on optimizing the axial load and combined loading methods for replication of in-vivo cervical spine kinematics are recommended. The data suggests that dynamically altering the compressive loading throughout the motion path is necessary in order to mimic the segmental contribution patterns exhibited in-vivo—a novel concept that will be explored in future investigations. Additionally, while this study focused on kinematics, future investigations exploring the effects on resulting kinetics (intradiscal pressure and load displacement curve) are also warranted, although the in-vivo data available for comparison is very limited making comparison difficult [22, 26].

6.0 OPTIMIZATION OF COMPRESSIVE LOADING PARAMETERS

6.1 BACKGROUND

The human cervical spine supports substantial compressive load in-vivo arising from muscle forces and the weight of the head. However, the traditional in-vitro testing methods rarely include compressive loads, especially in investigations of multi-segment cervical spine constructs. Various methods of modeling physiologic loading have been reported in the literature including axial forces produced with inclined loading plates, eccentric axial force application, follower load, as well as attempts to individually apply/model muscle forces in-vitro. The importance of proper compressive loading to recreate the segmental motion patterns exhibited in-vivo has been highlighted in previous studies [15, 21, 22]. However, appropriate methods of representing the weight of head and muscle loading are currently unknown.

Previously, a systematic comparison of standard pure moment with no compressive loading versus published and novel compressive loading techniques (follower load - FL, axial load - AL, and combined load - CL) was performed. The pure moment testing protocol without compression or with the application of follower load was not able to replicate the typical in-vivo segmental motion patterns throughout the entire motion path. Axial load or a combination of axial and follower load was undesirable through the neutral position but necessary to mimic the in-vivo segmental contributions at the extremes of the extension-flexion motion path. It was hypothesized that dynamically altering the compressive loading throughout the motion path is necessary to mimic the segmental contribution patterns exhibited in-vivo.

The systematic comparison of the compressive loading techniques was structured a priori using statistical design of experiments (DOE). Design of experiments is a statistical technique, used primarily in quality control, wherein experiments are intentionally planned at the data collection stage to ensure valid and defensible conclusions are determined with minimal cost [107]. DOE is particularly beneficial at the screening phase of experimentation when there is a hypothesized effect of some input factor, but the parameters surrounding the anticipated effect are largely unknown.

The DOE methodology was also chosen based on the goal of determining the optimal compressive loading parameters (multiple inputs) required to mimic the segmental motion patterns exhibited in-vivo (multiple outputs). One common technique for optimizing multiple response processes is the desirability function approach [108]. The desirability function approach searches for the optimal input conditions that provide the "most desirable" outputs. A unique attribute of the desirability function is that any output parameter outside of the desired limits is unacceptable and results in a desirability score of zero (0) whereas complete agreement for all response values results in an desirability score of one (1).

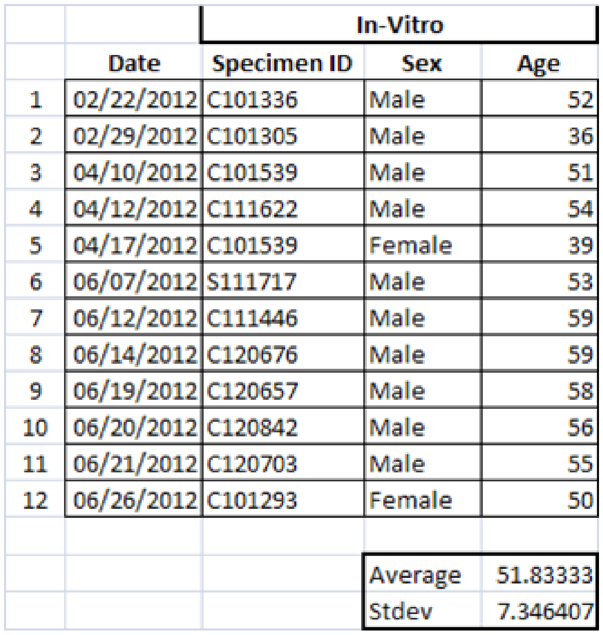
Therefore, this study utilized DOE and the desirability function to further explore the finding that axial load or a combination of axial and follower load was undesirable through the neutral position but necessary to mimic the in-vivo segmental contributions at the extremes of the extension-flexion motion path. The objective of this study was to determine the optimal compressive loading parameters to confirm or reject the hypothesis that dynamically altering the compressive loading throughout the motion path is necessary to mimic the segmental contribution patterns exhibited in-vivo.

6.2 METHODS

6.2.1 In-Vitro Dataset

N = 12 fresh-frozen human (C3-C7) cervical cadaveric specimens (51.8 years ± 7.3) were pre-screened with CT and dissected, preserving osteoligamentous structures (Table 4). Specimens were mounted in a robot/UFS testing (methods described previously (Chapter 5.2)). The robot was controlled via MATLAB (Mathworks, Inc.) and operates under adaptive displacement control to a pure moment target of 2.0 Nm for flexion and extension (FE) for each state in a randomized order (no compression, follower load = 100N, axial rotation = 50N, combined loading=150N). Segmental motion was recorded using a five camera VICON system tracking passive reflective markers rigidly attached as a marker group to each vertebral body. A hand held VICON digitizer was utilized to digitize the anatomical coordinate system for each vertebral body relative to the marker group and the Euler angles of C3-C4, C4-C5, C5-C6, and C6-C7 were determined and reported.

Table 4. Summary of In-Vitro Testing

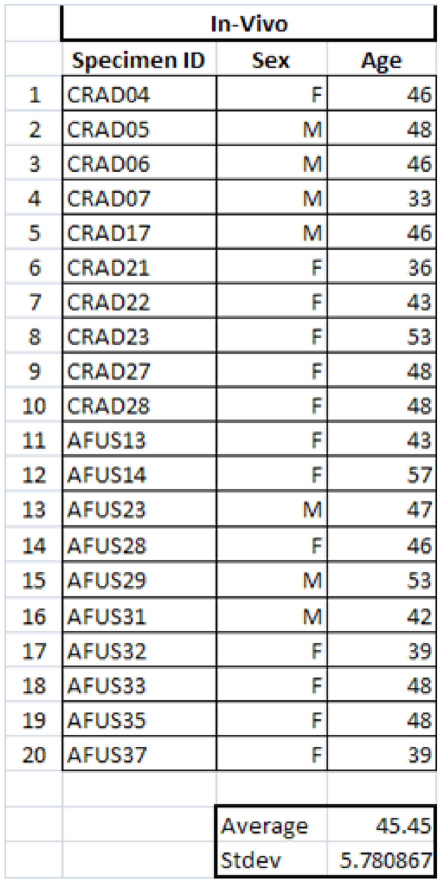


		In-Vitro		
	Date	Specimen ID	Sex	Age
1	02/22/2012	C101336	Male	52
2	02/29/2012	C101305	Male	36
3	04/10/2012	C101539	Male	51
4	04/12/2012	C111622	Male	54
5	04/17/2012	C101539	Female	39
6	06/07/2012	S111717	Male	53
7	06/12/2012	C111446	Male	59
8	06/14/2012	C120676	Male	59
9	06/19/2012	C120657	Male	58
10	06/20/2012	C120842	Male	56
11	06/21/2012	C120703	Male	55
12	06/26/2012	C101293	Female	50
			Average	51.83333
			Stdev	7.346407

6.2.2 In-Vivo Dataset

The previously reported in-vivo data set [100, 101] utilized in the study was reanalyzed for direct comparison with the current in-vitro data. In-vivo data consisted of N=20 asymptomatic control patients (45.5 years ± 5.8) consented to participate in an IRB approved protocol (Table 5). Subjects performed continuous, full ROM extension-flexion at a rate of one complete cycle every 3 seconds. Subject-specific bone models of C3-C7 were created from CT scans. A previously validated tracking process determined three-dimensional vertebral position with sub-millimeter accuracy by matching bone models from the CT scan to the biplane X-rays [31].

Table 5. Summary of In-Vivo Testing



	In-Vivo		
	Specimen ID	Sex	Age
1	CRAD04	F	46
2	CRAD05	M	48
3	CRAD06	M	46
4	CRAD07	M	33
5	CRAD17	M	46
6	CRAD21	F	36
7	CRAD22	F	43
8	CRAD23	F	53
9	CRAD27	F	48
10	CRAD28	F	48
11	AFUS13	F	43
12	AFUS14	F	57
13	AFUS23	M	47
14	AFUS28	F	46
15	AFUS29	M	53
16	AFUS31	M	42
17	AFUS32	F	39
18	AFUS33	F	48
19	AFUS35	F	48
20	AFUS37	F	39
		Average	45.45
		Stdev	5.780867

6.2.3 Data Analysis

The DOE methodology was chosen based on the goal of determining the optimal compressive loading parameters (multiple inputs) required to mimic the segmental motion patterns exhibited in-vivo (multiple outputs). More specifically, physiologic values of the independent variables (FL and AL) were tested singly and in combination in order to explore their effect on the dependent variables (segmental kinematics) (Table 6).

Table 6. Design of experiments – protocol design

		Independent		Dependent			
State	Run	FL	AL	C34	C45	C56	C67
No Compression	1	0	0	?	?	?	?
Follower Load	2	100	0	?	?	?	?
Axial Load	3	0	50	?	?	?	?
Combined Load	4	100	50	?	?	?	?

An analysis of variance (ANOVA) was performed to determine percentage of the variance in the dependent variables that is attributed to the independent variables which is quantified in terms of percent contribution and standardized effect size of the independent variables.

In-vivo data was uniformly scaled to match the in-vitro data by adjusting for mean differences between in-vitro and in-vivo total ROM and normalizing to percent ROM. Optimization was then performed based on the output of the ANOVA using desirability function approach. The desirability function approach searches for the optimal independent variables (FL and AL) that provide the "most desirable" independent variables (segmental kinematics). It was previously reported that effects of the loading conditions were dependent on percentage of range of motion (ROM) and segmental level, therefore the optimization was performed at 20%, 40%, 60%, 80%, and 100% of the overall extension- flexion motion path. Additionally, the dependent variables were defined as C3-C4, C4-C5, C5-C6, and C6-C7 in order to assess the effects independently. Depending on whether a particular response is to be maximized, minimized, or assigned a target value different desirability functions are used [109]. For this investigation since the goal was to optimize the in-vitro data used the in-vivo data as the target (gold standard) the "target is best" desirability function was used (Equation 2). The segment specific target was defined as the mean of the in-vivo data and the upper and lower limits were defined as the in-vivo mean plus or minus the in-vivo standard deviation respectively (Equation 3). The overall desirability (Equation 4) is determined by calculating the geometric mean of the individual desirabilities. The implication of the multiplicative term in the equation for calculating the overall desirability is that if any of the individual desirabilities is outside the acceptable limits (one standard deviation of the mean) then the overall desirability is equal to zero (0).

Equation 2. “Target is Best” Desirability Function

$$d_i(\hat{Y}_i) = \begin{cases} 0 & ,if\ \hat{Y}_i(x) < L_i \\ \left(\dfrac{(\hat{Y}_i(x) - L_i)}{T_i - L_i}\right)^s & ,if\ L_i \leq \hat{Y}_i(x) \leq T_i \\ \left(\dfrac{(\hat{Y}_i(x) - U_i)}{T_i - U_i}\right)^t & ,if\ T_i \leq \hat{Y}_i(x) \leq U_i \\ 0 & ,if\ \hat{Y}_i(x) > U_i \end{cases}$$

Equation 3. Desirability function – definition of variables

$$Target = T_i = \text{In-vivo}_{mean}$$

$$Upper\ Limit = U_i = \text{In-vivo}_{mean} + \text{In-vivo}_{STD}$$

$$Lower\ Limit = L_i = \text{In-vivo}_{mean} - \text{In-vivo}_{STD}$$

$$s, t = weights$$

Equation 4. Overall Desirability

$$\text{D}=(d_1(Y_1) \times d_2(Y_2) \times d_3(Y_3) \times d_4(Y_4))^{1/4}$$

As reported previously, the in-vitro ROM was on average 12.3% smaller than the in-vivo data set. Therefore, for the primary analysis, the in-vivo and in-vitro data the in-vivo segmental kinematics were uniformly “scaled” by a factor of 87.7%. Additionally, it was hypothesized that the reason for the higher ROM in the in-vivo data set may be due to disagreement in the mechanism for determining the end ROM. In the in-vivo dataset the subjects were asked to rotate to a self-determined maximum stopping point, whereas in the in-vitro dataset end ROM was defined as +/- 2.0 Nm. Therefore, as an alternative to scaling the in-vivo data set a secondary analysis was performed wherein the in-vivo dataset was “trimmed” based on the average flexion and extension endpoints relative to the neutral position. For the in-vivo data set the neutral position was defined based on a patient selected comfortable starting position whereas for the in-vitro data set neutral position was defined as extension-flexion moment equal to zero. However, it should be noted that this resulted in a non-uniform trimming between extension and flexion with approximately10% of the overall trimming occurring on the extension side and approximately 2% of the trimming occurring on the flexion side.

6.3 RESULTS

6.3.1 Scaled to Match In-Vitro ROM

A series of ANOVA’s were performed based on percentage of ROM and segmental level. Figure 26 illustrates the percent contribution of the dependent variables FL and AL as well as the combined effects. Percent contribution is a relative scale and is not an indication of positive or negative effect relative to the in-vivo data. Overall, AL had the largest effect for the majority of the segments throughout the entire motion path. FL exhibited the strongest at 20% and 40% ROM; however, this effect was predominantly isolated to specific segments. CL had the strongest effect at 80% and 100% ROM, which represent middle to full flexion and again this effect was segment specific.

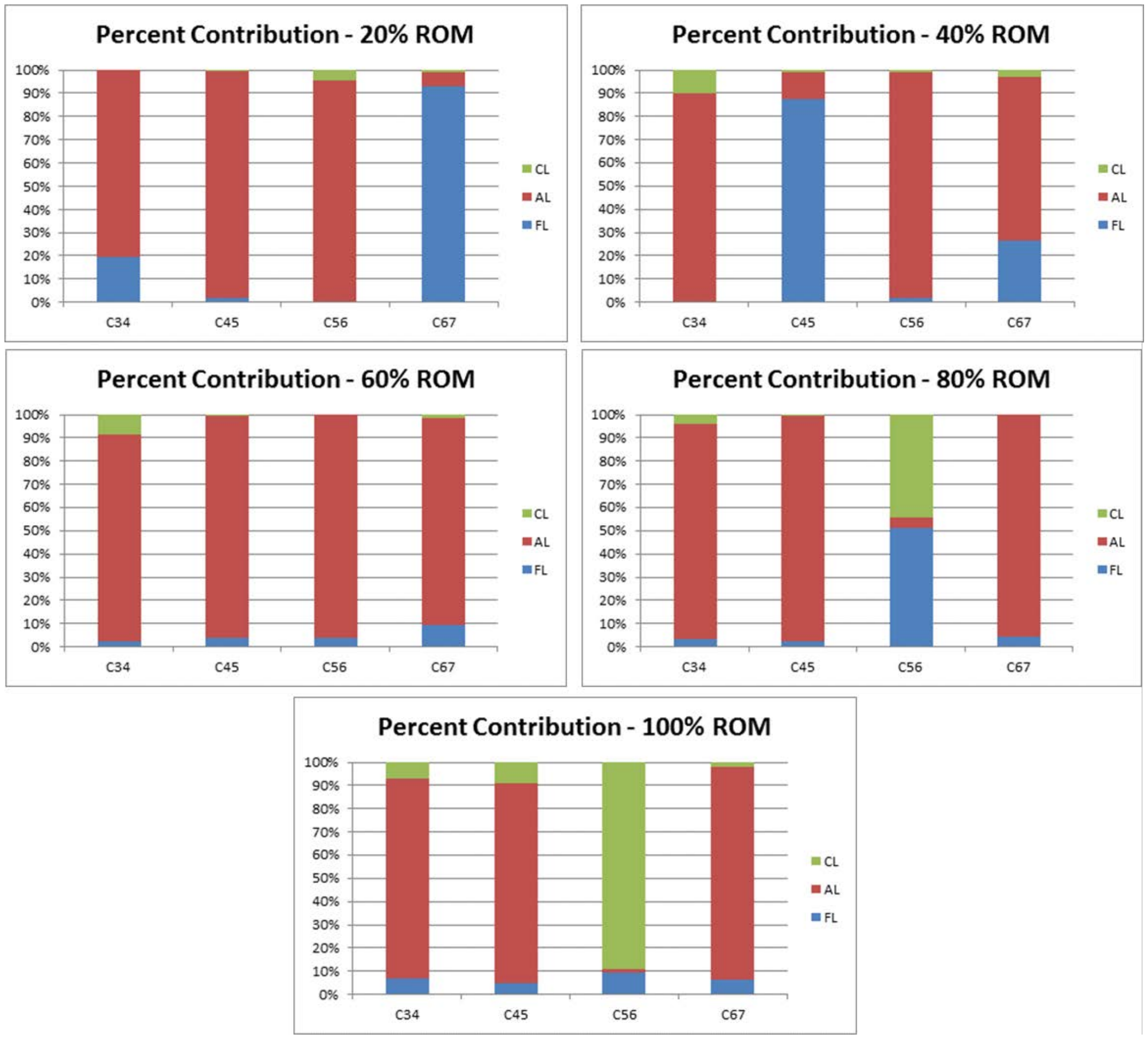


Figure 26. Percent Contribution of AL, FL and CL

The standardized effect size was plotted in Figure 27 and provides additional information compared to the percent contribution graphs. The advantage of the standardized effects graphs is that that results can be compared across graphs and that positive and negative effects are preserved. Overall, the application of AL was shown to have the largest effect in altering the segmental motion patterns. Similarly, to the percent contribution graphs the largest effects were observed throughout the middle of the motion path (40%, 60% and 80%). The effect of AL varied in magnitude throughout the motion path but the overall trend was preserved with a trend of increasing the ROM of C3-C4 and C4-C5 and decreasing the ROM of C5-C6 and C6-C7. The fact that the effect of applying AL was relatively large and the trend was preserved makes it an ideal candidate for optimization. A summary of the mean values for percent contribution and standardized effect size can be found in Table 7.

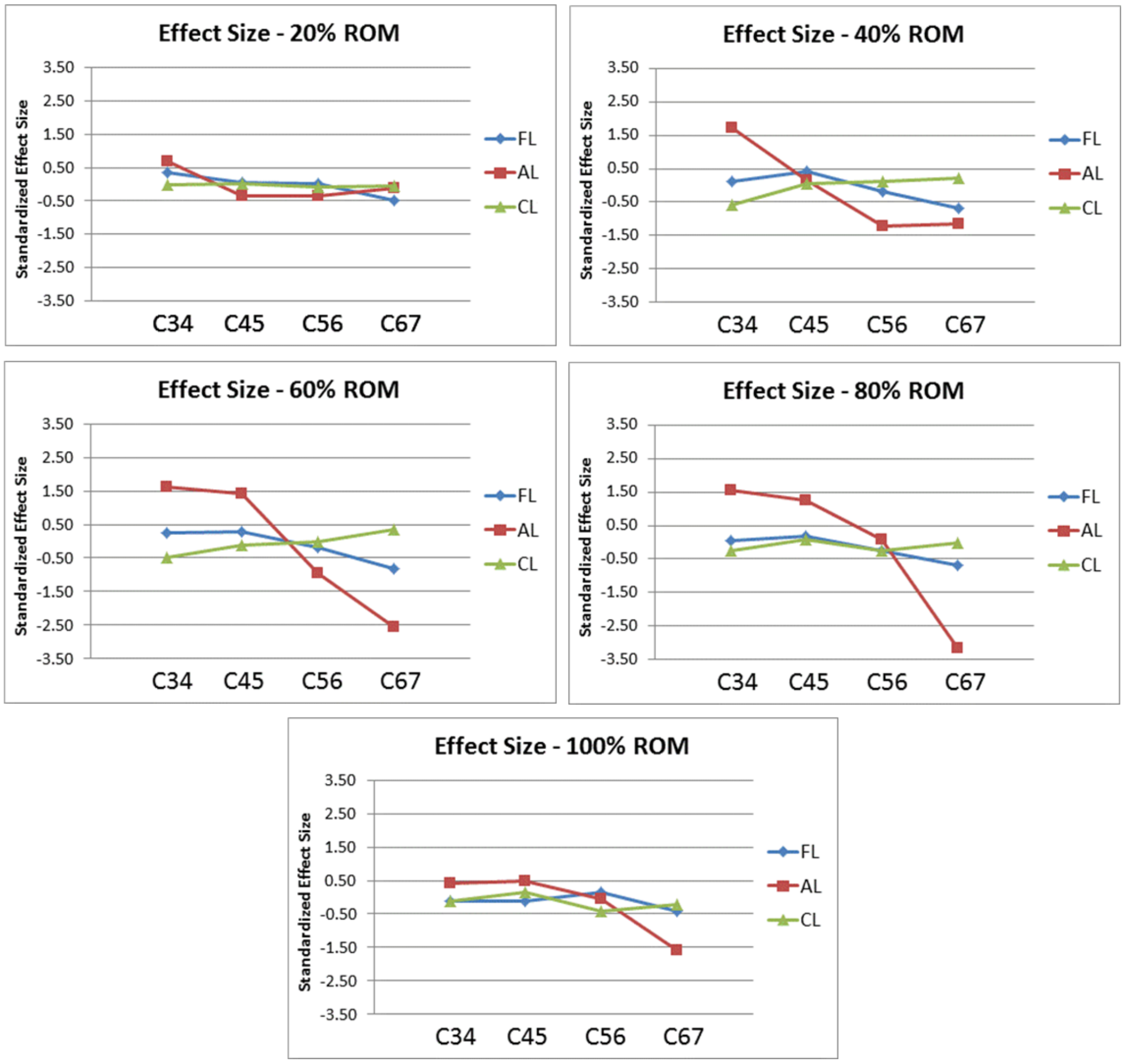
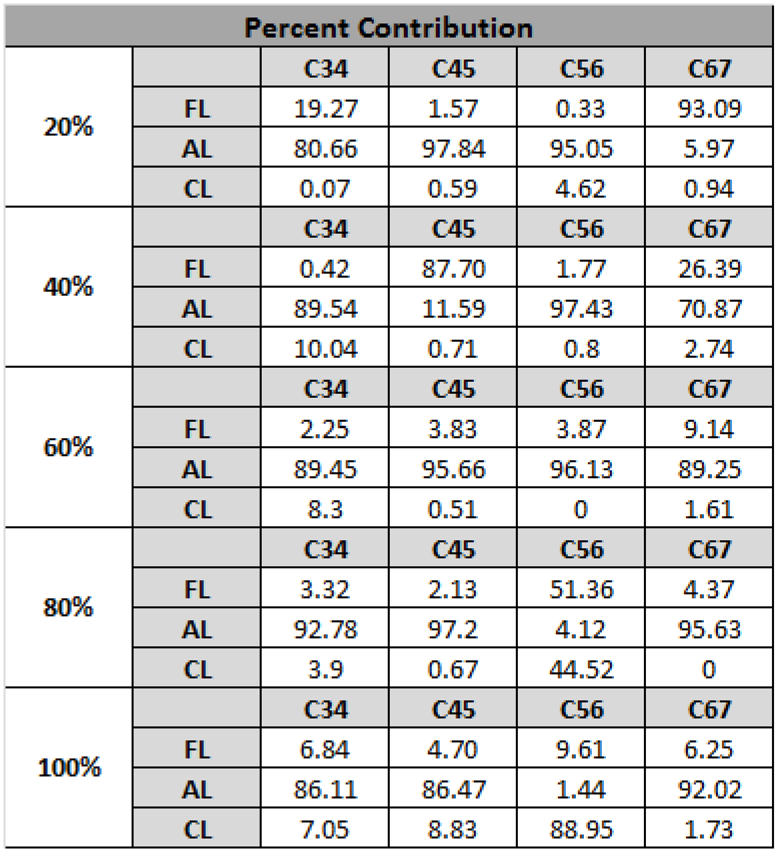


Figure 27. Standardized effect size

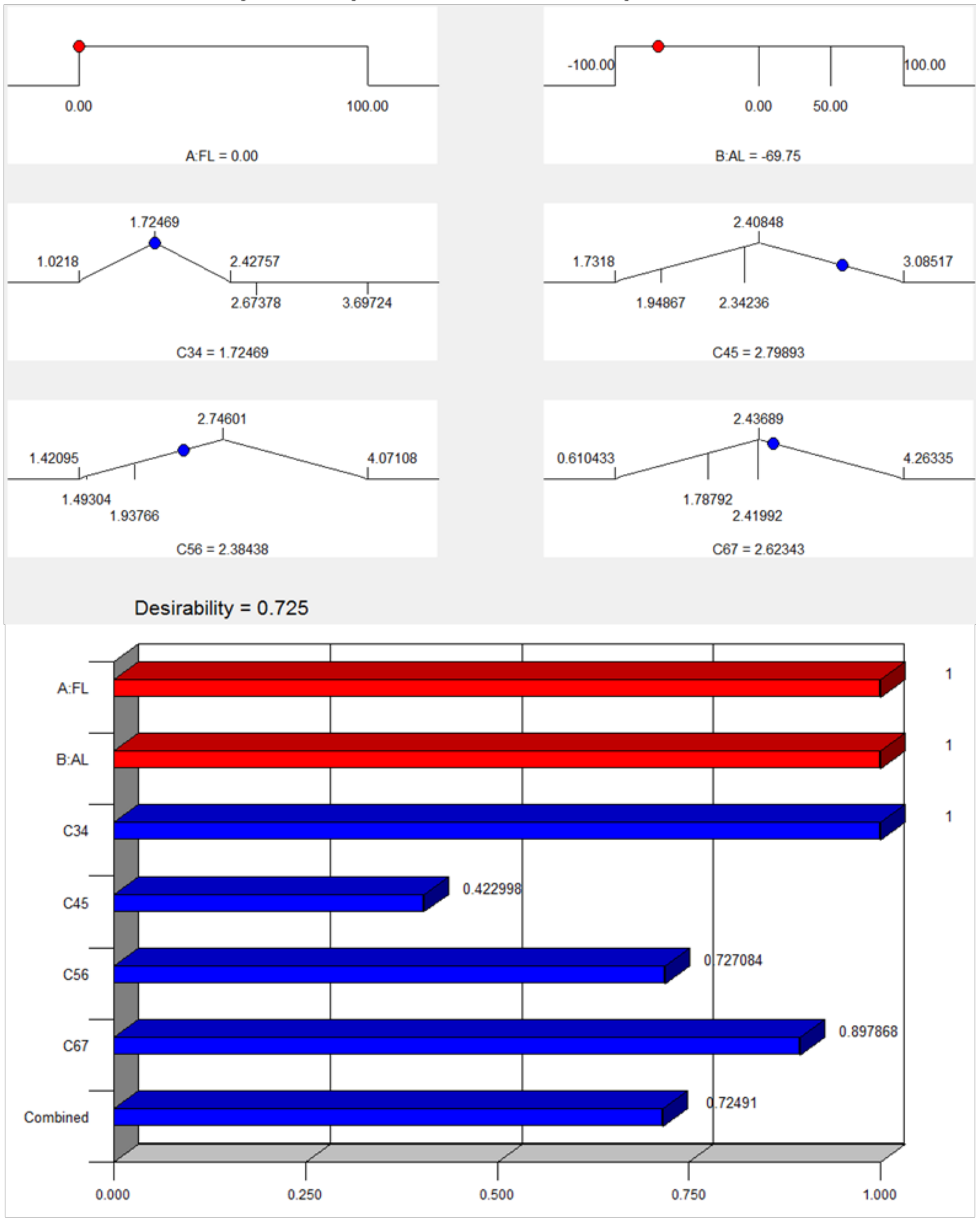
Table 7. Percent contribution



Percent Contribution					
20%		C34	C45	C56	C67
	FL	19.27	1.57	0.33	93.09
	AL	80.66	97.84	95.05	5.97
	CL	0.07	0.59	4.62	0.94
40%		C34	C45	C56	C67
	FL	0.42	87.70	1.77	26.39
	AL	89.54	11.59	97.43	70.87
	CL	10.04	0.71	0.8	2.74
60%		C34	C45	C56	C67
	FL	2.25	3.83	3.87	9.14
	AL	89.45	95.66	96.13	89.25
	CL	8.3	0.51	0	1.61
80%		C34	C45	C56	C67
	FL	3.32	2.13	51.36	4.37
	AL	92.78	97.2	4.12	95.63
	CL	3.9	0.67	44.52	0
100%		C34	C45	C56	C67
	FL	6.84	4.70	9.61	6.25
	AL	86.11	86.47	1.44	92.02
	CL	7.05	8.83	88.95	1.73

Figure 28-Figure 32 display the desirability ramps and bar graphs for 20%-100% ROM. Within the desirability ramps the values displayed above the ramps are indicative of the user input optimization parameters, whereas the values below the line indicate the range of the independent and dependent variables from the in-vitro dataset. For FL the optimization range and the tested range are equivalent whereas for AL the optimization range was extended beyond the testing range to include -100N to 100N of AL. The extended range was within the permissible limits of the design of experiments desirability function approach and deemed appropriated based on the nature of the AL response [110]. With the extended range for AL an acceptable optimized solution was determined throughout the motion path indicating that values for FL and AL were determined that resulted in C3-C4, C4-C5, C5-C6 and C6-C7 mean values within plus or minus one standard deviation of the in-vivo mean. The lowest individual desirability occurred at 20% ROM with C4-C5 having a desirability of 0.42. All remaining desirabilities were above 0.5 indicating a medium to large agreement with the in-vivo dataset. In many instances complete agreement was possible (desirability = 1).

