Modeling of the human trunk: the impact of stiffness gain and lumbar lordosis on stability

By

Valerie Jardon

Submitted to the University of Kansas' Bioengineering graduate program and faculty in partial fulfillment of the Master of Science in Bioengineering degree

_____

Sara E. Wilson, Ph.D, Chair

_____

Elizabeth A. Friis, Ph.D

_____

Carl W. Luchies, Ph.D

Date Defended: May 4, 2022

The thesis committee chair for Valerie Jardon certifies that this is the approved version of the following thesis:

Modeling of the human trunk: the impact of stiffness gain and lumbar lordosis on stability

_____

Sara E. Wilson, Ph.D, Chair

Date Approved: May 13, 2022

Abstract

Low back pain and injuries are prevalent and costly musculoskeletal conditions in our society, afflicting most Americans at some point throughout their lifetime. In an effort to develop effective treatment and rehabilitation methods, researchers have continued in their investigation of the potential risk factors and causes of low back pain through use of spine models and experimental data collection.

The focus of this work is the development and utilization of an 18 degree-of-freedom stability-based trunk model with 90 muscle fascicles, reflexes, and a lumped parameter intervertebral disc model to explore the potential impacts stiffness gain magnitude and lumbar lordosis angle have on the model's predictions for stability. Throughout the development of this model, a thorough verification procedure was utilized to minimize errors.

It was determined that as stiffness gain in the short-range muscle stiffness model increased from 0.9 to 40, the magnitude of the required metabolic power and muscle force to stabilize the system decreased. Above stiffness gains with a magnitude of 20, increasing the stiffness gains had little impact on the required metabolic power and muscle force. Additionally, for a stiffness gain of 0.5, the model predicted instability whereas all stiffness gains greater than or equal to 0.9 resulted in a stabilized system. It is evident that stiffness gain has the ability to influence model predictions, including the required metabolic power and muscle force.

In our investigation with lumbar lordosis, it was determined that the hyperlordosis case required more metabolic power and additional recruitment of the flexor muscles in order to stabilize in comparison to the other lordosis cases. The hypolordosis cases required less metabolic power and additional recruitment of extensor muscles to stabilize in comparison to the other lordosis cases. The additional recruitment of the flexor muscles needed to stabilize the hyperlordotic spine and the

additional recruitment of the extensor muscles needed to stabilize the hypolordotic spine can be explained by the line of gravity being positioned posteriorly and anteriorly for the lordosis cases respectively.

Future investigations should further explore the impact of lumbar lordosis angle on spine stability, especially experimentally and with EMG-based models, and should consider investigating the impacts of varying stiffness gains on spinal stability. Lastly, additional loading tasks should be simulated with the model, such as asymmetric tasks, differing load magnitudes and differing load application points.

Acknowledgements

I would like to extend my appreciation and thanks to all that have supported me and helped me along the way throughout this research project and my graduate education.

First, I would like to thank Dr. Wilson for her continuous guidance, support, and patience – none of this would have been possible without her. I cannot thank Dr. Wilson enough for her valuable insight and advice, all the knowledge she has shared with me, and her willingness and dedication to support her students in our research, academic and professional endeavors. Additionally, I would like to thank my committee members Dr. Friis and Dr. Luchies for all of their support and the knowledge they have shared with me throughout my undergraduate and graduate career. I truly couldn't have accomplished this without them.

I would also like to thank Mukui Mutunga for her impactful role as my research mentor throughout my undergraduate career. I am so thankful to have had a mentor so willing to share her passion, kindness, insight, and knowledge. I am thankful that you are still a part of my research journey today. Additionally, I would like to thank Tong Chu for her continuous encouragement and contributions, specifically her dedication, diligence and assistance in literature reviewing. I am thankful to have been a part of her research journey and grateful to have been her mentor!

I would like to thank my fiancé Andrew Yancey for his continuous support and encouragement throughout this research project. During the pandemic, he truly was my acting lab mate, and I am so grateful to have had him working alongside me. I am thankful for his guidance, his willingness to listen, and for his continuous encouragement throughout this journey. Additionally, I would like to thank my family for all they have done to support me and my passions. I am so thankful to have so many wonderful people encouraging me, supporting me, and inspiring me.

Lastly, I would like to thank the faculty of the KU Bioengineering department, Mechanical Engineering department, and the School of Music for all their support, instruction and for opportunities I have been granted. I am truly blessed to have such a wonderful community at KU.

Table of Contents

Chapter 1: Introduction 1

1.1 What causes low back pain and injury? 1

1.1.1 Prevalence of Low Back Pain 1

1.1.2 Modeling of the Human Trunk 1

1.2 Background: Stability-based Spine Models and Utilization 3

1.2.1 The importance of spine modeling in the investigation of low back pain 3

1.2.2 The importance of spine stability analysis in the investigation of low back pain 4

1.2.3 Stability-based Trunk Models in Literature 5

1.2.4 Summary 13

1.3 Model Development: Based on the Work of Franklin *et al.* 15

1.3.1 Model Summary 15

1.4 Current Investigation 16

1.4.1 Bergmark's Model of Short-range Stiffness 16

1.4.2 Hyperlordosis and Hypolordosis 17

1.5 Specific Aims and Hypotheses 18

Chapter 2: Model Development 20

2.1 Rigid Bodies 20

2.1.1 Dimensions 22

2.1.2 Body Origins 25

2.1.3 Mass and Mass Moment of Inertia 26

2.2 Rotations 27

2.2.1 Lordosis Angles 27

2.3 Intervertebral Discs 28

2.4 Muscle Anatomy 30

2.4.1 Calculating Transformed Muscle Anatomy 30

2.4.2 Calculating Muscle Length 31

2.4.3 Calculating Muscle Velocity 32

2.5 Muscle Model 33
2.6 Reflex Model 35
2.7 Dynamics 36
2.8 Linearization 38
2.9 Optimization Procedure 39
2.10 Linear Time Delay Methods 41
2.11 Nonlinear Verification 42
2.12 Summary 43
Chapter 3: Code Verification 44
3.1 Model Parameters 49
3.2 Initial Transformation of Cholewicki *et al.* Muscle and Skeletal Coordinates 50
3.2.1 Muscle and Skeletal Coordinates Verification 50
3.2.2 Muscle Path Verification 54
3.3 Import Muscle Geometry and Skeletal Coordinates 55
3.4 Import External Force 57
3.5 Rotation Matrices 58
3.6 Body Origins 60
3.7 Transform Muscle Geometry and Skeletal Coordinates 61
3.8 Muscle Properties 62
3.9 Transform External Force 63
3.10 Dynamics 64
3.11 Intervertebral Discs Moments 67
3.12 Generalized Forces 69
3.13 Optimization 70
3.13.1 Constraint Variables 70
3.13.2 Jacobians 73
3.13.3 Cost Function 75
3.14 Muscle Model 76
3.15 Reflex Model 76
3.16 Time Delay Analysis and Nonlinear Verification 76
3.17 Summary 80

Chapter 4: The impact of stiffness gain on stability 81
Chapter 5: The effects of hyperlordosis and hydolordosis on spine stability 104
Chapter 6: Conclusions and Future Work 138
6.1 Model Verification Procedure 138
6.2 Stiffness Gain and Spine Stability 139
6.3 Lordosis Angles and Spine Stability 139
6.4 Additional Future Work 140
Appendix A 149
A.1 Moment of Inertia Equations 149
A.2 Lordosis Angles 150
A.3 Skeletal and Muscle Coordinates 152
A.4 Rotations 162
Appendix B 164
B.1 Code Directory 165
B.2 MAT-file Directory 166
B.3 Personal Communication Summary 166
Appendix C : Main MATLAB Files 170
C.1 *spine_main.m*: Main Spine File 171
C.2 *For_verification.m:* Verification Tasks in MATLAB 176
C.3 *model_parameters.m*: Model Parameters 185
C.4 *rotationmatrices.mlx*: Determine the Rotation Matrices 189
C.5 *origin_body.mlx*: Define origins of the bodies 194
C.6 *transform_Cholewicki.mlx*: Import and Transform Skeletal Geometry 197
C.7 *load_musclefile.mlx*: Import Muscle Anatomy 199
C.8 *load_forcefile.mlx*: Import External Force File 202
C.9 *rotate_anatomy_sym.mlx*: Transform Muscle File, Calculate Muscle & Velocity 204
C.10 *rotateforce.mlx*: Transform External Force File 209
C.11 *dynam.mlx*: Dynamics 211
C.12 *generalizedForce.mlx*: Generalized Forces 217
C.13 *optimize.mlx*: Optimization 220
C.13.1 Main File 220

C.13.2 Angular Acceleration Functions 224
spineaccP.m file utilized for equilibrium constraint 224
spineaccA.m file utilized for equilibrium constraint 225
spineacc.m file utilized for equilibrium constraint 227
C.13.3 *run_fmincon.mlx*: Constrained Optimization Solver *fmincon* (MATLAB) 229
C.13.4 *costfun.m*: Metabolic Power Function for Constrained Optimization Procedure 230
C.13.5 *nlcon.m*: Nonlinear Constraints for Constrained Optimization Procedure 231
C.14 *musclemodel.mlx*: Muscle Model 232
C.15 *musclemodelR.mlx*: Reflex Model 233
C.16 *IVD_calcs.mlx*: Intervertebral Disc Moments 234
C.17 Time Delay Analysis 237
C.18 Nonlinear Verification 238
C.18.1 Run Nonlinear Simulation 238
C.18.2 *Spinedyn.mlx*: Function Handle for Delay Differential Equation Solver 239
C.18.3 Determine the Distance of the Normalized State-Space from Equilibrium 240
Appendix D : Additional MATLAB Files 241
D.1 *replace_sym.mlx*: Replace symbolic variables with numeric 241
D.2 *vect2ang.m*: Assign angles from input angle vectors to individual variables for function input 242
D.3 *sub_timevar.mlx*: Replaces angles in variables with time-dependent angles 243
D.4 *tofuncts.m*: Convert Symbolic Outputs to Function Handles 244

List of Figures

Figure 1-1: Cholewicki and McGill Model Utilization and Influence 9
Figure 1-2: Granata & Wilson Utilization & Influence 11
Figure 1-3: The expansion of the Wagner et al. model[12] 13
Figure 1-4: Model Process Summary 15
Figure 1-5: Visual of normal lordosis, hyperlordosis and hypolordosis in the lumbar spine region 18
Figure 2-1: Model rigid bodies 20
Figure 2-2: Vertebral body diameters. Not drawn to scale 22
Figure 2-3: Calculating the required vertebral height 24
Figure 2-4: Front view of vertebra. Origin defined at inferior vertebral surface 25
Figure 2-5: Front view of vertebra. Center of mass depicted by red circle 26
Figure 2-6: N basis vectors affixed to the ground. Axis 2 points anteriorly 27
Figure 2-7: Planar rotation 29
Figure 2-8: Twist rotation 29
Figure 2-9: Muscle Anatomy 30
Figure 2-10: Muscle attachment points visualization 31
Figure 2-11: Hill-type muscle model 33
Figure 3-1: Initializing the verification procedure in spine_main.m file 48
Figure 3-2: Example of exporting verification task results to Excel spreadsheet 49
Figure 3-3: "Removing" the default lordosis angles: Transforming coordinates to align the connection points of the vertebrae along Axis 3. 52
Figure 3-4: Determining Franklin's initial coordinates prior to transformation for selected attachment points. 53
Figure 3-5: Verification Tasks 2.2 and 2.4 56
Figure 3-6: Verification Task 2.5 Absolute Error 56
Figure 3-7: Verification Task 3.2 57
Figure 3-8: Verification Task 3.2 Absolute Error 57
Figure 3-9: Verification Task 4.1 59
Figure 3-10: Verification Task 4.1 Absolute Errors (presented in the MATLAB command window) 60
Figure 3-11: Verification Task 5.1 60
Figure 3-12: Verification Task 5.1 Absolute Error 61
Figure 3-13: Verification Task 6.1 62
Figure 3-14: Verification Task 6.1 Absolute Error from MATLAB Command Window 62
Figure 3-15: Verification Task 7.1 62
Figure 3-16: Verification Task 7.1 Absolute Error for muscle length 62
Figure 3-17: Verification Task 7.1 Absolute Error for muscle velocity 63
Figure 3-18: Verification Task 8.1 63
Figure 3-19: Verification Task 8.1 Absolute error presented in the MATLAB command window 63
Figure 3-20: Verification Task 9.1 64
Figure 3-21: Verification Task 9.1 Absolute error 64

Figure 3-22: Verification Task 9.2 65
Figure 3-23: Verification Task 9.2 Absolute Error for Gravity Vector 65
Figure 3-24: Verification Task 9.2 Absolute Error for Mass Matrix 65
Figure 3-25: Verification Task 9.4 66
Figure 3-26: Verification Task 9.4 Absolute Error for Gl 66
Figure 3-27: Verification Task 9.4 Absolute Error for Gr 67
Figure 3-28: Verification Task 10.1 68
Figure 3-29: Verification task 10.2 in the IVD_calcs.mlx script 68
Figure 3-30: Verification Task 11.1 69
Figure 3-31: Verification Task 11.1 Absolute Error 69
Figure 3-32: Verification Task 12.1 70
Figure 3-33: Continued verification of 12.1 task 71
Figure 3-34: Verification Task 12.1 Absolute Error for Aeq 71
Figure 3-35: Verification Task 12.1 Absolute Error for beq 71
Figure 3-36: Verification Task 12.1 Absolute Error for B 72
Figure 3-37: Verification Task 12.1 Absolute Error for A 72
Figure 3-38: Verification Task 12.5 73
Figure 3-39: Verification Task 12.3 73
Figure 3-40: Verification Task 12.3 Absolute Error 74
Figure 3-41: Verification Task 12.4 74
Figure 3-42: Verification Task 12.4 Absolute Error 75
Figure 3-43: Verification Task 12.6 75
Figure 3-44: Verification Task 12.6 Absolute Error 75
Figure 3-45: Verification task 13.1 76
Figure 3-46: Absolute error for muscle force 76
Figure 3-47: Dss for: simulated delay = 0.99 * delay margin 77
Figure 3-48: Dss for: simulated delay = 0.75 * delay margin 78
Figure 3-49: Delay Margin * 0.5 78
Figure 3-50: Dss for: simulated delay = 1.25 * delay margin 79
Figure 3-51: Dss for: simulated delay = delay margin 79
Figure 4-1: Spine model. This model includes five rigid lumbar vertebrae, one rigid body representing the cervical and thoracic spine regions, and a fixed pelvis. 90 muscle fascicles are included. The intervertebral discs are represented by a lumped parameter model 100
Figure 4-2: The relationship between required metabolic power and stiffness gain are presented for a simulation including an external load of 200 N. This load was applied at the T4 level and 20 cm anterior to the trunk. For these simulations, GP = 10, GD = 0 and b 101
Figure 4-3: The relationship between the required muscle force to stabilize the system and stiffness gain is presented. For these simulations, an external load of 200 N was applied at the T4 level and 20 cm anterior to the trunk and GP = 10, GD = 0 and b = 2. The 102

Figure 5-1: The current model incorporates 90 muscle fascicles, one rigid body for each of the lumbar vertebrae and one rigid body representing the thoracic and cervical region. A lumped parameter model is utilized to represent the intervertebral discs 126
Figure 5-2: The appearance of hyperlordosis, hypolordosis and normal lordosis in the lumbar spine ... 127
Figure 5-3: The metabolic power required to stabilize each lordosis case. One simulation was unloaded while the other was loaded with a 200 N load anterior to the trunk. 128
Figure 5-4: Muscle activation for the right-side muscles for the unloaded case 129
Figure 5-5: The distance of the normalized state-space from equilibrium versus time for the unloaded hyperlordotic spine 130
Figure 5-6: The distance of the normalized state-space from equilibrium versus time for the unloaded hypolordotic spine 131
Figure 5-7: The distance of the normalized state-space from equilibrium versus time for the unloaded spine with normal lordosis 132
Figure 5-8: The distance of the normalized state-space from equilibrium versus time for the hyperlordotic spine with a 200 N load applied anteriorly at the T4 level 133
Figure 5-9: The distance of the normalized state-space from equilibrium versus time for the hypolordotic spine with a 200 N load applied anteriorly at the T4 level 134
Figure 5-10: The distance of the normalized state-space from equilibrium versus time for the spine with normal lordosis and a 200 N load applied anteriorly at the T4 level 135
Figure A-1: Elliptical Cylinder Top View 149
Figure A-2: Lordosis angle applied around Axis 1 151
Figure A-3: Franklin vs Cholewicki Axes Definition. Franklin Axis 2 and Cholewicki X Axis point anteriorly 153
Figure A-4: Rotation around Axis 1 162
Figure A-5: Rotation around Axis 2 162
Figure A-6: Rotation around Axis 3 163
Figure B-1: General Model Flowchart 164

List of Tables

Table 1-1: Overview of pivotal stability-based trunk models 14
Table 1-2: Magnitude of the stiffness gains in literature 17
Table 2-1: Spine segment properties. 21
Table 3-1: Verification Tasks and Results 44
Table 3-2: Franklin muscle paths described in appendix for left side of the body 54
Table 3-3: Cholewicki and McGill muscle paths described in appendix for left side of the body 55
Table 4-1: Evidence of the varying stiffness gain q magnitudes utilized in previous investigations 103
Table 5-1: The percentage of metabolic power required to recruit certain muscle groups to achieve stability in the unloaded cases 136
Table 5-2: Critical stiffness for for the hypolordosis (32.1˚), normal lordosis (58.1˚), and hyperlordosis (84.1˚) cases for simulations with a 200 N load applied at level T4 and 20 cm anterior to the trunk 137
Table A-1: Development of the Normal Lordosis Case 150
Table A-2: Segmental lordosis angles for all lordosis cases studied in Chapter 5 151
Table A-3: Skeletal Geometry Coordinates 153
Table A-4: Muscle Attachment Points 158
Table B-1: Code Directory of the scripts utilized in the model with a general description of their purpose 165
Table B-2: MAT-file directory for the .mat files utilized in the model. Many outputs of functions were saved as symbolic .mat files, allowing for these outputs to be updated to function handles in the tofuncts.m file 166
Table B-3: Summary of the MATLAB files received from Dr. Michael Madigan and the files Franklin has published in his thesis 167

List of Abbreviations

AP diameter: Diameter in the anterior-posterior direction

AP radius: Radius in the anterior-posterior direction

DOF: Degrees of freedom

EMG: Electromyography

FEA: Finite Element Analysis

FEM: Finite Element Model

IVD: Intervertebral disc

KE: Kinetic Energy

LBP: Low back pain

LLA: Lumbar lordosis angle

ML diameter: Diameter in the medio-lateral direction

ML radius: Radius in the medio-lateral direction

PE: Potential Energy

Chapter 1: Introduction

1.1 What causes low back pain and injury?

1.1.1 Prevalence of Low Back Pain

Low back pain (LBP) and injuries place significant financial and physical burdens on Americans. The estimated cost of low back pain in the United States surpasses 100 billon dollars annually due to the need for medical intervention and lost income[1]. According to the 2015 National Health Interview Survey, approximately one out of four individuals reported experiencing low back pain within a span of three months[2].

The causes of low back pain and injuries have not been fully identified, but researchers have progressed in their understanding of the subject. Researchers have concluded that risk factors, history of low back pain[3], differences in muscle stiffness and recruitment strategies[4–6], and impaired reflexes[7,8] may contribute to an individual's likelihood to experience low back pain or injury; this list is not exhaustive but rather indicative of the progress that has been made in this area of study. Low back pain and injury is likely not the consequence of one simple cause, resulting in researchers considering and investigating a multitude of parameters that may impact spine behavior and potential injury. The investigation of the causes of low back pain is on-going and is crucial to the development of prevention strategies, and the advancement of rehabilitation techniques and treatment methods.

1.1.2 Modeling of the Human Trunk

Various biomechanical trunk models have been developed to further investigate the behavior of the lumbar spine and improve our understanding of the causes of low back pain and spinal injury[9]. Researchers have focused on characteristics such as spinal loading, buckling, spine stability, and muscle

recruitment patterns, but the methods used to model the lumbar spine may impact the model's ability to adequately predict spine behavior.

For example, in literature, biomechanical models vary in complexity and anatomical accuracy. Some researchers have represented the spine as a single pendulum with a pair of antagonistic muscles[10–12] or as a double inverted pendula model with more than one muscle pair[13,14]. This representation is simple in comparison to those that have modeled the spine with greater degrees of freedom and more realistic muscle architecture[15–17]. It has been determined that representing muscle paths as curved more accurately models the true behavior[18]; despite this conclusion, some researchers still implement straight muscle paths for simplicity. While more complex models incorporate more realistic anatomy, the development of these models is more elaborate, resulting in the increased likelihood of possible unidentified errors. Additionally, in the complex models, it may be difficult to determine the specific or root cause of the system's behavior.

For models that incorporate many muscles, the method for determining the muscle activation for each muscle can have limitations. Researchers can employ an optimization procedure that solves for the activation of each muscle, but this has been found to not always adequately predict the coactivation of the muscles[9,19] and/or its magnitude[13]. Coactivation has been proven to be crucial to spine stability and will result in an increased compressive load[13,19,20]. Other researchers utilize EMG-assisted or EMG-assisted optimization procedures to determine muscle activation. These methods have been found to better represent coactivation but introduce errors due to the limitations of surface EMG sensors, such as not being able to directly measure the activation of deep muscles[9,15].

In the literature, the classification of spine "stability" and "instability" has been shown to be inconsistent in use across researchers[21,22]. In this context, the behavior of the spine is of interest prior to and following an applied perturbation[21]. If the spine is unable to converge to its initial position within a

reasonable range, then this indicates instability of the spine and injury potential[21]. In the past, some researchers have neglected to include stability requirements in their models and have instead focused solely on the stiffness needed for the system to be in equilibrium following a perturbation. It has since been proven that a spinal system in equilibrium does not necessarily correlate to a stable system[13,23], and that when a stability constraint is included in the model, the predicted muscle activations more accurately represent those found in experimental studies[13].

Commonly, researchers have neglected to include reflexes in models due to their time-delayed properties and the added complexity of their implementation[9,24]. It has been determined that for certain loading conditions, reflexes are required for successful spine stabilization[16], and that while some loads can be stabilized solely by intrinsic stiffness, reflexes allow for the same loads to be stabilized with less coactivation required[16,17,25]. Additionally, individuals with low back pain have exhibited a greater reflex delay in comparison to healthy controls[4,8]. The consequence of this greater delay may be instability or injury, as the reflexes may be too delayed to aid in spine stabilization strategies and the system may be unable to compensate elsewhere to stabilize[21]. Impaired reflex response has been associated with a heightened risk of LBP and injuries[3]. Researchers have shown that reflexes impact spine behavior and should continue to be investigated experimentally and through modelling for the investigation of LBP.

Lastly, model predictions need to be verified with experimental findings when possible. For spine model predictions to be beneficial, it is crucial for models to predict realistic spine behavior. If unable to do so, refinement of the model should be explored.

1.2 Background: Stability-based Spine Models and Utilization

1.2.1 The importance of spine modeling in the investigation of low back pain

The complexity of the human spine makes it difficult to isolate certain components of the system and deduce their sole impact on spine behavior with confidence. This directly complicates the ongoing

investigation of the causes of low back pain. Experimental studies have provided researchers with vast information and insight to spine properties and movement, such as spine buckling, muscle activations in loading conditions, and vertebral and intervertebral disc (IVD) dimensions. For some of this information, like muscle recruitment strategies, it may be difficult to determine the specific mechanism or mechanisms that explain why this certain behavior occurred. Experimental studies are crucial for understanding *how* the spinal system reacts to loading conditions, whereas modelling allows for more insight as to *why* the system may have responded this way. Thus, it is critical that modeling and experimental data collection continue to be utilized in tandem to further our understanding of the mechanisms behind low back pain.

1.2.2 The importance of spine stability analysis in the investigation of low back pain

Recall that the concept of spine stability is based on the spinal system's ability to return to equilibrium within a reasonable range following a perturbation[21]. If unable to do so, this can result in potential LBP or injury. Additionally, it has become evident that when perturbed, the muscle recruitment strategy utilized is influenced by the requirements for spine stability[20] and when stability constraints are utilized in trunk models, the predicted activations better match those measured experimentally[13]. Thus, spine stability should continue to be considered and evaluated in future studies.

As mentioned previously, researchers have addressed the spine as a mechanical system, evaluating the components that contribute to its stability and factors that may result in its buckling[18,26]. Crisco *et al*. [27]determined that a compressive load of approximately 88 N will result in spine buckling, indicating that the spine itself cannot be the only contributor to spine stability[28]. To avoid buckling, muscles and reflexes are recruited to combat applied loads, maintain posture, and minimize injury potential. Using electromyography (EMG), researchers can measure the activation of the muscle groups in subjects, providing insight to the muscle recruitment strategies utilized by the subjects in different

loading conditions. This has proven to be useful in the comparison between healthy controls and those with LBP. Experimental studies have revealed that individuals with chronic low back pain have an increased reflex time-delay in comparison to healthy controls, which can adversely affect spine stability[4,7,11]. This may result in individuals with chronic LBP utilizing different muscle recruitment strategies in an attempt to stabilize the spine by other means[4,6]. In the next section, an overview of spine modeling will be provided and the seminal stability-based trunk models in literature will be discussed.

1.2.3 Stability-based Trunk Models in Literature

Spine models have varied in their utilization, development, and characteristics throughout history. First, equilibrium-based models will be discussed. These models focus on the balancing of forces and moments. For static analysis:

$$\Sigma\, Forces = 0 \tag{1-1}$$

$$\Sigma\, Moments = 0 \tag{1-2}$$

And for dynamic analysis:

$$\Sigma\, Forces = ma \tag{1-3}$$

$$\Sigma\, Moments = I\alpha \tag{1-4}$$

Where *m* is mass, *a* is acceleration, *I* is the mass moment of inertia and α is angular acceleration.

Finite element models and link-segment models have often been developed and utilized for equilibrium analysis of the spine[9]. While these equilibrium-based models assist in the understanding of spine loading, the structural properties of the spine are not evaulated[9]. To evaluate these structural properties, stability analysis has been implemented into trunk models in addition to the equilibrium

analysis. This implementation of stability analysis in trunk models was validated by a study completed by Granata and Orishimo which indicated that a spine in equilibrium does not necessarily reflect a stable spinal system, and that muscle recruitment strategies obtained experimentally more accurately matched those predicted for a system with a stability requirement[20].

Many trunk models were expanded to include stability analysis. These stability-based trunk models often implement the Minimum Potential Energy Method based on the Hessian matrix or Linear Stability Analysis based on the Jacobian matrix. For a system to be stable for the Minimum Potential Energy Method, the principal minors and the determinant of the Hessian matrix need to be positive[15]. For Linear Stability Analysis, the eigenvalues of the Jacobian must have negative real parts for a system to be stable[17]. Trunk models evaluating stability commonly consist of an inverted pendulum with a pair of antagonistic muscles or stacked inverted pendula where a single pendulum is used to represent each of the lumbar vertebrae. For the models that incorporate the stacked pendula, a more-detailed muscle architecture is included which allows for the roles of each of the muscle groups to be investigated more clearly.

With these trunk models, researchers often use optimization procedures, measured EMG from subjects, or EMG-based optimization procedures to determine the muscle activation utilized for an assigned task. Often, researchers utilize EMG to measure the muscle activations of human subjects that can be utilized for the model. This allows for realistic muscle recruitment strategies to be utilized in the model as subjects may utilize different strategies for the same task. Optimization procedures that do not require subject input are useful in that the muscle activation necessary to satisfy stability and equilibrium requirements can be determined in an efficient and timely manner. This allows for preliminary studies to be completed which can later be validated with experimental data. Additionally,

through the use of optimization-based modeling, vulnerable populations that may be unable to complete experimental tasks can still be investigated without putting subjects at risk.

In the following sections, the characteristics and utilization of trunk models in literature will be discussed. Due to the nature of the model utilized in this work, the focus will primarily be on lumped parameter stability-based trunk models. Finite element trunk models that include stability analysis will be discussed only briefly due to the differences in modeling methodologies and characteristics.

1.2.3.1 Overview of Finite Element Models

Finite element models (FEM) with stability analysis have been utilized to investigate spine behavior for different tasks[23,29–43]. Kiefer *et al*. and El-Rich *et al*.[30] utilized a FEM to investigate upright standing posture. This model included five rigid bodies representing the lumbar vertebrae, one rigid body representing the thoracic spine, six beams representing the stiffness of the functional spinal units, and 60 and 56 muscle fascicles respectively[31]. This model was also utilized by Arjmand and Shirazi-Adl[29] to determine stability margins and spinal loading in isometric forward flexion tasks with 56 muscle fascicles included in the model. Hajihosseinali *et al.*[40] developed a similar FEM incorporating stability requirements in the optimization procedure and implementing curved muscle paths, resulting in muscle activation predictions more realistically representing coactivation. The Minimum Potential Energy Method was utilized in the Hajihosseinali *et al.* FEM[40,44].

Gardner-Morse *et al*. utilized an FEM model in the investigation of the relationship between intrinsic muscle stiffness and spine stability[23]. This model incorporated 66 muscle fascicles, a fixed pelvis, utilized beam elements to represent the stiffness of the functional spinal units, and a rigid body to represent the thorax[23]. The stability of the system was analyzed using eigenvalue buckling analysis[23]. Additionally, Gardner-Morse and Stokes utilized this model to investigate the possible relationship between

coactivation and spine stability requirements[33]. For these investigations, stability analysis was completed and a the critical magnitude of stiffness gain *q* was determined[23,33].

1.2.3.2 Bergmark

The investigation of the mechanical stability of the lumbar spine by Bergmark has been an incredibly influential work[18], inspiring many researchers to use similar methods to model the trunk and analyze mechanical stability. In Chapter 6 of his work, Bergmark details his lumbar spine model, consisting of five lumbar vertebral rigid bodies and one rigid body representative of the thoracic spine region. These rigid bodies are stacked and connected by springs representative of the intervertebral discs, constrained to a fixed pelvis. 40 muscle fibers are included and represented as linear springs with Bergmark's model of short-range stiffness (discussed in section 1.4.1), with the global erector spinae being represented with curved paths and the other muscles utilizing straight paths. Through the investigation of unloaded standing and loaded standing with differences in posture, the critical magnitude of *q* could be determined to result for a stable system. The short-range stiffness model proposed by Bergmark[18] (section 1.4.1) is commonly used for muscles in trunk models that are stability-based.

1.2.3.3 Cholewicki & McGill

Cholewicki and McGill developed a more realistic lumbar spine model that incorporated 90 muscle fascicles and an EMG-assisted optimization procedure[15]. This model is unique in that it incorporates a rigid link body model for the external forces, a distribution-moment muscle model for muscle force approximations based on the measured EMG, and an 18 degree of freedom spine model used to estimate the moments generated by the muscles and passive tissues[15]. Using the EMG-based optimization procedure, the predicted muscle activations were scaled as necessary to ensure that the moments were balanced, and the system was mechanically stable.

This model evaluated the mechanical stability of the spinal system during slow dynamic tasks, utilizing the Minimum Potential Energy Method[15]. This model allowed for subject-specific data collection, with model parameters adjusting based on a subject's anatomy and recruitment strategies which can vary between subjects[15].

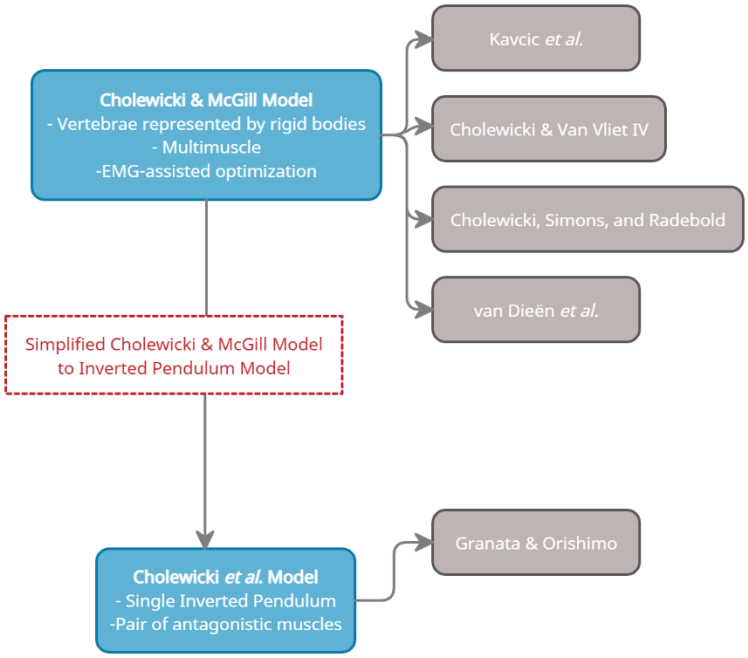


Figure 1-1: Cholewicki and McGill Model Utilization and Influence

This Cholewicki and McGill lumbar spine model[15] has been utilized to perform stability analysis in other experimental studies. For instance, utilizing the Cholewicki and McGill model[15], Cholewicki and Van Vilet[45] and Kavcic *et al.*[46] investigated the stabilizing potential of the muscle groups during different loading tasks to determine if a sole muscle group is primarily responsible for spinal stability (Figure 1-1). Both groups determined that no sole muscle was primarily responsible for spinal stability but rather other factors, such as the loading task, affected the muscle recruitment required to stabilize the system[45,46]. In the Van Dieën *et al.* study[6], the Cholewicki and McGill model[15] aided in the comparison of muscle recruitment strategies and stability in healthy controls and individuals with LBP, yielding the

conclusion that recruitment strategies differ between the groups likely due to stability requirements. Additionally, in the Cholewicki, Simons and Radebold study[47] focused on the influence of the positioning and magnitude of external loads on trunk stability, the Cholewicki and McGill model[15] was utilized for determining the spine stability based on the subjects' measured EMG.

Cholewicki *et al.*[10] developed a single inverted pendulum model with a pair of antagonistic muscles representative of the flexor and the extensor muscle groups, referred to as a simplified Cholewicki and McGill model[15] (Figure 1-1). The short-range stiffness model from Bergmark[18] was utilized and the Minimum Potential Energy method was utilized for stability analysis. This model[10] was expanded and modified by Granata and Orishimo in their investigation of the influence spinal stability has on the neuromuscular response during lifting tasks[20] through the expansion of the muscle model to include the contractile element and differing upright muscle heights. From the Granata and Orishimo study, it was evident that co-contraction increased with increased heights of the load in the lifting task indicating that the system is influenced by stability requirements[20].

1.2.3.4 Granata & Wilson

Granata and Wilson developed a novel double inverted pendulum to investigate the relationship between posture and stability[13]. This model incorporates 12 muscles modelled as linear springs with the short-range stiffness model developed by Bergmark[18]. The Minimum Potential Energy Method is utilized for the stability analysis. A constrained optimization procedure was utilized to determine the muscle recruitment strategy, requiring that stability and equilibrium conditions be met. Additionally, to verify the predicted muscle recruitment from the model, experimental data collection was performed, affirming that the model adequately predicted coactivation with the stability and equilibrium requirements implemented in the optimization procedure.

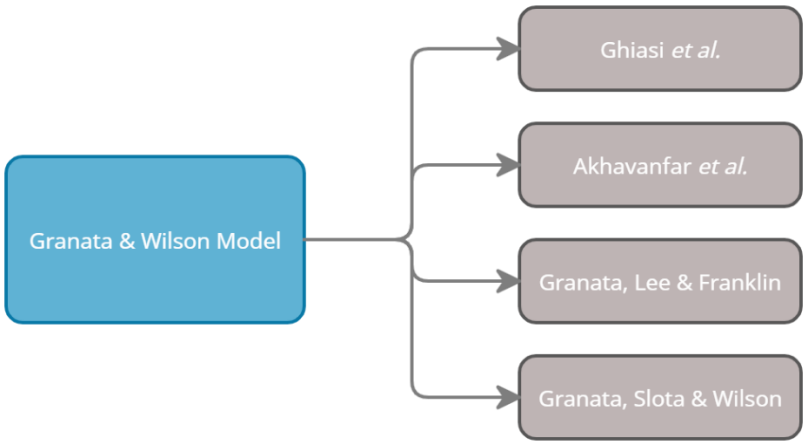


Figure 1-2: Granata & Wilson Utilization & Influence

Ghiasi *et al*. adjusted the optimization procedure from the Granata and Wilson model[13] to incorporating requirements of maximizing the stability index and minimizing the sum of the cubed muscle stresses[48]. Akhavanfar *et al.*[49] developed the Granata and Wilson model[13] in OpenSim, a musculoskeletal modeling software, and compared the predicted results from the model for a static optimization procedure with and without a stability constraint (Figure 1-2). Akhavanfar *et al*. determined that when the stability requirement was included in the optimization procedure, the predicted muscle activations were more similar to the experimental measurements obtained by Granata and Wilson[13]. This verifies the results from Granata and Wilson that stability requirements should be included in the optimization procedure to yield more realistic muscle recruitment strategies[13].

Granata, Lee and Franklin[14] utilized the Granata and Wilson model[13] in their investigation of spine stability in trunk extension and flexion tasks (Figure 1-2). The Granata and Wilson model[13] was utilized to determine the gain of the muscle forces required for the system to be stable, in equilibrium, and so the muscle activation matched the experimental measures. Then, excluding the stability requirement, the muscle activation required for the system to be in equilibrium could be determined.

The force required for co-contraction could be calculated as the difference between the muscle force for the system with solely the equilibrium condition and the system with equilibrium and stability conditions. Granata, Lee and Franklin determined that co-contraction was greater in flexion than in extension of the trunk[14].

The Granata and Wilson model[13] was used in the Granata, Slota and Wilson[50] study (Figure 1-2) to investigate the potential relationship between fatigue and stability of the spine. To implement fatigue into the model, the paraspinal muscle stiffness was reduced. In order for the system to stabilize with the reduced muscle stiffness, a greater magnitude of antagonistic co-contraction was required[50].

1.2.3.5 Franklin et al.

Franklin *et al.* developed a trunk model that incorporated time-delayed reflexes to evaluate the potential role that reflexes may have on spine stability[16,17,24]. Previous researchers did not consider or incorporate time-delayed reflexes into the trunk model for simplicity and due to the commonly accepted assumption that intrinsic stiffness has a more crucial role in spine stability and that reflex stiffness may be unnecessary[16,24]. Franklin *et al*. determined that in certain loading conditions, reflex stiffness was required to stabilize the system[16]. This conclusion motivated other researchers, such as Liebetrau *et al.*[11] and Zeinali-Davarani *et al*.[51], to expand models to include reflexes and continue the investigation of the mechanisms of low back pain while including reflexes in the analyses. Additionally, experimental works such as those by Radebold *et al.*[8] and Cholewicki *et al.*[3] emphasize the importance of including reflexes in spine models as they identified the plausible correlation between greater reflex response times and an increased risk of LBP and injuries in individuals with chronic low back pain in comparison to healthy controls. Clearly, reflexes need to be considered in the investigation of LBP.

In addition to the reflexes, the Franklin *et al.* model includes a fixed pelvis with independent rigid bodies representing each of the lumbar vertebrae and the thorax[16,17,24]. Each rigid body has the ability to

rotate, resulting in an 18-DOF system. Franklin *et al.* developed a lumped intervertebral disc model applied between the rigid bodies[16,17,24]. The skeletal and muscle coordinates are based on those used in the Cholewicki and McGill model[15]. There are 90 Hill-type muscle fascicles with Bergmark's model for short-range stiffness[18] included.

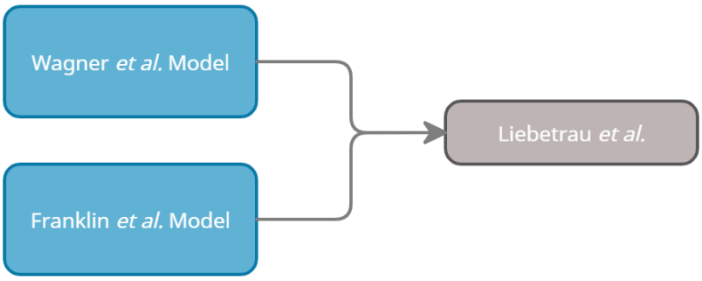


Figure 1-3: The expansion of the Wagner et al. model[12]

Liebetrau *et al*.[11] expanded the Wagner *et al.* model[12] through the incorporation of reflexes based on the work of Franklin *et al.*[16] (Figure 1-3). The Wagner *et al.* model incorporates an inverted pendulum with a pair of Hill-type antagonistic muscles[12]. Stability is assessed using Linear Stability Analysis based on the eigenvalues of the Jacobian matrix[12]. Through the investigation of lateral perturbations of healthy controls and individuals with low back pain, Liebetrau *et al*. determined that reflex delay and gain are inversely related for a stable spinal system.[11]

1.2.4 Summary

Table 1-1 summarizes characteristics of the models discussed in this section for comparison. These models range in complexity and in their characteristics but each model has had a vast impact on modeling techniques and the utilization of spine models to investigate the etiology of low back pain.

Table 1-1: Overview of pivotal stability-based trunk models

For the Minimum Potential Energy Method, a system is stable if the Hessian matrix, based on the second derivative of the system's potential energy, is positive definite.

For Linear Stability Analysis, after linearizing a system about equilibrium, a system is stable if the eigenvalues of the Jacobian need to have negative real parts.

	Model Type	Reflexes? (Y/N)	Muscle Paths	Muscle Model	Stability Method
Bergmark[18]	Stacked rigid bodies	No	Curved paths for global erector muscles; straight paths for others	Spring with Bergmark's stiffness model	Minimum Potential Energy Method
Cholewicki & McGill[15]	Stacked rigid bodies; EMG-Assisted Optimization	No	Centroid line approach; some curved lines of action with nodes	DM Muscle Model	Minimum Potential Energy Method
Cholewicki *et al.*[10]	Single Inverted Pendulum	No	Straight	Spring with Bergmark's stiffness model	Minimum Potential Energy Method
Franklin *et al.*[16,17,24]	Stacked rigid bodies; Optimization	Yes	Curved	Hill-Type with Bergmark's stiffness model	Linear Stability Analysis
Granata & Wilson[13]	Inverted Double Pendulum; Optimization	No	Straight	Spring with Bergmark's stiffness model	Minimum Potential Energy Method
Liebetrau *et al.*[11]	Inverted pendulum	Yes	Straight	Hill-Type	Linear Stability Analysis
Wagner *et al.*[12]	Single Inverted pendulum; optimization	No	Straight	Hill-Type	Linear Stability Analysis
Zeinali-Davarani *et al.*[51–53]	Single Inverted pendulum; optimization	Yes	Straight	Force-Length and Force-Velocity	Linear Stability Analysis

1.3 Model Development: Based on the Work of Franklin *et al.*

Throughout his graduate career at Virginia Polytechnic Institute and State University, Timothy C. Franklin developed a novel spine model that investigated the impact of reflexes on spinal stability[16,17,24]. At this point in time, many researchers did not include reflexes in their spine models, primarily for simplicity and/or because it was assumed that the spine may stabilize mainly or completely from intrinsic stiffness without the utilization of reflexes[24]. Franklin *et al.* investigated the impact of reflexes on spinal stability and determined that in certain loading conditions, reflexes were required for spine stability to be achieved[16,17,24].

Through personal communication with the Musculoskeletal Biomechanics Laboratory at Virginia Polytechnic Institute (Principal Investigators: Dr. Kevin P. Granata and Dr. Michael Madigan), the Human Motion Control Laboratory at the University of Kansas received some of the files developed by Franklin *et al*. for the model (Michael Madigan, personal communication, July 26, 2008). Through comparison of these files to the files published in the appendix of Franklin's thesis, it was determined that the files did not match the final version of the model's files. Therefore, our objective became redevelopment and improvement of the model.

1.3.1 Model Summary

In a brief summary, the Franklin *et al.* model includes separate rigid bodies representing each of the lumbar vertebrae, one rigid body representing the cervical and thoracic regions of the spine, and a fixed pelvis[24]. The muscle geometry was based on the coordinates provided by Cholewicki *et al.*, describing the attachment points of 90 muscle fascicles[15]. Through constrained optimization, the muscle activation that occurs when the metabolic

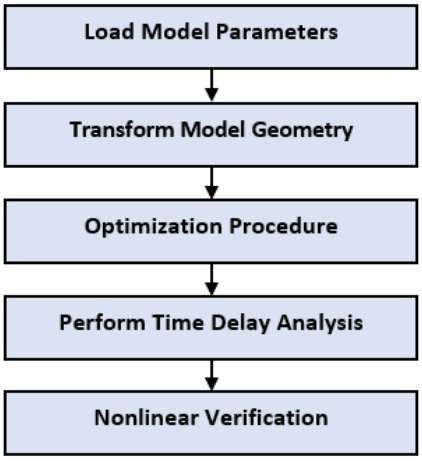


Figure 1-4: Model Process Summary

power is minimized and the system is stable and in equilibrium, can be determined for the linearized system. From the results of the optimization procedure, one can determine the time delay in which the system becomes unstable. Lastly, the nonlinear system can be simulated to verify the results from the linearized system.

Franklin *et al.* determined that reflexes were required to stabilize the spine under certain loading conditions[16] and determined that an increase in reflex gain resulted in less required metabolic power required for stabilization[16,17]. Additional conclusions and findings are discussed in their works[16,17,24].

1.4 Current Investigation

Building upon the studies completed by Franklin *et al.*[16,17,24], this work focuses on the investigation of the potential impact that dimensionless stiffness gain (discussed in 1.4.1) and lumbar lordosis angle (discussed in 1.4.2) may have on stability.

1.4.1 Bergmark's Model of Short-range Stiffness

Muscle stiffness has been characterized as demonstrating elastic behavior for small deformations referred to as short-range stiffness[18,54] (Equation 1-5):

$$K = \frac{q}{l_o} F \tag{1-5}$$

Where q is the dimensionless stiffness gain, F is muscle force, and l_o is the neutral length of the muscle.

Equation 1-5 is known as Bergmark's model for short-range stiffness[18]. For the dimensionless stiffness gain, Bergmark[18] utilized a value of 40 in his analysis. The magnitude of this dimensionless stiffness gain was further investigated by Crisco *et al.*[55] which resulted in the conclusion that this coefficient could fall within a range of 0.5 to 42. In past studies, the dimensionless stiffness gain q

utilized has varied between researchers (Table 1-2), prompting our interest as to how the magnitude of the dimensionless stiffness gain may impact the force required to stabilize the spine.

Table 1-2: Magnitude of the stiffness gains in literature

Stiffness gain is a dimensionless parameter ranging between 0.5 and 42 in magnitude.[55] This table demonstrates how the stiffness gain is inconsistent between studies.

Source	Stiffness Gain *q*
Akhavanfar *et al.*[49]	5
Bergmark[18]	40
Brown *et al.* [19]	10
Cholewicki *et al.*[10]	30
Franklin *et al.*[17]	10
Granata & Wilson[13]	5
Hajihosseinali *et al.*[40]	5
Samadi & Arjmand[56]	1.8, 5, 10
Shamsi *et al.*[57]	5
Stokes & Gardner-Morse[34]	5
Vakilzadeh *et al.*[58]	2, 4, 6, 10
Zeinali-Davarani *et al.*[51]	2, 4, 10

1.4.2 **Hyperlordosis and Hypolordosis**

Researchers have not come to a consensus regarding whether there is a significant correlation between LBP and lordosis angle or not.[59] Many of these studies incorporated radiographic

measurements or magnetic resonance imaging[60] but did not investigate the impact that lordosis may have on spine stability through modeling.

Pesenti *et al.*[61] measured total lumbar lordosis from the superior endplate of S1 to the superior endplate of L1, resulting in a mean standing lordotic angle of 58.1˚± 13˚. In order to define the lordosis angles for hyperlordosis and hypolordosis, the definition from Fernand *et al.*[62] was used: hyperlordosis and hypolordosis as lordotic angles whose values fall outside of two standard deviations from the mean. Based on the mean standing lordotic angle presented by Pesenti *et al.*[61] (58.1˚± 13˚), hyperlordosis and hypolordosis would be defined as a lordotic angle greater than 84.1˚ and less than 32.1˚ respectively.

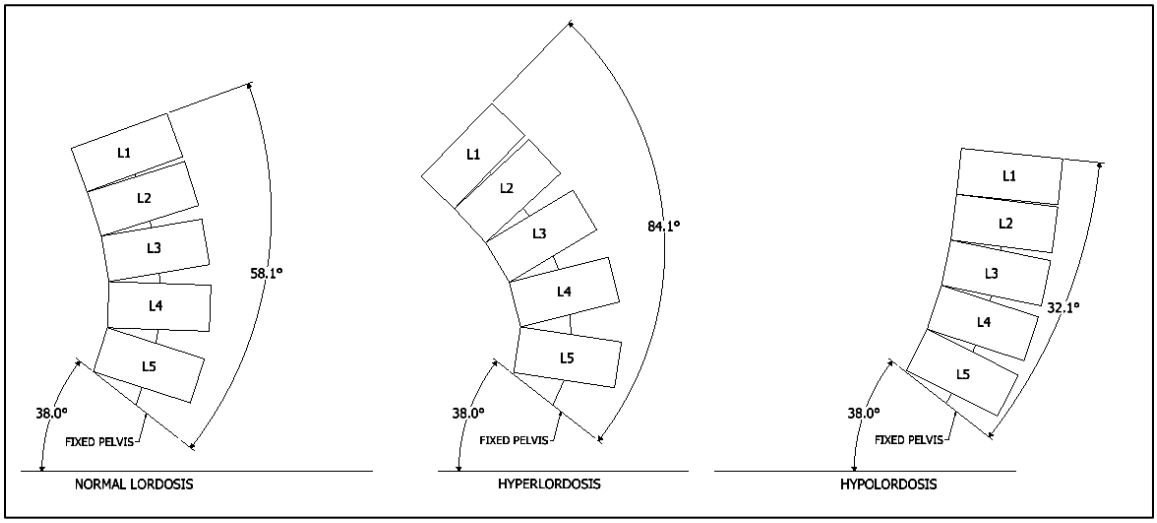


Figure 1-5: Visual of normal lordosis, hyperlordosis and hypolordosis in the lumbar spine region

1.5 Specific Aims and Hypotheses

Specific Aim 1: Redevelop the Franklin *et al.* spine model and perform code verification.

Specific Aim 2: Determine the influence the magnitude of stiffness gain *q* has on stability of the spine.

Hypothesis: Increasing the magnitude of stiffness gain *q* will allow for the spine to stabilize with less required force.

Specific Aim 3: Investigate the potential impact that hyperlordosis or hypolordosis may have on spine stability.

Hypothesis: With a load of 0 N applied vertically to T4 at a position 20 cm anterior to the trunk, the hyperlordotic spine will require increased recruitment of flexor muscles to stabilize, consequently increasing the required metabolic power.

Hypothesis: With a load of 0 N applied vertically to T4 at a position 20 cm anterior to the trunk, the hypolordotic spine will require increased recruitment of extensor muscles to stabilize, consequently increasing the required metabolic power.

Hypothesis: With a load of 200 N applied vertically to T4 at a position 20 cm anterior to the trunk, a greater critical stiffness will be required to stabilize the hypolordotic spine.

Chapter 2: Model Development

Our goal was to reconstruct the Franklin *et al.* spine model[16,17,24] and continue further investigation of the impact of model parameters on spinal stability. This model incorporates six rigid bodies, a fixed pelvis, a lumped parameter model representative of an intervertebral disc, 90 Hill-Type muscle fascicles and neuromuscular reflexes[16,17,24].

The model was reconstructed in MATLAB R2021b (Mathworks, Natick, MA). The MATLAB Scripts and Live Scripts utilized for the model can be viewed in Appendix C-D. The verification processes for these MATLAB scripts and functions are discussed in Chapter 3.

2.1 Rigid Bodies

The current spine model includes one rigid body representing the cervical and thoracic regions of the spine and five separate lumbar vertebrae rigid bodies, and a fixed pelvis[16,17,24] (Figure 2-1). Each of the vertebrae can rotate about all three axes, resulting in 18 degree of freedom system.

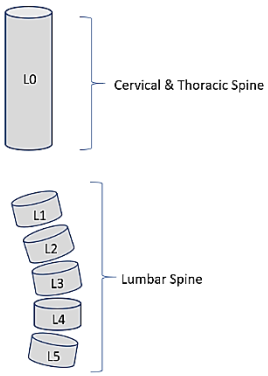


Figure 2-1: Model rigid bodies

The lumbar vertebral geometry has been measured by a few researchers[63–72] with discrepancies in the reported measurements most likely due to sample size, sample population characteristics, and methods used[73]. When determining the rigid body parameters that would be included in our model, sources were reviewed and compared.

Table 2-1: Spine segment properties.

The vertebral spacings for the lumbar vertebrae are based on Cholewicki et al.[54]. The coordinates provided by Cholewicki et al.[54] describe the spine in a neutral position, meaning that the lordosis of the spine is influencing these coordinates. To determine the skeletal coordinates without the influence of lordosis, these coordinates were rotated so that the connection points for the vertebrae were aligned along the vertical axis, resulting in the calculated vertebral spacing provided in this table.

For the cervical and thoracic region, the height of the cervical region is calculated based on the lumbar vertebral heights and lumbar intervertebral discs using a method by Gilad et al.[67] The utilized lumbar vertebral heights are from Berry et al.[64] and the calculated intervertebral disc height are based on the vertebral spacing. The height of the thoracic region is based on the vertebral heights from Panjabi et al.[74] and the thoracic intervertebral disc heights from Kunkel et al.[72]

The masses of the vertebrae are based on Liu et al.[63] and the lordosis angles for each body have been calculated based on the work of Pesenti et al.[61] See Appendix A.2 for more information on lordosis angle calculations for each of the bodies.

Spine Segment	Body Number	Vertebral Spacing (m)	Mass[63] (kg)	Normal Lordosis Angle[61] (deg)
Cervical & Thoracic Region	0	0.4241	16	0
L1	1	0.0363	2	20.1
L2	2	0.0386	2	17.5
L3	3	0.0379	2	9.6
L4	4	0.0370	2	-2
L5	5	0.0389	2	-17.8
Pelvis	p/6	N/A	N/A	-38

2.1.1 **Dimensions**

The rigid bodies are represented as cylinders with an elliptical cross-sectional area for simplicity[24]. In the current model, the radii of the trunk in the medio-lateral (ML) and the anterior-posterior (AP) direction are 0.024 meters and 0.017 meters respectively, based on body diameter measurements from Berry *et al.*[64] and Panjabi *et al.*[65] (Figure 2-2).

These measurements were obtained from the inferior and superior endplates of the lumbar vertebral bodies[64,65]. The average AP radius for the lumbar vertebrae based on the Panjabi *et al.*[65] and Berry *et al.*[64] measures was approximately 0.017 m for both studies. For the ML radius, there was more deviance in the results: for Panjabi *et al.*[65], the resulting average ML radius was approximately 0.023 m, while for Berry *et al.*[64], it was approximately 0.025 m. An average value of 0.024 m was utilized in our model for the ML radius.

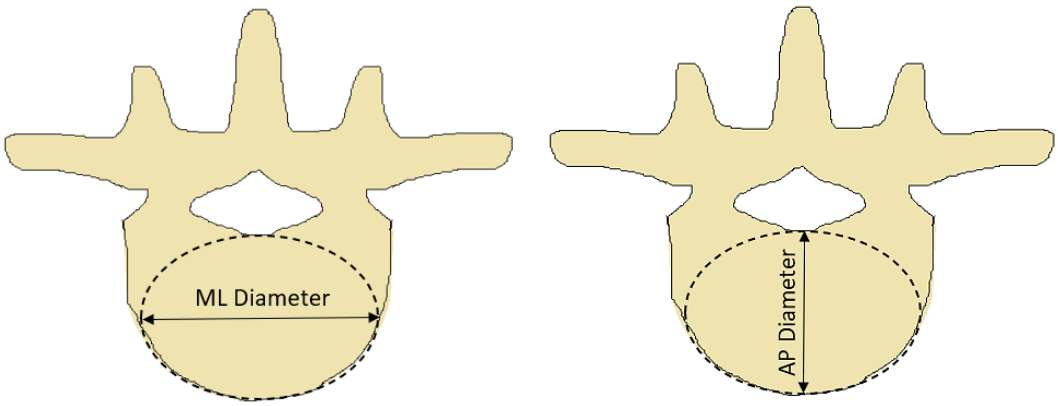


Figure 2-2: Vertebral body diameters. Not drawn to scale

While Franklin *et al.*[24] utilized the vertebral heights from Liu *et al.*, a cadaver study that included only one subject[63] the current model utilizes vertebral body heights based on the skeletal coordinates provided by Cholewicki *et al.*[54] These skeletal coordinates describe a spine in neutral position, indicating

that these coordinates are influenced by lumbar lordosis. In order to be able to adjust the lordosis angle magnitudes as necessary for our study, these coordinates were rotated so that the connection points for all vertebrae were aligned along Axis 3. Each vertebra has a connection point placed at the inferior and superior vertebral surfaces, which describes where the vertebra connects to the vertebra inferior and superior to it. The distance between these connection points will be referred to as the vertebral spacing for that vertebra in this work (Table 2-1).

The height of the cervical and thoracic regions of the spine were computed separately and then summed for a total height.

2.1.1.1 Cervical Spine Region Height

Gilad *et al.* measured the posterior and anterior height of the lumbar and cervical vertebrae and determined the height relationships between these regions of the spine. Using the vertebral spacing determined previously, the height of the lumbar vertebra and the intervertebral disc that encompasses that spacing needed to be calculated.

For the intervertebral disc heights of the lumbar region, measurements from Tibrewal *et al.* were utilized. Tibrewal *et al.* measured the lumbar intervertebral disc heights for 11 healthy controls and provided the range of disc heights. Using the maximum value of the range for the anterior and posterior disc height, the average maximum disc height could be determined. Using these maximum disc heights and the vertebral spacing from the Cholewicki *et al.* coordinates, the required vertebral height could be determined (Figure 2-3).

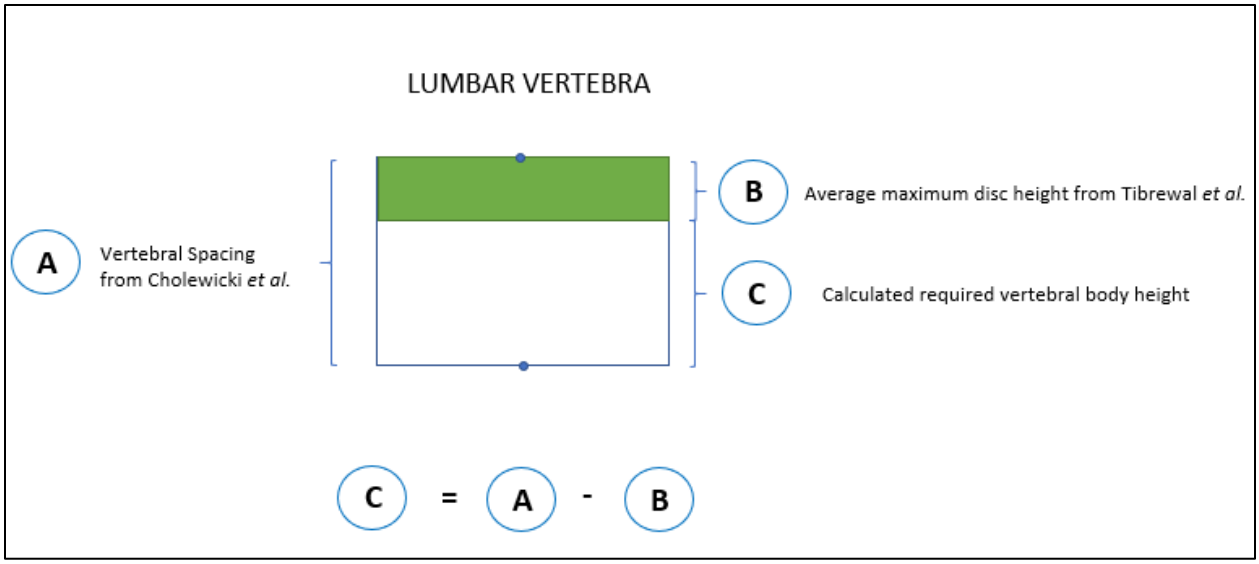


Figure 2-3: Calculating the required vertebral height

The mean lumbar vertebral heights from Gilad *et al.* were utilized to determine if the required vertebral height (Figure 2-3) was reasonable. Based on our calculations, the vertebral heights of L4 and L5 were sufficient without adjustment. L1, L2, and L3 required an increase in magnitude, correlating to +1.721*STD, +1.22*STD, and +0.239*STD. The average lumbar vertebral heights were adjusted to appropriately fit the vertebral spacing required by the skeletal coordinates.

Using the determined lumbar vertebral heights and intervertebral disc heights, the respective measures could be calculated for the cervical region.

2.1.1.2 Thoracic Spine Region Height

For the thoracic region, the height was determined from the spacing determined in Cholewicki *et al.* and compared to the values reported in Berry *et al.*[64] and Kunkel *et al.*[72]

Many of these sources did not differentiate between sexes in their results, so it is important to address the potential impact that sex may have on the average vertebral geometry measures across the sample. While it has been determined that lumbar vertebral heights are not sex-specific, as the

difference of the heights between sexes has been proven to be insignificant, the ML diameter does differ significantly between sexes[71,75].

2.1.1.3 Modifications to the 2006 Franklin Model

Franklin *et al.*[24] utilized a ML radius of 0.12 m and an AP radius of 0.06 m for the rigid bodies[24]. For the current model, the radii have been updated based on the vertebral diameters provided by Berry *et al.*[64] and Panjabi *et al.*[65] Additionally, Franklin et al. utilized body heights from Liu *et al.*[63] in his analysis whereas the current model utilized the vertebral spacing defined by Cholewicki *et al.*[54]

2.1.2 **Body Origins**

The origin of each vertebra is defined at the inferior vertebral endplate, as shown in Figure 2-4.

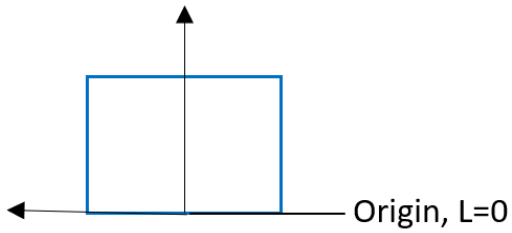


Figure 2-4: Front view of vertebra. Origin defined at inferior vertebral surface

The following equation can be used to determine the origin of a vertebral body:

$$origin\ of\ vertebral\ body = origin\ of\ inferior\ vertbral\ body + inferior\ vertebral\ spacing \quad (2\text{-}1)$$

The vertebral spacing (Table 2-1, Equation 2-1) is based on the muscle and skeletal coordinates provided by Cholewicki *et al*.[15] for a body in neutral spine position and is the height between the connection points between vertebrae. Comparing these coordinates to those provided by Franklin *et al*.[24], one can deduce that Franklin transformed the coordinates. The skeletal geometry of each vertebra was rotated so that the upper and lower connection points were in alignment along Axis 3 (Figure 2-4).

Additionally, Franklin defined the origin of the vertebra as the inferior connection point of the vertebra, and all body coordinates were translated to the local coordinate system of the vertebra from the global coordinate system[24]. The origin of the pelvis was defined at (0,0,0) meters.

2.1.3 Mass and Mass Moment of Inertia

Lui *et al.* determined the mass of each vertebra of a cadaver spine[63]. The lumbar vertebrae resulted in an average mass of approximately 2.0 kg. The cervical and thoracic region resulted in a total mass of approximately 15.3 kg. A mass of 16 kg was used for the cervical/thoracic region of the spine and a mass of 2 kg was used for each lumbar vertebra (Table 2-1).

In our model, each vertebra's center of mass was located at the center of the vertebral spacing for each vertebra (Figure 2-5).

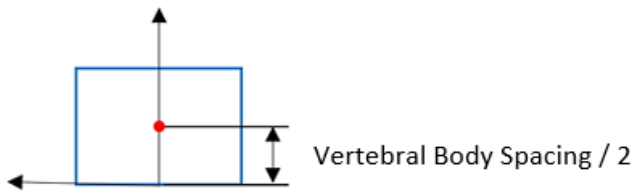


Figure 2-5: Front view of vertebra. Center of mass depicted by red circle

The mass moment of inertia for an elliptical cylinder can be determined using the equations in Appendix A.1. The vertebral spacing (Table 2-1) was used in the mass moment of inertia calculation.

2.1.3.1 Modifications to the 2006 Franklin Model

The cervical and thoracic spine region mass has been updated to 16 kg from a mass of 20 kg previously defined in the model by Franklin *et al.*[24]

2.2 Rotations

The basis vectors depicted in Figure 2-6 are the basis vectors affixed to the ground (*N* basis). Each vertebra is rotated first around Axis 1, followed by Axis 2, and lastly around Axis 3. This was accomplished using rotation matrices. Matrix multiplication can be used to develop a rotation matrix encompassing the rotation around all three axes:

$$^{N}R^{C} = {}^{N}R^{A} * {}^{A}R^{B} * {}^{B}R^{C} \quad (2\text{-}2)$$

This rotation matrix ($^{N}R^{C}$) defines the angular relationship between bases *N* and *C*. See Appendix A.4 for more information regarding the development of the rotation matrix.

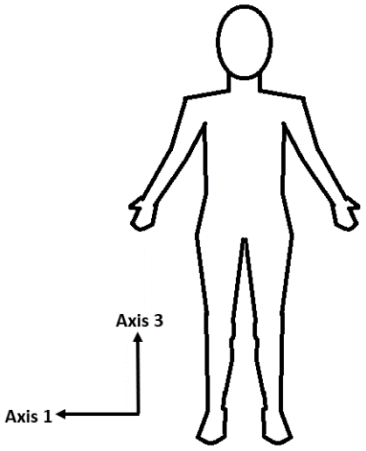


Figure 2-6: N basis vectors affixed to the ground. Axis 2 points anteriorly

2.2.1 Lordosis Angles

Lumbar lordosis is the curve of the lumbar region of the spine. The lordosis angle of each vertebra is applied as a rotation around Axis 1 where a counterclockwise rotation is defined as the positive direction. See Appendix A.2 for more information about the Lordosis Angles applied.

2.2.1.1 Modifications to the 2006 Franklin Model

The lordosis angles have been updated based on the average segmental lordosis angles determined by Pesenti *et al*. In this study, the radiographic data was collected for 119 adult healthy controls with no history of back pain or injury[61]. The updated lordosis angles for each vertebra are recorded in Table 2-1.

