Modeling and Measurement of Pressure Overload-Induced Changes in Left Ventricular Structure and Function

Genevieve E. Farrar

A dissertation
Submitted in partial fulfillment of the
Requirements for the degree of

Doctor of Philosophy

University of Washington

2014

Reading Committee:
Alexander I. Veress, Chair
Alberto Aliseda
Nathan Snidecki

Program Authorized to Offer Degree:
Mechanical Engineering

University of Washington

Abstract

Modeling and Measurement of Pressure Overload-Induced Changes in
Left Ventricular Structure and Function

Genevieve Farrar

Chair of the Supervisory Committee:
Dr. Alexander I. Veress
Department of Mechanical Engineering

Biomechanical modeling of pressure overload-induced changes in left ventricular (LV) structure and function has the potential to make significant contributions to both basic science and clinical treatment strategies for hypertension. Hypertension affects approximately one third of the population of the United States, and in spite of increased pharmacological treatment in recent decades, it continues to lead to heart failure for tens of thousands of people per year [1]. Cardiac biomechanical analysis is still a relatively new field, having had to wait for advances in computation, medical imaging, and theoretical frameworks appropriate for describing biological soft tissues. While finite element analysis of normal left ventricular mechanics is well described in the literature [2-4], the analysis and prediction of altered mechanics in diseased states is complicated by the adaptive nature of biological tissues. Growth and remodeling of the left ventricle in response to pressure overload (hypertension) is a decades-long process of thickening and stiffening that begins as a mild adaptive response but that ultimately can lead to heart failure. The degree of hypertrophy can actually predict medical outcomes [5]. The ability to predict hypertrophy, and understand the altered mechanics that occur in a hypertrophied LV is integral to the development of mechanics-based treatment strategies for an organ that is normally an incredibly effective pump. In this dissertation, the first chapter presents relevant background

information, the second chapter describes an experimental study of longitudinal changes in LV strain in hypertensive hypertrophy, and the third and fourth chapters describe a new approach to modeling load-induced hypertrophic adaptation used to assess systolic work as a potenetial mechanical growth stimulus for modeling hypertensive hypertrophy in the SHR.

In Chapter 2, an FEA-based image registration technique was used to determine strain distributions in the left ventricle (LV) over the cardiac cycle. This analysis technique was applied to microPET images from hypertensive rats and normal controls over their respective lifespans, and focused on gaining a better understanding of the longitudinal changes in LV deformation behavior that occur in relation to progressive hypertrophy in the hypertensive subjects. The results of this study indicated that diastolic dysfunction preceded systolic dysfunction, and that changes in diastolic and systolic strains are related to different physiological changes.

Chapter 3 describes (1) the development of a novel approach for modeling load-induced hypertrophic growth that can incorporate mechanical stimuli from throughout the cardiac cycle, and (2) a model study that uses this framework to evaluate the hypothesis that systolic work is an appropriate mechanical growth stimulus for pressure-overload hypertrophy. Growth and Remodeling (G&R) is a relatively new branch in the field of cardiac solid mechanics that seeks to model the changes in cardiac geometry that occur in response to altered or diseased mechanics. Hypertensive hypertrophy is an example of load induced cardiac adaptation in which G&R models can be applied. Prior work in this field had yielded a constitutive modeling approach to G&R and numerous computational implemetations. However a need was observed for a way to incorporate information from the whole cardiac cycle into the growth stimulus. A method was developed for predictive modeling of pressure overload-induced left ventricular growth, which can incorporate multiple mechanical stimuli from throughout the cardiac cycle.

This was implemented using FEBio, an open-source finite element analysis package designed for biomechanics (FEbio.org), and a custom program called CGR ("cardiac growth and remodeling"). The model has the ability to determine the degree of hypertrophy that might occur for a given level of pressure overload (hypertension), and has the ability to create a distribution of element growth through the LV wall that is based on any measure(s) of stress or strain from the full cardiac cycle mechanics. This approach was used to assess the hypothesis that systolic work can be used as an appropriate growth stimulus for the modeling of hypertensive hypertrophy. The systolic work normalization yielded predictions of wall thickness that were realistic compared with the experimental data, indicating that systolic work may indeed be a good choice of growth stimulus for pressure overload.