Desirability Ramps and Bar Graph – 20% ROM



Desirability

Figure 28. Desirability ramps and bar graphs at 20% ROM

Desirability Ramps and Bar Graph – 40% ROM

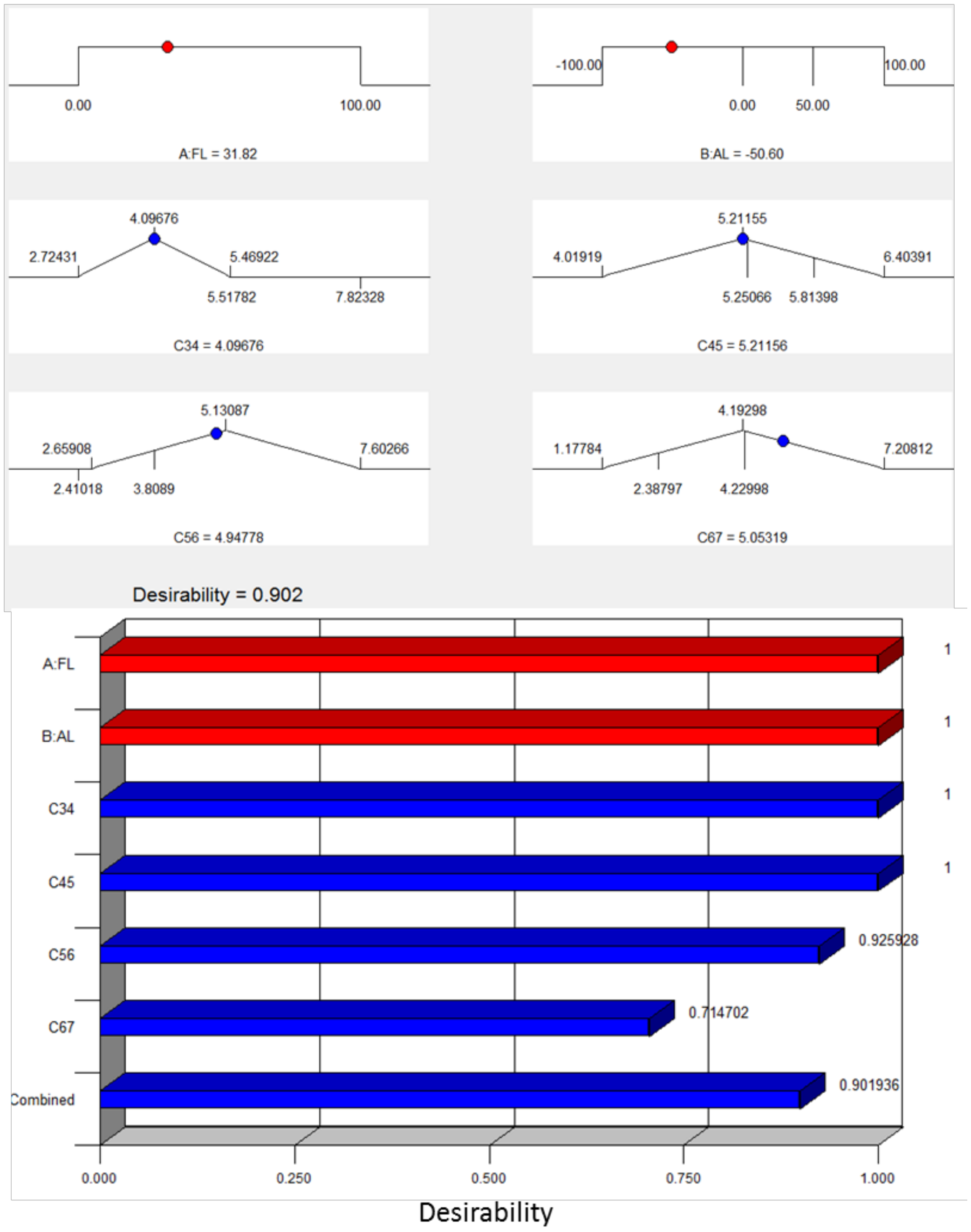


Figure 29. Desirability ramps and bar graphs at 40% ROM

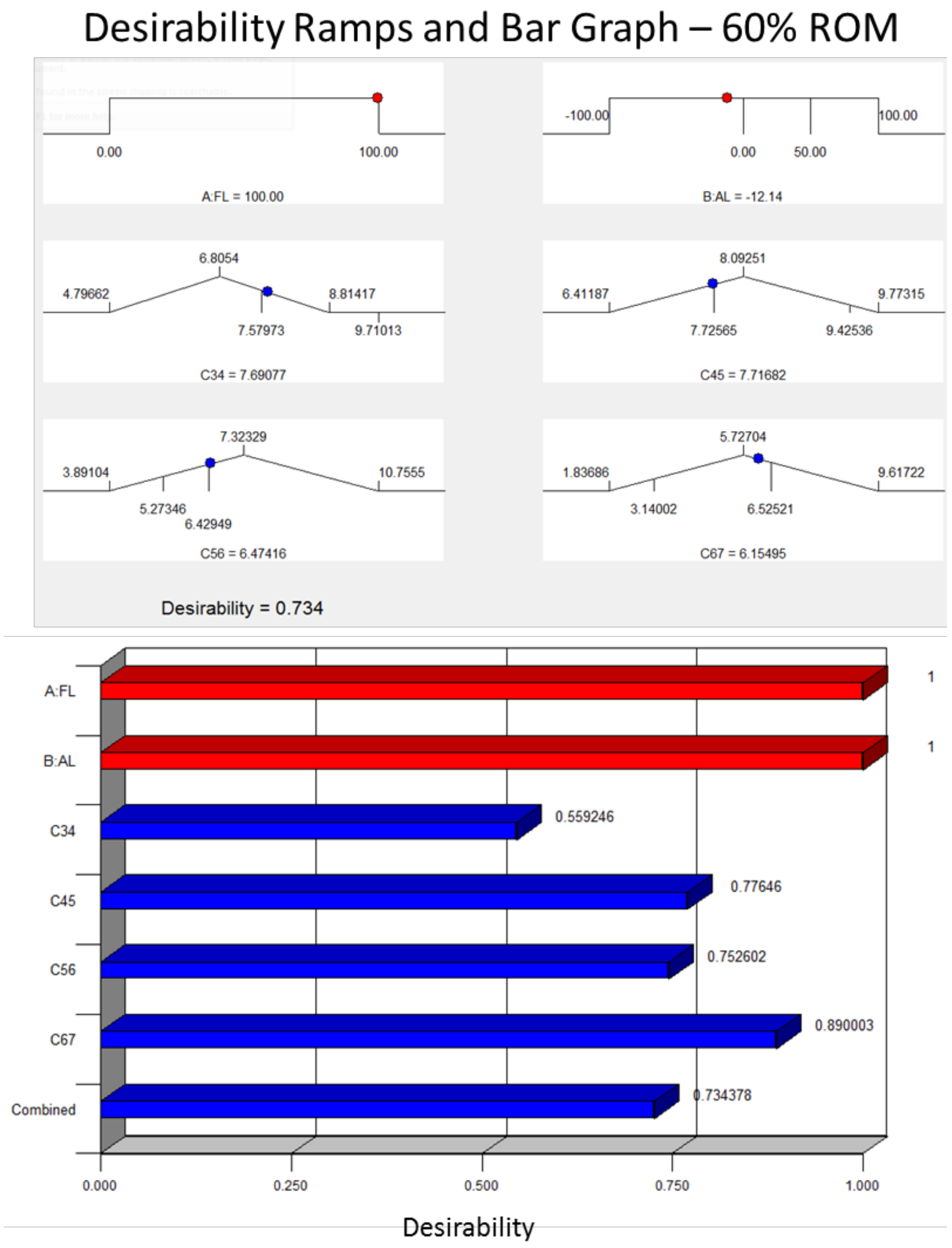


Figure 30. Desirability ramps and bar graphs at 60% ROM

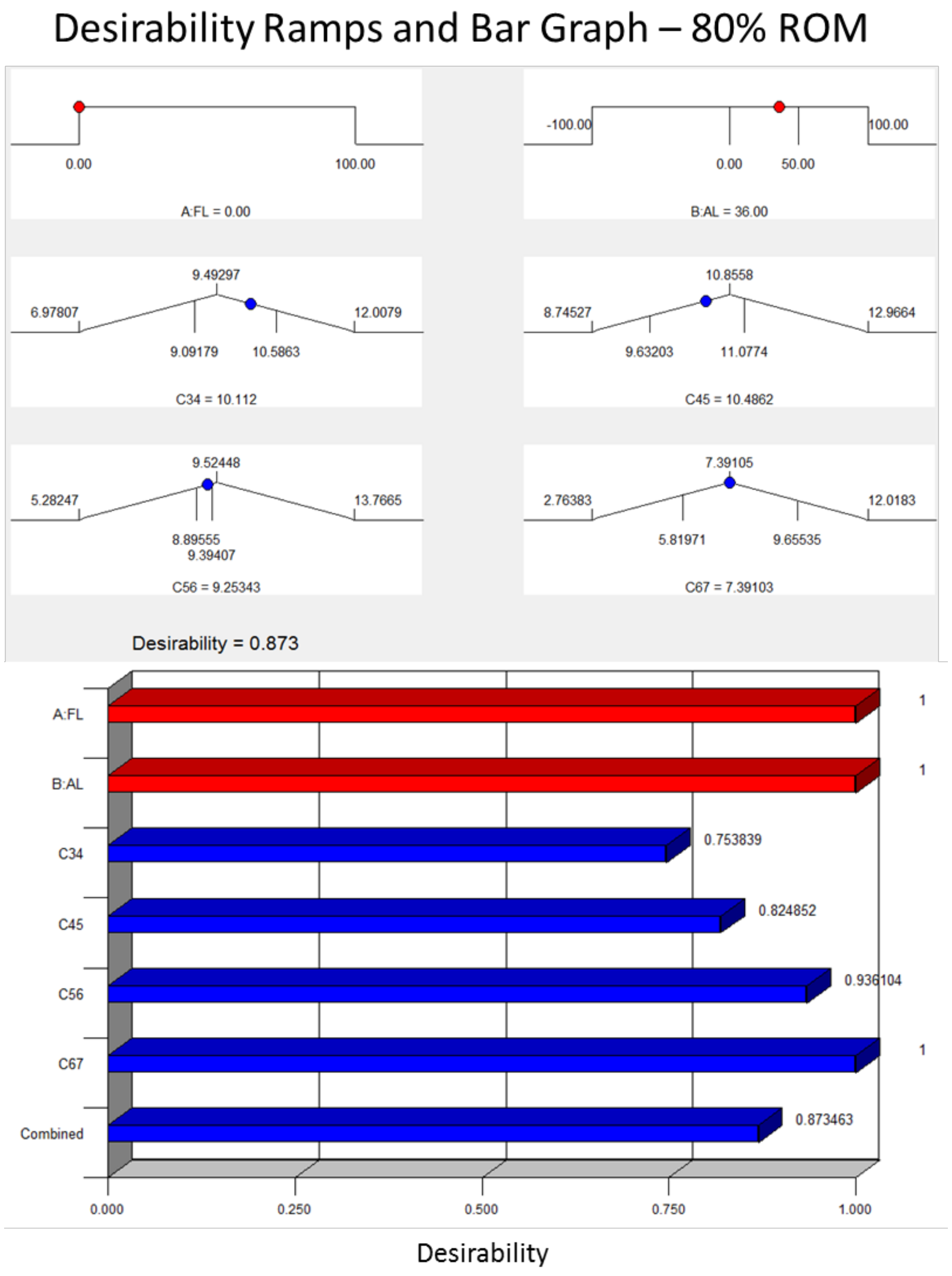


Figure 31. Desirability ramps and bar graphs at 80% ROM

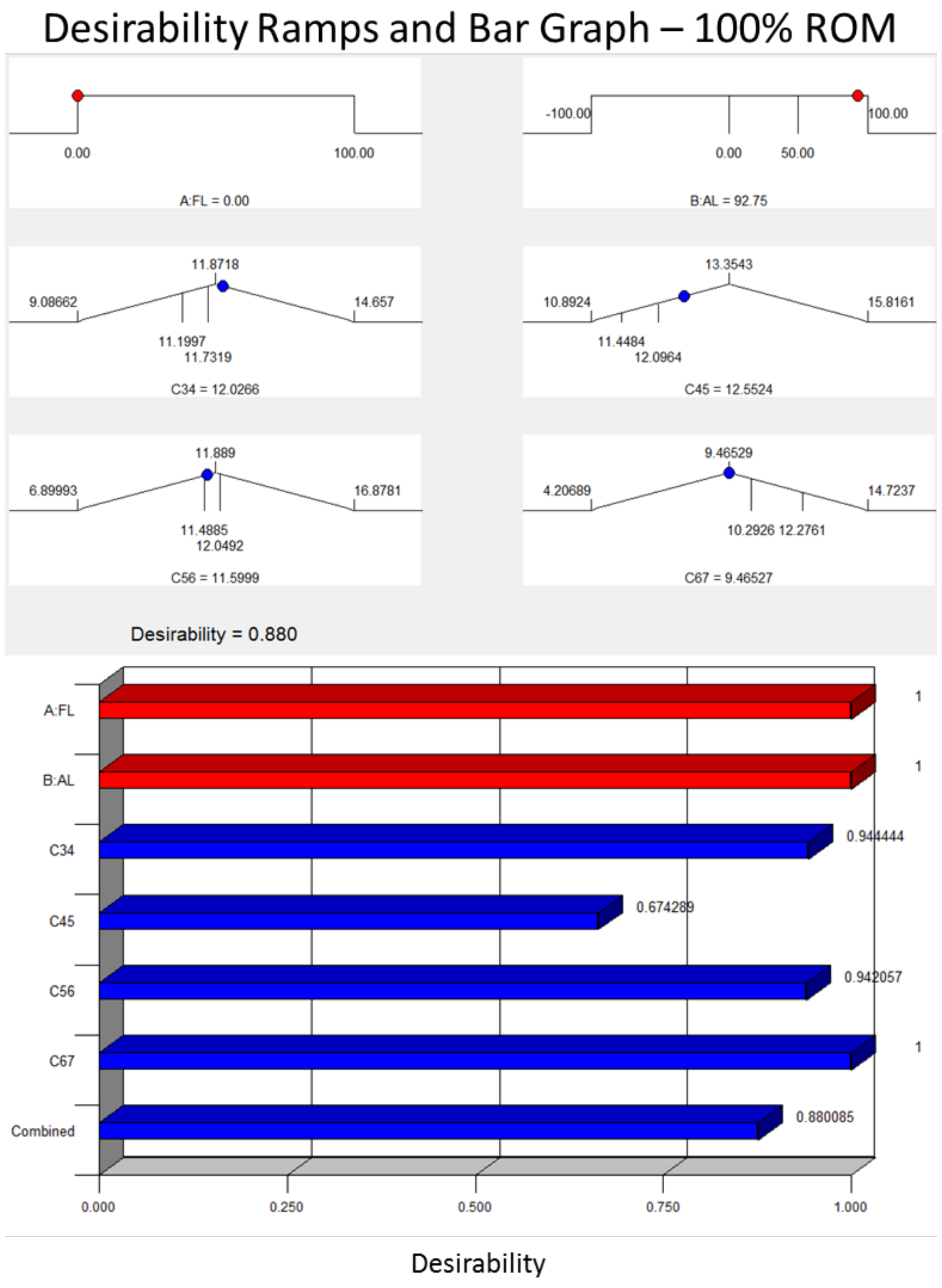


Figure 32. Desirability ramps and bar graphs at 100% ROM

Figure 33 summarizes the individual desirability as a bar graph of the overall desirability throughout the motion path. With regards to overall desirability, greater than 0.7 desirability was observed at all points in the motion path indicating large agreement with the in-vivo dataset. The lowest overall desirability was observed at 20% and 60% of the overall motion path and the best agreement was observed at 40%, 80% and 100% which all had close to a 0.9 overall desirability.

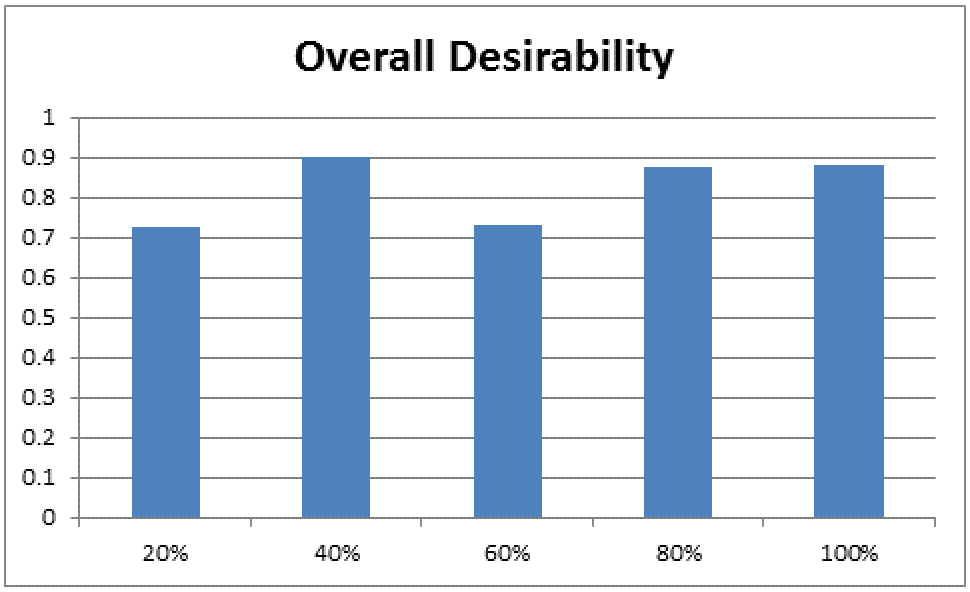


Figure 33. Overall desirability bar graph

Figure 34 displays the optimized loading parameters that were determined from the optimization procedure. FL was determined to be equal to zero at the extremes of the extension-flexion motion path but peaked near the middle portion of the motion path with a maximum value of 100N at 60% ROM. Conversely, AL increased linearly throughout the motion path with a range of -69.75N to 92.75N with the 60% ROM resulting in 0N of AL. A summary of the optimized independent variables, dependent variables and overall desirabilities in listed in Table 8.

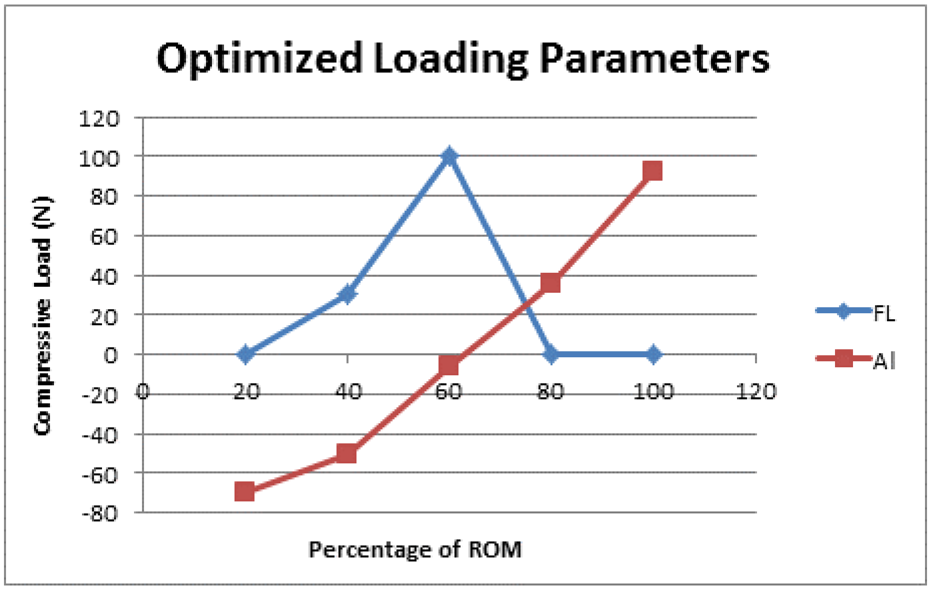


Figure 34. Optimized loading parameters

Table 8. Summary of optimized loading parameters

	Independent		Dependent				
	FL	AL	C34	C45	C56	C67	Desirability
20%	0.00	-69.75	1.72	2.80	2.38	2.62	0.72
40%	31.82	-50.60	4.10	5.21	4.95	5.05	0.90
60%	100.00	-5.92	7.89	7.89	6.35	5.84	0.73
80%	0.00	36.00	10.11	10.49	9.25	7.39	0.87
100%	0.00	92.75	12.03	12.55	11.60	9.47	0.88

Figure 35 displays a scatter plot of the optimized kinematics relative to the in-vivo mean values and the four in-vitro compressive loading states. It is clear that none of the un-optimized compressive loading states are individually able to replicate the in-vivo segmental motion patterns. However, the optimized compressive loading parameters are able to mimic the in-vivo segmental motion patterns throughout the entire motion path.

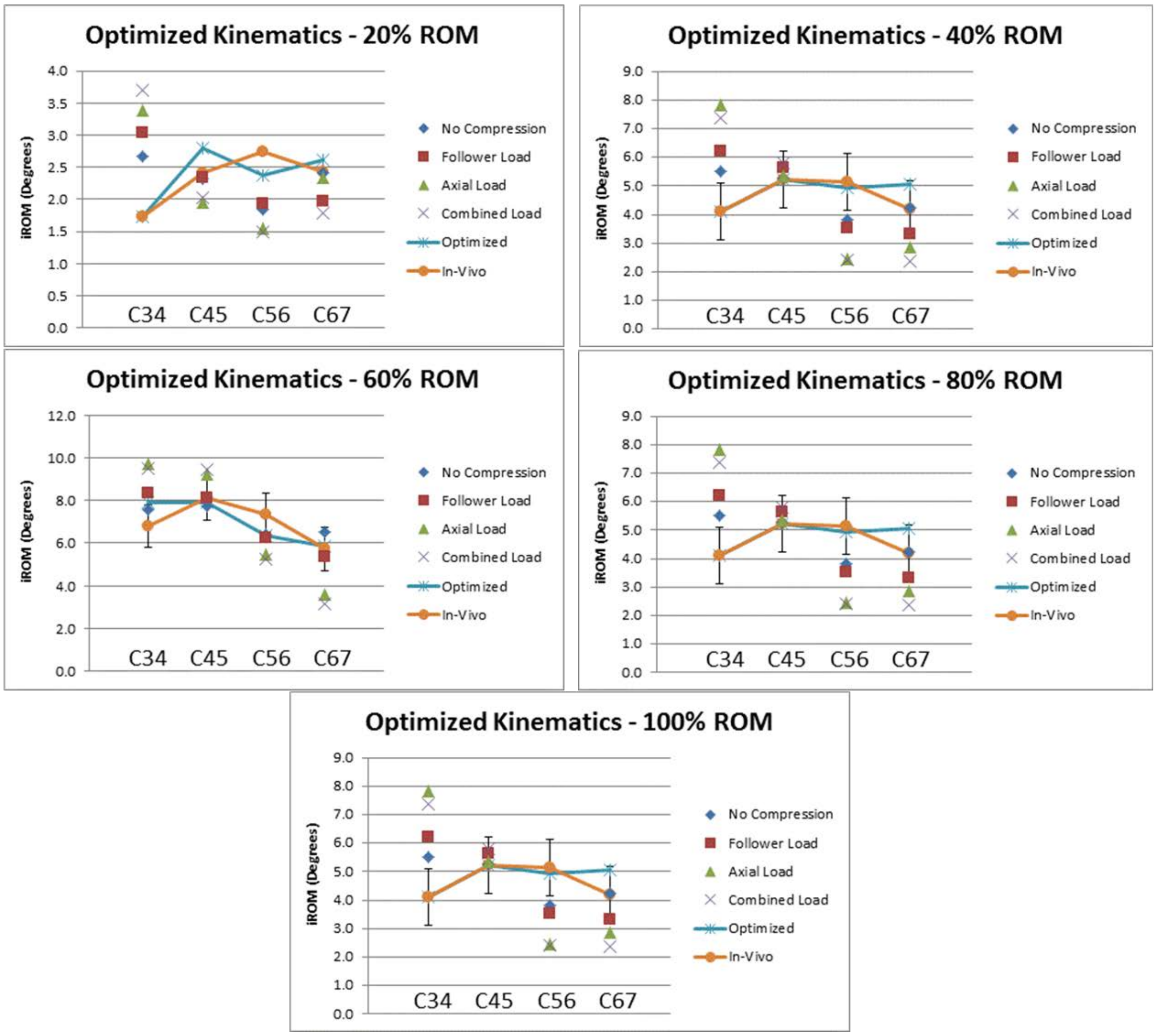


Figure 35. Optimized segmental kinematics

The agreement between compressive load states, including the optimized compressive loading parameters and the in-vivo segmental contributions were quantified using root mean square error (RMSE) and displayed in Figure 36. All compressive loading states had significantly higher RMSE ($p<0.05$) than the optimized loading parameters. Normalization of RMSE was performed and quantified as average percent error which is defined as the RMSE divided by the in-vivo mean ROM and displayed in Figure 37. Again, all un-optimized compressive load states had significantly higher ($p<0.05$) average percent error than the optimized compressive loading parameters. Additionally, the optimized compressive loading parameters were the only compressive loading states resulting in an average percent error less than 10%.

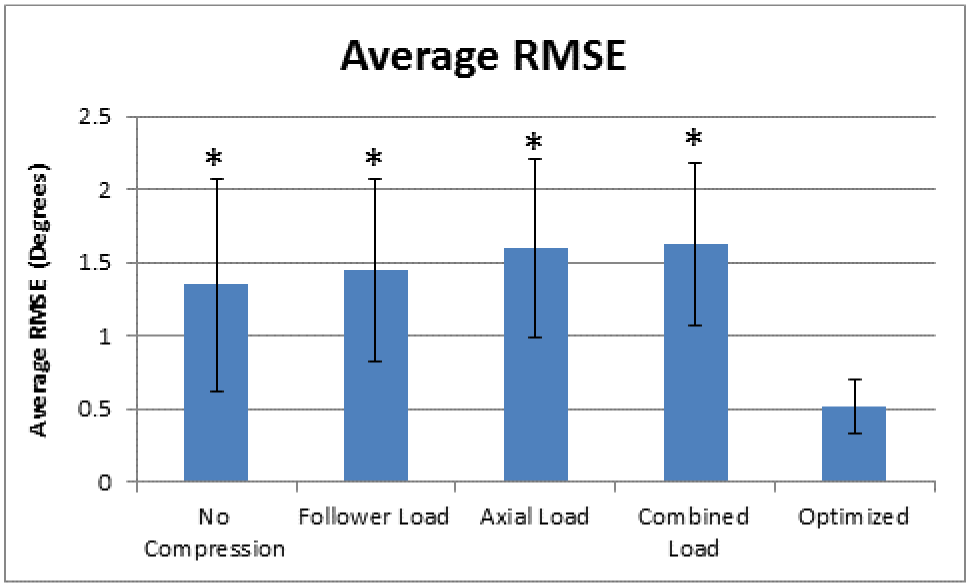


Figure 36. Average RMSE

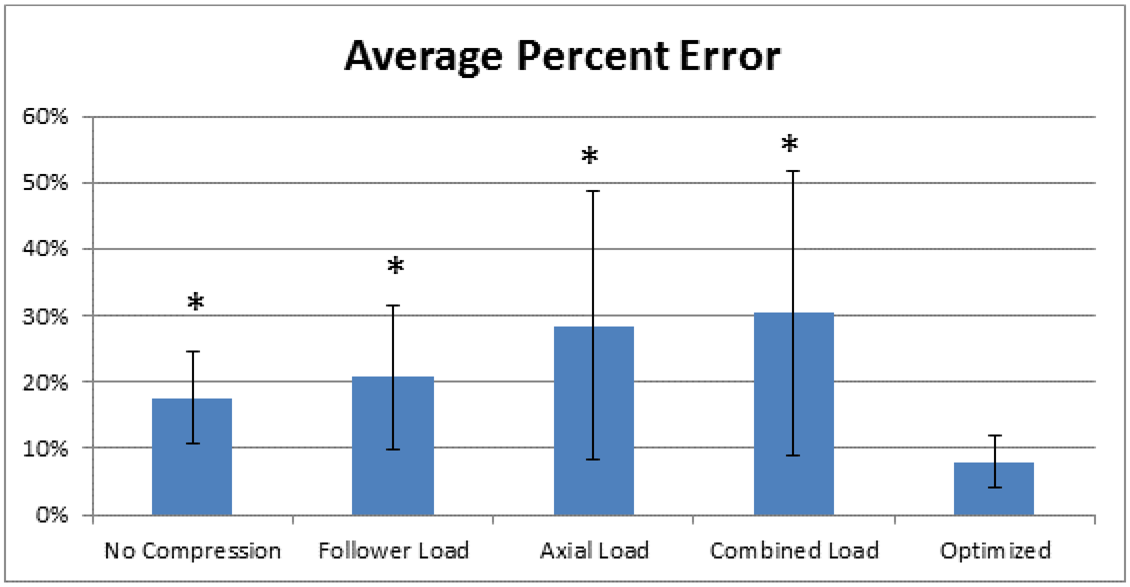
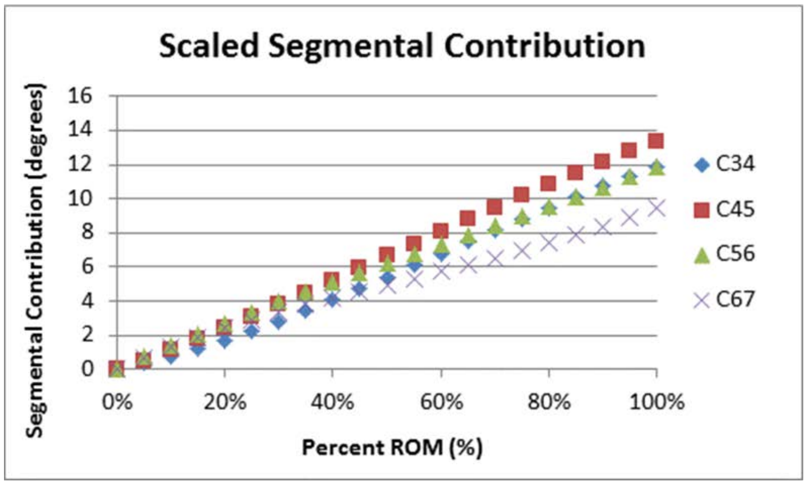


Figure 37. Average percent error

6.3.2 Trimmed to Match In-Vitro ROM

Figure 38 displays scatter plots of the “scaled” and “trimmed” in-vivo segmental kinematics as previously described. In the scaled segmental contribution plot the early portion of the curve (extension) has C4-C5, C5-C6 and C6-C7 contributing the most and C3-C4 lagging behind. When the early portion of the curve is trimmed off this effect is eliminated and a more continuous motion pattern is observed throughout the entire motion path. Additionally, more separation is observed in the trimmed segmental contribution plot in the later portion of the motion path (flexion). The overall result of the scaling and trimming was to ensure agreement between the combined C3-C7 overall ROM for the in-vivo and in-vitro datasets.



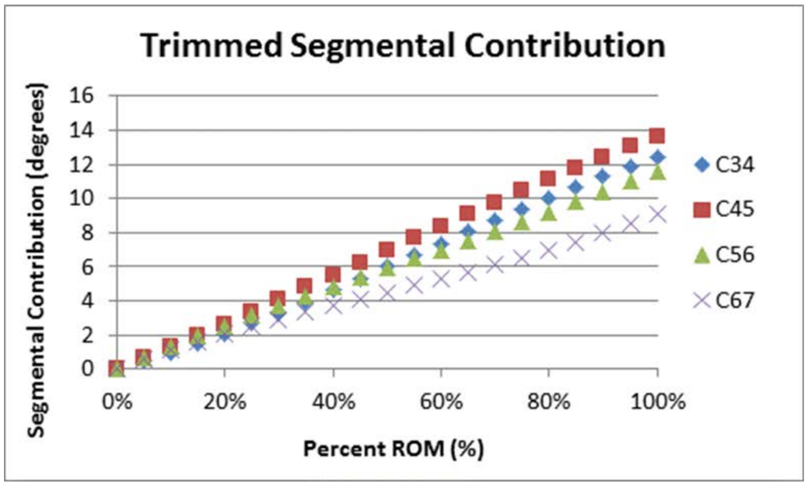


Figure 38. Scaled and trimmed segmental rotations

Optimization of the scaled segmental contribution data resulted in the large negative value for AL in the extreme extension portion of the motion path which was well outside of the tested parameters for the model. In an attempt to understand and avoid the large negative values for AL, optimization was performed with AL range restricted to 0N-100N. Using the scaled in-vivo dataset, restricting the AL to 0N-100N resulted in zero desirability at 20% and 40% ROM. Additionally, restricting the AL range from -100N to -10N resulted in zero desirability at 20% ROM and low (0.2) desirability at 40% ROM (Figure 39B). As stated previously, the trimming of the in-vivo dataset was non-uniform and primarily occurred on the extension side of the motion path—indicating that the trimmed dataset may strongly influence the optimization results in extension. The trimmed in-vivo segmental contribution data showed similar overall desirabilities to the scaled segmental contribution data when the AL range was preserved at -100N to 100N (Figure 39C). However, when the AL range was restricted to -10N to 100N acceptable optimized solutions were able to be found within one standard deviation of the in-vivo mean for all portions of the motion path (Figure 39D). Unfortunately, further restricting the AL to 0N-100N resulted in zero desirability at 20% ROM.

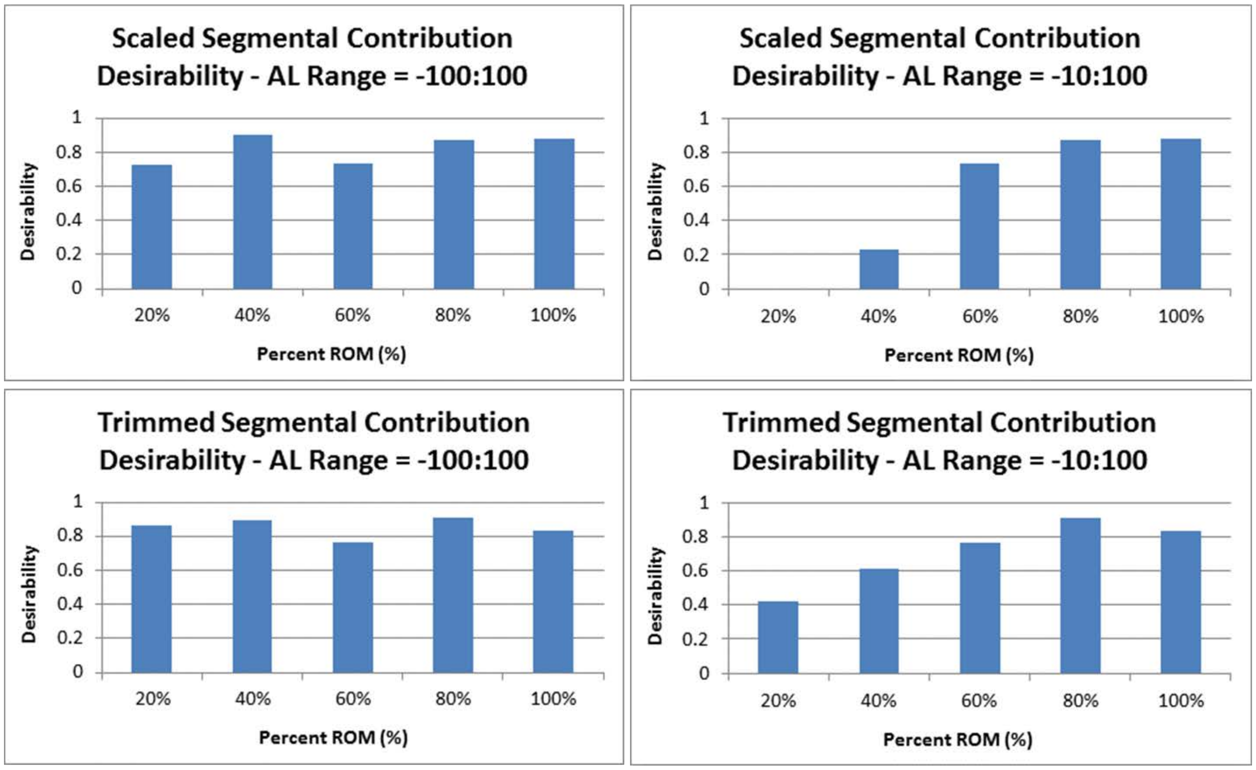


Figure 39. Desirability bar graphs with restricted AL range

The trimmed optimized loading parameters for the restricted AL range (-10N to 100N) are displayed in Figure 40. In comparison to the scaled optimized loading parameters the optimized values for FL still begin and end at 0N at the extremes of extension-flexion but the peak is now broadened and to extend across 40% to 60%. The optimized values for AL are now restricted to -10N and they non-linearly increase to a maximum value of 100N at 100% ROM.

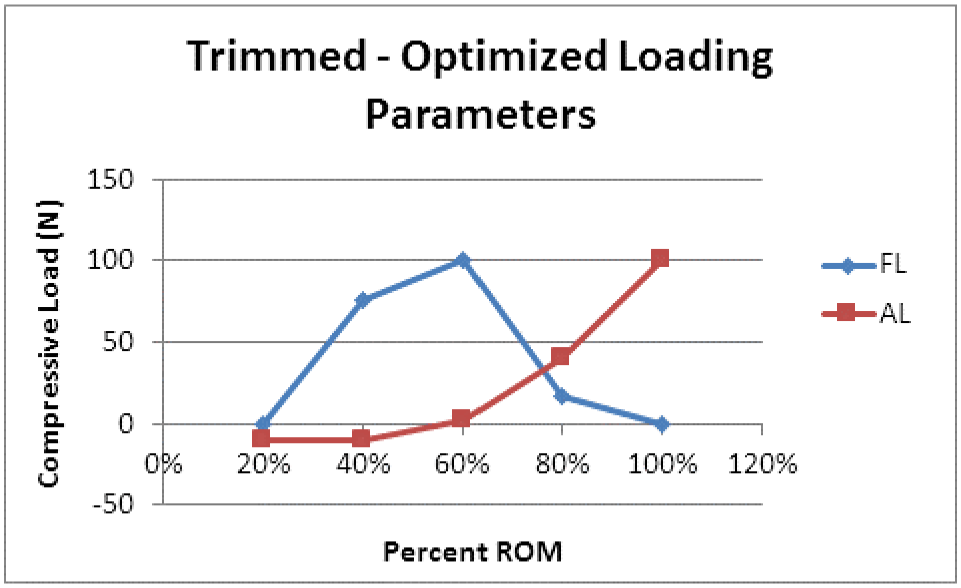


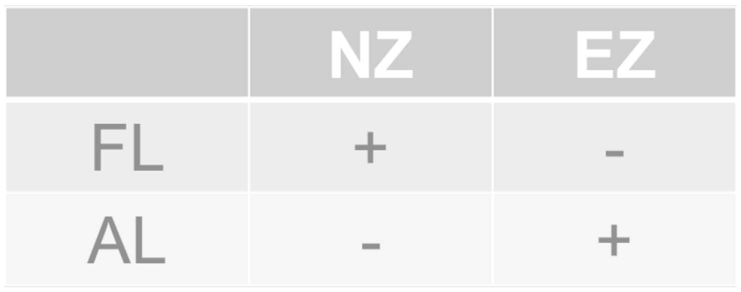
Figure 40. Trimmed optimized loading parameters

6.4 CONCLUSION

Previously, a systematic comparison of standard pure moment with no compressive loading versus published and novel compressive loading techniques (follower load - FL, axial load - AL, and combined load - CL) was performed. The pure moment testing protocol without compression or with the application of follower load was not able to replicate the typical in-vivo segmental motion patterns throughout the entire motion path. Axial load or a combination of axial and follower load was necessary to mimic the in-vivo segmental contributions at the extremes of the extension-flexion motion path. It was hypothesized that dynamically altering the compressive loading throughout the motion path is necessary to mimic the segmental contribution patterns exhibited in-vivo.