2.3 Intervertebral Discs

Franklin *et al.* developed a lumped parameter model for the modelling of the intervertebral discs[24] with parameters based on the work of Stokes *et al.*[76] and Izambert *et al*.[77]

Stokes *et al.* measured the rotational stiffness of cadaveric discs[76] and Izambert *et al.* determined that the damping coefficient was approximately 2 orders of magnitude less than the stiffness coefficient[77]. This resulted in a rotational stiffness (K_{IVD}) of 50 Nm/rad and a rotation damping (B_{IVD}) of 0.5 Nms/rad, which was applied to all three axes of the intervertebral disc model.

By determining the angular position and angular velocity, the magnitude of the moment generated by the intervertebral discs can be determined (Equation 2-3) and applied as equal and opposite moments to the vertebrae.

$$M = K_{IVD} * \theta + \ B_{IVD} * \dot{\theta} \tag{2-3}$$

For this calculation, the angles of rotation need to be determined, which can be achieved through use of a dummy plane[24]; the body below the vertebra being measured will act as the dummy plane. Franklin *et al*.[24] defined planar rotation to be the rotation of the body that causes the plane and body to not be parallel (Figure 2-7) and twist rotation to be the rotation of the body that causes the plane and body to not be aligned (Figure 2-8).

To determine these angles, the dummy plane must first be rotated to have the same lordosis angle as the vertebra above it. The difference in lordosis angles should not contribute to our angle measures. Once the plane and vertebra have the same lordosis angles, we can determine if the planes are parallel or if there is a planar rotation present. If the planes are parallel, Axis 3 of the vertebrae and plane will be aligned. If Axis 3 is not aligned, then the planar angle can be calculated by determining the angle between Axis 3 of the vertebra and the plane. Using the Rodrigues Rotation Formula[78], the vertebra is rotated so that it is parallel to the dummy plane.

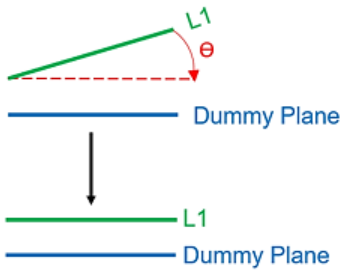


Figure 2-7: Planar rotation

Next, we can determine the twist rotation angle. If Axis 1 is aligned for the vertebra and dummy plane, then a twist rotation is not present. If Axis 1 is not aligned, then the rotation angle can be calculated by determining the angle between Axis 1 of the vertebra and the plane. Using the Rodrigues Rotation Formula[78], the vertebra is rotated. The vertebra should now be aligned with the dummy plane.

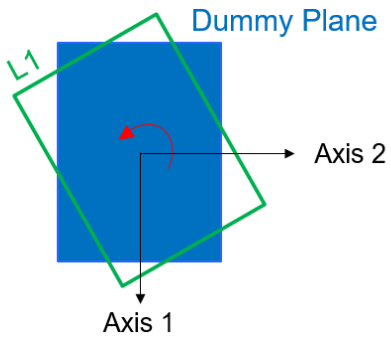


Figure 2-8: Twist rotation

2.4 Muscle Anatomy

This model incorporates 90 muscle fascicles with the attachment points detailed in Appendix A of Cholewicki *et al*.[15] Included are the origin, insertion and nodal coordinate points with respect to the skeletal geometry. The muscle coordinates have been transformed into body-fixed coordinates. The adjusted muscle coordinates and the process used to adjust the coordinates is further detailed in Appendix A-3.

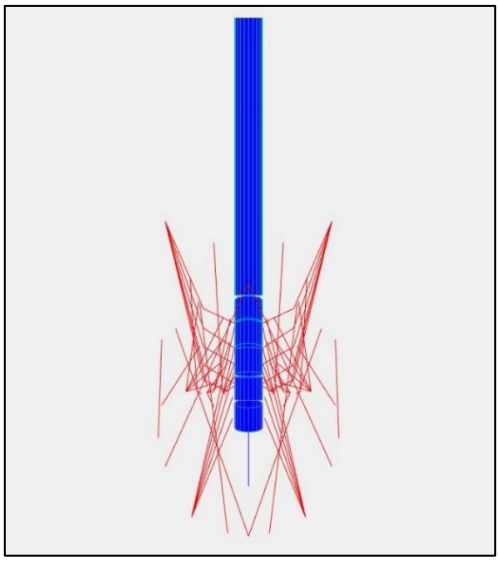


Figure 2-9: Muscle Anatomy

While many models only include muscles with straight lines of action, meaning muscles connecting from the origin directly to the insertion, the nodal points in our muscle anatomy allow for the muscles to have a more curved path as they would in the body[15] (Figure 2-9).

2.4.1 Calculating Transformed Muscle Anatomy

The transformed muscle anatomy can be determined through use of the rotation matrices and origins of the bodies. Each muscle attachment point will be rotated based on the body it is associated with and that body's origin:

$$pt_{transformed} = {}^{N}R^{C} * pt + origin \quad (2\text{-}4)$$

where ${}^{N}R^{C}$ is the rotation matrix from the *N* to *C* bases, and the coordinates of the muscle attachment point (*pt*) and origin of the body (*origin*) need to be column vectors.

2.4.2 Calculating Muscle Length

Muscle length should be calculated after the bodies have been rotated, so that lumbar lordosis has been implemented. Vectors can be determined between a muscle's attachment points (Figure 2-10, Equation 2-5):

Figure 2-10: Muscle attachment points visualization

$$\vec{p} = ma_{pt1} - ma_{pt2} = p_1\hat{c_1} + p_1\hat{c_2} + p_1\hat{c_3} \quad (2\text{-}5)$$

where ma_{pt1} is the first attachment point in the set and ma_{pt2} is the second attachment point.

Using resulting vector $\vec{p}$, the length of the vector can be calculated:

$$|\vec{p}| = \sqrt{p_1^2 + p_2^2 + p_3^2} = length\ of\ vector \quad (2\text{-}6)$$

Lastly, the total muscle length can be determined through the summation of the vector lengths between the muscle attachment points.

2.4.3 **Calculating Muscle Velocity**

In order to calculate the muscle velocity, we must calculate the time-derivative of the rotation matrix[79]:

$$\frac{d\,{}^{N}R^{C}}{dt} = \frac{\partial\,{}^{N}R^{C}}{\partial q_1}\frac{dq_1}{dt} + \frac{\partial\,{}^{N}R^{C}}{\partial q_2}\frac{dq_2}{dt} + \frac{\partial\,{}^{N}R^{C}}{\partial q_3}\frac{dq_3}{dt} \tag{2-7}$$

Where q_1, q_2, and q_3 are the angles of rotation around Axis 1, Axis 2 and Axis 3, respectively, and ${}^{N}R^{C}$ is the rotation matrix for each body that incorporates all three body rotations.

For example, for a muscle fascicle with only two attachment points, the following can be used to determine the linear velocity of a muscle attachment point[79]:

$$\dot{coord} = \left(\frac{d\,{}^{N}R^{C}}{dt} * coord + dOr\right) \tag{2-8}$$

Where $\frac{d\,{}^{N}R^{C}}{dt}$ is the time-derivative of the rotation matrix, *coord* are the muscle attachment point coordinates and *dOr* is the time-derivative of the origin of the body.

A muscle point velocity vector can then be determined between the two muscle attachment points using Equation 2-9:

$$\vec{v} = \left(\frac{d\,{}^{N}R^{C}}{dt} * coord_{origin} + dOr\right) - \left(\frac{d\,{}^{N}R^{C}}{dt} * coord_{terminal} + dOr\right) \tag{2-9}$$

Where $coord_{origin}$ is the origin attachment point and $coord_{terminal}$ is the terminal attachment point for the muscle fascicle.

The position vector is defined in Equation 2-10 as the vector between the attachment points:

$$\vec{p} = {}^{N}R^{C} * coord_{origin} - {}^{N}R^{C} * coord_{terminal} = \langle p_1, p_2, p_3 \rangle \tag{2-10}$$

Finding the unit vector of the position vector:

$$\hat{p} = \frac{\vec{p}}{|\vec{p}|} = \frac{\vec{p}}{\sqrt{{p_1}^2 + {p_2}^2 + {p_3}^2}} \tag{2-11}$$

Lastly, by performing the dot product, we can determine the projection of $\vec{v}$ in the direction of $\hat{p}$:

$$muscle\ velocity = \vec{v} \cdot \hat{p} \tag{2-12}$$

The resulting scalar is the magnitude of the muscle velocity. If the muscle path includes nodes, the velocity can be determined between muscle attachment points along the path of the muscle and summed to yield the total muscle velocity.

2.5 Muscle Model

Franklin *et al.*[24] incorporated a Hill-Type muscle model[80] that includes a contractile element, spring and damper, all in parallel (Figure 2-11):

$$F_m = F_{CE} + K_{PE}x + B_{DE}\dot{x} \tag{2-13}$$

where K_{PE} is stiffness, B_{DE} is the damping coefficient, x is muscle length, and $\dot{x}$ is stretch rate.

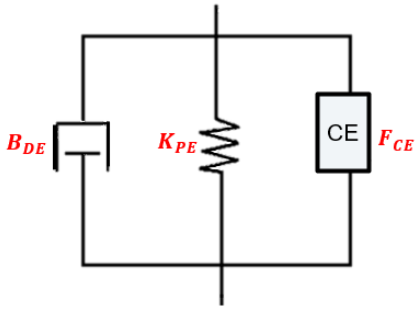


Figure 2-11: Hill-type muscle model

In the Hill muscle model, the contractile element is representative of the actin and myosin cross-bridges, the parallel elastic is representative of the connective tissue properties, and the series elastic, which has not been included in this work, is representative of the tendon properties[81,82].

The force of the contractile element was determined by scaling the maximum muscle force by muscle activation[24] (Equation 2-15). Muscle activation ranges between zero and one for all muscles, with a magnitude of one indicating a fully activated muscle and a magnitude of zero indicating an un-activated muscle. Maximum muscle force was determined by the product of maximum muscle stress (46 N/cm^2) by the muscle cross-sectional area[23] (Equation 2-14).

$$f_{max} = \left(46\,{}^{N}/_{cm^2}\right) * CSA \tag{2-14}$$

$$F_{_{CE}} = f_{max} * \alpha \tag{2-15}$$

The parallel elastic has a stiffness (K_{PE}) based on Bergmark's model of short-range stiffness where q is the stiffness gain[18]:

$$K_{PE} = q * \frac{F_{CE}}{x_o} \tag{2-16}$$

Similarly, the muscle damping was scaled by the damping gain b:

$$B_{DE} = b * \frac{F_{CE}}{x_o} \tag{2-17}$$

By substituting in the stiffness and damping coefficients, the equation for the muscle model is:

$$F_m = f_{max} * \alpha + \left(\frac{q * f_{max} * \alpha}{x_o} x(t) - x_o\right) + \left(\frac{b * f_{max} * \alpha}{x_o} \dot{x}(t)\right) \quad (2\text{-}18)$$

where f_{max} is the maximum muscle force, q is stiffness gain, b is damping gain, x is muscle length, x_o is equilibrium muscle length, $\dot{x}$ is stretch rate, and α is muscle activation.

2.6 Reflex Model

In simplistic terms, the muscle stretch reflex is the concept in which a stimulus causes the muscle to lengthen and the muscle spindles respond to this lengthening by opposing this stretch through contraction[83,84]. The muscle spindles have the ability to determine the velocity and length of the muscle, and through communication with the nervous system, oppose motion of the muscle[83,84].

This muscle stretch reflex mechanism can be represented through the application of a PD controller. The implementation of a PD controller in our system allows for the system to be responsive to the change in muscle length and velocity, like the muscle stretch reflex, creating a feedback control system[24,85]. Franklin *et al.*[24] defined muscle activation to be the sum of the steady-state activation α_o and the reflex activation α_r (Equation 2-19).

$$\alpha(t) = \alpha_o + \alpha_r = \alpha_o + \alpha_o * \left(G_P * \frac{x(t-\tau) - x_o}{x_o} + G_D * \frac{\dot{x}(t-\tau)}{x_o}\right) \quad (2\text{-}19)$$

where G_P is the proportional gain and G_D is the differential gain of the PD controller.

2.7 Dynamics

The derivative of the Lagrangian (*L*) was used to determine the equations of motion of the system[24]. The Lagrangian function is based on the difference between the system's kinetic (*KE*) and potential energy (*PE*)[86]:

$$L = KE - PE \tag{2-20}$$

Franklin *et al.* only included gravitational energy in the potential energy of the system[24], so the general equation for the potential energy of the system is as follows:

$$PE = \sum_{b=1}^{6} mgh \tag{2-21}$$

where *h* is the position of the center of mass of the body along Axis 3, *m* is the mass of the vertebra, and *b* is the body index.

The kinetic energy of the system includes rotational and translational kinetic energy:

$$KE = \sum_{b=1}^{6} \left(\frac{1}{2}mv^2 + \frac{1}{2}I\omega^2\right) \tag{2-22}$$

where *I* is the mass moment of inertia, and *v* is the velocity and ω is the angular velocity of the center of mass of the body.

The muscle forces, external forces and the IVD moments are utilized in the generalized force calculations (Equation 2-23).

$$Q_i = \sum_{b=1}^{6} \left(\left(\sum_{e=1}^{n} F_{be} \right) \cdot \frac{\partial v_b}{\partial \dot{q}_i} + \left(\sum_{e=1}^{n} r_{be} \times F_{be} \right) \cdot \frac{\partial \omega_b}{\partial \dot{q}_i} \right) \tag{2-23}$$

where *i* is the degree of freedom index for the degree of freedom q_i, *b* is the rigid body index, *e* is the applied forces index, F_{be} represents force applied to the body, v_b is the linear velocity of the body's origin, r_{be} is a vector between the point where the force is applied and the body's origin, and ω_b is the body's angular velocity.

The Lagrangian Derivative yields the equations of motion (Equation 2-24):

$$EOM = \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \frac{\partial L}{\partial q_i} = Q_i \tag{2-24}$$

where Q_i are the generalized forces and q_i are the generalized coordinates related to the degrees of freedom.

By substituting in Equation 2-20 into Equation 2-24, the following equation is derived:

$$EOM = \frac{d}{dt}\left(\frac{\partial KE}{\partial \dot{q}_i}\right) - \frac{d}{dt}\left(\frac{\partial PE}{\partial \dot{q}_i}\right) - \frac{\partial KE}{\partial q_i} + \frac{\partial PE}{\partial q_i} = Q_i \tag{2-25}$$

Lastly, potential energy of the mechanical system is not a function of velocity[87], therefore the second term in Equation 2-25 can be eliminated, resulting in our final equation for the generalized forces calculation:

$$EOM = \frac{d}{dt}\left(\frac{\partial KE}{\partial \dot{q}_i}\right) - \frac{\partial KE}{\partial q_i} + \frac{\partial PE}{\partial q_i} = Q_i \tag{2-26}$$

The dynamic equations can be used to determine the accelerations $\ddot{q}$ through forward dynamic simulation techniques (Equation 2-27):

$$M_m(q) \cdot \ddot{q} + G_m(q) + C_m(q, \dot{q}) = Q_i \tag{2-27}$$

Where M_m is the mass matrix, C_m is the Coriolis vector, and G_m is the gravity vector.

2.8 Linearization

The linearized representation of the system was determined through Taylor Series expansion. For a nonlinear system to be linearized, it must be in equilibrium.

Equation 2-28 is the state vector and Equation 2-29 is the differential state vector:

$$x = \left[\theta_1, \theta_2, \dots \theta_n, \dot{\theta}_1, \dot{\theta}_1, \dots, \dot{\theta}_n\right] \tag{2-28}$$

$$\dot{x} = f(x) \tag{2-29}$$

Where $\dot{x} = 0$ when at the equilibrium point ($x = \bar{x}$).

Each state variable was disturbed by angle δ_θ individually (Equation 2-30):

$$\dot{\delta_\theta} = f(\theta_1 + \delta_\theta, \theta_2, \dots \theta_n, \dot{\theta}_1, \dot{\theta}_1, \dots, \dot{\theta}_n) \tag{2-30}$$

The Taylor Series Expansion is performed, neglecting higher order terms:

$$\dot{\delta_\theta} = f(\bar{x}) + \frac{\partial f(x)}{\partial x}\Big|_{\mathrm{x}=\bar{\mathrm{x}}} \delta_\theta \tag{2-31}$$

At equilibrium, $f(\bar{x}) = 0$.

$$\dot{\delta_\theta} = \frac{\partial f(x)}{\partial x}\Big|_{\mathrm{x}=\bar{\mathrm{x}}} \delta_\theta \tag{2-32}$$

The $\frac{\partial f(x)}{\partial x}\Big|_{x=\bar{x}}$ term in Equation 2-32 is the Jacobian matrix. The Jacobian matrix can be obtained through finite differencing (Equation 2-33), which results in an approximate result of a derivative.

$$\frac{\partial \ddot{\theta_1}}{\partial \theta_1} = \frac{f\left(\overline{\theta_1} + \delta_\theta, \overline{\theta_2}, \ldots \overline{\theta_n}, \dot{\overline{\theta_1}}, \dot{\overline{\theta_2}}, \ldots, \dot{\overline{\theta_n}}\right) - f(\bar{x})}{\delta_\theta} \tag{2-33}$$

Neuromuscular reflexes are time-delayed, requiring for there to be a Jacobian for the instantaneous components and a separate Jacobian for the delayed components (Equation 2-34):

$$\dot{x}(t) = J_i \cdot x(t) + J_d \cdot x(t - \tau) \tag{2-34}$$

While the separation of these Jacobians is not necessary for the optimization procedure (Section 2.9), it will be necessary for the Linear Time Delay analysis (Section 2.10).

2.9 Optimization Procedure

A constrained optimization is performed to determine the muscle activation that meets the following requirements:

1. The system must be stable and in equilibrium.
2. Muscle activation magnitude must be between 0 and 1 for all muscles.
3. Metabolic power must be minimized.

2.9.1.1 Stability Requirement

The stability of the system is evaluated using the Jacobian matrix. A system is considered stable if the eigenvalues of its Jacobian have negative real parts[24,79].

2.9.1.2 Equilibrium Requirement

For a system in equilibrium, acceleration will have a magnitude of zero. For this criterion to be met, the acceleration due to the passive state equilibrium properties (*beq*) and the product of the active state equilibrium properties (*Aeq*) and the muscle activation (α) will need to be equivalent:

$$Aeq * \alpha = beq \tag{2-35}$$

This equilibrium condition will be utilized in the optimization procedure to ensure that the system is in equilibrium.

2.9.1.3 Muscle Activation Range

The magnitude of the muscle activation for each muscle must fall between 0 and 1.

2.9.1.4 Minimizing Metabolic Power

Anderson[88] defines the maintenance heat rate by the following equation, assuming a proportionate quantity of fast and slow twitch muscle fibers[24] (Equation 2-36):

$$P = m * \left(74 * 0.5 * \sin\left(\frac{\alpha * \pi}{2}\right) + 111 * 0.5 * \left(1 - \cos\frac{\alpha * \pi}{2}\right)\right) \tag{2-36}$$

The metabolic power can be minimized through use of the built-in function *fmincon* in MATLAB. This function performs a constrained optimization, which allows for the previously mentioned requirements to be met while solving for the muscle activations. The metabolic power is specified as the parameter that needs to be minimized, while the equilibrium condition and stability condition are specified as constraints. Additionally, the range for the muscle activation, which is between 0 and 1, is specified as a constraint for the Solver. A randomized muscle activation is input into the function, providing a starting point for the Solver. A minimum of five trials were completed, meaning that at least five different starting muscle activation vectors were utilized. Performing multiple trials with different initial activations increased the likelihood of the global minima being found[24].

2.10 Linear Time Delay Methods

The linear time delay methods are based on the work of Chen *et al.*[89] In this work, a procedure is outlined that allows for the time delay that would cause instability of the system to be determined.

For the optimization procedure, it is assumed that time delay has a magnitude of zero ($\tau = 0$), so the Jacobian contains the instantaneous and delayed components. After the optimization procedure has been completed, the Jacobian can be separated into the instantaneous and delayed components (Equation 2-34).

Based on Chen's method[89] involving frequency sweeping in the Laplace domain (Equation 2-37), the time delay in which the system becomes unstable can be determined, which would be when the real part of an eigenvalue of the Jacobian crosses the Imaginary Axis[24]. This is indicated by $\lambda = 1$ (Equation 2-38):

$$j\omega \cdot X = J_i \cdot X + J_D \cdot X \cdot e^{-j\omega\tau} \tag{2-37}$$

$$(j\omega \cdot I - J_i) \cdot X = \lambda \cdot J_D \cdot X, \qquad where\ \lambda = e^{-j\omega\tau} = e^{-j\theta} \tag{2-38}$$

Using the corresponding phase θ and the frequency ω, the delay margin τ_m, which is the time delay magnitude as to when the real part of the eigenvalue crosses the imaginary axis, can be computed:

$$\tau_m = \frac{\theta}{\omega} \tag{2-39}$$

Using the delay margin, the time delay that will be utilized in the nonlinear simulation (section 2.11) can be computed:

$$\tau = \tau_m * 0.99 \tag{2-40}$$

This time delay τ represents a time delay preceding the delay margin/the system becoming unstable. A time delay with a magnitude less than the delay margin should result in a stable system if the linear analysis adequately predicted the nonlinear results. Lastly, if the delay margin is less than 0.06 seconds, then the system is considered unstable on the basis that the neuromuscular reflexes cannot act faster than 0.06 seconds, indicating that they would not be able to contribute to stability at that time[7].

2.11 Nonlinear Verification

A nonlinear verification technique is used to ensure that the linear approximation appropriately represented the nonlinear system and successfully predicted its stability conditions. Using the *dde23* MATLAB Solver, the delayed differential equation with a constant delay could be solved. Based on the parameters defined by Franklin *et al.*, each simulation is run for five seconds[17]. Following a kinematic disturbance, the system should approach equilibrium; this behavior can be investigated using the normalized state space about equilibrium[17].

To determine if the system is approaching equilibrium during the nonlinear simulation, the difference between the normalized state space and equilibrium can be determined:

$$D = \sqrt{\sum_{i=1}^{18} \left(\left(\frac{\dot{\theta}(t) - \dot{\bar{\theta}}}{mean(\dot{\theta}(t) - \dot{\bar{\theta}})} \right)^2 + \left(\frac{\theta(t) - \bar{\theta}}{mean(\theta(t) - \bar{\theta})} \right)^2 \right)} \tag{2-41}$$

2.12 Summary

The current methodology and model are based on those developed by Franklin *et al.*[16,17,24]. This rigid body model with a fixed pelvis allows for the behavior of the lumbar spine to be explored. Using linear stability analysis in the optimization procedure, the muscle activations and required metabolic power can be determined while the equilibrium, stability and muscle activation constraints are met. Following this optimization procedure, the time delay analysis can be performed, resulting in the magnitude of the time delay that will be utilized in the nonlinear verification procedure. This model allows for the investigation of low back stability and the factors that may affect it.

Chapter 3: Code Verification

Throughout the development of the Franklin *et al.* model,[16,17,24] it was crucial that the methodology and implementation of this methodology be reviewed. As mentioned previously, based on the published code in the Appendix of Franklin's thesis[24] and the code received from the Musculoskeletal Biomechanics Laboratory at Virginia Polytechnic Institute (Michael Madigan, personal communication, July 26, 2008), it was evident that the Human Motion Control Lab did not possess the final version of the model. Through use of the materials and the published information, the model was developed. While adjustments have been made to the model, the methodology is based on Franklin's works[16,17,24] so similarities should be expected.

Verification tasks were completed to identify and eliminate potential errors in the codes[67]. The verification tasks for the main model codes can be seen in Table 3-1. The file *For_verification.m* is used to perform some of the verification tasks.

Table 3-1: Verification Tasks and Results

Purpose	Variable Output (Current model/Franklin)	Verification Tasks	Successfully Verified?
1. Define Model Parameters *model_parameters.m*	L0, L1, L2, L3, L4, L5, la, lb, m0, m1, m2, m3, m4, m5, g, cent1, cent2, cent3, IKp, IDp, IKt, IDt, IVD, Lord_Angs/…,AAPOA	1.1) Compare variables in Workspace to desired model parameters.	✓
2. Import Muscle File *transform_Cholewicki.mlx* *load_musclefile.m*	M/M	2.1) Compare Cholewicki muscle coordinates to the current model's muscle coordinates. This will be performed to ensure that data entry errors have not occurred in our Excel spreadsheet and that our MATLAB function imported/read the Excel file appropriately. Note: This comparison should be made prior to the	✓

musclefile_Cholewicki.xlsx		transformation aligning all vertebrae along Axis 3 and the application of lordosis angles.	
		2.2) Compare current model's muscle coordinates to Franklin's muscle coordinates. This will allow for the comparison between our Excel spreadsheet and Franklin's spreadsheet describing the muscle coordinates and paths. Additionally, this will be performed to ensure that our MATLAB function substituted the numeric coordinates on Sheet 2 of the Excel spreadsheet to the appropriate muscle attachment point name on Sheet 1 of the Excel spreadsheet. Note: This comparison should be made after the transformation is applied so that all vertebrae are aligned along Axis 3, but the lordosis angles should NOT be applied.	X
		2.3) Compare Cholewicki's muscle paths to the current model's muscle paths. Compare the muscle paths described on Sheet 1 of our Excel spreadsheet to the muscle paths described in Table A1 (Cholewicki *et al.*[54])	✓
		2.4) Compare the current model's muscle paths to Franklin's muscle paths. Compare the muscle paths described on Sheet 1 of our Excel spreadsheet and on Sheet 1 of Franklin's Excel spreadsheet to compare the muscle path of each muscle.	X
		2.5 Compare output of the current model's *load_musclefile.m* and Franklin's *muscinput.m* file Note: the Excel spreadsheet utilized by the functions should be the same to ensure constant inputs to the functions.	✓
3. Import External Force File *load_forcefile.m*	E/E	3.1) Manually check that the correct coordinates have been substituted for the assigned attachment points specified on the LoadLift.xls Excel sheet.	✓
		3.2) Compare outputs from Franklin's *forcinput* and *format* functions and the current model's *load_forcefile* function	✓
4. Determine Rotation Matrices *rotationmatrices.mlx*	rotation_matrices, d_rotation_matrices/Rm, dRm	4.1) Compare output from Franklin's *analys1.mex64* file and rotationmatrices.mlx	✓

5. Determine Body Origins *origin_body.mlx*	origin, d_origin/Or, dOr	5.1) Compare output from Franklin's analys1 MEX file and origin_body.mlx	✓
6. Transform Anatomy *rotate_anatomy.m*	Ma/Ma	6.1) Compare output from Franklin's analys1 MEX file and rotate_anatomy_sym.mlx Note: In order to compare the transformed Ma, the same input M needs to be used for both functions/files.	✓
7. Muscle Properties *rotate_anatomy_sym.mlx*	ML, MV/ ML, MV	7.1) Compare output from Franklin's *analys1.mex64* file and the *rotate_anatomy_sym.mlx* script	✓
		7.2) Perform hand calculations to verify the methodology implemented in the *rotate_anatomy_sym.mlx* script used to calculate muscle lengths and velocities	✓
8. Transforming the External Force *rotateforce.mlx*	Ea/Ea	8.1) Compare output from Franklin's analys1 MEX file and rotateforce.mlx	✓
		8.2) Compare output *Ea* from hand calculations and the current model's *rotateforce.mlx*	✓
9. Dynamics *dynam.mlx*	Cm/Cv	9.1) Compare output from Franklin's *dynmatsc* MEX file and *dynam.mlx*	X
		9.3a) Compare output from Franklin's Mathematica code and *dynam.mlx*	✓
	Gm/Gv Mm/Mv	9.2) Compare output from Franklin's *dynmatsc* MEX file and *dynam.mlx* 9.3b) Compare output from Franklin's Mathematica code and *dynam.mlx*	✓ ✓
	Gl/Gl Gr/Gr	9.4) Compare the "helper variables" from Franklin's *analys1.mex64* and *dynam.mlx*	✓

10. Intervertebral Disc Moment *IVD_calcs.mlx*	PRQ/PRQ	10.1) Compare output from Franklin's i*ntervertc* MEX file to the current model's *IVD_calcs.mlx* file	X
		10.2) Ensure that the body and plane were parallel after planar and twist rotations were applied to the plane.	✓
11. Generalized Forces *generalizedForce.mlx*	Q/Q	11.1) Compare output from Franklin's *geneforcesc* MEX file and the current model's *generalizedForce* function	✓
12. Optimization Procedure *optimize.m* *nlcon.mlx* *costfunct.mlx* *run_fmincon.mlx*	A B Aeq beq	12.1) Compare constraint variables used in *fmincon*/developed during the optimization procedure in the current model to Franklin's outputs Note: this procedure will verify the use of the spineaccA/AccelerationMA and spineaccP/AccelerationMP functions as they are used to determine Aeq and beq	✓
	Un_opt	12.2) Ensure that all muscle activations fall within the acceptable range between zero and one	✓
	qddP, qddA	12.3) Compare the angular accelerations for the current model's *spineaccP* and *spineaccA* functions with Franklin's *AccelerationMP* and *AccelerationMA* functions	✓
	Ja	12.4) Compare the Jacobians *Ja* and *Jp* from Franklin's *alg_optimize.m* and the current model's *optimize.m*	✓
		12.5) After the constrained optimization procedure is completed, ensure that the solution adheres to the stability requirement	✓
	P	12.6) Compare the output of the current model's *costfunct* function and Franklin's *costfun* function	✓
13. Muscle Model *musclemodel.mlx*	MF, dA	13.1) Compare Franklin's *muscmodel.m* output to the current model's *musclemodel.mlx* output	✓
14. Reflex Model *musclemodelR.mlx*	MFR, dAr	14.1) Perform hand calculations to verify the output of the current model's *musclemodelR* function	✓

15. Time Delay Analysis *tdinvest.m* *tdfind.m* *tdfmax.m*		15.1) Perform nonlinear simulation with delay margin. The linear system should successfully predict a delay margin that will result in a stable system. (Delay that is simulated should still be delay margin * 0.99)	✓
16. Nonlinear Verification *spine_dyn.mlx*		16.1) Calculate the distance of the normalized state variables from equilibrium *Dss* and determine if the system is approaching equilibrium.	✓

The verification procedure *For_verification.m* can be initialized in the *spine_main.m* file (Figure 3-1).

```
clear all; close all; clc;
%% Verification
% Create Excel File with verification outputs.
 global date
 date=datetime('now');
 verification_logical=1; %1 indicates that verification should be performed
 save('verification_initialize_vars.mat','date','verification_logical')

 if verification_logical==1
    For_verification %has clc and clear commands in file
    clear; clc
end
```

Figure 3-1: Initializing the verification procedure in spine_main.m file

For the verification procedure, the *Stateo_Pango_Rand.mat* file was used to load the Stateo and Pango vectors that were utilized as inputs of functions. These Stateo and Pango vectors were randomly generated, with each value in the vector ranging between zero and one, with all values being nonzero. The *Model_Parameters_Verify.mat* file was also used for the verification procedure. The variables *cent1*

and *cent2* needed to have a magnitude of zero for comparison between Franklin's output and our output, but all other values were set to be nonzero.

The absolute error calculated between our outputs and Franklin's were calculated in MATLAB (Equation 3-1):

$$Absolute\ Error = |Jardon\ output - Franklin\ output| \tag{3-1}$$

The verification tasks completed in the *For_verification.m* file are exported to an Excel spreadsheet (Figure 3-2). In some instances, for output matrices with large dimensions, the greatest magnitude of absolute error determined in the verification task was solely presented for simplicity, instead of the absolute error of all matrix indices.

```
%% Exporting Verification to Excel Spreadsheet
 global date
 warning('off','MATLAB:xlswrite:AddSheet');

 filename=sprintf('Verification Summary %s.xlsx',date);
 filename=replace(filename,":","."); %cannot have colons in file titles
 filename=string(append('C:\Users\Valerie
Jardon\Documents\Model\FINALIZED_CODE_updated\Simulation Runs Verification
Output\',filename));
 writematrix(date,filename,'Sheet',1,'Range','A1')

 writecell({'Rotation Matrices'},filename,'Sheet',2,'Range','C3')
 writematrix(Rm_diff,filename,'Sheet',2,'Range','C4')
```

Figure 3-2: Example of exporting verification task results to Excel spreadsheet

3.1 Model Parameters

For the verification of the model parameters, the stored variables in the MATLAB workspace were compared to the intended inputs to ensure input error did not occur. All model parameters defined in the *model_parameters.m* file were accurately drafted in MATLAB and stored correctly in the MATLAB workspace (Table 3-1, Verification Task 1.1).

3.2 Initial Transformation of Cholewicki *et al.* Muscle and Skeletal Coordinates

3.2.1 Muscle and Skeletal Coordinates Verification

Franklin *et al.*[16,17,24] uses the Cholewicki *et al.*[54] muscle geometry and skeletal coordinates. These coordinates are for the spine in neutral position, indicating that lumbar lordosis is present and impacting the coordinate positions, but no additional flexion, extension or bending of the spine is present.

Franklin specified the segmental lordosis angles of the vertebrae in the appendix of his thesis.[24] These lordosis angles differ from those incorporated into the Cholewicki *et al.* muscle/skeletal coordinates[15], so each vertebra needs to be rotated to "remove" the default lumbar lordosis (process shown in Figure 3-3). This will result in the alignment of the connection points of the vertebra along the vertical axis, indicating that the vertebral faces are parallel.

Franklin provides the body fixed coordinates in the Appendix of his thesis[24]. For these coordinates, the vertebrae have been rotated so that their connection points align along the vertical axis. The origin was subtracted from each point associated with that body, so that the coordinates would be described by their relative location from the origin.

There are three skeletal coordinates that differed between Franklin's muscle Excel file *CholeMusc.xls* and current model's muscle Excel file *CholeMusc_update.xls*. Upon further investigation, it appears that Franklin may have had data entry errors for points *RIB10*, *RIB21* and *RIB22* (Table 3-1, Verification Task 2.2).

The unadjusted global coordinates from Cholewicki *et al.*[15], following our coordinate system convention, are as follows:

$$RIB1 = [0, 7.2, 35.5] \quad \text{%global origin}$$

$$RIB10 = [2, (2 - 7.2), (53.5 - 35.5)] = [2, -5.2\ ,18]$$

$$RIB21 = [6.5, (3.6 - 7.2), (26.4 - 35.5)] = [6.5, -3.6, -9.1]$$

$$RIB22 = [2, (3.6 - 7.2), (24.2 - 35.5)] = [6.5, -3.6\ , -11.3]$$

By subtracting the origin of the body from these coordinates, the body-fixed coordinates can be determined:

$$RIB10 = [2, (2 - 7.2), (53.5 - 35.5)] = [2, -5.2\ ,18]$$

$$RIB21 = [6.5, (3.6 - 7.2), (26.4 - 35.5)] = [6.5, -3.6, -9.1]$$

$$RIB22 = [2, (3.6 - 7.2), (24.2 - 35.5)] = [6.5, -3.6\ , -11.3]$$

As shown in Figure 3-3, the coordinates can then be rotated to align along Axis 3, yielding:

$$RIB10 = [2, -5.36\ ,17.95]$$

$$RIB21 = [6.5, -3.51, -9.13]$$

$$RIB22 = [6.5, -3.49\ , -11.33]$$

```
clear; clc
syms q1 q2 q3
R1=[1 0 0; 0 cos(q1) -sin(q1); 0 sin(q1) cos(q1)]; %Rotation matrix for a rotation around axis 1; + CCW rotation
R2=[cos(q2) 0 sin(q2); 0 1 0; -sin(q2) 0 cos(q2)]; %Rotation matrix for a rotation around axis 2; + CCW rotation
R3=[cos(q3) -sin(q3) 0; sin(q3) cos(q3) 0; 0 0 1]; %Rotation matrix for a rotation around axis 3; + CCW rotation
R321=R3*R2*R1;

RIB10=[2 2-7.2 53.5-35.5]'; %[2 -5.2 18]
RIB21=[6.5 3.6-7.2 26.4-35.5]'; %[6.5 -3.6 -9.1]
RIB22=[6.5 3.6-7.2 24.2-35.5]'; % [6.5 -3.6 -11.3]

% RIB10=[2 -5.2 19]'; %coordinate Franklin uses
% RIB21=[6.5 -3.6 0.9]'; %coordinates Franklin uses
% RIB22=[6.5 -3.6 -1.3]'; %coordinates Franklin uses

R321=subs(R321,[q1,q2,q3],[deg2rad(0.528868999999543),0,0]);

pt_RIB10=double(R321*RIB10)
```

```
pt_RIB10 = 3×1
    2.0000
   -5.3659
   17.9512
```

```
pt_RIB21=double(R321*RIB21)
```

```
pt_RIB21 = 3×1
    6.5000
   -3.5159
   -9.1328
```

```
pt_RIB22=double(R321*RIB22)
```

```
pt_RIB22 = 3×1
    6.5000
   -3.4955
  -11.3327
```

Figure 3-3: "Removing" the default lordosis angles: Transforming coordinates to align the connection points of the vertebrae along Axis 3.

Next, using the coordinates Franklin provides in the appendix of his thesis[24] (recall that these coordinates are descriptive of when the vertebral connection points are aligned along the vertical axis), we can determine that these would have been the non-transformed coordinate points he would have sourced from Cholewicki and McGill[15]:

$$RIB10 = [2, -5.2\ , 19]$$

$$RIB21 = [6.5, -3.6, 0.9]$$

$$RIB22 = [6.5, -3.6\ , -1.3]$$

Figure 3-4 demonstrates that the transformation of these coordinate points would result in the coordinates provided in Franklin's thesis.[24]

```
clear; clc
syms q1 q2 q3
R1=[1 0 0; 0 cos(q1) -sin(q1); 0 sin(q1) cos(q1)]; %Rotation matrix for a rotation around axis 1; + CCW rotation
R2=[cos(q2) 0 sin(q2); 0 1 0; -sin(q2) 0 cos(q2)]; %Rotation matrix for a rotation around axis 2; + CCW rotation
R3=[cos(q3) -sin(q3) 0; sin(q3) cos(q3) 0; 0 0 1]; %Rotation matrix for a rotation around axis 3; + CCW rotation
R321=R3*R2*R1;

% RIB10=[2 2-7.2 53.5-35.5]'; %[2 -5.2 18]
% RIB21=[6.5 3.6-7.2 26.4-35.5]'; %[6.5 -3.6 -9.1]
% RIB22=[6.5 3.6-7.2 24.2-35.5]'; % [6.5 -3.6 -11.3]

RIB10=[2 -5.2 19]'; %coordinate Franklin uses
RIB21=[6.5 -3.6 0.9]'; %coordinates Franklin uses
RIB22=[6.5 -3.6 -1.3]'; %coordinates Franklin uses

R321=subs(R321,[q1,q2,q3],[deg2rad(0.528868999999543),0,0]);

pt_RIB10=double(R321*RIB10)
```

```
pt_RIB10 = 3×1
    2.0000
   -5.3752
   18.9512
```

```
pt_RIB21=double(R321*RIB21)
```

```
pt_RIB21 = 3×1
    6.5000
   -3.6082
    0.8667
```

```
pt_RIB22=double(R321*RIB22)
```

```
pt_RIB22 = 3×1
    6.5000
   -3.5878
   -1.3332
```

Figure 3-4: Determining Franklin's initial coordinates prior to transformation for selected attachment points.

Based on the non-transformed coordinates, it appears that errors in the z-coordinate for these attachment points resulted in the discrepancies between the current model's calculations and Franklin's calculations. It seems plausible that in this case, Franklin either adjusted the location of the point from the data provided by Cholewicki *et al.*[54] or data entry errors occurred. It is assumed that Franklin unintentionally adjusted the coordinates due to no mention that he knowingly adjusted them.

For the current model, the skeletal muscle coordinates that define paths of the muscle have been manually verified with Cholewicki *et al.* prior to transformation (Table 3-1, Verification Task 2.1).

3.2.2 **Muscle Path Verification**

The muscle paths provided in the appendix of Franklin's thesis[24] were compared to those provided in the appendix of Cholewicki and McGill[15], the source of the coordinates and muscle paths, after differences between the muscle anatomy for the current model and Franklin's model were discovered (Table 3-1, Verification Task 2.4). It became evident that the nodes of muscles *ParsL4*, *ParsL3*, *ParsL2*, *ParsL1*, *IlioLum*, *LongTP*, *LongTL5*, *LongTL4, LongTL3,* and *LongTL2* on both the left and right side of the body were not consistent between the studies. It appears that Franklin had a compounding data entry error for these muscles.

The *M* matrix describes the muscle path of each muscle as a row in the matrix, listing the coordinates starting with the origin and terminal, followed by the nodes. Evidently, when drafting his Excel spreadsheet, Franklin introduced the nodes of *ParsL2* two rows prior, so they are associated with *ParsL4* instead (Table 3-2). Franklin's muscle paths[24] for *ParsL4*, *ParsL3*, *ParsL2*, *ParsL1*, *IlioLum*, *LongTP*, *LongTL5*, *LongTL4, LongTL3,* and *LongTL2* for the left side of body (Table 3-2) can be compared to that of Cholewicki and McGill[15] (Table 3-3). The compounding data entry error can be identified.

Table 3-2: Franklin muscle paths described in appendix for left side of the body

LParsL4	LPEL9	LL43	LL410				
LParsL3	LPEL9	LL33	LL410				
LParsL2	LPEL9	LL23	LL411	LL310	LL210	LL19	
LParsL1	LPEL9	LL13	LL412	LL311	LL211	LL110	
LIlioLum	LPEL10	LRIB8	LL413	LL312	LL212	LL111	LRIB23
LLongTP	LPEL11	LRIB9	LL312	LL212	LL111	LRIB23	
LLongTL5	LL54	LRIB10	LL212	LL111	LRIB23		
LLongTL4	LL44	LRIB11	LL111	LRIB23			
LLongTL3	LL34	LRIB12					
LLongTL2	LL24	LRIB13					

Table 3-3: Cholewicki and McGill muscle paths described in appendix for left side of the body

LParsL4	LPEL9	LL43					
LParsL3	LPEL9	LL33					
LParsL2	LPEL9	LL23	LL410				
LParsL1	LPEL9	LL13	LL410				
LIlioLum	LPEL10	LRIB8	LL411	LL310	LL210	LL19	
LLongTP	LPEL11	LRIB9	LL412	LL311	LL211	LL110	
LLongTL5	LL54	LRIB10	LL413	LL312	LL212	LL111	LRIB23
LLongTL4	LL44	LRIB11	LL312	LL212	LL111	LRIB23	
LLongTL3	LL34	LRIB12	LL212	LL111	LRIB23		
LLongTL2	LL24	LRIB13	LL111	LRIB23			

For the current model, the muscle paths utilized match those provided by Cholewicki and McGill[15] (Table 3-1, Verification Task 2.3). In his thesis, Franklin indicates that his muscle anatomy is based on that of Cholewicki and McGill[15] and that no alterations have been made to the paths provided[24], confirming the belief that these adjustments to the muscle paths were most likely unintentional.

3.3 Import Muscle Geometry and Skeletal Coordinates

To import the muscle paths and coordinates into MATLAB for the model, Franklin utilized function *muscinput* to read an Excel spreadsheet that described the muscle paths on one sheet and the coordinates on another[24]. Due to errors determined in Franklin's muscle paths and skeletal coordinates (section 3.2), an updated copy of Franklin's Excel spreadsheet was created and utilized to compare the output of the functions used to import the muscle anatomy file.

The absolute error between the muscle anatomy matrix *M* output for Franklin's *muscinput* function and the current model's *load_musclefile* function should be calculated (Table 3-1, Verification Task 2.5).

There are two calculations present for the Verification 2.2 and 2.4 Tasks (Figure 3-5). These verification tasks incorporate comparing the current model's muscle coordinates and muscle paths to that of Franklin. The muscle paths and coordinates are used in the development of the muscle anatomy

matrix *M*. Thus, the muscle anatomy matrix *M* was utilized in these verification tasks. In Figure 3-5, the calculations performed using the *verify* function with the inputs *M* and *M_F* involve Franklin's original Excel file. This run of the verification task will result in unsuccessful verification due to the previous errors in Franklin's *M* file. The calculations performed using the *verify* function with the inputs *M* and *M_F2* involve use of the corrected Excel file in Franklin's function. This corrected Excel file should be an identical input that is used in the functions being compared, so that the resulting absolute error is based on the function procedure and not errors related to the functions' inputs.

```
[M_check_count, ~,M_diff,M_max_error]=verify(M(:,1:32),M_F(:,1:32),1E-3) %Verification Task 2.2 & 2.4
[M2_check_count, ~,M2_diff,M2_max_error]=verify(M(:,1:32),M_F2(:,1:32),1E-3) %Verification Task 2.2 & 2.4
```

Figure 3-5: Verification Tasks 2.2 and 2.4

For the resulting outputs of *M* from the current model's *load_musclefile.m* file and Franklin's *muscinput.m file*, the maximum error occurring between an element of the matrices was calculated, yielding a magnitude of 4.98e-07 (Figure 3-6). The magnitude of this error was not concerning, and therefore Verification Task 2.5 was satisfied.

```
>> max_M2_diff=max(M2_diff,[],'all')

max_M2_diff =

   4.9769e-07
```

Figure 3-6: Verification Task 2.5 Absolute Error

3.4 Import External Force

Similarly to the method used to import the muscle anatomy, an Excel spreadsheet *LoadLift.xls* was used to define the attachment points of the external load applied to the model. The first column of the *E* vector defines the number of attachment points describing the force whereas the four columns that follow describe the coordinates and body of the origin and the last four columns define the coordinates and body of the terminal.

The outputs of Franklin's *forcinput.m* and *format.m* files and the current model's *load_forcefile.m* file are compared (Figure 3-7), yielding a maximum error of 0 (Figure 3-8), so Verification Task 3.2 is satisfied (Table 3-1).

```
[E_check_count, E_check,E_diff,E_max_error]=verify(E,E_F,1E-15) %Verification Task 3.2
```

Figure 3-7: Verification Task 3.2

```
E_diff =

     0     0     0     0     0     0     0     0     0
```

Figure 3-8: Verification Task 3.2 Absolute Error

Additionally, it was manually reviewed and verified that the correct coordinates were substituted for the assigned attachment points specified on the *LoadLift.xls* Excel sheet, and that the external load was being applied in the intended location (Verification Task 3.1).

3.5 Rotation Matrices

Each rigid body is rotated first about Axis 1, then Axis 2, and lastly around Axis 3. To apply these rotations to the bodies, a rotation matrix was used. The current model's rotation matrices are based on the methods discussed by Yamaguchi[81] where a rotation in the counterclockwise direction was defined as positive.

$$R1_{CCW+} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\theta & -sin\theta \\ 0 & sin\theta & cos\theta \end{bmatrix}$$

$$R2_{CCW+} = \begin{bmatrix} cos\theta & 0 & sin\theta \\ 0 & 1 & 0 \\ -sin\theta & 0 & cos\theta \end{bmatrix}$$

$$R3_{CCW+} = \begin{bmatrix} cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Using matrix multiplication in the order that the rotations are applied to the body, a rotation matrix can be derived that includes all of the rotations applied to a body:

$$R123 = R1_{CCW+} * R2_{CCW+} * R3_{CCW+}$$

Based on the Mathematica (Wolfram, Champaign, IL) code provided by Franklin in Appendix A of his thesis,[24] it is clear that clockwise rotation is defined as positive in this case.

$$R1_{CW+} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\theta & sin\theta \\ 0 & -sin\theta & cos\theta \end{bmatrix}$$

$$R2_{CW+} = \begin{bmatrix} cos\theta & 0 & -sin\theta \\ 0 & 1 & 0 \\ sin\theta & 0 & cos\theta \end{bmatrix}$$

$$R3_{CW+} = \begin{bmatrix} cos\theta & sin\theta & 0 \\ -sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

When Franklin applies the rotation matrices in the Mathematica code, he often applies his rotation matrices using the transpose:

$$R123 = (R3_{CW+} * R2_{CW+} * R1_{CW+})^T$$

This would result in the same rotation matrix you would derive if counterclockwise rotations were defined as positive and the rotation matrices were multiplied sequentially in the order that the rotation was applied to the body:

$$R123 = (R3_{CW+} * R2_{CW+} * R1_{CW+})^T = R1_{CCW+} * R2_{CCW+} * R3_{CCW+}$$

In Franklin's *analys1.mexw64 file*, it is evident that the rotation matrices are in the form:

$$R321 = R3_{CW+} * R2_{CW+} * R1_{CW+}$$

Therefore, in order to be able to compare the current model's output to Franklin's output, the rotation matrices needed to be transposed and then the difference calculated between the outputs.

The *rotation_matrices* and *d_rotation_matrices* outputs of the function handles *Rm_funct* and *dRm_funct* were compared to the outputs *Rm* and *dRm* from Franklin's *analys1.mexw64* file (Figure 3-9).

```
277    [Rm_check_count, Rm_check, Rm_diff]=verify(rotation_matricesT,Rm_F,1E-15) %Verification Task 4.1
278    [dRm_check_count, dRm_check,dRm_diff]=verify(d_rotation_matricesT,dRm_F,1E-15) %Verification Task 4.1
```

Figure 3-9: Verification Task 4.1

The absolute difference for the rotation matrices and differential rotation matrices did not result in a magnitude greater than 1E-15 (Figure 3-10), therefore Verification Task 4.1 was satisfied (Table 3-1).

```
Rm_diff =

   1.0e-15 *

         0         0    0.0555
         0         0    0.0555
         0         0         0
         0         0         0
         0         0         0
         0         0         0
         0         0         0
         0         0         0
         0         0         0
         0         0    0.0555
         0         0    0.1110
         0         0         0
         0         0         0
         0         0         0
         0         0         0
         0         0    0.2220
         0    0.1110         0
         0         0    0.1110
         0    0.1110    0.0555
         0    0.0555         0
         0         0    0.1110
```

```
dRm_diff =

   1.0e-15 *

         0         0         0
    0.1110         0    0.0555
         0         0    0.0069
         0    0.0278         0
    0.0278         0    0.0035
         0    0.0555    0.1110
         0    0.0278    0.0416
         0         0    0.0555
         0    0.0278         0
    0.2220    0.0555         0
    0.0555    0.2220         0
         0    0.1110         0
         0    0.2220    0.0555
         0         0         0
         0    0.0555    0.0555
         0    0.2220         0
         0         0    0.2220
         0         0         0
         0         0    0.0555
    0.0139    0.1110    0.1388
         0    0.0139    0.0278
```

Figure 3-10: Verification Task 4.1 Absolute Errors (presented in the MATLAB command window)

3.6 Body Origins

The outputs *origins* and *d_origins* of the *origin_funct* and *dorigin_funct* function handles were compared to the outputs *Or* and *dOr* from Franklin's *analys1.mexw64* file (Figure 3-11).

```
280    [Or_check_count, Or_check,Or_diff]=verify(origin,Or_F,1E-15) %Verification Task 5.1
281    [dOr_check_count, dOr_check,dOr_diff]=verify(dorigin,dOr_F,1E-15) %Verification Task 5.1
```

Figure 3-11: Verification Task 5.1

The absolute difference for the origins and differential origins did not result in a magnitude greater than 1E-15 (Figure 3-12), therefore the magnitude of the error was not concerning, and Verification Task 5.1 was considered satisfied (Table 3-1).

Or_diff =

1.0e-15 *

0	0	0.0555
0	0	0.0555
0	0	0.0555
0	0.0278	0.1110
0	0.0278	0.0555
0	0.0278	0.0555
0	0	0

dOr_diff =

1.0e-16 *

0	0.1214	0.2776
0.2776	0.1041	0
0	0.1041	0
0.2776	0.1388	0
0.2776	0.1735	0
0.2776	0.0694	0
0	0	0

Figure 3-12: Verification Task 5.1 Absolute Error

3.7 Transform Muscle Geometry and Skeletal Coordinates

The process of transforming the skeletal coordinates/muscle attachment points is dependent on the origin and rotation matrices derived (Chapter 3.5 and 3.6).

Franklin's rotated muscle anatomy matrix *Ma* is a 90x33 matrix with each row describing the muscle attachment points for a muscle. Similarly, the current model's rotated muscle anatomy matrix *Ma* is a 90x35 matrix with the last two columns allocated to the CSA and the tendon lengths.

The output of the function handle *Ma_funct* was compared to the *Ma* output from Franklin's *analys1.mexw64* file (Figure 3-13). To complete this verification task, the functions needed to have identical muscle anatomy *M* intputs. The greatest magnitude of absolute error resulting was 1.11E-16 (Figure 3-14), therefore Verification Task 6.1 was considered satisfied (Table 3-1).

```
[Ma_check_count, ~,Ma_diff,Ma_max_error]=verify(Ma(:,1:33),Ma_F,1E-15) %Verification Task 6.1
```

Figure 3-13: Verification Task 6.1

```
Ma_max_error =

   1.1102e-16
```

Figure 3-14: Verification Task 6.1 Absolute Error from MATLAB Command Window

3.8 Muscle Properties

The muscle lengths and velocities from Franklin's *analys1.mexw64* file were compared to those obtained from the current model's function handles *ML_funct* and *MV_funct* (Figure 3-15). The magnitude of error between these measures was 1.11e-16 and 1.94e-16 for muscle length and muscle velocity respectively (Figures 16-17). Verification Task 7.1 was satisfied due to the small magnitudes of error calculated (Table 3-1).

```
[ML_check_count, ~,ML_diff,ML_max_error]=verify(ML,ML_F,1E-15) %Verification Task 7.1
[MV_check_count, ~,MV_diff,MV_max_error]=verify(MV,MV_F,1E-15) %Verification Task 7.1
```

Figure 3-15: Verification Task 7.1

```
>> max_ML_diff=max(ML_diff,[],'all')

max_ML_diff =

   1.1102e-16
```

Figure 3-16: Verification Task 7.1 Absolute Error for muscle length

```
>> max_MV_diff=max(MV_diff,[],'all')

max_MV_diff =

   1.9429e-16
```

Figure 3-17: Verification Task 7.1 Absolute Error for muscle velocity

Additionally, hand calculations were performed to determine a few of the muscle lengths and muscle velocities to verify the outputs of the current model's *rotate_anatomy_sym.mlx* script to ensure that the methodology was implemented in the code correctly. The results of these hand calculations matched the output of the *rotate_anatomy_sym.mlx* script, so the Verification 7.2 Task was successful (Table 3-1).

3.9 Transform External Force

The output of the function handle *Ea_funct* was compared to the output *Ea* from Franklin's *analys1.mexw64* file (Figure 3-18). The resulting absolute error was less than 1e-15 (Figure 3-19), so Verification Task 8.1 was successful (Table 3-1).

```
[Ea_check_count, ~,Ea_diff,Ea_max_error]=verify(Ea,Ea_F(1:9),1E-15) %Verification Task 8.1
```

Figure 3-18: Verification Task 8.1

```
Ea_diff =

   1.0e-15 *

         0         0    0.0139    0.0278    0.1110         0         0    0.0278         0
```

Figure 3-19: Verification Task 8.1 Absolute error presented in the MATLAB command window

Additionally, hand calculations were performed to ensure that the external force attachment points were being rotated correctly (Table 3-1, Verification Task 8.2). The output Ea from the current model's *rotateforce.mlx* script matched the hand calculations performed for this verification task. Therefore, this verification task was deemed successful (Table 3-1, Verification Task 8.2).

3.10 Dynamics

For Verification Task 9.1, the Coriolis vector was compared for the current model's *Cm_funct* function handle and Franklin's *dynmatsc.mex64* file (Figure 3-20). As shown in Figure 3-21, the magnitude of the error between the outputs is concerning for verification, with error ranging from zero to 0.1067 in magnitude. Verification Task 9.1 was unsuccessful (Table 3-1).

```
[Cm_check_count, ~,Cm_diff,Cm_max_error]=verify(Cm,Cm_F',1E-15) %Verification Task 9.1
```

Figure 3-20: Verification Task 9.1

```
Cm_diff =

  Columns 1 through 12

    0.0651    0.0831    0.0000    0.0632    0.0676    0.0000    0.0370    0.0703         0    0.0390    0.0388         0

  Columns 13 through 18

    0.0183    0.0187    0.0000    0.0511    0.1067         0
```

Figure 3-21: Verification Task 9.1 Absolute error

In the appendix of his thesis, Franklin provided the Mathematica code he developed and utilized that included the calculation of the Coriolis vector[24]. The outputs for *Cv* for the Mathematica code and the output yielded from *dynmatsc.mex64* revealed that Franklin's codes did not yield the same results. The results for the current model's *Cm_funct* function handle results agreed with the Mathematica output (Table 3-1, Verification Task 9.3a).

For Verification Task 9.2, the gravity vector and mass matrix were compared for the current model's *Gm_funct and Mm_funct* function handles and Franklin's *dynmatsc.mex64* file (Figure 3-22).

```
[Mm_check_count, ~,Mm_diff,Mm_max_error]=verify(Mm,Mm_F,1E-15) %Verification Task 9.2
[Gm_check_count, ~,Gm_diff,Gm_max_error]=verify(Gm,Gm_F',1E-15) %Verification Task 9.2
```

Figure 3-22: Verification Task 9.2

For the gravity vector and mass matrix, the maximum absolute error magnitude was respectively 4.4409e-16 and 5.5511e-17 (Figure 3-23 and 3-24), resulting in successful verification for Task 9.2 (Table 3-1).

```
Gm_diff =

   1.0e-15 *

  Columns 1 through 12

    0.2220    0.4441         0         0    0.2220         0         0         0         0    0.2220         0         0

  Columns 13 through 18

         0    0.0278         0         0         0         0
```

Figure 3-23: Verification Task 9.2 Absolute Error for Gravity Vector

```
Mm_max_error =

   5.5511e-17
```

Figure 3-24: Verification Task 9.2 Absolute Error for Mass Matrix

Additionally, the gravity vector and mass matrix outputs were compared between Franklin's Mathematica code and the current model's *Gm_funct and Mm_funct* function handles (Table 3-1,

Verification Task 9.3b). The error between the gravity vectors and mass matrix was small in magnitudes, so Verification Task 9.3b was deemed successful.

For Verification Task 9.4, the partial derivatives $\frac{\partial v_b}{\partial \dot{q}_i}$ and $\frac{\partial \omega_b}{\partial \dot{q}_i}$ utilized in the generalized force equation (Equation 2-25) were compared for the current model's *Gl_funct* and *Gr_funct* function handles and Franklin's *analys1.mex64* file (Figure 3-27).

```
[Gl_check_count, ~, Gl_diff,Gl_max_error]=verify(Gl,Gl_F,1E-15) %Verification Task 9.4
[Gr_check_count, ~,Gr_diff,Gr_max_error]=verify(Gr,Gr_F,1E-15) %Verification Task 9.4
```

Figure 3-25: Verification Task 9.4

The variables *Gl* and *Gr* are 108x3 matrices, so due to their vast size, the maximum absolute error was determined. The maximum absolute error for *Gl* was 5.2042e-18 and 1.1102e-16 for *Gr* (Figures 3-28 and 3-29). The magnitudes of these errors were minor, therefore Verification Task 9.4 was considered successful.

```
>> max_Gl_diff=max(Gl_diff,[],'all')

max_Gl_diff =

   5.2042e-18
```

Figure 3-26: Verification Task 9.4 Absolute Error for Gl

```
>> max_Gr_diff=max(Gr_diff,[],'all')

max_Gr_diff =

   1.1102e-16
```

Figure 3-27: Verification Task 9.4 Absolute Error for Gr

3.11 Intervertebral Discs Moments

When writing the code for calculating the intervertebral disc moments, it became evident that Franklin's *intervertc.mexw64* file did not rotate the dummy plane correctly. First, the plane should be rotated so that it has the same lordosis angle as the body it is being compared to. The remaining rotation around Axis 1 is deemed the planar rotation and contributes to the moment. Then the plane should be rotated by this planar angle around Axis 1, and the plane and body should be parallel. If Axis 3 of the plane and body are not aligned, then a twist angle is present. The plane should be rotated by the twist angle so that Axis 3 of the plane and body are aligned. For the body and plane to be aligned and parallel, they would end the calculations with the same rotation matrix. This would indicate that their body axes would be aligned, further confirming that all angles have been accounted for in our calculations.

Through recreation of Franklin's methodology to determine why our codes were not resulting in the same results, Franklin's error was discovered. The dummy plane and body were not aligned at the end of the code, impacting angle measurements and moment calculations.

Calculating the difference between the outputs of Franklin's *intervertc.mexw64* code and the current model's code *IVD_calcs.mlx* for a verification task resulted in a large magnitude of error, deeming Verification Task 10.1 as unsuccessful (Table 3-1).

```
[PRQ_check_count, ~,PRQ_diff,PRQ_max_error]=verify(PRQ,PRQ_F,1E-15) %Verification Task 10.1
```

Figure 3-28: Verification Task 10.1

As mentioned previously, the dummy plane and body should be aligned and parallel if the body was rotated appropriately by the planar angle and twist angle, according to Franklin's methodology[24]. Therefore, a verification task was implemented in the *IVD_calcs.mlx* script to ensure that the body and dummy plane are aligned and parallel through use of the resulting rotation matrices which describe the orientation of the axes.

```
%Check to see if planes are aligned and parallel
planes_check=body1_Rm_unadjusted-body2_Rm;
num_planes_check=numel(planes_check);
failure_vect=zeros(num_planes_check,1);

for j=1:num_planes_check
    if abs(planes_check(j)) > 1E-7
        failure_vect(j)=1;
    else
        failure_vect(j)=0;
    end
end
if sum(failure_vect)~=0
    error('Cannot continue. Planes are not successfully aligned and/or parallel.')
else
    fprintf('SUCCESS. Iteration %2d. This plane is aligned and parallel.\n',i)
end
```

Figure 3-29: Verification task 10.2 in the IVD_calcs.mlx script

This verification task is evaluated each time the *IVD_calcs.mlx* script is called in a simulation, ensuring that the dummy plane and body are parallel and aligned (Table 3-1, Verification Task 10.2).

3.12 Generalized Forces

The resulting generalized force vector *Q* was compared for the current model's *generalizedForce* function and Franklin's *genforcesc.mex64* (Figure 3-30). From the calculated absolute difference, the maximum error magnitude was 1.0658e-14 (Figure 3-31). Thus, due to the small magnitude of error, Verification Task 11.1 was considered successful (Table 3-1).

```
[Q_check_count, ~,Q_diff,Q_max_error]=verify(Q,Q_F,1E-15) %Verification Task 11.1
```

Figure 3-30: Verification Task 11.1

```
Q_diff =

   1.0e-13 *

    0.0178
         0
         0
    0.1066
         0
    0.0711
    0.0533
    0.0044
    0.0178
    0.0355
    0.0711
         0
    0.0355
    0.0178
    0.0488
    0.0711
    0.0888
    0.0355
```

Figure 3-31: Verification Task 11.1 Absolute Error

3.13 Optimization

3.13.1 Constraint Variables

For the optimization procedure, *fmincon* allows for constraints to be defined that the solver must adhere to while minimizing the metabolic power.

For the equilibrium constraint, the passive and active must be balanced:

$$Aeq * \alpha = beq$$

The constraint variables *Aeq* and *beq* are used for the equilibrium requirement. For Verification Task 12.1, the constraint variables were compared for Franklin's *alg_optimize* script and the current model's *optimize.m* script (Figure 3-32).

```
[Aeq_check_count, ~,~,Aeq_max_error]=verify(Aeq,Aeq_F,1E-15) %Verification Task 12.1
[beq_check_count, ~,~,beq_max_error]=verify(beq,Beq_F,1E-15) %Verification Task 12.1
```

Figure 3-32: Verification Task 12.1

Additionally, in the optimization procedure, *A* is an identity matrix multiplied by -1. *B* is defined as a zeros vector. With muscle activations falling between a magnitude of zero and one for each muscle, the following relationship should be met:

$$A * \alpha \leq B$$

If this condition is not met, then muscle activations have the incorrect sign assigned to them.

Verification Task 12.1 includes the evaluation of constraint variables A and B as well for Franklin's *alg_optimize* script and the current model's *optimize.m* script (Figure 3-33).

```
[A_check_count, ~,~,A_max_error]=verify(A,A_F,1E-15) %Verification Task 12.1
[B_check_count, ~,~,B_max_error]=verify(B,B_F,1E-15) %Verification Task 12.1
```

Figure 3-33: Continued verification of 12.1 task

The resulting maximum absolute errors are of a small magnitude, with constraint variable *Aeq* having the greatest error with a magnitude of 6.9122e-11, followed by *beq* with an error of 8.5265e-14 (Figures 3-34 and 3-35). The absolute error magnitude for constraint variable A and B was zero (Figures 3-36 and 3-37).The magnitude of these errors are small, so Verification Task 12.1 has been satisfied (Table 3-1).

```
>> max_Aeq_diff=max(Aeq_diff,[],'all')

max_Aeq_diff =

   6.9122e-11
```

Figure 3-34: Verification Task 12.1 Absolute Error for Aeq

```
>> max_beq_diff=max(beq_diff,[],'all')

max_beq_diff =

   8.5265e-14
```

Figure 3-35: Verification Task 12.1 Absolute Error for beq

```
>> max_B_diff=max(B_diff,[],'all')

max_B_diff =

     0
```

Figure 3-36: Verification Task 12.1 Absolute Error for B

```
>> max_A_diff=max(A_diff,[],'all')

max_A_diff =

     0
```

Figure 3-37: Verification Task 12.1 Absolute Error for A

Additionally, the constraint for muscle activation is as follows, limiting the muscle activation of each muscle:

$$0 \leq \alpha \leq 1$$

This constraint is implemented in the form of matrices, with the length correlating to the number of muscles in the model. The muscle activation is reviewed for the most optimal solution and to ensure that all muscle activations fall between zero and one (Verification Task 12.2).