The fourth chapter describes a model study which quantifies the effects of changes in systolic pressure and changes in material stiffness on the systolic workload, and the resulting growth predictions. Both systolic pressure and material stiffness caused linear increases in the LV workload. Increases in the LV workload caused increases in LV wall thickness with a nonlinear which increases slope as the workload increases. The range of systolic pressures and material stiffnesses evaluated were representative of what is observed experiementally. This study quantified the effects of incremental increases in systolic work, and revealed the importance of the way the growth law is defined. Potential future extensions of this work are discussed.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF EQUATIONS

ACKNOWLEDGMENTS

I would foremost like to acknowledge Dr. Alexander Veress for serving as my PhD advisor. It is likely that I would never have become involved in the field of cardiac biomechanics had he not invited me to join his research group. Since that time, and through our conversations, I have come to have a great appreciation for this subject, and I look forward to continuing my career in this field.

Of course I must acknowledge my dissertation committee members: Dr. Alberto Aliseda and Dr. Nathan Sniadecki from the Department of Mechanical Engineering, and Dr. Jim Bassingthwaighte from the Department of Bioengineering. I would like to thank all of them for their insights, as each of them had a uniquely applicable perspective.

My collaborators at Lawrence Berkeley National Laboratories must be ackowledged for providing the imaging data used to create the geometries for all of my finite element models, as well as additional data on the SHR. Dr. Grant Gullberg provided great insight on medical imaging technologies and was an excellent collaborator.

Finally, I would like to thank my family for their encouragement over the years, as well as my friends who have been a part of my graduate school experience.

This work was financially supported by NIH grants R01EB000121, R01EB07219, R01HL091036, R01CA134658 and R03EB008450, as well as the GAANN fellowship.

Chapter 1. Background and Motivation

1.1: Introduction

Cardiac solid mechanics is an interesting field of research because the heart is a complex system to model: The left ventricular myocardium has nonlinear, anisotropic, spatially and time varying material properties, non-uniform geometry and poorly defined boundary conditions. In addition, material constituents are constantly undergoing replacement, such that the global geometry and material properties are subject to change in response to their loading environment. This chapter provides general background information together with a literature review of prior research, which will provide motivation for the studies presented in the following chapters.

First, some general materials and methods will be described including the continuum mechanical principles necessary for solving initial and boundary value problems in solid mechanics, the basic theory of the nonlinear finite element method for solving the discretized equilibrium equations, and some common medical imaging modalities used for obtaining patient-specific data of cardiac structure and function.

Normal cardiovascular structure and function will be discussed such that the interested reader can understand the most important elements of cardiac solid mechanics that will be referred to in later chapters. This section will include a description of the myocardial structure and how this relates to the constitutive laws that have been proposed to describe its passive

mechanical behavior. Then, the physiological muscle contraction mechanisms will be discussed, leading to the active material property descriptions that are commonly used in cardiac modeling. The cardiac cycle and some general overall measures of cardiac mechanical function will also be discussed. These measures will later be used to describe changes in function that occur in diseases such as hypertension.

Hypertension and its resulting hypertrophic adaptation will then be described with human data as well as with data collected from animal models such as the spontaneously hypertensive rat (SHR). This section will provide motivation for the experimental study of the changes in left ventricular strain in hypertensive hypertrophy, which is the topic of Chapter 2.

A general discussion of finite element modeling applied the left ventricle will follow, including typical modeling assumptions related to the loads, boundary conditions, geometry and material properties.

The final section of this chapter will describe the development of theoretical and computational models of mechanical stimuli-induced volumetric growth applied to cardiac hypertrophy. This background will motivate the method for modeling left ventricular growth that will be the topic of chapter 3.