The systematic comparison of the compressive loading techniques was structured a priori using statistical design of experiments. The design of experiments methodology was also chosen based on the goal of determining the optimal compressive loading parameters using the desirability function approach. Optimization of the compressive load parameters enabled an optimized set of compressive loading parameters to be determined that resulted in-vitro segmental contributions that were within plus or minus one standard deviation of the in-vivo mean throughout the entire motion path. In terms average percent error the optimized compressive loading parameters resulted in in-vitro segmental contributions that were within 10% of the in-vivo mean. As hypothesized the values for the optimized independent variables of FL and AL varied dynamically throughout the motion path. FL was not necessary at the extremes of the extension-flexion motion path but peaked through the neutral position. Follower Load is critical through the “Neutral Zone” (NZ) for stability but detrimental in “Elastic Zone” (EZ). Whereas a large negative value of AL was necessary in extension and this parameter increased linearly to a large positive values in flexion. Axial Load critical in the EZ for appropriate sequencing but detrimental in NZ (Table 9).

Table 9. Summary of optimized AL and FL



	NZ	EZ
FL	+	-
AL	-	+

The linear increasing value for AL may be reflective of the in-vivo influence of increased muscular contribution with increasing extension-flexion. This is consistent with muscular models of the cervical spine which have shown that the resultant load experienced by the head increases with increasing extension-flexion [111-113]. In the present study the AL force was applied perpendicular to the superior most vertebral body (C3) which mimics the resultant muscular force on the head as it is transmitted to the vertebral column through the occiput.

The theoretical justification for increased muscular activity in flexion resulting in the need for increased AL breaks down when considering extension. The current model calls for high amount of negative AL in extension which cannot be explained by a model of muscular loading alone. It is also possible that facet contact which occurs in extension results in a lifting off of phenomenon with increased extension which could help explain why the model is predicting a negative AL in extreme extension [114]. However, future work is necessary to explore this theory and therefore a secondary analysis was performed wherein the in-vivo dataset was trimmed primarily on the extension side to align the magnitudes of the in-vivo and in-vitro data sets. The trimmed analysis had little effect on the comparison of the datasets in flexion but did enable the AL range to be restricted to -10N to 100N which is more consistent with the muscular loading model described above.

Another limitation of this study is the fact that the model includes only the osteoligmaentous structure of the lower cervical spine (C3-C7). A recent publication, based on the same in-vivo kinematics utilized for this study, reported that cervical spine intervertebral kinematics with respect to the head are different between flexion and extension . It was noted that the direction the head is moving should be accounted for when modeling muscle orientation and attachments [115]. It should also be noted that the upper cervical spine accounts for approximately 50% of the overall head to thoracic spine extension-flexion range of motion [116]. It is currently unknown if the testing methodology presented can sufficiently replicate boundary conditions adjacent to the lower cervical spine, in order to enable physiologic levels of loading to be applied. Future analysis of the in-vitro dataset should explore directionality for the muscle optimization and/or consider including additional motion segments. Additional structures present in-vivo have been removed in order to focus the testing on the osteoligamentous structures for in-vitro testing, such as muscle, fat, skin, etc. It is currently unknown how the removal of this tissue may affect the kinematics; however these removed structures do play a role in stabilization and end-range motion restriction. It is possible that the absence of these tissues could be contributing to the differences observed between the in-vivo and in-vitro datasets, or more specifically the high negative values for axial load predicted in the model.

In the present study the kinematic data was presented as a change in kinematics throughout the motion path from full extension to full flexion where full extension was defined as zero. This methodology was chosen due to lack of confidence in alignment of the neutral positions between the in-vivo and in-vitro datasets. In the in-vivo dataset the subject was asked to self-select a comfortable starting position whereas in the in-vitro dataset the neutral position was defined as the point of zero moment. In both scenarios subjectivity and sensitivity lead to low intra- and inter-testing repeatability in the neutral position making independent analysis of the extension and flexion tails of the motion path difficult. Additionally, a previous report focused on comparing in-vivo and in-vitro extension-flexion found in-vitro studies generally over estimate extension ROM and underestimate flexion ROM [117]. It should be noted however that the study performed by Adams et al. was performed in the lumbar spine and it is unclear if this observation can be applied to the cervical spine.

Future work aimed at validating the optimized parameters presented here is necessary to ensure confidence in the overall model. Future work should explore the effect of independent and combined application of dynamic FL and AL which will require increasing the range of the AL and exploring the influence of negative AL. Additionally, the present study performed optimization of the compressive loading parameters at percentages of the motion path leading to a proposed model of dynamic compression wherein all segments of the cervical spine construct would be subjected to the same loading parameters throughout the path of motion. However, optimization could also be performed such that an optimized set of compressive loads would be determined for each segment. Implementation of a segment specific loading scheme is less intuitive and may ultimately be difficult (or impossible) to achieve with the described loading schemes. However, it is theoretically possible to alter the moment each segment is experiencing throughout the motion path by optimizing/altering the line of action of the AL. This segment specific optimization would be best performed using a computational model of the cervical spine with simulated compression and moments [118-120] and is also the subject of future work.

Future work should also aim to further refine the resolution for the comparison between the in-vivo and in-vitro data and the resulting optimization. Currently, comparison only occurs at 20% increments throughout the extension-flexion path and although the predicted optimized load parameters appear to change continuously throughout the motion path the intermediate data has not been analyzed. Anderst et al. a mixed-model analysis to model the percent contributions from each motion segment throughout the motion path [100]. Future work aimed at optimizing the loading parameters based on a continuous model would highlight one of the largest strengths of this study which is direct access to dynamic muscle driven in-vivo kinematics.

7.0 CONCLUSION

The overall objective of this project was to utilize statistical design of experiments to systematically compare various compressive loading strategies with the goal of determining the optimal compressive loading conditions that would enable a robot/UFS testing system to recreate in-vivo cervical spine segmental kinematics. Based on this objective the following hypotheses were generated for evaluation:

H1a: Increasing magnitude of follower load will result in a uniform moment variation across the spine construct leading to overall stiffening (stability) of the spinal construct.

H1b: Increasing magnitude of axial load will result in a non-uniform moment variation across the spine construct leading to increased ROM of middle segments.

H1c: Combining axial compression with follower load will have a synergistic effect enabling (+/- 10%) agreement with recorded / published extension-flexion range of motion.

Prior to performing the systematic comparison, the robot/UFS testing system, needed to be upgraded to ensure the system was capable performing flexibility testing on multi-level cervical spine specimen at physiologic levels of compressive preload. The robot/UFS testing system was previously validated for use in lumbar spine functional spinal units (FSU), however, significant development was necessary to extend the capabilities of the system. The first step, described in chapter 2.0 , was to ensure that the robot/UFS testing system could perform a flexibility test on a cervical spine FSU using the hybrid control algorithm. The hybrid control algorithm enabled the robot/UFS testing system to apply pure moments to an FSU (in extension-flexion, lateral bending, or axial rotation) in an unconstrained manner through active control of secondary translational/rotational degrees-of-freedom—successfully minimizing coupled forces/moments. The characteristic nonlinear S-shaped curves of the primary moment-rotation responses were consistent with previous reports of the FSU having a region of low stiffness (neutral zone) bounded by regions of increasing stiffness (elastic zone). Direct comparison of "displacement control" and "hybrid control" showed that hybrid control was able to actively minimize off-axis forces and resulted in larger neutral zone and range of motion.

Once it was confirmed that the robot/UFS testing system could successfully perform a flexibility test on a cervical FSU the next step was to validate the system's ability to perform multi-level cervical spine testing at physiologic levels of compressive preload. The most commonly reported method of applying a compressive preload is the follower load methodology. Follower load is applied to the specimen along the natural curvature of the spinal construct, through the segmental centers of rotation. This mode of application enables the specimen to withstand physiologic levels of compressive loading, while preserving the pure moment assumption. Therefore, a follower load system was developed with actively controlled linear actuators and integrated into the robot/UFS testing system's hybrid control algorithm. Thorough investigation of the integrated system ensured that the pure moment assumption was upheld and enabled characterization of the kinetics resulting from the application of follower load.

Although numerous authors are actively using follower loading in the cervical spine for instrumentation testing, very little data exists on the effect of follower loading on normal (intact) cervical spine moment-rotation parameters. Therefore, in chapter 3.0 , an actively controlled follower load system was integrated with a robot/UFS testing system to explore the effect of follower load on multi-segment cervical spine moment-rotation parameters. Application of follower load significantly increased neutral zone stiffness, neutral zone width, and hysteresis. These findings are consistent with literature and confirm hypothesis H1a and the hypothesis presented by Patwardhan et al. (2000) that follower load increases the stability of the cervical spine, enabling flexibility testing at physiologic magnitudes of compressive preload to be performed. In chapter 4.0 the effects of applying follower load on the cervical spine intradiscal pressure (IDP) was explored. The effects of adding FL compression approximates the effect of the patient going from the supine to seated position. A high significant correlation between IDP – extension delta and disc height was also found which indicates that facet off-loading is increased with degeneration, limiting the change in IDP in extension, a finding not previously reported that may give insight into the mechanics of IDD.

The results of Chapter 2.0 - 4.0 ensured that the robot/UFS testing system was capable of performing the current standard cervical spine biomechanics testing methodology reported in literature. However, in addition to follower load, various other methods of applying compressive loading have been reported in the literature including axial forces produced with inclined loading plates, eccentric axial force application, as well as attempts to individually apply/model muscle forces in-vitro. The importance of proper compressive loading to recreate the segmental motion patterns exhibited in-vivo has been highlighted in previous studies. However, appropriate methods of representing the weight of head and muscle loading are currently unknown. Therefore, the objective of this project was to identify and verify the appropriate *in-vitro* loading conditions that would replicate the *in-vivo* kinematics and kinetics of the cervical spine, with the overall goal of improving the biofidelity of the experimental platform.

In chapter 5.0 , a systematic comparison of standard pure moment with no compressive loading versus published and novel compressive loading techniques (follower load - FL, axial load - AL, and combined load - CL) was performed. The present study is unique in that a direct comparison to continuous cervical kinematics over the entire extension- flexion motion path was possible. Confirming hypothesis H1a, follower load resulted in uniform moment variation across the spine. As a result the pure moment testing protocol without compression or with the application of follower load was not able to replicate the typical in-vivo segmental motion patterns throughout the entire motion path. Confirming hypothesis H1b, axial load resulted in a non-uniform moment variation across the spine construct and therefore axial load or a combination of axial and follower load was necessary to mimic the in-vivo segmental contributions at the extremes of the extension-flexion motion path. However, axial load and combined axial and follower were detrimental through the neutral position. Therefore, a (+/- 10%) agreement with in-vivo segmental kinematics was not achieved and hypothesis H1c was not confirmed. This data suggests that dynamically altering the compressive loading throughout the motion path is necessary to mimic the segmental contribution patterns exhibited in-vivo, leading to a new hypothesis:

> H2a: Dynamic alteration of the compressive load magnitude will enable segmental temporal sequencing of vertebral movement to be adjusted throughout the motion path resulting in (+/- 10%) agreement with recorded data.

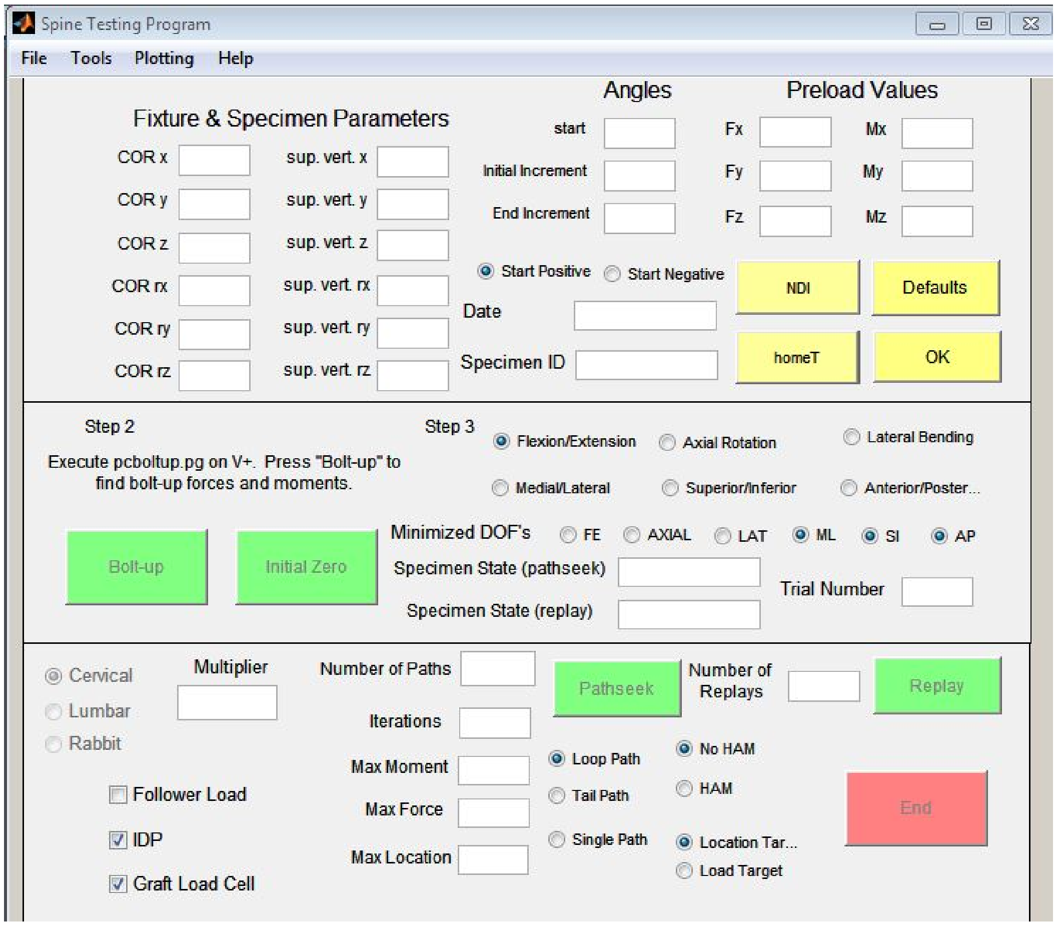
A model for dynamically altering the compressive loading throughout the motion path was explored in chapter 6.0 through optimization of the “target is best” desirability function at 20% increments of the overall extension-flexion motion path. An optimized set of compressive loading parameters was determined that resulted in-vitro segmental contributions within one standard deviation of the in-vivo mean throughout the entire motion path. In terms average percent error the optimized compressive loading parameters resulted in in-vitro segmental contributions that were within 10% of the in-vivo mean, confirming hypothesis H2a. As hypothesized the values for the optimized independent variables of follower load and axial load varied dynamically throughout the motion path. Follower load was determined to be critical through the neutral zone for stability but detrimental in elastic zone. On the contrary, axial load was shown to be critical in the elastic for appropriate sequencing but detrimental in neutral zone (Table 9). Although validation of the optimized parameters is necessary to ensure confidence in the dynamic loading model, the region specific (neutral zone / elastic zone) dependency of the loading parameters is consistent with models of muscular loading at rest and with active extension-flexion.

Future work should explore the effect of independent and combined application of dynamic FL and AL which will require increasing the range of the AL and exploring the influence of negative AL. Ultimately, the goal is to utilize the validated dynamic compressive loading model in assessments of spinal instrumentation function, especially for motion sparing devices such as total disc replacements, wherein the goal is restoration of normal physiologic motion. This future work is critical in response to the finding that pure-moment with no compressive preload or with follower loading fail to accurately reproduce in-vivo kinematics through the entire motion path and may lead to incorrect assessments of spinal instrumentation.

APPENDIX A

MATLAB CODE

A.1 ROBOT/UFS TESTING SYSTEM CONTROL ALGORITHM



Execute pcboltup.pg on V+. Press "Bolt-up" to find bolt-up forces and moments.

[✓ Start Positive] Start Negative

[✓ Flexion/Extension] Axial Rotation Lateral Bending Medial/Lateral Superior/Inferior Anterior/Poster...

Minimized DOF's: FE AXIAL LAT [✓ ML] [✓ SI] [✓ AP]

[✓ Cervical] Lumbar Rabbit

Follower Load [✓ IDP] [✓ Graft Load Cell]

[✓ Loop Path] Tail Path Single Path

[✓ No HAM] HAM

[✓ Location Tar...] Load Target

Figure 41. Graphical User Interface for Robot Control

A.1.1 Path Seek – Primary Robot Control Program

```
% Pathseek_Spine.m
% Kevin Bell
% Updated - 05/27/2013

pause on

% Disable buttons on GUI until Pathseek_Spine.m is done running
buttons_Spine(guihandles, 'off');

% TURN FOLLOWER LOAD ON
FL_flag = FL_value; %ON = 1, OFF = 0;
IDP_num = IDP_value; %OFF = 0, one sensor = 1, etc.
GLC_num = GLC_value; %ON = 1, OFF = 0;
vicon = 0; %ON = 1, OFF = 0;
vicon_pause = 0; %ON = 1, OFF = 0 - physicall pause / resume Vicon recording
pauselength = 2.0; % typically set to 5 - pause t
rFSU_flag = 1;  %ON = 1, OFF = 0;
CDI_flag = 0; % Need to add to GUI
clear ljHandle


%Labjack must be working
if exist('ljHandle') == 0
   Labjack_Test_U3;   %assigns in & defines ljHandle, LJ_ioGET_AIN
end

% to allow VICON to record enough data (if 5 hz = 25 point, could increase
% rate?)
timer1 = 0;
timer_period = .19999;

if timer1 == 1
   Timer_counter = 0;
   T_VICON = timer('TimerFcn', 'Timer_VICON','ExecutionMode','FixedRate',...
      'Period',timer_period);
end

% Setup naming for structures
date_ID = [dated '_' ID];

if LAT_ang < 0
   strLAT_ang = ['_' num2str(abs(LAT_ang))];
else
```

```
    strLAT_ang = num2str(LAT_ang);
end
if FE_ang < 0
    strFE_ang = ['_' num2str(abs(FE_ang))];
else
    strFE_ang = num2str(FE_ang);
end
if AXIAL_ang < 0
    strAXIAL_ang = ['_' num2str(abs(AXIAL_ang))];
else
    strAXIAL_ang = num2str(AXIAL_ang);
end

current_angles = (['LAT' strLAT_ang '_FE' strFE_ang '_AXIAL' strAXIAL_ang]);

% Input dialog box to get the filename for data storage
default_path = ['c:\Spine Testing\Data\' date_ID];
prompt = {'Enter Filename'};
title = 'Filename';
lines = 1;
def = {default_path};
answer = inputdlg(prompt,title,lines,def);
if isequal(answer,{}) == 1
    % Enable buttons on GUI
    buttons_Spine(guihandles, 'on');
else
    filename = answer{1};
end

% Clear variables created for inputdlg
clear prompt title lines def answer;

% initialize stiffness, target f/m, temp. f/m, temp positions Stiffness
% stiff = [64 82 205 2479 2787 879]; stiff = [100 100 100 10 10 10]; stiff
% = [1 1 1 10 10 10];
stiff = [250 250 250 500 500 500]; % Human Lumbar, cervical FSU, Sheep, thoracic
% stiff = [125 125 125 250 250 250];  %multisegemental human cervical stiff
% = [250 250 250 50 50 50]; %single FSU rabbit lumbar

if rFSU_flag == 1
    stiff = [250 250 250 50 50 50]; %single FSU rabbit lumbar
end

z_stiff = [100 100 100 10 10 10];
z_flag = [0 0 0 0 0 0];
z_stop = [120 120 120 9 9 9];
```

```
f_temp = [0 0 0 0 0 0];
p_temp = [0 0 0 0 0 0 0 0 0 0 0 0]';

% initialize iterations
z_ct = 1;      % keeps track of no. of iterations to reach min. force
z_count = num_iterations;    % limit to z_count iterations
z_ct_temp = z_count;  % keeps track of no. of iterations to reach min. force
z_step = 1;     % index to keep track of what direction and angle data gathered was at
z_xform = 1;    % index to keep track of global c.s. to tool c.s. xform info sent to Matlab
z_mom_flag = 1;     % how many rotation angles the moment > max.mom
z_index = 1;    % index to keep track of number of iterations per angle
temp800 = 0;
% define the limits for displacement, rotation, f/e moment and pathseek
% limit
lim_dis = .1; % mm
lim_mdis = .05; % degrees
% initialize direction
dir_flag = 0;   % change direction if dir_flag <> 0
step_counter = 0; % counter to keep track of number of steps for a given direction
startnum = 1;
posloop = startpos_value;
T_counter = 0;
T_dir_counter = 0;
sd_flag = 0;
final_loop = 0;
overall_counter = 0;
cancel = 0;
fuzzyone = 1;
mult_counter = 0;
delta_error_all(1:6) = 0;
delta_error = 0;
% Edited on 2/17/2012 - from 4 -> 2 -> 1
force_multiplier2(1:3) = mult;
delta_error_abs_all(1:6) = 0;
fuzzy_calc(1:6) = 0;
delta_error_abs = 0;
delta_err_abs = 0;
IDPmean = [0,0,0,0];
Loadcell_Pressure_True_N = 0;
%added 8/29/12
robot_return = 0;

if fuzzyone == 1
   % Fuzzy Logic Initialization
   Fuzzy_force = readfis('Fuzzy_force_new');
   delta_err = -30;
```

```
    err = 0;
end

if fuzzyone == 1
    % Fuzzy Logic Initialization
    Fuzzy_moment = readfis('Fuzzy_moment_new');
    delta_err = -3;
    err = 0;
end

stable_flag = 0;

% initialize timer
tic;

if FL_flag == 0
    % setup figure to graphically monitor loads
    [fx, fy, fz, mx, my, mz, handles, fh] = pathseek_display_Spine1;
    [handlesLD, fhLD] = pathseek_LDdisplay_Spine1(pathtype); %-added rah-8/29/12
elseif FL_flag == 1
    % setup figure to graphically monitor loads - including FL
    [fx, fy, fz, mx, my, mz, handles, fh, FLh] = FL_display_Spine1;
%     [handlesLD, fhLD] = pathseek_LDdisplay_Spine1(pathtype);
end

% send x1, y1, z1, rx1, ry1, rz1 to V+ to make tool transformation
ok = 0;
flag = 0.1;
fprintf(port1,'%f\n', [ok, flag]);
fprintf(port1,'%f\n', [(x1*1000)+.001, (y1*1000)+.001, (z1*1000)+.001,...
    rx1+.001, ry1+.001, rz1+.001]);

done_moving = fscanf(port1);
% ================================
timeout_Spine;
% ================================
done_moving = sscanf(done_moving, '%f');

if vicon == 1
    VICON_U3_OpenLabJack
    % Start VICON
    VICON_U3_Start
    if timer1 == 1
        start(T_VICON);
        timertic=tic;
    end
    if vicon_pause ==1
        pause(.1)
        VICON_U3_Pause
    end
end

% Apply Follower Load if FL_flag == 1
if FL_flag == 1
    FLvar = 'FL';
    increment_function_Spine;
    cycle = 1;
    clear Newton1 Newton2 FLcount_total
    FLcount_total = 0;
    Follower_Load_Control_Fuzzy;
end

%rFSU testing: asymmetric step sizing & moment target- modifying
%Spine_display var's
if rFSU_flag == 1
    w_neg = -w_ang;
    w_neg_end = -w_ang_end;
    max_mom_pos = max_mom;
    max_mom_neg = 0.15;
end


for cycle = 1:num_paths

    %'sequencenum' defines different types of paths based on
    %single/tail/loop and which type of loop based on number of cycles
    if pathsequence == 1
        if cycle == 1 | cycle == num_paths;
            sequencenum = 3 %will move through 3 paths: 0_pos, pos_neg, neg_pos
            if cycle == num_paths
                final_loop = 1; %final_loop=0 by default; =1 marks last path
            end
        end
        if cycle == 1 & cycle == num_paths; %one path only
            sequencenum = 4 %will move through 4 paths: 0_pos, pos_neg, neg_pos, pos_0
            if cycle == num_paths
                final_loop = 1;
            end
        end
        if cycle ~= 1 & cycle ~= num_paths; %"in-b/w" paths (not first or last)
            sequencenum = 2 %will move through 2 paths: pos_neg, neg_pos (loop)
        end
```

```
elseif pathsequence == 2
   sequencenum = 2; %will move through 2 paths: 0_pos, 0_neg (tail)
elseif pathsequence == 3
   sequencenum = 1; %will move through 1 path
end

for path_counter = 1:sequencenum

   if cancel == 0

      overall_counter = overall_counter + 1;

      % Increment Function
      increment_function_Spine

      % Index to keep track of endpoints
      endpt_index = 0;
      step_index = 0;
      first_index = 1;
      T_dir_counter = 0;

      if pathsequence ~= 1
         now = w_start;
      elseif pathsequence == 1 & overall_counter == 1
         now = w_start;
      end

      % changes direction if direction flag <> 0 (reachess final
      % increment or exceeds max moment)
      while dir_flag == 0

         robot_return = 0; %if in the while loop (pathseeking),

         endpt_index = endpt_index + 1;

         for n = 1:6
            z_sign(n) = 0;
            z_flag(n) = 0;
         end

         %================================================
         Load_Control_Fuzzy;  % Load control (inner) loop
         %================================================

         %display LD curve after load control loop finishes (final
         %<fm_tare6.m> should contain force/moment of this cycle
```

```
pathseek_LDdisplay_Spine2(fm_tcs, handlesLD, now, pathtype,...
    posloop, robot_return, fhLD, cycle) %-added rah - 8/29/12

if vicon == 1
    if vicon_pause == 1
        VICON_U3_Resume
        pause(pauselength)
        VICON_U3_Pause
    else
        pause(pauselength)
    end
end

z_ct = z_ct_temp;

if GLC_num == 1
    Loadcell_Pressure_True;
    Spine.(date_ID).(state).(current_angles).(pathtypestr)...
        .pathseek(trial).(HAM_str).(pathsequence_str)(cycle)...
        .(path_name).GLC(endpt_index)=Loadcell_Pressure_True_N;
end

% Get IDP measurements from IDP 202
if IDP_num > 0
    % Need to replace with new getIDP code
    get_IDP;
    % Build array of ((path_name)) IDP1
    Spine.(date_ID).(state).(current_angles).(pathtypestr)...
        .pathseek(trial).(HAM_str).(pathsequence_str)(cycle)...
        .(path_name).IDP(endpt_index,:) = IDPmean;
end

if FL_flag == 1
    Spine.(date_ID).(state).(current_angles).(pathtypestr)...
        .pathseek(trial).(HAM_str).(pathsequence_str)(cycle)...
        .(path_name).FLoad(1,endpt_index) = N1;
    Spine.(date_ID).(state).(current_angles).(pathtypestr)...
        .pathseek(trial).(HAM_str).(pathsequence_str)(cycle)...
        .(path_name).FLoad(2,endpt_index) = N2;
end

if CDI_flag == 1
    comm_cdi_function;
    Spine.(date_ID).(state).(current_angles).(pathtypestr)...
        .pathseek(trial).(HAM_str).(pathsequence_str)(cycle)...
        .(path_name).cdi_translation(endpt_index) = cdi_data;
```

```
end

%++++++++++++++++++ Data Storage
%++++++++++++++++++++++++++++++++++
% Build array of ((path_name)) positions data that could be
% for replay
Spine.(date_ID).(state).(current_angles).(pathtypestr)...
   .pathseek(trial).(HAM_str).(pathsequence_str)(cycle)...
   .(path_name).replay_global_pos(1:6,endpt_index) = Spine...
   .(date_ID).(state).(current_angles).(pathtypestr)...
   .pathseek(trial).(HAM_str).(pathsequence_str)(cycle)...
   .(path_name).global_pos(1:6,step_index);
% Build array of ((path_name))jt angles data that could be
% for replay
Spine.(date_ID).(state).(current_angles).(pathtypestr)...
   .pathseek(trial).(HAM_str).(pathsequence_str)(cycle)...
   .(path_name).replay_jt_angles(1:6,endpt_index) = Spine...
   .(date_ID).(state).(current_angles).(pathtypestr)...
   .pathseek(trial).(HAM_str).(pathsequence_str)(cycle)...
   .(path_name).jt_angles(1:6,step_index);
% Build array of ((path_name))rotation angles end point
% data
Spine.(date_ID).(state).(current_angles).(pathtypestr)...
   .pathseek(trial).(HAM_str).(pathsequence_str)(cycle)...
   .(path_name).rot_angle_end_pts(endpt_index) = now;
% Build array of ((path_name))load end point data
Spine.(date_ID).(state).(current_angles).(pathtypestr)...
   .pathseek(trial).(HAM_str).(pathsequence_str)(cycle)...
   .(path_name).load_end_pts(1:6,endpt_index) = Spine...
   .(date_ID).(state).(current_angles).(pathtypestr)...
   .pathseek(trial).(HAM_str).(pathsequence_str)(cycle)...
   .(path_name).load(1:6,step_index);
% Build array of ((path_name))stiffness end point data
% -added by RAH 5/16/12
Spine.(date_ID).(state).(current_angles).(pathtypestr)...
   .pathseek(trial).(HAM_str).(pathsequence_str)(cycle)...
   .(path_name).stiff_end_pts(1:6,endpt_index) = Spine...
   .(date_ID).(state).(current_angles).(pathtypestr)...
   .pathseek(trial).(HAM_str).(pathsequence_str)(cycle)...
   .(path_name).stiff(1:6,step_index);
% Build array of ((path_name)) step index
Spine.(date_ID).(state).(current_angles).(pathtypestr)...
   .pathseek(trial).(HAM_str).(pathsequence_str)(cycle)...
   .(path_name).step(endpt_index) = step_index;

% MINIMUM LOAD DATA STORAGE Build array of minimum
```

```
    % ((path_name)) positions data that could be for replay
    Spine.(date_ID).(state).(current_angles).(pathtypestr)...
        .pathseek(trial).(HAM_str).(pathsequence_str)(cycle)...
        .(path_name).replay_global_pos_min(1:6,endpt_index)...
        = min_global_pos;
    % Build array of minimum ((path_name))jt angles data that
    % could be for replay
    Spine.(date_ID).(state).(current_angles).(pathtypestr)...
        .pathseek(trial).(HAM_str).(pathsequence_str)(cycle)...
        .(path_name).replay_jt_angles_min(1:6,endpt_index)...
        = min_jt_angles;
    % Build array of minimum ((path_name))load end point data
    Spine.(date_ID).(state).(current_angles).(pathtypestr)...
        .pathseek(trial).(HAM_str).(pathsequence_str)(cycle)...
        .(path_name).load_end_pts_min(1:6,endpt_index) = min_load;
    % Build array of minimum ((path_name)) step index
    Spine.(date_ID).(state).(current_angles).(pathtypestr)...
        .pathseek(trial).(HAM_str).(pathsequence_str)(cycle)...
        .(path_name).step_min(endpt_index) = min_step;
    %++++++++++++++++++++++++++++++++++++++++++++++++

    %==========================================
    max_moment_Spine; % max moment loop
    %==========================================

    if dir_flag == 0  % if target is not reached - continue with current direction
        if HAM_value == 1
            % Use HAM to update the COR for displacement
            % control
            Rotate_HAM; %updates 'now'
            if sd_flag == 1
                inc = temp_inc; %resets 'inc' to value
                sd_flag = 0;
                disp(inc);
            end
        elseif HAM_value == 0
            % Do not use HAM - just use original fixed COR
            Rotate_noHAM; %updates 'now'
            if sd_flag == 1
                inc = temp_inc;
                sd_flag = 0;
            end
        end
    end

end
```

```
% now = w_start;
dir_flag = 0;

% Calculates the "tool motion" relative to the initial position
tool_motion_Spine

if pathsequence == 2 || pathsequence == 3
   % Step back through replay to starting position
   dialog = 1;
   robot_return = 1;
   zero = 0;
   dir_flag = 0;
   now = w_start;

   while zero == 0 & dialog == 0;
      button = questdlg(['Do you want to return robot to' ...
         'starting position?'],'Continue');
      if strcmp(button,'Yes')
         dialog = 1;
         robot_return = 1;
         now = w_start;
      elseif strcmp(button,'No')
         dialog = 1;
         zero = 1;
         display('[Pathseek cancelled because you did not'...
            'return robot to starting position'])
         robot_return = 0;
         cycle = num_paths + 1;
         path_counter = sequencenum + 10;
      elseif strcmp(button,'Cancel')
         dialog = 1;
         zero = 1;
         display('Program has been canceled')
         robot_return = 0;
         cycle = num_paths + 1;
         path_counter = sequencenum + 10;
         cancel = 1;
      end
   end

   if robot_return == 1
      load_return_ctr = 1;
      for rr_indx = size(Spine.(date_ID).(state).(current_angles)...
            .(pathtypestr).pathseek(trial).(HAM_str).(pathsequence_str)...
            (cycle).(path_name).replay_global_pos,2):-1:1
            ok = 0;
            flag = 5.1;
            fprintf(port1,'%f\n', [ok, flag]);
            reverse = Spine.(date_ID).(state).(current_angles)...
                .(pathtypestr).pathseek(trial).(HAM_str)...
                .(pathsequence_str)(cycle).(path_name)...
                .replay_global_pos(1:6,rr_indx);
            fprintf(port1,'%f\n',  reverse);
            done_moving = fscanf(port1);
            % ================================
            timeout_Spine;
            % ================================
            done_moving = sscanf(done_moving, '%f');

            %=========================================
            get_loads; % measure: forces and moments
            %=========================================

            %=========================================
            fm_tare6; % tare out bolt-up and fixture wt
            %=========================================

            Spine.(date_ID).(state).(current_angles).(pathtypestr)...
                .pathseek(trial).(HAM_str).(pathsequence_str)(cycle)...
                .(path_name).load_return_end_pts(1:6,load_return_ctr)...
                = transpose(fm_tcs);

            pathseek_LDdisplay_Spine2(fm_tcs, handlesLD, (Spine...
                .(date_ID).(state).(current_angles).(pathtypestr)...
                .pathseek(trial).(HAM_str).(pathsequence_str)(cycle)...
                .(path_name).rot_angle_end_pts(rr_indx)), pathtype,...
                posloop, robot_return, fhLD, cycle) %-added rah - 8/29/12

            load_return_ctr = load_return_ctr + 1;

            if vicon == 1
                if vicon_pause == 1
                    VICON_U3_Resume
                    pause(pauselength)
                    VICON_U3_Pause
                else
                    pause(pauselength)
                end
            end
        end
    end
```

```
            end
        end
    end

end


display(['the current angle is: ' num2str(now)]);

if pathtype == 1
    ML_DA = now;
    set(guihandles.ML_DA_edit,'String',ML_DA)
elseif pathtype == 2
    SI_DA = now;
    set(guihandles.SI_DA_edit,'String',SI_DA)
elseif pathtype == 3
    AP_DA = now;
    set(guihandles.AP_DA_edit,'String',AP_DA)
elseif pathtype == 4
    FE_ang = now;
    set(guihandles.FE_ang_edit,'String',FE_ang)
elseif pathtype == 5
    AXIAL_ang = now;
    set(guihandles.AXIAL_ang_edit,'String',AXIAL_ang)
elseif pathtype == 6
    LAT_ang = now;
    set(guihandles.LAT_ang_edit,'String',LAT_ang)
end

pause off

% Apply Follower Load if FL_flag == 1
if FL_flag == 1
    FLvar = 'FunL';
    increment_function_Spine;
    cycle = 1;
    Follower_unLoad_Control_Fuzzy;
end

if vicon == 1
    VICON_U3_Stop
    if timer1 == 1
        stop(T_VICON);
    end
end
```

```
% Save workspace
save(filename)
disp('Data has been saved.')

% Ending sounds alarm
load gong.mat
sound(y,Fs)

% Enable buttons on GUI when Pathseek_Spine.m is done running
buttons_Spine(guihandles, 'on');
```