Additionally, the nonlinear constraint function *nlcon.mlx* needs to be verified. This function ensures that the Jacobians have negative real parts, indicating that the system is stable. After the constrained optimization is performed and the optimal solution is determined, Verification Task 12.5 is performed (Figure 3-38) to ensure that the system is stable (Table 3-1).

```
% Check conditions once again for the optimal solution
Un_opt=Un_best;
    J=Jp;
        for i=1:36
            J(19:36,i)=J(19:36,i)+(Un_opt'*Ja(:,19:36,i))';
        end

% Check the eigenvalues
    eig_val=eig(J);
    max_eig=max(real(eig_val));
```

Figure 3-38: Verification Task 12.5

3.13.2 **Jacobians**

The angular acceleration can be determined using Equation 3-1:

$$\ddot{\theta}(\mathrm{t}) = {M_m}^{-1}(Q - C_m - G_m) \tag{3-1}$$

In the current model, the angular acceleration *qdd* is determined in the *spineaccA*, *spineaccP* and *spineacc* functions. The angular acceleration is used to build the Jacobians, for example (Equation 3-2):

$$Jp_i = \frac{qddP + beq}{\left(0.00001 * \frac{\pi}{180}\right)} \tag{3-2}$$

Therefore the angular accelerations need to be verified (Table 3-1, Verification Task 12.3). The difference between Franklin's angular acceleration outputs and the current model's angular acceleration can be determined (Figure 3-39). The dimensions of the *qddA* matrix is 90x36x36 but for the verification task, only the first page of *qddA* is compared for more efficient computation and simplicity.

```
[~, ~,~,qddA_max_error]=verify(qddA_J(1:90,1:36,1),qddA_F(1:90,1:36,1),1E-15) %Verification Task 12.3
[~, ~,~,qddP_max_error]=verify(qddP_J,qddP_F,1E-15) %Verification Task 12.3
```

Figure 3-39: Verification Task 12.3

In the angular acceleration verification task, the maximum absolute error for *qddA* resulted in a magnitude of 7.094e-11 and a magnitude of 1.137e-13 for *qddP* (Figure 3-40). These errors are of an acceptable magnitude, thus satisfying Verification Task 12.3.

```
qddA_max_error =

   7.0941e-11


qddP_max_error =

   1.1369e-13
```

Figure 3-40: Verification Task 12.3 Absolute Error

During the optimization procedure, the Jacobians are determined. For our verification procedure, the reflex gains G_p and G_d are assigned magnitudes of zero. The Jacobians built based on the delayed components follow the same methodology used for the active properties, with the difference being the utilized muscle model. Thus, in Verification Task 12.4, the instantaneous Jacobians from Franklin's *alg_optimize* script and the current model's *optimize.m* script can be compared (Figure 3-41) and the absolute error determined (Figure 3-42). These error magnitudes are greater than the majority of those determined in the entirety of this verification procedure. As was discussed with Verification Tasks 12.1 and 12.3, the absolute error for the optimization constraints and the angular accelerations were not of a concerning magnitude.

```
[~, ~,~,Ja_max_error]=verify(Jai(1:90,1:36,1),Ja_F(1:90,1:36,1),1E-15) %Verification Task 12.4
[~, ~,~,Jp_max_error]=verify(Jpi,Jp_F,1E-15) %Verification Task 12.4
```

Figure 3-41: Verification Task 12.4

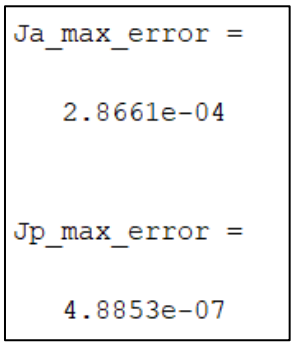


Figure 3-42: Verification Task 12.4 Absolute Error

Notice the term $0.00001 * \frac{\pi}{180}$ in the denominator of Equation 3-2. By dividing by a number with such a small magnitude, it makes sense that the error would be greater for the Jacobians. Therefore, Verification Task 12.4 will be considered satisfied (Table 3-1).

3.13.3 **Cost Function**

Verification Task 12.6 consists of comparing the metabolic power output *fval* for Franklin's *costfun.m* script and the current model's *costfunct.mlx* script (Figure 3-43). The results absolute error between the two outputs had a magnitude of zero (Figure 3-44) , therefore Verification Task 12.6 was successful (Table 3-1).

```
[Power_check_count, ~,Power_diff,Power_max_error]=verify(Power,Power_F,1E-15) %Verification Task 12.6
```

Figure 3-43: Verification Task 12.6

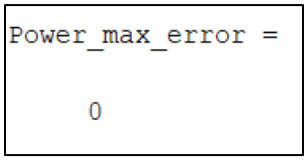


Figure 3-44: Verification Task 12.6 Absolute Error

3.14 Muscle Model

The resulting muscle force from the current model's *musclemodel.mlx* script and Franklin's *muscmodel.m* script were compared for Verification Task 13.1 (Figure 3-45). Verification Task 13.1 was satisfied (Figure 3-46).

```
[MF_check_count, ~,MF_diff,MF_max_error]=verify(fm,MF_F,1E-15) %Verification Task 13.1
```

Figure 3-45: Verification task 13.1

```
MF_max_error =

     0
```

Figure 3-46: Absolute error for muscle force

3.15 Reflex Model

For Verification Task 14.1 (Table 3-1), hand calculations were performed to compare to the output of the current model's *musclemodelR* function. This task was performed to ensure that the *musclemodelR* function syntax was written correctly and was being interpreted as it was intended by MATLAB. Verification Task 14.1 was successful (Table 3-1).

3.16 Time Delay Analysis and Nonlinear Verification

To verify the time delay analysis, a nonlinear simulation was performed with a delay magnitude of 99% of the delay margin. For Verification Task 15.1, the nonlinear simulation should be successful if they delay margin was determined adequately. Figure 3-47 shows the *Dss* vs. Time graph for a hyperlordotic spine with a 200 N applied external load. As time increases, *Dss* decreases, successfully approaching equilibrium. Verification Task 15.1 has been successfully completed (Table 3-1).

The nonlinear system is simulated to verify the results of the linearized system. To determine if the system is approaching equilibrium, the distance from state space was calculated from the nonlinear simulation results.

For Verification Task 16.1 (Table 3-1), nonlinear simulations have been simulated with different magnitude for time delay. Notice in the following plots that for Figure 3-47 the time duration was 5 seconds instead of 2 seconds like it is for the other cases. Time was reduced in the other cases for faster execution.

The Dss vs. Time plot is shown in Figure 3-47 for a hyperlordotic spine with a 200 N load external load applied. Parameters are defined as q=20, b=2, Gp=10 and Gd=0. Delay margin has a magnitude of 0.2006. In Figure 3-47, the simulated time delay is 0.1986, 99% of the delay margin. Notice that Dss is approaching equilibrium as time increases in Figure 3-47.

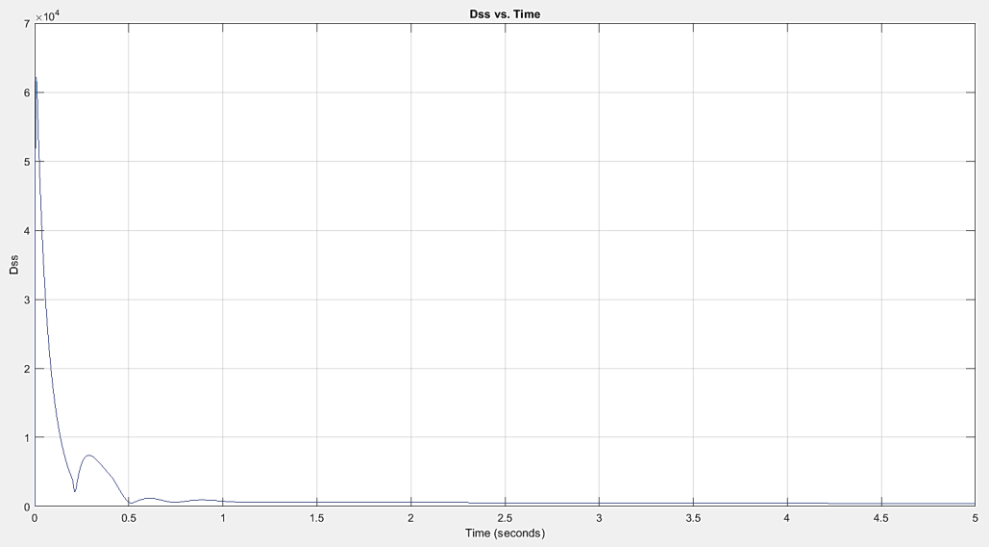


*Figure 3-47: Dss for: simulated delay = 0.99 * delay margin*

Figure 3-48 has the same loading conditions and model parameters. The simulated delay in this case is 0.1505, which is 75% of the delay margin. At Time=1.97 seconds, *Dss*= 8.89435. Visually, Dss appears to be approaching equilibrium. This was confirmed with the *Dss* vector.

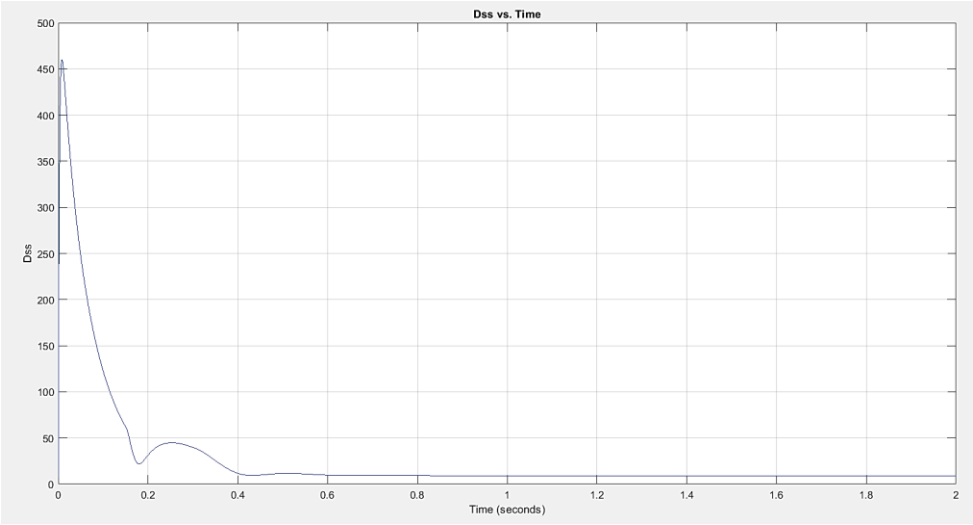


*Figure 3-48: Dss for: simulated delay = 0.75 * delay margin*

Figure 3-49 has the same loading conditions and model parameters. The simulated delay in this case is 0.1003, which is half of the delay margin. At Time=1.97 seconds, *Dss*=9.17. *Dss* is approaching equilibrium as time increases.

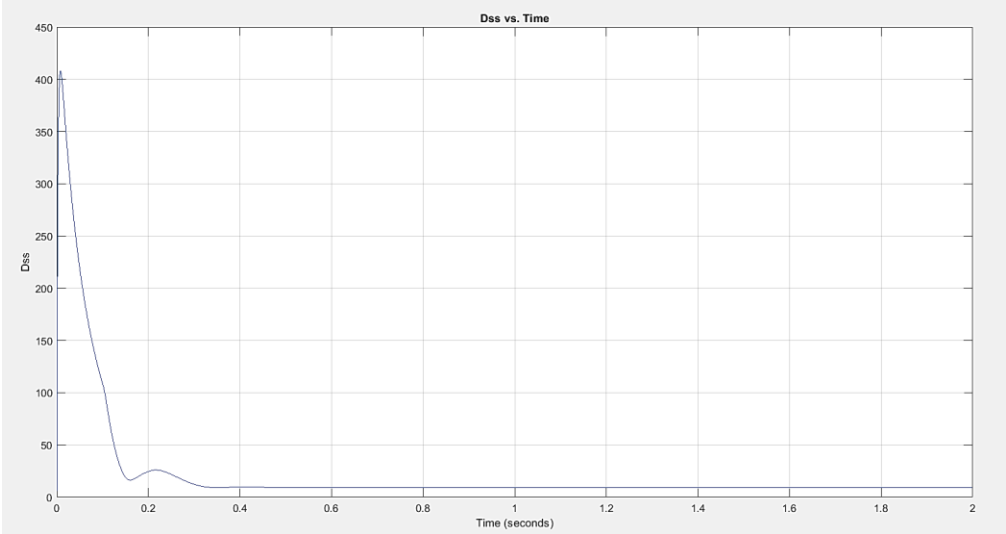


*Figure 3-49: Delay Margin * 0.5*

Figure 3-50 has the same loading conditions and model parameters. The simulated delay in this case is 0.2508, which is 125% of the delay margin. Notice in Figure 3-50 that as *Time* increases, *Dss* starts to increase. This indicates that it is moving away from equilibrium.

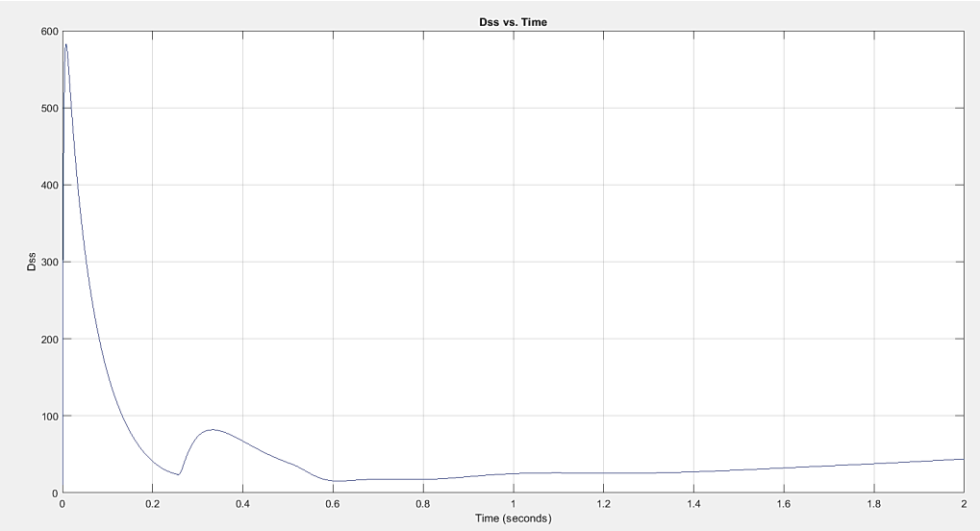


*Figure 3-50: Dss for: simulated delay = 1.25 * delay margin*

Figure 3-51 has the same loading conditions and model parameters. The simulated delay in this case is 0.2006, which is the delay margin. At Time=1.97 seconds, *Dss* = 11.0563. Notice that *Dss* is still approaching equilibrium even though the delay margin equals the simulated delay.

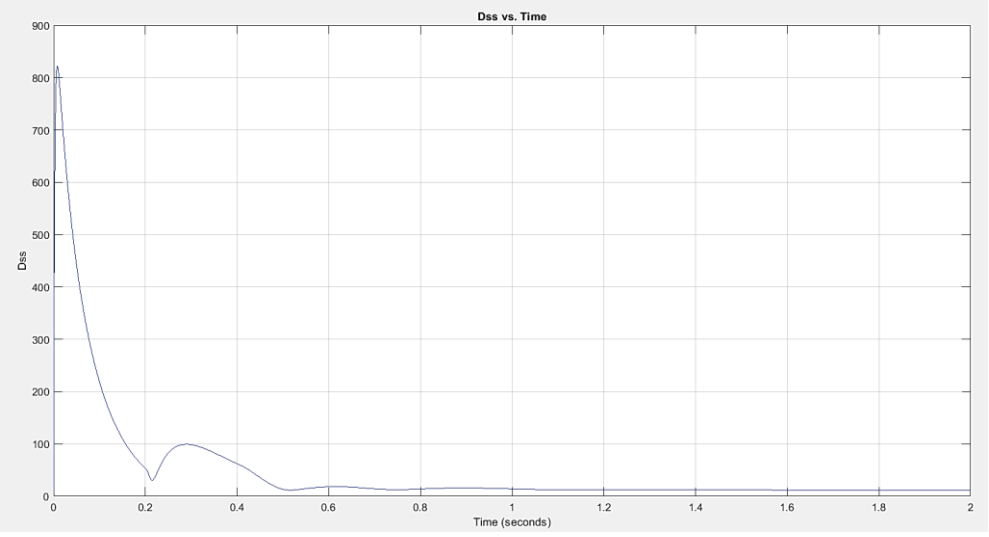


Figure 3-51: Dss for: simulated delay = delay margin

In Franklin's thesis[24], the relationship between the state-space slopes and the percent of the delay margin are compared for a few simulations. It is evident that in some cases, a delay with a greater magnitude than the delay margin can be simulated and still pass the nonlinear verification procedure[24]. It is evident from the simulations he provided that simulations with delays that had a magnitude of 99% of the delay margin had negative slopes[24]. If the simulated delay is greater or equal to the delay margin, then the simulation may not pass the nonlinear verification from some conditions.

Verification Task 16.1 is repeated with each completed nonlinear simulation but for our verification purposes, the model is yielding outputs that verify the linear predictions.

3.17 Summary

The purpose of this chapter was to verify the scripts and functions used in the current spine model to minimize potential errors. Many of these verification tasks included comparing the outputs of the current model to the model files received from the Musculoskeletal Biomechanics Laboratory at Virginia Polytechnic Institute (Michael Madigan, personal communication, July 26, 2008).

The Verification Tasks are presented in Table 3-1, including a column that indicates whether the task successfully verified the current model's script or function. Of the 33 verification tasks listed in Table 3-1, only four of the tasks did not result in successful verification. These failed verification tasks allowed for the errors to be reviewed and if necessary, corrected in the current model. By performing verification tasks, potential errors can be avoided or amended, removing the opportunity for unintentional input or methodology errors to impact the model's predictions.

Chapter 4: The impact of stiffness gain on stability

Modeling of the human trunk: the impact of stiffness gain on stability

Valerie Jardon

Department of Bioengineering, University of Kansas

3138 Learned Hall, Lawrence, KS 66045

valjardon@ku.edu

Dr. Sara E. Wilson[1]

Department of Mechanical Engineering, University of Kansas

3138 Learned Hall, Lawrence, KS 66045

sewilson@ku.edu

ASME Membership (if applicable)

ABSTRACT

In this study, the impact of the stiffness gain in Bergmark's model for short-range stiffness was investigated [6]. A stability-based trunk model was utilized to perform simulations with differing magnitudes of stiffness gain ranging from 0.9 to 40. It was determined that as stiffness gain increased in magnitude, the metabolic power and muscle force required to stabilize the system decreased. When compared to that of stiffness gains less than 10, the variance between the predicted required muscle force and metabolic power required varied less between stiffness gains with a magnitude of 10 or

[1] Corresponding author information: sewilson@ku.edu

higher. This study highlights the impact that stiffness gain can have on model predictions and spine stability.

INTRODUCTION

Prevalence of Low Back Pain. Low back pain (LBP) and injury are prevalent conditions in our society, with an estimated 80% of Americans affected by these conditions in their lifetime [1]. The financial burden of LBP and injury is vast in the United States, with a total annual expense of over 100 billion dollars due to medical intervention costs, lost earnings, and reduced productivity [2]. These substantial financial and physical burdens of LBP and injuries in our society reaffirms the need for continued investigation of the etiology of low back pain.

Spine Modeling and Mechanical Stability. Stability-based trunk models have been utilized by researchers to investigate the mechanisms of low back pain. In this context, spine stability refers to the spine's ability to return to equilibrium following a perturbation [3-4]. If the spine cannot return to equilibrium, then injury may occur. While early spine models neglected to include stability requirements, it became evident that stability must be considered when implementing a muscle recruitment strategy and that muscle recruitment patterns were more accurately predicted by models with the inclusion of stability criteria [5].

Bergmark's Model for Short-range Muscle Stiffness. For minor, quick deformations, muscles exhibit elastic behavior referred to as short-range muscle stiffness [6-7]:

$$K = \frac{q}{l_o} F \tag{1}$$

Where F is muscle force, l_o is the neutral length of the muscle, and q is the dimensionless stiffness gain.

While Bergmark utilized a stiffness gain *q* with a magnitude of 40 in his short-range stiffness model [6], Crisco *et al.* reviewed the literature and determined that the stiffness gain could more reasonably fall within the range of 0.5 to 42 [8]. This plausible range may explain why the dimensionless stiffness gain *q* utilized in studies varies (Table 1). While many researchers utilize a magnitude of 10 for *q* based on the average stiffness gain calculated by Crisco *et al.*[8], the stiffness gain *q* still varies vastly in the literature.

Due to the inconsistencies of the magnitude of the dimensionless stiffness gain *q* in the literature, in this work, the impact of dimensionless stiffness gain *q* on spine stability was evaluated. This relationship needs to be evaluated as muscle stiffness affects the stability analysis of the spinal system.

The hypothesis for this study is as follows: *Increasing the magnitude of the stiffness gain will allow for the spine to stabilize with less required force*.

TRUNK MODEL DEVELOPMENT

In this section, the 18 degree of freedom trunk model utilized in this investigation will be discussed. This trunk model and the methodology used are based on the model developed by Franklin *et al.* [9-11].

Vertebrae. The lumbar vertebrae are represented as independent rigid bodies stacked on a fixed pelvis. An additional rigid body is included in the model, representing the grouped cervical and thoracic regions of the spine. All rigid bodies are represented as elliptical cylinders with their body height based on the skeletal coordinates in Cholewicki *et al.* [12], although this anatomy has been transformed to allow for differing lumbar lordosis angles to be implemented into the model. In this work, the magnitude of the lordosis angle is 58.1°, the mean lordosis angle measured by Pesenti *et al*.[13] Using measurements from Panjabi *et al.* [14] and Berry *et al*. [15], the uniform radii assigned the

rigid bodies were defined with magnitudes of 0.024 m and 0.017 m in the medio-lateral and anterior-posterior directions respectively. Lastly, the masses of the vertebrae are based on the cadaver dissection performed by Lui *et al*. [16] The center of mass of each rigid body is positioned in the center.

Muscles. The skeletal coordinates from Cholewicki *et al*. [12] also describe the muscle paths for the 90 muscle fascicles included in the model. The muscles are modelled using the Hill muscle model [17], consisting of a contractile element, spring and damper (Equation 2):

$$F_m = F_{CE} + Kx + B\dot{x} \tag{2}$$

Using the product of maximum muscle stress (46 N/cm^2), muscle activation and physiological cross-sectional area of the muscles, the force of the contractile element F_{CE} can be calculated (Equation 3) [18]. The muscle activation of each muscle falls between zero and one, with a magnitude of one representing a fully activated muscle.

$$F_{CE} = \left(46\,{}^{N}/_{cm^2}\right) * PCSA * \alpha \tag{3}$$

Bergmark's model of short-range stiffness is used to define the stiffness coefficient (Equation 1). Recall that for this study, the magnitude of stiffness gain *q* varies for each simulation completed. Like the stiffness coefficient, the damping coefficient calculation included a dimensionless damping gain *b* (Equation 4):

$$B = b * \frac{F_{CE}}{x_o} \tag{4}$$

The stiffness coefficient (Equation 1) and damping coefficient (Equation 4) can be substituted into Equation 2, yielding this equation for muscle force:

$$F_m = F_{CE} + \left(\frac{q * f_{max} * \alpha}{x_o} x(t) - x_o\right) + \left(\frac{b * f_{max} * \alpha}{x_o} \dot{x}(t)\right) \quad (5)$$

Reflexes. A PD controller with proportional gain G_P and differential gain G_D is utilized to represent the muscle spindles' ability to respond to a change in velocity and length of the muscle. Muscle activation is calculated by summing the steady-state activation α_o and reflex activation α_r[9-11].

$$\alpha(t) = \alpha_o + \alpha_r = \alpha_o + \alpha_o * \left(G_P * \frac{x(t-\tau) - x_o}{x_o} + G_D * \frac{\dot{x}(t-\tau)}{x_o}\right) \quad (6)$$

Intervertebral Discs. Franklin *et al.* developed and implemented a lumped parameter intervertebral disc (IVD) model in their trunk model [9-11] with a rotational stiffness of 50 Nm/rad based on Stokes *et al*. [19] and rotational damping of 0.5 Nms/rad based on Izambert *et al*. [20].

The angular displacement and velocity of each rigid body needs to be determined and implemented into Equation 7 to calculate the intervertebral disc moment *M*:

$$M = K_{IVD} * \theta + B_{IVD} * \dot{\theta} \quad (7)$$

The methodology for determining the intervertebral disc moments is further explained in Franklin's thesis [10].

Dynamics. The Lagrangian function can be utilized to determine the equations of motion (Equation 8):

$$L = Kinetic\ Energy - Potential\ Energy \quad (8)$$

The total potential energy of the system *PE* is based on the sum of the gravitational energy of each body (Equation 9):

$$PE = \sum_{b=1}^{6} mgh \tag{9}$$

where *m* is the mass of the body, *h* is the vertical position of the center of mass of the body, and *g* is the gravity constant.

The total kinetic energy of the system can be determined by summing the translational and rotation kinetic energy of each body (Equation 10):

$$KE = \sum_{b=1}^{6} \left(\frac{1}{2} m v^2 + \frac{1}{2} I \omega^2\right) \tag{10}$$

where *v* is the velocity of the center of mass, *I* is the mass moment of inertia, and ω is the angular velocity of the body.

The generalized forces can be calculated using Equation 11:

$$Q_i = \sum_{b=1}^{6} \left(\left(\sum_{e=1}^{n} F_{be} \right) \cdot \frac{\partial v_b}{\partial \dot{q}_\iota} + \left(\sum_{e=1}^{n} r_{be} \times F_{be} \right) \cdot \frac{\partial \omega_b}{\partial \dot{q}_\iota} \right) \tag{11}$$

Where *b* is the index of the rigid bodies, F_{be} represents force applied to the body, *e* is the index of the applied forces, *i* is the degree of freedom index, v_b is the linear velocity of the origin of the body, ω_b is the angular velocity of the body and r_{be} is a vector between the point where the force is applied and the origin of the body.

Solving for the equations of motion by performing the Lagrangian Derivative (Equation 12):

$$EOM = \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \frac{\partial L}{\partial q_i} = Q_i \tag{12}$$

$$EOM = \frac{d}{dt}\left(\frac{\partial KE}{\partial \dot{q}_i}\right) - \frac{d}{dt}\left(\frac{\partial PE}{\partial \dot{q}_i}\right) - \frac{\partial KE}{\partial q_i} + \frac{\partial PE}{\partial q_i} = Q_i \quad (13)$$

Using forward dynamic simulation techniques, the dynamic equations can be used to determine the accelerations $\ddot{q}$ (Eq. 14):

$$M_m(q) \cdot \ddot{q} + G_m(q) + C_m(q, \dot{q}) = Q_i \quad (14)$$

Where M_m is the mass matrix, G_m is the gravity vector and C_m is the Coriolis vector.

Optimization Procedure. The linearized representation of the system was determined using the Taylor Series Expansion (Equation 15):

$$\dot{\delta_\theta} = \frac{\partial f(x)}{\partial x}\Big|_{\mathrm{x}=\overline{\mathrm{x}}} \delta_\theta \quad (15)$$

Where $\frac{\partial f(x)}{\partial x}\Big|_{\mathrm{x}=\overline{\mathrm{x}}}$ is the Jacobian matrix that can be approximated through finite differencing.

A constrained optimization is performed to minimize the metabolic power required while adhering to stability and equilibrium constraints. The stability constraint is evaluated using the Jacobian. If the eigenvalues of the Jacobian have negative real parts, then the system is stable.

For this optimization procedure, the instantaneous and delayed components act simultaneously ($\tau = 0$), so a total Jacobian can be determined (Equation 16):

$$J = J_i + J_d \quad (16)$$

This is crucial for the time delay analysis where the delay margin of the system can be determined based on the time delay that results in the system becoming unstable.

Equation 17 represents the metabolic power equation with an assumed equal magnitude of slow-twitch and fast-twitch muscle fibers from Anderson [21]:

$$P = m * \left(74 * 0.5 * \sin\left(\frac{\alpha * \pi}{2}\right) + 111 * 0.5 * \left(1 - \cos\frac{\alpha * \pi}{2}\right)\right) \quad (17)$$

After the optimization procedure is completed, the resulting muscle activation determined from the optimal solution can be utilized to separate the Jacobian into instantaneous and delayed components [9-11] (Eq. 18):

$$\dot{x}(t) = J_i \cdot x(t) + J_d \cdot x(t - \tau) \quad (18)$$

The Jacobian with the delayed components incorporates the time-delayed reflexes.

Time Delay Analysis. The minimum time delay that causes the system to become unstable, also known as the delay margin, can be calculated using a method developed by Chen [22]. The eigenvalues of the Jacobian are utilized for stability analysis. The time delay in which the system becomes unstable can be indicated by the crossing of the imaginary axis by the eigenvalue, violating the stability criteria that all real parts are negative.

A frequency sweeping technique in the Laplace domain can be utilized to determine this crossing event (Eq. 19-20) [9-11,22]. This crossing event is indicated by $\lambda = 1$.

$$j\omega \cdot X = J_i \cdot X + J_D \cdot X \cdot e^{-j\omega\tau} \quad (19)$$

$$(j\omega \cdot I - J_i) \cdot X = \lambda \cdot J_D \cdot X, \qquad where\ \lambda = e^{-j\omega\tau} = e^{-j\theta} \quad (20)$$

Lastly, the delay margin can be calculated as shown in Equation 21:

$$\tau = \frac{\theta}{\omega} \quad (21)$$

If the resulting delay margin has a magnitude of less than 0.060 seconds, then the system is classified as unstable because a delay less than 0.06 is not physiologically realistic for the reflexes [23].

Nonlinear Verification. Lastly, a nonlinear simulation is completed to verify the results predicted by the linear system. Using *the dde23* solver in MATLAB (Mathworks, Natick, MA), a delayed differential equation can be solved. The delay margin should be scaled by 0.99 to determine the time delay prior to the system becoming unstable. This is the delay margin that should be input into the *dde23* solver. In the simulation, a disturbance of 0.01° is applied to the cervical and thoracic region rigid body and each simulation has a minimum duration of five seconds [9-11].

Using Equation 22, the distance of the normalized state-space from equilibrium can be calculated to determine if the system is approaching equilibrium:

$$D = \sqrt{\sum_{i=1}^{18}\left(\left(\frac{\dot{\theta}(t) - \dot{\bar{\theta}}}{mean(\dot{\theta}(t) - \dot{\bar{\theta}})}\right)^2 + \left(\frac{\theta(t) - \bar{\theta}}{mean(\theta(t) - \bar{\theta})}\right)^2\right)} \tag{22}$$

If the system approached equilibrium, then the linear predictions are verified.

Muscle Force Calculations. The muscle force required to stabilize the system was determined using the maximum muscle force [18] and the muscle activations predicted by the model in the optimization procedure. In Equation 23, *CSA* is the cross-sectional area of the muscle and α is the muscle activations.

$$F = \left(46\frac{N}{cm^2}\right) * CSA * \alpha \tag{23}$$

SIMULATION PARAMETERS

For all simulations, based on the methodology utilized by Franklin *et al*. [9-11], an external load of 200 N was applied at the T4 level, 20 cm anterior to the trunk. The assigned magnitudes for the proportional gain G_p, differential gain G_d and damping gain *b* were 10, 0 and 2 respectively. The stiffness gain magnitudes utilized in the simulations ranged from 0.9 to 40, with *q* having the following

magnitudes: 0.9, 1, 2, 4, 5, 10, 15, 20, 25, 30, 35, and 40. Using a step of 0.1, it was determined that 0.9 was the critical stiffness $q_{critical}$ for these simulation parameters.

RESULTS

Metabolic Power. In Figure 2, it is evident that as the magnitude of stiffness gain increases, the metabolic power required to stabilize the system decreases. Of the stiffness gains utilized in the simulations, *q = 0.9* resulted in the greatest magnitude of metabolic power required with a magnitude of 99.18 W. Stiffness gain with a magnitude of 40 resulted in the least metabolic power required with only 35.07 W required. Above a magnitude of 20, increasing stiffness gains had little impact on the required metabolic power.

Muscle Force Required. As shown in Figure 3, as stiffness gain increased, the required muscle force necessary to stabilize the spine decreased. For q = 0.9 and q = 40, approximately 3615 N and 1583 N of muscle force are required to stabilize the spine respectively (Figure 3). For the smaller magnitudes of stiffness gain, the magnitude of required muscle force varies more than it does for the greater magnitudes of stiffness, such as the range from q = 20 to q = 40. There is a change in required muscle force of approximately 330 N from q = 5 to q = 10. Contrastingly, from q = 35 to q = 40, the change in required muscle force is approximately 8 N.

DISCUSSION

Short-range muscle stiffness describes the stiffness of a muscle for a small deformation where the muscle exhibits a more spring-like behavior [6,7,32]. When the stiffness gain magnitudes ranging from 0.9 and 40 were simulated with the 200 N applied external load, it was determined that the smaller magnitudes of stiffness gain required increased magnitudes of metabolic power and muscle force to achieve stability. As the magnitude of stiffness gain increased, the metabolic power and muscle

force required decreased (Figures 2-3). These findings supported our hypothesis. Both the metabolic power and required muscle force calculations include the predicted muscle activations resulting from the constrained optimization procedure and the physiological cross-sectional area of the muscles. Therefore, the similar trends revealed for the required metabolic power versus stiffness gain (Figure 2) and the required muscle force versus stiffness gain (Figure 3) could be expected.

In Figure 2 and Figure 3, it is evident that the required metabolic power and muscle force varies more for the simulations with stiffness gains spanning from q = 0.9 to q = 10 as opposed to stiffness gains with magnitudes greater than 10. For the magnitudes of q exceeding 10, the trend shown in Figure 2 that the required metabolic power decreases with increased stiffness gain is still prevalent but the magnitude in which the required metabolic power varies is not as vast. These same trends can be viewed in Figure 3, with the larger magnitudes of q resulting in less variance in magnitude of the required muscle force as opposed to the smaller magnitudes of q which varied more.

From these results, it is evident that the magnitude utilized for the dimensionless stiffness gain in Bergmark's model of short-range stiffness impacts the model's predictions for the metabolic power and muscle force required to stabilize the system. These findings agree with Bergmark's acknowledgement that the magnitude of the stiffness gain will impact the magnitudes of the model's predictions [6], although Bergmark did not explore to what degree the stiffness gain magnitude would affect model predictions. It is evident from these findings that stiffness gain will influence model predictions and the spinal system's ability to stabilize. For instance, the magnitude of the critical stiffness was determined to be 0.9. This indicates that for a stiffness gain of 0.5, a magnitude in the range provided by Crisco et al. [8], the model would have predicted an unstable system. Investigation needs to be continued regarding the magnitude of stiffness gain that should be utilized when modeling short-range stiffness. Brown and McGill determined an inverse relationship between muscle activation

and stiffness gain [24]. In the future, incorporating a stiffness gain that varies with muscle activation may yield more accurate model predictions.

Additionally, it should be mentioned that the tolerances utilized in the constrained optimization procedure may impact the model's predicted activations slightly. This may explain the slight increase in required muscle force for q = 20 as opposed to that required for q = 15 and q = 25.

CONCLUSIONS

In this study, it was determined that the magnitude of the stiffness gain may impact a model's prediction for spinal stability. The magnitude of the metabolic power and muscle force required to stabilize the system decreased as the stiffness gain increased. In our simulations, a stiffness gain with a magnitude of less than 0.9 would have resulted in instability whereas magnitudes equal to or greater than 0.9 would have been predicted to stabilize. This highlights the important of stiffness gain and its influence in stability analysis. Future studies should consider implementing stiffness gains that vary with muscle activation.

ACKNOWLEDGMENT

FUNDING

This publication was made possible by grant number R01AR046111 from the National Institute of Arthritis and Musculoskeletal and Skin Diseases (NIAMS) at the National Institutes of Health. Its contents are solely the responsibility of the authors and do not necessarily represent the official views of NAIMS.

NOMENCLATURE

KE	Kinetic Energy
PE	Potential Energy
IVD	Intervertebral Disc
q	Stiffness Gain
b	Damping Gain
Gp	Proportional Gain
Gd	Differential Gain
K	Stiffness coefficient
x_o	Equilibrium length
α	Muscle Activation
J_i	Jacobian of the instantaneous components
J_d	Jacobian of the delayed components
L	Lagrangian Derivative
τ	Time Delay

REFERENCES

[1] Rubin, Devon I. "Epidemiology and Risk Factors for Spine Pain." *Neurologic Clinics* 25, no. 2 (2007): 353–71. https://doi.org/10.1016/j.ncl.2007.01.004.

[2] Katz, Jeffrey N. "Lumbar Disc Disorders and Low-Back Pain: Socioeconomic Factors and Consequences." *Journal of Bone and Joint Surgery* 88, no. suppl_2 (2006): 21–24. https://doi.org/10.2106/jbjs.e.01273.

[3] Granata, Kevin P., Patrick E. Lee, and Timothy C. Franklin. "Co-Contraction Recruitment and Spinal Load during Isometric Trunk Flexion and Extension." *Clinical Biomechanics* 20, no. 10 (2005): 1029–37. https://doi.org/10.1016/j.clinbiomech.2005.07.006.

[4] Peter Reeves, N., Kumpati S. Narendra, and Jacek Cholewicki. "Spine Stability: The Six Blind Men and the Elephant." *Clinical Biomechanics* 22, no. 3 (2007): 266–74. https://doi.org/10.1016/j.clinbiomech.2006.11.011.

[5] Granata, K.P., and S.E. Wilson. "Trunk Posture and Spinal Stability." *Clinical Biomechanics* 16, no. 8 (2001): 650–59. https://doi.org/10.1016/s0268-0033(01)00064-x.

[6] Bergmark, Anders. 1989. "Stability of the Lumbar Spine." *Acta Orthopaedica Scandinavica* 60 (sup230): 1–54. https://doi.org/10.3109/17453678909154177.

[7] Morgan, D. L. "Separation of Active and Passive Components of Short-Range Stiffness of Muscle." *American Journal of Physiology-Cell Physiology* 232, no. 1 (1977): 45–49. https://doi.org/10.1152/ajpcell.1977.232.1.c45.

[8] Crisco, J J, and Manohar M. Panjabi. 1991. "The Intersegmental and Multisegmental Muscles of the Lumbar Spine." *Spine* 16 (7): 793–99. https://doi.org/10.1097/00007632-199107000-00018.

[9] Franklin, Timothy C., and Kevin P. Granata. "Role of Reflex Gain and Reflex Delay in Spinal Stability—a Dynamic Simulation." *Journal of Biomechanics* 40, no. 8 (2007): 1762–67. https://doi.org/10.1016/j.jbiomech.2006.08.007.

[10] Franklin, Timothy C., Kevin P. Granata, Michael L. Madigan, and Scott L. Hendricks. 2006. "Linear System Analyses of the Role of Reflex Gain and Delay in a Dynamic Human Spine Model." Thesis, Blacksburg: Virginia Tech. Thesis / Dissertation ETD. .

[11] Franklin, Timothy C., Kevin P. Granata, Michael L. Madigan, and Scott L. Hendricks. 2008. "Linear Time Delay Methods and Stability Analyses of the Human Spine. Effects of Neuromuscular Reflex Response." *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 16 (4): 353–59. https://doi.org/10.1109/tnsre.2008.920080.

[12] Cholewicki, J, and SM McGill. 1996. “Mechanical Stability of the in Vivo Lumbar Spine: Implications for Injury and Chronic Low Back Pain.” *Clinical Biomechanics* 11 (1): 1–15. https://doi.org/10.1016/0268-0033(95)00035-6.

[13] Pesenti, Sebastien, Renaud Lafage, Daniel Stein, Jonathan C. Elysee, Lawrence G. Lenke, Frank J. Schwab, Han Jo Kim, and Virginie Lafage. 2018. “The Amount of Proximal Lumbar Lordosis Is Related to Pelvic Incidence.” *Clinical Orthopaedics & Related Research* 476 (8): 1603–11. https://doi.org/10.1097/corr.0000000000000380.

[14] Panjabi, Manohar M., Vijay Goel, Thomas Oxland, Koichiro Takata, Joanne Duranceau, MARTIN KRAG, and MARK PRICE. “Human Lumbar Vertebrae.” *Spine* 17, no. 3 (1992): 299–306. https://doi.org/10.1097/00007632-199203000-00010.

[15] Berry, James L., James M. Moran, William S. Berg, and Arthur D. Steffee. “A Morphometric Study of Human Lumbar and Selected Thoracic Vertebrae.” *Spine* 12, no. 4 (1987): 362–67. https://doi.org/10.1097/00007632-198705000-00010.

[16] Liu, Y K, J M Laborde, and W C Van Buskirk. “Inertial Properties of a Segmented Cadaver Trunk: Their Implications in Acceleration InjuriesU.” *Aerospace Medicine* 42, no. 6 (June 1971): 650–57.

[17] Hill, A.V. 1938. “The Heat of Shortening and the Dynamic Constants of Muscle.” *Proceedings of the Royal Society of London. Series B - Biological Sciences* 126 (843): 136–95. https://doi.org/10.1098/rspb.1938.0050.

[18] Gardner-Morse, Mack, Ian A. Stokes, and Jeffrey P. Laible. “Role of Muscles in Lumbar Spine Stability in Maximum Extension Efforts.” *Journal of Orthopaedic Research* 13, no. 5 (1995): 802–8. https://doi.org/10.1002/jor.1100130521.

[19] Stokes, Ian A., Mack Gardner-Morse, David Churchill, and Jeffrey P. Laible. “Measurement of a Spinal Motion Segment Stiffness Matrix.” *Journal of Biomechanics* 35, no. 4 (2002): 517–21. https://doi.org/10.1016/s0021-9290(01)00221-4.

[20] Izambert, O., D. Mitton, M. Thourot, and F. Lavaste. 2003. “Dynamic Stiffness and Damping of Human Intervertebral Disc Using Axial Oscillatory Displacement under a Free Mass System.” *European Spine Journal* 12 (6): 562–66. https://doi.org/10.1007/s00586-003-0569-0.

[21] Anderson, Frank Clayton. “A Dynamic Optimization Solution for a Complete Cycle of Normal Gait.” Dissertation, Bell & Howell Information and Learning Company, 2000.

[22] Chen, Jie. 1995. “On Computing the Maximal Delay Intervals for Stability of Linear Delay Systems.” *IEEE Transactions on Automatic Control* 40 (6): 1087–93. https://doi.org/10.1109/9.388690.

[23] Reeves, N.P., J. Cholewicki, and T.E. Milner. “Muscle Reflex Classification of Low-Back Pain.” *Journal of Electromyography and Kinesiology* 15, no. 1 (2005): 53–60. https://doi.org/10.1016/j.jelekin.2004.07.001.

[24] Brown, Stephen H.M., and Stuart M. McGill. “The Relationship between Trunk Muscle Activation and Trunk Stiffness: Examining a Non-Constant Stiffness Gain.” *Computer Methods in Biomechanics and Biomedical Engineering* 13, no. 6 (2010): 829–35. https://doi.org/10.1080/10255841003630652.

[25] Akhavanfar, Mohammad H., Scott C.E. Brandon, Stephen H.M. Brown, and Ryan B. Graham. “Development of a Novel MATLAB-Based Framework for Implementing Mechanical Joint Stability Constraints within OpenSim Musculoskeletal Models.” *Journal of Biomechanics* 91 (2019): 61–68. https://doi.org/10.1016/j.jbiomech.2019.05.007.

[26] Brown, Stephen H.M., and Jim R. Potvin. “Constraining Spine Stability Levels in an Optimization Model Leads to the Prediction of Trunk Muscle Cocontraction and Improved Spine Compression Force Estimates.” *Journal of Biomechanics* 38, no. 4 (2005): 745–54. https://doi.org/10.1016/j.jbiomech.2004.05.011.

[27] Hajihosseinali, M., N. Arjmand, A. Shirazi-Adl, F. Farahmand, and M.S. Ghiasi. “A Novel Stability and Kinematics-Driven Trunk Biomechanical Model to Estimate Muscle and Spinal Forces.” *Medical Engineering & Physics* 36, no. 10 (2014): 1296–1304. https://doi.org/10.1016/j.medengphy.2014.07.009.

[28] Samadi, S, and N Arjmand. “A Novel Stability-Based EMG-Assisted Optimization Method for the Spine.” *Medical Engineering & Physics* 58 (2018): 13–22. https://doi.org/10.1016/j.medengphy.2018.04.019.

[29] Shamsi, MohammadBagher, Javad Sarrafzadeh, Aliashraf Jamshidi, Navid Arjmand, and Farshid Ghezelbash. “Comparison of Spinal Stability Following Motor Control and General Exercises in Nonspecific Chronic Low Back Pain Patients.” *Clinical Biomechanics* 48 (2017): 42–48. https://doi.org/10.1016/j.clinbiomech.2017.07.006.

[30] Stokes, Ian A.F., and Mack Gardner-Morse. “Lumbar Spinal Muscle Activation Synergies Predicted by Multi-Criteria Cost Function.” *Journal of Biomechanics* 34, no. 6 (2001): 733–40. https://doi.org/10.1016/s0021-9290(01)00034-3.

[31] Vakilzadeh, Vahid Khorsand. “A 3-D Stability-Based Dynamic Computational Model of Human Trunk Movement: Towards the Development of a Paradigm for Forensic Spinal Injury Biomechanical Analysis.” *Journal of Forensic Biomechanics* 06, no. 01 (2015). https://doi.org/10.4172/2090-2697.1000119.

[32] Zeinali-Davarani, Shahrokh, Aboulfazl Shirazi-Adl, Behzad Dariush, Hooshang Hemami, and Mohamad Parnianpour. “The Effect of Resistance Level and Stability Demands on

Recruitment Patterns and Internal Loading of Spine in Dynamic Flexion and Extension Using a Simple Trunk Model." *Computer Methods in Biomechanics and Biomedical Engineering* 14, no. 7 (2011): 645–56. https://doi.org/10.1080/10255842.2010.493511.

Figure Captions List

Fig. 1 Spine model. This model includes five rigid lumbar vertebrae, one rigid body representing the cervical and thoracic spine regions, and a fixed pelvis. 90 muscle fascicles are included. The intervertebral discs are represented by a lumped parameter model separating the rigid bodies.

Fig. 2 The relationship between required metabolic power and stiffness gain are presented for a simulation including an external load of 200 N. This load was applied at the T4 level and 20 cm anterior to the trunk. For these simulations, $G_P = 10$, $G_D = 0$ and $b = 2$.

Fig. 3 The relationship between the required muscle force to stabilize the system and stiffness gain is presented. For these simulations, an external load of 200 N was applied at the T4 level and 20 cm anterior to the trunk and $G_P = 10$, $G_D = 0$ and $b = 2$. The required muscle force was calculated utilizing the resulting muscle activations from the optimization procedure in which the metabolic power was minimized while adhering to stability and equilibrium constraints.

Table Caption List

Table 1 Evidence of the varying stiffness gain q magnitudes utilized in previous investigations

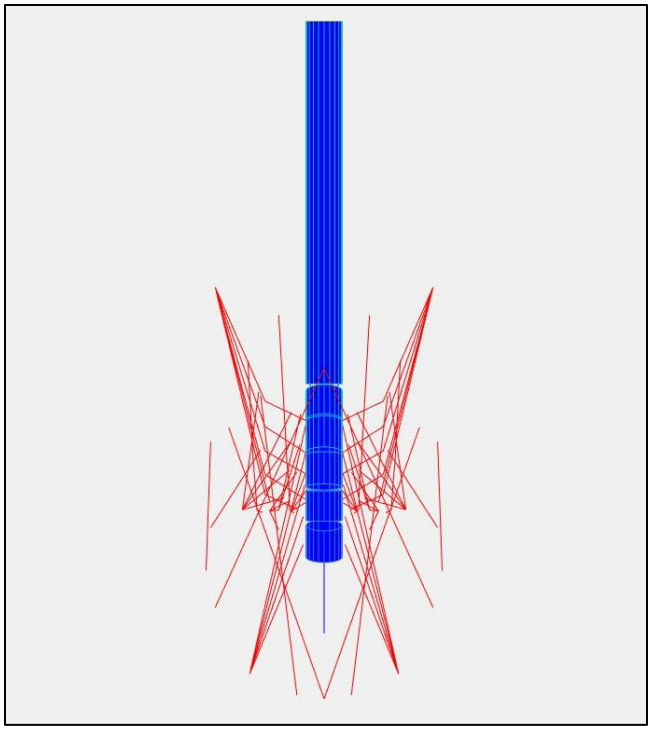


Figure 4-1: Spine model. This model includes five rigid lumbar vertebrae, one rigid body representing the cervical and thoracic spine regions, and a fixed pelvis. 90 muscle fascicles are included. The intervertebral discs are represented by a lumped parameter model

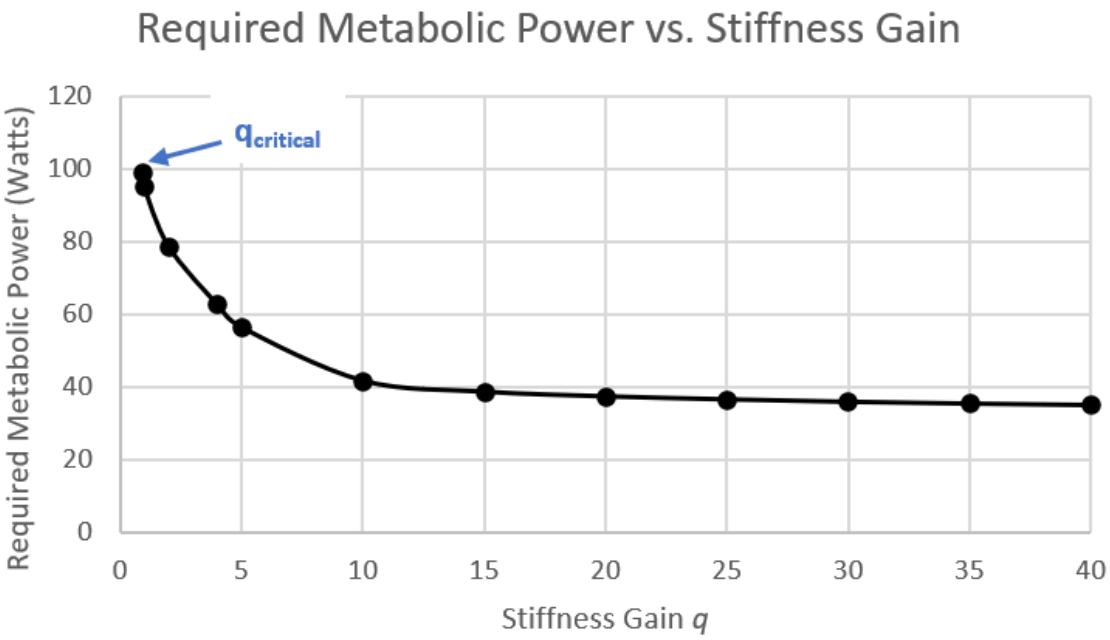


Figure 4-2: The relationship between required metabolic power and stiffness gain are presented for a simulation including an external load of 200 N. This load was applied at the T4 level and 20 cm anterior to the trunk. For these simulations, GP = 10, GD = 0 and b

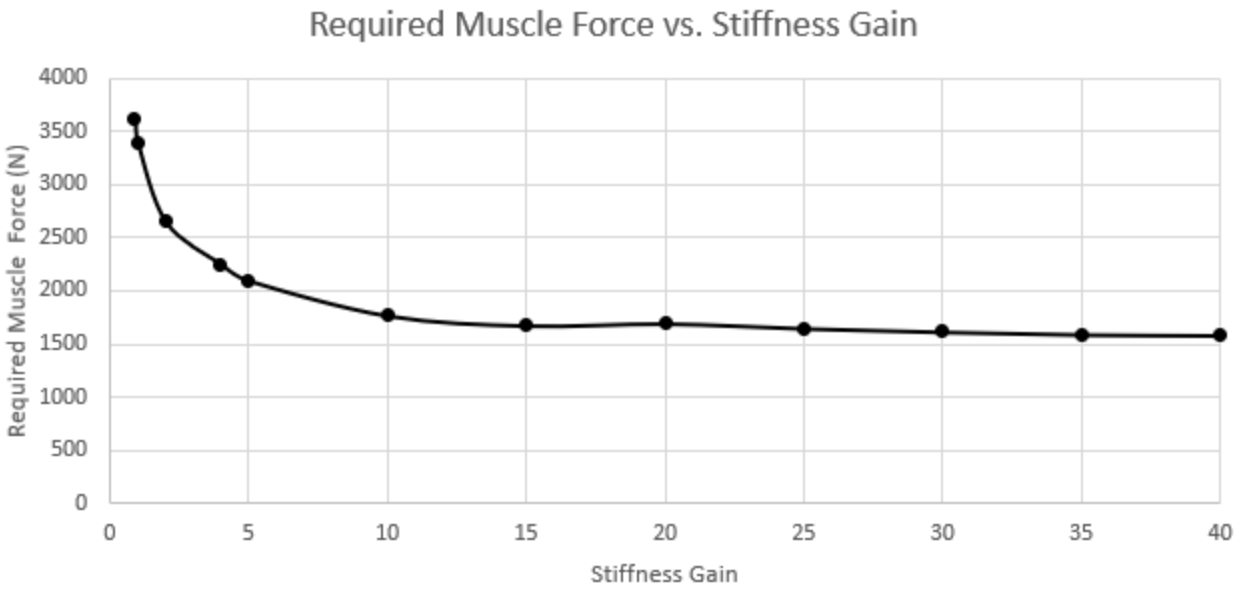


Figure 4-3: The relationship between the required muscle force to stabilize the system and stiffness gain is presented. For these simulations, an external load of 200 N was applied at the T4 level and 20 cm anterior to the trunk and GP = 10, GD = 0 and b = 2. The

Source	Stiffness Gain *q*
Akhavanfar *et al*. [25]	5
Bergmark [6]	40
Brown *et al.* [26]	10
Cholewicki *et al.* [12]	30
Franklin *et al*. [11]	10
Granata & Wilson [5]	5
Hajihosseinali *et al.* [27]	5
Samadi & Arjmand [28]	1.8, 5, 10
Shamsi *et al.* [29]	5
Stokes & Gardner-Morse [30]	5
Vakilzadeh *et al.* [31]	2, 4, 6, 10
Zeinali-Davarani *et al.* [32]	2, 4, 10

Table 4-1: Evidence of the varying stiffness gain q magnitudes utilized in previous investigations

Chapter 5: The effects of hyperlordosis and hydolordosis on spine stability

Modeling of the human trunk: the impact of lumbar lordosis on stability

Valerie Jardon

Department of Bioengineering, University of Kansas

3138 Learned Hall, Lawrence, KS 66045

valjardon@ku.edu

Dr. Sara E. Wilson[2]

Department of Mechanical Engineering, University of Kansas

3138 Learned Hall, Lawrence, KS 66045

sewilson@ku.edu

ASME Membership (if applicable)

ABSTRACT

In this study, the potential impact of lumbar lordosis angle on spine stability was investigated using a stability-based trunk model. Simulations were completed for hyperlordosis (84.1°), hypolordosis (32.1°) and normal lordosis (58.1°) for unloaded and loaded erect posture. When compared to the normal lordosis case, it was determined that the hyperlordosis case would require more metabolic power to stabilize and increased recruitment of muscle flexors. Contrastingly, the hypolordosis case required less metabolic power to stabilize with an increased recruitment of muscle extensors.

[2] Corresponding author information: sewilson@ku.edu

Prevalence of Low Back Pain. Low back pain (LBP) and injury cost the United States more than $100 billion annually [1] and affects approximately 80% of Americans in their lifetime [2]. In a 12-month long study of nearly 2,000 American manual material handling workers, 25% of the sample reported experiencing low back pain that lasted at least 7 days and 10% reported the inability to work due to their ailments [3]. This prevalence in our society proves that furthering the investigation of the causes of low back pain and injury is crucial to the expansion and improvement of current rehabilitation techniques and prevention strategies.

Spine Modeling and Mechanical Stability. Researchers have developed and utilized trunk models to investigate the behaviors of the spine and probable causes of low back pain and injury. To prevent potential injury of the spine, it is crucial that the spine is able to return to equilibrium after being disturbed [4-5]. This is the concept of mechanical stability.

The trunk muscles are vital to the stability of the spine, and in many loading conditions, prevent the spine from buckling [6]. Additionally, Franklin *et al.* determined that time-delayed neuromuscular reflexes are beneficial to the stabilization process and in certain cases necessary for stability, disproving researchers' initial beliefs that solely intrinsic stiffness may be adequate for stability [7].

In the development of trunk models, it is critical that stability analysis be incorporated to ensure that these models are adequately representing realistic spine behavior. Researchers have previously determined that equilibrium and stability criteria should be included in spine models that utilize optimization to determine muscle activations, so that the resulting muscle activation better represents co-contraction [8,9].

Lumbar Lordosis. Lumbar lordosis is the natural curve of the lumbar spine region. The angles of the vertebrae in this natural curve have been known to vary between humans, with a study by Pesenti *et*

al. revealing that in a population of approximately 120 healthy adults, the total lordosis angle ranged from 10-89° with a mean value of 58.1° [10].

The possible correlation between lumbar lordotic curvature and LBP is debated amongst researchers, requiring continued investigation and the comparison of study outcomes, parameters and methodologies [11]. Chun *et al.* completed a meta-analysis and review, with nine of the thirteen studies reporting that subjects with LBP had significantly smaller lumbar lordosis angles when compared to the controls [11]. The range of lordosis angles measured by Pesenti *et al.* [10] and the statistically significant relationship between lumbar lordosis angle and LBP supported by most researchers [11] inspired the current investigation of the possible influence lumbar lordosis angle may have on spinal behavior and stability.

To determine the magnitudes of lordosis angle to be simulated in our study, the definition of hyperlordosis and hypolordosis from Fernand *et al.* was used. Fernand *et al.* [12] defined a hyperlordotic angle as an angle that is greater than the mean lordotic angle plus the magnitude of two standard deviations from the mean. Contrastingly, a hypolordotic angle has been defined as an angle less than the mean lordotic angle minus the magnitude of two standard deviations from the mean. In this investigation, based on the mean total lordosis angle from Pesenti *et al*. [10], hyperlordosis and hypolordosis were defined as a lordotic angle greater than 84.1˚ and less than 32.1˚ respectively. In this study, lumbar lordosis angles of 10˚, 32.1˚, 58.1˚ 84.1˚ and 89˚ will be simulated. Our hypotheses for this study are as follows:

Hypothesis 1: With a load of 0 N applied vertically to T4 at a position 20 cm anterior to the trunk, the hyperlordotic spine will require increased recruitment of flexor muscles to stabilize, consequently increasing the required metabolic power.

Hypothesis 2: With a load of 0 N applied vertically to T4 at a position 20 cm anterior to the trunk, the hypolordotic spine will require increased recruitment of extensor muscles to stabilize, consequently increasing the required metabolic power.

Hypothesis 3: With a load of 200 N applied vertically to T4 at a position 20 cm anterior to the trunk, a greater critical stiffness will be required to stabilize the hypolordotic spine.

TRUNK MODEL DEVELOPMENT

Vertebrae. The spine model and methodology utilized in this investigation is based on the model developed by Franklin *et al.* [7,13-14]. This 18-DOF model incorporates six rigid bodies with each lumbar vertebra represented by an individual rigid body and one collective rigid body representing the cervical and thoracic regions (Figure 1). Each rigid body is an elliptic cylinder with its height based on the skeletal coordinates defined by Cholewicki *et al.* [15]. These coordinates have been rotated so that the vertebral connection points are aligned along the vertical axis. This allows for the bodies to be rotated by the segmental lordosis angle necessary for each lordosis case investigated in this study. The radii of the vertebrae are defined as 0.024 m and 0.017 m in the medio-lateral and anterior-posterior directions respectively, based on measurements from Berry *et al.* [16] and Panjabi *et al.* [17]. The masses of the vertebrae are based on measurements obtained from Liu *et al.* of a cadaver spine [18] and the center of mass of each vertebra is positioned in the center of the rigid body, equidistant from the vertebral faces.

Muscles. The muscle paths are described by the skeletal coordinates from Cholewicki *et al.* [15]. The muscle model utilized in the current model is based on the Hill muscle model [19] and includes a spring, damper, and contractile element in parallel (Eq. 1).

$$F_m = F_{CE} + Kx + B\dot{x} \quad (1)$$

The product of maximum muscle stress (46 N/cm^2), muscle activation α and the physiological cross-sectional area *PCSA* of the muscle [20] represents the force of the contractile element F_{CE} for that muscle(Eq. 2):

$$F_{CE} = \left(46\,{}^{N}/_{cm^2}\right) * PCSA * \alpha \tag{2}$$

Muscle activation can range in magnitude from zero to one, indicating an inactivated muscle and fully activated muscle respectively.

The stiffness coefficient *K* of the spring is based on Bergmark's model of short-range stiffness [21,22] where *q* is the dimensionless stiffness gain, F_{CE} is the force of the contractile element and x_o is the equilibrium muscle length (Eq. 3).

$$K = q * \frac{F_{CE}}{x_o} \tag{3}$$

Crisco and Panjabi [23] provided a range for the dimensionless stiffness gain *q* reported in literature from 0.5-42 with a mean value of 10. In the current study, a value of 20 was utilized for the dimensionless stiffness gain.

The damping coefficient *B* was determined through use of the dimensionless damping gain *b* (Eq. 4). The magnitude of *b* in this study was 2.

$$B = b * \frac{F_{CE}}{x_o} \tag{4}$$

Through the substitution of Equations 2-4 into Equation 1, the resulting muscle force equation can be obtained (Eq. 5):

$$F_m = f_{max} * \alpha + \left(\frac{q * f_{max} * \alpha}{x_o} x(t) - x_o\right) + \left(\frac{b * f_{max} * \alpha}{x_o} \dot{x}(t)\right) \tag{5}$$

Reflexes. The reflex response to the change in length and velocity of the muscle by the muscle spindles is represented by a PD controller with proportional gain G_P and differential gain G_D (Eq. 6). Franklin *et al.* [7,13-14] defined the muscle activation to be the sum of the steady-state α_o and reflex activations α_r (Eq. 6).

$$\alpha(t) = \alpha_o + \alpha_r = \alpha_o + \alpha_o * \left(G_P * \frac{x(t-\tau) - x_o}{x_o} + G_D * \frac{\dot{x}(t-\tau)}{x_o} \right) \quad (6)$$

Intervertebral Discs. The current model utilizes the lumped parameter model developed by Franklin *et al.* for the intervertebral discs that incorporated a rotational spring and damper that oppose angular motion [7,13-14]. To determine the intervertebral disc moments, the angular displacement and velocity must be determined for each rigid body. These values should be scaled by constants K_{IVD} and B_{IVD}, as shown in Equation 7.

$$M = K_{IVD} * \theta + B_{IVD} * \dot{\theta} \quad (7)$$

The stiffness coefficient K_{IVD} was obtained by cadaver measurements performed by Stokes *et al*. [24] and the damping coefficient was based on the relationship between stiffness and damping determined by Izambert *et al*. [25]. In the current study, K_{IVD} and B_{IVD} have magnitudes of 50 Nm/rad and 0.5 Nms/rad respectively.

Dynamics. The equations of motion were derived using the Lagrangian function *L*, which is defined as the difference between kinetic energy *KE* and potential energy *PE* (Eq. 8):

$$L = KE - PE \quad (8)$$

The potential energy of the system is based on the sum of the gravitational energy of each body where *h* is the position of the body's center of mass along the vertical axis, *g* is gravity and *m* is the mass of the body (Eq. 9):

$$PE = \sum_{b=1}^{6} mgh \tag{9}$$

The rotational and translational kinetic energy for the system can be described as shown in Equation 10, where *v* is the velocity of the center of mass, *I* is the mass moment of inertia, and ω is the angular velocity of the body:

$$KE = \sum_{b=1}^{6} \left(\frac{1}{2}mv^2 + \frac{1}{2}I\omega^2\right) \tag{10}$$

The muscle forces, external forces and the IVD moments are utilized in the generalized force calculations (Eq. 11).

$$Q_i = \sum_{b=1}^{6} \left(\left(\sum_{e=1}^{n} F_{be} \right) \cdot \frac{\partial v_b}{\partial \dot{q}_\imath} + \left(\sum_{e=1}^{n} r_{be} \times F_{be} \right) \cdot \frac{\partial \omega_b}{\partial \dot{q}_\imath} \right) \tag{11}$$

where *i* is the degree of freedom index for the degree of freedom q_i, *b* is the rigid body index, *e* is the applied forces index, F_{be} represents force applied to the body, v_b is the linear velocity of the body's origin, r_{be} is a vector between the point where the force is applied and the body's origin, and ω_b is the body's angular velocity.

The Lagrangian Derivative yields the equations of motion (Eq. 12-13):

$$EOM = \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \frac{\partial L}{\partial q_i} = Q_i \tag{12}$$

$$EOM = \frac{d}{dt}\left(\frac{\partial KE}{\partial \dot{q}_i}\right) - \frac{d}{dt}\left(\frac{\partial PE}{\partial \dot{q}_i}\right) - \frac{\partial KE}{\partial q_i} + \frac{\partial PE}{\partial q_i} = Q_i \tag{13}$$

Using forward dynamic simulation methods, the dynamic equations can be used to determine the accelerations $\ddot{q}$ (Eq. 14):

$$M_m(q) \cdot \ddot{q} + G_m(q) + C_m(q, \dot{q}) = Q_i \tag{14}$$

Where M_m is the mass matrix, C_m is the Coriolis vector, and G_m is the gravity vector.

Optimization Procedure. The system was linearized about equilibrium through use of the Taylor Series expansion. Neglecting higher order terms and applying $f(\bar{x}) = 0$ for equilibrium, Equation 15 represents the linearized system:

$$\dot{\delta_\theta} = \frac{\partial f(x)}{\partial x}\Big|_{\mathrm{x}=\bar{\mathrm{x}}} \delta_\theta \tag{15}$$

Where $\frac{\partial f(x)}{\partial x}\Big|_{\mathrm{x}=\bar{\mathrm{x}}}$ is the Jacobian matrix that can be approximated through finite differencing.

By nature, reflexes are time-delayed so for our purposes, the Jacobian was separated into instantaneous and delayed components [7,13-14] (Eq. 16). This separation of the Jacobian matrix will be necessary for the linear time delay calculations.

$$\dot{x}(t) = J_i \cdot x(t) + J_d \cdot x(t - \tau) \tag{16}$$

Next, the model utilizes a constrained optimization procedure that minimizes metabolic power while requiring the system to be stable and in equilibrium. The Jacobian matrix is used for stability analysis. Anderson [26] defines metabolic power using the following equation, with an even magnitude of fast and slow twitch fibers assumed (Eq. 17):

$$P = m * \left(74 * 0.5 * \sin\left(\frac{\alpha * \pi}{2}\right) + 111 * 0.5 * \left(1 - \cos\frac{\alpha * \pi}{2}\right)\right) \tag{17}$$

For the constrained optimization procedure, it is assumed that $\tau = 0$, meaning that the time-delayed components will act at the same time as the instantaneous components. Assuming $\tau = 0$ in this optimization procedure will allow for the delay margin to be determined.

Time Delay Analysis. Using a method developed by Chen [27], the time delay in which the system becomes unstable, referred to as the delay margin, can be determined [7,13-14]. To accomplish this task, the eigenvalues of the Jacobian can be used. By definition, a system is stable when the eigenvalues of its Jacobians have negative real parts. The delay margin can be determined by noting when the first real part of one of the eigenvalues crossed the imaginary axis, indicating instability.