1.2: General Materials and Methods

Continuum Mechanics

In typical solid mechanics analyses, the goal is to understand the stresses and strains in a body that occur in response to applied loads. In continuum mechanics, material bodies are considered to be a continuum, that is, the material properties are spread out in a homogenous manner. Continuum mechanical descriptions are generally considered applicable to physical phenomena in which the size of the constituent particles is at least two orders of magnitude smaller than the size of the body being analyzed. For example, in metals, the constituent atoms are many orders of magnitude smaller than most metal objects in engineering applications. And in cardiac muscle, individual cells are on the order of tens of micrometers, while the left ventricular wall is centimeters thick in humans, and millimeters thick in small animal models. Continuum mechanics has proven to be very useful in analyzing a wide variety of engineering applications, and is well-suited to whole-organ mechanical analysis of the heart, where the goal is to understand and describe the distributions of loading and deformation that occur in the cardiac muscle over the cardiac cycle. For a complete treatment of traditional continuum mechanics, see references [6, 7]. There are five main pieces of information necessary for solving problems in continuum mechanics: Kinematics (displacements, strains), Forces (stresses, tractions), Balance laws (momentum, energy), Constitutive relations (material behavior), and boundary conditions.

Kinematics describes the movement of a body, including the displacement, rotation, velocity, and deformation. The deformation gradient, $\boldsymbol{F}$, describes the motion. It is a second order tensor which maps an oriented line segment $d\boldsymbol{X}$ in the original (undeformed) configuration to $d\boldsymbol{x}$ in the current (deformed) configuration of a body: $d\boldsymbol{x} = \boldsymbol{F} \cdot d\boldsymbol{X}$. While $\boldsymbol{F}$ is the fundamental measure describing the motion, it is not very practical for analyses of the deformation because it contains rigid body motion information, it is not symmetric, and it is a

two-point tensor. The right, $\boldsymbol{C}$, and left, $\boldsymbol{b}$, Cauchy-Green tensors do not contain rigid body motion, are symmetric, and are one-point tensors:

$$\boldsymbol{C} = \boldsymbol{F}^T \cdot \boldsymbol{F} \ , \ \boldsymbol{b} = \boldsymbol{F} \cdot \boldsymbol{F}^T \qquad \textbf{Equation 1.1}$$

These measures reduce to the identity tensor when there is no deformation. It is useful in the development of constitutive laws to have a deformation measure that is zero when there is no deformation. The Green-Lagrange strain, $\boldsymbol{E}$, is defined from $\boldsymbol{C}$, and the Almansi strain, $\boldsymbol{\epsilon}$, is defined from $\boldsymbol{b}$:

$$\boldsymbol{E} = \frac{1}{2}(\boldsymbol{C} - \boldsymbol{I}) \ , \ \boldsymbol{\epsilon} = \frac{1}{2}(\boldsymbol{I} - \boldsymbol{b}^{-1}) \qquad \textbf{Equation 1.2}$$

In this work, the strain results reported will be based on the Green-Lagrange definition of strain.

Forces and Stresses describe the state of the loads on a body. Externally applied forces and tractions will result in internal stresses in the material. The 3D state of stress is described by a second order tensor. The Cauchy stress or "true stress" $\boldsymbol{\sigma}$ is defined by

$\boldsymbol{T} = \boldsymbol{n} \cdot \boldsymbol{\sigma}$, where $\boldsymbol{T}$ is a force acting on the surface defined by the surface normal $\boldsymbol{n}$.

In many applications it has proved to be useful to have a scalar measure of stress on a body. For example, in metals, yielding of the material can be predicted fairly well by the von Mises stress, which is an average measure of the stresses in the 3D state. For biological soft tissues, which often have a transversely isotropic or orthotropic material behavior, it is often the stress in one of the preferred (fiber) direction(s) that is of interest. In this case the scalar-valued fiber stress would be: $\sigma_f = \boldsymbol{\sigma} \cdot \boldsymbol{a}$, where $\boldsymbol{a}$ is the vector describing the fiber orientation.