A.1.2 Load Control – Function to Minimize Off-Axis Loads

```
% Load_Control_Fuzzy.m
% Kevin Bell
% March 17, 2005

z_ct = 1;
minimized = 1;
three_counter = 2;

time = toc;
tic;

delta_err_abs_all(1:6) = 0;

% limit to z_count iterations (will want to change to time limit)
while z_ct < z_count + 1

   %Cumulative index of every iteration
   step_index = step_index + 1;

   %============================================
   get_loads; % measure: forces and moments
   %============================================

   %============================================
   fm_tare6; % tare out bolt-up and fixture wt
   %============================================

   ok = 0;
   flag = 1.1;
   fprintf(port1,'%f\n', [ok, flag]);
   gt_jt_angles = fscanf(port1);
```

```
    gt_jt_angles = sscanf(gt_jt_angles, '%f');

    if FL_flag == 0
        % display f/m after taring out bolt-up and fixture wt
        pathseek_display_Spine2([fm_tcs, fx, fy, fz], [mx, my, mz], handles,...
            [now, z_ct, cycle, z_target],IDPmean, Loadcell_Pressure_True_N, fh);
    elseif FL_flag == 1
        % setup figure to graphically monitor loads - including FL
        % Get Follower Loads and update counter
        FLcount_total = FLcount_total + 1;
        ST_AN1(1,FLcount_total) = str2num(g.command('MG @AN[1]'));
        Newton1(1,FLcount_total) = (ST_AN1(1,FLcount_total)*Calib1(1))+Calib1(2);
        ST_AN2(1,FLcount_total) = str2num(g.command('MG @AN[2]'));
        Newton2(1,FLcount_total) = (ST_AN2(1,FLcount_total)*Calib2(1))+Calib2(2);
        % Display FL and robot loads
        FL_display_Spine2([fm_tcs, fx, fy, fz], [mx, my, mz], handles,...
            [now, z_ct, cycle, z_target], Newton1, Newton2, FLcount_total,...
            FLh, IDPmean, Loadcell_Pressure_True_N);
        % Store FL in Spine structure - moved to Pathseek_Spine.m - just
        % record at endpoints
        Spine.(date_ID).(state).(current_angles).(pathtypestr).pathseek(trial)...
            .(HAM_str).(pathsequence_str)(cycle).(path_name)...
            .FLoad_all(1,FLcount_total) = Newton1(1,FLcount_total);
        Spine.(date_ID).(state).(current_angles).(pathtypestr).pathseek(trial)...
            .(HAM_str).(pathsequence_str)(cycle).(path_name)...
            .FLoad_all(2,FLcount_total) = Newton2(1,FLcount_total);
    end

    if z_flag(1) == 0
        dis_tool_calc(1:6)=0;
        rotate(1:3)=0;
        dis_tool_actual(1:6)=0;
        ds_g(1:3)=0;
        drpy_g(1:3)=0;
    else
        % ds = ACTUAL MOTION OF ROBOT USED TO MAKE STIFFNESS MATRIX
        % find actual translations and rotations in global c.s., ...
        ds_g(1:3) = gt_jt_angles(1:3)-p_temp(1:3); % mm
        rpy1_g = rad2deg(tr2rpy(eul2tr(deg2rad(p_temp(4:6))))); % degrees
        rpy2_g = rad2deg(tr2rpy(eul2tr(deg2rad(gt_jt_angles(4:6))))); % degrees
        rpy1_g = flipdim(rpy1_g,2); % degrees
        rpy2_g = flipdim(rpy2_g,2); % degrees
        drpy_g = rpy2_g - rpy1_g;   % degrees

        % ... transform to tool c.s.
%           tool_loc(1:3) = rGT'*gt_jt_angles(1:3); % mm
%        tool_loc(4:6) = rGT'*rpy2_g'; % degrees
           ds(1:3) = rGT'*ds_g(1:3)'; % mm
           ds(4:6) = rGT'*drpy_g';  % degrees

           dis_tool_actual(1:3) = ds(1:3); % mm
           dis_tool_actual(4:6) = ds(4:6); % degrees
        end

        % store current position - asks for position in fm_tare
        for i = 1:6
           p_temp(i) = gt_jt_angles(i);
        end

        % Function to determine minimum Load,Position,Step
        if step_index == 1
           min_load(1:6) = 0;
           min_global_pos(1:6) = 0;
           min_jt_angles(1:6) = 0;
           min_step = 0;
        end

        [min_load, min_global_pos, min_jt_angles, min_step] = minimum_tracker_Spine...
           (fm_tcs, gt_jt_angles, endpt_index, min_flags, min_load, min_global_pos,...
           min_jt_angles, min_step, z_ct, step_index, preload);


        Spine.(date_ID).(state).(current_angles).(pathtypestr).pathseek(trial)...
           .(HAM_str).(pathsequence_str)(cycle).(path_name).dis_calc(1:6,step_index)...
           = [dis_tool_calc(1:3) dis_tool_calc(4:6)];
        Spine.(date_ID).(state).(current_angles).(pathtypestr).pathseek(trial)...

.(HAM_str).(pathsequence_str)(cycle).(path_name).dis_actual_tool(1:6,step_index)...
           = transpose(dis_tool_actual);
        Spine.(date_ID).(state).(current_angles).(pathtypestr).pathseek(trial)...

.(HAM_str).(pathsequence_str)(cycle).(path_name).dis_actual_global(1:6,step_index)...
           = [transpose(ds_g); transpose(drpy_g)];
        Spine.(date_ID).(state).(current_angles).(pathtypestr).pathseek(trial)...
           .(HAM_str).(pathsequence_str)(cycle).(path_name).load(1:6,step_index)...
           = transpose(fm_tcs);
        Spine.(date_ID).(state).(current_angles).(pathtypestr).pathseek(trial)...
           .(HAM_str).(pathsequence_str)(cycle).(path_name).stiff(1:6,step_index)...
           = transpose(z_stiff);
        Spine.(date_ID).(state).(current_angles).(pathtypestr).pathseek(trial)...
           .(HAM_str).(pathsequence_str)(cycle).(path_name).time_total(1:6,step_index) =
time;
```

```
                Spine.(date_ID).(state).(current_angles).(pathtypestr).pathseek(trial)...
                    .(HAM_str).(pathsequence_str)(cycle).(path_name).rot_angle(1:6,step_index) =
now;
                Spine.(date_ID).(state).(current_angles).(pathtypestr).pathseek(trial)...
                    .(HAM_str).(pathsequence_str)(cycle).(path_name).global_pos(1:6,step_index)...
                    = gt_jt_angles(1:6);
                Spine.(date_ID).(state).(current_angles).(pathtypestr).pathseek(trial)...
                    .(HAM_str).(pathsequence_str)(cycle).(path_name).jt_angles(1:6,step_index)...
                    = gt_jt_angles(7:12);
                Spine.(date_ID).(state).(current_angles).(pathtypestr).pathseek(trial)...
                    .(HAM_str).(pathsequence_str)(cycle).(path_name).error(1:6,step_index)...
                    = dis_tool_calc(1:6)-dis_tool_actual(1:6);
                Spine.(date_ID).(state).(current_angles).(pathtypestr).pathseek(trial)...

.(HAM_str).(pathsequence_str)(cycle).(path_name).delta_error_all(1:6,step_index)...
                    = delta_error_all;
                Spine.(date_ID).(state).(current_angles).(pathtypestr).pathseek(trial)...
                    .(HAM_str).(pathsequence_str)(cycle).(path_name).delta_error_abs_all...
                    (1:6,step_index) = delta_error_abs_all;
                Spine.(date_ID).(state).(current_angles).(pathtypestr).pathseek(trial)...
                    .(HAM_str).(pathsequence_str)(cycle).(path_name).fuzzy_calc(1:6,step_index)...
                    = fuzzy_calc;
                Spine.(date_ID).(state).(current_angles).(pathtypestr).pathseek(trial)...
                    .(HAM_str).(pathsequence_str)(cycle).(path_name).force_multiplier...
                    (1:3,step_index) = force_multiplier2;



                if z_ct < z_count
                    % are the measured sagittal plane forces < max allowable?
                    % if no, begin load control loop

                    delta_target(1) = (fm_tcs(1) - preload(1));
                    delta_target(2) = (fm_tcs(2) - preload(2));
                    delta_target(3) = (fm_tcs(3) - preload(3));
                    delta_target(4) = (fm_tcs(4) - preload(4));
                    delta_target(5) = (fm_tcs(5) - preload(5));
                    delta_target(6) = (fm_tcs(6) - preload(6));

                    % update counters
                    z_ct = z_ct + 1;            % Iteration counter

                    if abs(delta_target(1))*min_flags(1) > z_target(1) |
abs(delta_target(2))*min_flags(2)...
                        > z_target(2) | abs(delta_target(3))*min_flags(3) > z_target(3)...
                        | abs(delta_target(4))*min_flags(4) > z_target(4) | abs(delta_target(5))...
                        *min_flags(5) > z_target(5) | abs(delta_target(6))*min_flags(6) > z_target(6)

                    if z_flag(1) == 0
                        % compute: robot displacement vector using initial stiffness guess
                        for i = 1:6
                            z_flag(i) = 1;
                            f_temp(i) = fm_tcs(i); % keep previous f/m
                            dis(i) = delta_target(i)/z_stiff(i)/(1+1*z_sign(i)); % dis[i] is in the tool c.s.; i
= 1:3, mm; i = 4:6, degrees
                        end
                    else
                        for i = 1:6
                            z_flag(i) = 1;

                            % compute: FSU stiffness from previous measured force and position
                            if (delta_target(i) ~= f_temp(i)) & (ds(i) ~= 0) & (delta_target(i) ~= 0)
                                % STIFFNESS = old*1/3    +ABS(df/dx)*2/3
                                z_stiff(i) = z_stiff(i)/3+abs((delta_target(i)-f_temp(i))/ds(i))*2/3;
                            end

                            % if forces go from positive to negative used to divide motion in half
                            if sign(f_temp(i)*delta_target(i)) < 1
                                z_sign(i) = 1;
                            else
                                z_sign(i) = 0;
                            end

                            % compute: robot displacement vector to minimize forces and moments
                            f_temp(i) = delta_target(i); % keep previous f/m
                            dis(i) = delta_target(i)/z_stiff(i)/(1+1*z_sign(i)); % dis(i) is in the tool c.s.
                        end
                    end

                    % limit translations based on forces
                    for i = 1:3
                        if abs(dis(i)) > lim_dis
                            dis(i) = sign(dis(i))*lim_dis; % mm
                        end
                    end

                    % do not allow translation along an axis if the force along that axis is < threshold
                    for i = 1:3
                        if abs(delta_target(i)) > z_target(i)
                            dis(i) = dis(i); % mm
                        else
                            dis(i) = 0; % mm
```

```
    end
end

dis_tool_calc(1:3) = dis(1:3)'; % mm

% limit rotations based on moments
for i = 4:6
    if abs(dis(i)) > lim_mdis
        dis(i) = sign(dis(i))*lim_mdis; % degrees
    end

    dis(i) = deg2rad(dis(i));       % radians

end

% do not allow rotation about an axis if the moment about that axis is < threshold
for i = 4:6
    if abs(delta_target(i)) > z_target(i)
        dis(i) = dis(i); % radians
    else
        dis(i) = deg2rad(0.0000001); % radians
    end
end

dis_tool_calc(4:6) = dis(4:6)'; % radians

% dis_tool_calc[4]-[6] are rotations about x,y,z (yaw,pitch,roll),
% rot_xyz = rpy2tr(dis_tool_calc(6), dis_tool_calc(5), deg2rad(0.0000001));
% rotate = tr2eul(rot_xyz);
% rotate = rad2deg(rotate) + 0.0000001; % degrees

% Fuzzy Control - supercededs previous commands
for dof = 1:6
    if dof <= 3
        if fuzzyone == 1
            err = Spine.(date_ID).(state).(current_angles).(pathtypestr)...
                .pathseek(trial).(HAM_str).(pathsequence_str)(cycle).(path_name)...
                .load(dof,step_index) - preload(dof);
            err_abs = abs(err);
            if step_index > 3
                delta_err = err - (Spine.(date_ID).(state).(current_angles)...
                    .(pathtypestr).pathseek(trial).(HAM_str).(pathsequence_str)(cycle)...
                    .(path_name).load(dof,step_index-1) - preload(dof));
                delta_err_abs = abs(err) - abs((Spine.(date_ID).(state)...
                    .(current_angles).(pathtypestr).pathseek(trial).(HAM_str)...
                    .(pathsequence_str)(cycle).(path_name).load(dof,step_index-1)...
```

```
                                - preload(dof)));
                        else
                            delta_err = -30;
                        end

                        uni_step = -evalfis([err delta_err],Fuzzy_force);

                        fuzzy_calc(dof) = uni_step;
                        delta_error_all(dof) = delta_err;
                        delta_error_abs_all(dof) = delta_err_abs;
                        error_abs_all(dof) = err_abs;
                        % dis_tool_calc(1) = 0;
                        % dis_tool_calc(3) = 0;
                        % dis_tool_calc(2) = uni_step
                    end
                elseif dof >= 4
                    if fuzzyone == 1
                        err = Spine.(date_ID).(state).(current_angles)...
                            .(pathtypestr).pathseek(trial).(HAM_str)...
                            .(pathsequence_str)(cycle).(path_name).load(dof,step_index)...
                            - preload(dof);
                        err_abs = abs(err);
                        if step_index > 3
                            delta_err = err - (Spine.(date_ID).(state)...
                                .(current_angles).(pathtypestr).pathseek(trial)...
                                .(HAM_str).(pathsequence_str)(cycle).(path_name)...
                                .load(dof,step_index-1) - preload(dof));
                            delta_err_abs = abs(err) - abs((Spine.(date_ID)...
                                .(state).(current_angles).(pathtypestr).pathseek(trial)...
                                .(HAM_str).(pathsequence_str)(cycle).(path_name)...
                                .load(dof,step_index-1) - preload(dof)));
                        else
                            delta_err = -3;
                        end
                        uni_step = -evalfis([err delta_err],Fuzzy_moment);

%                         fuzzy_calc(dof) = uni_step*avg_force_multiplier(dof,mult_counter);
                        fuzzy_calc(dof) = uni_step;
                        delta_error_all(dof) = delta_err;
                        delta_error_abs_all(dof) = delta_err_abs;
                        error_abs_all(dof) = err_abs;

                    end
                end
            end
            force_mult_temp = force_multiplier2;

            dis_tool_calc = [fuzzy_calc(1)/force_mult_temp(1) fuzzy_calc(2)...
               /force_mult_temp(2) fuzzy_calc(3)/force_mult_temp(3)...
               (fuzzy_calc(4)) (fuzzy_calc(5)) (fuzzy_calc(6))];

            rotate = rad2deg(tr2eul(rpy2tr(deg2rad(dis_tool_calc(6))*min_flags(6)...
               + 0.0000001, deg2rad(dis_tool_calc(5))*min_flags(5)+ 0.0000001,...
               deg2rad(dis_tool_calc(4))*min_flags(4)+ 0.0000001))) + 0.0000001;
            % MOVE
            % move: translate superior vertebra to new "corrected" position
            ok = 0;
            flag = 4.1;
            fprintf(port1,'%f\n', [ok, flag]);
            fprintf(port1,'%f\n', [dis_tool_calc(1)*min_flags(1) dis_tool_calc(2)...
               *min_flags(2) dis_tool_calc(3)*min_flags(3) rotate(1) rotate(2)...
               rotate(3)]+.0000001);

            done_moving = fscanf(port1);
            % ================================
            timeout_Spine;
            % ================================
            done_moving = sscanf(done_moving, '%f');


        else
            % Commented out interation stop - will always do z_count
            % interations
            %three_counter = three_counter + 1;
            %if three_counter > 2
            %    z_ct_temp = z_ct;
            %    z_ct = z_count + 1;
            %end
        end
    else
        z_ct_temp = z_ct;
        z_ct = z_count + 1;
    end
end
```

A.1.3 Follower Load – Program to Apply and Maintain FL

```
% Follower_Load_Control_Fuzzy.m
% Follower_Load mixed with Load Control - slowly add follower load.
% Kevin Bell
% 5/27/2013

FL_Init;
FLmax = 50; % Newton - temporary - need to add to Spine_display
Loadcount_total = 0;

FLloop = 25;
delta_err_abs_all(1:6) = 0;


% limit to z_count iterations (will want to change to time limit)
for FLcount = 1:FLloop

   if FLcount == 2
      %FLPA = FLmax;
      FLPA1 = FLmax+0;
      FLPA2 = FLmax+0;
      FLPA_Step;
   end

   %========================================
   get_loads; % measure: forces and moments
   %========================================

   Loadcount_total = Loadcount_total + 1;
   FL_Load(1:6,Loadcount_total) = transpose(fm_tcs);

   %========================================
   fm_tare6; % tare out bolt-up and fixture wt
   %========================================

   ok = 0;
   flag = 1.1;
   fprintf(port1,'%f\n', [ok, flag]);
   gt_jt_angles = fscanf(port1);
   gt_jt_angles = sscanf(gt_jt_angles, '%f');

   % Get Follower Loads and update counter
   FLcount_total = FLcount_total + 1;
   ST_AN1(1,FLcount_total) = str2num(g.command('MG @AN[1]'));
   Newton1(1,FLcount_total) = (ST_AN1(1,FLcount_total)*Calib1(1))+Calib1(2);
   ST_AN2(1,FLcount_total) = str2num(g.command('MG @AN[2]'));
   Newton2(1,FLcount_total) = (ST_AN2(1,FLcount_total)*Calib2(1))+Calib2(2);

   N1=Newton1(1,FLcount_total);
```

```
            N2=Newton2(1,FLcount_total);

            % Store Follower load
            Spine.(date_ID).(state).(current_angles).(pathtypestr).pathseek(trial)...

.(HAM_str).(pathsequence_str)(cycle).(path_name).(FLvar).FLoad(1,Loadcount_total) = N1;
            Spine.(date_ID).(state).(current_angles).(pathtypestr).pathseek(trial)...

.(HAM_str).(pathsequence_str)(cycle).(path_name).(FLvar).FLoad(2,Loadcount_total) = N2;

            if GLC_num == 1
               Loadcell_Pressure_True;
               Spine.(date_ID).(state).(current_angles).(pathtypestr).pathseek(trial)...

.(HAM_str).(pathsequence_str)(cycle).(path_name).(FLvar).GLC(Loadcount_total)...
                  =Loadcell_Pressure_True_N;
            end

            % Get IDP measurements from IDP 202
            if IDP_num > 0
               % Need to replace with new getIDP code
               get_IDP;
               % Build array of ((path_name)) IDP1
               Spine.(date_ID).(state).(current_angles).(pathtypestr).pathseek(trial)...

.(HAM_str).(pathsequence_str)(cycle).(path_name).(FLvar).IDP(Loadcount_total,:)...
                  = IDPmean;
            end

            %+++++++++++++++++++ Data Storage
+++++++++++++++++++++++++++++++++++
            % Build array of ((path_name))load end point data
            Spine.(date_ID).(state).(current_angles).(pathtypestr).pathseek(trial).(HAM_str)...
               .(pathsequence_str)(cycle).(path_name).(FLvar).load(1:6,Loadcount_total)...
               = transpose(fm_tcs);
            % Build array of ((path_name)) positions data that could be for replay
            Spine.(date_ID).(state).(current_angles).(pathtypestr).pathseek(trial).(HAM_str)...
               .(pathsequence_str)(cycle).(path_name).(FLvar).global_pos(1:6,Loadcount_total)...
               = gt_jt_angles(1:6);
            % Build array of ((path_name))jt angles data that could be for replay
            Spine.(date_ID).(state).(current_angles).(pathtypestr).pathseek(trial).(HAM_str)...
               .(pathsequence_str)(cycle).(path_name).(FLvar).jt_angles(1:6,Loadcount_total)...
               = gt_jt_angles(7:12);

            FL_Load(1:6,FLcount_total) = transpose(fm_tcs);
```

```
        % Need to creat new FL display panel that does not interfere with
        % pathsee display
        % display f/m after taring out bolt-up and fixture wt
        FL_display_Spine2([fm_tcs, fx, fy, fz], [mx, my, mz], handles,...
            [now, z_ct, cycle, z_target], Newton1, Newton2, FLcount_total, FLh, IDPmean,...
            Loadcell_Pressure_True_N);

        % store current position - asks for position in fm_tare
        for i = 1:6
            p_temp(i) = gt_jt_angles(i);
        end

        % are the measured sagittal plane forces < max allowable?
        % if no, begin load control loop

        delta_target(1) = (fm_tcs(1) - preload(1));
        delta_target(2) = (fm_tcs(2) - preload(2));
        delta_target(3) = (fm_tcs(3) - preload(3));
        delta_target(4) = (fm_tcs(4) - preload(4));
        delta_target(5) = (fm_tcs(5) - preload(5));
        delta_target(6) = (fm_tcs(6) - preload(6));

        if abs(delta_target(1))*min_flags(1) > z_target(1) | abs(delta_target(2))...
                *min_flags(2) > z_target(2) | abs(delta_target(3))*min_flags(3) > z_target(3)...
                | abs(delta_target(4))*min_flags(4) > z_target(4) | abs(delta_target(5))...
                *min_flags(5) > z_target(5) | abs(delta_target(6))*min_flags(6) > z_target(6)

            % Fuzzy Control - supercededs previous commands
            for dof = 1:6
                if dof <= 3
                    if fuzzyone == 1
                        err = Spine.(date_ID).(state).(current_angles).(pathtypestr)...
                            .pathseek(trial).(HAM_str).(pathsequence_str)(cycle).(path_name)...
                            .(FLvar).load(dof,Loadcount_total) - preload(dof);
                        err_abs = abs(err);
                        if Loadcount_total > 3
                            if dof == 2
                                Spine.(date_ID).(state).(current_angles).(pathtypestr)...
                                    .pathseek(trial).(HAM_str).(pathsequence_str)(cycle)...
                                    .(path_name).(FLvar).load(dof,Loadcount_total-1);
                                preload(dof);
                            end
                            delta_err = err - (Spine.(date_ID).(state).(current_angles)...
                                .(pathtypestr).pathseek(trial).(HAM_str).(pathsequence_str)...
                                (cycle).(path_name).(FLvar).load(dof,Loadcount_total-1) -
preload(dof));
```

```
                        delta_err_abs = abs(err) - (abs(Spine.(date_ID).(state)...
                            .(current_angles).(pathtypestr).pathseek(trial).(HAM_str)...

.(pathsequence_str)(cycle).(path_name).(FLvar).load(dof,Loadcount_total-1))...
                            - preload(dof));
                    else
                        delta_err = -30;
                    end

                    uni_step = -evalfis([err delta_err],Fuzzy_force);

                    fuzzy_calc(dof) = uni_step;
                    delta_error_all(dof) = delta_err;
                    delta_error_abs_all(dof) = delta_err_abs;
                    error_abs_all(dof) = err_abs;

                end
            elseif dof >= 4
                err = FL_Load(dof,Loadcount_total) - preload(dof);
                err_abs = abs(err);
                if Loadcount_total > 3
                    delta_err = err - (Spine.(date_ID).(state).(current_angles)...
                        .(pathtypestr).pathseek(trial).(HAM_str).(pathsequence_str)(cycle)...
                        .(path_name).(FLvar).load(dof,Loadcount_total-1) - preload(dof));
                    delta_err_abs = abs(err) - (abs(Spine.(date_ID).(state)...
                        .(current_angles).(pathtypestr).pathseek(trial).(HAM_str)...

.(pathsequence_str)(cycle).(path_name).(FLvar).load(dof,Loadcount_total-1)) - preload(dof));
                else
                    delta_err = -3;
                end

                uni_step = -evalfis([err delta_err],Fuzzy_moment);

                fuzzy_calc(dof) = uni_step;
                delta_error_all(dof) = delta_err;
                delta_error_abs_all(dof) = delta_err_abs;
                error_abs_all(dof) = err_abs;

            end
        end

        force_mult_temp = force_multiplier2;

        dis_tool_calc = [fuzzy_calc(1)/force_mult_temp(1)
fuzzy_calc(2)/force_mult_temp(2)...
```

```
                    fuzzy_calc(3)/force_mult_temp(3) (fuzzy_calc(4)) (fuzzy_calc(5))
(fuzzy_calc(6))];

                rotate = rad2deg(tr2eul(rpy2tr(deg2rad(dis_tool_calc(6))*min_flags(6)+
0.0000001,...
                    deg2rad(dis_tool_calc(5))*min_flags(5)+ 0.0000001, deg2rad(dis_tool_calc(4))...
                    *min_flags(4)+ 0.0000001))) + 0.0000001; % degrees

                % MOVE
                % move: translate superior vertebra to new "corrected" position
                ok = 0;
                flag = 4.1;
                fprintf(port1,'%f\n', [ok, flag]);
                fprintf(port1,'%f\n', [dis_tool_calc(1)*min_flags(1) dis_tool_calc(2)*min_flags(2)...
                    dis_tool_calc(3)*min_flags(3) rotate(1) rotate(2) rotate(3)]+.0000001);

                done_moving = fscanf(port1);
                % ================================
                timeout_Spine;
                % ================================
                done_moving = sscanf(done_moving, '%f');

            end
        end
```

A.2 POSTPROCESSING – ANALYZE MOMENT-ROTATION AND IDP

A.2.1 Plot Robot Moment-Rotation Data

```
% BellPhD_Plot.m
% Kevin Bell
% Updated 5/31/2013

% fh=figure('Position',[150 100 700 600],'Color','w');

position = 'LAT0_FE0_AXIAL0';
HAM_str = 'HAM';
motion = 'pathseek';
pathsequence_str = 'loop';
trial = 1;
% Currently only plotting 2nd cycle
```

```
cycle = 2;
color_str1 = {'bo', 'ro', 'ko', 'go', 'mo'};
color_str2 = {'b+', 'r+', 'k+', 'g+', 'm+'};

pathtypestr = 'LAT';

pos = 0; % 1 = pos, 0 = neg

path_name_p1 = {'loop_0_pos'; 'loop_pos_neg'; 'loop_neg_pos'};
path_name_p2 = {'loop_pos_neg'; 'loop_neg_pos'; 'loop_pos_0'};
path_name_n1 = {'loop_0_neg'; 'loop_neg_pos'; 'loop_pos_neg'};
path_name_n2 = {'loop_neg_pos'; 'loop_pos_neg'; 'loop_neg_0'};

rotnum = 6;

fn_SA = fieldnames(Spine_ALL);

C45 = 1;
C56 = 2;

%for i = 3:length(fn_SA)
for i = 16:16

    fh1=figure('Position',[150 100 700 600],'Color','w');
    fh2=figure('Position',[150 100 700 600],'Color','w');
    fh3=figure('Position',[150 100 700 600],'Color','w');
    %     title([fn_SA{i}]);
    %subplot(3,2)

    %fn_SA_st = fieldnames(Spine_ALL.(fn_SA{i}));
    fn_SA_st = {'FL0_AL0_noHAM' 'FL0_AL0_HAM' 'FL100_AL0_HAM'...
        'FL0_AL50_HAM' 'FL100_AL50_HAM'};
    fn_SA_st2 = strrep(fn_SA_st,'_',' ');

    %for j = 1:1
    for j = 1:length(fn_SA_st)

        %                 if j>3
        %                     trial = 1;
        %                 end


        fn_SA_st{j};

        if findstr('noHAM', fn_SA_st{j}) > 0
            HAM_str = 'noHAM';
```

```
else
   HAM_str = 'HAM';
end

for pnc = 1:cycle
   for pn = 1:3

      if pnc == 1
         if pos == 1
            path_name = path_name_p1[121];
         else
            path_name = path_name_n1[121];
         end
      else
         if pos == 1
            path_name = path_name_p2[121];
         else
            path_name = path_name_n2[121];
         end
      end

      % Calculate "NEW" Tool Path - redefining FE based
      % on initial starting position.
      if pn == 1 && pnc == 1
         display([num2str(pn) num2str(pnc)])
         tool_path = Spine_ALL.(fn_SA{i}).(fn_SA_st{j})...
            .(position).(pathtypestr).(motion)(trial).(HAM_str)...
            .(pathsequence_str)(1,pnc).(path_name).tool_path(:,:);
         tool_path(:,2:end+1) = tool_path;
         tool_path(:,1) = 0;
         tool_end = tool_path(:,end);
         Spine_ALL.(fn_SA{i}).(fn_SA_st{j}).(position)...
            .(pathtypestr).(motion)(trial).(HAM_str)...
            .(pathsequence_str)(1,pnc).(path_name)...
            .tool_path_new(:,:) = tool_path;
      elseif pn > 1 | pnc > 1
         display([num2str(pn) num2str(pnc)])
         tool_path = Spine_ALL.(fn_SA{i}).(fn_SA_st{j})...
            .(position).(pathtypestr).(motion)(trial)...
            .(HAM_str).(pathsequence_str)(1,pnc)...
            .(path_name).tool_path(:,:);
         tool_path(:,2:end+1) = tool_path;
         tool_path(:,1) = 0;
         for tloop = 1:length(tool_path)
            tool_path(:,tloop) = tool_path(:,tloop)+tool_end;
         end
```

```
            tool_end = tool_path(:,end);
            [zero_val, zero_loc] = min(abs(tool_path(rotnum,:)));
            Spine_ALL.(fn_SA{i}).(fn_SA_st{j}).(position)...
                .(pathtypestr).(motion)(trial).(HAM_str)...
                .(pathsequence_str)(1,pnc).(path_name)...
                .tool_zero = [zero_val, zero_loc];
            Spine_ALL.(fn_SA{i}).(fn_SA_st{j}).(position)...
                .(pathtypestr).(motion)(trial).(HAM_str)...
                .(pathsequence_str)(1,pnc).(path_name)...
                .tool_path_new(:,:) = tool_path;
        end

        clear tool_path

    end

end

clear tool_end


%% Moment Plotting
figure(fh1);
for k = 1:2

    tool_path_new = Spine_ALL.(fn_SA{i}).(fn_SA_st{j}).(position)...
        .(pathtypestr).(motion)(trial).(HAM_str)...
        .(pathsequence_str)(1,cycle).(path_name_p2{k}).tool_path_new(:,:);

    load_end_pts = Spine_ALL.(fn_SA{i}).(fn_SA_st{j}).(position)...
        .(pathtypestr).(motion)(trial).(HAM_str).(pathsequence_str)...
        (1,cycle).(path_name_p2{k}).load_end_pts(:,:);

    if k == 1
        subplot(3,2,j); plot(tool_path_new(rotnum,:)...
            , load_end_pts(rotnum,:) ,color_str1{j});
        title([fn_SA_st2{j}]);
        xlabel('Rotation Angle (deg)');
        ylabel(['Moment (Nm)']);
        hold on; grid on
        subplot(3,2,length(fn_SA_st)+1); plot(tool_path_new(rotnum,:)...
            , load_end_pts(rotnum,:) ,color_str1{j});
        hold on; grid on
    else
        subplot(3,2,j); plot(tool_path_new(rotnum,:)...
            , load_end_pts(rotnum,:) ,color_str2{j});
        title([fn_SA_st2{j}]);
        xlabel('Rotation Angle (deg)');
        ylabel(['Moment (Nm)']);
        hold on; grid on
        subplot(3,2,length(fn_SA_st)+1); plot(tool_path_new(rotnum,:)...
            , load_end_pts(rotnum,:) ,color_str2{j});
        hold on; grid on
    end

    title('combined');
    xlabel('Rotation Angle (deg)');
    ylabel(['Moment (Nm)']);

end

clear tool_path_new load_end_pts

%% Calculate Neutral Zone / Elastic Zone
for k = 1:2

    tool_path_new = Spine_ALL.(fn_SA{i}).(fn_SA_st{j})...
        .(position).(pathtypestr).(motion)(trial).(HAM_str)...
        .(pathsequence_str)(1,cycle).(path_name_p2{k}).tool_path_new(:,:);

    load_end_pts = Spine_ALL.(fn_SA{i}).(fn_SA_st{j}).(position)...
        .(pathtypestr).(motion)(trial).(HAM_str)...
        .(pathsequence_str)(1,cycle).(path_name_p2{k}).load_end_pts(:,:);

    length(tool_path_new)

    if k == 1
        tool_path(k,:,j) = tool_path_new(rotnum,:);
        load_end(k,:,j) = load_end_pts(rotnum,:);
    end

    if k == 1
        subplot(3,2,j); plot(tool_path_new(rotnum,:)...
            , load_end_pts(rotnum,:) ,color_str1{j});
        title([fn_SA_st2{j}]);
        xlabel('Rotation Angle (deg)');
        ylabel(['Moment (Nm)']);
        hold on; grid on
        subplot(3,2,length(fn_SA_st)+1); plot(tool_path_new(rotnum,:)...
            , load_end_pts(rotnum,:) ,color_str1{j});
        hold on; grid on
    else
```

```
        subplot(3,2,j); plot(tool_path_new(rotnum,:)...
            , load_end_pts(rotnum,:) ,color_str2{j});
        title([fn_SA_st2{j}]);
        xlabel('Rotation Angle (deg)');
        ylabel(['Moment (Nm)']);
        hold on; grid on
        subplot(3,2,length(fn_SA_st)+1); plot(tool_path_new(rotnum,:)...
            , load_end_pts(rotnum,:) ,color_str2{j});
        hold on; grid on
    end

    title('combined');
    xlabel('Rotation Angle (deg)');
    ylabel(['Moment (Nm)']);

end

clear tool_path_new load_end_pts

%% IDP C45 Plotting
figure(fh2);
for k = 1:2

    tool_path_new = Spine_ALL.(fn_SA{i}).(fn_SA_st{j}).(position)...
        .(pathtypestr).(motion)(trial).(HAM_str)...
        .(pathsequence_str)(1,cycle).(path_name_p2{k}).tool_path_new(:,:);

    load_end_pts = Spine_ALL.(fn_SA{i}).(fn_SA_st{j}).(position)...
        .(pathtypestr).(motion)(trial).(HAM_str)...
        .(pathsequence_str)(1,cycle).(path_name_p2{k}).load_end_pts(:,:);

    IDP_end_pts = Spine_ALL.(fn_SA{i}).(fn_SA_st{j}).(position)...
        .(pathtypestr).(motion)(trial).(HAM_str)...
        .(pathsequence_str)(1,cycle).(path_name_p2{k}).IDP(:,:);

    if k == 1
        subplot(3,2,j); plot(load_end_pts(rotnum,:)...
            , IDP_end_pts(:,C45) ,color_str1{j});
        title([fn_SA_st2{j}]);
        xlabel('Moment (Nm)');
        ylabel(['C45: IDP (psi)']);
        hold on; grid on
        subplot(3,2,length(fn_SA_st)+1); plot(load_end_pts(rotnum,:)...
            , IDP_end_pts(:,C45) ,color_str1{j});
        hold on; grid on
    else
```

```
        subplot(3,2,j); plot(load_end_pts(rotnum,:)...
            , IDP_end_pts(:,C45) ,color_str2{j});
        title([fn_SA_st2{j}]);
        xlabel('Moment (Nm)');
        ylabel(['C45: IDP (psi)']);
        hold on; grid on
        subplot(3,2,length(fn_SA_st)+1); plot(load_end_pts(rotnum,:)...
            , IDP_end_pts(:,C45) ,color_str2{j});
        hold on; grid on
    end

    title('Combined');
    xlabel('Moment (Nm)');
    ylabel(['C45: IDP (psi)']);

end

clear tool_path_new IDP_end_pts load_end_pts

%% IDP C56 Plotting
figure(fh3);
for k = 1:2

    tool_path_new = Spine_ALL.(fn_SA{i}).(fn_SA_st{j}).(position)...
        .(pathtypestr).(motion)(trial).(HAM_str)...
        .(pathsequence_str)(1,cycle).(path_name_p2{k}).tool_path_new(:,:);

    load_end_pts = Spine_ALL.(fn_SA{i}).(fn_SA_st{j}).(position)...
        .(pathtypestr).(motion)(trial).(HAM_str)...
        .(pathsequence_str)(1,cycle).(path_name_p2{k}).load_end_pts(:,:);

    IDP_end_pts = Spine_ALL.(fn_SA{i}).(fn_SA_st{j}).(position)...
        .(pathtypestr).(motion)(trial).(HAM_str)...
        .(pathsequence_str)(1,cycle).(path_name_p2{k}).IDP(:,:);

    if k == 1
        subplot(3,2,j); plot(load_end_pts(rotnum,:)...
            , IDP_end_pts(:,C56) ,color_str1{j});
        title([fn_SA_st2{j}]);
        xlabel('Moment (Nm)');
        ylabel(['C56: IDP (psi)']);
        hold on; grid on
        subplot(3,2,length(fn_SA_st)+1); plot(load_end_pts(rotnum,:)...
            , IDP_end_pts(:,C56) ,color_str1{j});
        hold on; grid on
    else
```

```
                    subplot(3,2,j); plot(load_end_pts(rotnum,:)...
                        , IDP_end_pts(:,C56) ,color_str2{j});
                    title([fn_SA_st2{j}]);
                    xlabel('Moment (Nm)');
                    ylabel(['C56: IDP (psi)']);
                    hold on; grid on
                    subplot(3,2,length(fn_SA_st)+1); plot(load_end_pts(rotnum,:)...
                        , IDP_end_pts(:,C56) ,color_str2{j});
                    hold on; grid on
                end

                title('Combined');
                xlabel('Moment (Nm)');
                ylabel(['C56: IDP (psi)']);

            end

            clear tool_path_new IDP_end_pts load_end_pts
        end

    end

    % Save Figures

    % Moment Figure
    savestr = [(fn_SA{i}) '\' (fn_SA{i}) '_momvsrot'];
    saveas(fh1,['Z:\Ortho Research 3\FergusonLab\Staff\Bell,
Kevin\Kevin\Projects\PhD\Testing\'...
        savestr]);
    saveas(fh1,['Z:\Ortho Research 3\FergusonLab\Staff\Bell,
Kevin\Kevin\Projects\PhD\Testing\'...
        savestr],'jpg');
    close(fh1);

    % IDP C45 Figure
    savestr = [(fn_SA{i}) '\' (fn_SA{i}) '_C45_IDPvsmom'];
    saveas(fh2,['Z:\Ortho Research 3\FergusonLab\Staff\Bell,
Kevin\Kevin\Projects\PhD\Testing\'...
        savestr]);
    saveas(fh2,['Z:\Ortho Research 3\FergusonLab\Staff\Bell,
Kevin\Kevin\Projects\PhD\Testing\'...
        savestr],'jpg');
    close(fh2);

    % IDP C56 Figure
    savestr = [(fn_SA{i}) '\' (fn_SA{i}) '_C56_IDPvsmom'];
```

```
        saveas(fh3,['Z:\Ortho Research 3\FergusonLab\Staff\Bell,
Kevin\Kevin\Projects\PhD\Testing\'...
            savestr]);
        saveas(fh3,['Z:\Ortho Research 3\FergusonLab\Staff\Bell,
Kevin\Kevin\Projects\PhD\Testing\'...
            savestr],'jpg');
        close(fh3);
```