Using the frequency sweeping technique in the Laplace domain (Eq. 18) established by Chen [27], the real part of the eigenvalue crossing the imaginary axis is indicated by $\lambda = 1$ (Eq. 19) [7,13-14,27]:

$$j\omega \cdot X = J_i \cdot X + J_D \cdot X \cdot e^{-j\omega\tau} \tag{18}$$

$$(j\omega \cdot I - J_i) \cdot X = \lambda \cdot J_D \cdot X, \qquad where\ \lambda = e^{-j\omega\tau} = e^{-j\theta} \tag{19}$$

Using Equation 20, the delay margin can be calculated:

$$\tau = \frac{\theta}{\omega} \tag{20}$$

While linear stability is determined based on the eigenvalues, an additional stability criterion should be considered following the time delay analysis: if the delay margin is less than 0.060 seconds, the delay exhibited physiologically for the reflexes [28], then the system is also considered unstable.

Nonlinear Verification. Once the time delay analysis has been performed and the delay margin is determined, the nonlinear simulation can be performed to verify the linear results. For the nonlinear verification, the MATLAB dde23 solver (Mathworks, Natick, MA) is used to solve the delayed differential

equation. The time delay input for the dde23 solver should have the magnitude of 99% of the delay margin, indicating a time delay directly preceding instability. In this simulation, the thoracic/cervical regions' rigid body is disturbed by 0.01° to represent a lifting motion and each simulation was run for a minimum for 5 seconds. The function output can be used to determine if the system is approaching equilibrium by normalizing the state space and calculating its distance from equilibrium (Eq. 21):

$$D = \sqrt{\sum_{i=1}^{18}\left(\left(\frac{\dot{\theta}(t) - \dot{\bar{\theta}}}{mean(\dot{\theta}(t) - \dot{\bar{\theta}})}\right)^2 + \left(\frac{\theta(t) - \bar{\theta}}{mean(\theta(t) - \bar{\theta})}\right)^2\right)} \quad (21)$$

SIMULATION PARAMETERS

A standing unloaded simulation was completed for each lordosis angle (10˚, 32.1˚, 58.1˚ 84.1˚ and 89˚). For the unloaded simulations, the minimal metabolic power required to stabilize the system while adhering to the stability and equilibrium constraints was determined. As mentioned previously, q was assigned a value of 20 and b was assigned a value of 2 for the muscle model for the unloaded simulations. For the reflexes, G_P was assigned a magnitude of 10, and G_D a magnitude of zero.

Additionally, loaded simulations were completed, with a 200 N external load applied 20 cm anterior to the trunk at T4 [7,13-14]. For these simulations, the critical stiffness gain $q_{critical}$ was determined for the 32.1˚, 58.1˚ and 84.1˚ lordosis cases.

RESULTS

Metabolic Power. For the unloaded simulations, the resulting required metabolic power was the greatest for a lordosis angle of 84.1 degrees and the least for an angle of 32.1 degrees, with magnitudes of 58.2 W and 1.8 W respectively (Figure 3). The lordosis angle of 10 degrees resulted in a

metabolic power of 4.59 W while the 89 degrees lordosis angle resulted in a metabolic power of 49.57 W (Figure 3).

For the loaded simulations, the hypolordosis case resulted in the least metabolic power required with magnitudes of 16.7 W for the 10-degree angle and 16.48 W for the 32.1-degree angle (Figure 3). The hyperlordosis cases required a greater magnitude of required metabolic power in comparison to the normal lordosis case and the hypolordosis cases with a magnitude of 60.83 W and 46.65 W metabolic power required for the 84.1-degree angle and the 89-degree angle respectively (Figure 3).

Muscle Activation. For each muscle fascicle, the difference between the resulting magnitude of activation on the left-side and right-side of the body for the unloaded cases was determined. For all simulated lordosis angles, the greatest difference resulted in a magnitude of 0.0253.

The unloaded normal lordosis case resulted in the full recruitment of two multifidus fascicles (MultPL5 & MultL2T12) on each side of the body (Figure 4). While these fascicles were fully activated, they only accounted for 8.7% of the total metabolic power required to stabilize the system (Table 1). The recruitment of the internal obliques (IntOb2) resulted in 32.5% of the total metabolic power required to stabilize (Table 1). Additionally, a total of 44% of the total metabolic power required to stabilize resulted due to the recruitment of multiple Longissimus Thoracis Pars Thoracis fascicles (Figure 4, Table 1).

Similarly, the unloaded hyperlordosis cases resulted in over 30% of the total metabolic power required to stabilize due to recruitment of the internal obliques (Table 1). Additionally, 53.6% and 50.6% of the total metabolic power required to stabilize was due to recruitment of Latissimus Dorsi fascicles for the 84.1-degree and 89-degree lordosis angle respectively (Table 1). Like the normal lordosis case, the hyperlordosis cases also required full activation of some of the multifidi fascicles based on model predictions (Figure 4).

For the unloaded hypolordosis cases, none of the muscle fascicles were fully activated (Figure 4). The majority of muscle fascicles recruited were multifidus and Longissimus Thoracis Pars Lumborum fascicles. For the 10-degree lordosis angle, the multifidus fascicles and Longissimus Thoracis Pars Lumborum fascicles accounted for 68.8% and 24.5% of the total metabolic power required to stabilize, respectively (Table 1). For the 32.1-degree lordosis angle, the multifidus fascicles and Longissimus Thoracis Pars Lumborum fascicles accounted for 54.4% and 15.6% of the total metabolic power required to stabilize, respectively (Table 1).

Critical Stiffness. The critical stiffness $q_{critical}$ was determined for the hypolordosis (32.1˚), normal lordosis (58.1˚), and hyperlordosis (84.1˚) cases for simulations with a 200 N load applied at level T4 and 20 cm anterior to the trunk. The resulting critical stiffness was 9 for the hypolordosis case, 10.9 for the normal lordosis case and 13 for the hyperlordosis case (Table 2).

Nonlinear Verification. Nonlinear simulations were performed for lordosis angle cases of 32.1˚, 58.1˚ 84.1˚. The nonlinear verification involving the calculated distance between equilibrium and normalized state-space can be seen in Figures 5-10. For the unloaded task, the normal lordosis and hypolordosis cases approach equilibrium as time increases. The unloaded hyperlordosis has an oscillatory behavior that is still settling with an increase of time. In the loaded 200 N cases, the hypolordosis and normal lordosis cases are still settling but indicate a negative slope approaching equilibrium. The hyperlordosis case demonstrates an approach to equilibrium.

DISCUSSION

In this study, lumbar lordosis angles of 10˚, 32.1˚, 58.1˚ 84.1˚ and 89˚ were simulated to investigate the potential impact that lumbar lordosis angles may have on spinal stability and behavior.

As hypothesized, the hypolordosis cases required additional recruitment of extensor muscles to successfully adhere to the equilibrium and stability requirements in comparison to the normal lordosis

case (Figure 4). Unlike the normal lordosis and hyperlordosis cases, the hypolordosis cases did not require as much activation of the internal and external obliques, flexor muscles commonly utilized for stability. This was expected as the line of gravity is positioned anteriorly in the hypolordosis case [29], requiring the increased recruitment of the extensor muscles to stabilize and balance the system [21, 29]. The multifidus has been identified as a primary stabilizer of the lumbar spine [21, 29] with its characteristics, such as PCSA and path, resulting in less required metabolic power in comparison to the magnitude other extensors would require [21]. It has been determined that subjects with low back pain show lesser recruitment of the multifidus [30] which could be detrimental to spine stability if the multifidus is recruited as the main stabilizer in certain loading and activities.

The loaded hyperlordosis case required a greater magnitude of required metabolic power when compared to the other cases. In the hyperlordosis cases, flexor muscles were recruited as was anticipated due to the gravity line being located more posteriorly [29] (Figure 4). Interestingly, the system utilized the recruitment of the latissimus dorsi to stabilize the system, which previously has been neglected as a vital stabilizer for the lumbar spine [21, 31]. Future simulations should explore the ability of the system to stabilize with the exclusion of latissimus dorsi in the model for the hyperlordosis case. If the system is able to stabilize, then the alternative muscle recruitment strategy and the magnitude of the metabolic power would be of interest.

Unfortunately at this point in time, it appears that researchers have focused their investigations on the possible correlation between low back pain and lordosis angle magnitude but not the muscle activations needed to stabilize a hyperlordotic spine in an erect standing posture[32-34]. Thus, the model's predicted muscle activations for the hyperlodosis case could not be validated with experimental data.

The critical stiffness determined for the hypolordosis (32.1˚), normal lordosis (58.1˚) and hyperlordsis (84.1˚) cases indicated that the hyperlordosis case would be the lordosis case with the greatest magnitude of critical stiffness. This critical stiffness represents the minimal value of the stiffness gain *q* required to stabilize the spine. Given the required activation and metabolic power required to stabilize the spine, an increased critical stiffness would be reasonable. Based on the range provided for Crisco *et al.* [23] for the stiffness gain, this would indicate that it would be possible for this lordosis case to be considered unstable with the current loading parameters and a smaller stiffness gain.

Lastly, through the nonlinear verification, it was determined that for all lordosis cases and loading cases, the distance between normalized state-space and equilibrium was decreasing with an increase in time, indicating the approach to equilibrium. The unloaded hyperlordosis case was not as certain as the others but due to the wave height deceases for the longer waves present, it was considered to be evidence of it approaching equilibrium. In the future, longer nonlinear simulations could be completed to investigate this behavior over a greater span of time.

CONCLUSIONS

In this study, the impact of lumbar lordosis angle on spine stability was evaluated for normal lordosis, hyperlordosis and hypolordosis. Based on model predictions, the lumbar lordosis angle may have an impact on the metabolic power required to stabilize the spine and muscle activation. The hyperlordosis cases required more metabolic power in order to stabilize while the hypolordosis cases required the least. Additionally, the hypolordotic spine increased recruitment of the extensors while the hyperlordotic spine increase recruitment of the flexors due to the adjusted line of gravity.

ACKNOWLEDGMENT

FUNDING

This publication was made possible by grant number R03AR061597 from the National Institute of Arthritis and Musculoskeletal and Skin Diseases (NIAMS) at the National Institutes of Health. Its contents are solely the responsibility of the authors and do not necessarily represent the official views of NAIMS.

NOMENCLATURE

KE	Kinetic Energy
PE	Potential Energy
IVD	Intervertebral Disc
q	Stiffness Gain
b	Damping Gain
Gp	Proportional Gain
Gd	Differential Gain
K	Stiffness coefficient
x_o	Equilibrium length
α	Muscle Activation
J_i	Jacobian of the instantaneous components
J_d	Jacobian of the delayed components

REFERENCES

[1] Katz, Jeffrey N. "Lumbar Disc Disorders and Low-Back Pain: Socioeconomic Factors and Consequences." *Journal of Bone and Joint Surgery* 88, no. suppl_2 (2006): 21–24. https://doi.org/10.2106/jbjs.e.01273.

[2] Rubin, Devon I. "Epidemiology and Risk Factors for Spine Pain." *Neurologic Clinics* 25, no. 2 (2007): 353–71. https://doi.org/10.1016/j.ncl.2007.01.004.

[3] Ferguson, Sue A., Andrew Merryweather, Matthew S. Thiese, Kurt T. Hegmann, Ming-Lun Lu, Jay M. Kapellusch, and William S. Marras. "Prevalence of Low Back Pain, Seeking Medical Care, and Lost Time Due to Low Back Pain among Manual Material Handling Workers in the United States." *BMC Musculoskeletal Disorders* 20, no. 1 (2019). https://doi.org/10.1186/s12891-019-2594-0.

[4] Granata, Kevin P., Patrick E. Lee, and Timothy C. Franklin. "Co-Contraction Recruitment and Spinal Load during Isometric Trunk Flexion and Extension." *Clinical Biomechanics* 20, no. 10 (2005): 1029–37. https://doi.org/10.1016/j.clinbiomech.2005.07.006.

[5] Peter Reeves, N., Kumpati S. Narendra, and Jacek Cholewicki. "Spine Stability: The Six Blind Men and the Elephant." *Clinical Biomechanics* 22, no. 3 (2007): 266–74. https://doi.org/10.1016/j.clinbiomech.2006.11.011.

[6] Gardner-Morse, Mack, Ian A. Stokes, and Jeffrey P. Laible. "Role of Muscles in Lumbar Spine Stability in Maximum Extension Efforts." *Journal of Orthopaedic Research* 13, no. 5 (1995): 802–8. https://doi.org/10.1002/jor.1100130521.

[7] Franklin, Timothy C., and Kevin P. Granata. "Role of Reflex Gain and Reflex Delay in Spinal Stability—a Dynamic Simulation." *Journal of Biomechanics* 40, no. 8 (2007): 1762–67. https://doi.org/10.1016/j.jbiomech.2006.08.007.

[8] Cholewicki, Jacek, Manohar M. Panjabi, and Armen Khachatryan. "Stabilizing Function of Trunk Flexor-Extensor Muscles around a Neutral Spine Posture." *Spine* 22, no. 19 (1997): 2207–12. https://doi.org/10.1097/00007632-199710010-00003.

[9] Reeves, N. Peter, and Jacek Cholewicki. "Modeling the Human Lumbar Spine for Assessing Spinal Loads, Stability, and Risk of Injury." *Critical Reviews in Biomedical Engineering* 31, no. 1-2 (2003): 72–139. https://doi.org/10.1615/critrevbiomedeng.v31.i12.30.

[10] Pesenti, Sebastien, Renaud Lafage, Daniel Stein, Jonathan C. Elysee, Lawrence G. Lenke, Frank J. Schwab, Han Jo Kim, and Virginie Lafage. 2018. "The Amount of Proximal Lumbar Lordosis Is Related to Pelvic Incidence." *Clinical Orthopaedics & Related Research* 476 (8): 1603–11. https://doi.org/10.1097/corr.0000000000000380.

[11] Chun, Se-Woong, Chai-Young Lim, Keewon Kim, Jinseub Hwang, and Sun G. Chung. "The Relationships between Low Back Pain and Lumbar Lordosis: A Systematic Review and Meta-Analysis." *The Spine Journal* 17, no. 8 (2017): 1180–91. https://doi.org/10.1016/j.spinee.2017.04.034.

[12] Fernand, Robert, and Daniel E. Fox. 1985. “Evaluation of Lumbar Lordosis.” *Spine* 10 (9): 799–803. https://doi.org/10.1097/00007632-198511000-00003.

[13] Franklin, Timothy C., Kevin P. Granata, Michael L. Madigan, and Scott L. Hendricks. 2006. “Linear System Analyses of the Role of Reflex Gain and Delay in a Dynamic Human Spine Model.” Thesis, Blacksburg: Virginia Tech. Thesis / Dissertation ETD. .

[14] Franklin, Timothy C., Kevin P. Granata, Michael L. Madigan, and Scott L. Hendricks. 2008. “Linear Time Delay Methods and Stability Analyses of the Human Spine. Effects of Neuromuscular Reflex Response.” *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 16 (4): 353–59. https://doi.org/10.1109/tnsre.2008.920080.

[15] Cholewicki, J, and SM McGill. 1996. “Mechanical Stability of the in Vivo Lumbar Spine: Implications for Injury and Chronic Low Back Pain.” *Clinical Biomechanics* 11 (1): 1–15. https://doi.org/10.1016/0268-0033(95)00035-6.

[16] Berry, James L., James M. Moran, William S. Berg, and Arthur D. Steffee. “A Morphometric Study of Human Lumbar and Selected Thoracic Vertebrae.” *Spine* 12, no. 4 (1987): 362–67. https://doi.org/10.1097/00007632-198705000-00010.

[17] Panjabi, Manohar M., Vijay Goel, Thomas Oxland, Koichiro Takata, Joanne Duranceau, Martin Krag, and Mark Price. “Human Lumbar Vertebrae.” *Spine* 17, no. 3 (1992): 299–306. https://doi.org/10.1097/00007632-199203000-00010.

[18] Liu, Y K, J M Laborde, and W C Van Buskirk. “Inertial Properties of a Segmented Cadaver Trunk: Their Implications in Acceleration InjuriesU.” *Aerospace Medicine* 42, no. 6 (June 1971): 650–57.

[19] Hill, A.V. 1938. “The Heat of Shortening and the Dynamic Constants of Muscle.” *Proceedings of the Royal Society of London. Series B - Biological Sciences* 126 (843): 136–95. https://doi.org/10.1098/rspb.1938.0050.

[20] Gardner-Morse, Mack, Ian A. Stokes, and Jeffrey P. Laible. “Role of Muscles in Lumbar Spine Stability in Maximum Extension Efforts.” *Journal of Orthopaedic Research* 13, no. 5 (1995): 802–8. https://doi.org/10.1002/jor.1100130521.

[21] Bergmark, Anders. 1989. “Stability of the Lumbar Spine.” *Acta Orthopaedica Scandinavica* 60 (sup230): 1–54. https://doi.org/10.3109/17453678909154177.

[22] Morgan, D. L. “Separation of Active and Passive Components of Short-Range Stiffness of Muscle.” *American Journal of Physiology-Cell Physiology* 232, no. 1 (1977): 45–49. https://doi.org/10.1152/ajpcell.1977.232.1.c45.

[23] Crisco, J J, and Manohar M. Panjabi. 1991. “The Intersegmental and Multisegmental Muscles of the Lumbar Spine.” *Spine* 16 (7): 793–99. https://doi.org/10.1097/00007632-199107000-00018.

[24] Stokes, Ian A., Mack Gardner-Morse, David Churchill, and Jeffrey P. Laible. “Measurement of a Spinal Motion Segment Stiffness Matrix.” *Journal of Biomechanics* 35, no. 4 (2002): 517–21. https://doi.org/10.1016/s0021-9290(01)00221-4.

[25] Izambert, O., D. Mitton, M. Thourot, and F. Lavaste. 2003. “Dynamic Stiffness and Damping of Human Intervertebral Disc Using Axial Oscillatory Displacement under a Free Mass System.” *European Spine Journal* 12 (6): 562–66. https://doi.org/10.1007/s00586-003-0569-0.

[26] Anderson, Frank Clayton. “A Dynamic Optimization Solution for a Complete Cycle of Normal Gait.” Dissertation, Bell & Howell Information and Learning Company, 2000.

[27] Chen, Jie. 1995. “On Computing the Maximal Delay Intervals for Stability of Linear Delay Systems.” *IEEE Transactions on Automatic Control* 40 (6): 1087–93. https://doi.org/10.1109/9.388690.

[28] Reeves, N.P., J. Cholewicki, and T.E. Milner. “Muscle Reflex Classification of Low-Back Pain.” *Journal of Electromyography and Kinesiology* 15, no. 1 (2005): 53–60. https://doi.org/10.1016/j.jelekin.2004.07.001.

[29] Müller, Andreas, Robert Rockenfeller, Nicolas Damm, Michael Kosterhon, Sven R. Kantelhardt, Ameet K. Aiyangar, and Karin Gruber. “Load Distribution in the Lumbar Spine during Modeled Compression Depends on Lordosis.” *Frontiers in Bioengineering and Biotechnology* 9 (2021). https://doi.org/10.3389/fbioe.2021.661258.

[30] Danneels, L., P. Coorevits, A. Cools, G. Vanderstraeten, D. Cambier, E. Witvrouw, and H. De Cuyper. “Differences in Electromyographic Activity in the Multifidus Muscle and the Iliocostalis Lumborum between Healthy Subjects and Patients with Sub-Acute and Chronic Low Back Pain.” *European Spine Journal* 11, no. 1 (2001): 13–19. https://doi.org/10.1007/s005860100314.

[31] Bogduk, Nikolai, Garth Johnson, and Deborah Spalding. “The Morphology and Biomechanics of Latissimus Dorsi.” *Clinical Biomechanics* 13, no. 6 (1998): 377–85. https://doi.org/10.1016/s0268-0033(98)00102-8.

[32] Murrie, V.L., A.K. Dixon, W. Hollingworth, H. Wilson, and T.A.C. Doyle. “Lumbar Lordosis: Study of Patients with and without Low Back Pain.” *Clinical Anatomy* 16, no. 2 (2003): 144–47. https://doi.org/10.1002/ca.10114.

[33] Been, Ella, and Leonid Kalichman. “Lumbar Lordosis.” *The Spine Journal* 14, no. 1 (2014): 87–97. https://doi.org/10.1016/j.spinee.2013.07.464.

[34] Hansson, Tommy, Stanley Bigos, Patric Beecher, and Mark Wortley. “The Lumbar Lordosis in Acute and Chronic Low-Back Pain.” *Spine* 10, no. 2 (1985): 154–55. https://doi.org/10.1097/00007632-198503000-00008.

Figure Captions List

Fig. 1 The current model incorporates 90 muscle fascicles, one rigid body for each of the lumbar vertebrae and one rigid body representing the thoracic and cervical region. A lumped parameter model is utilized to represent the intervertebral discs

Fig. 2 The appearance of hyperlordosis, hypolordosis and normal lordosis in the lumbar spine

Fig. 3 The metabolic power required to stabilize each lordosis case. One simulation was unloaded while the other was loaded with a 200 N load anterior to the trunk.

Fig. 4 Muscle activation for the right-side muscles for the unloaded case

Fig. 5 The distance of the normalized state-space from equilibrium versus time for the unloaded hyperlordotic spine

Fig. 6 The distance of the normalized state-space from equilibrium versus time for the unloaded hypolordotic spine

Fig. 7 The distance of the normalized state-space from equilibrium versus time for the unloaded spine with normal lordosis

Fig. 8 The distance of the normalized state-space from equilibrium versus time for the hyperlordotic spine with a 200 N load applied anteriorly at the T4 level

Fig. 9 The distance of the normalized state-space from equilibrium versus time for the hypolordotic spine with a 200 N load applied anteriorly at the T4 level

Fig. 10 The distance of the normalized state-space from equilibrium versus time for the spine with normal lordosis and a 200 N load applied anteriorly at the T4 level

Table Caption List

Table 1 The percentage of metabolic power required to recruit certain muscle groups to achieve stability in the unloaded cases

Table 2 Critical stiffness for for the hypolordosis (32.1˚), normal lordosis (58.1˚), and hyperlordosis (84.1˚) cases for simulations with a 200 N load applied at level T4 and 20 cm anterior to the trunk

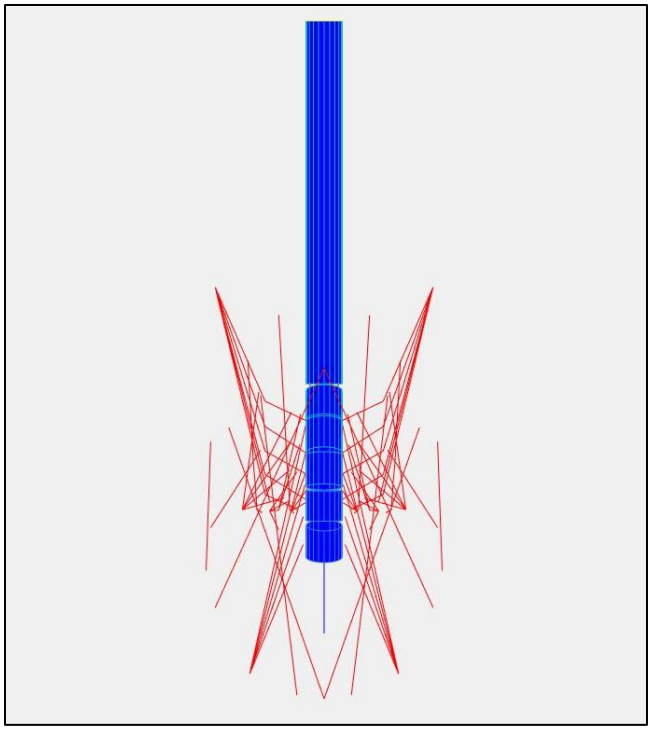


Figure 5-1: The current model incorporates 90 muscle fascicles, one rigid body for each of the lumbar vertebrae and one rigid body representing the thoracic and cervical region. A lumped parameter model is utilized to represent the intervertebral discs

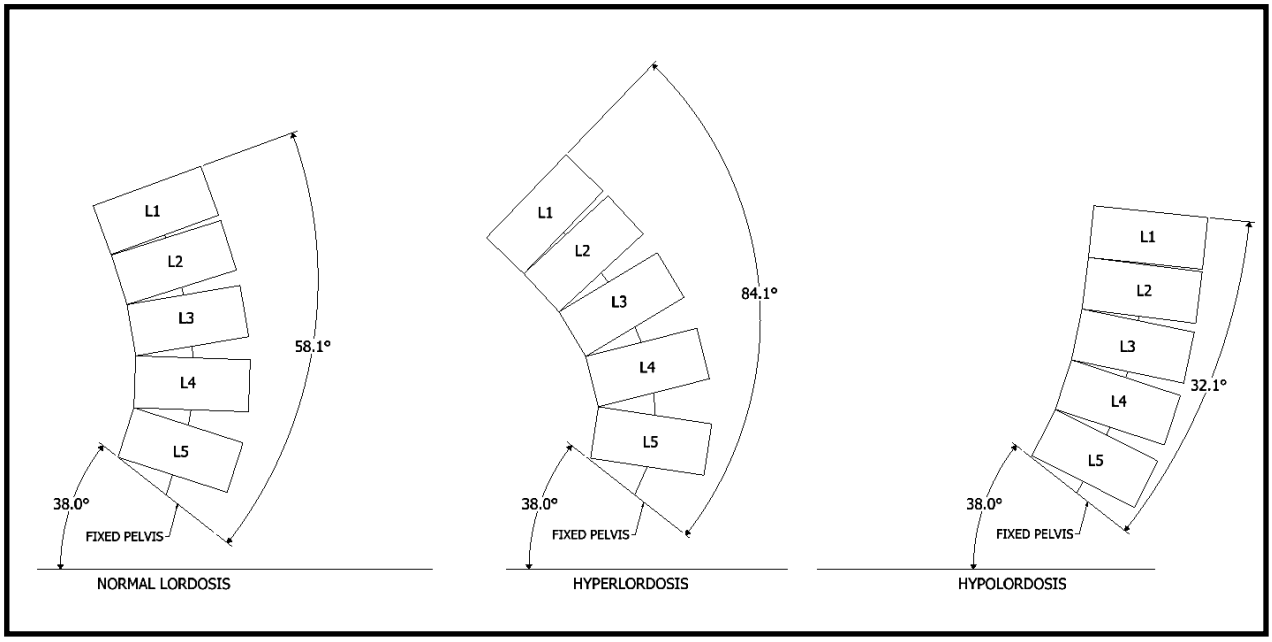


Figure 5-2: The appearance of hyperlordosis, hypolordosis and normal lordosis in the lumbar spine

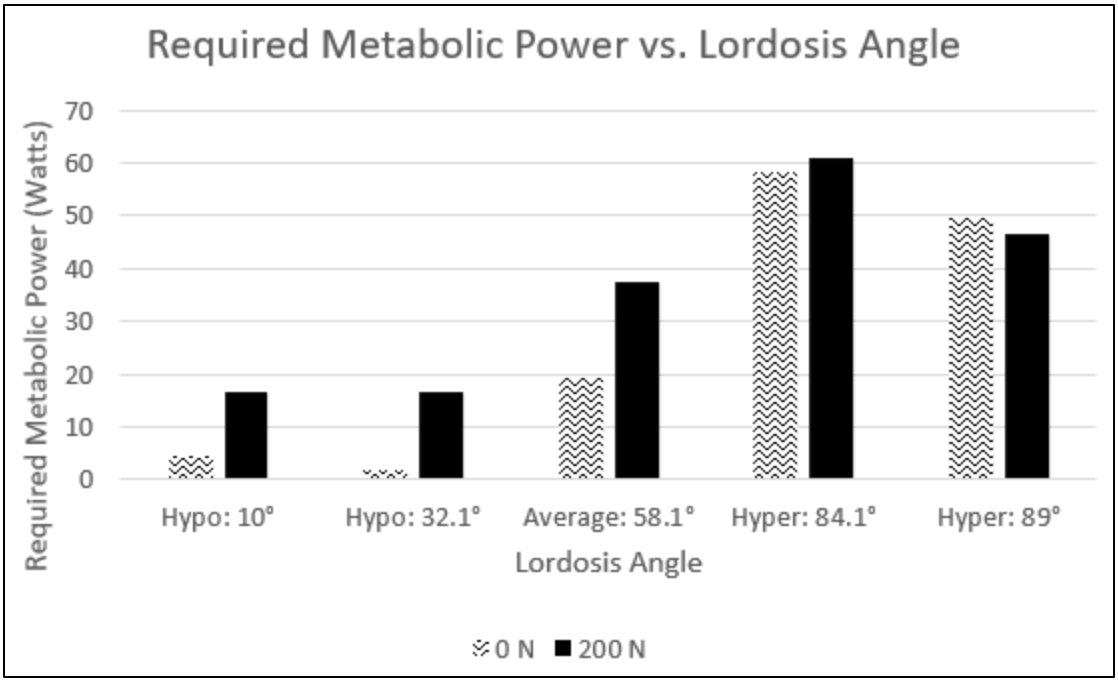


Figure 5-3: The metabolic power required to stabilize each lordosis case. One simulation was unloaded while the other was loaded with a 200 N load anterior to the trunk.

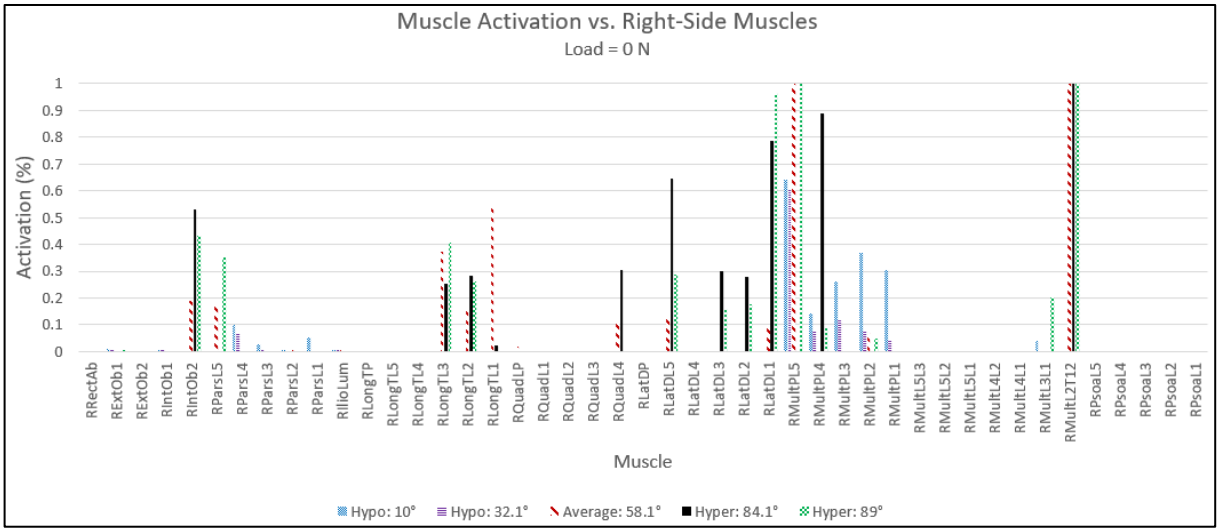


Figure 5-4: Muscle activation for the right-side muscles for the unloaded case

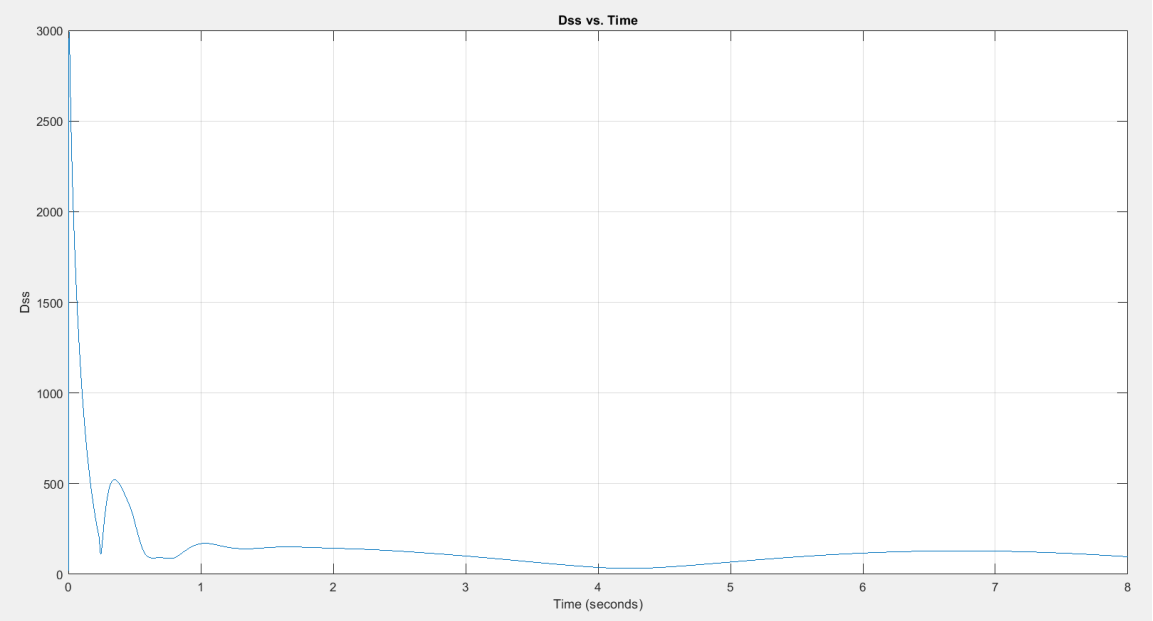


Figure 5-5: The distance of the normalized state-space from equilibrium versus time for the unloaded hyperlordotic spine

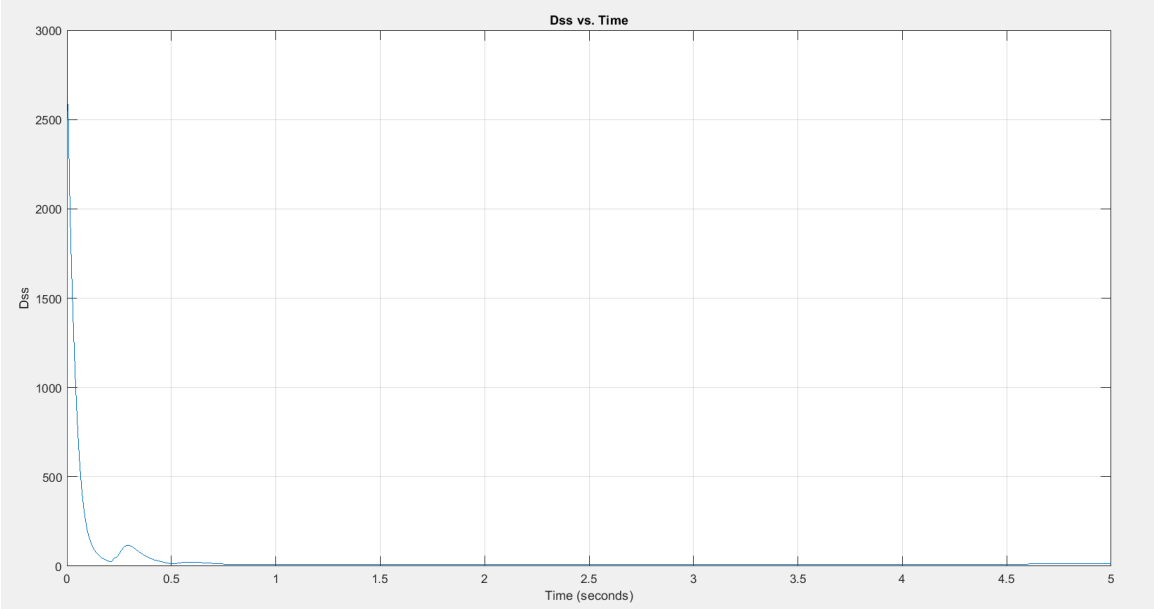


Figure 5-6: The distance of the normalized state-space from equilibrium versus time for the unloaded hypolordotic spine

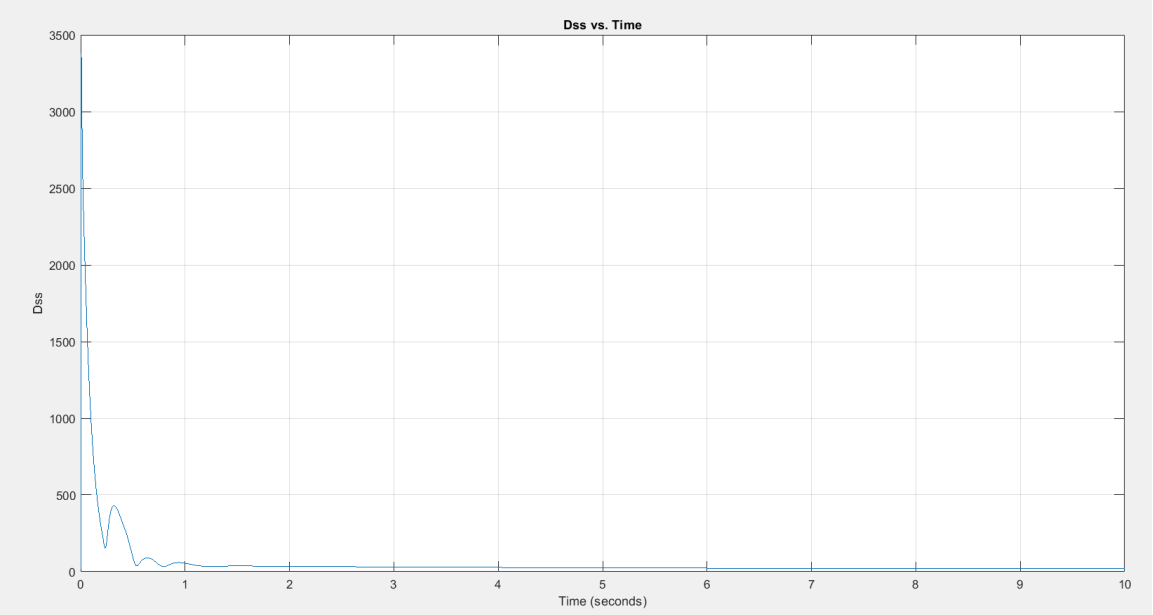


Figure 5-7*: The distance of the normalized state-space from equilibrium versus time for the unloaded spine with normal lordosis*

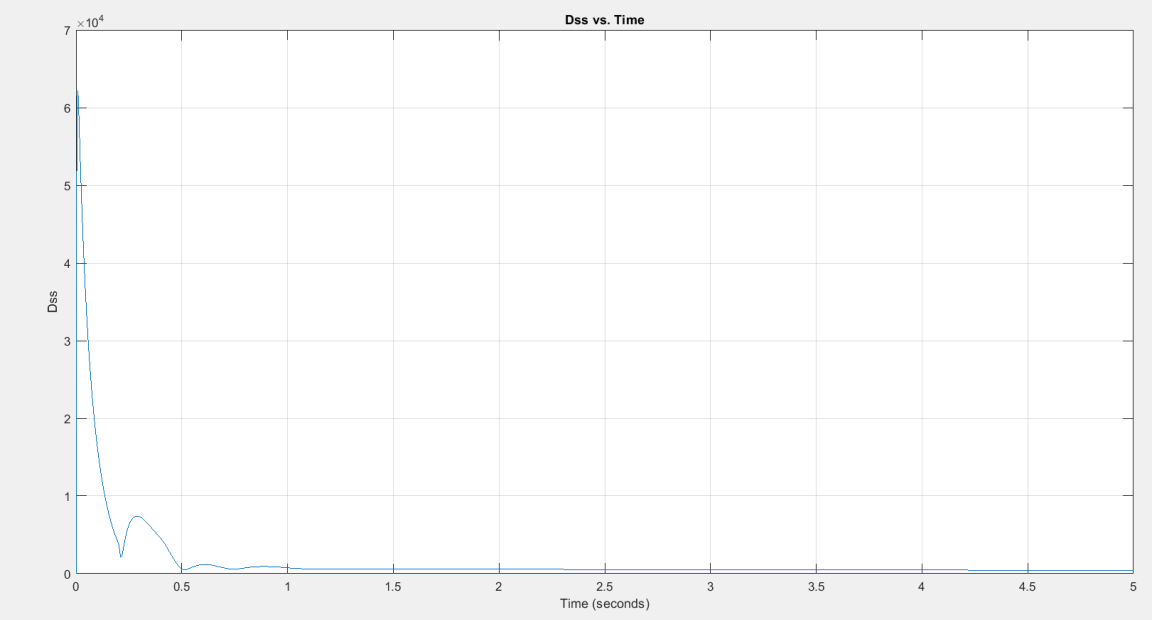


Figure 5-8*: The distance of the normalized state-space from equilibrium versus time for the hyperlordotic spine with a 200 N load applied anteriorly at the T4 level*

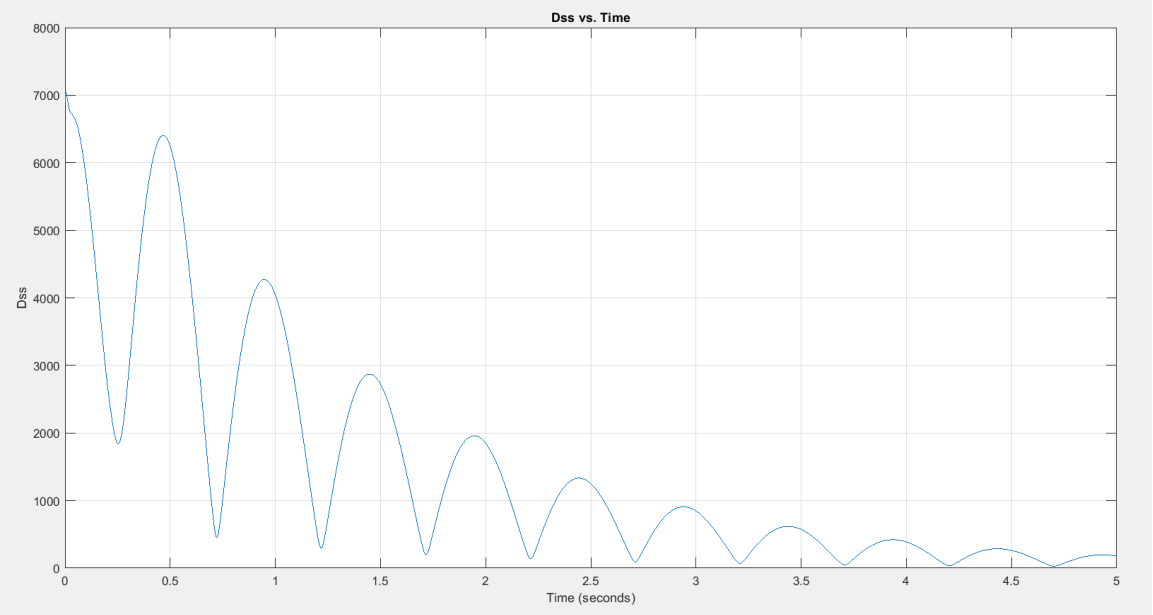


Figure 5-9*: The distance of the normalized state-space from equilibrium versus time for the hypolordotic spine with a 200 N load applied anteriorly at the T4 level*

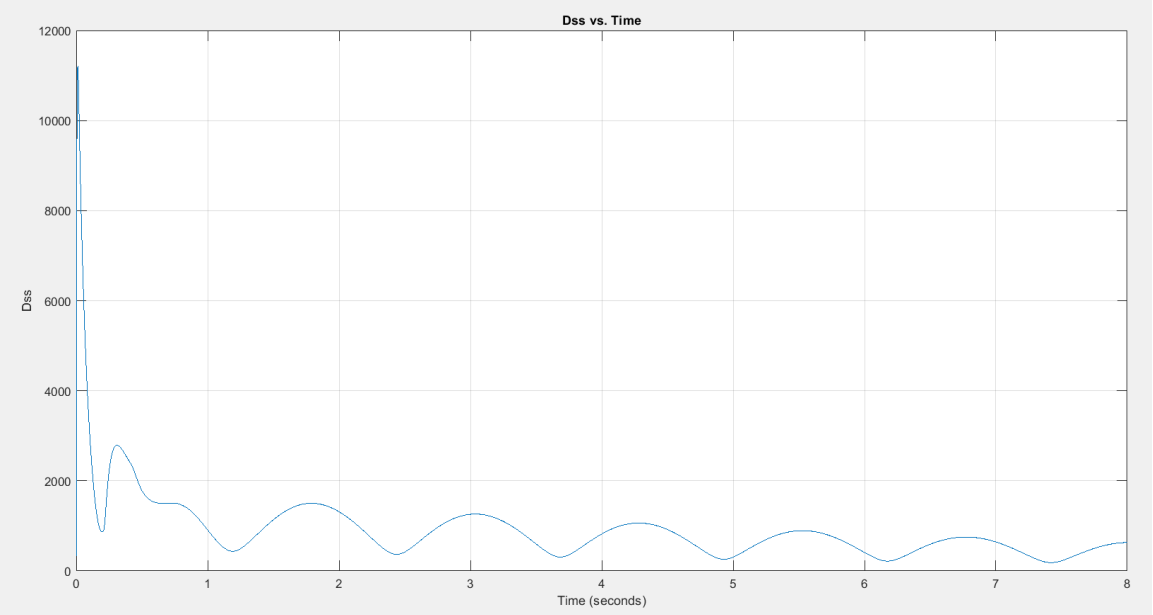


Figure 5-10*: The distance of the normalized state-space from equilibrium versus time for the spine with normal lordosis and a 200 N load applied anteriorly at the T4 level*

	Hypolordosis: 10°	**Hypolordosis: 32.1°**	**Normal Lordosis: 58.1°**	**Hyperlordosis: 84.1°**	**Hyperlordosis: 89°**
	% of total metabolic power required to stabilize				
Ext. Oblique	3.2%	5.3%	0.3%	0.0%	0.1%
Int. Oblique	0.8%	9.3%	32.5%	34.0%	31.9%
Pars Lum.	24.5%	15.6%	2.7%	0.0%	2.4%
Long. Thor.	0.0%	0.0%	44.0%	8.1%	11.0%
Lat. Dorsi	0.0%	0.0%	11.0%	53.6%	50.6%
Multifidus	68.8%	54.4%	8.7%	3.8%	4.0%
Quad. Lum.	0.0%	0.0%	0.7%	0.5%	0.0%
Ilio. Lum.	2.7%	15.5%	0.2%	0.0%	0.0%
Total	100.0%	100.0%	100.0%	100.0%	100.0%

Table 5-1: The percentage of metabolic power required to recruit certain muscle groups to achieve stability in the unloaded cases

	Hypolordosis 32.1°	Normal Lordosis 58.1°	Hyperlordosis 84.1°
$q_{critical}$	9	10.9	13

Table 5-2: Critical stiffness for for the hypolordosis (32.1˚), normal lordosis (58.1˚), and hyperlordosis (84.1˚) cases for simulations with a 200 N load applied at level T4 and 20 cm anterior to the trunk

Chapter 6: Conclusions and Future Work

In this work, the Franklin *et al.* model[16,17,24] was constructed and utilized to investigate the potential impact of stiffness gain and lumbar lordosis on spinal stability. Throughout the development of the model, the MATLAB code and methodology used for the current model was verified with the Franklin *et al.* model[16,17,24] files received through personal communication with the Musculoskeletal Biomechanics Laboratory at Virginia Polytechnic Institute when possible (Michael Madigan, personal communication, July 26, 2008). Additionally, Franklin provided some of the source code in the appendix of his thesis[24] which was utilized when constructing the model.

6.1 Model Verification Procedure

A main component of this work was constructing the Franklin *et al.* model[16,17,24] to utilize in our investigations. In many of the verification tasks, using the Franklin *et al.* files obtained from the Musculoskeletal Biomechanics Laboratory at Virginia Polytechnic Institute (Michael Madigan, personal communication, July 26, 2008), the function and script outputs could be compared for the Franklin *et al.* model and the current model. Additionally, the methodology used by Franklin was critically reviewed as the model was constructed. In many instances in this work, the differences between the current model and Franklin's model are mentioned, usually in reference to updated model parameters or methodology, allowing for the reader will be able to distinguish the updates made to the model and the original model.

Our in-depth verification procedure allowed for errors to be minimized to the best of our ability (Specific Aim 1). In the future, validation tasks should be performed to ensure that model predictions are realistic and correspond with experimental findings for human subjects. To fulfill these validation tasks, experimental findings existing in literature can be utilized or experimental studies can be completed.

6.2 Stiffness Gain and Spine Stability

For Specific Aim 2, the impact of dimensionless stiffness gain *q* on spine stability was evaluated. By increasing the magnitude of stiffness gain, it was determined that less metabolic power and force was required to stabilize the system, supporting our hypothesis. Crisco *et al.* proposed that a range spanning for 0.5 to 42 could be utilized for the stiffness gain in modeling of short-range muscle stiffness[55]. Our results prove that the demand on the muscles to stabilize the spinal greatly differs from that of $q = 0.9$ to $q = 40$, with the model predicting that an additional approximately 64 Watts of metabolic power are required to stabilize the $q = 0.9$ condition. In terms of muscle forces, an additional approximately 2000 N force would need to be recruited from the muscles for the $q = 0.9$ case. For stiffness gain magnitudes of less than 0.9, the system was unable to stabilize.

These results indicate that the magnitude of the stiffness gain selected when modeling short-range stiffness does impact the model predictions, such as the magnitude of metabolic power required, the magnitude of the muscle force required, and the ability for the spine to stabilize in general. Additionally, in the future, the concept of varying stiffness gains should be explored.[90]

6.3 Lordosis Angles and Spine Stability

From Specific Aim 3, the impact of lumbar lordosis angle on spine stability was evaluated. Through our simulations, it was determined that the hyperlordotic spine required more metabolic power and the additional recruitment of the flexor muscles in order to stabilize when compared to the normal lordosis case. This supported our first hypothesis. In contrast to our next hypothesis, the hypolordotic spine required less metabolic power to stabilize in comparison to the normal lordosis case. For the hypolordotic spine, there was an increase in recruitment of muscle extensors, as was hypothesized. In

the future, experimental studies should be completed to yield the muscle activations for a hyperlordotic spine in erect posture.

Many researchers have excluded the latissimus dorsi from their stability-based models. Due to the model predictions that the latissimus dorsi would be utilized to stabilize the normal and hyperlordotic spine, future simulations should explore the ability of the system to stabilize with the exclusion of latissimus dorsi in the model. If the system is able to stabilize in the absence of the latissimus dorsi, then the adjusted muscle recruitment strategy and metabolic power required would be of interest.

Studies have determined that pregnant women have increased lordosis during pregnancy related to the anatomical adjustments to their bodies such as weakened abdominal muscles, increased weight and adjusted center of gravity[91,92]. Liebetrau *et al*. utilized a two-dimensional stability-based model to investigate the behavior of the spine with these anatomical changes[92]. Morino *et al.* developed a model that can be utilized to predict co-contraction in pregnant women[91], assisting in the investigation of the mechanisms causing low back pain during pregnancy.

In the future, the model developed in this work could be expanded to contribute to the investigation of the causes of low back pain during pregnancy. This would involve adjustments to the muscle model to account for the weakened abdominal muscles with adjusted muscle paths, the magnitude of hyperlordosis commonly found in pregnant women be incorporated into the model parameters, and the shifted center of gravity[91].

6.4 Additional Future Work

In addition to the tasks suggested in Sections 6.1-6.3, loading during trunk flexion and rotation should be explored in the future as these have been deemed as risk factors for low back pain[93]. These investigations are crucial for material handlers working in warehouses and in production who utilize these motions frequently. Additionally, asymmetric loading tasks should be explored, including varying

load magnitudes and the point of application of the load. At this point in time, only symmetric loading tasks have been evaluated using this model and the magnitude of the loads have been consistent between researchers[16,17,24].

References

1. Katz JN. Lumbar disc disorders and low-back pain: socioeconomic factors and consequences. *J Bone Joint Surg Am*. 2006;88 Suppl 2:21-24. doi:10.2106/JBJS.E.01273

2. Luckhaupt SE, Dahlhamer JM, Gonzales GT, et al. Prevalence, Recognition of Work-Relatedness, and Effect on Work of Low Back Pain Among U.S. Workers. *Ann Intern Med*. 2019;171(4):301-304. doi:10.7326/M18-3602

3. Cholewicki J, Silfies SP, Shah RA, et al. Delayed trunk muscle reflex responses increase the risk of low back injuries. *Spine (Phila Pa 1976)*. 2005;30(23):2614-2620. doi:10.1097/01.brs.0000188273.27463.bc

4. Radebold A, Cholewicki J, Panjabi MM, Patel TCh. Muscle Response Pattern to Sudden Trunk Loading in Healthy Individuals and in Patients with Chronic Low Back Pain: *Spine*. 2000;25(8):947-954. doi:10.1097/00007632-200004150-00009

5. Silfies SP, Squillante D, Maurer P, Westcott S, Karduna AR. Trunk muscle recruitment patterns in specific chronic low back pain populations. *Clin Biomech (Bristol, Avon)*. 2005;20(5):465-473. doi:10.1016/j.clinbiomech.2005.01.007

6. van Dieën JH, Cholewicki J, Radebold A. Trunk muscle recruitment patterns in patients with low back pain enhance the stability of the lumbar spine. *Spine (Phila Pa 1976)*. 2003;28(8):834-841.

7. Reeves NP, Cholewicki J, Milner TE. Muscle reflex classification of low-back pain. *J Electromyogr Kinesiol*. 2005;15(1):53-60. doi:10.1016/j.jelekin.2004.07.001

8. Radebold A, Cholewicki J, Polzhofer GK, Greene HS. Impaired postural control of the lumbar spine is associated with delayed muscle response times in patients with chronic idiopathic low back pain. *Spine (Phila Pa 1976)*. 2001;26(7):724-730. doi:10.1097/00007632-200104010-00004

9. Reeves NP, Cholewicki J. Modeling the Human Lumbar Spine for Assessing Spinal Loads, Stability, and Risk of Injury. *Crit Rev Biomed Eng*. 2003;31(1-2):72-139. doi:10.1615/CritRevBiomedEng.v31.i12.30

10. Cholewicki J, Panjabi MM, Khachatryan A. Stabilizing function of trunk flexor-extensor muscles around a neutral spine posture. *Spine (Phila Pa 1976)*. 1997;22(19):2207-2212. doi:10.1097/00007632-199710010-00003

11. Liebetrau A, Puta C, Anders C, de Lussanet MHE, Wagner H. Influence of delayed muscle reflexes on spinal stability. *Human Movement Science*. 2013;32(5):954-970. doi:10.1016/j.humov.2013.03.006

12. Wagner H, Anders Ch, Puta Ch, et al. Musculoskeletal support of lumbar spine stability. *Pathophysiology*. 2005;12(4):257-265. doi:10.1016/j.pathophys.2005.09.007

13. Granata KP, Wilson SE. Trunk posture and spinal stability. *Clinical Biomechanics*. 2001;16(8):650-659. doi:10.1016/S0268-0033(01)00064-X

14. Granata KP, Lee PE, Franklin TC. Co-contraction recruitment and spinal load during isometric trunk flexion and extension. *Clin Biomech (Bristol, Avon)*. 2005;20(10):1029-1037. doi:10.1016/j.clinbiomech.2005.07.006

15. Cholewicki J, McGill SM. Mechanical stability of the in vivo lumbar spine: implications for injury and chronic low back pain. *Clin Biomech (Bristol, Avon)*. 1996;11(1):1-15. doi:10.1016/0268-0033(95)00035-6

16. Franklin TC, Granata KP. Role of reflex gain and reflex delay in spinal stability—A dynamic simulation. *Journal of Biomechanics*. 2007;40(8):1762-1767. doi:10.1016/j.jbiomech.2006.08.007

17. Franklin TC, Granata KP, Madigan ML, Hendricks SL. Linear Time Delay Methods and Stability Analyses of the Human Spine. Effects of Neuromuscular Reflex Response. *IEEE Trans Neural Syst Rehabil Eng*. 2008;16(4):353-359. doi:10.1109/TNSRE.2008.920080

18. Bergmark A. Stability of the lumbar spine: A study in mechanical engineering. *Acta Orthopaedica Scandinavica*. 1989;60(sup230):1-54. doi:10.3109/17453678909154177

19. Brown SHM, Potvin JR. Constraining spine stability levels in an optimization model leads to the prediction of trunk muscle cocontraction and improved spine compression force estimates. *Journal of Biomechanics*. 2005;38(4):745-754. doi:10.1016/j.jbiomech.2004.05.011

20. Granata KP, Orishimo KF. Response of trunk muscle coactivation to changes in spinal stability. *Journal of Biomechanics*. 2001;34(9):1117-1123. doi:10.1016/S0021-9290(01)00081-1

21. Reeves NP, Narendra KS, Cholewicki J. Spine stability: the six blind men and the elephant. *Clin Biomech (Bristol, Avon)*. 2007;22(3):266-274. doi:10.1016/j.clinbiomech.2006.11.011

22. Zazulak B, Cholewicki J, Reeves NP. Neuromuscular control of trunk stability: clinical implications for sports injury prevention. *J Am Acad Orthop Surg*. 2008;16(9):497-505.

23. Gardner-Morse M, Stokes IA, Laible JP. Role of muscles in lumbar spine stability in maximum extension efforts. *J Orthop Res*. 1995;13(5):802-808. doi:10.1002/jor.1100130521

24. Franklin TC. *Linear System Analyses of the Role of Reflex Gain and Delay in a Dynamic Human Spine Model*. Thesis. Virginia Polytechnic Institute and State University. http://scholar.lib.vt.edu/theses/available/etd-06162006-104419/

25. Moorhouse KM, Granata KP. Role of reflex dynamics in spinal stability: intrinsic muscle stiffness alone is insufficient for stability. *J Biomech*. 2007;40(5):1058-1065. doi:10.1016/j.jbiomech.2006.04.018

26. Crisco JJ, Panjabi MM. Euler stability of the human ligamentous lumbar spine. Part I: Theory. *Clin Biomech (Bristol, Avon)*. 1992;7(1):19-26. doi:10.1016/0268-0033(92)90003-M

27. Crisco JJ, Panjabi MM, Yamamoto I, Oxland TR. Euler stability of the human ligamentous lumbar spine. Part II: Experiment. *Clin Biomech (Bristol, Avon)*. 1992;7(1):27-32. doi:10.1016/0268-0033(92)90004-N

28. McGill SM, Grenier S, Kavcic N, Cholewicki J. Coordination of muscle activity to assure stability of the lumbar spine. *J Electromyogr Kinesiol*. 2003;13(4):353-359. doi:10.1016/s1050-6411(03)00043-9

29. Arjmand N, Shirazi-Adl A. Model and in vivo studies on human trunk load partitioning and stability in isometric forward flexions. *J Biomech*. 2006;39(3):510-521. doi:10.1016/j.jbiomech.2004.11.030

30. El-Rich M, Shirazi-Adl A, Arjmand N. Muscle activity, internal loads, and stability of the human spine in standing postures: combined model and in vivo studies. *Spine (Phila Pa 1976)*. 2004;29(23):2633-2642. doi:10.1097/01.brs.0000146463.05288.0e

31. Kiefer A, Shirazi-Adl A, Parnianpour M. Synergy of the human spine in neutral postures. *Eur Spine J*. 1998;7(6):471-479. doi:10.1007/s005860050110

32. Shirazi-Adl A, El-Rich M, Pop DG, Parnianpour M. Spinal muscle forces, internal loads and stability in standing under various postures and loads--application of kinematics-based algorithm. *Eur Spine J*. 2005;14(4):381-392. doi:10.1007/s00586-004-0779-0

33. Gardner-Morse MG, Stokes IA. The effects of abdominal muscle coactivation on lumbar spine stability. *Spine (Phila Pa 1976)*. 1998;23(1):86-91; discussion 91-92. doi:10.1097/00007632-199801010-00019

34. Stokes IAF, Gardner-Morse M. Lumbar spinal muscle activation synergies predicted by multi-criteria cost function. *Journal of Biomechanics*. 2001;34(6):733-740. doi:10.1016/S0021-9290(01)00034-3

35. Arjmand N, Shirazi-Adl A. Biomechanics of changes in lumbar posture in static lifting. *Spine (Phila Pa 1976)*. 2005;30(23):2637-2648. doi:10.1097/01.brs.0000187907.02910.4f

36. Bazrgari B, Shirazi-Adl A, Kasra M. Seated whole body vibrations with high-magnitude accelerations--relative roles of inertia and muscle forces. *J Biomech*. 2008;41(12):2639-2646. doi:10.1016/j.jbiomech.2008.06.026

37. Bazrgari B, Shirazi-Adl A, Trottier M, Mathieu P. Computation of trunk equilibrium and stability in free flexion-extension movements at different velocities. *J Biomech*. 2008;41(2):412-421. doi:10.1016/j.jbiomech.2007.08.010

38. Bazrgari B, Shirazi-Adl A, Parnianpour M. Transient analysis of trunk response in sudden release loading using kinematics-driven finite element model. *Clin Biomech (Bristol, Avon)*. 2009;24(4):341-347. doi:10.1016/j.clinbiomech.2009.02.002

39. Bazrgari B, Shirazi-Adl A, Larivière C. Trunk response analysis under sudden forward perturbations using a kinematics-driven model. *J Biomech*. 2009;42(9):1193-1200. doi:10.1016/j.jbiomech.2009.03.014

40. Hajihosseinali M, Arjmand N, Shirazi-Adl A, Farahmand F, Ghiasi MS. A novel stability and kinematics-driven trunk biomechanical model to estimate muscle and spinal forces. *Medical Engineering & Physics*. 2014;36(10):1296-1304. doi:10.1016/j.medengphy.2014.07.009

41. Mohammadi Y, Arjmand N, Shirazi-Adl A. Comparison of trunk muscle forces, spinal loads and stability estimated by one stability- and three EMG-assisted optimization approaches. *Medical Engineering & Physics*. 2015;37(8):792-800. doi:10.1016/j.medengphy.2015.05.018

42. El Ouaaid Z, Shirazi-Adl A, Plamondon A. Trunk response and stability in standing under sagittal-symmetric pull-push forces at different orientations, elevations and magnitudes. *J Biomech*. 2018;70:166-174. doi:10.1016/j.jbiomech.2017.10.008

43. El Ouaaid Z, Shirazi-Adl A, Plamondon A. Effects of variation in external pulling force magnitude, elevation, and orientation on trunk muscle forces, spinal loads and stability. *J Biomech*. 2016;49(6):946-952. doi:10.1016/j.jbiomech.2015.09.036

44. Hajihosseinali M, Arjmand N, Shirazi-Adl A. Effect of body weight on spinal loads in various activities: a personalized biomechanical modeling approach. *J Biomech*. 2015;48(2):276-282. doi:10.1016/j.jbiomech.2014.11.033

45. Cholewicki J, VanVliet JJ. Relative contribution of trunk muscles to the stability of the lumbar spine during isometric exertions. *Clin Biomech (Bristol, Avon)*. 2002;17(2):99-105. doi:10.1016/s0268-0033(01)00118-8

46. Kavcic N, Grenier S, McGill SM. Determining the Stabilizing Role of Individual Torso Muscles During Rehabilitation Exercises: *Spine*. 2004;29(11):1254-1265. doi:10.1097/00007632-200406010-00016

47. Cholewicki J, Simons APD, Radebold A. E!ects of external trunk loads on lumbar spine stability. *Journal of Biomechanics*. Published online 2000:9.

48. Ghiasi MS, Arjmand N, Boroushaki M, Farahmand F. Investigation of trunk muscle activities during lifting using a multi-objective optimization-based model and intelligent optimization algorithms. *Med Biol Eng Comput*. 2016;54(2-3):431-440. doi:10.1007/s11517-015-1327-2

49. Akhavanfar MH, Brandon SCE, Brown SHM, Graham RB. Development of a novel MATLAB-based framework for implementing mechanical joint stability constraints within OpenSim musculoskeletal models. *Journal of Biomechanics*. 2019;91:61-68. doi:10.1016/j.jbiomech.2019.05.007

50. Granata KP, Slota GP, Wilson SE. Influence of Fatigue in Neuromuscular Control of Spinal Stability. *Hum Factors*. 2004;46(1):81-91. doi:10.1518/hfes.46.1.81.30391

51. Zeinali-Davarani S, Shirazi-Adl A, Dariush B, Hemami H, Parnianpour M. The effect of resistance level and stability demands on recruitment patterns and internal loading of spine in dynamic flexion and extension using a simple trunk model. *Computer Methods in Biomechanics and Biomedical Engineering*. 2011;14(7):645-656. doi:10.1080/10255842.2010.493511

52. Zeinali-Davarani S, Hemami H, Barin K, Shirazi-Adl A, Parnianpour M. Dynamic Stability of Spine Using Stability-Based Optimization and Muscle Spindle Reflex. *IEEE Trans Neural Syst Rehabil Eng*. 2008;16(1):106-118. doi:10.1109/TNSRE.2007.906963

53. Davarani SZ, Shirazi-Adl A, Hemami H, Mousavi SJ, Parnianpour M. Dynamic iso-resistive trunk extension simulation: Contributions of the intrinsic and reflexive mechanisms to spinal stability. *THC*. 2007;15(6):415-431. doi:10.3233/THC-2007-15604

54. Cholewicki J, Pm SMM. Mechanical stability of the in viva lumbar spine: implications for injury and chronic low back pain. :15.

55. Crisco JJ, Panjabi MM. The Intersegmental and Multisegmental Muscles of the Lumbar Spine: A Biomechanical Model Comparing Lateral Stabilizing Potential. *Spine*. 1991;16(7):793-799. doi:10.1097/00007632-199107000-00018

56. Samadi S, Arjmand N. A novel stability-based EMG-assisted optimization method for the spine. *Medical Engineering & Physics*. 2018;58:13-22. doi:10.1016/j.medengphy.2018.04.019

57. Shamsi M, Sarrafzadeh J, Jamshidi A, Arjmand N, Ghezelbash F. Comparison of spinal stability following motor control and general exercises in nonspecific chronic low back pain patients. *Clinical Biomechanics*. 2017;48:42-48. doi:10.1016/j.clinbiomech.2017.07.006

58. Vakilzadeh VK. A 3-D Stability-Based Dynamic Computational Model of Human Trunk Movement: Towards the Development of a Paradigm for Forensic Spinal Injury Biomechanical Analysis. *Journal of Forensic Biomechanics*. 2015;06(01). doi:10.4172/2090-2697.1000119

59. Shortz SK, Haas M. Relationship Between Radiographic Lumbosacral Spine Mensuration and Chronic Low Back Pain Intensity: A Cross-sectional Study. *J Chiropr Med*. 2018;17(1):1-6. doi:10.1016/j.jcm.2017.10.005

60. Murrie VL, Dixon AK, Hollingworth W, Wilson H, Doyle T a. C. Lumbar lordosis: study of patients with and without low back pain. *Clin Anat*. 2003;16(2):144-147. doi:10.1002/ca.10114

61. Pesenti S, Lafage R, Stein D, et al. The Amount of Proximal Lumbar Lordosis Is Related to Pelvic Incidence. *Clin Orthop Relat Res*. 2018;476(8):1603-1611. doi:10.1097/CORR.0000000000000380

62. Fernand R, Fox DE. Evaluation of lumbar lordosis. A prospective and retrospective study. *Spine (Phila Pa 1976)*. 1985;10(9):799-803. doi:10.1097/00007632-198511000-00003

63. Liu YK, Laborde JM, Van Buskirk WC. Inertial properties of a segmented cadaver trunk: their implications in acceleration injuries. *Aerosp Med*. 1971;42(6):650-657.