Balance Laws are the fundamental first principals that enable the calculation of solutions, including the prediction of motion. In solid mechanics, conservation of linear momentum is the balance law that is applied to solve the boundary value problems.

$$\nabla \cdot \boldsymbol{\sigma} + \rho\, \boldsymbol{f} = \rho\, \ddot{\boldsymbol{u}} \qquad \textbf{Equation 1.3}$$

This is commonly known as the equation of Motion. In static analyses, the right hand side is zero, and it is referred to as the Equilibrium equation. It states that the internal stresses are in equilibrium with the applied loads.

There are two other balance laws that influence the results in typical solid mechanics analysis. Conservation of angular momentum results in the restriction that $\boldsymbol{\sigma} = \boldsymbol{\sigma}^{\boldsymbol{T}}$, which affects the types of constitutive laws that are admissible. Conservation of mass maintains the mass in the system by defining the Jacobian as $J = \rho/\rho_0$, the ratio of the mass densities in the initial and final configurations. The Jacobian is also used to describe the volume change of the deformed material $J = \det \boldsymbol{F}$.

Constitutive Relations describe the relationship between the stress and the resulting deformation (strain). It is worth noting that a constitutive law describes a behavior, not a material. That is, it describes the behavior of a material under specific conditions. There may be multiple constitutive laws used to describe the behavior of a given material under different conditions.

Stress can be determined from a strain energy function $W(\boldsymbol{F})$ of the deformation. (Note that this is restricted to isothermal processes, which is all that is considered here.) In general,

$$\boldsymbol{\sigma} = \frac{2}{J}\, \boldsymbol{F}\, \frac{\partial W}{\partial \boldsymbol{C}}\, \boldsymbol{F}^{\boldsymbol{T}}\ , \text{ or } \ \mathbf{S} = \frac{\partial W}{\partial \boldsymbol{E}} \qquad \textbf{Equation 1.4}$$

where $\mathbf{S}$ is the Second Piola-Kirckoff stress, which is defined with respect to the original configuration.

The particular constitutive relations used to describe the behavior of to the left ventricular myocardium over the cardiac cycle will be discussed in detail in Section 1.3. Because muscle material has an active contractile material behavior in addition to its passive mechanical

properties, proper mechanical analysis of the heart tissue requires the description of both types of material behavior. The passive material behavior will be described in the context of the tissue structure, and the active material behavior will be described in the context of cardiac muscle function.

Finite Element Method

The Finite Element Method is a numerical method for solving boundary-value and initial-value problems, in which the body is physically discretized into finite pieces, or elements, and the equations of motion are solved simultaneously for each element. This makes it possible to determine stress and strain distributions in a body with complex geometry. For a more complete discussion of the finite element method, see references [8, 9]. The nonlinear finite element method is based on solving the weak form of the equilibrium equation. When derived based on the principle of virtual work, this is:

$$\delta W = \int_v \boldsymbol{\sigma} : \delta \boldsymbol{d}\, dv - \int_v \boldsymbol{f} \cdot \delta \boldsymbol{v}\, dv - \int_{\partial v} \boldsymbol{t} \cdot \delta \boldsymbol{v}\, da = 0 \qquad \textbf{Equation 1.5}$$

where $\delta \boldsymbol{d}$ is the virtual rate of deformation and $\delta \boldsymbol{v}$ is the arbitrary virtual velocity. The first term represents the contribution from the internal Cauchy stress $\boldsymbol{\sigma}$ within the volume of the body, the second term is the contribution from body forces $\boldsymbol{f}$ within the volume of the body, and the third term is the contribution from the tractions $\boldsymbol{t}$ on the surface of the body. This equation is then linearized in the direction of an incremental displacement $\boldsymbol{u}$ with $\phi_{\boldsymbol{k}}$ being a trial solution.

$$\delta W(\phi_{\boldsymbol{k}}, \delta \boldsymbol{v}) + D\delta W(\phi_{\boldsymbol{k}}, \delta \boldsymbol{v})[\boldsymbol{u}] = 0 \qquad \textbf{Equation 1.6}$$

The calculation of the directional derivative of the virtual work equation leads to the stiffness matrix. This will have a component from internal virtual work as well as a component from external virtual work. The set of nonlinear equilibrium equations (in which the current nodal positions are the unknowns) is solved using an iterative procedure. FEBio uses the Broyden-Fletcher-Goldfarb-Shanno (BFGS) method, which is a quasi-Newton method [10]. Nike3D has several iterative methods available [11], and BFGS was selected.