A.2.2 Fit Moment-Rotation Data to Bi-Sigmoidal Curve

```
function [Spine_ALL] = DS_createFit(Spine_ALL, SP, SN)

% Kevin Bell
% Updated 5/31/2013

if PN1 == 1
    CFvar = 'CF';
    NZ_ALLvar = 'NZ_ALL';
    NZvar = 'NZ';
else
    CFvar = 'CF2';
    NZ_ALLvar = 'NZ_ALL2';
    NZvar = 'NZ2';
end

position = 'LAT0_FE0_AXIAL0';
motion = 'pathseek';
pathsequence_str = 'loop';
trial = 1;
% Only Plotting 2nd cycle currently
cycle = 2;
color_str1 = {'bo', 'ro', 'ko', 'go', 'mo'};
color_str2 = {'b+', 'r+', 'k+', 'g+', 'm+'};

pathtypestr = 'LAT';

pos = 0; % 1 = pos, 0 = neg

path_name_p1 = {'loop_0_pos'; 'loop_pos_neg'; 'loop_neg_pos'};
path_name_p2 = {'loop_pos_neg'; 'loop_neg_pos'; 'loop_pos_0'};
path_name_n1 = {'loop_0_neg'; 'loop_neg_pos'; 'loop_pos_neg'};
path_name_n2 = {'loop_neg_pos'; 'loop_pos_neg'; 'loop_neg_0'};

%Spine_ALL is a "Super Structure" that contains multiple Spine structures
```

```
fn_SA = fieldnames(Spine_ALL);

SPnum = SP;
Statenum = SN;
rotnum = 6;

if Statenum == 1
   HAM_str = 'noHAM';
else
   HAM_str = 'HAM';
end

for i = SPnum:SPnum

   %fn_SA_st = fieldnames(Spine_ALL.(fn_SA{i}));
   fn_SA_st = {'FL0_AL0_noHAM' 'FL0_AL0_HAM' 'FL100_AL0_HAM'...
      'FL0_AL50_HAM' 'FL100_AL50_HAM'};
   fn_SA_st2 = strrep(fn_SA_st,'_',' ');

   % State Names (FL0_AL0_noHAM, ...)
   for j = Statenum:Statenum

      % pos_neg = 1, neg_pos = 2
      for k = PN1:PN1

         tp1 = Spine_ALL.(fn_SA{i}).(fn_SA_st{j}).(position).(pathtypestr)...
            .(motion)(trial).(HAM_str).(pathsequence_str)(1,cycle)...
            .(path_name_p2{k}).tool_path_new(rotnum,:);

         le1 = Spine_ALL.(fn_SA{i}).(fn_SA_st{j}).(position).(pathtypestr)...
            .(motion)(trial).(HAM_str).(pathsequence_str)(1,cycle)...
            .(path_name_p2{k}).load_end_pts(rotnum,:);

      end
   end
end


le1 = le1(:);
tp1 = tp1(:);
tp1 = smooth(le1,tp1,10,'moving');


% --- Create fit "DS 1"
ok_ = isfinite(le1) & isfinite(tp1);
if ~all( ok_ )
```

```
    warning( 'GenerateMFile:IgnoringNansAndInfs', ...
        'Ignoring NaNs and Infs in data' );
end
st_ = [0.611339 0.179292 0.969041 0.169972 0.924737 0.5652998 0.275930];
ft_ = fittype('(1/(1+exp(-(a1+b1*L))))*c1+(1/(1+exp(-(a2+b2*L))))*c2+d',...
    'dependent',{'D'},'independent',{'L'},...
    'coefficients',{'a1', 'a2', 'b1', 'b2', 'c1', 'c2', 'd'});

% Fit this model using new data
[cf_,cf_gof] = fit(le1(ok_),tp1(ok_),ft_,'Startpoint',st_);

% Or use coefficients from the original fit:
if 0
    cv_ = { -0.37423, -4.14948, 405.39808, -362.83832, -17.49071};
    [cf_,cf_gof] = cfit(ft_,cv_{:});
end

CF.cf = cf_;
CF.cf_gof = cf_gof;

CF.le1 = le1;
CF.tp1 = tp1;
CF.ROM = tp1(1) - tp1(end);
CF.yfit = cf_(le1);

%% Analysis

CF.coeffnames = coeffnames(CF.cf);
CF.coeffvalues = coeffvalues(CF.cf);

[CF.dydx, CF.d2ydx2] = differentiate(CF.cf, CF.le1);

[CF.max_dydx(1),CF.max_dydx(2)]= max(CF.dydx);
[CF.min_dydx(1),CF.min_dydx(2)]= min(CF.dydx);

[CF.max_d2ydx2(1),CF.max_d2ydx2(2)]= max(CF.d2ydx2);
[CF.min_d2ydx2(1),CF.min_d2ydx2(2)]= min(CF.d2ydx2);

% Plot fit and 1st / 2nd derivative
fh = figure;
subplot(2,2,1), plot(CF.le1,cf_(CF.le1),'r');
hold on
plot(le1,tp1,'x');
title([fn_SA_st2(Statenum) ' Double Sigmoid Fit']);
subplot(2,2,2), plot(CF.le1(1:end),CF.dydx);
title([fn_SA_st2(Statenum) ' First Derivative']);
```

```
subplot(2,2,3), plot(CF.le1(1:end),CF.d2ydx2);
title([fn_SA_st2(Statenum) ' Second Derivative']);

%% Neutral Zone
if PN1 == 1
   CF.le_nz = le1(CF.min_d2ydx2(2):CF.max_d2ydx2(2));
   CF.tp_nz = tp1(CF.min_d2ydx2(2):CF.max_d2ydx2(2));
else
   CF.le_nz = le1(CF.max_d2ydx2(2):CF.min_d2ydx2(2));
   CF.tp_nz = tp1(CF.max_d2ydx2(2):CF.min_d2ydx2(2));
end

% --- Create fit "NZ"
ok_ = isfinite(CF.le_nz) & isfinite(CF.tp_nz);
if ~all( ok_ )
   warning( 'GenerateMFile:IgnoringNansAndInfs', ...
      'Ignoring NaNs and Infs in data' );
end
ft_ = fittype('poly1');

% Fit this model using new data
[nz_,nz_gof] = fit(CF.le_nz(ok_),CF.tp_nz(ok_),ft_);

% Or use coefficients from the original fit:
if 0
   cv_ = { -13.676878594897982, -4.1640363853494824};
   [nz_,nz_gof] = cfit(ft_,cv_{:});
end

CF.NZ = nz_;
CF.NZ_gof = nz_gof;

% Plot this fit
subplot(2,2,4),plot(CF.le_nz,CF.NZ(CF.le_nz),'r');
hold on
plot(CF.le_nz,CF.tp_nz ,'x');
title([fn_SA_st2(Statenum) ' Neutral Zone - Linear Fit']);

CF.NZ_coeffnames = coeffnames(CF.NZ);
CF.NZ_coeffvales = coeffvalues(CF.NZ);
CF.NZ_stiffness = 1/CF.NZ_coeffvales(1);

CF.NZ_le_width = le1(CF.max_d2ydx2(2)) - le1(CF.min_d2ydx2(2));
CF.NZ_tp_width = tp1(CF.max_d2ydx2(2)) - tp1(CF.min_d2ydx2(2));

CF.NZ_ALL = [CF.NZ_stiffness,CF.NZ_le_width,CF.NZ_tp_width, abs(CF.ROM)...
```

```
    , CF.cf_gof.rsquare];

% Edited to save neg_pos w/o overwriting pos_neg CF=CF2, NZ_ALL=NZ_ALL2
Spine_ALL.(fn_SA{i}).(fn_SA_st{j}).(CFvar) = CF;
Spine_ALL.(fn_SA{i}).(NZ_ALLvar)(Statenum,:) = CF.NZ_ALL;

% Moment Figure
savestr = [(fn_SA{i}) '\NZ\' (fn_SA{i}) '_' (fn_SA_st{j}) '_' NZvar];
saveas(fh,['Z:\Ortho Research 3\FergusonLab\Staff\Bell,
Kevin\Kevin\Projects\PhD\Testing\'...
    savestr]);
saveas(fh,['Z:\Ortho Research 3\FergusonLab\Staff\Bell,
Kevin\Kevin\Projects\PhD\Testing\'...
    savestr],'jpg');
%close(fh);
```

A.2.3 Calculate Delta IDP and IDP-Moment Curve Fit

```
function [Spine_ALL] = DS_createFit_IDP(Spine_ALL, SP, SN)

% Kevin Bell
% Updated 5/31/2013

position = 'LAT0_FE0_AXIAL0';
motion = 'pathseek';
pathsequence_str = 'loop';
trial = 1;
% Currently only plotting 2nd cycle
cycle = 2;
color_str1 = {'bo', 'ro', 'ko', 'go', 'mo'};
color_str2 = {'b+', 'r+', 'k+', 'g+', 'm+'};

pathtypestr = 'LAT';

pos = 0; % 1 = pos, 0 = neg

path_name_p1 = {'loop_0_pos'; 'loop_pos_neg'; 'loop_neg_pos'};
path_name_p2 = {'loop_pos_neg'; 'loop_neg_pos'; 'loop_pos_0'};
path_name_n1 = {'loop_0_neg'; 'loop_neg_pos'; 'loop_pos_neg'};
path_name_n2 = {'loop_neg_pos'; 'loop_pos_neg'; 'loop_neg_0'};

%Spine_ALL is a "Super Structure" that contains multiple Spine structures
```

```
fn_SA = fieldnames(Spine_ALL);

SPnum = SP;
Statenum = SN;
rotnum = 6;

C45 = 1;
C56 = 2;

if Statenum == 1
   HAM_str = 'noHAM';
else
   HAM_str = 'HAM';
end

for i = SPnum:SPnum

   %fn_SA_st = fieldnames(Spine_ALL.(fn_SA{i}));
   fn_SA_st = {'FL0_AL0_noHAM' 'FL0_AL0_HAM' 'FL100_AL0_HAM'...
      'FL0_AL50_HAM' 'FL100_AL50_HAM'};
   fn_SA_st2 = strrep(fn_SA_st,'_',' ');

   % State Names (FL0_AL0_noHAM, ...)
   for j = Statenum:Statenum

      if j>3
         trial = 1;
      end

      % pos_neg = 1, neg_pos = 2
      for k = 1:1

         % Changed to minpt maxpt to avoid neutral zone
         zeropt = Spine_ALL.(fn_SA{i}).(fn_SA_st{j}).(position)...
            .(pathtypestr).(motion)(trial).(HAM_str).(pathsequence_str)...
            (1,cycle).(path_name_p2{k}).tool_zero(2);
         if k == 1
            minpt = Spine_ALL.(fn_SA{i}).(fn_SA_st{j}).CF.min_d2ydx2(2);
            maxpt = Spine_ALL.(fn_SA{i}).(fn_SA_st{j}).CF.max_d2ydx2(2);
         elseif k == 2
            error('not setup for neg_pos yet')
         end

         if k == 1
            IDP45_p = Spine_ALL.(fn_SA{i}).(fn_SA_st{j}).(position)...
               .(pathtypestr).(motion)(trial).(HAM_str)...
```

```
                    .(pathsequence_str)(1,cycle).(path_name_p2{k}).IDP(1:minpt,C45);
                IDP45_n = Spine_ALL.(fn_SA{i}).(fn_SA_st{j}).(position)...
                    .(pathtypestr).(motion)(trial).(HAM_str)...
                    .(pathsequence_str)(1,cycle).(path_name_p2{k}).IDP(maxpt:end,C45);
                IDP56_p = Spine_ALL.(fn_SA{i}).(fn_SA_st{j}).(position)...
                    .(pathtypestr).(motion)(trial).(HAM_str)...
                    .(pathsequence_str)(1,cycle).(path_name_p2{k}).IDP(1:minpt,C56);
                IDP56_n = Spine_ALL.(fn_SA{i}).(fn_SA_st{j}).(position)...
                    .(pathtypestr).(motion)(trial).(HAM_str)...
                    .(pathsequence_str)(1,cycle).(path_name_p2{k}).IDP(maxpt:end,C56);
                lep = Spine_ALL.(fn_SA{i}).(fn_SA_st{j}).(position)...
                    .(pathtypestr).(motion)(trial).(HAM_str).(pathsequence_str)...
                    (1,cycle).(path_name_p2{k}).load_end_pts(rotnum,1:minpt)';
                len = Spine_ALL.(fn_SA{i}).(fn_SA_st{j}).(position)...
                    .(pathtypestr).(motion)(trial).(HAM_str)...
                    .(pathsequence_str)(1,cycle).(path_name_p2{k})...
                    .load_end_pts(rotnum,maxpt:end)';
            elseif k == 2
                error('not setup for neg_pos yet')
                IDP45_n = Spine_ALL.(fn_SA{i}).(fn_SA_st{j}).(position)...
                    .(pathtypestr).(motion)(trial).(HAM_str).(pathsequence_str)...
                    (1,cycle).(path_name_p2{k}).IDP(1:zeropt,C45);
                IDP45_p = Spine_ALL.(fn_SA{i}).(fn_SA_st{j}).(position)...
                    .(pathtypestr).(motion)(trial).(HAM_str).(pathsequence_str)...
                    (1,cycle).(path_name_p2{k}).IDP(zeropt:end,C45);
                IDP56_n = Spine_ALL.(fn_SA{i}).(fn_SA_st{j}).(position)...
                    .(pathtypestr).(motion)(trial).(HAM_str).(pathsequence_str)...
                (1,cycle).(path_name_p2{k}).IDP(1:zeropt,C56);
                IDP56_p = Spine_ALL.(fn_SA{i}).(fn_SA_st{j}).(position)...
                    .(pathtypestr).(motion)(trial).(HAM_str).(pathsequence_str)...
                    (1,cycle).(path_name_p2{k}).IDP(zeropt:end,C56);
                len = Spine_ALL.(fn_SA{i}).(fn_SA_st{j}).(position)...
                    .(pathtypestr).(motion)(trial).(HAM_str).(pathsequence_str)...
                    (1,cycle).(path_name_p2{k}).load_end_pts(rotnum,1:zeropt)';
                lep = Spine_ALL.(fn_SA{i}).(fn_SA_st{j}).(position)...
                    .(pathtypestr).(motion)(trial).(HAM_str).(pathsequence_str)...
                    (1,cycle).(path_name_p2{k}).load_end_pts(rotnum,zeropt:end)';
            end

        end
    end
end

%tp1 = smooth(le1,tp1,50,'moving');

fh=figure
```

```
%% IDP45 - Linear Curve Fitting (Positive)

IDPCF.IDP45_p = IDP45_p
IDPCF.lep = lep

% --- Create fit "NZ"
ok_ = isfinite(IDPCF.lep) & isfinite(IDPCF.IDP45_p);
if ~all( ok_ )
   warning( 'GenerateMFile:IgnoringNansAndInfs', ...
      'Ignoring NaNs and Infs in data' );
end
ft_ = fittype('poly1');

% Fit this model using new data
[idpcf_,idpcf_gof] = fit(IDPCF.lep(ok_),IDPCF.IDP45_p(ok_),ft_);

% Or use coefficients from the original fit:
if 0
   cv_ = { -173.64012402461469, -20.526413411621768};
   [idpcf_,idpcf_gof] = cfit(ft_,cv_{:});
end

IDPCF.p45 = idpcf_;
IDPCF.p45_gof = idpcf_gof;

% Plot this fit
subplot(2,1,1),plot(IDPCF.lep,IDPCF.p45(IDPCF.lep),'r');
hold on
plot(IDPCF.lep,IDPCF.IDP45_p ,'.');
hold on
%title([fn_SA_st2(Statenum) 'C45 IDP - Linear Fit']);

IDPCF.p45_coeffnames = coeffnames(IDPCF.p45);
IDPCF.p45_coeffvales = coeffvalues(IDPCF.p45);
IDPCF.p45_stiffness = IDPCF.p45_coeffvales(1);
IDPCF.p45_intercept = IDPCF.p45_coeffvales(2);

Spine_ALL.(fn_SA{i}).(fn_SA_st{j}).IDPCF = IDPCF;
%% IDP45 - Linear Curve Fitting (Negative)

IDPCF.IDP45_n = IDP45_n;
IDPCF.len = len;

% --- Create fit "NZ"
ok_ = isfinite(IDPCF.len) & isfinite(IDPCF.IDP45_n);
```

```
if ~all( ok_ )
    warning( 'GenerateMFile:IgnoringNansAndInfs', ...
        'Ignoring NaNs and Infs in data' );
end
ft_ = fittype('poly1');

% Fit this model using new data
[idpcf_,idpcf_gof] = fit(IDPCF.len(ok_),IDPCF.IDP45_n(ok_),ft_);

% Or use coefficients from the original fit:
if 0
    cv_ = { -13.676878594897982, -4.1640363853494824};
    [idpcf_,idpcf_gof] = cfit(ft_,cv_{:});
end

IDPCF.n45 = idpcf_;
IDPCF.n45_gof = idpcf_gof;

% Plot this fit
subplot(2,1,1),plot(IDPCF.len,IDPCF.n45(IDPCF.len),'k');
hold on
plot(IDPCF.len,IDPCF.IDP45_n ,'o');
hold on


IDPCF.n45_coeffnames = coeffnames(IDPCF.n45);
IDPCF.n45_coeffvales = coeffvalues(IDPCF.n45);
IDPCF.n45_stiffness = IDPCF.n45_coeffvales(1);
IDPCF.n45_intercept = IDPCF.n45_coeffvales(2);

IDPCF.crossx45 = (IDPCF.n45_coeffvales(2)-IDPCF.p45_coeffvales(2))/...
    (IDPCF.p45_coeffvales(1)-IDPCF.n45_coeffvales(1));
IDPCF.crossy45 =
IDPCF.n45_coeffvales(1)*IDPCF.crossx45+IDPCF.n45_coeffvales(2);

IDPCF.pn45_ALL = [IDPCF.p45_stiffness, IDPCF.p45_intercept,...
    IDPCF.p45_gof.rsquare IDPCF.n45_stiffness, IDPCF.n45_intercept,...
    IDPCF.n45_gof.rsquare, IDPCF.crossy45];

Spine_ALL.(fn_SA{i}).(fn_SA_st{j}).IDPCF = IDPCF;
Spine_ALL.(fn_SA{i}).IDPCF_45(Statenum,:) = IDPCF.pn45_ALL;

plot(IDPCF.crossx45,IDPCF.crossy45 ,'Color','m','Marker','x',...
    'MarkerSize',15, 'LineWidth', 3, 'LineStyle','none');

title([fn_SA_st2(Statenum) 'C45 IDP - Linear Fit']);
```

```
xlabel('Moment (Nm)');
ylabel(['IDP (PSI)']);
legend('Flexion (fit)', 'Flexion','Extension (fit)', 'Extension', 'Cross');

%% IDP56 - Linear Curve Fitting (Positive)

IDPCF.IDP56_p = IDP56_p;
IDPCF.lep = lep;

% --- Create fit "NZ"
ok_ = isfinite(IDPCF.lep) & isfinite(IDPCF.IDP56_p);
if ~all( ok_ )
   warning( 'GenerateMFile:IgnoringNansAndInfs', ...
      'Ignoring NaNs and Infs in data' );
end
ft_ = fittype('poly1');

% Fit this model using new data
[idpcf_,idpcf_gof] = fit(IDPCF.lep(ok_),IDPCF.IDP56_p(ok_),ft_);

% Or use coefficients from the original fit:
if 0
   cv_ = { -173.64012402461469, -20.526413411621768};
   [idpcf_,idpcf_gof] = cfit(ft_,cv_{:});
end

IDPCF.p56 = idpcf_;
IDPCF.p56_gof = idpcf_gof;

% Plot this fit
subplot(2,1,2),plot(IDPCF.lep,IDPCF.p56(IDPCF.lep),'r');
hold on
plot(IDPCF.lep,IDPCF.IDP56_p ,'.');
hold on
%title([fn_SA_st2(Statenum) 'C56 IDP - Linear Fit']);

IDPCF.p56_coeffnames = coeffnames(IDPCF.p56);
IDPCF.p56_coeffvales = coeffvalues(IDPCF.p56);
IDPCF.p56_stiffness = IDPCF.p56_coeffvales(1);
IDPCF.p56_intercept = IDPCF.p56_coeffvales(2);

% IDPCF.p56_ALL = [IDPCF.p56_stiffness, IDPCF.p56_gof.rsquare];

Spine_ALL.(fn_SA{i}).(fn_SA_st{j}).IDPCF = IDPCF;
% Spine_ALL.(fn_SA{i}).IDPCF_p56(Statenum,:) = IDPCF.p56_ALL;
```

```
%% IDP56 - Linear Curve Fitting (Negative)

IDPCF.IDP56_n = IDP56_n;
IDPCF.len = len;

% --- Create fit "NZ"
ok_ = isfinite(IDPCF.len) & isfinite(IDPCF.IDP56_n);
if ~all( ok_ )
   warning( 'GenerateMFile:IgnoringNansAndInfs', ...
      'Ignoring NaNs and Infs in data' );
end
ft_ = fittype('poly1');

% Fit this model using new data
[idpcf_,idpcf_gof] = fit(IDPCF.len(ok_),IDPCF.IDP56_n(ok_),ft_);

% Or use coefficients from the original fit:
if 0
   cv_ = { -13.676878594897982, -4.1640363853494824};
   [idpcf_,idpcf_gof] = cfit(ft_,cv_{:});
end

IDPCF.n56 = idpcf_;
IDPCF.n56_gof = idpcf_gof;

% Plot this fit
subplot(2,1,2),plot(IDPCF.len,IDPCF.n56(IDPCF.len),'k');
hold on
plot(IDPCF.len,IDPCF.IDP56_n ,'o');
hold on

IDPCF.n56_coeffnames = coeffnames(IDPCF.n56);
IDPCF.n56_coeffvales = coeffvalues(IDPCF.n56);
IDPCF.n56_stiffness = IDPCF.n56_coeffvales(1);
IDPCF.n56_intercept = IDPCF.n56_coeffvales(2);

IDPCF.crossx56 = (IDPCF.n56_coeffvales(2)-IDPCF.p56_coeffvales(2))/...
   (IDPCF.p56_coeffvales(1)-IDPCF.n56_coeffvales(1));
IDPCF.crossy56 =
IDPCF.n56_coeffvales(1)*IDPCF.crossx56+IDPCF.n56_coeffvales(2);

IDPCF.pn56_ALL = [IDPCF.p56_stiffness, IDPCF.p56_intercept,...
   IDPCF.p56_gof.rsquare IDPCF.n56_stiffness, IDPCF.n56_intercept,...
   IDPCF.n56_gof.rsquare, IDPCF.crossy56];
```

```
Spine_ALL.(fn_SA{i}).(fn_SA_st{j}).IDPCF = IDPCF;
Spine_ALL.(fn_SA{i}).IDPCF_56(Statenum,:) = IDPCF.pn56_ALL;

plot(IDPCF.crossx56,IDPCF.crossy56 ,'Color','m','Marker','x',...
    'MarkerSize',15, 'LineWidth', 3, 'LineStyle','none');

title([fn_SA_st2(Statenum) 'C56 IDP - Linear Fit']);
xlabel('Moment (Nm)');
ylabel(['IDP (PSI)']);
legend('Flexion (fit)', 'Flexion','Extension (fit)', 'Extension', 'Cross');

savestr = [(fn_SA{i}) '\IDP\' (fn_SA{i}) '_IDP_' (fn_SA_st{j})];
saveas(fh,['Z:\Ortho Research 3\FergusonLab\Staff\Bell,
Kevin\Kevin\Projects\PhD\Testing\'...
    savestr]);
saveas(fh,['Z:\Ortho Research 3\FergusonLab\Staff\Bell,
Kevin\Kevin\Projects\PhD\Testing\'...
    savestr],'jpg');
```

A.2.4 Calculate Hysteresis from Moment-Rotation Curve

```
function [Spine_ALL] = DS_Hysteresis(Spine_ALL, SP, SN)

% Kevin Bell
% Updated 5/31/2013

position = 'LAT0_FE0_AXIAL0';
motion = 'pathseek';
pathsequence_str = 'loop';
trial = 1;
% Only Plotting 2nd cycle currently
cycle = 2;
color_str1 = {'bo', 'ro', 'ko', 'go', 'mo'};
color_str2 = {'b+', 'r+', 'k+', 'g+', 'm+'};

pathtypestr = 'LAT';

pos = 0; % 1 = pos, 0 = neg

path_name_p1 = {'loop_0_pos'; 'loop_pos_neg'; 'loop_neg_pos'};
path_name_p2 = {'loop_pos_neg'; 'loop_neg_pos'; 'loop_pos_0'};
path_name_n1 = {'loop_0_neg'; 'loop_neg_pos'; 'loop_pos_neg'};
path_name_n2 = {'loop_neg_pos'; 'loop_pos_neg'; 'loop_neg_0'};
```

```
%Spine_ALL is a "Super Structure" that contains multiple Spine structures
fn_SA = fieldnames(Spine_ALL);

SPnum = SP;
Statenum = SN;
rotnum = 6;

if Statenum == 1
   HAM_str = 'noHAM';
else
   HAM_str = 'HAM';
end

for i = SPnum:SPnum

   %fn_SA_st = fieldnames(Spine_ALL.(fn_SA{i}));
   fn_SA_st = {'FL0_AL0_noHAM' 'FL0_AL0_HAM' 'FL100_AL0_HAM'...
      'FL0_AL50_HAM' 'FL100_AL50_HAM'};
   fn_SA_st2 = strrep(fn_SA_st,'_',' ');

   % State Names (FL0_AL0_noHAM, ...)
   for j = Statenum:Statenum

         tp_pn = Spine_ALL.(fn_SA{i}).(fn_SA_st{j}).CF.tp1;
         le_pn = Spine_ALL.(fn_SA{i}).(fn_SA_st{j}).CF.le1;
         cf_pn = Spine_ALL.(fn_SA{i}).(fn_SA_st{j}).CF.cf;

         tp_np = Spine_ALL.(fn_SA{i}).(fn_SA_st{j}).CF2.tp1;
         le_np = Spine_ALL.(fn_SA{i}).(fn_SA_st{j}).CF2.le1;
         cf_np = Spine_ALL.(fn_SA{i}).(fn_SA_st{j}).CF2.cf;

         fh = figure;
         subplot(2,1,1)
         plot(cf_pn,le_pn,tp_pn,'xb');
         hold on
         plot(cf_np,le_np,tp_np,'xr');
         title([fn_SA_st2(Statenum) ' Flexion/Extension vs Moment']);

%           areaPN = trapz(abs(le_pn),(tp_pn));
%           areaNP = trapz(abs(le_np),(tp_np));

         moment = [-2:.1:2];

         areaPN_int = integrate(cf_pn, moment, -2)
         areaNP_int = integrate(cf_np, moment, -2)
```

```
            subplot(2,1,2)
            plot(moment,(areaPN_int),'xb');
            hold on
            plot(moment,(areaNP_int),'xr');
            title([fn_SA_st2(Statenum) ' Integral']);

%           Hysteresis = abs(abs(areaNP) - abs(areaPN));

            Hysteresis_int = (areaPN_int(end)) - (areaNP_int(end));

            %HysteresisNorm(pnc,3) = Hysteresis(pnc,3)/Hysteresis(1,3);

%           Spine_ALL.(fn_SA{i}).(fn_SA_st{j}).Hysteresis.areaPN = areaPN;
%           Spine_ALL.(fn_SA{i}).(fn_SA_st{j}).Hysteresis.areaNP = areaNP;
%           Spine_ALL.(fn_SA{i}).(fn_SA_st{j}).Hysteresis.HYS = Hysteresis;

            Spine_ALL.(fn_SA{i}).(fn_SA_st{j}).Hysteresis.areaPN_int = areaPN_int;
            Spine_ALL.(fn_SA{i}).(fn_SA_st{j}).Hysteresis.areaNP_int = areaNP_int;
            Spine_ALL.(fn_SA{i}).(fn_SA_st{j}).Hysteresis.HYS_int = Hysteresis_int;

%           Spine_ALL.(fn_SA{i}).HYS_ALL(Statenum,:) = [areaPN areaNP Hysteresis];
            Spine_ALL.(fn_SA{i}).HYS_int_ALL(Statenum,:)...
                = [areaPN_int(end) areaNP_int(end) Hysteresis_int];

            % Moment Figure
            savestr = [(fn_SA{i}) '\HYS\' (fn_SA{i}) '_' (fn_SA_st{j}) '_HYS'];
            saveas(fh,['Z:\Ortho Research 3\FergusonLab\Staff\Bell, 
Kevin\Kevin\Projects\PhD\Testing\'...
                savestr]);
            saveas(fh,['Z:\Ortho Research 3\FergusonLab\Staff\Bell, 
Kevin\Kevin\Projects\PhD\Testing\'...
                savestr],'jpg');

        end
    end
```