64. Berry JL, Moran JM, Berg WS, Steffee AD. A Morphometric Study of Human Lumbar and Selected Thoracic Vertebrae. *Spine (Phila Pa 1976)*. 1987;12(4):362-367. doi:10.1097/00007632-198705000-00010

65. Panjabi MM, Goel V, Oxland T, et al. Human lumbar vertebrae. Quantitative three-dimensional anatomy. *Spine (Phila Pa 1976)*. 1992;17(3):299-306. doi:10.1097/00007632-199203000-00010

66. Scoles PV, Linton AE, Latimer B, Levy ME, Digiovanni BF. Vertebral body and posterior element morphology: the normal spine in middle life. *Spine (Phila Pa 1976)*. 1988;13(10):1082-1086. doi:10.1097/00007632-198810000-00002

67. Gilad I, Nissan M. A study of vertebra and disc geometric relations of the human cervical and lumbar spine. *Spine (Phila Pa 1976)*. 1986;11(2):154-157. doi:10.1097/00007632-198603000-00010

68. Zhou SH, McCarthy ID, McGregor AH, Coombs RR, Hughes SP. Geometrical dimensions of the lower lumbar vertebrae--analysis of data from digitised CT images. *Eur Spine J*. 2000;9(3):242-248. doi:10.1007/s005860000140

69. Klein A, Nagel K, Gührs J, et al. On the relationship between stature and anthropometric measurements of lumbar vertebrae. *Sci Justice*. 2015;55(6):383-387. doi:10.1016/j.scijus.2015.05.004

70. Ross PD, Wasnich RD, Davis JW, Vogel JM. Vertebral dimension differences between Caucasian populations, and between Caucasians and Japanese. *Bone*. 1991;12(2):107-112. doi:10.1016/8756-3282(91)90008-7

71. Gilsanz V, Boechat MI, Gilsanz R, Loro ML, Roe TF, Goodman WG. Gender differences in vertebral sizes in adults: biomechanical implications. *Radiology*. 1994;190(3):678-682. doi:10.1148/radiology.190.3.8115610

72. Kunkel ME, Herkommer A, Reinehr M, Böckers TM, Wilke HJ. Morphometric analysis of the relationships between intervertebral disc and vertebral body heights: an anatomical and radiographic study of the human thoracic spine. *J Anat*. 2011;219(3):375-387. doi:10.1111/j.1469-7580.2011.01397.x

73. Magee J, McClelland B, Winder J. Current issues with standards in the measurement and documentation of human skeletal anatomy. *J Anat*. 2012;221(3):240-251. doi:10.1111/j.1469-7580.2012.01535.x

74. Panjabi MM, Takata K, Goel V, et al. Thoracic human vertebrae. Quantitative three-dimensional anatomy. *Spine (Phila Pa 1976)*. 1991;16(8):888-901. doi:10.1097/00007632-199108000-00006

75. Nieves JW, Formica C, Ruffing J, et al. Males have larger skeletal size and bone mass than females, despite comparable body size. *J Bone Miner Res*. 2005;20(3):529-535. doi:10.1359/JBMR.041005

76. Stokes IA, Gardner-Morse M, Churchill D, Laible JP. Measurement of a spinal motion segment stiffness matrix. *J Biomech*. 2002;35(4):517-521. doi:10.1016/s0021-9290(01)00221-4

77. Izambert O, Mitton D, Thourot M, Lavaste F. Dynamic stiffness and damping of human intervertebral disc using axial oscillatory displacement under a free mass system. *Eur Spine J*. 2003;12(6):562-566. doi:10.1007/s00586-003-0569-0

78. Loh, Chin-Hsiung, Huang, Yu-Ting, Hsiung, Wan-Ying, Yang, Yuan-Sen, Loh, Kenneth J. Vibration-based identification of rotating blades using Rodrigues' rotation formula from a 3-D measurement. *Wind and Structures*. 2015;21(6):677-691. doi:10.12989/WAS.2015.21.6.677

79. Spong MW, Hutchinson S, Vidyasagar M. *Robot Modeling and Control*. John Wiley & Sons; 2006.

80. Hill AV. The heat of shortening and the dynamic constants of muscle. *Proc R Soc Lond B*. 1938;126(843):136-195. doi:10.1098/rspb.1938.0050

81. Yamaguchi GT. *Dynamic Modeling of Musculoskeletal Motion: A Vectorized Approach for Biomechanical Analysis in Three Dimensions*. Kluwer Academic Publishers; 2001.

82. Ethier CR, Simmons CA. *Introductory Biomechanics: From Cells to Organisms*. 13th printing. Cambridge University Press; 2018.

83. Shibasaki H, Hallett M. *The Neurological Examination: Scientific Basis for Clinical Diagnosis*. Oxford University Press; 2016.

84. Brodal P. *The Central Nervous System: Structure and Function*. 4th ed. Oxford University Press; 2010.

85. Kelly R, Santibáñez V, Loría A. *Control of Robot Manipulators in Joint Space*. Springer; 2005.

86. Winter DA. *Biomechanics and Motor Control of Human Movement*. 4th ed. Wiley; 2009.

87. Jazar RN. *Advanced Dynamics: Rigid Body, Multibody, and Aerospace Applications*. Wiley; 2011.

88. Anderson III FC. *A Dynamic Optimization Solution for a Complete Cycle of Normal Gait*. Thesis. University of Texas at Austin; 1999.

89. Jie Chen. On computing the maximal delay intervals for stability of linear delay systems. *IEEE Trans Automat Contr*. 1995;40(6):1087-1093. doi:10.1109/9.388690

90. Brown SHM, McGill SM. The relationship between trunk muscle activation and trunk stiffness: examining a non-constant stiffness gain. *Computer Methods in Biomechanics and Biomedical Engineering*. 2010;13(6):829-835. doi:10.1080/10255841003630652

91. Morino S, Takahashi M. Estimating Co-Contraction Activation of Trunk Muscles Using a Novel Musculoskeletal Model for Pregnant Women. *Applied Sciences*. 2017;7(10):1067. doi:10.3390/app7101067

92. Liebetrau A, Puta C, Schinowski D, Wulf T, Wagner H. Besteht ein Zusammenhang zwischen Rückenschmerz und der Stabilität der lumbalen Wirbelsäule in der Schwangerschaft?: Eine modellbasierte Hypothese. *Schmerz*. 2012;26(1):36-45. doi:10.1007/s00482-011-1125-1

93. Hoogendoorn WE, Bongers PM, de Vet HC, et al. Flexion and rotation of the trunk and lifting at work are risk factors for low back pain: results of a prospective cohort study. *Spine (Phila Pa 1976)*. 2000;25(23):3087-3092. doi:10.1097/00007632-200012010-00018

94. Bailey JF, Sparrey CJ, Been E, Kramer PA. Morphological and postural sexual dimorphism of the lumbar spine facilitates greater lordosis in females. *J Anat*. 2016;229(1):82-91. doi:10.1111/joa.12451

Appendix A

A.1 Moment of Inertia Equations

The following equations can be used to calculate the mass moment of inertia of an elliptical cylinder:

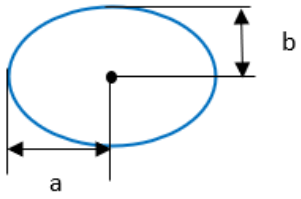


Figure A-1: Elliptical Cylinder Top View

$$I = \frac{1}{4}ma^2 + \frac{1}{3}mL^2$$

Equation A-1: Moment of Inertia about the X-Axis

$$I = \frac{1}{4}mb^2 + \frac{1}{3}mL^2$$

Equation A-2: Moment of Inertia about the Y-Axis

$$I = \frac{1}{4}m(a^2 + b^2)$$

Equation A-3: Moment of Inertia About the Z-Axis

where *a* is the radius along the major axis of the ellipse, *b* is the radius along the minor axis of the ellipse, *m* is the mass of the body, and *L* is the length or vertebral height of the body.

A.2 Lordosis Angles

Pesenti *et al.*[61] measured the cumulative lordosis angles with respect to the superior endplate of S1 for 119 healthy subjects (Table A-1, Column 1). The sacral slope, measured at 38 degrees, defines the angle between the superior endplate of S1 and the horizontal plane[61]. In Column 2 of Table A-1, these angles have been adjusted to their measures with respect to Axis 1, which is parallel to the horizontal plane. This will allow for the lordosis angle of each segment to be applied as a rotation around Axis 1 (Figure A-2). This is the method that has been used previously to implement lordosis in the Franklin *et al.*[24] model.

Table A-1: Development of the Normal Lordosis Case

Column 1 of this table indicates the cumulative lordosis provided in Pesenti *et al*.[61] Adjusting the pelvis angle to -38°, the segmental lordosis for each of the lumbar vertebrae can be redefined using this reference (Column 2). Column 3 represents the lordosis angles Franklin utilized in the original model[24]

	Pesenti *et al.*[61] Cumulative Lordosis* (deg)	**Pesenti *et al.*[61] Adjusted Lordosis (deg)**	**Franklin *et al.*[24] Lordosis (deg)**
L1	58.1	20.1	10
L2	55.5	17.5	15
L3	47.6	9.6	18
L4	36	-2	8
L5	20.2	-17.8	-14
Pelvis	0	-38	-20

*Measuring the lordotic angle of each vertebra with respect to the superior endplate of S1.

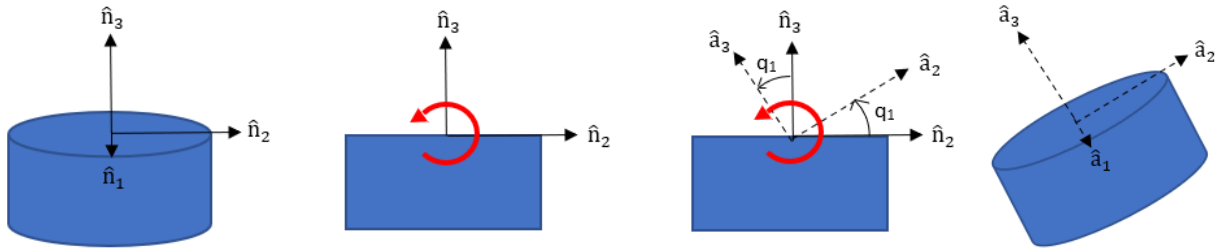


Figure A-2: Lordosis angle applied around Axis 1

In their study, Pesenti *et al.* measured the lordosis angles of 81 women and 38 men[61]. If it is determined that sex impacts skeletal anatomy and lumbar lordosis measures significantly, then the mean lordosis angles reported in the Pesenti *et al.* study may be skewed. At this point in time, results are inconclusive and future investigation is required[94].

The lordosis angles used previously by Franklin *et al*. are presented in Column 3 of Table A-2.

Additionally, for our investigation of the possible impact of lordosis angle on spine stability, the segmental lordosis angles for each case can be seen in Table A-3.

Table A-2: Segmental lordosis angles for all lordosis cases studied in Chapter 5

	Hyperlordosis 89°	**Hyperlordosis 84.1°**	**Normal Lordosis 58.1°**	**Hypolordosis 32.1°**	**Hypolordosis 10°**
L0	0	0	0	0	0
L1	51	46.1	20.1	-5.9	-28
L2	33.2	42.736	17.5	-7.184	-30
L3	15.4	30.962	9.6	-11.678	-32
L4	-2.4	14.142	-2	-18.098	-34
L5	-20.2	-8.565	-17.8	-26.765	-36
Pelvis	-38	-38	-38	-38	-38

A.3 Skeletal and Muscle Coordinates

The skeletal coordinates and muscle attachment points provided by Cholewicki *et al.* [15] were used. Adjustments were made to the muscle attachment points based on the deduced methodology used by Franklin[24] (transformation of coordinates discussed in 2.1.2 Body Origins). Franklin provides the muscle attachment and skeletal geometry coordinates used in the appendix of this thesis[24], but due to discrepancies between a few of the transformed coordinates, the coordinates used in this work are provided in Tables A-3 and A-4. Attachment points *RIB10*, *RIB21*, and *RIB22* differ between the two works. This is further discussed in Chapter 3. Additionally, the muscle attachment point coordinates used in our work (Table A-4) differ from that provided in Franklin's thesis[24]; it appears that Franklin's work may have had data input errors as the nodal points are not associated with the appropriate muscle fascicle specified by Cholewicki *et al.*[15] Additionally, RPEL1 and LPEL1 were adjusted to (0,0,0), acting as the origin of the pelvis.

The global coordinates were transformed to the local coordinate systems through the following process, where Equation A-1 represents translation and Equation A-2 represents rotation in the transformation:

$$coord_{body,local} = coord_{body,global} - origin_{body,global} \quad \text{(A-4)}$$

where $coord_{b,global}$ is the muscle attachment point in global coordinates, $origin_{b,global}$ is the origin of the body in global coordinates, and $coord_{b,local}$ is the muscle attachment point in local coordinates.

$$coord_{body,local} = R1 * coord_{body,local} \quad \text{(A-5)}$$

where *R1* is the rotation matrix for a rotation around Axis 1.

Figure A-3 presents the relationships between the defined axes for Franklin[24] and Cholewicki *et al.*[15]:

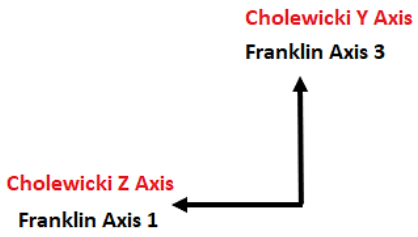


Figure A-3: Franklin vs Cholewicki Axes Definition. Franklin Axis 2 and Cholewicki X Axis point anteriorly

Table A-3: Skeletal Geometry Coordinates

These coordinates originate from Cholewicki *et al*.[15] Each measure is presented in meters.

Nomenclature: In the coordinate name, the first letter signifies what side of the body the point is on (R=right, L=left), followed by the body the coordinate is associated with, followed by the coordinate number. For example, RPEL1: This attachment point is located on the right-side of the body and is the first point associated with the pelvis.

RPEL1	0.0000	0.0000	0.0000	Origin	**LPEL1**	0.0000	0.0000	0.0000
RPEL2	0.0000	0.0000	0.1065	Connect L5	**LPEL2**	0.0000	0.0000	0.1065
RPEL3	0.0300	0.0780	-0.0254		**LPEL3**	-0.0300	0.0780	-0.0254
RPEL4	0.1300	0.0350	0.1152		**LPEL4**	-0.1300	0.0350	0.1152
RPEL5	0.0000	0.0839	-0.0260		**LPEL5**	0.0000	0.0839	-0.0260
RPEL6	0.1250	-0.0001	0.1477		**LPEL6**	-0.1250	-0.0001	0.1477
RPEL7	0.1200	0.0644	0.0863		**LPEL7**	-0.1200	0.0644	0.0863
RPEL8	0.1300	0.0350	0.1152		**LPEL8**	-0.1300	0.0350	0.1152
RPEL9	0.0600	-0.0693	0.1170		**LPEL9**	-0.0600	-0.0693	0.1170
RPEL10	0.0680	-0.0804	0.1060		**LPEL10**	-0.0680	-0.0804	0.1060
RPEL11	0.0330	-0.0805	0.1050		**LPEL11**	-0.0330	-0.0805	0.1050

RPEL12	0.0900	-0.0301	0.1495		**LPEL12**	-0.0900	-0.0301	0.1495
RPEL13	0.0360	-0.0671	0.1188		**LPEL13**	-0.0360	-0.0671	0.1188
RPEL14	0.0150	-0.0771	0.0776		**LPEL14**	-0.0150	-0.0771	0.0776
RPEL15	0.0300	-0.0561	0.1298		**LPEL15**	-0.0300	-0.0561	0.1298
RPEL16	0.0600	-0.0419	0.1516		**LPEL16**	-0.0600	-0.0419	0.1516
RPEL17	0.0820	0.0441	-0.0222		**LPEL17**	-0.0820	0.0441	-0.0222
RPEL18	0.0000	-0.0814	0.1061		**LPEL18**	0.0000	-0.0814	0.1061
RPEL19	0.0620	-0.0364	0.1461		**LPEL19**	-0.0620	-0.0364	0.1461
RL51	0.0000	0.0000	0.0000	Origin	**LL51**	0.0000	0.0000	0.0000
RL52	0.0000	0.0000	0.0389	Connect L4	**LL52**	0.0000	0.0000	0.0389
RL53	0.0500	-0.0264	0.0230		**LL53**	-0.0500	-0.0264	0.0230
RL54	0.0020	-0.0606	0.0119		**LL54**	-0.0020	-0.0606	0.0119
RL55	0.0050	-0.0591	0.0071		**LL55**	-0.0050	-0.0591	0.0071
RL56	0.0150	-0.0395	0.0051		**LL56**	-0.0150	-0.0395	0.0051
RL57	0.0230	-0.0119	0.0108		**LL57**	-0.0230	-0.0119	0.0108
RL58	0.0000	0.0192	0.0251		**LL58**	0.0000	0.0192	0.0251
RL59	0.0000	-0.0162	0.0242		**LL59**	0.0000	-0.0162	0.0242
RL510	0.0000	-0.0251	0.0287		**LL510**	0.0000	-0.0251	0.0287
RL511	0.0360	-0.0264	0.0230		**LL511**	-0.0360	-0.0264	0.0230
RL512	0.0200	-0.0257	0.0337		**LL512**	-0.0200	-0.0257	0.0337
RL513	0.0240	-0.0351	0.0370		**LL513**	-0.0240	-0.0351	0.0370
RL514	0.0000	-0.0561	0.0175		**LL514**	0.0000	-0.0561	0.0175
RL515	0.0000	-0.0434	0.0238		**LL515**	0.0000	-0.0434	0.0238
RL516	0.0000	-0.0384	0.0275		**LL516**	0.0000	-0.0384	0.0275
RL41	0.0000	0.0000	0.0000	Origin	**LL41**	0.0000	0.0000	0.0000
RL42	0.0000	0.0000	0.0370	Connect L3	**LL42**	0.0000	0.0000	0.0370

RL43	0.0400	-0.0320	0.0230		**LL43**	-0.0400	-0.0320	0.0230
RL44	0.0020	-0.0680	0.0130		**LL44**	-0.0020	-0.0680	0.0130
RL45	0.0440	-0.0340	0.0270		**LL45**	-0.0440	-0.0340	0.0270
RL46	0.0050	-0.0650	0.0040		**LL46**	-0.0050	-0.0650	0.0040
RL47	0.0150	-0.0390	0.0010		**LL47**	-0.0150	-0.0390	0.0010
RL48	0.0020	-0.0700	0.0130		**LL48**	-0.0020	-0.0700	0.0130
RL49	0.0230	-0.0140	0.0070		**LL49**	-0.0230	-0.0140	0.0070
RL410	0.0500	-0.0810	0.0120		**LL410**	-0.0500	-0.0810	0.0120
RL411	0.0740	-0.0810	0.0120		**LL411**	-0.0740	-0.0810	0.0120
RL412	0.0380	-0.0810	0.0120		**LL412**	-0.0380	-0.0810	0.0120
RL413	0.0150	-0.0810	0.0120		**LL413**	-0.0150	-0.0810	0.0120
RL414	0.0000	0.0180	0.0110		**LL414**	0.0000	0.0180	0.0110
RL415	0.0000	-0.0190	0.0120		**LL415**	0.0000	-0.0190	0.0120
RL416	0.0000	-0.0260	0.0150		**LL416**	0.0000	-0.0260	0.0150
RL417	0.0360	-0.0320	0.0230		**LL417**	-0.0360	-0.0320	0.0230
RL418	0.0200	-0.0340	0.0100		**LL418**	-0.0200	-0.0340	0.0100
RL419	0.0160	-0.0340	0.0090		**LL419**	-0.0160	-0.0340	0.0090
RL420	0.0000	-0.0700	0.0090		**LL420**	0.0000	-0.0700	0.0090
RL421	0.0000	-0.0600	0.0070		**LL421**	0.0000	-0.0600	0.0070
RL422	0.0000	-0.0700	0.0120		**LL422**	0.0000	-0.0700	0.0120
RL423	0.0000	0.0160	0.0030		**LL423**	0.0000	0.0160	0.0030
RL424	0.0000	0.0010	0.0060		**LL424**	0.0000	0.0010	0.0060
RL425	0.0000	-0.0160	0.0060		**LL425**	0.0000	-0.0160	0.0060
RL31	0.0000	0.0000	0.0000	Origin	**LL31**	0.0000	0.0000	0.0000
RL32	0.0000	0.0000	0.0379	Connect L2	**LL32**	0.0000	0.0000	0.0379
RL33	0.0300	-0.0324	0.0254		**LL33**	-0.0300	-0.0324	0.0254

RL34	0.0020	-0.0715	0.0195		**LL34**	-0.0020	-0.0715	0.0195
RL35	0.0380	-0.0388	0.0288		**LL35**	-0.0380	-0.0388	0.0288
RL36	0.0050	-0.0662	0.0061		**LL36**	-0.0050	-0.0662	0.0061
RL37	0.0150	-0.0431	0.0042		**LL37**	-0.0150	-0.0431	0.0042
RL38	0.0020	-0.0734	0.0200		**LL38**	-0.0020	-0.0734	0.0200
RL39	0.0210	-0.0165	0.0087		**LL39**	-0.0210	-0.0165	0.0087
RL310	0.0760	-0.0832	0.0221		**LL310**	-0.0760	-0.0832	0.0221
RL311	0.0390	-0.0832	0.0221		**LL311**	-0.0390	-0.0832	0.0221
RL312	0.0150	-0.0832	0.0221		**LL312**	-0.0150	-0.0832	0.0221
RL21	0.0000	0.0000	0.0000	Origin	**LL21**	0.0000	0.0000	0.0000
RL22	0.0000	0.0000	0.0386	Connect L1	**LL22**	0.0000	0.0000	0.0386
RL23	0.0270	-0.0337	0.0236		**LL23**	-0.0270	-0.0337	0.0236
RL24	0.0020	-0.0710	0.0138		**LL24**	-0.0020	-0.0710	0.0138
RL25	0.0380	-0.0398	0.0275		**LL25**	-0.0380	-0.0398	0.0275
RL26	0.0050	-0.0678	0.0035		**LL26**	-0.0050	-0.0678	0.0035
RL27	0.0150	-0.0396	0.0003		**LL27**	-0.0150	-0.0396	0.0003
RL28	0.0020	-0.0729	0.0144		**LL28**	-0.0020	-0.0729	0.0144
RL29	0.0200	-0.0187	0.0108		**LL29**	-0.0200	-0.0187	0.0108
RL210	0.0780	-0.0822	0.0182		**LL210**	-0.0780	-0.0822	0.0182
RL211	0.0410	-0.0822	0.0182		**LL211**	-0.0410	-0.0822	0.0182
RL212	0.0150	-0.0822	0.0182		**LL212**	-0.0150	-0.0822	0.0182
RL11	0.0000	0.0000	0.0000	Origin	**LL11**	0.0000	0.0000	0.0000
RL12	0.0000	0.0000	0.0362	Connect L0	**LL12**	0.0000	0.0000	0.0362
RL13	0.0260	-0.0367	0.0233		**LL13**	-0.0260	-0.0367	0.0233
RL14	0.0020	-0.0714	0.0193		**LL14**	-0.0020	-0.0714	0.0193
RL15	0.0360	-0.0395	0.0267		**LL15**	-0.0360	-0.0395	0.0267

RL16	0.0050	-0.0674	0.0087		**LL16**	-0.0050	-0.0674	0.0087
RL17	0.0020	-0.0732	0.0201		**LL17**	-0.0020	-0.0732	0.0201
RL18	0.0190	-0.0187	0.0096		**LL18**	-0.0190	-0.0187	0.0096
RL19	0.0800	-0.0823	0.0242		**LL19**	-0.0800	-0.0823	0.0242
RL110	0.0430	-0.0823	0.0242		**LL110**	-0.0430	-0.0823	0.0242
RL111	0.0150	-0.0823	0.0242		**LL111**	-0.0150	-0.0823	0.0242
RRIB1	0.0000	0.0000	0.0000	Origin	**LRIB1**	0.0000	0.0000	0.0000
RRIB2	0.0000	0.0000	0.3250	Connect C7	**LRIB2**	0.0000	0.0000	0.3250
RRIB3	0.0700	0.1180	-0.0039		**LRIB3**	-0.0700	0.1180	-0.0039
RRIB4	0.1250	-0.0115	-0.0551		**LRIB4**	-0.1250	-0.0115	-0.0551
RRIB5	0.1050	0.0534	-0.0395		**LRIB5**	-0.1050	0.0534	-0.0395
RRIB6	0.0700	0.0786	-0.0643		**LRIB6**	-0.0700	0.0786	-0.0643
RRIB7	0.0000	0.1178	0.0261		**LRIB7**	0.0000	0.1178	0.0261
RRIB8	0.0840	-0.0563	0.0345		**LRIB8**	-0.0840	-0.0563	0.0345
RRIB9	0.0500	-0.0528	0.0845		**LRIB9**	-0.0500	-0.0528	0.0845
RRIB10	0.0200	-0.0537	0.1795		**LRIB10**	-0.0200	-0.0537	0.1795
RRIB11	0.0200	-0.0541	0.2245		**LRIB11**	-0.0200	-0.0541	0.2245
RRIB12	0.0200	-0.0495	0.2696		**LRIB12**	-0.0200	-0.0495	0.2696
RRIB13	0.0200	-0.0499	0.3116		**LRIB13**	-0.0200	-0.0499	0.3116
RRIB14	0.0200	-0.0500	0.3296		**LRIB14**	-0.0200	-0.0500	0.3296
RRIB15	0.0720	-0.0370	-0.0003		**LRIB15**	-0.0720	-0.0370	-0.0003
RRIB16	0.0050	-0.0539	-0.0135		**LRIB16**	-0.0050	-0.0539	-0.0135
RRIB17	0.1200	0.0169	0.1152		**LRIB17**	-0.1200	0.0169	0.1152
RRIB18	0.0650	-0.0599	-0.0106		**LRIB18**	-0.0650	-0.0599	-0.0106
RRIB19	0.0650	-0.0476	-0.0384		**LRIB19**	-0.0650	-0.0476	-0.0384
RRIB20	0.0650	-0.0414	-0.0654		**LRIB20**	-0.0650	-0.0414	-0.0654

RRIB21	0.0650	-0.0352	-0.0913	**LRIB21**	-0.0650	-0.0352	-0.0913
RRIB22	0.0650	-0.0350	-0.1133	**LRIB22**	-0.0650	-0.0350	-0.1133
RRIB23	0.0150	-0.0699	-0.0156	**LRIB23**	-0.0150	-0.0699	-0.0156
RRIB24	0.0000	0.1098	0.0220	**LRIB24**	0.0000	0.1098	0.0220
RRIB25	0.0000	0.0428	0.0174	**LRIB25**	0.0000	0.0428	0.0174
RRIB26	0.0000	-0.0259	-0.0072	**LRIB26**	0.0000	-0.0259	-0.0072

Table A-4: Muscle Attachment Points

The muscle attachment points and physiological cross-sectional areas are from Cholewicki *et al.*[15]

Muscle	**Origin**	**Terminal**	**Node1**	**Node2**	**Node3**	**Node4**	**Node5**	**Node6**	**PCSA (cm^2)**
RRectAb	RPEL3	RRIB3							10
RExtOb1	RPEL4	RRIB4							10
RExtOb2	RPEL5	RRIB5							9
RIntOb1	RPEL6	RRIB6							9
RIntOb2	RPEL7	RRIB7							8
RParsL5	RPEL9	RL53							5.6
RParsL4	RPEL9	RL43							6
RParsL3	RPEL9	RL33							5.7
RParsL2	RPEL9	RL23	RL410						5
RParsL1	RPEL9	RL13	RL410						4
RIlioLum	RPEL10	RRIB8	RL411	RL310	RL210	RL19			11
RLongTP	RPEL11	RRIB9	RL412	RL311	RL211	RL110			16.8
RLongTL5	RL54	RRIB10	RL413	RL312	RL212	RL111	RRIB23		0.7
RLongTL4	RL44	RRIB11	RL312	RL212	RL111	RRIB23			1.8
RLongTL3	RL34	RRIB12	RL212	RL111	RRIB23				1.2
RLongTL2	RL24	RRIB13	RL111	RRIB23					1.5
RLongTL1	RL14	RRIB14							1.3
RQuadLP	RPEL12	RRIB15							2

RQuadL1	RPEL12	RL15		1
RQuadL2	RPEL12	RL25		1
RQuadL3	RPEL12	RL35		1
RQuadL4	RPEL12	RL45		1
RLatDP	RPEL16	RRIB17		2
RLatDL5	RPEL15	RRIB17	RRIB22	2
RLatDL4	RL48	RRIB17	RRIB21	4
RLatDL3	RL38	RRIB17	RRIB20	4
RLatDL2	RL28	RRIB17	RRIB19	4
RLatDL1	RL17	RRIB17	RRIB18	4
RMultPL5	RPEL14	RL55		1.4
RMultPL4	RPEL14	RL46		2.9
RMultPL3	RPEL13	RL36		2.4
RMultPL2	RPEL13	RL26		1.5
RMultPL1	RPEL13	RL16		0.9
RMultL5L3	RL56	RL36		0.8
RMultL5L2	RL56	RL26		0.6
RMultL5L1	RL56	RL16		0.6
RMultL4L2	RL47	RL26		0.6
RMultL4L1	RL47	RL16		0.6
RMultL3L1	RL37	RL16		0.6
RMultL2T12	RL27	RRIB16		0.6
RPsoaL5	RPEL17	RL57		4.4
RPsoaL4	RPEL17	RL49		4.4
RPsoaL3	RPEL17	RL39		4.4
RPsoaL2	RPEL17	RL29		4.4
RPsoaL1	RPEL17	RL18		4.4
LRectAb	LPEL3	LRIB3		10
LExtOb1	LPEL4	LRIB4		10
LExtOb2	LPEL5	LRIB5		9
LIntOb1	LPEL6	LRIB6		9
LIntOb2	LPEL7	LRIB7		8

LParsL5	LPEL9	LL53						5.6
LParsL4	LPEL9	LL43						6
LParsL3	LPEL9	LL33						5.7
LParsL2	LPEL9	LL23	LL410					5
LParsL1	LPEL9	LL13	LL410					4
LIlioLum	LPEL10	LRIB8	LL411	LL310	LL210	LL19		11
LLongTP	LPEL11	LRIB9	LL412	LL311	LL211	LL110		16.8
LLongTL5	LL54	LRIB10	LL413	LL312	LL212	LL111	LRIB23	0.7
LLongTL4	LL44	LRIB11	LL312	LL212	LL111	LRIB23		1.8
LLongTL3	LL34	LRIB12	LL212	LL111	LRIB23			1.2
LLongTL2	LL24	LRIB13	LL111	LRIB23				1.5
LLongTL1	LL14	LRIB14						1.3
LQuadLP	LPEL12	LRIB15						2
LQuadL1	LPEL12	LL15						1
LQuadL2	LPEL12	LL25						1
LQuadL3	LPEL12	LL35						1
LQuadL4	LPEL12	LL45						1
LLatDP	LPEL16	LRIB17						2
LLatDL5	LPEL15	LRIB17	LRIB22					2
LLatDL4	LL48	LRIB17	LRIB21					4
LLatDL3	LL38	LRIB17	LRIB20					4
LLatDL2	LL28	LRIB17	LRIB19					4
LLatDL1	LL17	LRIB17	LRIB18					4
LMultPL5	LPEL14	LL55						1.4
LMultPL4	LPEL14	LL46						2.9
LMultPL3	LPEL13	LL36						2.4
LMultPL2	LPEL13	LL26						1.5
LMultPL1	LPEL13	LL16						0.9
LMultL5L3	LL56	LL36						0.8
LMultL5L2	LL56	LL26						0.6
LMultL5L1	LL56	LL16						0.6
LMultL4L2	LL47	LL26						0.6

LMultL4L1	LL47	LL16	0.6
LMultL3L1	LL37	LL16	0.6
LMultL2T12	LL27	LRIB16	0.6
LPsoaL5	LPEL17	LL57	4.4
LPsoaL4	LPEL17	LL49	4.4
LPsoaL3	LPEL17	LL39	4.4
LPsoaL2	LPEL17	LL29	4.4
LPsoaL1	LPEL17	LL18	4.4

A.4 Rotations

Each vertebra is rotated around Axis 1, followed by Axis 2, and lastly by Axis 3. The counterclockwise direction is defined as positive.

A rotation is applied around Axis 1($\hat{n}_1$) by an angle q_1 (Figure A-4).

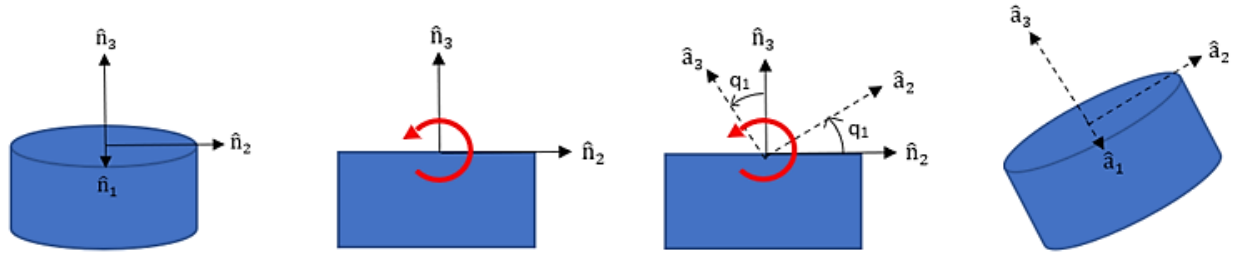


Figure A-4: Rotation around Axis 1

The following rotation matrix defines the angular relationship between the N and A bases:

$$ {}^{N}R^{A} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(q_1) & -\sin(q_1) \\ 0 & \sin(q_1) & \cos(q_1) \end{bmatrix} \tag{A-6} $$

A rotation is applied around Axis 2($\hat{a}_2$) by an angle q_2 (Figure A-5).

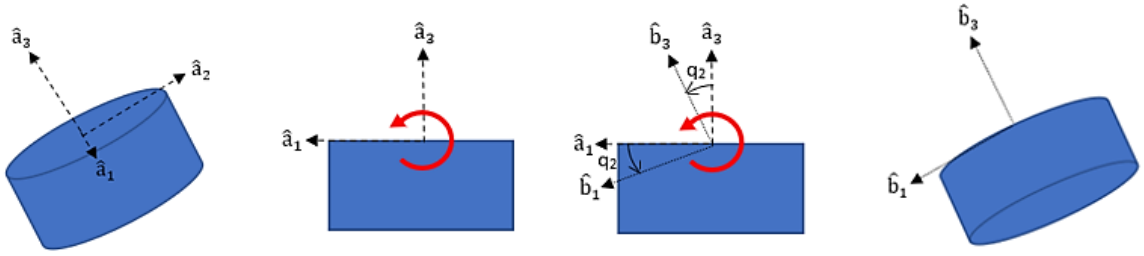


Figure A-5: Rotation around Axis 2

The following rotation matrix defines the angular relationship between the A and B bases:

$$ {}^{A}R^{B} = \begin{bmatrix} \cos(q_2) & 0 & \sin(q_2) \\ 0 & 1 & 0 \\ -\sin(q_2) & 0 & \cos(q_2) \end{bmatrix} \tag{A-7} $$

A rotation is applied around Axis 3($\hat{b}_3$) by an angle q_3 (Figure A-6).

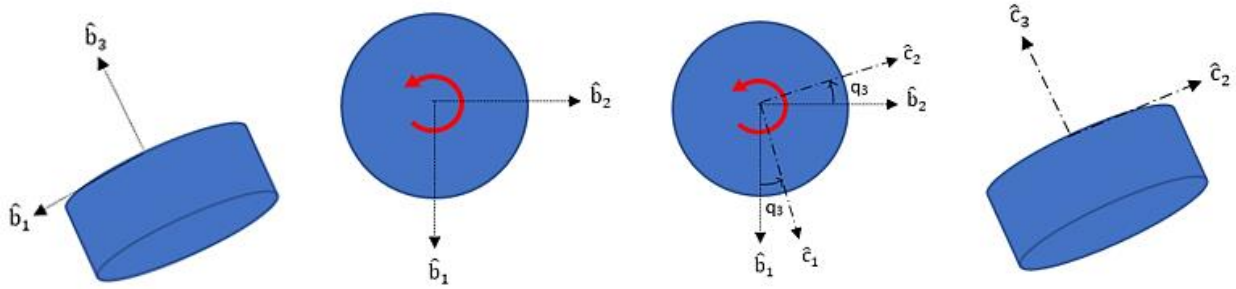


Figure A-6: Rotation around Axis 3

The following rotation matrix defines the angular relationship between the B and C bases:

$$ {}^{B}R^{C} = \begin{bmatrix} \cos(q_3) & -\sin(q_3) & 0 \\ \sin(q_3) & \cos(q_3) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{A-8} $$

A rotation matrix was determined that would allow for all three rotations to be applied to the vertebra with one matrix[81]:

$$ {}^{N}R^{C} = {}^{N}R^{A} * {}^{A}R^{B} * {}^{B}R^{C} \tag{A-9} $$

The matrix ${}^{N}R^{C}$ is the rotation matrix from basis *N* to basis *C*.

Appendix B

The following flowchart provides a general overview of the model processes. This was designed to aid the reader in their understanding of the model and its functions. Appendix C and D provide the scripts and files utilized in the model.

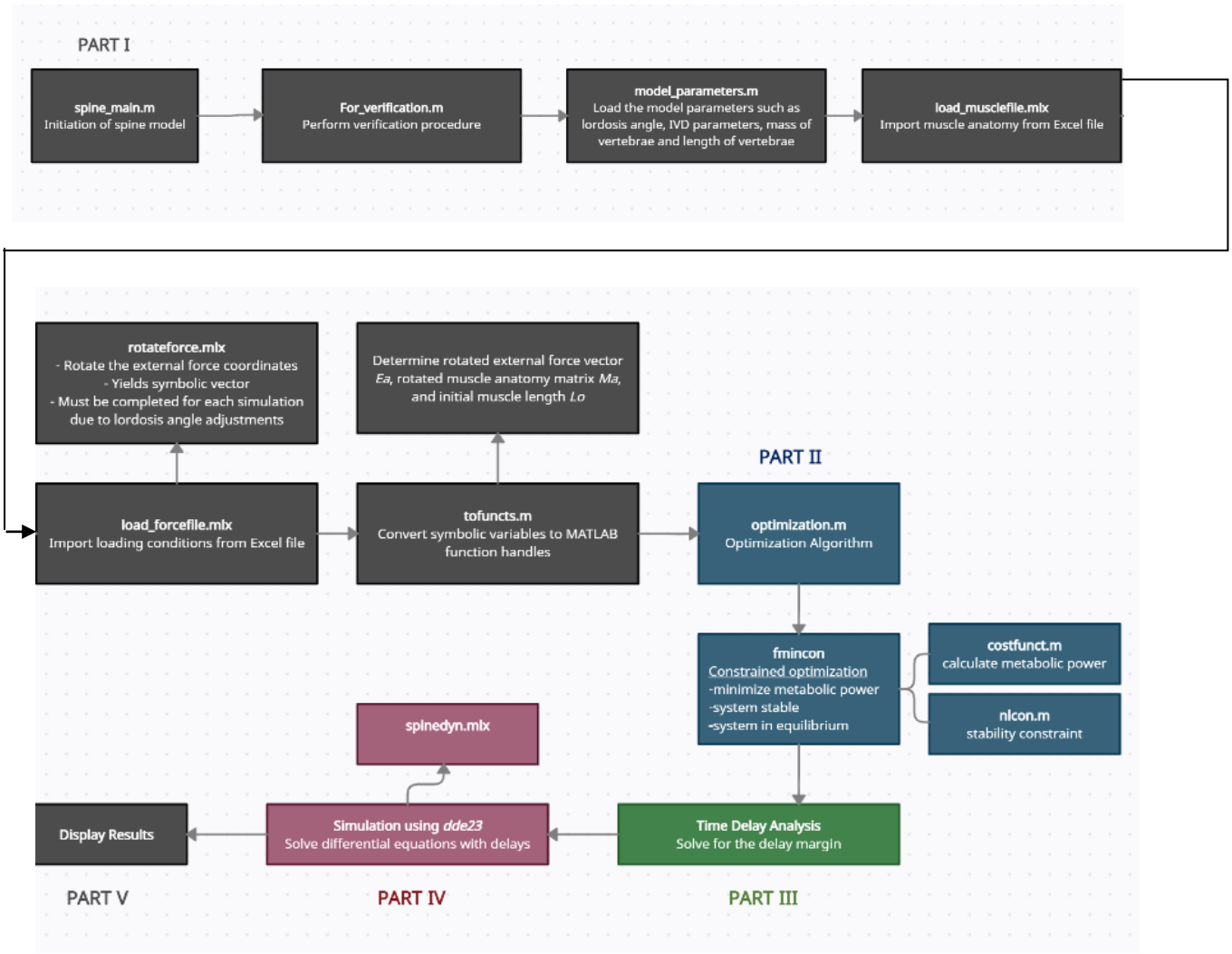


Figure B-1: General Model Flowchart

B.1 Code Directory

Table B-1: Code Directory of the scripts utilized in the model with a general description of their purpose

Script	Description
costfun.m	Calculates the metabolic power required in the optimization procedure
For_verification.m	Performs MATLAB-based verification tasks
load_musclefile.m	Imports the muscle anatomy from Excel spreadsheet
model_parameters.m	Script that contains the models parameters
optimize.m	Calculate the constraints for the constrained optimization procedure; call *run_fmincon*, and determine optimal solution
spine_main.m	Main spine model file
spineacc.m	Acceleration used for constructing the Jacobians
spineaccA.m	Acceleration used for equilibrium constraint in optimization
spineaccP.m	Acceleration used for equilibrium constraint in optimization
tdfind.m	Calculate the time delay
tdfmax.m	Determine the maximum frequency for time delay analysis
tdinvest.m	Main time delay file
tofuncts.m	Convert symbolic outputs to function handles
vect2ang.m	Input the Stateo and Pango vectors and output individual angles based on these vectors
verify.m	Used to compare function inputs and determine absolute error between the inputs
dynam.mlx	Perform the Lagrangian Derivative. Determine the symbolic generalized force partial derivatives, mass matrix, gravity vector, and Coriolis vector
generalizedForce.mlx	Perform generalized force calculations
IVD_calcs.mlx	Calculate the intervertebral disc moments
load_forcefile.mlx	Import the load details from Excel spreadsheet
musclemodel.mlx	Muscle model
musclemodelR.mlx	Reflex model
nlcon.mlx	Nonlinear constraints for constrained optimization
origin_body.mlx	Determine the symbolic origins of the bodies
replace_sym.mlx	Function used to replace symbolic variables with numeric values from the Stateo, Pango, and Vars vectors
rotate_anatomy_sym.mlx	Determine the symbolic rotated muscle anatomy, muscle length and muscle velocity
rotateforce.mlx	Determine the symbolic rotated external force vector
rotationmatrices.mlx	Determine the symbolic rotation matrices
run_fmincon.mlx	Run the constrained optimization procedure
spinedyn.mlx	Function handle used in delay differential equation solver
sub_timevar.mlx	Replaces angles in variables with time-dependent angles
transform_Cholewicki.mlx	Transform the Cholewicki *et al*. anatomy[54] and align vertebral connection points

B.2 MAT-file Directory

Table B-2: MAT-file directory for the .mat files utilized in the model. Many outputs of functions were saved as symbolic .mat files, allowing for these outputs to be updated to function handles in the tofuncts.m file

MAT-file	Description
dynam_sym_output.mat	Symbolic dynamics file variables
Ea_sym.mat	Symbolic rotated external force vector
model_parameters_mat.mat	Model parameters defined
Model_Parameters_Verify.mat	Model parameters utilized for verification
optimization_varsA.mat	Optimization variables used to increase efficiency
origin_sym_0toP.mat	Symbolic origin 7x3 matrix; first row describes the origin of body 0, continues in order, and last row ends with the pelvis origin
origin_sym_Pto0.mat	Symbolic origin 7x3 matrix; first row describes the origin of the pelvis, continues in order, and last row ends with body 0
origin_sym_separate.mat	Separate symbolic origin vectors for each body
rotate_anatomy_sym_MaMLMV.mat	Symbolic outputs for rotated muscle anatomy, muscle length and muscle velocity
rotation_matrices_sym_0toP.mat	Symbolic rotation 21x3 matrix; first three rows describe the rotation matrix of body 0, continues in order, and last three rows describe the pelvis' rotation matrix
rotation_matices_sym_Pto0.mat	Symbolic rotation 21x3 matrix; first three rows describe the rotation matrix of the pelvis, continues in order, and last three rows describe the rotation matrix for body 0
rotation_matrices_sym_separate.mat	Separate symbolic rotation matrices for each body
Stateo_Pango_Rand.mat	Stateo and Pango vectors for verification
transformed_Cholewicki.mat	Muscle coordinates after the transformation that aligns the vertebral bodies along the vertical axis
verification_initialize_vars.mat	Load the variables that were defined prior to the verification procedure; all preceding variables are cleared during the verification procedure

B.3 Personal Communication Summary

Through personal communication with the Musculoskeletal Biomechanics Laboratory at Virginia Polytechnic Institute (Principal Investigators: Dr. Kevin P. Granata and Dr. Michael Madigan), the Human Motion Control Laboratory at the University of Kansas received some of the files developed by Franklin *et al*. for the model (Michael Madigan, personal communication, July 26, 2008). It was discovered that the files in our possession were not the finalized versions.

Throughout the redevelopment process, the model files were reconstructed and may vary from the work of Franklin[16,17,24]. Franklin's files were used as a reference, mostly to ensure that the model was being developed following a similar methodology, but the majority of the files were redrafted and verified. The exception to this would be the time delay analysis codes sourced from the appendix of Franklin's thesis. Therefore, while many of the scripts and function may be similar, there may be differences in the files as well. Additionally, we did not utilize the MEX files that Franklin developed in our model, outside of for verification procedures. While this did increase run time for the model, this allowed for the files to be redrafted with more of an explanation as to what the purpose of the procedure was and the methodology. In some cases, the MEX files provided a symbolic output where the input values just needed to be substituted in. This did not allow for a thorough explanation of the methodology. Additionally, when possible, the symbolic output for certain calculations would be saved in .mat files to increase efficiency of the MATLAB calculations. These symbolic outputs can be converted to MATLAB functions using the MATLAB *matlabFunction* as well to increase efficiency.

Table B-3: Summary of the MATLAB files received from Dr. Michael Madigan and the files Franklin has published in his thesis

	File Names Provisional file names are **bolded.** Non-bolded file names indicate the name used in Franklin's thesis appendix.	**HMCL possesses provisional file**	**Published in Franklin's Thesis**[24]
Spine Acceleration Functions	**AccelerationMP**, **AccelerationSP**, **AccelerationMA**, spineaccP, spineaccA, spineaccR	X	
Optimization	**alg_optimize**, **alg_optimizeR**, optimizationscript	X	X
Cholewicki	**CholeMusc.xls**, printed in thesis	X	X

Transformed Muscle File	Appendix		
Dynamic Simulation File	**Dynamicsim**, spinedyn	X	X
Dynamics	**Dynmatsc.mexw64**	X	X*
Main File	**Execsim**, **EXECSIM**, **SolverP**, **Simulation**, Spinesim	X	X
Used in dde23 Solver	**Edcheck**, Extrema	X	
Import External Force Coordinates from Excel	**Forcinput, Forcmat**	X	
Generalized Forces Calculations	**Geneforcesc.mexw64**	X	
Muscle Maintanance Heat Rate	**helperfunctC**, costfun	X	X
Intervertebral Disc Calculations	**Intervertc.mexw64**	X	
Kinematics	**kinematicsc.mexw64/analys1.mexw64**	X	X*
External Force Excel File	**LoadLift.xls**	X	
Import Muscle Anatomy from	**muscinput, muscmat**	X	

Excel Spreadsheet

Muscle Model	**Muscmodel**	X	
Nonlinear constraints for optimization procedure	**fminconhelper1**, nlcon	X	X
Model Parameters	**Para**, **Anatomvert**, included in spinesim file	X	X
Adjusting coordinates	**Pointscope1**, **Pointscope2**	X	
Determining length of the muscles	**Propmusc**	X	
Rotation Matrices	**Rotate1**, **Rotate2**, **Rotate3**, **Rotate4**, **Rotate5**, **Rotate6**, **Rotate7**, rotation matrices included in **kinematicsc.mex file**	X	
Simulation Plots	Simgraphs	X	
Time Delay Analysis	**Tdfind**, tdfind	X	X
	Tdfmax, tdfmax	X	X
	Tdinvest, tdinvest	X	X
Model Visualization 3D Plot	vischeme	X	

*Franklin provided Mathematica code in his appendix for some of the variables included in this MEX file.

Appendix C: **Main MATLAB Files**

The following MATLAB files and methodology were based on the work of Franklin *et al.*[16,17,24] Franklin's files were obtained through means of personal communication (Michael Madigan, personal communication, July 26, 2008) and from the appendix of his thesis[24]

C.1 *spine_main.m*: Main Spine File

```
% Valerie Jardon
% Department of Bioengineering, University of Kansas
% spine_main.m
% Last Edited: 4/29/22
%% Description
% This is the main script used to execute the spine simulation of the
% biomechanical model.

% This file (spine_main.m) is based on the spinesim.m code developed by
% Timothy C. Franklin, published in Appendix B of his thesis (Franklin,
% 2006, p. 99-101).

% Reference:
% Franklin, T. C. (2006). Linear System Analyses of the Role of Reflex Gain and Delay ↙
in a Dynamic Human Spine Model (Master's thesis, Thesis / Dissertation ETD). ↙
Blacksburg: Virginia Tech. Retrieved June 19, 2020, from https://vtechworks.lib.vt. ↙
edu/handle/10919/33605.

%-------------------------------------------------------------------------
%% Copyright - Model

%This model is based on the spine model developed by Timothy C. Franklin and
%Dr. Kevin P. Granata from Virginia Polytechnic Institute and State
%University (Franklin, 2006).

% The Human Motion Control Laboratory at the University of Kansas received some of ↙
the files developed by Franklin et al. for the model (Michael Madigan, personal ↙
communication, July 26, 2008).

% Franklin's published thesis can be downloaded from the Virginia Tech Electronic
% Theses and Dissertations library:
% https://vtechworks.lib.vt.edu/handle/10919/33605
%    Author: Timothy C. Franklin
%    Date Issued: 2006-06-08
%    Title: Linear System Analyses of the Role of Reflex Gain and Delay in a Dynamic ↙
Human Spine Model
%    Department: Engineering Mechanics
%    Type: Thesis
%    Degree Grantor: Virginia Polytechnic Institute and State University
%    Committee Chair: Kevin P. Granata
%    Committee Members: Michael L. Madigan & Scott Hendricks

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

clear all; close all; clc;
%% Verification Procedure
% Create Excel File with verification outputs.
 global date
 date=datetime('now');
```

```
 verification_logical=0; %1 indicates that verification should be performed
 save('verification_initialize_vars.mat','date','verification_logical')

 if verification_logical==1
    For_verification %has clc and clear commands in file
    clear; clc
end
%% Load the Defined Model Parameters
% If you would like to adjust the model parameters, edit the
% "model_parameters.m" file
load('verification_initialize_vars.mat')
model_parameters;

%% Import Muscle Files

[M]=load_musclefile(); %this loads the CholeMusc.xls file
 %% Import External Force File
 % If you would like to adjust the external force applied, adjust the
 % "load_forcefile.m" file or the LoadLift.xls Excel spreadsheet being imported

[E]=load_forcefile(); %this loads the LoadLift.xls file
EFo=0; %force magnitude (units: N)

rotateforce % determine the symbolic force vector

%% Define State Variables
 V.SperM=1; %Number of state variables per muscle
 V.States=38+V.SperM*length(M); %Number of state variables
 Stateo(37:V.States,1)=0;

 %% Transform the Applied External Force

 % yield individual angle variables based on state variables vectors; this
 % will be used for function inputs
[A0,A1,A2,A3,A4,A5,Ap,B0,B1,B2,B3,B4,B5,Bp,DA0,DA1,DA2,DA3,DA4,DA5,DAp,DB0,DB1,DB2, ↙
DB3,DB4,DB5,DBp,DT0,DT1,DT2,DT3,DT4,DT5,DTp,T0,T1,T2,T3,T4,T5,Tp, ...
    A0dot,A1dot,A2dot,A3dot,A4dot,A5dot,T0dot,T1dot,T2dot,T3dot,T4dot,T5dot,B0dot, ↙
B1dot,B2dot,B3dot,B4dot,B5dot,Apdot,Bpdot,Tpdot]=vect2ang(Stateo,Pango);

 % convert symbolic outputs to function handles
[Gl_funct,Gr_funct,Mm_funct,Cm_funct,Gm_funct,Ma_funct,ML_funct,MV_funct,Ea_funct, ↙
Rm_funct,dRm_funct,origin_funct,dorigin_funct]=tofuncts();

% determine the rotated force
Ea=Ea_funct(A0,Ap,B0,B1,B2,B3,B4,B5,Bp,L1,L2,L3,L4,L5,T0,T1,T2,T3,T4,T5,Tp,cent1, ↙
cent2,cent3);

% ensure that the external force attachment points are aligned as necessary
% (if the lordosis case is adjusted, then the external force coordinates


% also need to be to be adjusted)
if abs(Ea(4)-Ea(8))>0.0001
    error('Check external force rotated coordinates.')
end
```

```
% writing to the Verification Excel Spreadsheet
 if verification_logical==1
 filename=sprintf('Verification Summary %s.xlsx',date);
 filename=replace(filename,":","."); %cannot have colons in file titles
 filename=string(append('C:\Users\Valerie↙
Jardon\Documents\Model\FINALIZED_CODE_updated\Simulation Runs Verification Output\',↙
filename));

 writecell({'External Force Magnitude'},filename,'Sheet',1,'Range','B21')
 writematrix(EFo,filename,'Sheet',1,'Range','B22')

 writecell({'Rotated External Force'},filename,'Sheet',1,'Range','B24')
 writematrix(Ea,filename,'Sheet',1,'Range','B25')
 end
%% Determine rotated muscle anatomy and resting muscle length
Ma=Ma_funct(A0,A1,A2,A3,A4,A5,Ap,B0,B1,B2,B3,B4,B5,Bp,L1,L2,L3,L4,L5,T0,T1,T2,T3,T4,↙
T5,Tp,cent1,cent2,cent3);
ML=ML_funct(A0,A1,A2,A3,A4,A5,Ap,B0,B1,B2,B3,B4,B5,Bp,L1,L2,L3,L4,L5,T0,T1,T2,T3,T4,↙
T5,Tp,cent1,cent2,cent3);
Lo=ML; %initial muscle length

%% Perform Optimization Procedure
optimize

%% Time Delay Analysis
% For the time delay analysis, the Jacobians must be separated by
% instantanteous and delayed components

disp('-----Time Delay Analysis Started-----')

% Building the Jacobian for the instantaneous components
Ji=Jpi;
for i=1:36
    Ji(19:36,i)=Ji(19:36,i)+(Un_opt'*Jai(:,19:36,i))';
end

% Building the Jacobian for the delayed components (reflexes)
Jr=Jpr;
for i=1:36
    Jr(19:36,i)=Jr(19:36,i)+(Un_opt'*Jar(:,19:36,i))';
end
%%
% Determine the delay margin in which the system would become unstable
DrawSet=1;


[Td]=tdinvest(Ji,Jr,DrawSet);

% Display determined time delay and muscle model parameters in command window
fprintf('The Time Delay is: %3.3f\n',Td)
fprintf('Muscle Model Parameters: q= %3.3f, b=%3.3f, Gp=%3.3f,Gd=%3.3f\n',q,b,Gp,Gd)
```

```
%% Simulation
% In this section, the nonlinear simulation is completed to compare the
% linear analysis results from the optimization procedure.

disp('-----Simulation Started-----')

StateDist=Stateo;
StateDist(16)=StateDist(16)+deg2rad(0.01); % disturbance applied to thoracic/cervical spine rigid body

Tfinal=5;
Tode=[0,Tfinal]; %defining the bounds of integration

%the delay margin (Td) defines the time delay in which the system becomes
%unstable, therefore the simulation should include the time delay prior to
%instability
Lags=Td*0.99;

%Delay must be greater than 0.06 seconds to be physiologically
%representative of reflexes and their contribution to stability
if Lags < 0.06
    error('Simulation cannot be performed. Required lag is not physiologically possible.')
end

%-----Set Simulation Options
 options=ddeset('OutputFcn',@plotscheme,'Events',@edcheck,'RelTol',1e-4,'AbsTol',1e-9,'Stats','on');

%-----Simulation
tic

sol=dde23(@spinedyn,Lags,StateDist,Tode,options,Un_opt,Pn,Pango,Vars,Gl_funct, Gr_funct,Mm_funct,Cm_funct,Gm_funct,Ma_funct,ML_funct,MV_funct,Ea_funct,Rm_funct, dRm_funct,origin_funct,dorigin_funct,Lord_Angs,IVD,Lo,CSA,EFo,ChSpace);
Time=sol.x;
State=sol.y';
Stated=State-ones(length(Time),1)*Stateo';

%Display Magnitude of Lag Simulated & Simulation Run Time
disp(['Lag Simulated: ' num2str(Lags(1))]);

elapsedtime=toc;


elapsedtime_min=fix(elapsedtime/60);
fprintf('The simulation took %3d minutes and %3d seconds to run.',elapsedtime_min,(elapsedtime-(60*elapsedtime_min)))
```

```
%% Calculate the distance from equilibrium using the normalized state-space
% Determine if the system is approaching equilibrium

position=Stated(:,1:18);
mean_position=mean(position,'all'); %average of all positions
position_Dss=position./mean_position;

velocity=Stated(:,19:36);
mean_velocity=mean(velocity,'all');
velocity_Dss=velocity./mean_velocity;

Dss=zeros(size(Stated,1),18);
    for i=1:18
        Dss(:,i)=(position_Dss(:,i)).^2+(velocity_Dss(:,i)).^2;
    end

        Dss=sqrt(sum(Dss,2)); %sum of each row - each row is a time step
        figure(4); plot(Time,Dss); grid on; xlabel('Time (seconds)'); ylabel('Dss');
title('Dss vs. Time')

% ----------------------- End of spine_main.m script
```