Many element shapes are used in different finite element applications in order to capture the appropriate mechanical behavior. The displacements are solved at the nodes, and the results between nodes are interpolated based on the shape functions of the given element type. In cardiac finite element modeling, it is common practice use 8-node solid "brick" elements. In this case, displacements are linearly interpolated between nodes.

Cardiac Imaging

Many imaging modalities have been developed in order to study the structure and function of the heart in both clinical and research settings. The two categories of cardiac imaging of interest here are (1) nuclear imaging such as positron emission tomography (PET) and single photon emission computed tomography (SPECT), and (2) magnetic resonance imaging (MRI).

In PET imaging, a positron-emitting radiotracer is injected into the bloodstream attached to a biologically active molecule, which is then taken up by tissues of interest. The imaging system then detects pairs of gamma rays emitted by the tracer, and 3D images are created of the distribution of the tracer in the body. SPECT works in a similar manner in which emitted photons are detected. Nuclear imaging is very versatile in that, depending on the chosen radio

tracer, a number of different aspects of functional physiology can be measured [12]. In cardiac imaging, PET can be used to measure perfusion of blood within the heart wall as well as the use of various metabolic substrates such as glucose and fatty acids [13, 14]. Cardiac perfusion images can also be used to obtain the geometry of the LV over the cardiac cycle [15, 16]. A particular set-up for PET imaging of small animal models known as microPET [17] can be used to study changes in function that occur over the lifespan of the animals. In Chapter 2 of this document, microPET perfusion images were used in combination with an image registration technique to determine strains in the left ventricle.

Magnetic resonance imaging works by using a magnetic field to align the magnetization of atomic nuclei in the body, which causes the nuclei to produce a rotating magnetic field detectable by the scanner, and this information is used to construct an image [18]. There are several useful applications of MRI in cardiac assessments. In tagged MRI, lines or "tags" are placed in the image in the form of a grid which deforms as the heart deforms, and this is used to estimate strains in the cardiac tissue. However, because the tag spacing is relatively coarse, strain distributions obtained in this manner should be interpreted with caution [16]. Cine MRI has a relatively high temporal resolution and it is possible to observe the deformation of the whole heart over the cardiac cycle. Diffusion tensor MRI, or DTMRI, is a means of determining ex-vivo fiber orientations of the left ventricle [19].

1.3: Normal Cardiac Structure and Function

Introduction to the Cardiovascular System

The human cardiovascular system consists of the heart, the vasculature, and the blood. The heart is a complex and efficient mechanical pump, the vasculature is a vast branching array of tubes that carry blood throughout the body, and the blood is the medium for carrying the oxygen and nutrients that are necessary for living tissues. The vasculature and the heart are coupled to each other in a closed loop of circulation, which has varying pressures throughout the system. The cardiovascular system has near-immediate reaction to changes in need for blood supply to muscles to supply oxygen during exercise, by adapting the heart rate and diameter of blood vessels. In order to meet these functional demands, the heart and vasculature have a complex structure that facilitates efficient use of energy to achieve adequate circulation. Blood travels through two "loops" in series: the pulmonary circulation, and the systemic circulation. The pulmonary circulation carries oxygen-depleted blood from the right side of the heart, through the lungs where it is oxygenated, and back to the left side of the heart. The systemic circulation carries the oxygenated blood from the left side of the heart to all of the tissues of the body, and which then returns to the right side of the heart. The left side of the heart will be the focus of the discussion, in relation to its ability to supply the systemic circulation.