A.3 VICON DATA COLLECTION

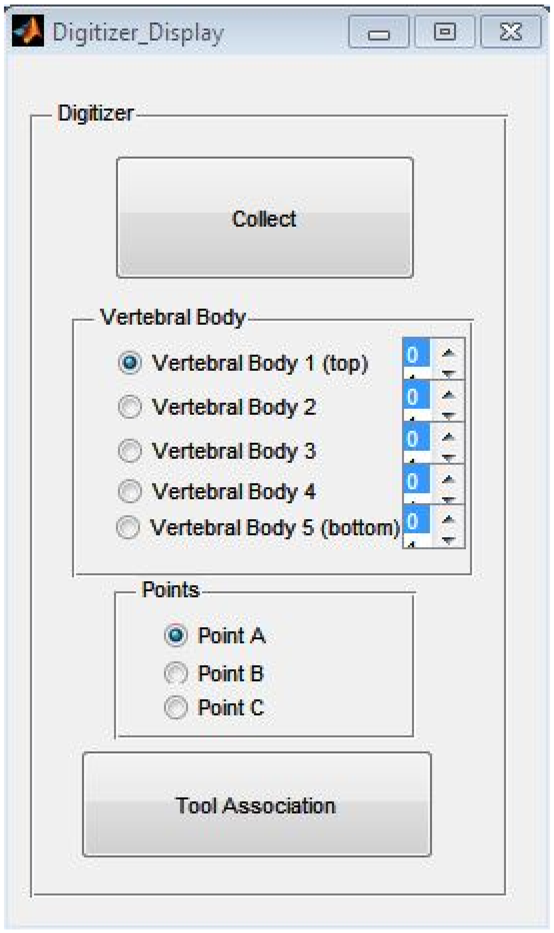


Figure 42. Digitizer Graphical User Interface

A.3.1 Digitization of Anatomical Landmarks

```
function VICON = VICON_Digitize(VICON)

% Kevin Bell
% 5/27/2013

pause on;
recordtime = 5;

% Ask user how many Trans  are there
```

```
question{1}='Enter Number of Vertebral Bodies (Tools):';
prompt = {question{1}};
dlg_title = '';
num_lines = 1;
def = {'4'};
answer = inputdlg(prompt,dlg_title,num_lines,def);
NumberTrans=0;
if isempty(answer)==0
    NumberVBs=str2num(answer{1});
end

for i = 1:NumberVBs

    for ABC = 1:3

        ABC

        ABCcell = {'A', 'B', 'C'};

        % Construct a questdlg with three options
        choice = questdlg(['Are you ready to digitize segment ' num2str(i) ABCcell{ABC}
'?'],['Record for ' num2str(recordtime) ' seconds'],'Yes','Cancel','Yes');
        % Handle response
        switch choice
            case 'Yes'

                tic;

                t = toc;
                h = waitbar(0,['Recording segment ' num2str(i) ABCcell{ABC} '...']);

                while t < recordtime
                    t = toc;
                    % computations take place here
                    waitbar(t / recordtime)
                end
                close(h)

            case 'Cancel'
                error('Digitizer has been cancelled');
        end

    end

end
```

```
pause off;
```

A.3.2 Arrange Digitized Points into an Organized Structure

```
function VICON = Digitize_filter(VICON)

% Kevin Bell
% 5/27/2013

C3Dname = VICON.Options.C3Dname;

trial = 1;
% pathsequence = 1-loop, 2-tail, 3-single

ttotal = [0 0 0 0 0 0];
tarray = [0 0 0 0 0 0];
Digtotal = [0 0 0 0 0 0];
Digarray = [0 0 0 0 0 0];
Digtotal2 = [0 0 0 0 0 0];
Digarray2 = [0 0 0 0 0 0];
% tname_cell = {'_t1','_t2','_t3','_t4','_t5','_t6'}
% tname_cell1 = {'t1','t2','t3','t4','t5','t6'}
% tname_cell = {'_s1','_s2','_s3','_s4'}
tname_cell = {'S1','S2','S3','S4','S5','S6'};
Dig_mark_cell={'T1','T2','T3'};
T_cell={'T'};
Dig_abcd_cell={'Pa','Pb','Pc','Pd'};
frames = VICON.(C3Dname).frames;
markers = VICON.(C3Dname).markers;

temp_mnames = sort(VICON.(C3Dname).mnames);
VICON.(C3Dname).mnames = temp_mnames;

for m = 1:markers
   for mc = 1:length(Dig_mark_cell)
      if isempty(findstr(Dig_mark_cell{mc},char(VICON.(C3Dname).mnames{m})))==0
         Digtotal(mc) = Digtotal(mc) + 1;
         Digarray(mc) = 1;
         tempDignames{mc,Digtotal(mc)} = VICON.(C3Dname).mnames{m};
      end
   end
   for tc=1:length(tname_cell)
      if isempty(findstr(tname_cell{tc},char(VICON.(C3Dname).mnames{m})))==0 &
isempty(findstr(T_cell{1},char(VICON.(C3Dname).mnames{m})))==1
```

```
            ttotal(tc) = ttotal(tc) + 1;
            tarray(tc) = 1;
            tempnames{ttotal(tc),tc} = VICON.(C3Dname).mnames{m};
        end
    end
end

VICON.(C3Dname).Dignames = tempDignames;
VICON.(C3Dname).tnames = tempnames;

for tc = 1:length(tname_cell)
    tcount=0;
    for m = 1:size(VICON.(C3Dname).Dignames,2)
        if isempty(findstr(tname_cell{tc},char(VICON.(C3Dname).Dignames{1,m})))==0
            tc
            tcount=tcount+1;
            Digtotal2(tc) = Digtotal2(tc) + 1;
            Digarray2(tc) = 1;
            for i = 1:3
                tempDignames2{i,tcount,tc} = VICON.(C3Dname).Dignames(i,m);
            end
        end
    end
end

VICON.(C3Dname).Dignames2 = tempDignames2;

size(tempDignames2)

for segnum=1:size(VICON.(C3Dname).Dignames2,3)
    for abc=1:size(VICON.(C3Dname).Dignames2,2)
        for mnum=1:size(VICON.(C3Dname).Dignames2,1)

VICON.(C3Dname).(char(VICON.(C3Dname).Dignames2{mnum,abc,segnum}))(any(isnan(VI
CON.(C3Dname).(char(VICON.(C3Dname).Dignames2{mnum,abc,segnum}))),2),:) = [];

Dig.(tname_cell{segnum})(mnum,:,abc)=mean(VICON.(C3Dname).(char(VICON.(C3Dname).
Dignames2{mnum,abc,segnum})));
        end
    end
end

VICON.(C3Dname).DigValues=Dig;


for segnum=1:size(VICON.(C3Dname).Dignames2,3)
```

```
    for abc=1:size(VICON.(C3Dname).Dignames2,2)

        clear Tmn mn tr

        mn(1,:)=VICON.(C3Dname).DigValues.(tname_cell{segnum})(1,:,abc);
        mn(2,:)=VICON.(C3Dname).DigValues.(tname_cell{segnum})(2,:,abc);
        mn(3,:)=VICON.(C3Dname).DigValues.(tname_cell{segnum})(3,:,abc);

        % Calculating location of origin
        O = mean([mn(1,:);mn(2,:)]);

        X = (mn(2,:)-O);
        X = X/norm(X);

        % can add (-) or inverse cross if markers are missing.
        OZ = (mn(3,:)-O)/norm(mn(3,:)-O);
        Y = cross(OZ,X); Y = Y/norm(Y);
        Z=cross(X,Y); Z=Z/norm(Z);

        Tmn(1:3,1)=X; Tmn(1:3,2)=Y; Tmn(1:3,3)=Z;
        Tmn(1:3,4)=O + (Tmn*[239; .25; -.25])';
        Tmn(4,1:4)=[0 0 0 1];

        VICON.(C3Dname).DigT.(tname_cell{segnum})(:,:,abc) = Tmn;

        tr = [Tmn(1,4) Tmn(2,4) Tmn(3,4)];

        VICON.(C3Dname).Digtr.(tname_cell{segnum})(abc,:) = tr;

    end
end
```

A.3.3 Transform Digitized Points to Align with VICON Tools

```
function VICON = Digitize_link(VICON)

% Kevin Bell
% Updated 5/31/2013

C3Dname = VICON.Options.C3Dname;
ttotal = [0 0 0 0 0 0];
tarray = [0 0 0 0 0 0];
Digtotal = [0 0 0 0 0 0];
```

```
        Digarray = [0 0 0 0 0 0];
        Digtotal2 = [0 0 0 0 0 0];
        Digarray2 = [0 0 0 0 0 0];
        % tname_cell = {'_t1','_t2','_t3','_t4','_t5','_t6'}
        % tname_cell1 = {'t1','t2','t3','t4','t5','t6'}
        % tname_cell = {'_s1','_s2','_s3','_s4'}
        tname_cell = {'S1','S2','S3','S4','S5','S6'};
        Dig_mark_cell={'T1','T2','T3','T4','T5','T6'};
        T_cell={'T'};
        Dig_abcd_cell={'Pa','Pb','Pc','Pd'};
        frames = VICON.(C3Dname).frames;
        markers = VICON.(C3Dname).markers;

        temp_mnames = sort(VICON.(C3Dname).mnames);
        VICON.(C3Dname).mnames = temp_mnames;

        %need to form anatomical CS for each VB -LOOP-
        for toolnum = 1:size(VICON.(C3Dname).Dignames2,3) %# of segments/VBs (Digitized)

          %form anatomical CS (T_G_A) from (A,B,C)
          pointA = VICON.(C3Dname).DigT.(tname_cell{toolnum})(1:3,4,1);
          pointB = VICON.(C3Dname).DigT.(tname_cell{toolnum})(1:3,4,2);
          pointC = VICON.(C3Dname).DigT.(tname_cell{toolnum})(1:3,4,3);

          %function that forms anatomical RF from 3 points
          VICON.(C3Dname).AnatomicalT.(tname_cell{toolnum})(:,:) =
Anatomical(pointA,pointB,pointC);

        end

        %---FIX any remaining NAN prior to calculating rotations---
          for toolnum = 1:size(VICON.(C3Dname).tnames,2) %# of tools (2-6)
             for markernum = 1:size(VICON.(C3Dname).tnames,1) %# of markers per tool
(usually 3)

                temp =
VICON.(C3Dname).(char(VICON.(C3Dname).tnames{markernum,toolnum}));
                temp(any(isnan(temp),2),:) = [];
                VICON.(C3Dname) = rmfield(VICON.(C3Dname),
char(VICON.(C3Dname).tnames{markernum,toolnum}));
                VICON.(C3Dname).(char(VICON.(C3Dname).tnames{markernum,toolnum})) =
temp;
        %           clear temp;
             end
          end
        %to do: update the frames variable based on NaN's removed
        %----finish NaN fixing-------------------------------

        %----collect marker position data for each segment----
          %calculate average position of each marker on each tool over the static trial
          for toolnum = 1:size(VICON.(C3Dname).tnames,2) %loop through all tools

            for markernum = 1:size(VICON.(C3Dname).tnames,1) %loop through each marker
on each tool

        %           disp(toolnum);
        %           disp(markernum);


VICON.(C3Dname).AveragePosition.(VICON.(C3Dname).tnames{markernum,toolnum})(1) =
mean(VICON.(C3Dname).(VICON.(C3Dname).tnames{markernum,toolnum})(:,1));

VICON.(C3Dname).AveragePosition.(VICON.(C3Dname).tnames{markernum,toolnum})(2) =
mean(VICON.(C3Dname).(VICON.(C3Dname).tnames{markernum,toolnum})(:,2));

VICON.(C3Dname).AveragePosition.(VICON.(C3Dname).tnames{markernum,toolnum})(3) =
mean(VICON.(C3Dname).(VICON.(C3Dname).tnames{markernum,toolnum})(:,3));

            end
          end
        %-----------------------------------------------------

        %---calculate transformation for each tool of tool w/r/t global (T_G_M)(M:= measured)--
-
          for toolnum = 1:size(VICON.(C3Dname).Dignames2,3)

            %markers per tool (set of markers) - each mki is (x,y,z)
            mk1 =
VICON.(C3Dname).AveragePosition.(VICON.(C3Dname).tnames{1,toolnum});
            mk2 =
VICON.(C3Dname).AveragePosition.(VICON.(C3Dname).tnames{2,toolnum});
            mk3 =
VICON.(C3Dname).AveragePosition.(VICON.(C3Dname).tnames{3,toolnum});

            % Calculating location of origin
            O = mean([mk1;mk2]);
            X = (mk2-O);
            X = X/norm(X); %toward controller; robot tool x

            % can add (-) or inverse cross if markers are missing.
            OZ = (mk3-O)/norm(mk3-O);
            Y = cross(OZ,X); Y = Y/norm(Y); %robot tool y
            Z=cross(X,Y); Z=Z/norm(Z); %robot tool z

            %test orthogonality
            testxy = dot(X,Y);
            testyz = dot(Y,Z);
            testxz = dot(X,Z);

            %T_G_M: each measured (tool) CS w/r/t Global CS
            T_G_M(1:3,1)=X; T_G_M(1:3,2)=Y; T_G_M(1:3,3)=Z;
            T_G_M(1:3,4)=O;
            T_G_M(4,1:4)=[0 0 0 1];

            %transformation b/w markers and global reference frame
            VICON.(C3Dname).ToolT.(tname_cell{toolnum})(:,:) = T_G_M;

            %translations & rotations in the global RF
            ypr = rad2deg(tr2ypr(T_G_M));
            yprtr = [T_G_M(1,4) T_G_M(2,4) T_G_M(3,4) ypr(1) ypr(2) ypr(3)];
            VICON.(C3Dname).Tooltr.(tname_cell{toolnum}) = yprtr;

        end
    %---saved transformation & ypr of measured (tool) w/r/t global---

    %% anatomical w/r/t measured CS (T_M_A)
    %for each segment/VB, calculate the transformation of the anatomical w/r/t measured
(tool)
    %!!!does the correct segment line up w/ the correct level!!!
    for toolnum = 1:size(VICON.(C3Dname).Dignames2,3) %this should correspond to # of
segments/VBs

    %     %typicvally not necessary (1:4, not 2:5) for digitized points
    %     toolnum_ana = toolnum-1;

        %T_M_A = inv(T_G_M) * T_G_A
        T_M_A = inv(VICON.(C3Dname).ToolT.(tname_cell{toolnum})) *
VICON.(C3Dname).AnatomicalT.(tname_cell{toolnum});

        %transformation saved in ToolAnatomicT
        VICON.(C3Dname).ToolAnatomicT.(tname_cell{toolnum}) = T_M_A;

        %translations & rotations in the global RF
        ypr = rad2deg(tr2ypr((T_M_A(:,:))));
        yprtr = [T_M_A(1,4) T_M_A(2,4) T_M_A(3,4) ypr(1) ypr(2) ypr(3)];
        VICON.(C3Dname).ToolAnatomictr.(tname_cell{toolnum}) = yprtr;
```

```
end
```

A.3.4 Plot VICON Tools and Anatomical Landmarks

```
function VICON = Digitize_plot(VICON)

% Kevin Bell
% Updated 5/31/2013

fh = figure;
hold on

%Temporary Fix
%C3Dname = 'Digitizer01';
C3Dname = VICON.Options.C3Dname;
SPname = VICON.Options.SPname;

tname_cell = {'S1','S2','S3','S4','S5','S6'};

colorindex1 = {'r-*' 'g-o' 'b-+' 'm-*' 'k-*'};
colorindex2 = {'ro' 'go' 'bo' 'mo' 'ko'};
colorindex3 = {'r+' 'g+' 'b+' 'm+' 'k+'};

count_12 = 0;

for segnum=1:size(VICON.(C3Dname).Dignames2,3)
%for segnum=1:4

   for abc=1:size(VICON.(C3Dname).Dignames2,2)

       count_12 = count_12+1;

       mn(1,:)=VICON.(C3Dname).DigValues.(tname_cell{segnum})(1,:,abc);
       mn(2,:)=VICON.(C3Dname).DigValues.(tname_cell{segnum})(2,:,abc);
       mn(3,:)=VICON.(C3Dname).DigValues.(tname_cell{segnum})(3,:,abc);
       mn(4,:)=VICON.(C3Dname).DigValues.(tname_cell{segnum})(1,:,abc);

       distance_12(count_12) = sqrt((mn(1,1)-mn(2,1))^2+(mn(1,2)-mn(2,2))^2+(mn(1,3)-
mn(2,3))^2);

       plot3(mn(:,1),mn(:,2),mn(:,3),char(colorindex1(segnum)), 'linewidth',3);
```

```
text(str2num(mat2str(mn(1,1))),str2num(mat2str(mn(1,2))),str2num(mat2str(mn(1,3)))+15,[char(
tname_cell{segnum}) '.1']);

text(str2num(mat2str(mn(2,1))),str2num(mat2str(mn(2,2))),str2num(mat2str(mn(2,3)))+15,[char(
tname_cell{segnum}) '.2']);

text(str2num(mat2str(mn(3,1))),str2num(mat2str(mn(3,2))),str2num(mat2str(mn(3,3)))+15,[char(
tname_cell{segnum}) '.3']);

            end
        end

        VICON.(C3Dname).distance_12=distance_12;

        grid off
        view(0,20)

        % figure
        hold on

        for segnum=1:size(VICON.(C3Dname).Dignames2,3)
        %for segnum=1:4

              Dabc(1,:)=VICON.(C3Dname).Digtr.(tname_cell{segnum})(1,:);
              Dabc(2,:)=VICON.(C3Dname).Digtr.(tname_cell{segnum})(2,:);
              Dabc(3,:)=VICON.(C3Dname).Digtr.(tname_cell{segnum})(3,:);
              Dabc(4,:)=VICON.(C3Dname).Digtr.(tname_cell{segnum})(1,:);

              plot3(Dabc(:,1),Dabc(:,2),Dabc(:,3),char(colorindex1(segnum)), 'linewidth',3);

        text(str2num(mat2str(Dabc(1,1))),str2num(mat2str(Dabc(1,2))),str2num(mat2str(Dabc(1,
3)))+15,'a');

text(str2num(mat2str(Dabc(2,1))),str2num(mat2str(Dabc(2,2))),str2num(mat2str(Dabc(2,3)))+15
,'b');

text(str2num(mat2str(Dabc(3,1))),str2num(mat2str(Dabc(3,2))),str2num(mat2str(Dabc(3,3)))+15
,'c');

              %Distance from A-B
              distance_AB(segnum) = sqrt((Dabc(1,1)-Dabc(2,1))^2+(Dabc(1,2)-
Dabc(2,2))^2+(Dabc(1,3)-Dabc(2,3))^2);
              VICON.(C3Dname).DistanceAB(segnum) = distance_AB(segnum);
              %Distance from B-C
```

```
                distance_BC(segnum) = sqrt((Dabc(2,1)-Dabc(3,1))^2+(Dabc(2,2)-
Dabc(3,2))^2+(Dabc(2,3)-Dabc(3,3))^2);
                VICON.(C3Dname).DistanceBC(segnum) = distance_BC(segnum);
                %Distance from A-C
                distance_AC(segnum) = sqrt((Dabc(1,1)-Dabc(3,1))^2+(Dabc(1,2)-
Dabc(3,2))^2+(Dabc(1,3)-Dabc(3,3))^2);
                VICON.(C3Dname).DistanceAC(segnum) = distance_AC(segnum);

        end

        % Moment Figure
        savestr = [(SPname) '\VICON\' (SPname) '_DigPlot'];
        saveas(fh,['Z:\Ortho Research 3\FergusonLab\Staff\Bell,
Kevin\Kevin\Projects\PhD\Testing\' savestr]);
        saveas(fh,['Z:\Ortho Research 3\FergusonLab\Staff\Bell,
Kevin\Kevin\Projects\PhD\Testing\' savestr],'jpg');
        %close(fh);

        grid off
```

A.4 POSTPROCESSING – ANALYZE VICON DATA

A.4.1 Identify Rotation Step and Cycles from VICON Data

```
function VICON = VICON_Rotation(VICON)

% Kevin Bell
% Updated 5/31/2013

C3Dname = VICON.Options.C3Dname;
SPname = VICON.Options.SPname;
posneg = VICON.Options.posneg;
pathsequence = VICON.Options.pathsequence;

binum = 1;
if pathsequence == 1
   binum = 2;
end

pnTrans = VICON.(C3Dname).pnTrans;
```

```
if strcmp(posneg,'pos')==1
    pos1 = 1;
else
    pos1 = 0;
end

EE_Correct = [1 0 0 0; 0 1 0 0; 0 0 1 0; 0 0 0 1];
tname_cell = {'S1','S2','S3','S4','S5','S6'};
tname_cell1 = {'s1','s2','s3','s4','s5','s6'};
frames = VICON.(C3Dname).frames;
markers = VICON.(C3Dname).markers;


%% FIX any remaining NAN prior to calculating rotations
for toolnum = 1:size(VICON.(C3Dname).tnames,1)

    for markernum = 1:size(VICON.(C3Dname).tnames,2)

        TEMP = VICON.(C3Dname).(VICON.(C3Dname).tnames{toolnum,markernum});
        NAN =
isnan(VICON.(C3Dname).(VICON.(C3Dname).tnames{toolnum,markernum}));

        nancount = 0;
        stringcount = 0;
        for nannum = 1:size(NAN,1)
            if NAN(nannum,1) == 1 | NAN(nannum,2) == 1 | NAN(nannum,3) == 1
                nancount = nancount + 1;
                stringcount = stringcount + 1;
                nanfix(nancount,1) = nannum;
                nanfix(nancount,2) = stringcount;
                if stringcount > 1
                    nanfix(nancount-1,2)=0;
                end
            else
                stringcount = 0;
            end
        end

        if exist('nanfix','var') == 1
            for nanfixnum = 1:size(nanfix,1)
                if nanfix(nanfixnum,2) > 0
                    warning(['NANs detected: length = '...
                        num2str(nanfix(nanfixnum,2)) ' in '...
                        char((VICON.(C3Dname).tnames{toolnum,markernum}))]);
                    for fixnum = 1:nanfix(nanfixnum,2)
                        TEMP(nanfix(nanfixnum,1)-(nanfix(nanfixnum,2))+(fixnum),:)...
                    = mean([TEMP(nanfix(nanfixnum,1)...
                    -(nanfix(nanfixnum,2)),:);TEMP(nanfix(nanfixnum,1)+1,:)]);
                end
            end
        end
    end

    VICON.(C3Dname).(VICON.(C3Dname).tnames{toolnum,markernum}) = TEMP;

    clear NAN TEMP nanfix

  end
end
%-------------end of NaN fixing - IS THIS NECESSARY AT THIS STEP???-----

for pnnum = 1:pnTrans  %pnTrans is 1-single, 2-tails

  if pathsequence ~= 1
    [path_name] = VICON_path_name(pathsequence,pos1,pnnum)
  end

  for bidirect = 1:binum      %binum is 1 for 'out', 2 for 'back'

    for cyclenum = 1:VICON.(C3Dname).cyclenum  %cycles per motion

      cyclen = cyclenum;

      if pathsequence == 1
        if pos1 == 1
          path_name = 'loop_pos_neg';
          if cyclenum == VICON.(C3Dname).cyclenum
            path_name = 'loop_0_pos';
            cyclen = 1;
          end
        else
          path_name = 'loop_neg_pos';
          if cyclenum == VICON.(C3Dname).cyclenum
            path_name = 'loop_0_neg';
            cyclen = 1;
          end
        end
      end

      for toolnum = 1:size(VICON.(C3Dname).tnames,1) %number of tools

        for markernum = 1:size(VICON.(C3Dname).tnames,2)  %markers on that tool
```

```
                        for stepnum = 1:1:size(VICON.(C3Dname).(path_name).CycleIndex,2)
%indiv. steps per cycle

                          width = 5;

                          CI =
round(VICON.(C3Dname).(path_name).CycleIndex(cyclen,stepnum,bidirect));


VICON.(C3Dname).(path_name).filtered(cyclen).(VICON.(C3Dname).tnames{toolnum,marker
num})(stepnum,1,bidirect) =
mean(VICON.(C3Dname).(VICON.(C3Dname).tnames{toolnum,markernum})(CI-
width:CI+width,1)); %x position (column) of marker throughout cycle (row-step) there & back
(3rd dim)

VICON.(C3Dname).(path_name).filtered(cyclen).(VICON.(C3Dname).tnames{toolnum,marker
num})(stepnum,2,bidirect) =
mean(VICON.(C3Dname).(VICON.(C3Dname).tnames{toolnum,markernum})(CI-
width:CI+width,2));%y

VICON.(C3Dname).(path_name).filtered(cyclen).(VICON.(C3Dname).tnames{toolnum,marker
num})(stepnum,3,bidirect) =
mean(VICON.(C3Dname).(VICON.(C3Dname).tnames{toolnum,markernum})(CI-
width:CI+width,3)); %z

                        end
                      end
                    end
                  end

                end

              end

              assignin('base','VICON',VICON);

              for pnnum = 1:pnTrans %pnTrans 1-single, 2-tails, 2-loop

                if pathsequence ~= 1
                  [path_name] = VICON_path_name(pathsequence,pos1,pnnum)
                end

                for bidirect = 1:binum

                  for cyclenum = 1:VICON.(C3Dname).cyclenum  %cycles per motion
```

```
                    cyclen = cyclenum;

                    if pathsequence == 1
                        if pos1 == 1
                            path_name = 'loop_pos_neg';
                            if cyclenum == VICON.(C3Dname).cyclenum
                                path_name = 'loop_0_pos';
                                cyclen = 1;
                            end
                        else
                            path_name = 'loop_neg_pos';
                            if cyclenum == VICON.(C3Dname).cyclenum
                                path_name = 'loop_0_neg';
                                cyclen = 1;
                            end
                        end
                    end

                    for toolnum = 1:size(VICON.(C3Dname).tnames,1)

                        for stepnum = 1:1:size(VICON.(C3Dname).(path_name).CycleIndex,2)

                            % Marker position (x,y,z) per step per tool per cycle out and back
                            mn1 =
VICON.(C3Dname).(path_name).filtered(cyclen).(VICON.(C3Dname).tnames{toolnum,1})(step
num,:,bidirect); %marker position (x,y,z)
                            mn2 =
VICON.(C3Dname).(path_name).filtered(cyclen).(VICON.(C3Dname).tnames{toolnum,2})(step
num,:,bidirect);
                            mn3 =
VICON.(C3Dname).(path_name).filtered(cyclen).(VICON.(C3Dname).tnames{toolnum,3})(step
num,:,bidirect);

                            % Calculating location of origin
                            O = mean([mn1;mn2]);

                            X = (mn2-O);
                            X = X/norm(X);

                            % can add (-) or invers cross if markers are missing.
                            OZ = (mn3-O)/norm(mn3-O);
                            Y = cross(OZ,X); Y = Y/norm(Y);
                            Z=cross(X,Y); Z=Z/norm(Z);

                            %test orthogonality
                            testxy = dot(X,Y);
                            testyz = dot(Y,Z);
                            testxz = dot(X,Z);

                            %form marker-to-global transformation matrix from marker positions
                            Tmn(1:3,1)=X; Tmn(1:3,2)=Y; Tmn(1:3,3)=Z;
                            Tmn(1:3,4)=O;
                            Tmn(4,1:4)=[0 0 0 1];

                            %save transformation matrix (T_G_M) and computed ypr

VICON.(C3Dname).(path_name).Transform(cyclen).T_G.(tname_cell1{toolnum})(:,:,stepnum,b
idirect) = Tmn;

                            ypr = rad2deg(tr2ypr((Tmn(:,:))));
                            yprtr = [Tmn(1,4) Tmn(2,4) Tmn(3,4) ypr(1) ypr(2) ypr(3)];


VICON.(C3Dname).(path_name).Transform(cyclen).yprtr_G.(tname_cell1{toolnum})(stepnum,:
,bidirect) = yprtr;


                            %deal with 179-0 issue
                            for i=4:6
                                if
range(VICON.(C3Dname).(path_name).Transform(cyclen).yprtr_G.(tname_cell1{toolnum})(:,i,
bidirect)) > 180 &&
range(VICON.(C3Dname).(path_name).Transform(cyclen).yprtr_G.(tname_cell1{toolnum})(ste
pnum,i,bidirect)) < 181
                              if
mean(VICON.(C3Dname).(path_name).Transform(cyclen).yprtr_G.(tname_cell1{toolnum})(:,i,
bidirect)) -
VICON.(C3Dname).(path_name).Transform(cyclen).yprtr_G.(tname_cell1{toolnum})(stepnum,i
,bidirect) > abs(90)  %abnormal data point


VICON.(C3Dname).(path_name).Transform(cyclen).yprtr_G.(tname_cell1{toolnum})(stepnum,i
,bidirect) = 180 -
VICON.(C3Dname).(path_name).Transform(cyclen).yprtr_G.(tname_cell1{toolnum})(stepnum,i
,bidirect);

                              end
                                end
                            end
                            if toolnum == 1
                                Tmn = Tmn*EE_Correct; % Confirm subscripts
                            end

                            if stepnum==1
                                Tmn_0 = Tmn;
                            end

                            %---temporal: measured in the global CS----
                            %w/r/t t=0 -> in the local (w/r/t itself)
                            Tmn_02N = inv(Tmn_0)*Tmn;
                            %save in struct

VICON.(C3Dname).(path_name).Transform(cyclen).T_G_02N.(tname_cell1{toolnum})(:,:,step
num,bidirect) = Tmn_02N;

                            ypr_02N = rad2deg(tr2ypr((Tmn_02N(:,:))));
                            yprtr_02N = [Tmn_02N(1,4) Tmn_02N(2,4) Tmn_02N(3,4) ypr_02N(1)
ypr_02N(2) ypr_02N(3)];

                            %transl & rot in the local RF

VICON.(C3Dname).(path_name).Transform(cyclen).yprtr_G_02N.(tname_cell1{toolnum})(step
num,:,bidirect) = yprtr_02N;
                            %-----------------------------------------

                            %----get transformations in anatomical CS-----
                            %using T_M_A from <Digitize_link.m> -> MUST RUN BEFORE
                            % Made Digitizer01 static and created tname_cell
                            % - Dependent on "Digitize01" but naming of digitizer C3D may be different
& cause error
                            % Temporary Adjustment
                            T_G_A = Tmn *
VICON.Digitizer01.ToolAnatomicT.(tname_cell{toolnum}); %:=T_M_A
                            % T_G_A = Tmn * EE_Correct;
                            %save in struct

VICON.(C3Dname).(path_name).Transform(cyclen).T_G_A.(tname_cell1{toolnum})(:,:,stepnu
m,bidirect) = T_G_A;

                            %temporal: anatomical to anatomical
                            %T_G_A_02N = inv(T_G_A_0) * T_G_A at each stepnum
                            T_G_A_02N =
inv(VICON.(C3Dname).(path_name).Transform(cyclen).T_G_A.(tname_cell1{toolnum})(:,:,1,b
idirect)) *
```

```
VICON.(C3Dname).(path_name).Transform(cyclen).T_G_A.(tname_cell1{toolnum})(:,:,stepnu
m,bidirect);
                    %save in struct

VICON.(C3Dname).(path_name).Transform(cyclen).T_G_A_02N.(tname_cell1{toolnum})(:,:,st
epnum,bidirect) = T_G_A_02N;

                  end
                end
              end
            end

          end

          numtools = size(VICON.(C3Dname).tnames,1);

          for pnnum = 1:pnTrans %1-single, 2-tails

            [path_name] = VICON_path_name(pathsequence,pos1,pnnum);

            for bidirect = 1:binum

              for cyclenum = 1:VICON.(C3Dname).cyclenum  %cycles per motion

                cyclen = cyclenum;

                if pathsequence == 1
                  if pos1 == 1
                    path_name = 'loop_pos_neg';
                    if cyclenum == VICON.(C3Dname).cyclenum
                      path_name = 'loop_0_pos';
                      cyclen = 1;
                    end
                  else
                    path_name = 'loop_neg_pos';
                    if cyclenum == VICON.(C3Dname).cyclenum
                      path_name = 'loop_0_neg';
                      cyclen = 1;
                    end
                  end
                end

                for stepnum = 1:1:size(VICON.(C3Dname).(path_name).CycleIndex,2)


                  for toolnum = 1:numtools
```

```
                        % Create temporary for T_G_M
                        TG = ['T_G' num2str(toolnum)];
                        TGcell{toolnum} = TG;
                        tempG.(TG) =
VICON.(C3Dname).(path_name).Transform(cyclen).T_G.(tname_cell1{toolnum})(:,:,stepnum,b
idirect);
                        % Create temporary for T_G_A
                        TGA = ['T_G_A' num2str(toolnum)];
                        TGAcell{toolnum} = TGA;
                        tempA.(TGA) =
VICON.(C3Dname).(path_name).Transform(cyclen).T_G_A.(tname_cell1{toolnum})(:,:,stepnu
m,bidirect);

                     end

                     for toolnum = 1:numtools

                        %set up naming for transformations
                        Tseg = ['T_' num2str(numtools) num2str(toolnum)];
                        % Tsegcell = w/r/t base (T_51, T_52, etc)
                        Tsegcell{toolnum} = Tseg;
                        yprseg = ['ypr_' num2str(numtools) num2str(toolnum)];
                        yprtrseg = ['yprtr_' num2str(numtools) num2str(toolnum)];

                        % Global w/r/t base (T & ypr)
                        tempG.(Tseg) =
inv(tempG.(TGcell{numtools}))*tempG.(TGcell{toolnum});

VICON.(C3Dname).(path_name).Transform(cyclen).global.(Tseg)(:,:,stepnum,bidirect) =
tempG.(Tseg);
                        tempG.(yprseg) = rad2deg(tr2ypr(tempG.(Tseg)(:,:)));
                        tempG.(yprtrseg) = [tempG.(Tseg)(1,4) tempG.(Tseg)(2,4)
tempG.(Tseg)(3,4) tempG.(yprseg)(1) tempG.(yprseg)(2) tempG.(yprseg)(3)];

VICON.(C3Dname).(path_name).Transform(cyclen).global.(yprtrseg)(stepnum,:,bidirect) =
tempG.(yprtrseg);

                        % Anatomical w/r/t base (T & ypr)
                        tempA.(Tseg) =
inv(tempA.(TGAcell{numtools}))*tempA.(TGAcell{toolnum});

VICON.(C3Dname).(path_name).Transform(cyclen).anatomical.(Tseg)(:,:,stepnum,bidirect) =
tempA.(Tseg);
                        tempA.(yprseg) = rad2deg(tr2ypr(tempA.(Tseg)(:,:)));
```

```
                    tempA.(yprtrseg) = [tempA.(Tseg)(1,4) tempA.(Tseg)(2,4)
tempA.(Tseg)(3,4) tempA.(yprseg)(1) tempA.(yprseg)(2) tempA.(yprseg)(3)];

VICON.(C3Dname).(path_name).Transform(cyclen).anatomical.(yprtrseg)(stepnum,:,bidirect) =
tempA.(yprtrseg);

                  end

                  if numtools > 1
                    numseg = numtools-1;

                    for toolnum = 1:numseg

                      Tseg = ['T_' num2str(toolnum+1) num2str(toolnum)];
                      % Tsecell2 = intersegmental (T_21, T_32, etc)
                      Tsegcell2{toolnum} = Tseg;
                      yprseg = ['ypr_' num2str(toolnum+1) num2str(toolnum)];
                      yprtrseg = ['yprtr_' num2str(toolnum+1) num2str(toolnum)];

                      % Global - intersegmental motion
                      tempG.(Tseg) =
inv(tempG.(Tsegcell{toolnum+1}))*tempG.(Tsegcell{toolnum});

VICON.(C3Dname).(path_name).Transform(cyclen).global.(Tseg)(:,:,stepnum,bidirect) =
tempG.(Tseg);
                      tempG.(yprseg) = rad2deg(tr2ypr(tempG.(Tseg)(:,:)));
                      tempG.(yprtrseg) = [tempG.(Tseg)(1,4) tempG.(Tseg)(2,4)
tempG.(Tseg)(3,4) tempG.(yprseg)(1) tempG.(yprseg)(2) tempG.(yprseg)(3)];

VICON.(C3Dname).(path_name).Transform(cyclen).global.(yprtrseg)(stepnum,:,bidirect) =
tempG.(yprtrseg);

                      % Anatomical - intersegmental motion
                      tempA.(Tseg) =
inv(tempA.(Tsegcell{toolnum+1}))*tempA.(Tsegcell{toolnum});

VICON.(C3Dname).(path_name).Transform(cyclen).anatomical.(Tseg)(:,:,stepnum,bidirect) =
tempA.(Tseg);
                      tempA.(yprseg) = rad2deg(tr2ypr(tempA.(Tseg)(:,:)));
                      tempA.(yprtrseg) = [tempA.(Tseg)(1,4) tempA.(Tseg)(2,4)
tempA.(Tseg)(3,4) tempA.(yprseg)(1) tempA.(yprseg)(2) tempA.(yprseg)(3)];

VICON.(C3Dname).(path_name).Transform(cyclen).anatomical.(yprtrseg)(stepnum,:,bidirect) =
tempA.(yprtrseg);

                    end
                    end

                    if stepnum == 1
                       for toolnum = 1:numtools
                          T0 = [Tsegcell{toolnum} '_0'];
                          T0cell{toolnum} = T0;

                          tempG.(T0) = tempG.(Tsegcell{toolnum});
                          tempA.(T0) = tempA.(Tsegcell{toolnum});

                       end

                       if numtools > 1
                          numseg = numtools-1;

                          for toolnum = 1:numseg

                             T02 = [Tsegcell2{toolnum} '_0'];
                             T0cell2{toolnum} = T02;

                             tempG.(T02) = tempG.(Tsegcell2{toolnum});
                             tempA.(T02) = tempA.(Tsegcell2{toolnum});

                          end
                       end

                    end

                    for toolnum = 1:numtools
                       T02N = [Tsegcell{toolnum} '_02N'];
                       ypr02N = ['ypr_' num2str(numtools) num2str(toolnum) '_02N'];
                       yprtr02N = ['yprtr_' num2str(numtools) num2str(toolnum) '_02N'];

                       % Temporal transformation of Global w/r/t base
                       tempG.(T02N) =
inv(tempG.(T0cell{toolnum}))*tempG.(Tsegcell{toolnum});

VICON.(C3Dname).(path_name).Transform(cyclen).global.(T02N)(:,:,stepnum,bidirect) =
tempG.(T02N);
                       tempG.(ypr02N) = rad2deg(tr2ypr(tempG.(T02N)(:,:)));
                       tempG.(yprtr02N) = [tempG.(T02N)(1,4) tempG.(T02N)(2,4)
tempG.(T02N)(3,4) tempG.(ypr02N)(1) tempG.(ypr02N)(2) tempG.(ypr02N)(3)];

VICON.(C3Dname).(path_name).Transform(cyclen).global.(yprtr02N)(stepnum,:,bidirect) =
tempG.(yprtr02N);
```

```
                        % Temporal transformation of Global w/r/t base
                        tempA.(T02N) =
inv(tempA.(T0cell{toolnum}))*tempA.(Tsegcell{toolnum});

VICON.(C3Dname).(path_name).Transform(cyclen).anatomical.(T02N)(:,:,stepnum,bidirect) =
tempA.(T02N);
                        tempA.(ypr02N) = rad2deg(tr2ypr(tempA.(T02N)(:,:)));
                        tempA.(yprtr02N) = [tempA.(T02N)(1,4) tempA.(T02N)(2,4)
tempA.(T02N)(3,4) tempA.(ypr02N)(1) tempA.(ypr02N)(2) tempA.(ypr02N)(3)];

VICON.(C3Dname).(path_name).Transform(cyclen).anatomical.(yprtr02N)(stepnum,:,bidirect)
= tempA.(yprtr02N);

                    end

                    if numtools > 1
                        numseg = numtools-1;

                        for toolnum = 1:numseg

                            T02N = [Tsegcell2{toolnum} '_02N'];
                            ypr02N = ['ypr_' num2str(toolnum+1) num2str(toolnum) '_02N'];
                            yprtr02N = ['yprtr_' num2str(toolnum+1) num2str(toolnum) '_02N'];

                            % Temporal transformation of Global intersegmental
                            tempG.(T02N) =
inv(tempG.(T0cell2{toolnum}))*tempG.(Tsegcell2{toolnum});

VICON.(C3Dname).(path_name).Transform(cyclen).global.(T02N)(:,:,stepnum,bidirect) =
tempG.(T02N);
                            tempG.(ypr02N) = rad2deg(tr2ypr(tempG.(T02N)(:,:)));
                            tempG.(yprtr02N) = [tempG.(T02N)(1,4) tempG.(T02N)(2,4)
tempG.(T02N)(3,4) tempG.(ypr02N)(1) tempG.(ypr02N)(2) tempG.(ypr02N)(3)];

VICON.(C3Dname).(path_name).Transform(cyclen).global.(yprtr02N)(stepnum,:,bidirect) =
tempG.(yprtr02N);

                            % Temporal transformation of Global intersegmental
                            tempA.(T02N) =
inv(tempA.(T0cell2{toolnum}))*tempA.(Tsegcell2{toolnum});

VICON.(C3Dname).(path_name).Transform(cyclen).anatomical.(T02N)(:,:,stepnum,bidirect) =
tempA.(T02N);
                            tempA.(ypr02N) = rad2deg(tr2ypr(tempA.(T02N)(:,:)));
                            tempA.(yprtr02N) = [tempA.(T02N)(1,4) tempA.(T02N)(2,4)
tempA.(T02N)(3,4) tempA.(ypr02N)(1) tempA.(ypr02N)(2) tempA.(ypr02N)(3)];
```

```
VICON.(C3Dname).(path_name).Transform(cyclen).anatomical.(yprtr02N)(stepnum,:,bidirect)
= tempA.(yprtr02N);

                        end
                    end

                end
            end

        end
    end

    %% Calculate ROM / cycle
    rotnum = 4;

    for cyclenum = 1:size(VICON.(C3Dname).(path_name).CycleIndex,1)

        fh = figure;
        % basic ROM bar plot
        subplot(pnTrans,1,pnnum);
        hold on

        for pnnum = 1:pnTrans

            [path_name] = VICON_path_name(pathsequence,pos1,pnnum);


            for bidirect = 1:1


                for toolnum = 1:1
                    T02N = [Tsegcell{toolnum} '_02N'];
                    ypr02N = ['ypr_' num2str(numtools) num2str(toolnum) '_02N'];
                    yprtr02N = ['yprtr_' num2str(numtools) num2str(toolnum) '_02N'];
                    tr = ['C' num2str(toolnum+2) num2str(toolnum+6)];

                    % NEED to finish = check progess

VICON.(C3Dname).(path_name).ROM.anatomical.ALL.(yprtr02N)(cyclenum,:) =
VICON.(C3Dname).(path_name).Transform(cyclenum).anatomical.(yprtr02N)(end,:,bidirect);
                    subplot(pnTrans,1,pnnum);

bar(1,[VICON.(C3Dname).(path_name).ROM.anatomical.ALL.(yprtr02N)(1,rotnum)]);
                    hold on
                    ticktemp{toolnum} = '';
                    ticktemp{toolnum+1} = tr;

                end

                if numtools > 1
                    numseg = numtools-1;

                    for toolnum = 1:numseg

                        T02N = [Tsegcell2{toolnum} '_02N'];
                        ypr02N = ['ypr_' num2str(toolnum+1) num2str(toolnum) '_02N'];
                        yprtr02N = ['yprtr_' num2str(toolnum+1) num2str(toolnum) '_02N'];
                        tr = ['C' num2str(toolnum+2) num2str(toolnum+3)];

                        % NEED to finish = check progess

VICON.(C3Dname).(path_name).ROM.anatomical.ALL.(yprtr02N)(cyclenum,:) =
VICON.(C3Dname).(path_name).Transform(cyclenum).anatomical.(yprtr02N)(end,:,bidirect);
                        subplot(pnTrans,1,pnnum);

bar(toolnum+1,[VICON.(C3Dname).(path_name).ROM.anatomical.ALL.(yprtr02N)(1,rotnum)])
;
                        hold on

                        ticktemp{2*toolnum+1} = '';
                        ticktemp{2*toolnum+2} = tr;

                    end
                end

                for c = 1:length(C3Dname)
                    if strcmp(C3Dname(c),'_')==1
                        C3Dtitle(c) = ' ';
                    else
                        C3Dtitle(c) = C3Dname(c);
                    end
                end

                for p = 1:length(path_name)
                    if strcmp(path_name(p),'_')==1
                        path_title(p) = ' ';
                    else
                        path_title(p) = path_name(p);
                    end
                end
                title([C3Dtitle]);
                ticktemp
                set(gca,'XTickLabel',ticktemp);

            end
        end
end

% Combinedbar Figure
savestr = [(SPname) '\VICON\' (SPname) '_' (C3Dname) '_Combinedbar'];
saveas(fh,['Z:\Ortho Research 3\FergusonLab\Staff\Bell,
Kevin\Kevin\Projects\PhD\Testing\' savestr]);
saveas(fh,['Z:\Ortho Research 3\FergusonLab\Staff\Bell,
Kevin\Kevin\Projects\PhD\Testing\' savestr],'jpg');
%close(fh);
```