C.2 *For_verification.m:* Verification Tasks in MATLAB

```
clear; clc
%% 1. Initalize Verification Variables for Jardon Calculations
load('Stateo_Pango_Rand.mat')
load('Model_Parameters_Verify.mat')
cent1=0; cent2=0; %these values need to have a magnitude of zero for comparison with Franklin values
Vars=[L0,L1,L2,L3,L4,L5,la,lb,cent3,g,m0,m1,m2,m3,m4,m5,cent1,cent2];
Lord_Angs=[Stateo(16) Stateo(13) Stateo(10) Stateo(7) Stateo(4) Stateo(1) Pango(1)];
%Lordosis Angle Array (L0,L1....Pelvis)
IVD=zeros(1,4); %initializing the intervertebral disc parameters array
IKp=50; %Planar
IDp=0.5;
IKt=50; %Twist
IDt=0.5;
IVD=[IKp,IDp,IKt,IDt]; %IVD parameters
A=Stateo(37:end);

% Verification Vectors/Matrices
Lo_verification=[1:1:90]'; %initial muscle length array for verification
MF_verification=[1:1:90]'; %muscle force array for verification
EF_verification=1; %external force magnitude for verification
PRQ_verification=[1 2 3; 4 5 6; 7 8 9; 5 4 3; 2 1 9; 9 6 2; 3 2 1]; %IVD moments matrix for verification
U_verification=0.5*zeros(90,1);
Pn_verification=zeros(90,1);

[E]=load_forcefile();
rotateforce
% Jardon function handles
[Gl_funct,Gr_funct,Mm_funct,Cm_funct,Gm_funct,Ma_funct,ML_funct,MV_funct,Ea_funct, Rm_funct,dRm_funct,origin_funct,dorigin_funct]=tofuncts();

[A0,A1,A2,A3,A4,A5,Ap,B0,B1,B2,B3,B4,B5,Bp,DA0,DA1,DA2,DA3,DA4,DA5,DAp,DB0,DB1,DB2, DB3,DB4,DB5,DBp,DT0,DT1,DT2,DT3,DT4,DT5,DTp,T0,T1,T2,T3,T4,T5,Tp, ...
    A0dot,A1dot,A2dot,A3dot,A4dot,A5dot,T0dot,T1dot,T2dot,T3dot,T4dot,T5dot,B0dot, B1dot,B2dot,B3dot,B4dot,B5dot,Apdot,Bpdot,Tpdot]=vect2ang(Stateo,Pango);

T5ddot=0; B5ddot=0; A5ddot=0;
T4ddot=0; B4ddot=0; A4ddot=0;
T3ddot=0; B3ddot=0; A3ddot=0;
T2ddot=0; B2ddot=0; A2ddot=0;
T1ddot=0; B1ddot=0; A1ddot=0;
T0ddot=0; B0ddot=0; A0ddot=0;
Tpddot=0; Bpddot=0; Apddot=0;

L0=Vars(1); L1=Vars(2); L2=Vars(3); L3=Vars(4); L4=Vars(5); L5=Vars(6);
la=Vars(7); lb=Vars(8); g=Vars(10);
cent1=Vars(17); cent2=Vars(18); cent3=Vars(9);
m0=Vars(11); m1=Vars(12); m2=Vars(13); m3=Vars(14); m4=Vars(15); m5=Vars(16);
```

```
% ----- End of "Initalize Verification Variables for Jardon Calcs" section

%% 2. Perform Jardon Calculations for Verification Tasks
Gl=Gl_funct(B1,B2,B3,B4,B5,L1,L2,L3,L4,L5,T1,T2,T3,T4,T5);
Gr=Gr_funct(B0,B1,B2,B3,B4,B5,T0,T1,T2,T3,T4,T5);
Mm=Mm_funct(A0,A1,A2,A3,A4,A5,B0,B1,B2,B3,B4,B5,L0,L1,L2,L3,L4,L5,T0,T1,T2,T3,T4,T5, ↙
la,lb,m0,m1,m2,m3,m4,m5);
Cm=Cm_funct(A0,A1,A2,A3,A4,A5,A0dot,A1dot,A2dot,A3dot,A4dot,A5dot,A0ddot,A1ddot, ↙
A2ddot,A3ddot,A4ddot,A5ddot,Ap,Apdot,Apddot,B0,B1,B2,B3,B4,B5,B0dot,B1dot,B2dot, ↙
B3dot,B4dot,B5dot,B0ddot,B1ddot,B2ddot,B3ddot,B4ddot,B5ddot,Bp,Bpdot,Bpddot,L0,L1,L2, ↙
L3,L4,L5,T0,T1,T2,T3,T4,T5,T0dot,T1dot,T2dot,T3dot,T4dot,T5dot,T0ddot,T1ddot,T2ddot, ↙
T3ddot,T4ddot,T5ddot,Tp,Tpdot,Tpddot,cent1,cent2,cent3,g,la,lb,m0,m1,m2,m3,m4,m5);
Gm=Gm_funct(B0,B1,B2,B3,B4,B5,L0,L1,L2,L3,L4,L5,T0,T1,T2,T3,T4,T5,g,m0,m1,m2,m3,m4, ↙
m5);
Ma=Ma_funct(A0,A1,A2,A3,A4,A5,Ap,B0,B1,B2,B3,B4,B5,Bp,L1,L2,L3,L4,L5,T0,T1,T2,T3,T4, ↙
T5,Tp,cent1,cent2,cent3);
ML=ML_funct(A0,A1,A2,A3,A4,A5,Ap,B0,B1,B2,B3,B4,B5,Bp,L1,L2,L3,L4,L5,T0,T1,T2,T3,T4, ↙
T5,Tp,cent1,cent2,cent3);
MV=MV_funct(A0,A1,A2,A3,A4,A5,Ap,B0,B1,B2,B3,B4,B5,Bp,DA0,DA1,DA2,DA3,DA4,DA5,DAp, ↙
DB0,DB1,DB2,DB3,DB4,DB5,DBp,DT0,DT1,DT2,DT3,DT4,DT5,DTp,L1,L2,L3,L4,L5,T0,T1,T2,T3, ↙
T4,T5,Tp,cent1,cent2,cent3);
Ea=Ea_funct(A0,Ap,B0,B1,B2,B3,B4,B5,Bp,L1,L2,L3,L4,L5,T0,T1,T2,T3,T4,T5,Tp,cent1, ↙
cent2,cent3);
rotation_matrices=Rm_funct(A0,A1,A2,A3,A4,A5,Ap,B0,B1,B2,B3,B4,B5,Bp,T0,T1,T2,T3,T4, ↙
T5,Tp);
d_rotation_matrices=dRm_funct(A0,A1,A2,A3,A4,A5,Ap,B0,B1,B2,B3,B4,B5,Bp,DA0,DA1,DA2, ↙
DA3,DA4,DA5,DAp,DB0,DB1,DB2,DB3,DB4,DB5,DBp,DT0,DT1,DT2,DT3,DT4,DT5,DTp,T0,T1,T2,T3, ↙
T4,T5,Tp);
origin=origin_funct(Ap,B1,B2,B3,B4,B5,Bp,L1,L2,L3,L4,L5,T1,T2,T3,T4,T5,Tp,cent1, ↙
cent2,cent3);
dorigin=dorigin_funct(Ap,B1,B2,B3,B4,B5,Bp,DAp,DB1,DB2,DB3,DB4,DB5,DBp,DT1,DT2,DT3, ↙
DT4,DT5,DTp,L1,L2,L3,L4,L5,T1,T2,T3,T4,T5,Tp,cent1,cent2,cent3);
PRQ=IVD_calcs(rotation_matrices,d_rotation_matrices,Stateo,Pango,Lord_Angs,IVD);
[M]=load_musclefile(); %this loads the CholeMusc.xls file - Jardon function
CSA=M(:,end); % last column of M
[Q]=generalizedForce(Ma,real(MF_verification),Ea,EF_verification,PRQ_verification,Gl, ↙
Gr,origin);
[fm,dA]=musclemodel(U_verification,Pn_verification,A,ML,MV,Lo_verification,CSA);
% ----- End of "Perform Jardon Calculations for Verification Tasks" Section

%% 3. Perform Franklin Calculations for Verification Tasks
% Adjust variables necessary for Franklin's function inputs
Vars=[L0,L1,L2,L3,L4,L5,la,lb,cent3,g,m0,m1,m2,m3,m4,m5];
P=zeros(9,1); P(1:8)=Pango(1:8);
PRQ_verification=PRQ_verification(1:6,:); %needs to be trimmed for Franklin's MEX ↙
file

Q_F=genforcesc(Ma,real(MF_verification),Ea,EF_verification,PRQ_verification,Gl,Gr, ↙
origin); %should use the Jardon variables for consistent input
```

```
[Mm_F,Cm_F,Gm_F]=dynmatsc(Stateo,P,Vars);

% Determining the M matrix
M_F=struct;
M_F=muscinput(M_F,'.\ChMusc\CholeMusc.xls'); %Cholewicki muscle file from Franklin

% Additional verification task necessary due to errors in Franklin's
% CholeMusc.xls spreadsheet (discussed in Chapter 3 of Jardon thesis)
M_F2=struct;
M_F2=muscinput(M_F2,'.\ChMusc\CholeMusc_update.xls'); %Cholewicki muscle file with
user input errors corrected

[E]=load_forcefile(); %this loads the LoadLift.xls file - Jardon function
E_F=struct;
E_F=forcinput(E_F,'LoadLift.xls'); % input external forces
E_F=forcmat(E_F);

[Ma_F,Ea_F,ML_F,MV_F,Gl_F,Gr_F,Rm_F,Or_F,dRm_F,dOr_F,RBV_F,dRBV_F] = analysl(Stateo,
P,Vars,M,E);

[MF_F,dA]=muscmodel(U_verification,Pn_verification,A,ML_F,MV_F,Lo_verification,CSA);
PRQ_F=intervertc(RBV_F,dRBV_F,Lord_Angs,IVD);
% ----- End of "Perform Franklin Calculations for Verification Tasks" Section

%% 4. Verification Optimization Variables
% Initialize Jardon variables
load('Stateo_Pango_Rand.mat')
load('Model_Parameters_Verify.mat')
cent1=0; cent2=0;
Vars=[L0,L1,L2,L3,L4,L5,la,lb,cent3,g,m0,m1,m2,m3,m4,m5,cent1,cent2];
Lord_Angs=[Stateo(16) Stateo(13) Stateo(10) Stateo(7) Stateo(4) Stateo(1) Pango(1)];
%Lordosis Angle Array (L0,L1....Pelvis)
IVD=zeros(1,4); %initializing the intervertebral disc parameters array
IKp=50; %Planar
IDp=0.5;
IKt=50; %Twist
IDt=0.5;
IVD=[IKp,IDp,IKt,IDt]; %IVD parameters
CSA=M(:,end); % last column of M
Lo_verification=[1:1:90]'; %initial muscle length array for verification
eps=0.00001*pi/180;
EF_verification=1;
Po=zeros(size(M,1),1);Pn=Po;

% Jardon function handles
[Gl_funct,Gr_funct,Mm_funct,Cm_funct,Gm_funct,Ma_funct,ML_funct,MV_funct,Ea_funct,
Rm_funct,dRm_funct,origin_funct,dorigin_funct]=tofuncts();

M=load_musclefile(); %this loads the CholeMusc.xls file - Jardon function
```

```
E=load_forcefile(); %this loads the LoadLift.xls file - Jardon function

% ----------------------------------Jardon
% Optimization
length_array=zeros(90,1);

%Initializing the constraints for fmincon
Aeq=zeros(18,90); %equilibrium
beq=zeros(18,1); %equilibrium
A=-eye(90);
B=zeros(90,1);

% Passive Properties
% For the passive properties (t=0 & activation=0 & no muscle force)
% Includes intervertebral disc moments, external force
U=zeros(90,1); %activation = 0 for passive
beq=-spineaccP(0,Stateo,Pango,U,EF_verification,Vars,M,E,Lord_Angs,IVD,Gl_funct, ↙
Gr_funct,Mm_funct,Cm_funct,Gm_funct,Ma_funct,ML_funct,MV_funct,Ea_funct,Rm_funct, ↙
dRm_funct,origin_funct,dorigin_funct,[],'verification');

% Active Properties
% For active properties (t=0 & no external force & no IVD moments)
for i=1:length(length_array)
    U=zeros(90,1); %initializing a vector
    U(i)=1;
    Aeq(:,i)=spineaccA(i,0,Stateo,Pango,U,Pn,EF_verification,Vars,M,E, ↙
Lo_verification,CSA,Gl_funct,Gr_funct,Mm_funct,Cm_funct,Gm_funct,Ma_funct,ML_funct, ↙
MV_funct,Ea_funct,Rm_funct,dRm_funct,origin_funct,dorigin_funct,[],'verification');
end

Jpi=zeros(36,36);
qddP_J=zeros(36,36);

Jpi(1:18,19:36)=eye(18,18);
Jai=zeros(size(M,1),36,36);
qddA=zeros(size(M,1),36,36);
Jpr=zeros(36,36); %no contribution to passive
Jar=zeros(size(M,1),36,36);

for h=1:36
    dStateo=Stateo;
    dStateo(h)=dStateo(h)+eps;
        for i=1:90
            Uo=zeros(size(M,1),1);
            Uo(i)=1;
            [qddA,qddP,qddR,ML_mat(i,:,h),MV_mat(i,:,h)]=spineacc(i,dStateo,Pango,Uo, ↙
Pn,EF_verification,Vars,Lo_verification,CSA,Lord_Angs,IVD,Gl_funct,Gr_funct,Mm_funct, ↙
Cm_funct,Gm_funct,Ma_funct,ML_funct,MV_funct,Ea_funct,Rm_funct,dRm_funct, ↙
origin_funct,dorigin_funct,[],'verification');
```

```
            Jai(i,19:36,h)=(qddA-Aeq(:,i))/eps;
            qddA_J(i,19:36,h)=qddA;
            Jpi(19:36,h)=(qddP+beq)/eps;
            qddP_J(19:36,h)=qddP;

            Jar(i,19:36,h)=(qddR)/eps;
        end
end

% ---------------------------------- Franklin
% Adjust variables necessary for Franklin's function inputs
Vars=[L0,L1,L2,L3,L4,L5,la,lb,cent3,g,m0,m1,m2,m3,m4,m5];
P=zeros(9,1); P(1:8)=Pango;
S=Stateo; %replaced "Statevar2" with "Stateo
Pang=P; %replaced "Statevar1" with "P"

% Optimization - Franklin
Aeq_F=[];
Beq_F=[];
len=size(M,1);
A_F=-eye(size(M,1));
B_F=zeros(len,1);
Uo=zeros(size(M,1),1);
Aeq_F=zeros(18,size(M,1));
Beq_F=zeros(18,1);
Beq_F=-AccelerationMP(0,S,Pang,Lord_Angs,M,E,Lo_verification,CSA,Uo,Pn, ↙
EF_verification,Vars,IVD,'verification');

for i=1:size(M,1)
    Uo=zeros(size(M,1),1);
    Uo(i)=1;
    Aeq_F(:,i)=AccelerationMA(0,S,Pang,Lord_Angs,M,E,Lo_verification,CSA,Uo,Pn, ↙
EF_verification,Vars,IVD);
end

Jp_F=zeros(36,36);
qddP_F=zeros(36,36);

Jp_F(1:18,19:36)=eye(18,18);    % jacobians
Ja_F=zeros(size(M,1),36,36);
qddA_F=zeros(size(M,1),36,36);

for h=1:36
    Sd=S;
    Sd(h)=Sd(h)+eps;
        for i=1:size(M,1)
            Uo=zeros(size(M,1),1);
            Uo(i)=1;
            Ja_F(i,19:36,h)=(AccelerationMA(0,Sd,P,Lord_Angs,M,E,Lo_verification,CSA, ↙
```

```
Uo,Pn,EF_verification,Vars,IVD)-Aeq_F(:,i))/eps;
            qddA_F(i,19:36,h)=AccelerationMA(0,Sd,P,Lord_Angs,M,E,Lo_verification,
CSA,Uo,Pn,EF_verification,Vars,IVD);

        end
    Uo=zeros(size(M,1),1);
    qddP_F(19:36,h)=AccelerationMP(0,Sd,P,Lord_Angs,M,E,Lo_verification,CSA,Uo,Pn,
EF_verification,Vars,IVD,'verification');
        Jp_F(19:36,h)=(AccelerationMP(0,Sd,P,Lord_Angs,M,E,Lo_verification,CSA,Uo,Pn,
EF_verification,Vars,IVD,'verification')+Beq_F)/eps;

end
U_verification=0.5*zeros(90,1);
Power_F=costfun(U_verification,S,Jp_F,Ja_F,Lo_verification,CSA);
Power=costfunct(U_verification,Stateo,Jp_F,Ja_F,Lo_verification,CSA);
% ----- End of "Verification Optimization Variables" Section
%% 5. Perform Verification Tasks
% Calculate the difference between our output and Franklin's output.

[M_check_count, ~,M_diff,M_max_error]=verify(M(:,1:32),M_F(:,1:32),1E-3) %
Verification Task 2.2 & 2.4
[M2_check_count, ~,M2_diff,M2_max_error]=verify(M(:,1:32),M_F2(:,1:32),1E-3) %
Verification Task 2.2 & 2.4

[E_check_count, E_check,E_diff,E_max_error]=verify(E,E_F,1E-15) %Verification Task
3.2

%for Verification Task 4.1
rotation_matricesT=zeros(size(Rm_F));
d_rotation_matricesT=zeros(size(Rm_F));

dim_mat=size(rotation_matrices);
j=1;
for i=1:(dim_mat(1)/3)
    rotation_matricesT(j:j+2,:)=transpose(rotation_matrices(j:j+2,:));
    d_rotation_matricesT(j:j+2,:)=transpose(d_rotation_matrices(j:j+2,:));
    j=j+3;
end

[Rm_check_count, ~, Rm_diff,Rm_max_error]=verify(rotation_matricesT,Rm_F,1E-15) %
Verification Task 4.1
[dRm_check_count, ~,dRm_diff,dRm_max_error]=verify(d_rotation_matricesT,dRm_F,1E-15)
%Verification Task 4.1
%-------------------------------------------------------------------------

[Or_check_count, ~,Or_diff,Or_max_error]=verify(origin,Or_F,1E-15) %Verification Task
5.1
[dOr_check_count, ~,dOr_diff,dOr_max_error]=verify(dorigin,dOr_F,1E-15) %Verification
Task 5.1
```

```
[Ma_check_count, ~,Ma_diff,Ma_max_error]=verify(Ma(:,1:33),Ma_F,1E-15)  %Verification Task 6.1

[ML_check_count, ~,ML_diff,ML_max_error]=verify(ML,ML_F,1E-15)  %Verification Task 7.1
[MV_check_count, ~,MV_diff,MV_max_error]=verify(MV,MV_F,1E-15)  %Verification Task 7.1

[Ea_check_count, ~,Ea_diff,Ea_max_error]=verify(Ea,Ea_F(1:9),1E-15)  %Verification Task 8.1

[Cm_check_count, ~,Cm_diff,Cm_max_error]=verify(Cm,Cm_F',1E-15)  %Verification Task 9.1

[Gl_check_count, ~, Gl_diff,Gl_max_error]=verify(Gl,Gl_F,1E-15)  %Verification Task 9.4
[Gr_check_count, ~,Gr_diff,Gr_max_error]=verify(Gr,Gr_F,1E-15)  %Verification Task 9.4

[Mm_check_count, ~,Mm_diff,Mm_max_error]=verify(Mm,Mm_F,1E-15)  %Verification Task 9.2
[Gm_check_count, ~,Gm_diff,Gm_max_error]=verify(Gm,Gm_F',1E-15)  %Verification Task 9.2

[PRQ_check_count, ~,PRQ_diff,PRQ_max_error]=verify(PRQ,PRQ_F,1E-15)  %Verification Task 10.1

[Q_check_count, ~,Q_diff,Q_max_error]=verify(Q,Q_F,1E-15)  %Verification Task 11.1

[Aeq_check_count, ~,~,Aeq_max_error]=verify(Aeq,Aeq_F,1E-15)  %Verification Task 12.1
[beq_check_count, ~,~,beq_max_error]=verify(beq,Beq_F,1E-15)  %Verification Task 12.1
[A_check_count, ~,~,A_max_error]=verify(A,A_F,1E-15)  %Verification Task 12.1
[B_check_count, ~,~,B_max_error]=verify(B,B_F,1E-15)  %Verification Task 12.1

[~, ~,~,qddA_max_error]=verify(qddA_J(1:90,1:36,1),qddA_F(1:90,1:36,1),1E-15)  %Verification Task 12.3
[~, ~,~,qddP_max_error]=verify(qddP_J,qddP_F,1E-15)  %Verification Task 12.3

[~, ~,~,Ja_max_error]=verify(Jai(1:90,1:36,1),Ja_F(1:90,1:36,1),1E-15)  %Verification Task 12.4
[~, ~,~,Jp_max_error]=verify(Jpi,Jp_F,1E-15)  %Verification Task 12.4


[MF_check_count, ~,MF_diff,MF_max_error]=verify(fm,MF_F,1E-15)  %Verification Task 13.1


[Power_check_count, ~,Power_diff,Power_max_error]=verify(Power,Power_F,1E-15)  %Verification Task 12.6

% ----- End of "Perform Verification Tasks" Section
```

```
%% Exporting Verification to Excel Spreadsheet
 global date
 warning('off','MATLAB:xlswrite:AddSheet');

 filename=sprintf('Verification Summary %s.xlsx',date);
 filename=replace(filename,":","."); %cannot have colons in file titles
 filename=string(append('C:\Users\Valerie↙
Jardon\Documents\Model\FINALIZED_CODE_updated\Simulation Runs Verification Output\',↙
filename));
 writematrix(date,filename,'Sheet',1,'Range','A1')

 writecell({'Rotation Matrices'},filename,'Sheet',2,'Range','C3')
 writematrix(Rm_diff,filename,'Sheet',2,'Range','C4')

 writecell({'Differential Rotation Matrices'},filename,'Sheet',2,'Range','G3')
 writematrix(dRm_diff,filename,'Sheet',2,'Range','G4')

 writecell({'Origins'},filename,'Sheet',3,'Range','C3')
 writematrix(Or_diff,filename,'Sheet',3,'Range','C4')

 writecell({'Differential Origins'},filename,'Sheet',3,'Range','G3')
 writematrix(dOr_diff,filename,'Sheet',3,'Range','G4')

 writecell({'Rotated Muscle Anatomy (Ma)'},filename,'Sheet',4,'Range','C3')
 writematrix(Ma_diff,filename,'Sheet',4,'Range','C4')

 writecell({'Non-Rotated Muscle Anatomy (M)'},filename,'Sheet',5,'Range','C3')
 writematrix(M2_diff,filename,'Sheet',5,'Range','C4')

 writecell({'Rotated External Force (Ea)'},filename,'Sheet',6,'Range','C3')
 writematrix(Ea_diff,filename,'Sheet',6,'Range','C4')

 writecell({'Non-Rotated External Force (E)'},filename,'Sheet',6,'Range','C5')
 writematrix(E_diff,filename,'Sheet',6,'Range','C6')

 writecell({'Gravity Vector (Gm)'},filename,'Sheet',7,'Range','C5')
 writematrix(Gm_diff,filename,'Sheet',7,'Range','C6')

 writecell({'Coriolis Vector (Cm)'},filename,'Sheet',7,'Range','C8')
 writematrix(Cm_diff,filename,'Sheet',7,'Range','C9')

 writecell({'Mass Matrix (Mm)'},filename,'Sheet',7,'Range','C11')
 writematrix(Mm_diff,filename,'Sheet',7,'Range','C12')

 writecell({' (Gl)'},filename,'Sheet',8,'Range','C5')
 writematrix(Gl_diff,filename,'Sheet',8,'Range','C6')

 writecell({' (Gr)'},filename,'Sheet',8,'Range','G5')
 writematrix(Gr_diff,filename,'Sheet',8,'Range','G6')
```

```
writecell({' Optimization (A)'},filename,'Sheet',9,'Range','C5')
writematrix(A_max_error,filename,'Sheet',9,'Range','C6')

writecell({' Optimization (B)'},filename,'Sheet',9,'Range','D5')
writematrix(B_max_error,filename,'Sheet',9,'Range','D6')

writecell({' Optimization (Aeq)'},filename,'Sheet',9,'Range','E5')
writematrix(Aeq_max_error,filename,'Sheet',9,'Range','E6')

writecell({' Optimization (beq)'},filename,'Sheet',9,'Range','F5')
writematrix(beq_max_error,filename,'Sheet',9,'Range','F6')

writecell({' Optimization (Jp)'},filename,'Sheet',9,'Range','G5')
writematrix(Jp_max_error,filename,'Sheet',9,'Range','G6')

writecell({' Optimization (Ja)'},filename,'Sheet',9,'Range','H5')
writematrix(Ja_max_error,filename,'Sheet',9,'Range','H6')

writecell({' Generalized Forces (Q)'},filename,'Sheet',10,'Range','C5')
writematrix(Q_diff,filename,'Sheet',10,'Range','C6')

writecell({' Muscle Length (ML)'},filename,'Sheet',10,'Range','D5')
writematrix(ML_diff,filename,'Sheet',10,'Range','D6')

writecell({' Muscle Velocity (MV)'},filename,'Sheet',10,'Range','E5')
writematrix(MV_diff,filename,'Sheet',10,'Range','E6')

writecell({' Muscle Force (MF)'},filename,'Sheet',10,'Range','F5')
writematrix(MF_diff,filename,'Sheet',10,'Range','F6')

writecell({' Intervertebral Disc Moments (PRQ)'},filename,'Sheet',11,'Range','C5')
writematrix(PRQ_diff,filename,'Sheet',11,'Range','C6')
% ----- End of "Exporting Verification to Excel Spreadsheet" Section
```

C.3 *model_parameters.m*: Model Parameters

```
%Valerie Jardon
%Department of Bioengineering, University of Kansas
%model_parameters.m
%Last Edited: 4/29/2022
%% Description
%     In the model_parameters.m file, the model parameters are defined.

%     This file (model_parameters.m) is based on the "Define Simulation Variables", ↙
"Intervertebral Disc Parameters",
%     "Initialize Position Vectors" and "Physiological Lordosis Angles" sections of ↙
the spinesim.m file
%     developed by Timothy C. Franklin, published in Appendix B of his thesis ↙
(Franklin, 2006, p. 99-100).

%     Additionally, this code includes the lordosis angles calculations and
%     incorporates them into the position vectors.

% ----- Reference:
% Franklin, T. C. (2006). Linear System Analyses of the Role of Reflex Gain and Delay ↙
in a Dynamic Human Spine Model (Master's thesis, Thesis / Dissertation ETD). ↙
Blacksburg: Virginia Tech. Retrieved June 19, 2020, from https://vtechworks.lib.vt. ↙
edu/handle/10919/33605.
%--------------------------------------------------------------------------
%--------------------------------------------------------------------------
%% Defining the Model Parameters
%This section is based on the "Define Simulation Variables" section of the spinesim.m ↙
file
%developed by Timothy C. Franklin, published in Appendix B of his thesis (Franklin, ↙
2006, p. 99).
%--------------------------------------------------------------------------
 L0=0.4241; %body 0 - thoracic and cervical spine (units: m)

 % Cholewicki-Defined Spacing in Muscle Geometry
 L1_ChSpace=0.03625; %units: m
 L2_ChSpace=0.03860; %units: m
 L3_ChSpace=0.0379; %units: m
 L4_ChSpace=0.0370; %units: m
 L5_ChSpace=0.0389; %units: m
 ChSpace=[L1_ChSpace; L2_ChSpace; L3_ChSpace; L4_ChSpace; L5_ChSpace];

 L1=L1_ChSpace;
 L2=L2_ChSpace;
 L3=L3_ChSpace;
 L4=L4_ChSpace;
 L5=L5_ChSpace;

 % Mass of bodies
 m0=16; %body 0 - thoracic and cervical spine (units: kg)
 m1=2; %body 1 - L1 (units: kg)


 m2=2; %body 2 - L2 (units: kg)
 m3=2; %body 3 - L3 (units: kg)
 m4=2; %body 4 - L4 (units: kg)
 m5=2; %body 5 - L5 (units: kg)
```

```
la=0.024; %Radius of trunk, ML
lb=0.017; %Radius of trunk, AP

cent1=0;
cent2=0;
cent3=0.1067; %Location of L5 pivot in P in the Axis 3 direction

g=9.8; %Gravity positive

%Creating an array for Model Variables
Vars=[L0,L1,L2,L3,L4,L5,la,lb,cent3,g,m0,m1,m2,m3,m4,m5,cent1,cent2];
%------------------End of Define the Model Parameters Section

%% Intervertebral Disc Parameters
%This section is based on the "Intervertebral Disc Parameters" section of the ↙
spinesim.m file
%developed by Timothy C. Franklin, published in Appendix B of his thesis (Franklin, ↙
2006, p. 99).
%-------------------------------------------------------------------------

IVD=zeros(1,4); %initializing the intervertebral disc parameters array
IKp=50; %Planar
IDp=0.5;
IKt=50; %Twist
IDt=0.5;
IVD=[IKp,IDp,IKt,IDt];
%------------------End of Intervertebral Disc Parameters Section

%% Initialize Position Vectors
%This section is based on the "Initialize Position Vectors" section of the spinesim.m ↙
file
%developed by Timothy C. Franklin, published in Appendix B of his thesis (Franklin, ↙
2006, p. 99).
%-------------------------------------------------------------------------
Stateo=zeros(36,1); %initializing state vector
Pango=zeros(9,1); %initializing pelvis vector

%------------------End of Initialize Position Vectors Section

%% Lordosis Angles
%This section is based on the "Physiological Lordosis Angles" section of the ↙
spinesim.m file
%developed by Timothy C. Franklin, published in Appendix B of his thesis (Franklin, ↙
2006, p. 100).


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

% Franklin's Values
%  Pango(1)=deg2rad(-20); %Pelvis Lordosis Angle (units: radians)
%  Stateo(1)=deg2rad(-14); %L5 Lordosis Angle (units: radians)
%  Stateo(4)=deg2rad(8); %L4 Lordosis Angle (units: radians)
%  Stateo(7)=deg2rad(18); %L3 Lordosis Angle (units: radians)
%  Stateo(10)=deg2rad(15); %L2 Lordosis Angle (units: radians)
%  Stateo(13)=deg2rad(10); %L1 Lordosis Angle (units: radians)
%  Stateo(16)=deg2rad(0); %L0 Lordosis Angle (units: radians)
```

```
% Normal Lordosis
%  Pango(1)=deg2rad(-38); %Pelvis Lordosis Angle (units: radians)
%  Stateo(1)=deg2rad(-17.8); %L5 Lordosis Angle (units: radians)
%  Stateo(4)=deg2rad(-2); %L4 Lordosis Angle (units: radians)
%  Stateo(7)=deg2rad(9.6); %L3 Lordosis Angle (units: radians)
%  Stateo(10)=deg2rad(17.5); %L2 Lordosis Angle (units: radians)
%  Stateo(13)=deg2rad(20.1); %L1 Lordosis Angle (units: radians)
%  Stateo(16)=deg2rad(0); %L0 Lordosis Angle (units: radians)

%  Hyperlordosis
 Pango(1)=deg2rad(-38); %Pelvis Lordosis Angle (units: radians)
 Stateo(1)=deg2rad(-8.565); %L5 Lordosis Angle (units: radians)
 Stateo(4)=deg2rad(14.142); %L4 Lordosis Angle (units: radians)
 Stateo(7)=deg2rad(30.962); %L3 Lordosis Angle (units: radians)
 Stateo(10)=deg2rad(42.736); %L2 Lordosis Angle (units: radians)
 Stateo(13)=deg2rad(46.1); %L1 Lordosis Angle (units: radians)
 Stateo(16)=deg2rad(0); %L0 Lordosis Angle (units: radians)

 %Hyperlordosis - range of Pesenti values
%  Pango(1)=deg2rad(-38); %Pelvis Lordosis Angle (units: radians)
%  Stateo(1)=deg2rad(-20.2); %L5 Lordosis Angle (units: radians)
%  Stateo(4)=deg2rad(-2.4); %L4 Lordosis Angle (units: radians)
%  Stateo(7)=deg2rad(15.4); %L3 Lordosis Angle (units: radians)
%  Stateo(10)=deg2rad(33.2); %L2 Lordosis Angle (units: radians)
%  Stateo(13)=deg2rad(51); %L1 Lordosis Angle (units: radians)
%  Stateo(16)=deg2rad(0); %L0 Lordosis Angle (units: radians)

 %Hypolordosis
%   Pango(1)=deg2rad(-38); %Pelvis Lordosis Angle (units: radians)
%  Stateo(1)=deg2rad(-26.765); %L5 Lordosis Angle (units: radians)
%  Stateo(4)=deg2rad(-18.098); %L4 Lordosis Angle (units: radians)
%  Stateo(7)=deg2rad(-11.678); %L3 Lordosis Angle (units: radians)
%  Stateo(10)=deg2rad(-7.184); %L2 Lordosis Angle (units: radians)
%  Stateo(13)=deg2rad(-5.9); %L1 Lordosis Angle (units: radians)
%  Stateo(16)=deg2rad(0); %L0 Lordosis Angle (units: radians)

 %Hypolordosis - range of Pesenti values
%   Pango(1)=deg2rad(-38); %Pelvis Lordosis Angle (units: radians)



%  Stateo(1)=deg2rad(-36); %L5 Lordosis Angle (units: radians)
%  Stateo(4)=deg2rad(-34); %L4 Lordosis Angle (units: radians)
%  Stateo(7)=deg2rad(-32); %L3 Lordosis Angle (units: radians)
%  Stateo(10)=deg2rad(-30); %L2 Lordosis Angle (units: radians)
%  Stateo(13)=deg2rad(-28); %L1 Lordosis Angle (units: radians)
%  Stateo(16)=deg2rad(0); %L0 Lordosis Angle (units: radians)

 Lord_Angs=[Stateo(16) Stateo(13) Stateo(10) Stateo(7) Stateo(4) Stateo(1) Pango(1)]; ↙
%Lordosis Angle Array (L0,L1....Pelvis)
 %------------------End of Lordosis Angles Section
```

```
%% Exporting Model Parameters to Excel Spreadsheet
%Exporting the Model Parameters to the Summary Excel Spreadsheet
global date

if verification_logical==1
   task=2;
else
   task=1;
end

for i=1:task
    if i==1
filename=sprintf('Model Parameters Summary %s.xlsx',date);
filename=replace(filename,":","."); %cannot have colons in file titles
filename=string(append('C:\Users\Valerie↙
Jardon\Documents\Model\FINALIZED_CODE_updated\Simulation Runs Model Parameters ↙
Summary\',filename));
writematrix(date,filename,'Sheet',1,'Range','A1')
    else
filename=sprintf('Verification Summary %s.xlsx',date);
filename=replace(filename,":","."); %cannot have colons in file titles
filename=string(append('C:\Users\Valerie↙
Jardon\Documents\Model\FINALIZED_CODE_updated\Simulation Runs Verification Output\' ,↙
filename));
writematrix(date,filename,'Sheet',1,'Range','A1')
    end
heading={'Body #','Length(m)','Mass(kg)','Lordosis Angle(rad)'};
writecell(heading,filename,'Sheet',1,'Range','C3')
body_str={'L0','L1','L2','L3','L4','L5','Pelvis'}';
body_num=[0 1 2 3 4 5 6]';
writecell(body_str,filename,'Sheet',1,'Range','B4')
writematrix(body_num,filename,'Sheet',1,'Range','C4')

writematrix(Vars(1:6)',filename,'Sheet',1,'Range','D4') %Length Array
writematrix(Vars(11:16)',filename,'Sheet',1,'Range','E4') %Mass Array
writematrix(Lord_Angs',filename,'Sheet',1,'Range','F4') %Mass Array

writematrix('cent values',filename,'Sheet',1,'Range','B12') %cent



writematrix([cent1,cent2,cent3],filename,'Sheet',1,'Range','C12') %cent

writematrix('Radius ML',filename,'Sheet',1,'Range','B13') %radius
writematrix(la,filename,'Sheet',1,'Range','C13') %radius
writematrix('Radius AP',filename,'Sheet',1,'Range','B14') %radius
writematrix(lb,filename,'Sheet',1,'Range','C14') %radius

heading={'Planar - K','Planar - D','Twist - K','Twist - D'};
writecell(heading',filename,'Sheet',1,'Range','B16') %IVD
writematrix(IVD',filename,'Sheet',1,'Range','C16') %radius
end

%-----------------End of Exporting Model Parameters to Excel Spreadsheet

save('model_parameters_mat.↙
mat','L0','L1','L2','L3','L4','L5','m0','m1','m2','m3','m4','m5','la','lb','cent1','c↙
ent2','cent3','g','IVD','Vars','L5_ChSpace','L4_ChSpace','L3_ChSpace','L2_ChSpace','L↙
1_ChSpace','ChSpace')
```

C.4 *rotationmatrices.mlx*: Determine the Rotation Matrices

Valerie Jardon

Department of Bioengineering, University of Kansas

rotationmatrices.mlx

Last Edited: 4/29/2022

Determining the Rotation Matrix for Each Body

Description: This file (rotationmatrices.mlx) can be used to determine the rotation matrix for each body. Each body rotates around Axis 1, then around Axis 2, and lastly around Axis 3. In this file, rotation in the counterclockwise (CCW) direction is considered positive. Lastly, the rotation matrices are compiled into one 21x3 matrix. Note: symbolic variables were used for the symbolic .mat files that were created.

Notes:

1) The use of the variable names (such as "Stateo", "Pango", etc.) is based on the work of Timothy C. Franklin (Franklin, 2006).

2) Franklin defines the order of the rotations and explains the use of these body rotations in Chapter 3A of his thesis (Franklin, 2006, p. 9).

Reference:

Franklin, T. C. (2006). *Linear System Analyses of the Role of Reflex Gain and Delay in a Dynamic Human Spine Model* (Master's thesis, Thesis / Dissertation ETD). Blacksburg: Virginia Tech. Retrieved June 19, 2020, from https://vtechworks.lib.vt.edu/handle/10919/33605.

Assigning Angle Values from the State Vectors

In this section, the values from the vectors Stateo and Pango are assigned as single values to variables. If the angle name begins with "T", then it is the rotation around Axis 1. If an angle begins with "B", then it is rotation around Axis 2. If an angle begins "A", then it is rotation around Axis 3. The second letter in the angle name defines the body. Body 0 is the thoracic/cervical spine region, Body 1 is L1, Body 2 is L2, Body 3 is L3, Body 4 is L4. Body 5 is L5 and Body p is the pelvis.

Example: the angle name "A5" means the angle of rotation around Axis 3 for Body 5.

```
clear; clc
syms Tp Bp Ap T5 B5 A5 T4 B4 A4 T3 B3 A3 T2 B2 A2 T1 B1 A1 T0 B0 A0 %used for creating symboli
```

```
%  Tp=Pango(1);
%  Bp=Pango(2);
%  Ap=Pango(3);
%  T5=Stateo(1);
%  B5=Stateo(2);
%  A5=Stateo(3);
```

```
%  T4=Stateo(4);
%  B4=Stateo(5);
%  A4=Stateo(6);
%  T3=Stateo(7);
%  B3=Stateo(8);
%  A3=Stateo(9);
%  T2=Stateo(10);
%  B2=Stateo(11);
%  A2=Stateo(12);
%  T1=Stateo(13);
%  B1=Stateo(14);
%  A1=Stateo(15);
%  T0=Stateo(16);
%  B0=Stateo(17);
%  A0=Stateo(18);
```

---------- End of "Assigning Angle Values from the State Vectors" section

Defining the Axis Rotations

In this section, the rotations around the axes are defined as follows:

```
syms q1 q2 q3
R1=[1 0 0; 0 cos(q1) -sin(q1); 0 sin(q1) cos(q1)]; %Rotation matrix for a rotation around axis
R2=[cos(q2) 0 sin(q2); 0 1 0; -sin(q2) 0 cos(q2)]; %Rotation matrix for a rotation around axis
R3=[cos(q3) -sin(q3) 0; sin(q3) cos(q3) 0; 0 0 1]; %Rotation matrix for a rotation around axis
R123=R1*R2*R3; %matrix multiplication to determine Body123 rotation matrix

%Pelvis
P_rotate = subs(R123, [q1 q2 q3], [Tp Bp Ap]); %substituting in rotation angles for the Pelvis

%L5
five_rotate = subs(R123, [q1 q2 q3], [T5 B5 A5]); %substituting in rotation angles for L5

%L4
four_rotate = subs(R123, [q1 q2 q3], [T4 B4 A4]);

%L3
three_rotate = subs(R123, [q1 q2 q3], [T3 B3 A3]);

%L2
two_rotate = subs(R123, [q1 q2 q3], [T2 B2 A2]);

%L1
one_rotate = subs(R123, [q1 q2 q3], [T1 B1 A1]);

%Thoracic/Cervical Region
zero_rotate = subs(R123, [q1 q2 q3], [T0 B0 A0]);
```

---------- End of "Defining the Axis Rotations" section

Compiling the rotation matrices for each body into a single 21x3 matrix

In this section, each resulting rotation matrix for each body is compiled into a 21x3 rotation matrix.

```
rotation_matrices=zeros(21,3); %initializing the 21x3 matrix
rotation_matrices=sym(rotation_matrices); %used for symbolic output

rotation_matrices(1:3,:)=P_rotate;
% P_rotate=double(P_rotate); %double precision function not used with symbolic
rotation_matrices(4:6,:)=five_rotate;
% five_rotate=double(five_rotate);
rotation_matrices(7:9,:)=four_rotate;
% four_rotate=double(four_rotate);
rotation_matrices(10:12,:)=three_rotate;
% three_rotate=double(three_rotate);
rotation_matrices(13:15,:)=two_rotate;
% two_rotate=double(two_rotate);
rotation_matrices(16:18,:)=one_rotate;
% one_rotate=double(one_rotate);
rotation_matrices(19:21,:)=zero_rotate;
% zero_rotate=double(zero_rotate);

% rotation_matrices=double(rotation_matrices);
```

---------- End of "Compiling the rotation matrices for each body into a single 21x3 matrix" section

Determining the Rotation Matrices - Differential

Assigning Angle Values from the State Vectors - Differential

In this section, the values from the vectors Stateo and Pango are assigned as single values to variables. See section "Assigning Angle Values from the State Vectors" for more information about naming procedure and meaning.

```
syms DTp DBp DAp DT5 DB5 DA5 DT4 DB4 DA4 DT3 DB3 DA3 DT2 DB2 DA2 DT1 DB1 DA1 DT0 DB0 DA0 %used
```

```
%  DTp=Pango(4);
%  DBp=Pango(5);
%  DAp=Pango(6);
%  DT5=Stateo(19);
%  DB5=Stateo(20);
%  DA5=Stateo(21);
```

```
%  DT4=Stateo(22);
%  DB4=Stateo(23);
%  DA4=Stateo(24);
%  DT3=Stateo(25);
%  DB3=Stateo(26);
%  DA3=Stateo(27);
%  DT2=Stateo(28);
%  DB2=Stateo(29);
%  DA2=Stateo(30);
%  DT1=Stateo(31);
%  DB1=Stateo(32);
%  DA1=Stateo(33);
%  DT0=Stateo(34);
%  DB0=Stateo(35);
%  DA0=Stateo(36);
```

---------- End of "Assigning Angle Values from the State Vectors - Differential" section

Defining the Axis Rotations - Differential

In this section, the total differential is calculated and the rotation matrices are determined. Recall the equation for the total differential for functions with three variables:

$$dw = \frac{\partial w}{\partial x}dx + \frac{\partial w}{\partial y}dy + \frac{\partial w}{\partial z}dz$$

```
syms DAngT DAngB DAngA
dR123=diff(R123,q1)*DAngT+diff(R123,q2)*DAngB+diff(R123,q3)*DAngA; %calculating the total diffe

%Pelvis
d_P_rotate = subs(dR123, [q1 q2 q3 DAngT DAngB DAngA], [Tp Bp Ap DTp DBp DAp]);

%L5
d_five_rotate = subs(dR123, [q1 q2 q3 DAngT DAngB DAngA], [T5 B5 A5 DT5 DB5 DA5]);

%L4
d_four_rotate = subs(dR123, [q1 q2 q3 DAngT DAngB DAngA], [T4 B4 A4 DT4 DB4 DA4]);

%L3
d_three_rotate = subs(dR123, [q1 q2 q3 DAngT DAngB DAngA], [T3 B3 A3 DT3 DB3 DA3]);

%L2
d_two_rotate = subs(dR123, [q1 q2 q3 DAngT DAngB DAngA], [T2 B2 A2 DT2 DB2 DA2]);

%L1
d_one_rotate = subs(dR123, [q1 q2 q3 DAngT DAngB DAngA], [T1 B1 A1 DT1 DB1 DA1]);

%Thoracic/Cervical Region
```

```
d_zero_rotate = subs(dR123, [q1 q2 q3 DAngT DAngB DAngA], [T0 B0 A0 DT0 DB0 DA0]);
```

---------- End of "Defining the Axis Rotations - Differential" section

Compiling the rotation matrices for each body into a single 21x3 matrix

In this section, each resulting rotation matrix for each body is compiled into a 21x3 rotation matrix.

```
d_rotation_matrices=zeros(21,3); %initializing the 21x3 matrix
d_rotation_matrices=sym(d_rotation_matrices); %used for symbolic output

d_rotation_matrices(1:3,:)=d_P_rotate;
% d_P_rotate=double(d_P_rotate); %double precision function not used with symbolic
d_rotation_matrices(4:6,:)=d_five_rotate;
% d_five_rotate=double(d_five_rotate);
d_rotation_matrices(7:9,:)=d_four_rotate;
% d_four_rotate=double(d_four_rotate);
d_rotation_matrices(10:12,:)=d_three_rotate;
% d_three_rotate=double(d_three_rotate);
d_rotation_matrices(13:15,:)=d_two_rotate;
% d_two_rotate=double(d_two_rotate);
d_rotation_matrices(16:18,:)=d_one_rotate;
% d_one_rotate=double(d_one_rotate);
d_rotation_matrices(19:21,:)=d_zero_rotate;
% d_zero_rotate=double(d_zero_rotate);

% d_rotation_matrices=double(d_rotation_matrices);
```

---------- End of "Compiling the rotation matrices for each body into a single 21x3 matrix" section

Create .mat file for output

```
save('rotation_matrices_sym_Pto0.mat','rotation_matrices','d_rotation_matrices')

%adjusting the order of rows
rearrange_mat=zeros(size(rotation_matrices)); drearrange_mat=zeros(size(rotation_matrices));
rearrange_mat=sym(rearrange_mat); drearrange_mat=sym(drearrange_mat);

j=1;
for i=1:numel(rotation_matrices)/9
    rearrange_mat(j:j+2,:)=rotation_matrices((end-j-1):(end-j+1),:);
    drearrange_mat(j:j+2,:)=d_rotation_matrices((end-j-1):(end-j+1),:);
    j=j+3;
end
rotation_matrices=rearrange_mat;
d_rotation_matrices=drearrange_mat;

save('rotation_matrices_sym_0toP.mat','rotation_matrices','d_rotation_matrices')

save('rotation_matrices_sym_separate','zero_rotate','one_rotate','two_rotate','three_rotate','
```

C.5 *origin_body.mlx*: Define origins of the bodies

Valerie Jardon

Department of Bioengineering, University of Kansas

origin_body.mlx

Last edited: 4/29/2022

This file (origin_body.mlx) is based on the Mathematica Code developed by Timothy C. Franklin, published in Appendix A of his thesis, specifically sections "Define Inertial Coordinate System", "Define Body Fixed (c) Coordinate Systems..." and "Define Origin of Each Body" (Franklin, 2006, p. 78-79).

Determining the Origin of Each Body

Description: This function is used to determine the origin of each body based on the rotation matrices and geometry of the vertebrae.

Notes:

1) The use of the variable names (such as "Orp", "Or5", etc.) is based on the work of Timothy C. Franklin (Franklin, 2006).

Reference:

Franklin, T. C. (2006). *Linear System Analyses of the Role of Reflex Gain and Delay in a Dynamic Human Spine Model* (Master's thesis, Thesis / Dissertation ETD). Blacksburg: Virginia Tech. Retrieved June 19, 2020, from https://vtechworks.lib.vt.edu/handle/10919/33605.

```
clear; clc
syms cent1 cent2 cent3 g L0 L1 L2 L3 L4 L5 la lb m0 m1 m2 m3 m4 m5

%load rotation matrices
load('rotation_matrices_sym_separate')
```

Defining Global Coordinate System

This section is based on the "Define Inertial Coordinate System" section of the Mathematica file developed by Timothy C. Franklin, published in Appendix A of his thesis (Franklin, 2006, p. 78).

```
n1=[1 0 0]'; %Axis 1
n2=[0 1 0]'; %Axis 2
n3=[0 0 1]'; %Axis 3
```

----------End of "Defining Global Coordinate System" Section

Define Body Coordinate Systems

This section is based on the "Define Body Fixed (c) Coordinate Systems..." section of the Mathematica file developed by Timothy C. Franklin, published in Appendix A of his thesis (Franklin, 2006, p. 78).

```
%Pelvis
cp_Axis1=P_rotate*n1; %Axis 1 - after rotation
cp_Axis2=P_rotate*n2; %Axis 2 - after rotation
cp_Axis3=P_rotate*n3; %Axis 3 - after rotation

cL5_Axis3=five_rotate*n3; %L5 Axis 3 - after rotation
cL4_Axis3=four_rotate*n3; %L4 Axis 3 - after rotation
cL3_Axis3=three_rotate*n3; %L3 Axis 3 - after rotation
cL2_Axis3=two_rotate*n3; %L2 Axis 3 - after rotation
cL1_Axis3=one_rotate*n3; %L1 Axis 3 - after rotation
cL0_Axis3=zero_rotate*n3; %L0 Axis 3 - after rotation
```

----------End of "Define Body Coordinate Systems" Section

Determine Origin of Each Body

This section is based on the "Define Origin of Each Body" section of the Mathematica file developed by Timothy C. Franklin, published in Appendix A of his thesis (Franklin, 2006, p. 79).

In this section of code, we start by determining the origin of the pelvis, which is is positioned at (0,0,0). cent3 is the location of the L5 pivot in body p (pelvis) in the Axis 3 direction. Using the known origin of the pelvis and cent3, we can determine the origin of body 5 (Or5).

Using the known origin of the previous body in the spinal column, the previous body's coordinate system and the height of the previous vertebra, we can determine the origin of the current body, as shown below.

```
Orp=0*n1; %pelvis origin
Or5=Orp+cent1*cp_Axis1+cent2*cp_Axis2+cent3*cp_Axis3; %L5 origin
Or4=Or5+L5*cL5_Axis3; %L4 origin
Or3=Or4+L4*cL4_Axis3; %L3 origin
Or2=Or3+L3*cL3_Axis3; %L2 origin
Or1=Or2+L2*cL2_Axis3; %L1 origin
Or0=Or1+L1*cL1_Axis3; %L0 origin
```

----------End of "Determine Origin of Each Body" Section

Define Body Coordinate Systems - Differential

Based on the methodology used previously in the "Define Body Coordinate Systems" section.

```
%Pelvis
cd_p_Axis1=d_P_rotate*n1; %Axis 1 - after rotation
cd_p_Axis2=d_P_rotate*n2; %Axis 2 - after rotation
cd_p_Axis3=d_P_rotate*n3; %Axis 3 - after rotation

cd_L5_Axis3=d_five_rotate*n3; %L5 Axis 3 - after rotation
cd_L4_Axis3=d_four_rotate*n3; %L4 Axis 3 - after rotation
cd_L3_Axis3=d_three_rotate*n3; %L3 Axis 3 - after rotation
cd_L2_Axis3=d_two_rotate*n3; %L2 Axis 3 - after rotation
cd_L1_Axis3=d_one_rotate*n3; %L1 Axis 3 - after rotation
cd_L0_Axis3=d_zero_rotate*n3; %L0 Axis 3 - after rotation
```

----------End of "Define Body Coordinate Systems - Differential" Section

Determine Origin of Each Body - Differential

Based on the methodology used previously in the "Determine Origin of Each Body" section.

```
d_Orp=0*n1; %pelvis origin
d_Or5=d_Orp+cent1*cd_p_Axis1+cent2*cd_p_Axis2+cent3*cd_p_Axis3; %L5 origin
d_Or4=d_Or5+L5*cd_L5_Axis3; %L4 origin
d_Or3=d_Or4+L4*cd_L4_Axis3; %L3 origin
d_Or2=d_Or3+L3*cd_L3_Axis3; %L2 origin
d_Or1=d_Or2+L2*cd_L2_Axis3; %L1 origin
d_Or0=d_Or1+L1*cd_L1_Axis3; %L0 origin
```

----------End of "Determine Origin of Each Body - Differential" Section

Save output in .mat file

```
origin=zeros(7,3); origin=sym(origin);
origin=[Or0';Or1';Or2';Or3';Or4';Or5';Orp'];
d_origin=[d_Or0';d_Or1';d_Or2';d_Or3';d_Or4';d_Or5';d_Orp'];
save('origin_sym_0toP.mat','origin','d_origin')

origin=flipud(origin); d_origin=flipud(d_origin); %rearrange row order
save('origin_sym_Pto0.mat','origin','d_origin')

Or0=Or0'; Or1=Or1'; Or2=Or2'; Or3=Or3'; Or4=Or4'; Or5=Or5'; Orp=Orp';
d_Or0=d_Or0'; d_Or1=d_Or1'; d_Or2=d_Or2'; d_Or3=d_Or3'; d_Or4=d_Or4'; d_Or5=d_Or5'; d_Orp=d_Or
save('origin_sym_separate.mat','Or0','Or1','Or2','Or3','Or4','Or5','Orp','d_Or0','d_Or1','d_Or
```

C.6 *transform_Cholewicki.mlx*: Import and Transform Skeletal Geometry

Valerie Jardon

Department of Bioengineering, University of Kansas

transform_Cholewicki.mlx

Last Edited: 10/24/2021

Transforming Cholewicki's Anatomy

Description: This code is used to transform the skeletal geometry provided in Appendix A of Cholewicki and McGill (Cholewicki and McGill, 1996, p. 12-13).

From Franklin's coordinates provided in Table Ap.A.2 and Table Ap.A.3 (Franklin, 2006, p. 89-93), we could determine that Franklin rotated and translated the coordinates from Cholewicki and McGill to determine the body-fixed coordinates.

The following methodology is not provided by Franklin, but was determined from the coordinates provided by Franklin.

References:

Cholewicki, J., & McGill, S. M. (1996). Mechanical stability of the in vivo lumbar spine: implications for injury and chronic low back pain. *Clinical biomechanics (Bristol, Avon)*, *11*(1), 1–15. https://doi.org/10.1016/0268-0033(95)00035-6

Franklin, T. C. (2006). *Linear System Analyses of the Role of Reflex Gain and Delay in a Dynamic Human Spine Model* (Master's thesis, Thesis / Dissertation ETD). Blacksburg: Virginia Tech. Retrieved June 19, 2020, from https://vtechworks.lib.vt.edu/handle/10919/33605

```
clear all; clc
%Load the muscle file
[~,~,raw]=xlsread('musclefile_Cholewicki.xlsx',1); %load sheet 1
[~,~,raw2]=xlsread('musclefile_Cholewicki.xlsx',2); %load sheet 2

%Trimming data
%Converting from cell to string
 musclefile=raw(3:92,:); musclefile=string(musclefile);
 points=raw2(3:244,:);
 points=points(:,3:5); points=double(string(points));
 raw2=raw2(3:end,:);

syms q1 q2 q3
R1=[1 0 0; 0 cos(q1) -sin(q1); 0 sin(q1) cos(q1)]; %Rotation matrix for a rotation around axis 1; + CCW rotation
R2=[cos(q2) 0 sin(q2); 0 1 0; -sin(q2) 0 cos(q2)]; %Rotation matrix for a rotation around axis 2; + CCW rotation
R3=[cos(q3) -sin(q3) 0; sin(q3) cos(q3) 0; 0 0 1]; %Rotation matrix for a rotation around axis 3; + CCW rotation
R321=R3*R2*R1;

pt_transform=zeros(size(points));

 %convert coordinates to body-fixed coordinates
for i=1:size(pt_transform,1)/2   %for the right side muscle anatomy
    pt_transform=transform_anatomy(pt_transform,raw2,points,R321,i);
end

%for the left side muscle anatomy (based on the right)
for i=122:size(pt_transform,1)
    pt_transform(i,:)=[-1*pt_transform(i-121,1),pt_transform(i-121,2),pt_transform(i-121,3)];
end

%connection points - check
connect_vect=zeros(14,3);
j=1;
for i=[1,20,36,61,73,85,96]
        connect_vect(j,:)=pt_transform(i,:);
        connect_vect(j+1,:)=pt_transform(i+1,:);
        j=j+2;
end
```

```
%Transformed anatomy matrix
M_Choltransformed=raw2(1:242,1:5);
M_Choltransformed(:,1)=raw2(:,1);
M_Choltransformed(:,2)=raw2(:,2);
M_Choltransformed(:,3:5)=num2cell(pt_transform);
save('transformed_Cholewicki.mat','M_Choltransformed')
```

```
function [pt_transform]=transform_anatomy(pt_transform,raw2,points,R123,i)
 syms q1 q2 q3

 if strcmp(raw2(i,2),'p')
     translate_pt=[0 points(1,2) points(1,3)];
     pt_transform_pt=points(i,:)-translate_pt;
     R123=subs(R123,[q1,q2,q3],[deg2rad(-5.3893119999999),0,0]);
     pt_transform(i,:)=R123*pt_transform_pt';
 elseif double(string(raw2(i,2)))==5
     translate_pt=[0 points(20,2) points(20,3)];
     pt_transform_pt=points(i,:)-translate_pt;
     R123=subs(R123,[q1,q2,q3],[deg2rad(17.9691382700114),0,0]);
     pt_transform(i,:)=R123*pt_transform_pt';
 elseif double(string(raw2(i,2)))==4
      translate_pt=[0 points(36,2) points(36,3)];
      pt_transform(i,:)=points(i,:)-translate_pt;
      %does not need to be rotated
 elseif double(string(raw2(i,2)))==3
    translate_pt=[0 points(61,2) points(61,3)];
    pt_transform_pt=points(i,:)-translate_pt;
    R123=subs(R123,[q1,q2,q3],[deg2rad(-12.2004700000026),0,0]);
    pt_transform(i,:)=R123*pt_transform_pt';
 elseif double(string(raw2(i,2)))==2
    translate_pt=[0 points(73,2) points(73,3)];
    pt_transform_pt=points(i,:)-translate_pt;
    R123=subs(R123,[q1,q2,q3],[deg2rad(-16.557071999997),0,0]);
    pt_transform(i,:)=R123*pt_transform_pt';
 elseif double(string(raw2(i,2)))==1
    translate_pt=[0 points(85,2) points(85,3)];
    pt_transform_pt=points(i,:)-translate_pt;
    R123=subs(R123,[q1,q2,q3],[deg2rad(-24.4439562999983),0,0]);
    pt_transform(i,:)=R123*pt_transform_pt';
 elseif double(string(raw2(i,2)))==0
    translate_pt=[0 points(96,2) points(96,3)];
    pt_transform_pt=points(i,:)-translate_pt;
    R123=subs(R123,[q1,q2,q3],[deg2rad(0.528868999999543),0,0]);
    pt_transform(i,:)=R123*pt_transform_pt';
 end
end
```

C.7 *load_musclefile.mlx*: Import Muscle Anatomy

```
%Valerie Jardon
%Department of Bioengineering, University of Kansas
%load_musclefile.m
%Last Edited: 4/29/2022

% This function (load_musclefile.m) is based on the muscinput.m and muscmat.m
% codes developed by Timothy C. Franklin, Virginia Polytechnic Institute and
% State University. The the muscinput.m and muscmat.m codes were obtained
% through means of personal communication (Michael Madigan, personal communication, July 26, 2008).
%
------------------------------------------------------------------------------------------
%% Load the Muscle File
% Description: This function imports the muscle paths and geometry needed
% for the model.
%
% The muscle attachment points and skeletal geometry are
% based on Cholewicki et al. (Cholewicki and McGill, 1996, p. 12-13).
% The built-in lordosis angle in Cholewicki's data has been
% removed so that the connection points of the vertebrae align along the
% vertical axis. This adjusted muscle data is located in the 'transformed
% Cholewicki.mat' file.
% Sheet 1 of the Excel spreadsheet ("musclefile_Cholewicki.xlsx") details
% the muscle paths for the muscles. A row is allocated for each muscle path. The
% attachment points are listed as strings, meaning it is text.
% These strings are then replaced by the adjusted muscle data.

% References:
% Cholewicki, J., & McGill, S. M. (1996). Mechanical stability of the in vivo lumbar spine: implications for injury and chronic low back pain. Clinical biomechanics (Bristol, Avon), 11(1), 1-15. https://doi.org/10.1016/0268-0033(95)00035-6
% Franklin, T. C. (2006). Linear System Analyses of the Role of Reflex Gain and Delay in a Dynamic Human Spine Model (Master's thesis, Thesis / Dissertation ETD). Blacksburg: Virginia Tech. Retrieved June 19, 2020, from https://vtechworks.lib.vt.edu/handle/10919/33605.
%-------------------------------------------------------------------------

function [M]=load_musclefile()
[~,~,raw]=xlsread('musclefile_Cholewicki.xlsx',1); %load sheet 1; muscle path of each muscle
load('transformed_Cholewicki.mat'); %load muscle points; built in lordosis angle "rotated out" of Cholewicki data... vertebrae connections pts aligned along vertical axis

%Trimming raw and raw2 to remove the column headings
%Converting from cell to string
musclepath=raw(3:92,:); musclepath=string(musclepath);
musclepoints=string(M_Choltransformed);
```

```
%Adjusting the pelvis origin to be aligned with the other connections
%points; Cholewicki has PELX(1) defined as the hip joint which we can
%adjust
musclepoints(1,3)="0";
musclepoints(size(musclepoints,1)/2+1,3)="0";

%Currently in a row, the first point is the origin, the second point
%is the terminal, and the following points are the nodes.
%In this section, the points for each muscle are rearranged
%so that the first point is the origin, the nodes follow, and the last
%point is the terminal.
%Additionally, the number of attachment points for each muscle is
%determined and stored in the first column of matrix M.

musclepath_trim=musclepath(:,2:9); %trimming the data to not include muscle name, ↙
rest length or CSA
musclepathR=musclepath_trim; % initailizing the matrix for reordered muscle paths (in ↙
order from origin to terminal)

    for i=1:size(musclepath,1) %an iteration for each muscle
        no_points=sum(~ismissing(musclepath_trim(i,:))); %count the number of ↙
attachment points in a row (not counting the <missing> elements)

        for j=1:no_points
            if j==1
                musclepathR(i,j)=musclepath_trim(i,1);
            elseif j==no_points
                musclepathR(i,j)=musclepath_trim(i,2);
            else
                musclepathR(i,j)=musclepath_trim(i,j+1);
            end
        end

    M(i,1)=no_points; M=string(M); %storing the number of attachment points for the ↙
muscle in column 1 of matrix M
    end

% Creating the M Matrix
j=0; %initializing variable
for h=[2:4:32]
    j=j+1;
    M(:,h)=musclepathR(:,j); %attachment point
end
M(:,34)=musclepath(:,10); %CSA (cm^2)

%Redefining pelvis body number to 6 instead of 'p'
musclepoints=strrep(musclepoints,'p','6');
```

```
%In this section, the strings of the attachment points in matrix M are
%replaced with the coordinates of these attachment points.
for k=[2:4:(size(M,2)-3)] %these are the columns in matrix M with the attachment points listed
    for j=1:size(M,1) %for each of the 90 muscles
        str=M(j,k);
        for i=1:size(musclepoints,1) %for each of the attachment points
            if musclepoints(i,1)==str
            M(j,k:k+3)=str2double(musclepoints(i,2:5));
            end
        end
    end
end

M=double(M); %convert to data type double

%coverting coordinates from cm to meters
for k=[3:4:size(M,2)]
    for j=1:90
        M(j,k:k+2)=M(j,k:k+2)/100;
    end
end

%Replacing the "NaN"s in the matrix with zeros
clear index_NaN
index_NaN=isnan(M); M(index_NaN)=0;

%% Verification Task

% Check that the right side and left side coordinates are the same (the
% only difference should be what side of the body the point is on)
sum_diff=0;
musclepoints_trim=double(musclepoints(:,3:5));
musclepoints_trim((size(musclepoints,1)/2)+1:end,1)=-1*musclepoints_trim((size(musclepoints,1)/2)+1:end,1);

for k=1:(size(musclepoints,1)/2)
    vect_diff=musclepoints_trim(k,:)-musclepoints_trim((k+size(musclepoints,1)/2),:);
    sum_diff=sum_diff+sqrt(sum(vect_diff.^2));
end

% fprintf('Verification Task: The right and left side anatomy has a total error of %3.16e\n',sum_diff)
end
```

C.8 *load_forcefile.mlx*: Import External Force File

Valerie Jardon

Department of Bioengineering, University of Kansas

load_forcefile.mlx

Last Edited: 4/29/2022

This file (load_forcefile.mlx) is based on the forcinput.m and forcmat.m codes developed by Timothy C. Franklin, Virginia Polytechnic Institute and State University. The forcinput.m and forcmat.m codes were obtained through means of personal communication (Michael Madigan, personal communication, July 26, 2008).

Reference:

Franklin, T. C. (2006). *Linear System Analyses of the Role of Reflex Gain and Delay in a Dynamic Human Spine Model* (Master's thesis, Thesis / Dissertation ETD). Blacksburg: Virginia Tech. Retrieved June 19, 2020, from https://vtechworks.lib.vt.edu/handle/10919/33605.

Load the External Force File

Description: Using this function (load_forcefile), the external force data can be imported from the specified Excel spreadsheet ("LoadLift.xls"). Sheet 1 details the origin and terminal points for the force. Sheet 2 gives the coordinates and body of the origin and terminal points.

```
function [E]=load_forcefile()
[~,txt,~]=xlsread('LoadLift.xls',1); %load sheet 1
[~,~,raw2]=xlsread('LoadLift.xls',2); %load sheet 2

%Trimming txt and raw2 to remove the column headings
%Converting from cell to string
forcefile=txt(3,:); forcefile=string(forcefile);
points=raw2(3:4,:); points=string(points);

%Redefining Newtonian body number to 7 instead of 'n'
points=strrep(points,'n','7');

% Creating the E array
E(:,2)=forcefile(:,2); %origin
E(:,6)=forcefile(:,3); %terminal
```

In this section, the strings of the points in matrix E are replaced with the coordinates of these points.

```
for k=[2,6] %these are the columns in array E with the points listed as strings
    str=E(1,k);
    for i=1:2
        if points(i,1)==str
        E(1,k:k+3)=str2double(points(i,2:5));
        end
```

```
    end
end

E=double(E); %convert to data type double
E(1,3:5)=E(1,3:5)/100; %convert to meters
E(1,7:9)=E(1,7:9)/100; %convert to meters
```

In this section, the number of points for each force will be calculated and stored in the first column of array E. Each point has 4 elements due to the coordinates and body number associated with the point.

```
    E(1,1)=0; %initializing the first column of the E matrix; this column will be used to indi

    num_points=size(E,2)/4; %calculating the number of columns of E and dividing by 4 to deter
    num_points=floor(num_points);
    E(1,1)=num_points;
end
```

C.9 *rotate_anatomy_sym.mlx*: Transform Muscle File, Calculate Muscle & Velocity

Valerie Jardon

Department of Bioengineering, University of Kansas

rotate_anatomy_sym.mlx

Last Edited: 4/29/2022

This file (rotate_anatomy_sym.mlx) is based on the analys1.c and kinematicsc.c codes developed by Timothy C. Franklin, Virginia Polytechnic Institute and State University. The analys1.c and kinematicsc.c codes were obtained through means of personal communication (Michael Madigan, personal communication, July 26, 2008).

Reference:

Franklin, T. C. (2006). *Linear System Analyses of the Role of Reflex Gain and Delay in a Dynamic Human Spine Model* (Master's thesis, Thesis / Dissertation ETD). Blacksburg: Virginia Tech. Retrieved June 19, 2020, from https://vtechworks.lib.vt.edu/handle/10919/33605.

Rotating the Muscle Anatomy

Description: This code is used to calculate the muscle rotated anatomy (Ma), the muscle point velocities (Mv), the muscle lengths (ML) and muscle velocities (MV). This code uses the symbolic rotation matrices and origins to yield a symbolic output for the requested parameters.

This section of code was used for the verification process:

```
clear; clc
% load('Stateo_Pango_Rand.mat')
% load('Model_Parameters_Verify.mat')
% Vars=[L0,L1,L2,L3,L4,L5,la,lb,cent3,g,m0,m1,m2,m3,m4,m5,cent1,cent2];
```

Initializing the Ma Matrix

The methodology used to determine the muscle anatomy matrix (Ma) was based on Franklin's work. This can be viewed on page 16 of Franklin's analys1.c file in the Kinematics() function.

```
M=load_musclefile();
Ma=zeros(size(M));
Ma(:,1)=M(:,1); %number of points
Ma(:,2)=M(:,2); %origin body
Ma(:,6)=M(:,6); %second point body
Ma(:,10)=M(:,10); %third point body
Ma(:,14)=M(:,14); %fourth point body
Ma(:,18)=M(:,18); %fifth point body
Ma(:,22)=M(:,22); %sixth point body
Ma(:,26)=M(:,26);
```

```
Ma(:,30)=M(:,30);
Ma(:,34)=M(:,34); %CSA
Ma=sym(Ma);
```

----------End of "Initializing the Ma Matrix" section

Rotating and Translating the Muscle Anatomy

In this section, the muscle attachment points are rotated and translated based on the origin and the rotation matrix of the body it is associated with.

The methodology used to determine the muscle anatomy matrix (Ma) was based on Franklin's work. This can be viewed on page 16 of Franklin's analys1.c file in the Kinematics() function.

```
% Rotate the Muscle Anatomy
for i=1:90
    num_pts=M(i,1); %number of muscle attachment points
    Ma=point_transform(M,Ma,num_pts,i);
end
```

----------End of "Rotating the Muscle Anatomy" section

Calculate the Length of Each Muscle

In this section, the length of each muscle is calculated. This is achieved by summing the length of the vectors in the muscle path.

For example, for a muscle with three attachment points, we would find the length of the position vector between the origin and node 1, and the length of the position vector between node 1 and the terminal. Then we could sum the length of both of these vectors to yield the total muscle length.

General Procedure:

First, we must determine the position vector between two muscle attachment points. Then we calculate the magnitude of this vector which is representative of the vector's length. This procedure is used for all muscle attachment points in the order that they appear in the muscle path, starting with muscle origin, then the nodes and finally the terminal point. Then the vector lengths can be summed to determine the total length of the muscle.

The methodology used to determine the lengths of the muscles was based on Franklin's work. This can be viewed on pages 4-5 of Franklin's analys1.c file in the MuscProps() function.

```
ML=sym('ML');
for i=1:90
    num_pts=Ma(i,1); %number of muscle attachment points
    ML(i,1)=musc_length(Ma,i,num_pts);
end
```

----------End of "Calculate the Length of Each Muscle" section

Initializing the Mv Matrix

The methodology used to determine the muscle point velocity matrix (Mv) was based on Franklin's work. This can be viewed on page 16 of Franklin's analys1.c file in the Kinematics() function.

```
Mv=zeros(size(M));
Mv(:,1)=M(:,1); %number of points
Mv(:,2)=M(:,2); %origin body
Mv(:,6)=M(:,6); %second point body
Mv(:,10)=M(:,10); %third point body
Mv(:,14)=M(:,14); %fourth point body
Mv(:,18)=M(:,18); %fifth point body
Mv(:,22)=M(:,22); %sixth point body
Mv(:,26)=M(:,26);
Mv(:,30)=M(:,30);
Mv(:,34)=M(:,34); %CSA
Mv=sym(Mv);
```

----------End of "Initializing the Mv Matrix" section

Calculating the Linear Velocity of the Muscle Attachment Points (Mv)

In this section, we calculate the linear velocity of the muscle attachment points for each muscle.

The methodology used to determine the muscle point velocity matrix (Mv) was based on Franklin's work. This can be viewed on page 16 of Franklin's analys1.c file in the Kinematics() function.

```
for i=1:90
    num_pts=M(i,1); %number of muscle attachment points
    Mv=point_transform_v(M,Mv,num_pts,i);
end
```

----------End of "Calculating the Linear Velocity of the Muscle Attachment Points (Mv)" section

Calculate the Muscle Velocity

In this section, the velocity of each muscle is calculated. Like the muscle length calculations, this involves using muscle segments, which are based on the muscle attachment points, to perform our calculations.

General Procedure:

First, the velocity vector and position vector between two muscle attachment points must be determined. The position unit vector can be calculated. The dot product of the velocity vector and the position unit vector is used to determine the magnitude of the velocity vector in the direction of the position unit vector. This procedure is used for all muscle attachment points in the order that they appear in the muscle path, starting with muscle

origin, then the nodes and finally the terminal point. The scalar result of the dot product can be summed for each segment of the muscle path to determine the total velocity of the muscle.

The methodology used to determine the velocities of the muscles was based on Franklin's work. This can be viewed on pages 4-5 of Franklin's analys1.c file in the MuscProps() function.

```
MV=sym('MV');
for i=1:90
    num_pts=M(i,1); %number of muscle attachment points
    MV(i,1)=musc_vel(Mv,Ma,i,num_pts);
end
```

----------End of "Calculate the Muscle Velocity" section

Save output in .mat file

```
save('rotate_anatomy_sym_MaMLMV.mat','Ma','MV','ML')
```

```
function [Ma]=point_transform(M,Ma,num_pts,i)
load('rotation_matrices_sym_separate')
load('origin_sym_separate.mat')

for n=0:num_pts-1
    if M(i,2+4*n)==6 %pelvis
        Ma(i,3+4*n:5+4*n)=P_rotate*M(i,3+4*n:5+4*n)'+Orp';
    elseif M(i,2+4*n)==5 %body five
        Ma(i,3+4*n:5+4*n)=five_rotate*M(i,3+4*n:5+4*n)'+Or5';
    elseif M(i,2+4*n)==4 %body four
        Ma(i,3+4*n:5+4*n)=four_rotate*M(i,3+4*n:5+4*n)'+Or4';
    elseif M(i,2+4*n)==3 %body three
        Ma(i,3+4*n:5+4*n)=three_rotate*M(i,3+4*n:5+4*n)'+Or3';
    elseif M(i,2+4*n)==2 %body two
        Ma(i,3+4*n:5+4*n)=two_rotate*M(i,3+4*n:5+4*n)'+Or2';
    elseif M(i,2+4*n)==1 %body one
        Ma(i,3+4*n:5+4*n)=one_rotate*M(i,3+4*n:5+4*n)'+Or1';
    elseif M(i,2+4*n)==0 %body zero
        Ma(i,3+4*n:5+4*n)=zero_rotate*M(i,3+4*n:5+4*n)'+Or0';
    end
end
end

function [ML]=musc_length(Ma,i,num_pts)
ML=0; ML=sym(ML);
    for n=0:num_pts-2
        ML_coords=Ma(i,3+4*n:5+4*n)-Ma(i,3+4*(n+1):5+4*(n+1)); %for the distance formula
        ML=ML+sqrt(sum(ML_coords.^2)); %distance formula
    end
end

function [Mv]=point_transform_v(M,Mv,num_pts,i)
load('rotation_matrices_sym_separate')
```

```
load('origin_sym_separate.mat')

for n=0:num_pts-1
    if M(i,2+4*n)==6 %pelvis
        Mv(i,3+4*n:5+4*n)=d_P_rotate*M(i,3+4*n:5+4*n)'+d_Orp';
    elseif M(i,2+4*n)==5 %body five
        Mv(i,3+4*n:5+4*n)=d_five_rotate*M(i,3+4*n:5+4*n)'+d_Or5';
    elseif M(i,2+4*n)==4 %body four
        Mv(i,3+4*n:5+4*n)=d_four_rotate*M(i,3+4*n:5+4*n)'+d_Or4';
    elseif M(i,2+4*n)==3 %body three
        Mv(i,3+4*n:5+4*n)=d_three_rotate*M(i,3+4*n:5+4*n)'+d_Or3';
    elseif M(i,2+4*n)==2 %body two
        Mv(i,3+4*n:5+4*n)=d_two_rotate*M(i,3+4*n:5+4*n)'+d_Or2';
    elseif M(i,2+4*n)==1 %body one
        Mv(i,3+4*n:5+4*n)=d_one_rotate*M(i,3+4*n:5+4*n)'+d_Or1';
    elseif M(i,2+4*n)==0 %body zero
        Mv(i,3+4*n:5+4*n)=d_zero_rotate*M(i,3+4*n:5+4*n)'+d_Or0';
    end
end
end

function [MV]=musc_vel(Mv,Ma,i,num_pts)
    MV=0; MV=sym(MV);
    for n=0:num_pts-2
        Ma_coords=Ma(i,3+4*n:5+4*n)-Ma(i,3+4*(n+1):5+4*(n+1));
        MV_coords=Mv(i,3+4*n:5+4*n)-Mv(i,3+4*(n+1):5+4*(n+1));
        Ma_coords_u=(1/sqrt(sum(Ma_coords.^2)))*Ma_coords; %unit vector
        MV=MV+dot(MV_coords,Ma_coords_u);
    end
end
```

C.10 *rotateforce.mlx*: Transform External Force File

Valerie Jardon

Department of Bioengineering, University of Kansas

rotateforce.mlx

Last Edited: 4/29/2022

This file (rotateforce.mlx) is based on the analys1.c and kinematicsc.c codes developed by Timothy C. Franklin, Virginia Polytechnic Institute and State University. The analys1.c and kinematics.c codes were obtained through means of personal communication (Michael Madigan, personal communication, July 26, 2008).

Transform the External Force

Description: This code is used to transform the external force points with respect to the body the point is associated with. The symbolic output of variable *Ea* is stored in the 'Ea_sym.mat' file.

Reference:

Franklin, T. C. (2006). *Linear System Analyses of the Role of Reflex Gain and Delay in a Dynamic Human Spine Model* (Master's thesis, Thesis / Dissertation ETD). Blacksburg: Virginia Tech. Retrieved June 19, 2020, from https://vtechworks.lib.vt.edu/handle/10919/33605.

```
% clear; clc
[E]=load_forcefile(); %load the external force coordinates

Ea=zeros(1,size(E,2)); Ea=sym(Ea); %initialize the rotated external force vector
num_pts=E(1); %number of points
Ea(1)=E(1);
Ea(2)=E(2);
Ea(6)=E(6);
Ea=point_transform(E,Ea,num_pts);
save('Ea_sym.mat','Ea') %save the symbolic output of Ea
```

```
function [Ea]=point_transform(E,Ea,num_pts)
load('origin_sym_separate.mat')
load('rotation_matrices_sym_separate.mat')

for n=0:num_pts-1
    if E(2+4*n)==6 %pelvis
        Ea(3+(4*n):5+(4*n))=P_rotate*(E(3+(4*n):5+(4*n))')+Orp';
    elseif E(2+4*n)==5 %body five
        Ea(3+4*n:5+4*n)=five_rotate*E(3+4*n:5+4*n)'+Or5';
    elseif E(2+4*n)==4 %body four
        Ea(3+4*n:5+4*n)=four_rotate*E(3+4*n:5+4*n)'+Or4';
    elseif E(2+4*n)==3 %body three
        Ea(3+4*n:5+4*n)=three_rotate*E(3+4*n:5+4*n)'+Or3';
```

```
    elseif E(2+4*n)==2 %body two
        Ea(3+4*n:5+4*n)=two_rotate*E(3+4*n:5+4*n)'+Or2';
    elseif E(2+4*n)==1 %body one
        Ea(3+4*n:5+4*n)=one_rotate*E(3+4*n:5+4*n)'+Or1';
    elseif E(2+4*n)==0 %body zero
        Ea(3+4*n:5+4*n)=zero_rotate*E(3+4*n:5+4*n)'+Or0';
    else
        Ea(3+4*n:5+4*n)=E(3+4*n:5+4*n);
    end
end
end
```

C.11 *dynam.mlx*: Dynamics

Valerie Jardon

Department of Bioengineering, University of Kansas

dynam.mlx

Last Edited: 4/29/22

This file (dynam.mlx) is based on the Mathematica Code developed by Timothy C. Franklin, published in Appendix A of his thesis (Franklin, 2006, p. 78-81)

Dynamics

Description: This file (dynam.mlx) is utilized for the system dynamics. The center of mass, velocity of the center of mass and the mass moment of inertia are defined for each body. The Lagrangian derivative is used for the Equations of Motion.

Reference:

Franklin, T. C. (2006). *Linear System Analyses of the Role of Reflex Gain and Delay in a Dynamic Human Spine Model* (Master's thesis, Thesis / Dissertation ETD). Blacksburg: Virginia Tech. Retrieved June 19, 2020, from https://vtechworks.lib.vt.edu/handle/10919/33605

```
clear; clc
```

Initialize Variables

The symbolic angles represented in the following variables need to be as a function of time for our calculations in this code. The function 'sub_timevar' can be used to adjust the .mat file variables.

```
load('origin_sym_separate.mat')
load('rotation_matrices_sym_separate.mat')

%Need angles as a function of time
Or0=sub_timevar(Or0)';
Or1=sub_timevar(Or1)';
Or2=sub_timevar(Or2)';
Or3=sub_timevar(Or3)';
Or4=sub_timevar(Or4)';
Or5=sub_timevar(Or5)';
Orp=sub_timevar(Orp)';

%Need angles as a function of time
zero_rotate=sub_timevar(zero_rotate);
one_rotate=sub_timevar(one_rotate);
two_rotate=sub_timevar(two_rotate);
three_rotate=sub_timevar(three_rotate);
four_rotate=sub_timevar(four_rotate);
five_rotate=sub_timevar(five_rotate);
P_rotate=sub_timevar(P_rotate);

syms la lb cent1 cent2 cent3 g t L0 L1 L2 L3 L4 L5 m0 m1 m2 m3 m4 m5
```

----------End of the "Initialize Variables" Section

Determining the Symbolic General Rotation Matrix

```
%Rotations
syms q1 q2 q3
R1=[1 0 0; 0 cos(q1) -sin(q1); 0 sin(q1) cos(q1)]; %Rotation axis 1 +CCW Rotation
R2=[cos(q2) 0 sin(q2); 0 1 0; -sin(q2) 0 cos(q2)]; %Rotation axis 2 +CCW Rotation
R3=[cos(q3) -sin(q3) 0; sin(q3) cos(q3) 0; 0 0 1]; %Rotation axis 3 +CCW Rotation
R123=R1*R2*R3;
R12=R1*R2;
```

----------End of the "Determining the Symbolic General Rotation Matrix" Section

Defining the Coordinate Systems

```
%Define the Inertial Coordinate System
n1=[1 0 0]';
n2=[0 1 0]';
n3=[0 0 1]';

%Define Body Fixed Coordinate Systems
syms Ap(t) Bp(t) Tp(t) T5(t) B5(t) A5(t) T4(t) B4(t) A4(t) T3(t) B3(t) A3(t) T2(t) B2(t) A2(t) T1(t) B1(t) A1(t) T0(t) B0(t) A0(t)

cp_Axis1=P_rotate*n1; %Pelvis Axis 1 - after rotation
cp_Axis2=P_rotate*n2; %Pelvis Axis 2 - after rotation
cp_Axis3=P_rotate*n3; %Pelvis Axis 3 - after rotation
cL5_Axis3=five_rotate*n3; %L5 Axis 3 - after rotation
cL4_Axis3=four_rotate*n3; %L4 Axis 3 - after rotation
cL3_Axis3=three_rotate*n3; %L3 Axis 3 - after rotation
```

```
cL2_Axis3=two_rotate*n3; %L2 Axis 3 - after rotation
cL1_Axis3=one_rotate*n3; %L1 Axis 3 - after rotation
cL0_Axis3=zero_rotate*n3; %L0 Axis 3 - after rotation
```

---------- End of the "Defining the Coordinate Systems" Section

Defining the Angular Velocities and Vectors

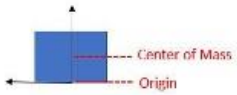
```
syms Apdot(t) Bpdot(t) Tpdot(t) T5dot(t) B5dot(t) A5dot(t) T4dot(t) B4dot(t) A4dot(t) T3dot(t) B3dot(t) A3dot(t) T2dot(t) B2dot(t) A2dot(t
syms Apddot Bpddot Tpddot T5ddot B5ddot A5ddot T4ddot B4ddot A4ddot T3ddot B3ddot A3ddot T2ddot B2ddot A2ddot T1ddot B1ddot A1ddot T0ddot

[NwCp_Abasis,NwCp_Nbasis]=ang_vel(R3,R123,Tp,Bp,Ap,Tpdot,Bpdot,Apdot); %Pelvis
[NwC5_Abasis,NwC5_Nbasis]=ang_vel(R3,R123,T5,B5,A5,T5dot,B5dot,A5dot); %L5
[NwC4_Abasis,NwC4_Nbasis]=ang_vel(R3,R123,T4,B4,A4,T4dot,B4dot,A4dot); %L4
[NwC3_Abasis,NwC3_Nbasis]=ang_vel(R3,R123,T3,B3,A3,T3dot,B3dot,A3dot); %L3
[NwC2_Abasis,NwC2_Nbasis]=ang_vel(R3,R123,T2,B2,A2,T2dot,B2dot,A2dot); %L2
[NwC1_Abasis,NwC1_Nbasis]=ang_vel(R3,R123,T1,B1,A1,T1dot,B1dot,A1dot); %L1
[NwC0_Abasis,NwC0_Nbasis]=ang_vel(R3,R123,T0,B0,A0,T0dot,B0dot,A0dot); %L0
```

---------- End of the "Defining the Angular Velocities and Vectors" Section

Determine the Center of Mass and the Velocity of the COM for Each Body

Recall that the origin of each rigid body is at the center of the inferior vertebral endplate. The center of mass is equidistant from the vertebral plates.




```
G5=Or5+L5/2*cL5_Axis3; %L5
G4=Or4+L4/2*cL4_Axis3; %L4
G3=Or3+L3/2*cL3_Axis3; %L3
G2=Or2+L2/2*cL2_Axis3; %L2
G1=Or1+L1/2*cL1_Axis3; %L1
G0=Or0+L0/2*cL0_Axis3; %L0

%Determining the velocity of the center of mass
vG5=diff(G5,t); %L5
vG4=diff(G4,t); %L4
vG3=diff(G3,t); %L3
vG2=diff(G2,t); %L2
vG1=diff(G1,t); %L1
vG0=diff(G0,t); %L0
```

---------- End of "Determine the Center of Mass and the Velocity of the COM for Each Body" Section

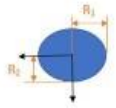
Define 3D Inertial Matrix of Each Body

The Moment of Inertia about the Centroidal Axes

$X-Axis: \frac{1}{4}mR_2^2+\frac{1}{12}ml^2$

$Y-Axis: \frac{1}{4}mR_1^2+\frac{1}{12}ml^2$

$Z-Axis: \frac{1}{4}m(R_1^2+R_2^2)$



Based on Franklin's variable definitions, *la* is the radius along the medial-lateral axis and *lb* is the radius along the anterior-posterior axis.

```
I5=[.25*m5*(lb^2)+(1/12)*m5*(L5^2),0,0;0,.25*m5*(la^2)+(1/12)*m5*(L5^2),0;0,0,.25*m5*((la^2)+(lb^2))];
I4=[.25*m4*(lb^2)+(1/12)*m4*(L4^2),0,0;0,.25*m4*(la^2)+(1/12)*m4*(L4^2),0;0,0,.25*m4*((la^2)+(lb^2))];
I3=[.25*m3*(lb^2)+(1/12)*m3*(L3^2),0,0;0,.25*m3*(la^2)+(1/12)*m3*(L3^2),0;0,0,.25*m3*((la^2)+(lb^2))];
I2=[.25*m2*(lb^2)+(1/12)*m2*(L2^2),0,0;0,.25*m2*(la^2)+(1/12)*m2*(L2^2),0;0,0,.25*m2*((la^2)+(lb^2))];
I1=[.25*m1*(lb^2)+(1/12)*m1*(L1^2),0,0;0,.25*m1*(la^2)+(1/12)*m1*(L1^2),0;0,0,.25*m1*((la^2)+(lb^2))];
I0=[.25*m0*(lb^2)+(1/12)*m0*(L0^2),0,0;0,.25*m0*(la^2)+(1/12)*m0*(L0^2),0;0,0,.25*m0*((la^2)+(lb^2))];
```

---------- End of "Define 3D Inertial Matrix of Each Body" Section

```
%Coverting symfun to sym for Kinetic Energy Calc
NwC5_rel=formula(NwC5_Abasis);
NwC4_rel=formula(NwC4_Abasis);
NwC3_rel=formula(NwC3_Abasis);
NwC2_rel=formula(NwC2_Abasis);
NwC1_rel=formula(NwC1_Abasis);
NwC0_rel=formula(NwC0_Abasis);
```

Kinetic and Potential Energy Definitions

The Lagrangian function (L) is as follows:

L=[Kinetic Energy of the System] - [Potential Energy of the System]

Note: Potential Energy only includes the energy based on gravity. The other passive properties are included in the generalized forces.

```
%Kinetic Energy: KE=0.5*m*v^2+0.5*I*w^2
T=(.5*m5*dot(vG5,vG5))+(.5*m4*dot(vG4,vG4))+(.5*m3*dot(vG3,vG3))+(.5*m2*dot(vG2,vG2))+(.5*m1*dot(vG1,vG1))+(.5*m0*dot(vG0,vG0))+(.5*dot((

T=subs(T,[diff(T5,t) diff(T4,t) diff(T3,t) diff(T2,t) diff(T1,t) diff(T0,t) diff(Tp,t)],[T5dot T4dot T3dot T2dot T1dot T0dot Tpdot]);
T=subs(T,[diff(A5,t) diff(A4,t) diff(A3,t) diff(A2,t) diff(A1,t) diff(A0,t) diff(Ap,t)],[A5dot A4dot A3dot A2dot A1dot A0dot Apdot]);
T=subs(T,[diff(B5,t) diff(B4,t) diff(B3,t) diff(B2,t) diff(B1,t) diff(B0,t) diff(Bp,t)],[B5dot B4dot B3dot B2dot B1dot B0dot Bpdot]);

%Potential Energy: PE=mgh
V=g*m5*G5(3)+g*m4*G4(3)+g*m3*G3(3)+g*m2*G2(3)+g*m1*G1(3)+g*m0*G0(3);
```

The Lagrangian Derivative is as follows:

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}_i} - \frac{\partial L}{\partial q_i} = Q_i$$

where *Qi*=generalized forces, *qi*=generalized coordinates, *i*=degrees of freedom.

Substituting in the Lagrangian function into the Lagrangian Derivative:

$$\frac{d}{dt}\frac{\partial T}{\partial \dot{q}_i} - \frac{d}{dt}\frac{\partial V}{\partial \dot{q}_i} - \frac{\partial T}{\partial q_i} + \frac{\partial V}{\partial q_i} = Q_i$$

Potential Energy not a function of velocity.

Resulting equation:

$$\frac{d}{dt}\frac{\partial T}{\partial \dot{q}_i} - \frac{\partial T}{\partial q_i} + \frac{\partial V}{\partial q_i} = Q_i$$

See local function EL for the implementation of this formula in our calculations.

```
%EL
ELT5=EL(T,V,T5(t),T5dot(t));
ELB5=EL(T,V,B5(t),B5dot(t));
ELA5=EL(T,V,A5(t),A5dot(t));

ELT4=EL(T,V,T4(t),T4dot(t));
ELB4=EL(T,V,B4(t),B4dot(t));
ELA4=EL(T,V,A4(t),A4dot(t));

ELT3=EL(T,V,T3(t),T3dot(t));
ELB3=EL(T,V,B3(t),B3dot(t));
ELA3=EL(T,V,A3(t),A3dot(t));

ELT2=EL(T,V,T2(t),T2dot(t));
ELB2=EL(T,V,B2(t),B2dot(t));
ELA2=EL(T,V,A2(t),A2dot(t));

ELT1=EL(T,V,T1(t),T1dot(t));
ELB1=EL(T,V,B1(t),B1dot(t));
ELA1=EL(T,V,A1(t),A1dot(t));

ELT0=EL(T,V,T0(t),T0dot(t));
ELB0=EL(T,V,B0(t),B0dot(t));
ELA0=EL(T,V,A0(t),A0dot(t));

EL_mat=[ELT5; ELB5; ELA5; ELT4; ELB4; ELA4; ELT3; ELB3; ELA3; ELT2; ELB2; ELA2; ELT1; ELB1; ELA1; ELT0; ELB0; ELA0];
```

Calculating the Mass Matrix, Gravity Vector and Coriolis/Crossterms Vector

```
[Mdd_ELB5,Gm_ELB5]=Mmatrix(ELB5);
[Mdd_ELA5,Gm_ELA5]=Mmatrix(ELA5);
[Mdd_ELT5,Gm_ELT5]=Mmatrix(ELT5);

[Mdd_ELB4,Gm_ELB4]=Mmatrix(ELB4);
[Mdd_ELA4,Gm_ELA4]=Mmatrix(ELA4);
[Mdd_ELT4,Gm_ELT4]=Mmatrix(ELT4);

[Mdd_ELB3,Gm_ELB3]=Mmatrix(ELB3);
[Mdd_ELA3,Gm_ELA3]=Mmatrix(ELA3);
[Mdd_ELT3,Gm_ELT3]=Mmatrix(ELT3);

[Mdd_ELB2,Gm_ELB2]=Mmatrix(ELB2);
[Mdd_ELA2,Gm_ELA2]=Mmatrix(ELA2);
[Mdd_ELT2,Gm_ELT2]=Mmatrix(ELT2);

[Mdd_ELB1,Gm_ELB1]=Mmatrix(ELB1);
[Mdd_ELA1,Gm_ELA1]=Mmatrix(ELA1);
[Mdd_ELT1,Gm_ELT1]=Mmatrix(ELT1);
```

```
[Mdd_ELB0,Gm_ELB0]=Mmatrix(ELB0);
[Mdd_ELA0,Gm_ELA0]=Mmatrix(ELA0);
[Mdd_ELT0,Gm_ELT0]=Mmatrix(ELT0);

Mm=[Mdd_ELT5; Mdd_ELB5; Mdd_ELA5; Mdd_ELT4; Mdd_ELB4; Mdd_ELA4; Mdd_ELT3; Mdd_ELB3; Mdd_ELA3; Mdd_ELT2; Mdd_ELB2; Mdd_ELA2; Mdd_ELT1; Mdd_
Gm=[Gm_ELT5 Gm_ELB5 Gm_ELA5 Gm_ELT4 Gm_ELB4 Gm_ELA4 Gm_ELT3 Gm_ELB3 Gm_ELA3 Gm_ELT2 Gm_ELB2 Gm_ELA2 Gm_ELT1 Gm_ELB1 Gm_ELA1 Gm_ELT0 Gm_ELB

CmT5=Cm_calc(ELT5,Mdd_ELT5,Gm_ELT5);
CmB5=Cm_calc(ELB5,Mdd_ELB5,Gm_ELB5);
CmA5=Cm_calc(ELA5,Mdd_ELA5,Gm_ELA5);

CmT4=Cm_calc(ELT4,Mdd_ELT4,Gm_ELT4);
CmB4=Cm_calc(ELB4,Mdd_ELB4,Gm_ELB4);
CmA4=Cm_calc(ELA4,Mdd_ELA4,Gm_ELA4);

CmT3=Cm_calc(ELT3,Mdd_ELT3,Gm_ELT3);
CmB3=Cm_calc(ELB3,Mdd_ELB3,Gm_ELB3);
CmA3=Cm_calc(ELA3,Mdd_ELA3,Gm_ELA3);

CmT2=Cm_calc(ELT2,Mdd_ELT2,Gm_ELT2);
CmB2=Cm_calc(ELB2,Mdd_ELB2,Gm_ELB2);
CmA2=Cm_calc(ELA2,Mdd_ELA2,Gm_ELA2);

CmT1=Cm_calc(ELT1,Mdd_ELT1,Gm_ELT1);
CmB1=Cm_calc(ELB1,Mdd_ELB1,Gm_ELB1);
CmA1=Cm_calc(ELA1,Mdd_ELA1,Gm_ELA1);

CmT0=Cm_calc(ELT0,Mdd_ELT0,Gm_ELT0);
CmB0=Cm_calc(ELB0,Mdd_ELB0,Gm_ELB0);
CmA0=Cm_calc(ELA0,Mdd_ELA0,Gm_ELA0);

Cm=[CmT5 CmB5 CmA5 CmT4 CmB4 CmA4 CmT3 CmB3 CmA3 CmT2 CmB2 CmA2 CmT1 CmB1 CmA1 CmT0 CmB0 CmA0];
```

---------- End of "Calculating the Mass Matrix, Gravity Vector and Coriolis/Crossterms Vector" Section

```
NwC5_Nbasis=formula(NwC5_Nbasis); NwC4_Nbasis=formula(NwC4_Nbasis); NwC3_Nbasis=formula(NwC3_Nbasis); NwC2_Nbasis=formula(NwC2_Nbasis); Nw

% Vars=[L0,L1,L2,L3,L4,L5,la,lb,cent3,g,m0,m1,m2,m3,m4,m5]; %symbolic vector
% Stateo=[T5 B5 A5 T4 B4 A4 T3 B3 A3 T2 B2 A2 T1 B1 A1 T0 B0 A0 T5dot B5dot A5dot T4dot B4dot A4dot T3dot B3dot A3dot T2dot B2dot A2dot T1
% Pango=[Tp Bp Ap Tpdot Bpdot Apdot]; %symbolic vector
[QL_0, QR_0]=help_Gl(Or0,NwC0_Nbasis);
[QL_1, QR_1]=help_Gl(Or1,NwC1_Nbasis);
[QL_2, QR_2]=help_Gl(Or2,NwC2_Nbasis);
[QL_3, QR_3]=help_Gl(Or3,NwC3_Nbasis);
[QL_4, QR_4]=help_Gl(Or4,NwC4_Nbasis);
[QL_5, QR_5]=help_Gl(Or5,NwC5_Nbasis);

Gl=[QL_0; QL_1; QL_2; QL_3; QL_4; QL_5];
Gr=[QR_0; QR_1; QR_2; QR_3; QR_4; QR_5];
```

Save the Symbolic Output

```
save('dynam_sym_output.mat','Mm','Cm','Gm','Gl','Gr','EL_mat')
```

```
function [output]=EL(T,V,ang,d_ang)
syms Apddot Bpddot Tpddot T5ddot B5ddot A5ddot T4ddot B4ddot A4ddot T3ddot B3ddot A3ddot T2ddot B2ddot A2ddot T1ddot B1ddot A1ddot T0ddot
syms la lb cent1 cent2 cent3 g t L0 L1 L2 L3 L4 L5 m0 m1 m2 m3 m4 m5
syms T5(t) T5dot(t) B5(t) B5dot(t) A5(t) A5dot(t) T4(t) T4dot(t) B4(t) B4dot(t) A4(t) A4dot(t) T3(t) T3dot(t) B3(t) B3dot(t) A3(t) A3dot(t
syms T5_val T5dot_val B5_val B5dot_val A5_val A5dot_val T4_val T4dot_val B4_val B4dot_val A4_val A4dot_val T3_val T3dot_val B3_val B3dot_v

ang_val=sym('ang');
d_ang_val=sym('d_ang');
```

Calculating the Lagrangian Derivative:

$$\underbrace{\frac{d}{dt}\frac{\partial T}{\partial \dot{q}_i}}_{\text{First term}} - \overbrace{\frac{\partial T}{\partial q_i}}^{\text{Second term}} + \underbrace{\frac{\partial V}{\partial q_i}}_{\text{Third term}} = Q_i$$

ang and *d_ang* were both as a function of time.

ang_val and *d_ang_val* were not a function of time.

```
T=subs(T,[ang d_ang],[ang_val d_ang_val]);
V=subs(V,ang,ang_val);
EL_1=diff(T,d_ang_val); %first term
```

```
EL_1=subs(EL_1,[ang_val d_ang_val],[ang d_ang]);
EL_2=diff(EL_1,t); %second step of first term
EL_3=diff(T,ang_val); %second term
EL_4=diff(V,ang_val); %third term

output=EL_2-EL_3+EL_4;
output=subs(output,[ang_val d_ang_val],[ang d_ang]);

output=subs(output,[diff(T5,t) diff(T4,t) diff(T3,t) diff(T2,t) diff(T1,t) diff(T0,t) diff(Tp,t)],[T5dot T4dot T3dot T2dot T1dot T0dot Tpc
output=subs(output,[diff(A5,t) diff(A4,t) diff(A3,t) diff(A2,t) diff(A1,t) diff(A0,t) diff(Ap,t)],[A5dot A4dot A3dot A2dot A1dot A0dot Apc
output=subs(output,[diff(B5,t) diff(B4,t) diff(B3,t) diff(B2,t) diff(B1,t) diff(B0,t) diff(Bp,t)],[B5dot B4dot B3dot B2dot B1dot B0dot Bpc

output=subs(output,[diff(T5dot,t) diff(T4dot,t) diff(T3dot,t) diff(T2dot,t) diff(T1dot,t) diff(T0dot,t) diff(Tpdot,t)],[T5ddot T4ddot T3dc
output=subs(output,[diff(A5dot,t) diff(A4dot,t) diff(A3dot,t) diff(A2dot,t) diff(A1dot,t) diff(A0dot,t) diff(Apdot,t)],[A5ddot A4ddot A3dc
output=subs(output,[diff(B5dot,t) diff(B4dot,t) diff(B3dot,t) diff(B2dot,t) diff(B1dot,t) diff(B0dot,t) diff(Bpdot,t)],[B5ddot B4ddot B3dc
end

function [Mdd,Gm]=Mmatrix(EL)
syms g Apddot Bpddot Tpddot T5ddot B5ddot A5ddot T4ddot B4ddot A4ddot T3ddot B3ddot A3ddot T2ddot B2ddot A2ddot T1ddot B1ddot A1ddot T0ddc
syms T5(t) T5dot(t) B5(t) B5dot(t) A5(t) A5dot(t) T4(t) T4dot(t) B4(t) B4dot(t) A4(t) A4dot(t) T3(t) T3dot(t) B3(t) B3dot(t) A3(t) A3dot(t
syms T5_val T5dot_val B5_val B5dot_val A5_val A5dot_val T4_val T4dot_val B4_val B4dot_val A4_val A4dot_val T3_val T3dot_val B3_val B3dot_v

Mdd=[diff(EL,T5ddot) diff(EL,B5ddot) diff(EL,A5ddot) diff(EL,T4ddot) diff(EL,B4ddot) diff(EL,A4ddot) diff(EL,T3ddot) diff(EL,B3ddot) diff(
    diff(EL,T2ddot) diff(EL,B2ddot) diff(EL,A2ddot) diff(EL,T1ddot) diff(EL,B1ddot) diff(EL,A1ddot) diff(EL,T0ddot) diff(EL,B0ddot) diff(E
Mdd=subs(Mdd,[T5 T5dot B5 B5dot A5 A5dot T4 T4dot B4 B4dot A4 A4dot T3 T3dot B3 B3dot A3 A3dot T2 T2dot B2 B2dot A2 A2dot T1 T1dot B1 B1dc

clear T5(t) T5dot(t) B5(t) B5dot(t) A5(t) A5dot(t) T4(t) T4dot(t) B4(t) B4dot(t) A4(t) A4dot(t) T3(t) T3dot(t) B3(t) B3dot(t) A3(t) A3dot(
syms T5 T5dot B5 B5dot A5 A5dot T4 T4dot B4 B4dot A4 A4dot T3 T3dot B3 B3dot A3 A3dot T2 T2dot B2 B2dot A2 A2dot T1 T1dot B1 B1dot A1 A1dc

Mdd=subs(Mdd,[T5_val T5dot_val B5_val B5dot_val A5_val A5dot_val T4_val T4dot_val B4_val B4dot_val A4_val A4dot_val T3_val T3dot_val B3_va
    [T5 T5dot B5 B5dot A5 A5dot T4 T4dot B4 B4dot A4 A4dot T3 T3dot B3 B3dot A3 A3dot T2 T2dot B2 B2dot A2 A2dot T1 T1dot B1 B1dot A1 A1dc

%Gm
Gm=g*diff(EL,g);
Gm=subs(Gm,[T5 T5dot B5 B5dot A5 A5dot T4 T4dot B4 B4dot A4 A4dot T3 T3dot B3 B3dot A3 A3dot T2 T2dot B2 B2dot A2 A2dot T1 T1dot B1 B1dot

clear T5(t) T5dot(t) B5(t) B5dot(t) A5(t) A5dot(t) T4(t) T4dot(t) B4(t) B4dot(t) A4(t) A4dot(t) T3(t) T3dot(t) B3(t) B3dot(t) A3(t) A3dot(
syms T5 T5dot B5 B5dot A5 A5dot T4 T4dot B4 B4dot A4 A4dot T3 T3dot B3 B3dot A3 A3dot T2 T2dot B2 B2dot A2 A2dot T1 T1dot B1 B1dot A1 A1dc

Gm=subs(Gm,[T5_val T5dot_val B5_val B5dot_val A5_val A5dot_val T4_val T4dot_val B4_val B4dot_val A4_val A4dot_val T3_val T3dot_val B3_val
    [T5 T5dot B5 B5dot A5 A5dot T4 T4dot B4 B4dot A4 A4dot T3 T3dot B3 B3dot A3 A3dot T2 T2dot B2 B2dot A2 A2dot T1 T1dot B1 B1dot A1 A1dc
end

function [Cm]=Cm_calc(EL,Mm,Gm)
syms T5(t) T5dot(t) B5(t) B5dot(t) A5(t) A5dot(t) T4(t) T4dot(t) B4(t) B4dot(t) A4(t) A4dot(t) T3(t) T3dot(t) B3(t) B3dot(t) A3(t) A3dot(t
syms Tpddot Bpddot Apddot T5ddot B5ddot A5ddot T4ddot B4ddot A4ddot T3ddot B3ddot A3ddot T2ddot B2ddot A2ddot T1ddot B1ddot A1ddot T0ddot
syms T5_val T5dot_val B5_val B5dot_val A5_val A5dot_val T4_val T4dot_val B4_val B4dot_val A4_val A4dot_val T3_val T3dot_val B3_val B3dot_v

d2_ang_vector=[T5ddot B5ddot A5ddot T4ddot B4ddot A4ddot T3ddot B3ddot A3ddot T2ddot B2ddot A2ddot T1ddot B1ddot A1ddot T0ddot B0ddot A0dc
Cm=EL-Mm*d2_ang_vector-Gm;
Cm=subs(Cm,[Apdot(t) Bpdot(t) Tpdot(t) T5dot(t) B5dot(t) A5dot(t) T4dot(t) B4dot(t) A4dot(t) T3dot(t) B3dot(t) A3dot(t) T2dot(t) B2dot(t)

clear T5(t) T5dot(t) B5(t) B5dot(t) A5(t) A5dot(t) T4(t) T4dot(t) B4(t) B4dot(t) A4(t) A4dot(t) T3(t) T3dot(t) B3(t) B3dot(t) A3(t) A3dot(
syms T5 T5dot B5 B5dot A5 A5dot T4 T4dot B4 B4dot A4 A4dot T3 T3dot B3 B3dot A3 A3dot T2 T2dot B2 B2dot A2 A2dot T1 T1dot B1 B1dot A1 A1dc
syms T5_val T5dot_val B5_val B5dot_val A5_val A5dot_val T4_val T4dot_val B4_val B4dot_val A4_val A4dot_val T3_val T3dot_val B3_val B3dot_v

Cm=subs(Cm,[Apdot Bpdot Tpdot T5dot B5dot A5dot T4dot B4dot A4dot T3dot B3dot A3dot T2dot B2dot A2dot T1dot B1dot A1dot T0dot B0dot A0dot]
Cm=subs(Cm,[T5 T5dot B5 B5dot A5 A5dot T4 T4dot B4 B4dot A4 A4dot T3 T3dot B3 B3dot A3 A3dot T2 T2dot B2 B2dot A2 A2dot T1 T1dot B1 B1dot
Cm=subs(Cm,[T5_val T5dot_val B5_val B5dot_val A5_val A5dot_val T4_val T4dot_val B4_val B4dot_val A4_val A4dot_val T3_val T3dot_val B3_val
    [T5 T5dot B5 B5dot A5 A5dot T4 T4dot B4 B4dot A4 A4dot T3 T3dot B3 B3dot A3 A3dot T2 T2dot B2 B2dot A2 A2dot T1 T1dot B1 B1dot A1 A1dc
end

function [QL_vect QR_vect]=help_Gl(Or,NwC)
syms Tpddot Bpddot Apddot
syms T5(t) T5dot(t) B5(t) B5dot(t) A5(t) A5dot(t) T4(t) T4dot(t) B4(t) B4dot(t) A4(t) A4dot(t) T3(t) T3dot(t) B3(t) B3dot(t) A3(t) A3dot(t
syms T5_val T5dot_val B5_val B5dot_val A5_val A5dot_val T4_val T4dot_val B4_val B4dot_val A4_val A4dot_val T3_val T3dot_val B3_val B3dot_v

QL_first=diff(Or,t);
QL_second=subs(QL_first,[diff(B5,t) diff(B4,t) diff(B3,t) diff(B2,t) diff(B1,t) diff(B0,t) diff(Bp,t) diff(T5,t) diff(T4,t) diff(T3,t) dif
QL=subs(QL_second,[T5 T5dot B5 B5dot A5 A5dot T4 T4dot B4 B4dot A4 A4dot T3 T3dot B3 B3dot A3 A3dot T2 T2dot B2 B2dot A2 A2dot T1 T1dot B1
QL_1=diff(QL,T5dot_val); QL_2=diff(QL,B5dot_val); QL_3=diff(QL,A5dot_val);
QL_4=diff(QL,T4dot_val); QL_5=diff(QL,B4dot_val); QL_6=diff(QL,A4dot_val);
QL_7=diff(QL,T3dot_val); QL_8=diff(QL,B3dot_val); QL_9=diff(QL,A3dot_val);
QL_10=diff(QL,T2dot_val); QL_11=diff(QL,B2dot_val); QL_12=diff(QL,A2dot_val);
QL_13=diff(QL,T1dot_val); QL_14=diff(QL,B1dot_val); QL_15=diff(QL,A1dot_val);
QL_16=diff(QL,T0dot_val); QL_17=diff(QL,B0dot_val); QL_18=diff(QL,A0dot_val);

QL_vect=[QL_1; QL_2; QL_3; QL_4; QL_5; QL_6; QL_7; QL_8; QL_9; QL_10; QL_11; QL_12; QL_13; QL_14; QL_15; QL_16; QL_17; QL

clear T5 T5dot B5 B5dot A5 A5dot T4 T4dot B4 B4dot A4 A4dot T3 T3dot B3 B3dot A3 A3dot T2 T2dot B2 B2dot A2 A2dot T1 T1dot B1 B1dot A1 A1c
syms T5(t) T5dot(t) B5(t) B5dot(t) A5(t) A5dot(t) T4(t) T4dot(t) B4(t) B4dot(t) A4(t) A4dot(t) T3(t) T3dot(t) B3(t) B3dot(t) A3(t) A3dot(t
```

```
QR=subs(NwC,[T5 T5dot B5 B5dot A5 A5dot T4 T4dot B4 B4dot A4 A4dot T3 T3dot B3 B3dot A3 A3dot T2 T2dot B2 B2dot A2 A2dot T1 T1dot B1 B1dot
QR_1=diff(QR,T5dot_val); QR_2=diff(QR,B5dot_val); QR_3=diff(QR,A5dot_val);
QR_4=diff(QR,T4dot_val); QR_5=diff(QR,B4dot_val); QR_6=diff(QR,A4dot_val);
QR_7=diff(QR,T3dot_val); QR_8=diff(QR,B3dot_val); QR_9=diff(QR,A3dot_val);
QR_10=diff(QR,T2dot_val); QR_11=diff(QR,B2dot_val); QR_12=diff(QR,A2dot_val);
QR_13=diff(QR,T1dot_val); QR_14=diff(QR,B1dot_val); QR_15=diff(QR,A1dot_val);
QR_16=diff(QR,T0dot_val); QR_17=diff(QR,B0dot_val); QR_18=diff(QR,A0dot_val);

QR_vect=[QR_1'; QR_2'; QR_3'; QR_4'; QR_5'; QR_6'; QR_7'; QR_8'; QR_9'; QR_10'; QR_11'; QR_12'; QR_13'; QR_14'; QR_15'; QR_16'; QR_17'; QR

clear T5(t) T5dot(t) B5(t) B5dot(t) A5(t) A5dot(t) T4(t) T4dot(t) B4(t) B4dot(t) A4(t) A4dot(t) T3(t) T3dot(t) B3(t) B3dot(t) A3(t) A3dot(
syms T5 T5dot B5 B5dot A5 A5dot T4 T4dot B4 B4dot A4 A4dot T3 T3dot B3 B3dot A3 A3dot T2 T2dot B2 B2dot A2 A2dot T1 T1dot B1 B1dot A1 A1dc

QL_vect=subs(QL_vect,[T5_val T5dot_val B5_val B5dot_val A5_val A5dot_val T4_val T4dot_val B4_val B4dot_val A4_val A4dot_val T3_val T3dot_v
    [T5 T5dot B5 B5dot A5 A5dot T4 T4dot B4 B4dot A4 A4dot T3 T3dot B3 B3dot A3 A3dot T2 T2dot B2 B2dot A2 A2dot T1 T1dot B1 B1dot A1 A1dc
QR_vect=subs(QR_vect,[T5_val T5dot_val B5_val B5dot_val A5_val A5dot_val T4_val T4dot_val B4_val B4dot_val A4_val A4dot_val T3_val T3dot_v
    [T5 T5dot B5 B5dot A5 A5dot T4 T4dot B4 B4dot A4 A4dot T3 T3dot B3 B3dot A3 A3dot T2 T2dot B2 B2dot A2 A2dot T1 T1dot B1 B1dot A1 A1dc

end

function [NwA_Abasis,NwA_Nbasis]=ang_vel(R3,R123,ang1,ang2,ang3,Tdot,Bdot,Adot)
syms q1 q2 q3
syms Ap(t) Bp(t) Tp(t) T5(t) B5(t) A5(t) T4(t) B4(t) A4(t) T3(t) B3(t) A3(t) T2(t) B2(t) A2(t) T1(t) B1(t) A1(t) T0(t) B0(t) A0(t)

R123=subs(R123,[q1 q2 q3],[ang1 ang2 ang3]);
R3=subs(R3,[q3],[ang3]);

Axis1_rotate=R123(1,:); %N1
Axis2_rotate=R3(2,:); %T2=B2
Axis3_rotate=R3(3,:); %B3

NwA_Abasis=Tdot*Axis1_rotate+Bdot*Axis2_rotate+Adot*Axis3_rotate; %in A basis

R1=[1 0 0; 0 cos(q1) -sin(q1); 0 sin(q1) cos(q1)]; %Rotation axis 1
R1=subs(R1,[q1],[ang1]);

Axis1_rotateN=R1(:,1); %N1=T1
Axis2_rotateN=R1(:,2); %T2=B2
Axis3_rotateN=R123(:,3); %B3=A3
NwA_Nbasis=Tdot*Axis1_rotateN+Bdot*Axis2_rotateN+Adot*Axis3_rotateN; %in N basis
end
```

C.12 *generalizedForce.mlx*: Generalized Forces

Valerie Jardon

Department of Bioengineering, University of Kansas

generalizedForce.mlx

Last Edited: 4/29/2022

This file (generalizedForce.mlx) is based on the genforcesc.c codes developed by Timothy C. Franklin, Virginia Polytechnic Institute and State University. The genforcesc.c codes were obtained through means of personal communication (Michael Madigan, personal communication, July 26, 2008).

Generalized Forces

Description: In this script, the generalized forces for the degrees of freedom can be determined. First, the intervertebral disc moments are considered. These are applied to the bodies as pure moments. Following, the muscle forces are considered. Lastly, the external forces are considered. The resulting Q vector will be an 18x1 vector due to the 18 degrees of freedom.

$$Q_i = \sum_{b=1}^{6}\left(\left(\sum_{e=1}^{n} F_{be}\right)\cdot\frac{\partial v_b}{\partial \dot{q}_i} + \left(\sum_{e=1}^{n} r_{be}\times F_{be}\right)\cdot\frac{\partial \omega_b}{\partial \dot{q}_i}\right)$$

Where b is the index of the rigid bodies, F_{be} represents force applied to the body, e is the index of the applied forces, i is the degree of freedom index, v_b is the linear velocity of the origin of the body, ω_b is the angular velocity of the body and r_{be} is a vector between the point where the force is applied and the origin of the body.

Reference:

Franklin, T. C. (2006). *Linear System Analyses of the Role of Reflex Gain and Delay in a Dynamic Human Spine Model* (Master's thesis, Thesis / Dissertation ETD). Blacksburg: Virginia Tech. Retrieved June 19, 2020, from https://vtechworks.lib.vt.edu/handle/10919/33605.

```
function [Q]=generalizedForce(Ma,MF,Ea,EF,PRQ,Gl,Gr,Or)

%Initialize Variables
DOF=18; %degrees of freedom
nM=size(Ma,1); %number of muscles
nE=size(Ea,1); %number of external forces
Q=zeros(DOF,1); %initializing the output array

% Intervertebral Disc Moments - applied as pure moments
% Only the second term in the generalized force eqn will be used for IVDs
for b=0:5 %body number
    for j=1:DOF %degrees of freedom
        Gl_b=Gl(b*18+j,:);
        Gr_b=Gr(b*18+j,:);
        PRQ_b=PRQ(b+1,:);
        Q(j)=Q(j)+dot(PRQ_b,Gr_b); %generalized force vector
    end
end

% Muscle Force
for i=1:nM %muscle number
        for j=1:DOF %degrees of freedom

      %------- if the muscle has only two attachment points
            if Ma(i,1)==2 %if the muscle has two attachment points(origin and terminal)
                b1=Ma(i,2); %muscle origin point body number
                muscle_origin_coords=Ma(i,3:5); %muscle origin coordinates
                b2=Ma(i,6); %muscle terminal point body number
                muscle_term_coords=Ma(i,7:9); %muscle terminal coordinates
                origin_b1=Or(b1+1,:); %origin coordinates of body 1
                origin_b2=Or(b2+1,:); %origin coordinates of body 2

                %for the origin
                    if b1<6
                        [F_vector1,M_vector1]=M_F_vects(muscle_origin_coords,muscle_term_coords,[],origin_b1,'origin');
                        Gl_b1=Gl(b1*18+j,:);
                        Gr_b1=Gr(b1*18+j,:);
                        Q(j)=Q(j)+MF(i)*(dot(F_vector1,Gl_b1)+dot(M_vector1,Gr_b1));
                    end
```

```
            %for the terminal
                if b2<6
                    [F_vector2,M_vector2]=M_F_vects(muscle_origin_coords,muscle_term_coords,[],origin_b2,'terminal');
                    Gl_b2=Gl(b2*18+j,:);
                    Gr_b2=Gr(b2*18+j,:);
                    Q(j)=Q(j)+MF(i)*(dot(F_vector2,Gl_b2)+dot(M_vector2,Gr_b2));
                end

    %------- if the muscle has more than two attachment points
        else
            points_num=Ma(i,1); %determine the number of points
            node_num=points_num-2; %calculate the number of nodes

            % for the origin
                b1=Ma(i,2); %muscle origin point body number
                muscle_origin_coords=Ma(i,3:5); %muscle origin coordinates
                b2=Ma(i,6); %muscle node 1 body
                muscle_term_coords=Ma(i,7:9); %muscle node 1 coordinates
                origin_b1=Or(b1+1,:); %origin coordinates of body 1
            if b1<6
                [F_vector1,M_vector1]=M_F_vects(muscle_origin_coords,muscle_term_coords,[],origin_b1,'origin');
                Gl_b1=Gl(b1*18+j,:);
                Gr_b1=Gr(b1*18+j,:);
                Q(j)=Q(j)+MF(i)*(dot(F_vector1,Gl_b1)+dot(M_vector1,Gr_b1));
            end

            for n=1:node_num
                b1=Ma(i,2+4*(n-1)); %point 1 body
                b2=Ma(i,2+4*n); %point 2 body
                b3=Ma(i,2+4*(n+1)); %point 3 body

                point1_coords=Ma(i,3+4*(n-1):5+4*(n-1)); %point 1 coordinates
                point2_coords=Ma(i,3+4*n:5+4*n); %point 2 coordinates
                point3_coords=Ma(i,3+4*(n+1):5+4*(n+1)); %point 3 coordinates

                origin_b2=Or(b2+1,:); %origin coordinates of body 2

                if b2<6
                    [F_vector1,M_vector1]=M_F_vects(point1_coords,point2_coords,point3_coords,origin_b2,'node');
                    Gl_b2=Gl(b2*18+j,:);
                    Gr_b2=Gr(b2*18+j,:);
                    Q(j)=Q(j)+MF(i)*(dot(F_vector1,Gl_b2)+dot(M_vector1,Gr_b2));
                end
            end

            %for the terminal
                b1=Ma(i,2+4*(points_num-2)); %node before terminal
                node_coords=Ma(i,3+4*(points_num-2):5+4*(points_num-2)); %node before terminal coordinates
                b2=Ma(i,2+4*(points_num-1)); %terminal point body
                muscle_term_coords=Ma(i,3+4*(points_num-1):5+4*(points_num-1)); %terminal coordinates
                origin_b2=Or(b2+1,:); %origin coordinates of body 2
            if b2<6
                [F_vector2,M_vector2]=M_F_vects(node_coords,muscle_term_coords,[],origin_b2,'terminal');
                Gl_b2=Gl(b2*18+j,:);
                Gr_b2=Gr(b2*18+j,:);
                Q(j)=Q(j)+MF(i)*(dot(F_vector2,Gl_b2)+dot(M_vector2,Gr_b2));
            end
        end
    end
end

% External Forces
for i=1:nE %number of external forces applied
    for j=1:18 %degrees of freedom
        if Ea(1,1)~=2
            error('Error. External force needs to have two points.')
        else
            b1=Ea(i,2); %force origin body number
            force_origin_coords=Ea(i,3:5); %force origin coordinates
            b2=Ea(i,6); %force terminal point body number
            force_term_coords=Ea(i,7:9); %force terminal coordinates

            %for the origin
                if b1<6
                    origin_b1=Or(b1+1,:); %origin coordinates of body 1
```

```
                [F_vector1,M_vector1]=M_F_vects(force_origin_coords,force_term_coords,[],origin_b1,'origin');
                Gl_b1=Gl(b1*18+j,:);
                Gr_b1=Gr(b1*18+j,:);
                Q(j)=Q(j)+EF(i)*(dot(F_vector1,Gl_b1)+dot(M_vector1,Gr_b1));
            end

        %for the terminal
            if b2<6
                origin_b2=Or(b2+1,:); %origin coordinates of body 2
                [F_vector2,M_vector2]=M_F_vects(force_origin_coords,force_term_coords,[],origin_b2,'terminal');
                Gl_b2=Gl(b2*18+j,:);
                Gr_b2=Gr(b2*18+j,:);
                Q(j)=Q(j)+EF(i)*(dot(F_vector2,Gl_b2)+dot(M_vector2,Gr_b2));
            end
        end
    end
end
```

```
function [vect_out]=unit_vect(vect_in)
     vect_mag=sqrt(sum(vect_in.^2));
     vect_out=vect_in./vect_mag; %calculate unit vector
end


function [F_vector,M_vector]=M_F_vects(muscle_p1_coords,muscle_p2_coords,muscle_p3_coords,body_origin,point_type)
% calculating F and M= r x F

    if strcmp(point_type,'origin')
        vect_btw_coords=muscle_p1_coords-muscle_p2_coords; %vector from point to point
        F_vector=unit_vect(vect_btw_coords); %unit vector

        origins_diff_vect=muscle_p1_coords-body_origin; %difference between body origin coordinates and muscle point coordinates
        M_vector=cross(origins_diff_vect,F_vector);

    elseif strcmp(point_type,'terminal')

        vect_btw_coords=muscle_p2_coords-muscle_p1_coords; %vector from point to point
        F_vector=unit_vect(vect_btw_coords); %unit vector

        origins_diff_vect=muscle_p2_coords-body_origin; %difference between body origin coordinates and muscle point coordinates
        M_vector=cross(origins_diff_vect,F_vector);

    elseif strcmp(point_type,'node')
        vect_btw_coords_21=muscle_p2_coords-muscle_p1_coords; %vector from origin point to terminal point
        vect_btw_coords_23=muscle_p2_coords-muscle_p3_coords; %vector from origin point to terminal point
        F_vector_1=unit_vect(vect_btw_coords_21); %unit vector
        F_vector_2=unit_vect(vect_btw_coords_23); %unit vector
        F_vector=F_vector_1+F_vector_2;

        origins_diff_vect=muscle_p2_coords-body_origin; %difference between body origin coordinates and muscle point coordinates
        M_vector=cross(origins_diff_vect,F_vector);
    end
end
```

C.13 *optimize.mlx*: Optimization

C.13.1 Main File

```
% Valerie Jardon
% Department of Bioengineering, University of Kansas
% optimize.mlx
% Last Edited: 4/30/2022

% Based on the optimizationscript.m code developed by Timothy C. Franklin,
% Virginia Polytechnic Institute and State University.
%% Description
%     In the optimize.m file, muscle activation is determined for the
%     system while the metabolic power is minimized, and the system is stable
%     and in equilibrium. This is achieved by using the MATLAB fmincon
%     built-in function which allows for a minimum to be determined while
%     meeting certain constraints.

%     This file (optimize.m) is based on the optimizationscript.m file
%     developed by Timothy C. Franklin, published in Appendix A of his thesis (pages ↙
85-87).

% ----- Reference:
%    Franklin, T. C. (2006). Linear System Analyses of the Role of Reflex Gain and ↙
Delay in a Dynamic Human Spine Model...
%    (Master's thesis, Virginia Polytechnic Institute and State University) (pp. 85- ↙
87). Virginia Tech. Retrieved June 19, 2020,...
%    from https://vtechworks.lib.vt.edu/handle/10919/33605?show=full.
%-------------------------------------------------------------------------
%-------------------------------------------------------------------------

%%
%This section is based on the "Optimization Settings" and "Equality Assembly" ↙
sections of the optimizationscript.m file
%developed by Timothy C. Franklin, published in Appendix A of his thesis.
%-------------------------------------------------------------------------

% Initializing Variables
Uo=zeros(size(M,1),1); Un=Uo; %creating zeros matrices
Po=zeros(size(M,1),1); Pn=Po; %creating zeros matrices
CSA=Ma(:,34);

eps=0.00001*pi/180; %disturbance
length_array=zeros(90,1);

global PrevEig JPrevEig cequal optiter
            PrevEig=zeros(18,1); % global v
            JPrevEig=zeros(36,1); % settings
            optiter=0;
            cequal=zeros(18,1);

%Initializing the constraints for fmincon
Aeq=zeros(18,90); %equilibrium
```

```
beq=zeros(18,1); %equilibrium
A=-eye(90);
B=zeros(90,1);
%% Equilibrium Constraints

disp('-----Determining Equilibrium Conditions - Passive-----')
% Includes intervertebral disc moments and external force
% Does NOT include muscle force

U=zeros(90,1); %activation = 0 for passive
beq=-spineaccP(0,Stateo,Pango,U,EFo,Vars,M,E,Lord_Angs,IVD,Gl_funct,Gr_funct,
Mm_funct,Cm_funct,Gm_funct,Ma_funct,ML_funct,MV_funct,Ea_funct,Rm_funct,dRm_funct,
origin_funct,dorigin_funct,ChSpace,[]);


disp('-----Determining Equilibrium Conditions - Active-----')
% Includes Muscle Forces
% Does NOT include intervertebral disc moments or external force

for i=1:length(length_array)
    U=zeros(90,1); %initializing a vector
    U(i)=1;
    [Aeq(:,i),q,b,Gp,Gd]=spineaccA(i,0,Stateo,Pango,U,Pn,EFo,Vars,M,E,Lo,CSA,
Gl_funct,Gr_funct,Mm_funct,Cm_funct,Gm_funct,Ma_funct,ML_funct,MV_funct,Ea_funct,
Rm_funct,dRm_funct,origin_funct,dorigin_funct,ChSpace,[]);
end
%% Constructing the Jacobians
%This section is based on the "Jacobian Assembly", "Reflex" and "For Optimization (no
time delay assumption)" sections
%of the optimizationscript.m file developed by Timothy C. Franklin, published in
Appendix A of his thesis.
%-------------------------------------------------------------------------
disp('-----Building the Jacobians.-----')
tic

% Initializing Jacobian Matrices
Jpi=zeros(36,36);
Jpi(1:18,19:36)=eye(18,18);
Jai=zeros(size(M,1),36,36);
Jpr=zeros(36,36); %no passive contribution
Jar=zeros(size(M,1),36,36);

    for h=1:36
        dStateo=Stateo;
        dStateo(h)=dStateo(h)+eps;
            for i=1:90
                Uo=zeros(size(M,1),1);
                Uo(i)=1;
                [qddA,qddP,qddR]=spineacc(i,dStateo,Pango,Uo,Pn,EFo,Vars,Lo,CSA,
```

```
Lord_Angs,IVD,Gl_funct,Gr_funct,Mm_funct,Cm_funct,Gm_funct,Ma_funct,ML_funct, ↙
MV_funct,Ea_funct,Rm_funct,dRm_funct,origin_funct,dorigin_funct,ChSpace,[]);
                Jai(i,19:36,h)=(qddA-Aeq(:,i))/eps;
                Jpi(19:36,h)=(qddP+beq)/eps;
                Jar(i,19:36,h)=(qddR)/eps;
            end
    end

elapsedTime=toc;
time_c=elapsedTime/60; time_c=fix(time_c);
fprintf('Building the Jacobian took %2d minutes and %3.1f seconds. \n',time_c,↙
(elapsedTime-(time_c*60)))

%% For Optimization - No Time Delay Assumption
Jp=Jpi+Jpr;
Ja=Jai+Jar;
save('Jacobians.mat','Ja','Jp','Jai','Jar','Jpi','Jpr')
%% Perform the Constrained Optimization Procedure

disp('-----Running fmincon to determine muscle activation-----')

% Run fmincon to determine muscle activation and metabolic power
[exitflag_success,fval_success,Un_total_success,Un_total_initial_success]=run_fmincon ↙
(A,B,Aeq,beq,CSA,Lo,Stateo,Jp,Ja,M);

% Check the optimization output - is the system stable and in equilibrium?
max_eig_vect=zeros(numel(exitflag_success),1);
for h=1:numel(exitflag_success)
    if fval_success(h)~=Inf
    Un_opt=Un_total_success(:,h);
    J=Jp;
        for i=1:36
            J(19:36,i)=J(19:36,i)+(Un_opt'*Ja(:,19:36,i))';
        end

% Check the eigenvalues
    eig_val=eig(J);
    max_eig_vect(h)=max(real(eig_val));
    if max_eig_vect(h)>=0
        fval_success(h)=Inf;
    end
% Plot the eigenvalues
    figure(5)
    plot(real(eig_val),imag(eig_val),'r*'); xlabel('Real'); ylabel('Imaginary');
    grid on

% Check Equilibrium Condition
    equilibrium_constraint=Aeq*Un_opt-beq;
    equilibrium_constaint_max=max(abs(equilibrium_constraint));
```

```
    if equilibrium_constaint_max>1e-5
        fval_success(h)=100000000;
    end
    end
end

% Determine the optimal trial
[M_best,I_best]=min(fval_success);
Un_initial_best=Un_total_initial_success(:,I_best);
Un_best=Un_total_success(:,I_best); %resulting activations
exitflag_best=exitflag_success(I_best);

%% Display Optimization Results

% For the optimal solution
Un_opt=Un_best;
    J=Jp;
        for i=1:36
            J(19:36,i)=J(19:36,i)+(Un_opt'*Ja(:,19:36,i))';
        end

% Check the eigenvalues
    eig_val=eig(J);
    max_eig=max(real(eig_val));

% Plot the eigenvalues
    figure(5)
    plot(real(eig_val),imag(eig_val),'r*'); xlabel('Real'); ylabel('Imaginary');
    grid on

% Check Equilibrium Condition
    equilibrium_constraint=Aeq*Un_opt-beq;
    equilibrium_constaint_max=max(abs(equilibrium_constraint));

disp('Optimization Successful. System is stable and in equilibrium.')
disp(['fval=' num2str(M_best) ' Watts'])
disp(['Maximum real part of eigenvalue=' num2str(max_eig)])
disp(['Maximum equilibrium error=' num2str(equilibrium_constaint_max)])
disp('-----End of Optimization Procedure.-----')
```