A.4.2 Extract VICON Data from Overall Path

```
function VICON = VICON_Rotation_LoopFix(VICON)

C3Dname = VICON.Options.C3Dname;
SPname = VICON.Options.SPname;
posneg = VICON.Options.posneg;
pathsequence = VICON.Options.pathsequence;

binum = 2;
if pathsequence == 1
    binum = 2;
end

pnTrans = VICON.(C3Dname).pnTrans;

if strcmp(posneg,'pos')==1
    pos1 = 1;
else
    pos1 = 0;
end

numtools = size(VICON.(C3Dname).tnames,1);
looptotal = binum * (VICON.(C3Dname).cyclenum);

temp_ana = [0 0 0 0 0 0];
```

```
color_str2 = {'b+', 'r+', 'k+', 'g+', 'm+'};
rotnum = 4;

for toolnum = 1:numtools

    yprtr02N = ['yprtr_' num2str(numtools) num2str(toolnum) '_02N'];

    for lpt = 1:looptotal

        if pos1 == 1

            if lpt == 1
                path_name = 'loop_0_pos';
                cyclen = 1;
                bidirect = 1;
            elseif lpt == 2
                path_name = 'loop_pos_neg';
                cyclen = 1;
                bidirect = 1;
            elseif lpt == 3
                path_name = 'loop_pos_neg';
                cyclen = 1;
                bidirect = 2;
            elseif lpt == 4
                path_name = 'loop_pos_neg';
                cyclen = 2;
                bidirect = 1;
            elseif lpt == 5
                path_name = 'loop_pos_neg';
                cyclen = 2;
                bidirect = 2;
            elseif lpt == 6
                path_name = 'loop_0_pos';
                cyclen = 1;
                bidirect = 2;
            end

        elseif pos1 == 0

            if lpt == 1
                path_name = 'loop_0_neg';
                cyclen = 1;
                bidirect = 1;
            elseif lpt == 2
                path_name = 'loop_neg_pos';
```

```
                        cyclen = 1;
                        bidirect = 1;
                    elseif lpt == 3
                        path_name = 'loop_neg_pos';
                        cyclen = 1;
                        bidirect = 2;
                    elseif lpt == 4
                        path_name = 'loop_neg_pos';
                        cyclen = 2;
                        bidirect = 1;
                    elseif lpt == 5
                        path_name = 'loop_neg_pos';
                        cyclen = 2;
                        bidirect = 2;
                    elseif lpt == 6
                        path_name = 'loop_0_neg';
                        cyclen = 1;
                        bidirect = 2;
                    end

                end

                if lpt == 6

                    for steps =
1:size(VICON.(C3Dname).(path_name).Transform(cyclen).anatomical.(yprtr02N)(:,:,bidirect),1)


VICON.(C3Dname).(path_name).Transform(cyclen).anatomical_loop.(yprtr02N)(steps,:,bidirect
) =
temp_ana+VICON.(C3Dname).(path_name).Transform(cyclen).anatomical.(yprtr02N)(steps,:,bi
direct);

                    end
                elseif lpt > 1 & lpt < 6
                    for steps =
1:size(VICON.(C3Dname).(path_name).Transform(cyclen).anatomical.(yprtr02N)(:,:,bidirect),1)


VICON.(C3Dname).(path_name).Transform(cyclen).anatomical_loop.(yprtr02N)(steps,:,bidirect
) = temp_ana-
VICON.(C3Dname).(path_name).Transform(cyclen).anatomical.(yprtr02N)(steps,:,bidirect);

                    end

                else
```

```
VICON.(C3Dname).(path_name).Transform(cyclen).anatomical_loop.(yprtr02N)(:,:,bidirect) =
VICON.(C3Dname).(path_name).Transform(cyclen).anatomical.(yprtr02N)(:,:,bidirect);
                    end

                    temp_ana =
VICON.(C3Dname).(path_name).Transform(cyclen).anatomical_loop.(yprtr02N)(end,:,bidirect);

                end

                temp_ana = [0 0 0 0 0 0];

            end


            temp_ana = [0 0 0 0 0 0];

            fh1 = figure;

            if numtools > 1

                numseg = numtools-1;

                for toolnum = 1:numseg

                    yprtr02N = ['yprtr_' num2str(toolnum+1) num2str(toolnum) '_02N'];
                    tr= ['C' num2str(toolnum+2) num2str(toolnum+3)];

                    legend_str{toolnum} = tr;

                    %figure
                    rct = 0;

                    for lpt = 1:looptotal

                        if pos1 == 1

                            if lpt == 1
                                path_name = 'loop_0_pos';
                                cyclen = 1;
                                bidirect = 1;
                            elseif lpt == 2
                                path_name = 'loop_pos_neg';
                                cyclen = 1;
                                bidirect = 1;
                            elseif lpt == 3
```

```
        path_name = 'loop_pos_neg';
        cyclen = 1;
        bidirect = 2;
    elseif lpt == 4
        path_name = 'loop_pos_neg';
        cyclen = 2;
        bidirect = 1;
    elseif lpt == 5
        path_name = 'loop_pos_neg';
        cyclen = 2;
        bidirect = 2;
    elseif lpt == 6
        path_name = 'loop_0_pos';
        cyclen = 1;
        bidirect = 2;
    end

elseif pos1 == 0

    if lpt == 1
        path_name = 'loop_0_neg';
        cyclen = 1;
        bidirect = 1;
    elseif lpt == 2
        path_name = 'loop_neg_pos';
        cyclen = 1;
        bidirect = 1;
    elseif lpt == 3
        path_name = 'loop_neg_pos';
        cyclen = 1;
        bidirect = 2;
    elseif lpt == 4
        path_name = 'loop_neg_pos';
        cyclen = 2;
        bidirect = 1;
    elseif lpt == 5
        path_name = 'loop_neg_pos';
        cyclen = 2;
        bidirect = 2;
    elseif lpt == 6
        path_name = 'loop_0_neg';
        cyclen = 1;
        bidirect = 2;
    end

end
```

```
                if lpt == 6

                    for steps =
1:size(VICON.(C3Dname).(path_name).Transform(cyclen).anatomical.(yprtr02N)(:,:,bidirect),1)


VICON.(C3Dname).(path_name).Transform(cyclen).anatomical_loop.(yprtr02N)(steps,:,bidirect
) =
temp_ana+VICON.(C3Dname).(path_name).Transform(cyclen).anatomical.(yprtr02N)(steps,:,bi
direct);

                        rct = rct + 1;

plot(rct,VICON.(C3Dname).(path_name).Transform(cyclen).anatomical_loop.(yprtr02N)(steps,r
otnum,bidirect),color_str2{toolnum})
                        hold on

                    end
                elseif lpt > 1 & lpt < 6
                    for steps =
1:size(VICON.(C3Dname).(path_name).Transform(cyclen).anatomical.(yprtr02N)(:,:,bidirect),1)


VICON.(C3Dname).(path_name).Transform(cyclen).anatomical_loop.(yprtr02N)(steps,:,bidirect
) = temp_ana-
VICON.(C3Dname).(path_name).Transform(cyclen).anatomical.(yprtr02N)(steps,:,bidirect);
                        rct = rct + 1;

plot(rct,VICON.(C3Dname).(path_name).Transform(cyclen).anatomical_loop.(yprtr02N)(steps,r
otnum,bidirect),color_str2{toolnum})
                        hold on
                    end

                    if bidirect == 2

VICON.(C3Dname).(path_name).Transform(1).anatomical_loop.(tr).ROM(cyclen,1)=VICON.(
C3Dname).(path_name).Transform(cyclen).anatomical_loop.(yprtr02N)(1,rotnum,bidirect);

VICON.(C3Dname).(path_name).Transform(1).anatomical_loop.(tr).ROM(cyclen,2)=VICON.(
C3Dname).(path_name).Transform(cyclen).anatomical_loop.(yprtr02N)(end,rotnum,bidirect);
                    end

                else
```

```
VICON.(C3Dname).(path_name).Transform(cyclen).anatomical_loop.(yprtr02N)(:,:,bidirect) =
VICON.(C3Dname).(path_name).Transform(cyclen).anatomical.(yprtr02N)(:,:,bidirect);
                        for steps =
1:size(VICON.(C3Dname).(path_name).Transform(cyclen).anatomical.(yprtr02N)(:,:,bidirect),1)
                            rct = rct + 1;
                            AH(toolnum) =
plot(rct,VICON.(C3Dname).(path_name).Transform(cyclen).anatomical_loop.(yprtr02N)(steps,r
otnum,bidirect),color_str2{toolnum});
                            hold on
                        end
                    end

                    temp_ana =
VICON.(C3Dname).(path_name).Transform(cyclen).anatomical_loop.(yprtr02N)(end,:,bidirect);

                end

                titlename = strrep(C3Dname, '_','-');
                title(titlename);
                xlabel('Steps');
                ylabel(['Rotation (deg)']);

                temp_ana = [0 0 0 0 0 0];

                end

                legend(AH,legend_str,'Location','BestOutside');

            end

            % Segplot Figure
            savestr = [(SPname) '\VICON\' (SPname) '_' (C3Dname) '_segplot'];
            saveas(fh1,['Z:\Ortho Research 3\FergusonLab\Staff\Bell,
Kevin\Kevin\Projects\PhD\Testing\' savestr]);
            saveas(fh1,['Z:\Ortho Research 3\FergusonLab\Staff\Bell,
Kevin\Kevin\Projects\PhD\Testing\' savestr],'jpg');
            %close(fh1);

            fh2 = figure;

            if numtools > 1

                numseg = numtools-1;

                for toolnum = 1:numseg
                    yprtr02N = ['yprtr_' num2str(toolnum+1) num2str(toolnum) '_02N'];
                    tr= ['C' num2str(toolnum+2) num2str(toolnum+3)];

                    tr_str{toolnum} = tr;

                    if pos1 == 1
                        path_name = 'loop_pos_neg';
                    elseif pos1 == 0
                        path_name = 'loop_neg_pos';
                    end

                    ROM(1) =
mean(VICON.(C3Dname).(path_name).Transform(1).anatomical_loop.(tr).ROM(:,1),1)
                    ROM(2) =
mean(VICON.(C3Dname).(path_name).Transform(1).anatomical_loop.(tr).ROM(:,2),1)

                    bar(toolnum,ROM(1));
                    hold on
                    bar(toolnum,ROM(2));
                    hold on

                    set(gca,'XTickLabel',tr);

                end

                titlename = strrep(C3Dname, '_','-');
                title(titlename);
                set(gca,'XTick',1:1:4)
                set(gca,'XTickLabel',tr_str);
                ylabel(['Rotation (deg)']);

            end

            % Segbar Figure
            savestr = [(SPname) '\VICON\' (SPname) '_' (C3Dname) '_segbar'];
            saveas(fh2,['Z:\Ortho Research 3\FergusonLab\Staff\Bell,
Kevin\Kevin\Projects\PhD\Testing\' savestr]);
            saveas(fh2,['Z:\Ortho Research 3\FergusonLab\Staff\Bell,
Kevin\Kevin\Projects\PhD\Testing\' savestr],'jpg');
            %close(fh2);
```

A.4.3 Organize VICON Data for Comparison With In-Vivo Data

```
function VICON_ALL = VICON_Rotation_EF(VICON_ALL,SA,binum)


fn_SA = fieldnames(VICON_ALL)

VICON = VICON_ALL.(fn_SA{SA});
statename = (fn_SA{SA});

% Determine neg_pos = 1, pos_neg = 2
binum = binum;
if binum == 1
   biname = 'neg_pos';
elseif binum == 2
   biname = 'pos_neg';
end
dof = 3; %FE

loop = 2;
rotFE = ['rotFE_AP' num2str(binum)];

statename = {'No HAM', 'No Compression', 'Follower Load', 'Axial Load', 'Combined
Load'};
columnname = {'C34' 'C45' 'C56' 'C67' 'C37'};
ypr = {'yprtr_21_02N', 'yprtr_32_02N','yprtr_43_02N','yprtr_54_02N'};

for state = 1:5

   for SAstep = 1:SA

      for level = 1:4

         % Temporary addition for composite processing
         fn_SA_st = {'FL0_AL0_noHAM_FE_N01' 'FL0_AL0_HAM_FE_N01'
'FL100_AL0_HAM_FE_N01' 'FL0_AL50_HAM_FE_N01' 'FL100_AL50_HAM_FE_N01'};

         ALL =
VICON_ALL.(fn_SA{SAstep}).(fn_SA_st{state}).loop_neg_pos.Transform(loop).anatomical_l
oop.(ypr{level})(:,dof,binum);

         lx=length(ALL);
         x = 1:lx;
         x20 = 1:lx/21:lx;
```

```
                        VICON_ALL.(rotFE).(columnname{level}).(fn_SA_st{state})(:,SAstep) =
interp1(x,ALL,x20);

                        clear ALL

                    end

                end

            end
```

A.5 ANALYZE IN-VIVO DATA

A.5.1 Import In-Vivo Data Into MATLAB

```
            % BDL_Read.m
            % Kevin Bell
            % Updated 5/31/2013

            folder='Z:\Ortho Research 3\FergusonLab\Staff\Bell,
Kevin\Kevin\Projects\PhD\BDL\BDL\';

            clear data
            clear temp
            loopct = 0;
            figure

            for static_flex = 1:2

                if static_flex == 1 % Static Trials

                    for groupnum = 1:2

                        groupname = {'CRad' 'CSpine'};

                        if groupnum == 1
                            group = 'CRad\Kinematics\';
```

```
    flex = '\Day1\static'; % consider looping through and averaging
    ID = {'04' '05' '06' '07' '17' '21' '22' '23' '27' '28'};
    flexnum = 1;
else
    group = 'CSpine\Kinematics\';
    flex = '\Day1\static'; % consider looping through and averaging
    ID = {'13' '14' '23' '28' '29' '31' '32' '33' '35' '37'};
    flexnum = 1;
end

for loop = 1:length(ID)

    groupID = [(groupname{groupnum}) ID{loop}];

    for flexloop = 1:flexnum

        file=[group ID{loop} flex num2str(flexloop)...
            '\KinematicsMeasurementReport.csv'];

        % clear all;
        [data,temp]=importdata([folder file],',', 2);
        [data1,temp1]=importdata([folder file],',', 1);

        CRot = {'C3-C4_Rotation','C4-C5_Rotation',...
            'C5-C6_Rotation','C6-C7_Rotation'};
        CTrans = {'C3-C4_Translation','C4-C5_Translation',...
            'C5-C6_Translation','C6-C7_Translation'};

        for Cloop = 1:4
            % Find rotation column names
            smRot = strmatch(CRot{Cloop}, data1);
            rot_columnnum(Cloop) = smRot(1);
            % Find translation column names
            smTrans = strmatch(CTrans{Cloop}, data1);
            trans_columnnum(Cloop) = smTrans(1);
        end

        BDL.(groupID).static_ALL = data.data;

        % rot_columnnum = [6 9 12 15];
        rot_columnname = {'C34' 'C45' 'C56' 'C67' 'C37'};

        % trans_columnnum = [25 29 33 37];
        trans_columnname = {'C34' 'C45' 'C56' 'C67' 'C37'};

        groupID = [(groupname{groupnum}) ID{loop}];
```

```
                sloop = ['static' num2str(flexloop)];

                for level = 1:length(rot_columnnum)
                    BDL.(groupID).(rot_columnname{level}).(sloop).rot...
                        = data.data(:,rot_columnnum(level):...
                        rot_columnnum(level)+2);   %FE, AR, LB
                    BDL.(groupID).(trans_columnname{level}).(sloop)...
                        .trans = data.data(:,trans_columnnum(level):...
                        trans_columnnum(level)+2);   %FE, AR, LB
                end

            end

            for level = 1:length(rot_columnnum)
                % Rotation
                BDL.(groupID).(rot_columnname{level}).(sloop).rot_mean...
                    = mean(BDL.(groupID).(rot_columnname{level}).(sloop).rot,1);
                BDL.(groupID).(trans_columnname{level}).(sloop).rot_std...
                    = std(BDL.(groupID).(trans_columnname{level}).(sloop).rot,0,1);
                % Translations
                BDL.(groupID).(rot_columnname{level}).(sloop).trans_mean...
                    = mean(BDL.(groupID).(rot_columnname{level}).(sloop).trans,1);
                BDL.(groupID).(trans_columnname{level}).(sloop).trans_std...
                    = std(BDL.(groupID).(trans_columnname{level}).(sloop).trans,0,1);
            end

            clear data
            %clear temp

        end

    end

elseif static_flex == 2 % Flexion - Extension Trials

    for groupnum = 1:2

        groupname = {'CRad' 'CSpine'};

        if groupnum == 1
            group = 'CRad\Kinematics\';
            flex = '\Day1\flex'; % consider looping through and averaging
            ID = {'04' '05' '06' '07' '17' '21' '22' '23' '27' '28'};
            flexnum = 2;
        else
    group = 'CSpine\Kinematics\';
    flex = '\Day1\flexion'; % consider looping through and averaging
    ID = {'13' '14' '23' '28' '29' '31' '32' '33' '35' '37'};
    flexnum = 3;
end

for loop = 1:length(ID)

    loopct = loopct + 1;

    for flexloop = 1:flexnum

        file=[group ID{loop} flex num2str(flexloop)...
            '\KinematicsMeasurementReport.csv'];

        % clear all;
        [data,temp]=importdata([folder file],',', 2);
        [data1,temp1]=importdata([folder file],',', 1);

        CRot = {'C3-C4_Rotation','C4-C5_Rotation'...
            ,'C5-C6_Rotation','C6-C7_Rotation'};
        CTrans = {'C3-C4_Translation','C4-C5_Translation'...
            ,'C5-C6_Translation','C6-C7_Translation'};

        for Cloop = 1:4
            % Find rotation column names
            smRot = strmatch(CRot{Cloop}, data1);
            rot_columnnum(Cloop) = smRot(1);
            % Find translation column names
            smTrans = strmatch(CTrans{Cloop}, data1);
            trans_columnnum(Cloop) = smTrans(1);
        end

        dALL = ['data_ALL' num2str(flexloop)];

        BDL.(groupID).(dALL) = data.data;

        % rot_columnnum = [6 9 12 15];
        rot_columnname = {'C34' 'C45' 'C56' 'C67'};

        % trans_columnnum = [25 29 33 37];
        trans_columnname = {'C34' 'C45' 'C56' 'C67'};

        groupID = [(groupname{groupnum}) ID{loop}];

        floop = ['flex' num2str(flexloop)];
```

```
for level = 1:length(rot_columnnum)

   BDL.(groupID).(rot_columnname{level}).(floop).rot...
      = data.data(:,rot_columnnum(level):rot_columnnum(level)+2);
   BDL.(groupID).(trans_columnname{level}).(floop).trans...
      = data.data(:,trans_columnnum(level):trans_columnnum(level)+2);

   for j = 1:90
      for i = 1:3
         BDL.(groupID).(rot_columnname{level}).(floop)...
            .rot_zero(j,i) = BDL.(groupID)...
            .(rot_columnname{level}).(floop).rot(j,i)...
            -BDL.(groupID).(rot_columnname{level})...
            .(sloop).rot_mean(1,i);
         BDL.(groupID).(rot_columnname{level}).(floop)...
            .trans_zero(j,i) = BDL.(groupID)...
            .(rot_columnname{level}).(floop)...
            .trans(j,i)-BDL.(groupID)...
            .(rot_columnname{level}).(sloop).trans_mean(1,i);
      end
   end

end

rot_columnname = {'C34' 'C45' 'C56' 'C67' 'C37'};

% Combine - GIVING ERROR NEED TO FIX
A1 = BDL.(groupID).(rot_columnname{1}).(floop).rot_zero(:,1);
A2 = BDL.(groupID).(rot_columnname[5]).(floop).rot_zero(:,1);
A3 = BDL.(groupID).(rot_columnname{3}).(floop).rot_zero(:,1);
A4 = BDL.(groupID).(rot_columnname{4}).(floop).rot_zero(:,1);
BDL.(groupID).(rot_columnname{5}).(floop).rot_zero...
   = A1+A2+A3+A4;
[max0, max0loc] = max(BDL.(groupID).(rot_columnname{5})...
   .(floop).rot_zero(1:end,1));
[min0, min0loc] = min(BDL.(groupID).(rot_columnname{5})...
   .(floop).rot_zero(1:end,1));

for level = 1:length(rot_columnnum)

   rotEF = ['rot_EF' num2str(flexloop)];
   transEF = ['trans_EF' num2str(flexloop)];

   if max0loc > min0loc
      BDL.EF(loopct, flexloop) = 1;
```

```
            BDL.(groupID).(rot_columnname{level}).(rotEF)...
                = BDL.(groupID).(rot_columnname{level})...
                .(floop).rot_zero(min0loc:max0loc,:);
            BDL.(groupID).(trans_columnname{level})...
                .(transEF) = BDL.(groupID)...
                .(trans_columnname{level}).(floop)...
                .trans_zero(min0loc:max0loc,:);
        else
            BDL.EF(loopct, flexloop) = 2;
            BDL.(groupID).(rot_columnname{level}).(rotEF)...
                = flipdim(BDL.(groupID)...
                .(rot_columnname{level}).(floop)...
                .rot_zero(max0loc:min0loc,:),1);
            BDL.(groupID).(trans_columnname{level})...
                .(transEF) = flipdim(BDL.(groupID)...
                .(trans_columnname{level}).(floop)...
                .trans_zero(max0loc:min0loc,:),1);
        end

    end

    rotEF_E0 = ['rotEF_E0' num2str(flexloop)];
    transEF_E0 = ['transEF_E0' num2str(flexloop)];

    for level = 1:length(rot_columnnum)

        for E0loop = 1:length(BDL.(groupID)...
                .(rot_columnname{level}).(rotEF))
            BDL.(groupID).(rot_columnname{level})...
                .(rotEF_E0)(E0loop,:) = BDL.(groupID)...
                .(rot_columnname{level}).(rotEF)(E0loop,:)...
                -BDL.(groupID).(rot_columnname{level})...
                .(rotEF)(1,:);
            BDL.(groupID).(rot_columnname{level})...
                .(transEF_E0)(E0loop,:) = BDL.(groupID)...
                .(rot_columnname{level}).(transEF)(E0loop,:)...
                -BDL.(groupID).(rot_columnname{level})...
                .(transEF)(1,:);
        end

    end

end

end
```

```
        end

    end

end
```