C.13.2 Angular Acceleration Functions

spineaccP.m file utilized for equilibrium constraint

```
% Valerie Jardon
% Department of Bioengineering, University of Kansas
% spineaccP.m
% Last Edited: 4/30/2022
% This file (spineaccP.m) is based on the AccelerationMP.m file developed by Timothy C. Franklin, Virginia Polytechnic Institute and State University.
% The AccelerationMP.m file was obtained through means of personal communication (Michael Madigan, personal communication, July 26, 2008).

% Reference:
% Franklin, T. C. (2006). Linear System Analyses of the Role of Reflex Gain and Delay in a Dynamic Human Spine Model (Master's thesis, Thesis / Dissertation ETD). Blacksburg: Virginia Tech. Retrieved June 19, 2020, from https://vtechworks.lib.vt.edu/handle/10919/33605.
%
--------------------------------------------------------------------------------
----------------------------------------


function [qdd]=spineaccP(t,Stateo,Pango,U,EFo,Vars,M,E,Lord_Angs,IVD,Gl_funct, Gr_funct,Mm_funct,Cm_funct,Gm_funct,Ma_funct,ML_funct,MV_funct,Ea_funct,Rm_funct, dRm_funct,origin_funct,dorigin_funct,ChSpace,verification)
% acceleration due to passive

[A0,A1,A2,A3,A4,A5,Ap,B0,B1,B2,B3,B4,B5,Bp,DA0,DA1,DA2,DA3,DA4,DA5,DAp,DB0,DB1,DB2, DB3,DB4,DB5,DBp,DT0,DT1,DT2,DT3,DT4,DT5,DTp,T0,T1,T2,T3,T4,T5,Tp, ...
    A0dot,A1dot,A2dot,A3dot,A4dot,A5dot,T0dot,T1dot,T2dot,T3dot,T4dot,T5dot,B0dot, B1dot,B2dot,B3dot,B4dot,B5dot,Apdot,Bpdot,Tpdot]=vect2ang(Stateo,Pango);

T5ddot=0; B5ddot=0; A5ddot=0;
T4ddot=0; B4ddot=0; A4ddot=0;
T3ddot=0; B3ddot=0; A3ddot=0;
T2ddot=0; B2ddot=0; A2ddot=0;
T1ddot=0; B1ddot=0; A1ddot=0;
T0ddot=0; B0ddot=0; A0ddot=0;
Tpddot=0; Bpddot=0; Apddot=0;

L0=Vars(1); L1=Vars(2); L2=Vars(3); L3=Vars(4); L4=Vars(5); L5=Vars(6);
la=Vars(7); lb=Vars(8); g=Vars(10);
cent1=Vars(17); cent2=Vars(18); cent3=Vars(9);
m0=Vars(11); m1=Vars(12); m2=Vars(13); m3=Vars(14); m4=Vars(15); m5=Vars(16);

% Recalculate due to the disturbance
Gl=Gl_funct(B1,B2,B3,B4,B5,L1,L2,L3,L4,L5,T1,T2,T3,T4,T5);
Gr=Gr_funct(B0,B1,B2,B3,B4,B5,T0,T1,T2,T3,T4,T5);
Mm=Mm_funct(A0,A1,A2,A3,A4,A5,B0,B1,B2,B3,B4,B5,L0,L1,L2,L3,L4,L5,T0,T1,T2,T3,T4,T5, la,lb,m0,m1,m2,m3,m4,m5);
Cm=Cm_funct(A0,A1,A2,A3,A4,A5,A0dot,A1dot,A2dot,A3dot,A4dot,A5dot,A0ddot,A1ddot, A2ddot,A3ddot,A4ddot,A5ddot,Ap,Apdot,Apddot,B0,B1,B2,B3,B4,B5,B0dot,B1dot,B2dot,
```

```
B3dot,B4dot,B5dot,B0ddot,B1ddot,B2ddot,B3ddot,B4ddot,B5ddot,Bp,Bpdot,Bpddot,L0,L1,L2,
L3,L4,L5,T0,T1,T2,T3,T4,T5,T0dot,T1dot,T2dot,T3dot,T4dot,T5dot,T0ddot,T1ddot,T2ddot,
T3ddot,T4ddot,T5ddot,Tp,Tpdot,Tpddot,cent1,cent2,cent3,g,la,lb,m0,m1,m2,m3,m4,m5);
Gm=Gm_funct(B0,B1,B2,B3,B4,B5,L0,L1,L2,L3,L4,L5,T0,T1,T2,T3,T4,T5,g,m0,m1,m2,m3,m4,
m5);
Ma=Ma_funct(A0,A1,A2,A3,A4,A5,Ap,B0,B1,B2,B3,B4,B5,Bp,L1,L2,L3,L4,L5,T0,T1,T2,T3,T4,
T5,Tp,cent1,cent2,cent3);
Ea=Ea_funct(A0,Ap,B0,B1,B2,B3,B4,B5,Bp,L1,L2,L3,L4,L5,T0,T1,T2,T3,T4,T5,Tp,cent1,
cent2,cent3);
rotation_matrices=Rm_funct(A0,A1,A2,A3,A4,A5,Ap,B0,B1,B2,B3,B4,B5,Bp,T0,T1,T2,T3,T4,
T5,Tp);
d_rotation_matrices=dRm_funct(A0,A1,A2,A3,A4,A5,Ap,B0,B1,B2,B3,B4,B5,Bp,DA0,DA1,DA2,
DA3,DA4,DA5,DAp,DB0,DB1,DB2,DB3,DB4,DB5,DBp,DT0,DT1,DT2,DT3,DT4,DT5,DTp,T0,T1,T2,T3,
T4,T5,Tp);
origin=origin_funct(Ap,B1,B2,B3,B4,B5,Bp,L1,L2,L3,L4,L5,T1,T2,T3,T4,T5,Tp,cent1,
cent2,cent3);

PRQ=IVD_calcs(rotation_matrices,d_rotation_matrices,Stateo,Pango,Lord_Angs,IVD);
PRQ=PRQ(1:6,:);

% for the verification script
    if strcmp('verification',verification)
        Cm=zeros(size(Cm)); %verification Cm
        PRQ=[1 2 3; 4 5 6; 7 8 9; 10 11 12; 1 2 3; 5 5 4; 1 2 3]; %verification PRQ
        PRQ=PRQ(1:6,:);
    end

MF=zeros(size(U)); %no muscle force

Q=generalizedForce(Ma,MF,Ea,EFo,PRQ,Gl,Gr,origin);

% solve for the angular acceleration
qdd=Mm\(Q-transpose(Cm)-transpose(Gm));
end
```

spineaccA.m file utilized for equilibrium constraint

```
% Valerie Jardon
% Department of Bioengineering, University of Kansas
% spineaccA.m
% Last Edited: 4/30/2022
% This file (spineaccA.m) is based on the AccelerationMA.m file developed by Timothy
C. Franklin, Virginia Polytechnic Institute and State University.
% The AccelerationMA.m file was obtained through means of personal communication
(Michael Madigan, personal communication, July 26, 2008).

% Reference:
% Franklin, T. C. (2006). Linear System Analyses of the Role of Reflex Gain and Delay
in a Dynamic Human Spine Model (Master's thesis, Thesis / Dissertation ETD).
Blacksburg: Virginia Tech. Retrieved June 19, 2020, from https://vtechworks.lib.vt.
edu/handle/10919/33605.
%
--------------------------------------------------------------------------------
----------------------------------------

function [qdd,q,b,Gp,Gd]=spineaccA(i,t,Stateo,Pango,U,P,EFo,Vars,M,E,Lo,CSA,Gl_funct,
Gr_funct,Mm_funct,Cm_funct,Gm_funct,Ma_funct,ML_funct,MV_funct,Ea_funct,Rm_funct,
dRm_funct,origin_funct,dorigin_funct,ChSpace,verification)
% acceleration due to active
```

```
[A0,A1,A2,A3,A4,A5,Ap,B0,B1,B2,B3,B4,B5,Bp,DA0,DA1,DA2,DA3,DA4,DA5,DAp,DB0,DB1,DB2,
DB3,DB4,DB5,DBp,DT0,DT1,DT2,DT3,DT4,DT5,DTp,T0,T1,T2,T3,T4,T5,Tp, ...
    A0dot,A1dot,A2dot,A3dot,A4dot,A5dot,T0dot,T1dot,T2dot,T3dot,T4dot,T5dot,B0dot,
B1dot,B2dot,B3dot,B4dot,B5dot,Apdot,Bpdot,Tpdot]=vect2ang(Stateo,Pango);

T5ddot=0; B5ddot=0; A5ddot=0;
T4ddot=0; B4ddot=0; A4ddot=0;
T3ddot=0; B3ddot=0; A3ddot=0;
T2ddot=0; B2ddot=0; A2ddot=0;
T1ddot=0; B1ddot=0; A1ddot=0;
T0ddot=0; B0ddot=0; A0ddot=0;
Tpddot=0; Bpddot=0; Apddot=0;

L0=Vars(1); L1=Vars(2); L2=Vars(3); L3=Vars(4); L4=Vars(5); L5=Vars(6);
la=Vars(7); lb=Vars(8); g=Vars(10);
cent1=Vars(17); cent2=Vars(18); cent3=Vars(9);
m0=Vars(11); m1=Vars(12); m2=Vars(13); m3=Vars(14); m4=Vars(15); m5=Vars(16);

% Recalculate due to the disturbance
Gl=Gl_funct(B1,B2,B3,B4,B5,L1,L2,L3,L4,L5,T1,T2,T3,T4,T5);
Gr=Gr_funct(B0,B1,B2,B3,B4,B5,T0,T1,T2,T3,T4,T5);
Mm=Mm_funct(A0,A1,A2,A3,A4,A5,B0,B1,B2,B3,B4,B5,L0,L1,L2,L3,L4,L5,T0,T1,T2,T3,T4,T5,
la,lb,m0,m1,m2,m3,m4,m5);
Ma=Ma_funct(A0,A1,A2,A3,A4,A5,Ap,B0,B1,B2,B3,B4,B5,Bp,L1,L2,L3,L4,L5,T0,T1,T2,T3,T4,
T5,Tp,cent1,cent2,cent3);
ML=ML_funct(A0,A1,A2,A3,A4,A5,Ap,B0,B1,B2,B3,B4,B5,Bp,L1,L2,L3,L4,L5,T0,T1,T2,T3,T4,


T5,Tp,cent1,cent2,cent3);
MV=MV_funct(A0,A1,A2,A3,A4,A5,Ap,B0,B1,B2,B3,B4,B5,Bp,DA0,DA1,DA2,DA3,DA4,DA5,DAp,
DB0,DB1,DB2,DB3,DB4,DB5,DBp,DT0,DT1,DT2,DT3,DT4,DT5,DTp,L1,L2,L3,L4,L5,T0,T1,T2,T3,
T4,T5,Tp,cent1,cent2,cent3);
Ea=Ea_funct(A0,Ap,B0,B1,B2,B3,B4,B5,Bp,L1,L2,L3,L4,L5,T0,T1,T2,T3,T4,T5,Tp,cent1,
cent2,cent3);
origin=origin_funct(Ap,B1,B2,B3,B4,B5,Bp,L1,L2,L3,L4,L5,T1,T2,T3,T4,T5,Tp,cent1,
cent2,cent3);

A=Stateo(37:end);

[MF,dA]=musclemodel(U,P,A,ML,MV,Lo,CSA);
[MFR,dA,q,b,Gp,Gd]=musclemodelR(U,P,A,ML,MV,Lo,CSA,verification);

% for the verification script
    if strcmp('verification',verification)
        MFR=zeros(size(MFR));
    end

MF=MF+MFR;

EFo=0; %external force incorporated in spineaccP
Q=generalizedForce(Ma,real(MF),Ea,EFo,zeros(7,3),Gl,Gr,origin);

% solve for the angular acceleration
qdd=Mm\(Q);
end
```

spineacc.m file utilized for equilibrium constraint

```
% Valerie Jardon
% Department of Bioengineering, University of Kansas
% spineacc.m
% Last Edited: 4/30/2022
% This file (spineacc.m) is based on the AccelerationSP.m file developed by Timothy C. Franklin, Virginia Polytechnic Institute and State University.
% The AccelerationSP.m file was obtained through means of personal communication (Michael Madigan, personal communication, July 26, 2008).

% Reference:
% Franklin, T. C. (2006). Linear System Analyses of the Role of Reflex Gain and Delay in a Dynamic Human Spine Model (Master's thesis, Thesis / Dissertation ETD). Blacksburg: Virginia Tech. Retrieved June 19, 2020, from https://vtechworks.lib.vt.edu/handle/10919/33605.
%
-------------------------------------------------------------------------------------------
----------------------------------------

function [qddA,qddP,qddR]=spineacc(i,Stateo,Pango,U,P,EFo,Vars,Lo,CSA,Lord_Angs,IVD, Gl_funct,Gr_funct,Mm_funct,Cm_funct,Gm_funct,Ma_funct,ML_funct,MV_funct,Ea_funct, Rm_funct,dRm_funct,origin_funct,dorigin_funct,ChSpace,verification)

[A0,A1,A2,A3,A4,A5,Ap,B0,B1,B2,B3,B4,B5,Bp,DA0,DA1,DA2,DA3,DA4,DA5,DAp,DB0,DB1,DB2, DB3,DB4,DB5,DBp,DT0,DT1,DT2,DT3,DT4,DT5,DTp,T0,T1,T2,T3,T4,T5,Tp, ...
    A0dot,A1dot,A2dot,A3dot,A4dot,A5dot,T0dot,T1dot,T2dot,T3dot,T4dot,T5dot,B0dot, B1dot,B2dot,B3dot,B4dot,B5dot,Apdot,Bpdot,Tpdot]=vect2ang(Stateo,Pango);

T5ddot=0; B5ddot=0; A5ddot=0;
T4ddot=0; B4ddot=0; A4ddot=0;
T3ddot=0; B3ddot=0; A3ddot=0;
T2ddot=0; B2ddot=0; A2ddot=0;
T1ddot=0; B1ddot=0; A1ddot=0;
T0ddot=0; B0ddot=0; A0ddot=0;
Tpddot=0; Bpddot=0; Apddot=0;

L0=Vars(1); L1=Vars(2); L2=Vars(3); L3=Vars(4); L4=Vars(5); L5=Vars(6);
la=Vars(7); lb=Vars(8); g=Vars(10);
cent1=Vars(17); cent2=Vars(18); cent3=Vars(9);
m0=Vars(11); m1=Vars(12); m2=Vars(13); m3=Vars(14); m4=Vars(15); m5=Vars(16);

% Recalculate due to the disturbance
if i==1
    Gl=Gl_funct(B1,B2,B3,B4,B5,L1,L2,L3,L4,L5,T1,T2,T3,T4,T5);
    Gr=Gr_funct(B0,B1,B2,B3,B4,B5,T0,T1,T2,T3,T4,T5);
    Mm=Mm_funct(A0,A1,A2,A3,A4,A5,B0,B1,B2,B3,B4,B5,L0,L1,L2,L3,L4,L5,T0,T1,T2,T3,T4, T5,la,lb,m0,m1,m2,m3,m4,m5);
    Cm=Cm_funct(A0,A1,A2,A3,A4,A5,A0dot,A1dot,A2dot,A3dot,A4dot,A5dot,A0ddot,A1ddot, A2ddot,A3ddot,A4ddot,A5ddot,Ap,Apdot,Apddot,B0,B1,B2,B3,B4,B5,B0dot,B1dot,B2dot, B3dot,B4dot,B5dot,B0ddot,B1ddot,B2ddot,B3ddot,B4ddot,B5ddot,Bp,Bpdot,Bpddot,L0,L1,L2,


L3,L4,L5,T0,T1,T2,T3,T4,T5,T0dot,T1dot,T2dot,T3dot,T4dot,T5dot,T0ddot,T1ddot,T2ddot, T3ddot,T4ddot,T5ddot,Tp,Tpdot,Tpddot,cent1,cent2,cent3,g,la,lb,m0,m1,m2,m3,m4,m5);
    Gm=Gm_funct(B0,B1,B2,B3,B4,B5,L0,L1,L2,L3,L4,L5,T0,T1,T2,T3,T4,T5,g,m0,m1,m2,m3, m4,m5);
    Ma=Ma_funct(A0,A1,A2,A3,A4,A5,Ap,B0,B1,B2,B3,B4,B5,Bp,L1,L2,L3,L4,L5,T0,T1,T2,T3, T4,T5,Tp,cent1,cent2,cent3);
    ML=ML_funct(A0,A1,A2,A3,A4,A5,Ap,B0,B1,B2,B3,B4,B5,Bp,L1,L2,L3,L4,L5,T0,T1,T2,T3, T4,T5,Tp,cent1,cent2,cent3);
```

```
    MV=MV_funct(A0,A1,A2,A3,A4,A5,Ap,B0,B1,B2,B3,B4,B5,Bp,DA0,DA1,DA2,DA3,DA4,DA5,
DAp,DB0,DB1,DB2,DB3,DB4,DB5,DBp,DT0,DT1,DT2,DT3,DT4,DT5,DTp,L1,L2,L3,L4,L5,T0,T1,T2,
T3,T4,T5,Tp,cent1,cent2,cent3);
    Ea=Ea_funct(A0,Ap,B0,B1,B2,B3,B4,B5,Bp,L1,L2,L3,L4,L5,T0,T1,T2,T3,T4,T5,Tp,cent1,
cent2,cent3);
    rotation_matrices=Rm_funct(A0,A1,A2,A3,A4,A5,Ap,B0,B1,B2,B3,B4,B5,Bp,T0,T1,T2,T3,
T4,T5,Tp);
    d_rotation_matrices=dRm_funct(A0,A1,A2,A3,A4,A5,Ap,B0,B1,B2,B3,B4,B5,Bp,DA0,DA1,
DA2,DA3,DA4,DA5,DAp,DB0,DB1,DB2,DB3,DB4,DB5,DBp,DT0,DT1,DT2,DT3,DT4,DT5,DTp,T0,T1,T2,
T3,T4,T5,Tp);
    origin=origin_funct(Ap,B1,B2,B3,B4,B5,Bp,L1,L2,L3,L4,L5,T1,T2,T3,T4,T5,Tp,cent1,
cent2,cent3);
    dorigin=dorigin_funct(Ap,B1,B2,B3,B4,B5,Bp,DAp,DB1,DB2,DB3,DB4,DB5,DBp,DT1,DT2,
DT3,DT4,DT5,DTp,L1,L2,L3,L4,L5,T1,T2,T3,T4,T5,Tp,cent1,cent2,cent3);

    PRQ=IVD_calcs(rotation_matrices,d_rotation_matrices,Stateo,Pango,Lord_Angs,IVD);
    PRQ=PRQ(1:6,:);
        if strcmp('verification',verification)
            Cm=zeros(size(Cm)); %verification Cm
            PRQ=[1 2 3; 4 5 6; 7 8 9; 10 11 12; 1 2 3; 5 5 4; 1 2 3]; %verification
PRQ
            PRQ=PRQ(1:6,:);
        end
    save('optimization_varsA.
mat','rotation_matrices','d_rotation_matrices','origin','dorigin','Ea','Ma','ML','MV'
,'Mm','Gl','Gr','PRQ','Gm','Cm')
else
    load('optimization_varsA.mat')
end

A=Stateo(37:end);

[MFa,dAa]=musclemodel(U,P,A,ML,MV,Lo,CSA); %ACTIVE
MFp=zeros(size(U)); %PASSIVE no muscle force
[MFr,dAr,~,~,~,~]=musclemodelR(U,P,A,ML,MV,Lo,CSA,verification); %REFLEX

[QA]=generalizedForce(Ma,real(MFa),Ea,0,zeros(7,3),Gl,Gr,origin);
qddA=Mm\(QA);

[QP]=generalizedForce(Ma,MFp,Ea,EFo,PRQ,Gl,Gr,origin);
qddP=Mm\(QP-Cm'-Gm');



[QR]=generalizedForce(Ma,real(MFr),Ea,0,zeros(7,3),Gl,Gr,origin);
qddR=Mm\(QR);
end
```

C.13.3 *run_fmincon.mlx*: Constrained Optimization Solver *fmincon* (MATLAB)

Valerie Jardon

Department of Bioengineering, University of Kansas

run_fmincon.mlx

Last Edited: 4/30/2022

Based on the optimizationscript.m code developed by Timothy C. Franklin, Virginia Polytechnic Institute and State University.

Description: This function (run_fmincon.mlx) is used to run fmincon, allowing for the muscle activation and metabolic power to be determined, while ensuring that the system is stable and in equilibrium.

This section is based on the "Run Optimization routine" section of the optimizationscript.m file developed by Timothy C. Franklin, published in Appendix A of his thesis.

Reference:

Franklin, T. C. (2006). *Linear System Analyses of the Role of Reflex Gain and Delay in a Dynamic Human Spine Model* (Master's thesis, Thesis / Dissertation ETD). Blacksburg: Virginia Tech. Retrieved June 19, 2020, from https://vtechworks.lib.vt.edu/handle/10919/33605

```
function [exitflag_success,fval_success,Un_total_success,Un_total_initial_success]=run_fmincon(A,B,Aeq,beq,CSA,Lo,Stateo,Jp,Ja,M)

num_trials=20; %number of trials

%Initializing Variables
Un_total=zeros(90,num_trials); %matrix that will store the muscle activation for all 10 trials
fval_total=zeros(1,num_trials); %vector that will store the metabolic power for all 10 trials
Un_total_initial=zeros(90,num_trials); %matrix that will store the initial guess of muscle activation for all 10 trials
exitflag_total=zeros(1,num_trials); %vector that will store the fmincon exitflag for all 10 trials

% Optimization Procedure
for K=1:num_trials %Running for 20 trials
    Un=rand(size(M,1),1); %random initial seed for muscle activation
    Un_initial=Un;

    up_bound=ones(90,1); %muscle activation: fully activated muscle
    low_bound=zeros(90,1); %muscle activation: unactivated muscle

% Setting the options for fmincon
    options=optimoptions('fmincon','Algorithm','sqp-legacy','MaxFunctionEvaluations',1e5,'MaxIterations',400,'StepTolerance',1e-9,'Functi
    [Un,fval,exitflag]=fmincon(@costfun,Un,A,B,Aeq,beq,low_bound,up_bound,@nlcon,options,Stateo,Jp,Ja,Lo,CSA);
    fprintf('Trial %2d: %3.3f Watts\n',K,fval)

% Storing select inputs and outputs for this trial
    Un_total(:,K)=Un;
    fval_total(1,K)=fval;
    Un_total_initial(:,K)=Un_initial;
    exitflag_total(1,K)=exitflag;
end
```

Determine the Success of the Trials

If tolerances were not met or the solver converged to an infeasible point, then the trial was not successful. Exitflags from the solver indicate its potential success or failure. The successful trial information should be stored in vectors and matrices and used as output for this function.

In *optimize.m*, stability and equilibrium conditions will be evaluated and the optimal solution will be utilized.

```
% Store optimization output from all trials
% Initialize Variables
success_vector=zeros(1,num_trials);
exitflag_success=zeros(1,num_trials);
fval_success=Inf*ones(1,num_trials);
Un_total_initial_success=zeros(90,num_trials);
Un_total_success=zeros(90,num_trials);

for j=1:num_trials
    if exitflag_total(j)==1 || exitflag_total(j)==2 %indicating that tolerances have been met and local minima found
        exitflag_success(j)=exitflag_total(j);
        fval_success(j)=fval_total(j);
        Un_total_success(:,j)=Un_total(:,j);
        Un_total_initial_success(:,j)=Un_total_initial(:,j);
    end
end
end
```

C.13.4 *costfun.m*: Metabolic Power Function for Constrained Optimization Procedure

```
% Valerie Jardon
% Department of Bioengineering, University of Kansas
% costfun.m
% Last Edited: 4/30/2022
% Based on the costfun.m code developed by Timothy C. Franklin, Virginia Polytechnic Institute and State University.

% Description: This function (costfun.m) is used with fmincon to calculate the metabolic power. This is the parameter being minimized during the constrained optimization procedure.
% This function is based on the costfun.m file developed by Timothy C. Franklin, published in Appendix A of his thesis.

% References:
% Franklin, T. C. (2006). Linear System Analyses of the Role of Reflex Gain and Delay in a Dynamic Human Spine Model (Master's thesis, Virginia Polytechnic Institute and State University) (pp. 87). Virginia Tech. Retrieved June 19, 2020,from https://vtechworks.lib.vt.edu/handle/10919/33605?show=full.
% Anderson, F. C., III. (2000). A dynamic optimization solution for a complete cycle of normal gait (Doctoral dissertation, University of Texas at Austin, 1999). Ann Arbor, MI: Bell & Howell Information and Learning. Retrieved September 7, 2021, from https://www2.lib.ku.edu/login?url=https://www.proquest.com/dissertations-theses/dynamic-optimization-solution-complete-cycle/docview/304528709/se-2?accountid=14556.
%
-------------------------------------------------------------------------------------------------
-----------------------------------------------

function Power=costfun(Un,S,Jp,Ja,Lo,CSA)

rho=1000; % units: kg/m^3
mass=rho*(CSA/100/100).*Lo; %CSA converted to m^2; m=density*volume

aslow=sin(pi/2*Un); %excitation level (Anderson 1999)
afast=1-cos(pi/2*Un); %excitation level (Anderson 1999)

%assuming 50% slow twitch fibers and 50% fast twitch fibers
percent_slowtwitch=0.5;
percent_fasttwitch=0.5;

%Total Maintenance Heat Rate (Anderson 1999 page 120)
Power=sum(mass.*(74*percent_slowtwitch*aslow+111*percent_fasttwitch*afast));     %
W=J/s
```

C.13.5 *nlcon.m*: Nonlinear Constraints for Constrained Optimization Procedure

Author: Timothy C. Franklin

nlcon.m

nlcon.m file provided in the appendix of his thesis (Franklin, 2006, p. 87-88)

Source: Franklin, T. C. (2006). *Linear System Analyses of the Role of Reflex Gain and Delay in a Dynamic Human Spine Model* (Master's thesis, Thesis / Dissertation ETD). Blacksburg: Virginia Tech. Retrieved June 19, 2020, from https://vtechworks.lib.vt.edu/handle/10919/33605

Edits to original function made by Valerie Jardon

Department of Bioengineering, University of Kansas

Last Edited: 4/30/2022

Nonlinear Constraints for Constrained Optimization

```
function [c,ceq]=nlcon(Un,S,Jp,Ja,Lo,CSA)
%This function involves the nonlinear constraints utilized in the
%constrained optimization procedure
% The eigenvalues of the Jacobian are evaluated to determine stability

global PrevEig JPrevEig cequal optiter
optiter=optiter+1;

% Nonlinear Constraint for fmincon #1
ceq=0;

% Nonlinear Constraint for fmincon #2

% determine the complete Jacobian using the muscle activation
J=Jp;
    for i=1:36
        J(19:36,i)=J(19:36,i)+(Un'*Ja(:,19:36,i))';
    end

    ap=eig(J); %eig returns a column vector containing the eigenvalues of square matrix J
    b=zeros(36,1);
    p=zeros(36,36);

    for j=1:36
        dist=JPrevEig-ap(j);
        p(j,:)=sqrt(real(dist).^2+imag(dist).^2)';
    end

    for j=1:36
        [q,indc]=min(min(p));
        [q,indr]=min(p(:,indc));

        b(indc)=ap(indr);
        p(:,indc)=1e12;
        p(indr,:)=1e12;
    end

JPrevEig=b;
c=real(b)+1e-4; %Allow 1e-4 slop
end
```

C.14 *musclemodel.mlx*: Muscle Model

Valerie Jardon

Department of Bioengineering, University of Kansas

musclemodel.mlx

Last Edited: 4/30/2022

This file (musclemodel.mlx) is based on the muscmodel.m code developed by Timothy C. Franklin, Virginia Polytechnic Institute and State University. The muscmodel.m code was obtained through means of personal communication (Michael Madigan, personal communication, July 26, 2008).

Muscle Model

Description: This function (musclemodel.mlx) is based on the Hill-type muscle model discussed in Franklin's thesis (Franklin, 2006, p. 12-13). This Hill-type muscle model includes a force generator, a spring and a damper, with all elements in parallel.

Reflexes should not be included in this function, so the proportional and differential reflex gains should be zero. See function musclemodelR.mlx for reflexes.

References:

Bergmark A. (1989). Stability of the lumbar spine. A study in mechanical engineering. *Acta orthopaedica Scandinavica. Supplementum*, *230*, 1–54. https://doi.org/10.3109/17453678909154177

Franklin, T. C. (2006). *Linear System Analyses of the Role of Reflex Gain and Delay in a Dynamic Human Spine Model* (Master's thesis, Thesis / Dissertation ETD). Blacksburg: Virginia Tech. Retrieved June 19, 2020, from https://vtechworks.lib.vt.edu/handle/10919/33605

Gardner-Morse, M., Stokes, I. A., & Laible, J. P. (1995). Role of muscles in lumbar spine stability in maximum extension efforts. *Journal of orthopaedic research : official publication of the Orthopaedic Research Society*, *13*(5), 802–808. https://doi.org/10.1002/jor.1100130521

```
function [fm,dA]=musclemodel(U,P,A,ML,MV,Lo,CSA)

fo=46*CSA; %units: N; maximum muscle force= maximum muscle stress * CSA (Gardner-Morse et al., 1995, pp. 803)
q=20; % Bergmark dimensionless stiffness gain (Bergmark, 1989, pp. 26-27)
b=2; %dimensionless damping gain

dX=ML-Lo;

% Steady-State Muscle Activation
U=U;

% Muscle Model Equation
fm=-fo.*U.*(q*(dX./Lo)+b*(MV./Lo)+1);
dA=zeros(size(A));
end
```

C.15 *musclemodelR.mlx*: Reflex Model

Valerie Jardon

Department of Bioengineering, University of Kansas

musclemodelR.mlx

Last Edited: 4/30/2022

This file (musclemodelR.mlx) is based on the muscmodel.m code developed by Timothy C. Franklin, Virginia Polytechnic Institute and State University. The muscmodel.m code was obtained through means of personal communication (Michael Madigan, personal communication, July 26, 2008).

Muscle Model

Description: This function (musclemodelR.mlx) incorporates the reflex activation (Franklin, 2006, p. 13). The muscle model is based on the Hill-type muscle model discussed in Franklin's thesis (Franklin, 2006, p. 12-13). This Hill-type muscle model includes a force generator, a spring and a damper, with all elements in parallel.

References:

Bergmark A. (1989). Stability of the lumbar spine. A study in mechanical engineering. *Acta orthopaedica Scandinavica. Supplementum*, *230*, 1–54. https://doi.org/10.3109/17453678909154177

Franklin, T. C. (2006). *Linear System Analyses of the Role of Reflex Gain and Delay in a Dynamic Human Spine Model* (Master's thesis, Thesis / Dissertation ETD). Blacksburg: Virginia Tech. Retrieved June 19, 2020, from https://vtechworks.lib.vt.edu/handle/10919/33605.

Gardner-Morse, M., Stokes, I. A., & Laible, J. P. (1995). Role of muscles in lumbar spine stability in maximum extension efforts. *Journal of orthopaedic research : official publication of the Orthopaedic Research Society*, *13*(5), 802–808. https://doi.org/10.1002/jor.1100130521

```
function [fm,dA,q,b,Gp,Gd]=musclemodelR(U,P,A,ML,MV,Lo,CSA,verification)

fo=46*CSA; %units: N; maximum muscle force= maximum muscle stress * CSA (Gardner-Morse et al., 1995, p. 803)
q=20; % Bergmark dimensionless stiffness gain (Bergmark, 1989, p. 26-27)
b=2; %dimensionless damping gain

dX=ML-Lo;
Gp=10; % proportional reflex gain
Gd=0; % differential reflex gain

% for verification procedure
if strcmp('verification',verification)
    Gp=0;
    Gd=0;
end

% Reflex Activation
U=U.*(Gp*(dX./Lo)+Gd*(MV./Lo));

% Muscle Model Equation
fm=-fo.*U.*(q*(dX./Lo)+b*(MV./Lo)+1);
dA=zeros(size(A));
end
```

C.16 *IVD_calcs.mlx*: Intervertebral Disc Moments

Valerie Jardon

University of Kansas

IVD_calcs.mlx

Last Edited: 4/30/2022

This file (IVD_calcs.mlx) is based on the intervertc.c code developed by Timothy C. Franklin, Virginia Polytechnic Institute and State University. The intervertc.c code was obtained through means of personal communication (Michael Madigan, personal communication, July 26, 2008).

Reference:

Franklin, T. C. (2006). *Linear System Analyses of the Role of Reflex Gain and Delay in a Dynamic Human Spine Model* (Master's thesis, Thesis / Dissertation ETD). Blacksburg: Virginia Tech. Retrieved June 19, 2020, from https://vtechworks.lib.vt.edu/handle/10919/33605

Intervertebral Disc Moments

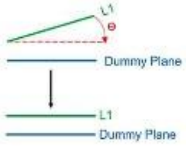
Description of Code:

The purpose of this code is to calculate the intervertebral disc moments.

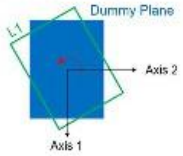
In these calculations, two neighboring planes will be used where one plane will remain stationary and the other plane will be manipulated through rotation.

In order to determine the planar rotation angle, the difference of the lordosis angles must first be rotated out so it will not influence our measures. In this case, the difference between the lordosis angles should be determined and one of the planes should be rotated so that this difference in lordosis angle is no longer present between the planes.

Any remaining rotation about axis 1 is due to the planar rotation. The planar rotation angle can be calculated. Then the planar angle is "rotated out" for the same plane as done previously for the lordosis angle. The resulting plane should be parallel to the plane that has not been manipulated. Conceptually:



The twist rotation occurs around Axis 3. The twist angle can be determined using the parallel planes.



Lastly, the intervertbral disc moment can be calculated. Equal and opposite moments implemented.

The verification task should be performed following analysis to ensure that the planes are in fact parallel and aligned.

```
function [PRQ_total]=IVD_calcs(rotation_matrices,d_rotation_matrices,Stateo,Pango,Lord_Angs,IVD)

% Initializing Variables
PRQ=zeros(7,3);
PRQ_p=zeros(7,3);
PRQ_t=zeros(7,3);
PRQ1=zeros(6,3);
PRQ2=zeros(6,3);
PRQ3=zeros(6,3);
PRQ4=zeros(6,3);
PRQ5=zeros(6,3);
PRQ6=zeros(6,3);
PRQ7=zeros(6,3);
PRQ8=zeros(6,3);

rot_matrix=rotation_matrices;
d_rot_matrix=d_rotation_matrices;

for i=1:6

%% Planar Calculation
    % Load Rotation Matrices for Each Body
    body2_Rm=rot_matrix(i*3-2:i*3,:); %body 2 is the upper body (ex. thorax)
    body1_Rm=rot_matrix((i+1)*3-2:(i+1)*3,:); %body 1 is the lower body (ex. L1)
    body2_dRm=d_rot_matrix(i*3-2:i*3,:); %body 2 is the upper body (ex. thorax)
    body1_Rm_unadjusted=body1_Rm; % this will be used to compare if the planes are parallel
```

```
  % Determine the difference between the lordosis angles of the two bodies
   ang=Lord_Angs(i+1)-Lord_Angs(i);

   % This rotation matrix will be used to rotate out the lordosis angle
   syms q1
   R1=[1 0 0; 0 cos(q1) -sin(q1); 0 sin(q1) cos(q1)]; %Rotation axis 1
   rotate_out=subs(R1,q1,ang); rotate_out=double(rotate_out);

   body2_Rm=body2_Rm*rotate_out; % nRd
   body2_dRm=body2_dRm*rotate_out; % nRd

   % Vectors along axis 3 of each body needed to determine planar angle
   TV1=body2_Rm(:,3); %axis d3
   TV2=body1_Rm(:,3);

   planar_ang=acos(dot(TV1,TV2)); % if the dot product of V1 & V2 equalled 1, then the angle would be zero and we would know that axis 3

   if planar_ang > 0.000000000001 %Franklin's tolerance criteria

   % Determine the perpendicular vector along axis 1 using the cross product -
   % this should be a unit vector
   TV=cross(TV1,TV2);
   TV_mag=sqrt(sum(TV.^2));
   TV=TV/TV_mag;
   RV=TV*planar_ang; %scaling the vector

   % Calculating the moments - equal and opposite (completed later on)
   PRQ1(i,:)=IVD(1)*RV;
   PRQ2(i,:)=-IVD(1)*RV;

   % This rotation matrix will be used to rotate out the planar angle using
   % the lower body - planes should now be parallel
   RV_mag=sqrt(sum(RV.^2));
   Rvec=RV/RV_mag; Rvec=Rvec';

   body2_Rm=transpose(body2_Rm); %so that dRn
   body1_Rm=transpose(body1_Rm); %so that dRn
   body2_dRm=transpose(body2_dRm); %so that dRn

   body2_Rm=rodrigues_rotation(body2_Rm,Rvec,planar_ang); % rodrigues rotation formula
   body2_dRm=rodrigues_rotation(body2_dRm,Rvec,planar_ang);
 else
   body2_Rm=transpose(body2_Rm); %so that dRn
   body1_Rm=transpose(body1_Rm); %so that dRn
   body2_dRm=transpose(body2_dRm); %so that dRn
 end

%% Twist Calculation
   clear TV1 TV2 TV TV_mag RV

   % Vectors along axis 1 of each body needed to determine twist angle
   TV1=body2_Rm(1,:); %axis d1
   TV2=body1_Rm(1,:);

   twist_ang=acos(dot(TV1,TV2)); % if the dot product of V1 & V2 equalled 1, then the angle would be zero and we would know that axis 1 w

 if twist_ang > .000000000001 %Franklin's criteria

   % Determine the perpendicular vector along axis 3 using the cross product -
   % this should be a unit vector
   TV=cross(TV1,TV2);
   TV_mag=sqrt(sum(TV.^2));
   TV=TV/TV_mag;
   RV=TV*twist_ang; %scaling the vector

   % Calculating the moments - equal and opposite (completed later on)
   PRQ3(i,:)=IVD(3)*RV;
   PRQ4(i,:)=-IVD(3)*RV;

   % This rotation matrix will be used to rotate out the twist angle
   RV_mag=sqrt(sum(RV.^2));
   Rvec=RV/RV_mag;

   body2_Rm=rodrigues_rotation(body2_Rm,Rvec,twist_ang);
   body2_dRm=rodrigues_rotation(body2_dRm,Rvec,twist_ang);

   body2_Rm=transpose(body2_Rm); %so that nRd
   body1_Rm=transpose(body1_Rm); %so that nRd
   body2_dRm=transpose(body2_dRm); %so that nRd
 else
   body2_Rm=transpose(body2_Rm); %so that nRd
```

```
    body1_Rm=transpose(body1_Rm); %so that nRd
    body2_dRm=transpose(body2_dRm); %so that nRd
 end

% Verification Procedure
%Check to see if planes are aligned and parallel
planes_check=body1_Rm_unadjusted-body2_Rm;
num_planes_check=numel(planes_check);
failure_vect=zeros(num_planes_check,1);

for j=1:num_planes_check
    if abs(planes_check(j)) > 1E-7
        failure_vect(j)=1;
    else
        failure_vect(j)=0;
    end
end

if sum(failure_vect)~=0
    error('Cannot continue. Planes are not successfully aligned and/or parallel.')
% else
%     fprintf('SUCCESS. Iteration %2d. This plane is aligned and parallel.\n',i)
end

% Damping - recall that body2 is the one being adjusted and body 1 is
% stationary
    body2_dRm=d_rot_matrix(i*3-2:i*3,:); %body 2 is the upper body (ex. thorax)
    body1_dRm=d_rot_matrix((i+1)*3-2:(i+1)*3,:); %body 1 is the lower body (ex, L1)
    TV=body2_dRm(:,3)-body1_dRm(:,3); %relative velocity of body 2
    dPT=sqrt(sum(TV.^2)); %calculate speed

   if dPT> .000000000001
       TV1=body2_Rm(:,3);
       TV2=cross(TV1,TV); %perpendicular vector
       TV2_mag=sqrt(sum(TV2.^2)); % magnitude
       RV=TV2*dPT/TV2_mag; %unit vector

    % Calculating the moments - equal and opposite (completed later on)
        PRQ5(i,:)=-IVD(2)*RV;
        PRQ6(i,:)=IVD(2)*RV;
   end

   %twist
   dTT=sqrt((body2_dRm(2,1)-body1_dRm(2,1))*(body2_dRm(2,1)-body1_dRm(2,1)));
        if dTT> .000000000001
            Vthree=[0 0 1];
            RV=(body2_dRm(2,1)-body1_dRm(2,1)).*Vthree;
            PRQ7(i,:)=-IVD(4)*RV;
            PRQ8(i,:)=IVD(4)*RV;
        end
end

% Finishing Moment Calculations - equal and opposite
for i=1:7
    if i==1
        PRQ_p(i,:)=PRQ1(i,:)+PRQ5(i,:);
        PRQ_t(i,:)=PRQ3(i,:)+PRQ7(i,:);
    elseif i==7
        PRQ_p(i,:)=PRQ2(i-1,:)+PRQ6(i-1,:);
        PRQ_t(i,:)=PRQ4(i-1,:)+PRQ8(i-1,:);
    else
        PRQ_p(i,:)=PRQ1(i,:)+PRQ2(i-1,:)+PRQ5(i,:)+PRQ6(i-1,:);
        PRQ_t(i,:)=PRQ3(i,:)+PRQ4(i-1,:)+PRQ7(i,:)+PRQ8(i-1,:);
    end
end
PRQ_total=PRQ_p+PRQ_t;
end
```

```
function [rotated_vectors]=rodrigues_rotation(vectors,rotation_axis,theta)
%theta needs to be in radians

%axis of rotation must be a unit vector
rotation_axis_mag=sqrt(sum(rotation_axis.^2));
rotation_axis_unit=rotation_axis/rotation_axis_mag;

rotated_vectors=zeros(3);
    for i=1:3
        rotated_vectors(i,:)=(dot(vectors(i,:),rotation_axis_unit)*rotation_axis_unit)*(1-cos(theta))+...
        cross(rotation_axis_unit,vectors(i,:))*sin(theta)+vectors(i,:)*cos(theta);
    end
end
```

C.17 Time Delay Analysis

The Time Delay Analysis files (*tdinvest.m*, *tdfmax.m* and *tdfind.m*) from the appendix of Franklin's thesis should be utilized to perform the Time Delay Analysis (pages 94-97) [24].

C.18 Nonlinear Verification

C.18.1 Run Nonlinear Simulation

From spine_main.m file:

```
%% Simulation
% In this section, the nonlinear simulation is completed to compare the
% linear analysis results from the optimization procedure.

disp('-----Simulation Started-----')

StateDist=Stateo;
StateDist(16)=StateDist(16)+deg2rad(0.01); % disturbance applied to thoracic/cervical ↙
spine rigid body

Tfinal=5;
Tode=[0,Tfinal]; %defining the bounds of integration

%the delay margin (Td) defines the time delay in which the system becomes
%unstable, therefore the simulation should include the time delay prior to
%instability
Lags=Td*0.99;

%Delay must be greater than 0.06 seconds to be physiologically
%representative of reflexes and their contribution to stability
if Lags < 0.06
    error('Simulation cannot be performed. Required lag is not physiologically ↙
possible.')
end

%-----Set Simulation Options
 options=ddeset('OutputFcn',@plotscheme,'Events',@edcheck,'RelTol',1e-4,'AbsTol',1e-↙
9,'Stats','on');

%-----Simulation
tic

sol=dde23(@spinedyn,Lags,StateDist,Tode,options,Un_opt,Pn,Pango,Vars,Gl_funct, ↙
Gr_funct,Mm_funct,Cm_funct,Gm_funct,Ma_funct,ML_funct,MV_funct,Ea_funct,Rm_funct, ↙
dRm_funct,origin_funct,dorigin_funct,Lord_Angs,IVD,Lo,CSA,EFo,ChSpace);
Time=sol.x;
State=sol.y';
Stated=State-ones(length(Time),1)*Stateo';

%Display Magnitude of Lag Simulated & Simulation Run Time
disp(['Lag Simulated: ' num2str(Lags(1))]);

elapsedtime=toc;


elapsedtime_min=fix(elapsedtime/60);
fprintf('The simulation took %3d minutes and %3d seconds to run.',elapsedtime_min,↙
(elapsedtime-(60*elapsedtime_min)))
```

C.18.2 *Spinedyn.mlx*: Function Handle for Delay Differential Equation Solver

Valerie Jardon

Department of Bioengineering, University of Kansas

spinedyn.mlx

Last Edited: 4/30/2022

This file (spinedyn.mlx) is based on the spinedyn.mlx developed by Timothy C. Franklin, published in Appendix A of his thesis (Franklin, 2006, p. 88-89)

Reference:

Franklin, T. C. (2006). *Linear System Analyses of the Role of Reflex Gain and Delay in a Dynamic Human Spine Model* (Master's thesis, Thesis / Dissertation ETD). Blacksburg: Virginia Tech. Retrieved June 19, 2020, from https://vtechworks.lib.vt.edu/handle/10919/33605

```
function [dS]=spinedyn(t,Stateo,Z,U,P,Pango,Vars,Gl_funct,Gr_funct,Mm_funct,Cm_funct,Gm_funct,Ma_funct,ML_funct,MV_funct,Ea_funct,Rm_funct
% Initialize Variables
[A0,A1,A2,A3,A4,A5,Ap,B0,B1,B2,B3,B4,B5,Bp,DA0,DA1,DA2,DA3,DA4,DA5,DAp,DB0,DB1,DB2,DB3,DB4,DB5,DBp,DT0,DT1,DT2,DT3,DT4,DT5,DTp,T0,T1,T2,T3
    A0dot,A1dot,A2dot,A3dot,A4dot,A5dot,T0dot,T1dot,T2dot,T3dot,T4dot,T5dot,B0dot,B1dot,B2dot,B3dot,B4dot,B5dot,Apdot,Bpdot,Tpdot]=vect2ar

T5ddot=0; B5ddot=0; A5ddot=0;
T4ddot=0; B4ddot=0; A4ddot=0;
T3ddot=0; B3ddot=0; A3ddot=0;
T2ddot=0; B2ddot=0; A2ddot=0;
T1ddot=0; B1ddot=0; A1ddot=0;
T0ddot=0; B0ddot=0; A0ddot=0;
Tpddot=0; Bpddot=0; Apddot=0;

L0=Vars(1); L1=Vars(2); L2=Vars(3); L3=Vars(4); L4=Vars(5); L5=Vars(6);
la=Vars(7); lb=Vars(8); g=Vars(10);
cent1=Vars(17); cent2=Vars(18); cent3=Vars(9);
m0=Vars(11); m1=Vars(12); m2=Vars(13); m3=Vars(14); m4=Vars(15); m5=Vars(16);

Gl=Gl_funct(B1,B2,B3,B4,B5,L1,L2,L3,L4,L5,T1,T2,T3,T4,T5);
Gr=Gr_funct(B0,B1,B2,B3,B4,B5,T0,T1,T2,T3,T4,T5);
Mm=Mm_funct(A0,A1,A2,A3,A4,A5,B0,B1,B2,B3,B4,B5,L0,L1,L2,L3,L4,L5,T0,T1,T2,T3,T4,T5,la,lb,m0,m1,m2,m3,m4,m5);
Cm=Cm_funct(A0,A1,A2,A3,A4,A5,A0dot,A1dot,A2dot,A3dot,A4dot,A5dot,A0ddot,A1ddot,A2ddot,A3ddot,A4ddot,A5ddot,Ap,Apdot,Apddot,B0,B1,B2,B3,B4
Gm=Gm_funct(B0,B1,B2,B3,B4,B5,L0,L1,L2,L3,L4,L5,T0,T1,T2,T3,T4,T5,g,m0,m1,m2,m3,m4,m5);
Ma=Ma_funct(A0,A1,A2,A3,A4,A5,Ap,B0,B1,B2,B3,B4,B5,Bp,L1,L2,L3,L4,L5,T0,T1,T2,T3,T4,T5,Tp,cent1,cent2,cent3);
ML=ML_funct(A0,A1,A2,A3,A4,A5,Ap,B0,B1,B2,B3,B4,B5,Bp,L1,L2,L3,L4,L5,T0,T1,T2,T3,T4,T5,Tp,cent1,cent2,cent3);
MV=MV_funct(A0,A1,A2,A3,A4,A5,Ap,B0,B1,B2,B3,B4,B5,Bp,DA0,DA1,DA2,DA3,DA4,DA5,DAp,DB0,DB1,DB2,DB3,DB4,DB5,DBp,DT0,DT1,DT2,DT3,DT4,DT5,DTp,
Ea=Ea_funct(A0,Ap,B0,B1,B2,B3,B4,B5,Bp,L1,L2,L3,L4,L5,T0,T1,T2,T3,T4,T5,Tp,cent1,cent2,cent3);
rotation_matrices=Rm_funct(A0,A1,A2,A3,A4,A5,Ap,B0,B1,B2,B3,B4,B5,Bp,T0,T1,T2,T3,T4,T5,Tp);
d_rotation_matrices=dRm_funct(A0,A1,A2,A3,A4,A5,Ap,B0,B1,B2,B3,B4,B5,Bp,DA0,DA1,DA2,DA3,DA4,DA5,DAp,DB0,DB1,DB2,DB3,DB4,DB5,DBp,DT0,DT1,DT
origin=origin_funct(Ap,B1,B2,B3,B4,B5,Bp,L1,L2,L3,L4,L5,T1,T2,T3,T4,T5,Tp,cent1,cent2,cent3);
dorigin=dorigin_funct(Ap,B1,B2,B3,B4,B5,Bp,DAp,DB1,DB2,DB3,DB4,DB5,DBp,DT1,DT2,DT3,DT4,DT5,DTp,L1,L2,L3,L4,L5,T1,T2,T3,T4,T5,Tp,cent1,cent

PRQ=IVD_calcs(rotation_matrices,d_rotation_matrices,Stateo,Pango,Lord_Angs,IVD);
PRQ=PRQ(1:6,:);
A=Stateo(37:end);

%angles are now for Z
[A0,A1,A2,A3,A4,A5,Ap,B0,B1,B2,B3,B4,B5,Bp,DA0,DA1,DA2,DA3,DA4,DA5,DAp,DB0,DB1,DB2,DB3,DB4,DB5,DBp,DT0,DT1,DT2,DT3,DT4,DT5,DTp,T0,T1,T2,T3
    A0dot,A1dot,A2dot,A3dot,A4dot,A5dot,T0dot,T1dot,T2dot,T3dot,T4dot,T5dot,B0dot,B1dot,B2dot,B3dot,B4dot,B5dot,Apdot,Bpdot,Tpdot]=vect2ar

% for reflex calculations
MLR=ML_funct(A0,A1,A2,A3,A4,A5,Ap,B0,B1,B2,B3,B4,B5,Bp,L1,L2,L3,L4,L5,T0,T1,T2,T3,T4,T5,Tp,cent1,cent2,cent3);
MVR=MV_funct(A0,A1,A2,A3,A4,A5,Ap,B0,B1,B2,B3,B4,B5,Bp,DA0,DA1,DA2,DA3,DA4,DA5,DAp,DB0,DB1,DB2,DB3,DB4,DB5,DBp,DT0,DT1,DT2,DT3,DT4,DT5,DTp

[MF,dA]=musclemodel(U,P,A,ML,MV,Lo,CSA); %ACTIVE
[MFR,dAR,~,~,~,~]=musclemodelR(U,P,A,MLR,MVR,Lo,CSA,[]); %REFLEX
MF=MF+MFR; %total muscle force

[Q]=generalizedForce(Ma,real(MF),Ea,EFo,PRQ,Gl,Gr,origin);

qdd=Mm\(Q-Cm'-Gm');

 dS=zeros(length(Stateo),1);
 dS(1:18)=Stateo(19:36);
 dS(19:36)=qdd;
 dS(37:end)=dA;
 end
```

C.18.3 Determine the Distance of the Normalized State-Space from Equilibrium

From spine_main.m file:

```
%% Calculate the distance from equilibrium using the normalized state-space
% Determine if the system is approaching equilibrium

position=Stated(:,1:18);
mean_position=mean(position,'all'); %average of all positions
position_Dss=position./mean_position;

velocity=Stated(:,19:36);
mean_velocity=mean(velocity,'all');
velocity_Dss=velocity./mean_velocity;

Dss=zeros(size(Stated,1),18);
    for i=1:18
        Dss(:,i)=(position_Dss(:,i)).^2+(velocity_Dss(:,i)).^2;
    end

        Dss=sqrt(sum(Dss,2)); %sum of each row - each row is a time step
        figure(4); plot(Time,Dss); grid on; xlabel('Time (seconds)'); ylabel('Dss');
title('Dss vs. Time')

% ---------------------- End of spine_main.m script
```

Appendix D: **Additional MATLAB Files**

D.1 *replace_sym.mlx*: Replace symbolic variables with numeric

Valerie Jardon

Department of Bioengineering, University of Kansas

replace_sym.mlx

Last Edited: 4/30/2022

Description: The function "replace_sym" is used to replace the symbolic variables that are within our variables in the Workspace with numeric values. Once all symbolic variables have been replaced with these numeric values, then the output can be converted to double data type.

Replaces model parameters variables and angles.

```
function [output]=replace_sym(input,Stateo,Pango,Vars,ChSpace)
syms la lb cent1 cent2 cent3 g m0 m1 m2 m3 m4 m5 L0 L1 L2 L3 L4 L5
syms DTp DBp DAp DT5 DB5 DA5 DT4 DB4 DA4 DT3 DB3 DA3 DT2 DB2 DA2 DT1 DB1 DA1 DT0 DB0 DA0
syms T5 T5dot B5 B5dot A5 A5dot T4 T4dot B4 B4dot A4 A4dot T3 T3dot B3 B3dot A3 A3dot T2 T2dot B2 B2dot A2 A2dot T1 T1dot B1 B1dot A1 A1dc
syms L1_ChSpace L2_ChSpace L3_ChSpace L4_ChSpace L5_ChSpace L1_space L2_space L3_space L4_space L5_space
%replace the symbolic variables with numeric values
% output=subs(input,[Tp, T5, T4, T3, T2, T1, T0, Bp, B5, B4, B3, B2, B1, B0, Ap, A5, A4, A3, A2, A1, A0],...
%     [Pango(1),Stateo(1),Stateo(4),Stateo(7),Stateo(10),Stateo(13),Stateo(16),Pango(2),Stateo(2),Stateo(5),Stateo(8),Stateo(11),Stateo(14

output=subs(input,[Tp,Bp,Ap,T5,B5,A5,T4,B4,A4,T3,B3,A3,T2,B2,A2,T1,B1,A1,T0,B0,A0],...
    [Pango(1),Pango(2),Pango(3),Stateo(1),Stateo(2),Stateo(3),Stateo(4),Stateo(5),Stateo(6),Stateo(7),Stateo(8),Stateo(9),Stateo(10),State

output=subs(output,[DTp, DT5, DT4, DT3, DT2, DT1, DT0, DBp, DB5, DB4, DB3, DB2, DB1, DB0, DAp, DA5, DA4, DA3, DA2, DA1, DA0],...
    [Pango(4),Stateo(19),Stateo(22),Stateo(25),Stateo(28),Stateo(31),Stateo(34),Pango(5),Stateo(20),Stateo(23),Stateo(26),Stateo(29),State

output=subs(output,[Tpdot, T5dot, T4dot, T3dot, T2dot, T1dot, T0dot, Bpdot, B5dot, B4dot, B3dot, B2dot, B1dot, B0dot, Apdot, A5dot, A4dot,
    [Pango(4),Stateo(19),Stateo(22),Stateo(25),Stateo(28),Stateo(31),Stateo(34),Pango(5),Stateo(20),Stateo(23),Stateo(26),Stateo(29),State

output=subs(output,[L0,L1,L2,L3,L4,L5,la,lb,cent3,g,m0,m1,m2,m3,m4,m5,cent1,cent2],...
    [Vars(1),Vars(2),Vars(3),Vars(4),Vars(5),Vars(6),Vars(7),Vars(8),Vars(9),Vars(10),Vars(11),Vars(12),Vars(13),Vars(14),Vars(15),Vars(16

output=subs(output,[L1_ChSpace,L2_ChSpace,L3_ChSpace,L4_ChSpace,L5_ChSpace],...
    [ChSpace(1,:),ChSpace(2,:),ChSpace(3,:),ChSpace(4,:),ChSpace(5,:)]);

output=subs(output,[L1_space,L2_space,L3_space,L4_space,L5_space],...
    [ChSpace(1,:),ChSpace(2,:),ChSpace(3,:),ChSpace(4,:),ChSpace(5,:)]);

%convert the data type
output=double(output);
end
```

D.2 *vect2ang.m*: Assign angles from input angle vectors to individual variables for function input

```
% Valerie Jardon
% Department of Bioengineering, University of Kansas
% vect2ang.m
% Last Edited: 4/30/2022

% This function takes vector inputs of angles and separates them into
% individual angles for analysis
function [A0,A1,A2,A3,A4,A5,Ap,B0,B1,B2,B3,B4,B5,Bp,DA0,DA1,DA2,DA3,DA4,DA5,DAp,DB0, ↙
DB1,DB2,DB3,DB4,DB5,DBp,DT0,DT1,DT2,DT3,DT4,DT5,DTp,T0,T1,T2,T3,T4,T5,Tp, ...
    A0dot,A1dot,A2dot,A3dot,A4dot,A5dot,T0dot,T1dot,T2dot,T3dot,T4dot,T5dot,B0dot, ↙
B1dot,B2dot,B3dot,B4dot,B5dot,Apdot,Bpdot,Tpdot]=vect2ang(Stateo,Pango)

T5=Stateo(1);  B5=Stateo(2); A5=Stateo(3);
T4=Stateo(4);  B4=Stateo(5); A4=Stateo(6);
T3=Stateo(7);  B3=Stateo(8); A3=Stateo(9);
T2=Stateo(10); B2=Stateo(11); A2=Stateo(12);
T1=Stateo(13); B1=Stateo(14); A1=Stateo(15);
T0=Stateo(16); B0=Stateo(17); A0=Stateo(18);
Tp=Pango(1);   Bp=Pango(2);   Ap=Pango(3);

DT5=Stateo(19); DB5=Stateo(20); DA5=Stateo(21);
DT4=Stateo(22); DB4=Stateo(23); DA4=Stateo(24);
DT3=Stateo(25); DB3=Stateo(26); DA3=Stateo(27);
DT2=Stateo(28); DB2=Stateo(29); DA2=Stateo(30);
DT1=Stateo(31); DB1=Stateo(32); DA1=Stateo(33);
DT0=Stateo(34); DB0=Stateo(35); DA0=Stateo(36);
DTp=Pango(4);   DBp=Pango(5);   DAp=Pango(6);

T5dot=Stateo(19); B5dot=Stateo(20); A5dot=Stateo(21);
T4dot=Stateo(22); B4dot=Stateo(23); A4dot=Stateo(24);
T3dot=Stateo(25); B3dot=Stateo(26); A3dot=Stateo(27);
T2dot=Stateo(28); B2dot=Stateo(29); A2dot=Stateo(30);
T1dot=Stateo(31); B1dot=Stateo(32); A1dot=Stateo(33);
T0dot=Stateo(34); B0dot=Stateo(35); A0dot=Stateo(36);
Tpdot=Pango(4);   Bpdot=Pango(5);   Apdot=Pango(6);
end
```

D.3 *sub_timevar.mlx*: Replaces angles in variables with time-dependent angles

Valerie Jardon

Department of Bioengineering, University of Kansas

sub_timevar.mlx

Last Edited: 4/30/2022

This function is used to replace variables without time-dependency to variables with time-dependency

```
function [output]=sub_timevar(input)
syms T5 B5 A5 T4 B4 A4 T3 B3 A3 T2 B2 A2 T1 B1 A1 T0 B0 A0 Tp Bp Ap
syms T5_val B5_val A5_val T4_val B4_val A4_val T3_val B3_val A3_val T2_val B2_val A2_val T1_val B1_val A1_val T0_val B0_val A0_val Tp_val
output=subs(input,[T5 B5 A5 T4 B4 A4 T3 B3 A3 T2 B2 A2 T1 B1 A1 T0 B0 A0 Tp Bp Ap],[T5_val B5_val A5_val T4_val B4_val A4_val T3_val B3_va

syms T5(t) B5(t) A5(t) T4(t) B4(t) A4(t) T3(t) B3(t) A3(t) T2(t) B2(t) A2(t) T1(t) B1(t) A1(t) T0(t) B0(t) A0(t) Tp(t) Bp(t) Ap(t)
output=subs(output,[T5_val B5_val A5_val T4_val B4_val A4_val T3_val B3_val A3_val T2_val B2_val A2_val T1_val B1_val A1_val T0_val B0_val
end
```

D.4 *tofuncts.m*: Convert Symbolic Outputs to Function Handles

```
%Valerie Jardon
%Department of Bioengineering, University of Kansas
%tofuncts.m
%Last Edited: 4/30/2022

%Description: This function (tofuncts.m) is used to convert our symbolic
%outputs to function handles using the built-in MATLAB function
%"matlabFunction()".

function [Gl_funct,Gr_funct,Mm_funct,Cm_funct,Gm_funct,Ma_funct,ML_funct,MV_funct, ↙
Ea_funct,Rm_funct,dRm_funct,origin_funct,dorigin_funct]=tofuncts()
syms T5 T5dot B5 B5dot A5 A5dot T4 T4dot B4 B4dot A4 A4dot T3 T3dot B3 B3dot A3 A3dot↙
T2 T2dot B2 B2dot A2 A2dot T1 T1dot B1 B1dot A1 A1dot T0 T0dot B0 B0dot A0 A0dot Tp↙
Bp Ap Tpdot Bpdot Apdot
syms Tpdot_val Bpdot_val Apdot_val T5_val T5dot_val B5_val B5dot_val A5_val A5dot_val↙
T4_val T4dot_val B4_val B4dot_val A4_val A4dot_val T3_val T3dot_val B3_val B3dot_val↙
A3_val A3dot_val T2_val T2dot_val B2_val B2dot_val A2_val A2dot_val T1_val T1dot_val↙
B1_val B1dot_val A1_val A1dot_val T0_val T0dot_val B0_val B0dot_val A0_val A0dot_val↙
la lb Apddot Bpddot Tpddot T5ddot B5ddot A5ddot T4ddot B4ddot A4ddot T3ddot B3ddot↙
A3ddot T2ddot B2ddot A2ddot T1ddot B1ddot A1ddot T0ddot B0ddot A0ddot Tp_val Bp_val↙
Ap_val

%load the symbolic output of our files
load('rotation_matrices_sym_0toP.mat') %rotation matrices
load('origin_sym_0toP.mat') %origin
load('rotate_anatomy_sym_MaMLMV.mat') %Ma, ML, MV
load('dynam_sym_output.mat') %Gl, Gr, Mm, Cm, Gm
load('Ea_sym.mat') %Ea

%rotate anatomy sym outputs
Ma_funct=matlabFunction(Ma);
ML_funct=matlabFunction(ML);
MV_funct=matlabFunction(MV);

%rotate external force sym output
Ea_funct=matlabFunction(Ea);

%rotation matrices and origins
Rm_funct=matlabFunction(rotation_matrices);
dRm_funct=matlabFunction(d_rotation_matrices);
origin_funct=matlabFunction(origin);
dorigin_funct=matlabFunction(d_origin);

%dynamics sym outputs
Gl=subs(Gl,[T5 T5dot B5 B5dot A5 A5dot T4 T4dot B4 B4dot A4 A4dot T3 T3dot B3 B3dot ↙
A3 A3dot T2 T2dot B2 B2dot A2 A2dot T1 T1dot B1 B1dot A1 A1dot T0 T0dot B0 B0dot A0 ↙
A0dot Tp Bp Ap Tpdot Bpdot Apdot],[T5_val T5dot_val B5_val B5dot_val A5_val A5dot_val ↙
T4_val T4dot_val B4_val B4dot_val A4_val A4dot_val T3_val T3dot_val B3_val B3dot_val ↙
A3_val A3dot_val T2_val T2dot_val B2_val B2dot_val A2_val A2dot_val T1_val T1dot_val ↙
```

```
B1_val B1dot_val A1_val A1dot_val T0_val T0dot_val B0_val B0dot_val A0_val A0dot_val ↙
Tp_val Bp_val Ap_val Tpdot_val Bpdot_val Apdot_val]);
Gl=subs(Gl,[T5_val T5dot_val B5_val B5dot_val A5_val A5dot_val T4_val T4dot_val ↙
B4_val B4dot_val A4_val A4dot_val T3_val T3dot_val B3_val B3dot_val A3_val A3dot_val ↙
T2_val T2dot_val B2_val B2dot_val A2_val A2dot_val T1_val T1dot_val B1_val B1dot_val ↙
A1_val A1dot_val T0_val T0dot_val B0_val B0dot_val A0_val A0dot_val Tp_val Bp_val ↙
Ap_val Tpdot_val Bpdot_val Apdot_val],[T5 T5dot B5 B5dot A5 A5dot T4 T4dot B4 B4dot ↙
A4 A4dot T3 T3dot B3 B3dot A3 A3dot T2 T2dot B2 B2dot A2 A2dot T1 T1dot B1 B1dot A1 ↙
A1dot T0 T0dot B0 B0dot A0 A0dot Tp Bp Ap Tpdot Bpdot Apdot]);
Gr=subs(Gr,[T5 T5dot B5 B5dot A5 A5dot T4 T4dot B4 B4dot A4 A4dot T3 T3dot B3 B3dot ↙
A3 A3dot T2 T2dot B2 B2dot A2 A2dot T1 T1dot B1 B1dot A1 A1dot T0 T0dot B0 B0dot A0 ↙
A0dot Tp Bp Ap Tpdot Bpdot Apdot],[T5_val T5dot_val B5_val B5dot_val A5_val A5dot_val ↙
T4_val T4dot_val B4_val B4dot_val A4_val A4dot_val T3_val T3dot_val B3_val B3dot_val ↙
A3_val A3dot_val T2_val T2dot_val B2_val B2dot_val A2_val A2dot_val T1_val T1dot_val ↙
B1_val B1dot_val A1_val A1dot_val T0_val T0dot_val B0_val B0dot_val A0_val A0dot_val ↙
Tp_val Bp_val Ap_val Tpdot_val Bpdot_val Apdot_val]);
Gr=subs(Gr,[T5_val T5dot_val B5_val B5dot_val A5_val A5dot_val T4_val T4dot_val ↙
B4_val B4dot_val A4_val A4dot_val T3_val T3dot_val B3_val B3dot_val A3_val A3dot_val ↙
T2_val T2dot_val B2_val B2dot_val A2_val A2dot_val T1_val T1dot_val B1_val B1dot_val ↙
A1_val A1dot_val T0_val T0dot_val B0_val B0dot_val A0_val A0dot_val Tp_val Bp_val ↙
Ap_val Tpdot_val Bpdot_val Apdot_val],[T5 T5dot B5 B5dot A5 A5dot T4 T4dot B4 B4dot ↙
A4 A4dot T3 T3dot B3 B3dot A3 A3dot T2 T2dot B2 B2dot A2 A2dot T1 T1dot B1 B1dot A1 ↙
A1dot T0 T0dot B0 B0dot A0 A0dot Tp Bp Ap Tpdot Bpdot Apdot]);
Gl_funct=matlabFunction(Gl);
Gr_funct=matlabFunction(Gr);

Mm=subs(Mm,[T5 T5dot B5 B5dot A5 A5dot T4 T4dot B4 B4dot A4 A4dot T3 T3dot B3 B3dot ↙
A3 A3dot T2 T2dot B2 B2dot A2 A2dot T1 T1dot B1 B1dot A1 A1dot T0 T0dot B0 B0dot A0 ↙
A0dot Tp Bp Ap Tpdot Bpdot Apdot],[T5_val T5dot_val B5_val B5dot_val A5_val A5dot_val ↙
T4_val T4dot_val B4_val B4dot_val A4_val A4dot_val T3_val T3dot_val B3_val B3dot_val ↙
A3_val A3dot_val T2_val T2dot_val B2_val B2dot_val A2_val A2dot_val T1_val T1dot_val ↙
B1_val B1dot_val A1_val A1dot_val T0_val T0dot_val B0_val B0dot_val A0_val A0dot_val ↙
Tp_val Bp_val Ap_val Tpdot_val Bpdot_val Apdot_val]);
Mm=subs(Mm,[T5_val T5dot_val B5_val B5dot_val A5_val A5dot_val T4_val T4dot_val ↙
B4_val B4dot_val A4_val A4dot_val T3_val T3dot_val B3_val B3dot_val A3_val A3dot_val ↙
T2_val T2dot_val B2_val B2dot_val A2_val A2dot_val T1_val T1dot_val B1_val B1dot_val ↙
A1_val A1dot_val T0_val T0dot_val B0_val B0dot_val A0_val A0dot_val Tp_val Bp_val ↙
Ap_val Tpdot_val Bpdot_val Apdot_val],[T5 T5dot B5 B5dot A5 A5dot T4 T4dot B4 B4dot ↙
A4 A4dot T3 T3dot B3 B3dot A3 A3dot T2 T2dot B2 B2dot A2 A2dot T1 T1dot B1 B1dot A1 ↙
A1dot T0 T0dot B0 B0dot A0 A0dot Tp Bp Ap Tpdot Bpdot Apdot]);
Mm_funct=matlabFunction(Mm);

syms t T5(t) T5dot(t) B5(t) B5dot(t) A5(t) A5dot(t) T4(t) T4dot(t) B4(t) B4dot(t) A4↙
(t) A4dot(t) T3(t) T3dot(t) B3(t) B3dot(t) A3(t) A3dot(t) T2(t) T2dot(t) B2(t) B2dot↙
(t) A2(t) A2dot(t) T1(t) T1dot(t) B1(t) B1dot(t) A1(t) A1dot(t) T0(t) T0dot(t) B0(t)↙
B0dot(t) A0(t) A0dot(t) Tp(t) Bp(t) Ap(t) Tpdot(t) Bpdot(t) Apdot(t)
Cm=subs(Cm,[T5(t) T5dot(t) B5(t) B5dot(t) A5(t) A5dot(t) T4(t) T4dot(t) B4(t) B4dot↙
(t) A4(t) A4dot(t) T3(t) T3dot(t) B3(t) B3dot(t) A3(t) A3dot(t) T2(t) T2dot(t) B2(t) ↙
B2dot(t) A2(t) A2dot(t) T1(t) T1dot(t) B1(t) B1dot(t) A1(t) A1dot(t) T0(t) T0dot(t) ↙
```

```
B0(t) B0dot(t) A0(t) A0dot(t) Tp(t) Bp(t) Ap(t) Tpdot(t) Bpdot(t) Apdot(t)],[T5_val ↙
T5dot_val B5_val B5dot_val A5_val A5dot_val T4_val T4dot_val B4_val B4dot_val A4_val ↙
A4dot_val T3_val T3dot_val B3_val B3dot_val A3_val A3dot_val T2_val T2dot_val B2_val ↙
B2dot_val A2_val A2dot_val T1_val T1dot_val B1_val B1dot_val A1_val A1dot_val T0_val ↙
T0dot_val B0_val B0dot_val A0_val A0dot_val Tp_val Bp_val Ap_val Tpdot_val Bpdot_val ↙
Apdot_val]);
Gm=subs(Gm,[T5 T5dot B5 B5dot A5 A5dot T4 T4dot B4 B4dot A4 A4dot T3 T3dot B3 B3dot ↙
A3 A3dot T2 T2dot B2 B2dot A2 A2dot T1 T1dot B1 B1dot A1 A1dot T0 T0dot B0 B0dot A0 ↙
A0dot Tp Bp Ap Tpdot Bpdot Apdot],[T5_val T5dot_val B5_val B5dot_val A5_val A5dot_val ↙
T4_val T4dot_val B4_val B4dot_val A4_val A4dot_val T3_val T3dot_val B3_val B3dot_val ↙
A3_val A3dot_val T2_val T2dot_val B2_val B2dot_val A2_val A2dot_val T1_val T1dot_val ↙
B1_val B1dot_val A1_val A1dot_val T0_val T0dot_val B0_val B0dot_val A0_val A0dot_val ↙
Tp_val Bp_val Ap_val Tpdot_val Bpdot_val Apdot_val]);

syms T5 T5dot B5 B5dot A5 A5dot T4 T4dot B4 B4dot A4 A4dot T3 T3dot B3 B3dot A3 A3dot↙
T2 T2dot B2 B2dot A2 A2dot T1 T1dot B1 B1dot A1 A1dot T0 T0dot B0 B0dot A0 A0dot Tp↙
Bp Ap Tpdot Bpdot Apdot
Cm=subs(Cm,[T5_val T5dot_val B5_val B5dot_val A5_val A5dot_val T4_val T4dot_val ↙
B4_val B4dot_val A4_val A4dot_val T3_val T3dot_val B3_val B3dot_val A3_val A3dot_val ↙
T2_val T2dot_val B2_val B2dot_val A2_val A2dot_val T1_val T1dot_val B1_val B1dot_val ↙
A1_val A1dot_val T0_val T0dot_val B0_val B0dot_val A0_val A0dot_val Tp_val Bp_val ↙
Ap_val Tpdot_val Bpdot_val Apdot_val],[T5 T5dot B5 B5dot A5 A5dot T4 T4dot B4 B4dot ↙
A4 A4dot T3 T3dot B3 B3dot A3 A3dot T2 T2dot B2 B2dot A2 A2dot T1 T1dot B1 B1dot A1 ↙
A1dot T0 T0dot B0 B0dot A0 A0dot Tp Bp Ap Tpdot Bpdot Apdot]);
Gm=subs(Gm,[T5_val T5dot_val B5_val B5dot_val A5_val A5dot_val T4_val T4dot_val ↙
B4_val B4dot_val A4_val A4dot_val T3_val T3dot_val B3_val B3dot_val A3_val A3dot_val ↙
T2_val T2dot_val B2_val B2dot_val A2_val A2dot_val T1_val T1dot_val B1_val B1dot_val ↙
A1_val A1dot_val T0_val T0dot_val B0_val B0dot_val A0_val A0dot_val Tp_val Bp_val ↙
Ap_val Tpdot_val Bpdot_val Apdot_val],[T5 T5dot B5 B5dot A5 A5dot T4 T4dot B4 B4dot ↙
A4 A4dot T3 T3dot B3 B3dot A3 A3dot T2 T2dot B2 B2dot A2 A2dot T1 T1dot B1 B1dot A1 ↙
A1dot T0 T0dot B0 B0dot A0 A0dot Tp Bp Ap Tpdot Bpdot Apdot]);
Cm_funct=matlabFunction(Cm);
Gm_funct=matlabFunction(Gm);
end
```