A.5.2 Organize In-Vivo Data for Comparison with In-Vitro Data

```
% BDL_Analyze.m
% Kevin Bell
% Updated 5/31/2013

ct = 0;

for groupnum = 1:2

    groupname = {'CRad' 'CSpine'};

    if groupnum == 1
        group = 'CRad\Kinematics\';
        flex = '\Day1\flex'; % can consider looping through and averaging
        ID = {'04' '05' '06' '07' '17' '21' '22' '23' '27' '28'};
        flexnum = 2;
    else
        group = 'CSpine\Kinematics\';
        flex = '\Day1\flexion'; % can consider looping through and averaging
        ID = {'13' '14' '23' '28' '29' '31' '32' '33' '35' '37'};
        flexnum = 3;
    end

    for loop = 1:length(ID)

        ct = ct + 1;

        for flexloop = 1:flexnum

            columnname = {'C34' 'C45' 'C56' 'C67' 'C37'};

            groupID = [(groupname{groupnum}) ID{loop}];

            floop = ['flex' num2str(flexloop)];

            rotEF = ['rot_EF' num2str(flexloop)];
```

```
transEF = ['trans_EF' num2str(flexloop)];

for level = 1:length(rot_columnnum)

    for dof = 1:6
        if dof < 4
            BDL.(groupID).(columnname{level}).Ext(flexloop,dof)...
                = BDL.(groupID).(columnname{level}).(rotEF)(1,dof);
            BDL.(groupID).(columnname{level}).Flex(flexloop,dof)...
                = BDL.(groupID).(columnname{level}).(rotEF)(end,dof);
            BDL.(groupID).(columnname{level}).FE(flexloop,dof)...
                =abs(BDL.(groupID).(columnname{level})...
                .Flex(flexloop,dof))+abs(BDL.(groupID)...
                .(columnname{level}).Ext(flexloop,dof));
        else
            BDL.(groupID).(columnname{level}).Ext(flexloop,dof)...
                = BDL.(groupID).(columnname{level}).(transEF)(1,dof-3);
            BDL.(groupID).(columnname{level}).Flex(flexloop,dof)...
                = BDL.(groupID).(columnname{level}).(transEF)(end,dof-3);
            BDL.(groupID).(columnname{level}).FE(flexloop,dof)...
                =abs(BDL.(groupID).(columnname{level})...
                .Flex(flexloop,dof))+abs(BDL.(groupID)...
                .(columnname{level}).Ext(flexloop,dof));
        end

        BDL.(groupID).(columnname{level}).FE_mean...
            = mean(BDL.(groupID).(columnname{level}).FE,1);
        BDL.(groupID).(columnname{level}).Flex_mean...
            = mean(BDL.(groupID).(columnname{level}).Flex,1);
        BDL.(groupID).(columnname{level}).Ext_mean...
            = mean(BDL.(groupID).(columnname{level}).Ext,1);

    end

    BDL.ALL.(columnname{level}).FE.ALL(ct,:)...
        = BDL.(groupID).(columnname{level}).FE_mean;
    BDL.ALL.(columnname{level}).Flex.ALL(ct,:)...
        = BDL.(groupID).(columnname{level}).Flex_mean;
    BDL.ALL.(columnname{level}).Ext.ALL(ct,:)...
        = BDL.(groupID).(columnname{level}).Ext_mean;

    rotEF_E0 = ['rotEF_E0' num2str(flexloop)];
    transEF_E0 = ['transEF_E0' num2str(flexloop)];

    rotEF_P20 = ['rotEF_P20' num2str(flexloop)];
    transEF_P20 = ['transEF_P20' num2str(flexloop)];
```

```
rotEF20 = ['rotEF20' num2str(flexloop)];
transEF20 = ['transEF20' num2str(flexloop)];

rot_ALL = ['rotEF20' num2str(flexloop)];
trans_ALL = ['transEF20' num2str(flexloop)];

rot_ALL2 = ['rotEF_P20' num2str(flexloop)];
trans_ALL2 = ['transEF_P20' num2str(flexloop)];


% downsample using interp1 to create equal length arrays.
for dof = 1:6
  if dof < 4

    lx=length(BDL.(groupID).(columnname{level})...
      .(rotEF_E0)(:,dof));
    x = 1:lx;
    x20 = 1:lx/21:lx;

    BDL.(groupID).(columnname{level}).(rotEF20)(:,dof)...
      = interp1(x,BDL.(groupID).(columnname{level})...
      .(rotEF)(:,dof),x20);

    %Save into ALL
    BDL.ALL.(columnname{level}).(rot_ALL)(:,ct,dof)...
      = BDL.(groupID).(columnname{level}).(rotEF20)(:,dof);

    clear x lx inttemp

  else

    lx=length(BDL.(groupID).(columnname{level})...
      .(transEF_E0)(:,dof-3));
    x = 1:lx;
    x20 = 1:lx/21:lx;

    BDL.(groupID).(columnname{level}).(transEF20)(:,dof-3)...
      = interp1(x,BDL.(groupID).(columnname{level})...
      .(transEF)(:,dof-3),x20);

    %Save into ALL
    BDL.ALL.(columnname{level}).(trans_ALL)(:,ct,dof-3)...
      = BDL.(groupID).(columnname{level}).(transEF20)(:,dof-3);

    clear x lx
    end
end


% E0 - Extension redefined as zero
% downsample using interp1 to create equal length arrays.
for dof = 1:6
    if dof < 4

        lx=length(BDL.(groupID).(columnname{level}).(rotEF_E0)(:,dof));
        x = 1:lx;
        x20 = 1:lx/21:lx;

        BDL.(groupID).(columnname{level}).(rotEF_P20)(:,dof)...
            = interp1(x,BDL.(groupID).(columnname{level})...
            .(rotEF_E0)(:,dof),x20);

        %Save into ALL
        BDL.ALL.(columnname{level}).(rot_ALL2)(:,ct,dof)...
            = BDL.(groupID).(columnname{level}).(rotEF_P20)(:,dof);

        if BDL.EF(ct,flexloop) == 1
            BDL.ALLEF.(columnname{level}).(rot_ALL2)(:,ct,dof)...
                = BDL.(groupID).(columnname{level}).(rotEF_P20)(:,dof);
        elseif BDL.EF(ct,flexloop) == 2
            BDL.ALLFE.(columnname{level}).(rot_ALL2)(:,ct,dof)...
                =BDL.(groupID).(columnname{level}).(rotEF_P20)(:,dof);
        end

        clear x lx inttemp

    else

        lx=length(BDL.(groupID).(columnname{level})...
            .(transEF_E0)(:,dof-3));
        x = 1:lx;
        x20 = 1:lx/21:lx;

        BDL.(groupID).(columnname{level}).(transEF_P20)(:,dof-3)...
            = interp1(x,BDL.(groupID).(columnname{level})...
            .(transEF_E0)(:,dof-3),x20);

        %Save into ALL
        BDL.ALL.(columnname{level}).(trans_ALL2)(:,ct,dof-3)...
            = BDL.(groupID).(columnname{level}).(transEF_P20)(:,dof-3);
```

```
                if BDL.EF(ct,flexloop) == 1
                    BDL.ALLEF.(columnname{level}).(rot_ALL2)(:,ct,dof)...
                        = BDL.(groupID).(columnname{level}).(rotEF_P20)(:,dof-3);
                elseif BDL.EF(ct,flexloop) == 2
                    BDL.ALLFE.(columnname{level}).(rot_ALL2)(:,ct,dof)...
                        = BDL.(groupID).(columnname{level}).(rotEF_P20)(:,dof-3);
                end

                clear x lx
            end
        end

    end
  end

 end

end

% Find the max FE path
for ct = 1:20

   if BDL.EF(ct,3) == 0

      for level = 1:length(rot_columnnum)

         [max1,maxloc1] = max([BDL.ALL.(columnname{level})...
            .rotEF201(end,ct,1), BDL.ALL.(columnname{level}).rotEF202(end,ct,1)]);

         rot_ALL = ['rotEF20' num2str(maxloc1)];
         trans_ALL = ['transEF20' num2str(maxloc1)];

         BDL.ALL.(columnname{level}).rotEF_max(:,ct,:)...
            = BDL.ALL.(columnname{level}).(rot_ALL)(:,ct,:);
         BDL.ALL.(columnname{level}).transEF_max(:,ct,:)...
            = BDL.ALL.(columnname{level}).(trans_ALL)(:,ct,:);

      end

   else

      for level = 1:length(rot_columnnum)

         [max2,maxloc2] = max([BDL.ALL.(columnname{level})...
            .rotEF201(end,ct,1),BDL.ALL.(columnname{level})...
            .rotEF202(end,ct,1),BDL.ALL.(columnname{level})...
            .rotEF203(end,ct,1)]);

          rot_ALL = ['rotEF20' num2str(maxloc2)];
          trans_ALL = ['transEF20' num2str(maxloc2)];

          BDL.ALL.(columnname{level}).rotEF_max(:,ct,:)...
            = BDL.ALL.(columnname{level}).(rot_ALL)(:,ct,:);
          BDL.ALL.(columnname{level}).transEF_max(:,ct,:)...
            = BDL.ALL.(columnname{level}).(trans_ALL)(:,ct,:);

        end

    end

end

% Find the max mean path
for ct = 1:20

    if BDL.EF(ct,3) == 0

        for level = 1:length(rot_columnnum)

          for dof = 1:3
            BDL.ALL.(columnname{level}).rotEF_mean(:,ct,dof)...
              = mean([BDL.ALL.(columnname{level}).rotEF201(:,ct,dof)...
              , BDL.ALL.(columnname{level}).rotEF202(:,ct,dof)],2);
            BDL.ALL.(columnname{level}).transEF_mean(:,ct,dof)...
              = mean([BDL.ALL.(columnname{level}).transEF201(:,ct,dof)...
              , BDL.ALL.(columnname{level}).transEF202(:,ct,dof)],2);
          end

        end

    else

        for level = 1:length(rot_columnnum)

          for dof = 1:3
            BDL.ALL.(columnname{level}).rotEF_mean(:,ct,dof)...
              = mean([BDL.ALL.(columnname{level}).rotEF201(:,ct,dof)...
              , BDL.ALL.(columnname{level}).rotEF202(:,ct,dof)...
              , BDL.ALL.(columnname{level}).rotEF203(:,ct,dof)],2);
            BDL.ALL.(columnname{level}).transEF_mean(:,ct,dof)...
              = mean([BDL.ALL.(columnname{level}).transEF201(:,ct,dof)...
              , BDL.ALL.(columnname{level}).transEF202(:,ct,dof)...
```

```
                    , BDL.ALL.(columnname{level}).transEF203(:,ct,2)],2);
            end

        end

    end

end
```

A.5.3 Interpolate In-Vivo Data to Align Step Size with In-Vitro Data

```
% BDL_Analyze_Interp.m
% Kevin Bell
% Updated 5/31/2013

% clear C34 C45 C56 C67 C37
clear C34_interp C45_interp C56_interp C67_interp C37_interp
clear C34_ef C45_ef C56_ef C67_ef C37_ef
clear realx37

C34_ef = BDL.ALL.C34.rotEF202(:,:,1)';
C45_ef = BDL.ALL.C45.rotEF202(:,:,1)';
C56_ef = BDL.ALL.C56.rotEF202(:,:,1)';
C67_ef = BDL.ALL.C67.rotEF202(:,:,1)';

for i = 1:size(C34_ef,1)
    for j = 1:size(C34_ef,2)
        C34(i,j) = C34_ef(i,j)-C34_ef(i,1);
        C45(i,j) = C45_ef(i,j)-C45_ef(i,1);
        C56(i,j) = C56_ef(i,j)-C56_ef(i,1);
        C67(i,j) = C67_ef(i,j)-C67_ef(i,1);

    end

    C37(i,:) = C34(i,:) + C45(i,:) + C56(i,:) + C67(i,:);
end

for i = 1:size(C34,1)
    C37(i,:) = C34(i,:) + C45(i,:) + C56(i,:) + C67(i,:);
end
C37_AVG = mean(C37(:,:), 1);
```

```
% Interpolation - E=0
x = [0 5 20 40 60 80 100];
% x = [0 5 10 15 20 25 30 35 40 45];
%stop = size(C34,2);

for i = 1:size(C34,1)

    realx37 = (C37(i,1:end)/C37(i,end))*100;
    %realx37 = C37(i,1:end);

    C34_interp(i,:) = interp1(realx37,C34(i,1:end),x);

    C45_interp(i,:) = interp1(realx37,C45(i,1:end),x);

    C56_interp(i,:) = interp1(realx37,C56(i,1:end),x);

    C67_interp(i,:) = interp1(realx37,C67(i,1:end),x);

    C37_interp(i,:) = interp1(realx37,C37(i,1:end),x);

end
```

BIBLIOGRAPHY

1. Galbusera, F., et al., *Biomechanical studies on cervical total disc arthroplasty: a literature review.* Clin Biomech (Bristol, Avon), 2008. **23**(9): p. 1095-104.

2. Puttlitz, C.M., et al., *Intervertebral disc replacement maintains cervical spine kinetics.* Spine (Phila Pa 1976), 2004. **29**(24): p. 2809-14.

3. Panjabi, M.M., *Biomechanical evaluation of spinal fixation devices: I. A conceptual framework.* Spine (Phila Pa 1976), 1988. **13**(10): p. 1129-34.

4. Panjabi, M.M., *Hybrid multidirectional test method to evaluate spinal adjacent-level effects.* Clin Biomech (Bristol, Avon), 2007. **22**(3): p. 257-65.

5. Edwards, C.C., 2nd, et al., *Cervical myelopathy. current diagnostic and treatment strategies.* Spine J, 2003. **3**(1): p. 68-81.

6. Geisler, F.H., et al., *Reoperation in patients after anterior cervical plate stabilization in degenerative disease.* Spine (Phila Pa 1976), 1998. **23**(8): p. 911-20.

7. Hilibrand, A.S., et al., *Radiculopathy and myelopathy at segments adjacent to the site of a previous anterior cervical arthrodesis.* J Bone Joint Surg Am, 1999. **81**(4): p. 519-28.

8. Hilibrand, A.S. and M. Robbins, *Adjacent segment degeneration and adjacent segment disease: the consequences of spinal fusion?* Spine J, 2004. **4**(6 Suppl): p. 190S-194S.

9. Robertson, J.T., S.M. Papadopoulos, and V.C. Traynelis, *Assessment of adjacent-segment disease in patients treated with cervical fusion or arthroplasty: a prospective 2-year study.* J Neurosurg Spine, 2005. **3**(6): p. 417-23.

10. Wigfield, C., et al., *Influence of an artificial cervical joint compared with fusion on adjacent-level motion in the treatment of degenerative cervical disc disease.* J Neurosurg, 2002. **96**(1 Suppl): p. 17-21.

11. Eck, J.C., et al., *Biomechanical study on the effect of cervical spine fusion on adjacent-level intradiscal pressure and segmental motion.* Spine (Phila Pa 1976), 2002. **27**(22): p. 2431-4.

12. Matsunaga, S., et al., *Strain on intervertebral discs after anterior cervical decompression and fusion.* Spine (Phila Pa 1976), 1999. **24**(7): p. 670-5.

13. Bartels, R.H., et al., *Comparison of biomechanical properties of cervical artificial disc prosthesis: a review.* Clin Neurol Neurosurg, 2008. **110**(10): p. 963-7.

14. Phillips, F.M. and S.R. Garfin, *Cervical disc replacement.* Spine (Phila Pa 1976), 2005. **30**(17 Suppl): p. S27-33.

15. DiAngelo, D.J. and K.T. Foley, *An improved biomechanical testing protocol for evaluating spinal arthroplasty and motion preservation devices in a multilevel human cadaveric cervical model.* Neurosurg Focus, 2004. **17**(3): p. E4.

16. Patwardhan, A.G., et al., *Load-carrying capacity of the human cervical spine in compression is increased under a follower load.* Spine (Phila Pa 1976), 2000. **25**(12): p. 1548-54.

17. Adams, M.A. and P. Dolan, *Spine biomechanics.* J Biomech, 2005. **38**(10): p. 1972-83.

18. Cook, D.J. and University of Pittsburgh. School of Engineering, *Characterization of the Response of the Cadaveric Human Spine to Loading in a Six-Degree-of-Freedom Spine Testing Apparatus*. 2009, University of Pittsburgh: Pittsburgh, PA.

19. Cripton, P.A., et al., *In vitro axial preload application during spine flexibility testing: towards reduced apparatus-related artefacts.* J Biomech, 2000. **33**(12): p. 1559-68.

20. Goel, V.K., et al., *Test protocols for evaluation of spinal implants.* J Bone Joint Surg Am, 2006. **88 Suppl 2**: p. 103-9.

21. Miura, T., M.M. Panjabi, and P.A. Cripton, *A method to simulate in vivo cervical spine kinematics using in vitro compressive preload.* Spine (Phila Pa 1976), 2002. **27**(1): p. 43-8.

22. Panjabi, M.M., et al., *Development of a system for in vitro neck muscle force replication in whole cervical spine experiments.* Spine (Phila Pa 1976), 2001. **26**(20): p. 2214-9.

23. Wilke, H.J., et al., *A universal spine tester for in vitro experiments with muscle force simulation.* Eur Spine J, 1994. **3**(2): p. 91-7.

24. Wilke, H.J., et al., *Is it possible to simulate physiologic loading conditions by applying pure moments? A comparison of in vivo and in vitro load components in an internal fixator.* Spine (Phila Pa 1976), 2001. **26**(6): p. 636-42.

25. Wilke, H.J., K. Wenger, and L. Claes, *Testing criteria for spinal implants: recommendations for the standardization of in vitro stability testing of spinal implants.* Eur Spine J, 1998. **7**(2): p. 148-54.

26. Pospiech, J., et al., *Intradiscal pressure recordings in the cervical spine.* Neurosurgery, 1999. **44**(2): p. 379-84; discussion 384-5.

27. Steffen, T., et al., *A new technique for measuring lumbar segmental motion in vivo. Method, accuracy, and preliminary results.* Spine, 1997. **22**(2): p. 156-66.

28. Plamondon, A., M. Gagnon, and G. Maurais, *Application of a stereoradiographic method for the study of intervertebral motion.* Spine, 1988. **13**(9): p. 1027-32.

29. Pearcy, M.J. and M.W. Whittle, *Movements of the lumbar spine measured by three-dimensional X-ray analysis.* J Biomed Eng, 1982. **4**(2): p. 107-12.

30. Anderst, W.J., R. Vaidya, and S. Tashman, *A technique to measure three-dimensional in vivo rotation of fused and adjacent lumbar vertebrae.* Spine J, 2008. **8**(6): p. 991-7.

31. Anderst, W.J., et al., *Validation of a noninvasive technique to precisely measure in vivo three-dimensional cervical spine movement.* Spine (Phila Pa 1976), 2011. **36**(6): p. E393-400.

32. Bell, K.M., et al., *Investigation of the Integrated Passive, Active, and Control Subsystems of the Spine Utilizing a Robotics-Based Spine Testing System*, in *53th Annual Meeting of the Orthopaedic Research Society (ORS)*. 2007: San Diego, CA.

33. Gilbertson, L.G., et al., *Improvement of accuracy in a high-capacity, six degree-of-freedom load cell: Application to robotic testing of musculoskeletal joints.* Annals of Biomedical Engineering, 1999. **27**(6): p. 839-843.

34. Bell, K.M., et al., *In vitro spine testing using a robot-based testing system: Comparison of displacement control and "hybrid control"*. J Biomech, 2013.

35. Panjabi, M.M., *Biomechanical Evaluation of Spinal Fixation Devices .1. A Conceptual-Framework.* Spine, 1988. **13**(10): p. 1129-1134.

36. Goel, V.K., et al., *Biomechanical testing of the spine. Load-controlled versus displacement-controlled analysis.* Spine (Phila Pa 1976), 1995. **20**(21): p. 2354-7.

37. Adams, M.A. and W.C. Hutton, *The relevance of torsion to the mechanical derangement of the lumbar spine.* Spine (Phila Pa 1976), 1981. **6**(3): p. 241-8.

38. Goodwin, R.R., et al., *Distraction and compression loads enhance spine torsional stiffness.* J Biomech, 1994. **27**(8): p. 1049-57.

39. Panjabi, M.M., R.A. Brand, Jr., and A.A. White, 3rd, *Mechanical properties of the human thoracic spine as shown by three-dimensional load-displacement curves.* J Bone Joint Surg Am, 1976. **58**(5): p. 642-52.

40. Panjabi, M.M., R.A. Brand, Jr., and A.A. White, 3rd, *Three-dimensional flexibility and stiffness properties of the human thoracic spine.* J Biomech, 1976. **9**(4): p. 185-92.

41. Edwards, W.T., et al., *Variation of lumbar spine stiffness with load.* J Biomech Eng, 1987. **109**(1): p. 35-42.

42. Grassmann, S., et al., *Constrained testing conditions affect the axial rotation response of lumbar functional spinal units.* Spine (Phila Pa 1976), 1998. **23**(10): p. 1155-62.

43. Kunz, D.N., et al., *A multi-degree of freedom system for biomechanical testing.* J Biomech Eng, 1994. **116**(3): p. 371-3.

44. Gedet, P., P.A. Thistlethwaite, and S.J. Ferguson, *Minimizing errors during in vitro testing of multisegmental spine specimens: considerations for component selection and kinematic measurement.* J Biomech, 2007. **40**(8): p. 1881-5.

45. Wheeler, D.J., et al., *Inter-laboratory variability in in vitro spinal segment flexibility testing.* J Biomech, 2011. **44**(13): p. 2383-7.

46. Raibert, M.H. and J.J. Craig, *Hybrid Position-Force Control of Manipulators.* Journal of Dynamic Systems Measurement and Control-Transactions of the Asme, 1981. **103**(2): p. 126-133.

47. Carlin, G.J., et al., *In-situ forces in the human posterior cruciate ligament in response to posterior tibial loading.* Annals of Biomedical Engineering, 1996. **24**(2): p. 193-197.

48. Fujie, H., et al., *The Use of Robotics Technology to Study Human Joint Kinematics - a New Methodology.* Journal of Biomechanical Engineering-Transactions of the Asme, 1993. **115**(3): p. 211-217.

49. Livesay, G.A., et al., *Determination of the in-Situ Forces and Force Distribution within the Human Anterior Cruciate Ligament.* Annals of Biomedical Engineering, 1995. **23**(4): p. 467-474.

50. Rudy, T.W., et al., *A combined robotic/universal force sensor approach to determine in situ forces of knee ligaments.* Journal of Biomechanics, 1996. **29**(10): p. 1357-1360.

51. Bogduk, N. and S. Mercer, *Biomechanics of the cervical spine. I: Normal kinematics.* Clin Biomech (Bristol, Avon), 2000. **15**(9): p. 633-48.

52. Smit, T.H., et al., *Quantifying intervertebral disc mechanics: a new definition of the neutral zone.* BMC Musculoskelet Disord, 2011. **12**: p. 38.

53. Schmidt, H., et al., *The relation between the instantaneous center of rotation and facet joint forces - A finite element analysis.* Clinical Biomechanics, 2008. **23**(3): p. 270-278.

54. Kawchuk, G.N., et al., *Identification of spinal tissues loaded by manual therapy: a robot-based serial dissection technique applied in porcine motion segments.* Spine (Phila Pa 1976), 2010. **35**(22): p. 1983-90.

55. Hartman, R.A., K.M. Bell, and J.D. Kang, *Analyses of the Components of the Posterior Column in a Distractive-Flexion Injury Model*, in *55th Annual Meeting of the Orthopaedic Research Society (ORS)*. 2009: Las Vegas, NV.

56. Gillespie, K.A. and J.P. Dickey, *Biomechanical role of lumbar spine ligaments in flexion and extension: determination using a parallel linkage robot and a porcine model.* Spine (Phila Pa 1976), 2004. **29**(11): p. 1208-16.

57. Patwardhan, A.G., et al., *A follower load increases the load-carrying capacity of the lumbar spine in compression.* Spine (Phila Pa 1976), 1999. **24**(10): p. 1003-9.

58. Panjabi, M.M., et al., *Critical load of the human cervical spine: an in vitro experimental study.* Clin Biomech (Bristol, Avon), 1998. **13**(1): p. 11-17.

59. Ng, H.W. and E.C. Teo, *Influence of preload magnitudes and orientation angles on the cervical biomechanics: a finite element study.* J Spinal Disord Tech, 2005. **18**(1): p. 72-9.

60. Patwardhan, A.G., K.P. Meade, and B. Lee, *A frontal plane model of the lumbar spine subjected to a follower load: implications for the role of muscles.* J Biomech Eng, 2001. **123**(3): p. 212-7.

61. Janevic, J., J.A. Ashton-Miller, and A.B. Schultz, *Large compressive preloads decrease lumbar motion segment flexibility.* J Orthop Res, 1991. **9**(2): p. 228-36.

62. Panjabi, M.M., et al., *Effects of preload on load displacement curves of the lumbar spine.* Orthop Clin North Am, 1977. **8**(1): p. 181-92.

63. Patwardhan, A.G., et al., *Effect of compressive follower preload on the flexion-extension response of the human lumbar spine.* J Orthop Res, 2003. **21**(3): p. 540-6.

64. Paxinos, O., et al., *Anterior cervical discectomy and fusion with a locked plate and wedged graft effectively stabilizes flexion-distraction stage-3 injury in the lower cervical spine: a biomechanical study.* Spine (Phila Pa 1976), 2009. **34**(1): p. E9-15.

65. Cho, B.Y., et al., *Biomechanical analysis of the range of motion after placement of a two-level cervical ProDisc-C versus hybrid construct.* Spine (Phila Pa 1976), 2010. **35**(19): p. 1769-76.

66. Finn, M.A., et al., *Local and global subaxial cervical spine biomechanics after single-level fusion or cervical arthroplasty.* European Spine Journal, 2009. **18**(10): p. 1520-7.

67. Finn, M.A., et al., *Two-level noncontiguous versus three-level anterior cervical discectomy and fusion: a biomechanical comparison.* Spine (Phila Pa 1976), 2011. **36**(6): p. 448-53.

68. Lee, S.H., et al., *Comparison of cervical spine biomechanics after fixed- and mobile-core artificial disc replacement: a finite element analysis.* Spine (Phila Pa 1976), 2011. **36**(9): p. 700-8.

69. Martin, S., et al., *Kinematics of cervical total disc replacement adjacent to a two-level, straight versus lordotic fusion.* Spine (Phila Pa 1976), 2011. **36**(17): p. 1359-66.

70. Snyder, J.T., et al., *Effect of uncovertebral joint excision on the motion response of the cervical spine after total disc replacement.* Spine (Phila Pa 1976), 2007. **32**(26): p. 2965-9.

71. Tawackoli, W., R. Marco, and M.A. Liebschner, *The effect of compressive axial preload on the flexibility of the thoracolumbar spine.* Spine (Phila Pa 1976), 2004. **29**(9): p. 988-93.

72. Dreischarf, M., et al., *Optimised loads for the simulation of axial rotation in the lumbar spine.* J Biomech, 2011. **44**(12): p. 2323-7.

73. Dreischarf, M., et al., *Optimised in vitro applicable loads for the simulation of lateral bending in the lumbar spine.* Med Eng Phys, 2012. **34**(6): p. 777-80.

74. Kim, K., Y.H. Kim, and S. Lee, *Investigation of optimal follower load path generated by trunk muscle coordination.* J Biomech, 2011. **44**(8): p. 1614-7.

75. Panjabi, M.M., et al., *Mechanical properties of the human cervical spine as shown by three-dimensional load-displacement curves.* Spine (Phila Pa 1976), 2001. **26**(24): p. 2692-700.

76. Thompson, R.E., T.M. Barker, and M.J. Pearcy, *Defining the Neutral Zone of sheep intervertebral joints during dynamic motions: an in vitro study.* Clin Biomech (Bristol, Avon), 2003. **18**(2): p. 89-98.

77. Wilke, H.J., K. Wenger, and L. Claes, *Testing criteria for spinal implants: recommendations for the standardization of in vitro stability testing of spinal implants.* European Spine Journal, 1998. **7**(2): p. 148-54.

78. Sarver, J.J. and D.M. Elliott, *Mechanical differences between lumbar and tail discs in the mouse.* J Orthop Res, 2005. **23**(1): p. 150-5.

79. Panjabi, M.M., *The stabilizing system of the spine. Part II. Neutral zone and instability hypothesis.* J Spinal Disord, 1992. **5**(4): p. 390-6; discussion 397.

80. Gardner-Morse, M.G. and I.A. Stokes, *Physiological axial compressive preloads increase motion segment stiffness, linearity and hysteresis in all six degrees of freedom for small displacements about the neutral posture.* J Orthop Res, 2003. **21**(3): p. 547-52.

81. Wilke, H.J., et al., *Stability increase of the lumbar spine with different muscle groups. A biomechanical in vitro study.* Spine (Phila Pa 1976), 1995. **20**(2): p. 192-8.

82. Papp, T., et al., *An in vitro study of the biomechanical effects of flexible stabilization on the lumbar spine.* Spine (Phila Pa 1976), 1997. **22**(2): p. 151-5.

83. Nachemson, A., *The Influence of Spinal Movements on the Lumbar Intradiscal Pressure and on the Tensil Stresses in the Annulus Fibrosus.* Acta Orthopaedica Scandinavica, 1963. **33**: p. 183-207.

84. Nachemson, A. and J.M. Morris, *Invivo Measurements of Intradiscal Pressure - Discometry, a Method for the Determination of Pressure in the Lower Lumbar Discs.* Journal of Bone and Joint Surgery-American Volume, 1964. **46**(5): p. 1077-1092.

85. Nachemson, A., *The Effect of Forward Leaning on Lumbar Intradiscal Pressure.* Acta Orthopaedica Scandinavica, 1965. **35**: p. 314-28.

86. Nachemson, A.L., *Disc pressure measurements.* Spine (Phila Pa 1976), 1981. **6**(1): p. 93-7.

87. Wilke, H.J., et al., *New in vivo measurements of pressures in the intervertebral disc in daily life.* Spine (Phila Pa 1976), 1999. **24**(8): p. 755-62.

88. Hattori, S., H. Oda, and S. Kawai, *Cervical Intra-Discal Pressure in Movements and Traction of the Cervical-Spine.* Zeitschrift Fur Orthopadie Und Ihre Grenzgebiete, 1981. **119**(6): p. 568-569.

89. Pospiech, J., et al., *In vitro measurements of cervical intra-discal pressure in different situations.* Langenbecks Archiv Fur Chirurgie, 1996. **381**(6): p. 303-308.

90. Cripton, P.A., et al., *In vitro axial preload application during spine flexibility testing: towards reduced apparatus-related artefacts.* Journal of Biomechanics, 2000. **33**(12): p. 1559-68.

91. An, H.S., et al., *Intradiscal administration of osteogenic protein-1 increases intervertebral disc height and proteoglycan content in the nucleus pulposus in normal adolescent rabbits.* Spine (Phila Pa 1976), 2005. **30**(1): p. 25-31; discussion 31-2.

92. Jacobs, L.J., N. Vo, and J.D. Kang, *Identifying inflammatory targets for biologic therapies for spine pain.* PM R, 2011. **3**(6 Suppl 1): p. S12-7.

93. Lebow, R.L., et al., *Asymptomatic same-site recurrent disc herniation after lumbar discectomy: results of a prospective longitudinal study with 2-year serial imaging.* Spine (Phila Pa 1976), 2011. **36**(25): p. 2147-51.

94. Mariconda, M., et al., *Frequency and clinical meaning of long-term degenerative changes after lumbar discectomy visualized on imaging tests.* European Spine Journal, 2010. **19**(1): p. 136-43.

95. Masuda, K., et al., *A novel rabbit model of mild, reproducible disc degeneration by an anulus needle puncture: correlation between the degree of disc injury and radiological and histological appearances of disc degeneration.* Spine (Phila Pa 1976), 2005. **30**(1): p. 5-14.

96. Cripton, P.A., G.A. Dumas, and L.P. Nolte, *A minimally disruptive technique for measuring intervertebral disc pressure in vitro: application to the cervical spine.* Journal of Biomechanics, 2001. **34**(4): p. 545-549.

97. Wu, L.P., et al., *Influence of cervical spine position, turning time, and cervical segment on cadaver intradiscal pressure during cervical spinal manipulative therapy.* J Manipulative Physiol Ther, 2012. **35**(6): p. 428-36.

98. Skrzypiec, D.M., et al., *The internal mechanical properties of cervical intervertebral discs as revealed by stress profilometry.* European Spine Journal, 2007. **16**(10): p. 1701-9.

99. Wigfield, C.C., et al., *Internal stress distribution in cervical intervertebral discs: the influence of an artificial cervical joint and simulated anterior interbody fusion.* J Spinal Disord Tech, 2003. **16**(5): p. 441-9.

100. Anderst, W.J., et al., *Cervical Motion Segment Percent Contributions to Flexion-Extension During Continuous Functional Movement in Control Subjects and Arthrodesis Patients.* Spine (Phila Pa 1976), 2013.

101. Anderst, W.J., et al., *Cervical spine intervertebral kinematics with respect to the head are different during flexion and extension motions.* J Biomech, 2013. **46**(8): p. 1471-5.

102. Dvorak, J., et al., *Functional radiographic diagnosis of the cervical spine: flexion/extension.* Spine (Phila Pa 1976), 1988. **13**(7): p. 748-55.

103. Frobin, W., et al., *Sagittal plane segmental motion of the cervical spine. A new precision measurement protocol and normal motion data of healthy adults.* Clin Biomech (Bristol, Avon), 2002. **17**(1): p. 21-31.

104. Reitman, C.A., et al., *Intervertebral motion between flexion and extension in asymptomatic individuals.* Spine (Phila Pa 1976), 2004. **29**(24): p. 2832-43.

105. Wu, S.K., et al., *The quantitative measurements of the intervertebral angulation and translation during cervical flexion and extension.* Eur Spine J, 2007. **16**(9): p. 1435-44.

106. Wu, S.K., et al., *Segmental percentage contributions of cervical spine during different motion ranges of flexion and extension.* J Spinal Disord Tech, 2010. **23**(4): p. 278-84.

107. Anderson, M.J. and P.J. Whitcomb, *DOE simplified : practical tools for effective experimentation*. 2000, Portland, Or.: Productivity. xiii, 236 p.

108. National Institute of Standards and Technology (U.S.) and International SEMATECH., *Engineering statistics handbook*. NIST.: Gaithersburg, Md.

109. Derringer, G. and R. Suich, *Simultaneous-Optimization of Several Response Variables.* Journal of Quality Technology, 1980. **12**(4): p. 214-219.

110. Anderson, M.J. and P.J. Whitcomb, *DOE simplified : practical tools for effective experimentation*. 2nd ed. 2007, New York, N.Y.: Productivity Press. xiii, 241 p.

111. Cheng, C.H., K.H. Lin, and J.L. Wang, *Co-contraction of cervical muscles during sagittal and coronal neck motions at different movement speeds.* Eur J Appl Physiol, 2008. **103**(6): p. 647-54.

112. Johnston, V., et al., *Neck movement and muscle activity characteristics in female office workers with neck pain.* Spine (Phila Pa 1976), 2008. **33**(5): p. 555-63.

113. Schuldt, K., *On neck muscle activity and load reduction in sitting postures. An electromyographic and biomechanical study with applications in ergonomics and rehabilitation.* Scand J Rehabil Med Suppl, 1988. **19**: p. 1-49.

114. Hussain, M., et al., *Patterns of height changes in anterior and posterior cervical disc regions affects the contact loading at posterior facets during moderate and severe disc degeneration: a poroelastic C5-C6 finite element model study.* Spine (Phila Pa 1976), 2010. **35**(18): p. E873-81.

115. Anderst, W.J., et al., *Subject-specific inverse dynamics of the head and cervical spine during in vivo dynamic flexion-extension.* J Biomech Eng, 2013. **135**(6): p. 61007-8.

116. White, A.A. and M.M. Panjabi, *Clinical biomechanics of the spine*. 2nd ed. 1990, Philadelphia: Lippincott. xxiii, 722 p.

117. Adams, M.A., *The biomechanics of back pain*. 3rd ed. 2012, Edinburgh: Elsevier. p.

118. Ahn, H.S. and D.J. DiAngelo, *Biomechanical testing simulation of a cadaver spine specimen: development and evaluation study.* Spine (Phila Pa 1976), 2007. **32**(11): p. E330-6.

119. de Jongh, C.U., A.H. Basson, and C. Scheffer, *Dynamic simulation of cervical spine following single-level cervical disc replacement.* Conf Proc IEEE Eng Med Biol Soc, 2007. **2007**: p. 4289-92.

120. Marin, F., et al., *In vivo intersegmental motion of the cervical spine using an inverse kinematics procedure.* Clin Biomech (Bristol, Avon), 2010. **25**(5): p. 389-96.

121. Bradtmoller, M., et al., *Impaired Pten expression in human malignant peripheral nerve sheath tumours.* PLoS One, 2012. **7**(11): p. e47595.