Heart Structure

Heart Chambers

The human heart consists of 4 chambers: the right atrium, right ventricle, left atrium, and left ventricle [**Error! Reference source not found.**]. The right atrium collects blood returning to

the heart from the body via the vena cava. Then the blood passes through the tricuspid valve, into the right ventricle. The right ventricle pumps blood out of the pulmonic valve, into the pulmonary arteries, which carry blood to the lungs for oxygenation. The left atrium collects blood returning to the heart from the lungs via the pulmonary veins. Then the blood passes through the mitral valve into the left ventricle, which then pumps blood to the rest of the body [20, 21]. The atria are thin-walled low-pressure chambers that essentially serve as reservoirs for the ventricles [22]. In that sense, the heart can be thought of as two pumps: the right side serving the pulmonary circulation and the left side serving the systemic circulation. The left ventricle is much thicker and more muscular than the right ventricle because it is pumping blood to all of the tissue of the body and therefore, works against a much greater resistance to flow resulting in a higher pressure.

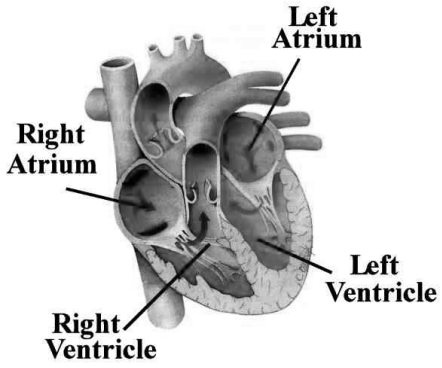

Figure 1-1: The four chambers of the heart.

Pericardium

The whole heart is held within a fluid-lined non-compliant sac called the pericardium. The pericardium has little distensibility, so it resists sudden large increases in cardiac size, and contributes to the mechanical interaction between the two ventricles. When cardiac growth is slow and sustained, such as in the case of hypertrophy (as opposed to a sudden extreme pressure increase) the pericardium will gradually stretch to accommodate the growth of the heart [22].

Heart Layers

The heart wall can be divided into three layers: the epicardium, the endocardium, and the myocardium. The epicardium is a layer of connective tissue on the outer surface of the heart. The endocardium lines the inner surfaces of the atria and ventricles. It consists of a layer of endothelial cells and a mesh of collagen and elastin. The myocardium is the thick central layer between the endo- and epicardium, which contains the functional contractile muscle of the heart. Cardiac myocytes (muscle cells) make of up most of the mass of the myocardium [20]. Myocytes regularly replace their proteins, which enables them to increase (hypertrophy) or decrease (atrophy) in size in response to their mechanical environment (loading). The myocardium also contains a significant amount of extra-cellular matrix (ECM), which is composed largely of collagen. The ECM serves to provide structural scaffolding for the myocytes, and contributes to the passive stiffness of the myocardium. The connective tissue fibers that make up the ECM are secreted and maintained by fibroblasts, the most numerous cell type in the myocardium. The collagen fibers of the ECM undergo regular turnover, and this enables the heart to adapt in stiffness in response to the mechanical environment [21]. In fact, the diameter of the collagen fibers (collagen molecules assemble themselves into fibers) is thought to depend on the mechanical stress field in the ECM. In the cardiovascular system, the half-life of collagen is 15 to 90 days, and so there is continuous synthesis and degradation to maintain the ECM. [21].

Coronary Circulation Blood is supplied to the heart tissue via the coronary circulation. The primary coronary arteries are the left main (LM), left anterior descending (LAD), the left circumflex (CIRC), the right coronary artery (RCA), and the posterior descending artery (PDA). The LM and the RCA originate from the aorta. The LM splits into the LAD and the CIRC. The

PDA is a continuation of either the CIRC or the RCA. These large coronary arteries run along the epicardial (outer) surface of the heart. The large epicardial coronary arteries branch out into smaller vessels as they travel inward toward the myocardium. The coronary circulation perfuses the myocardium primarily during the ventricular filling phase. When the left ventricle contracts, the vessels are temporarily compressed. The myocytes near the endocardial surface are perfused directly from the ventricles via small arteriosinusoidal and arterioluminal vessels. Because this additional means of circulation only affects a very small proportion of the total myocardium (only the myocytes at the endocardial surface), it is insufficient to provide auxiliary circulation when one of the large coronary arteries becomes occluded. Coronary veins collect the deoxygenated blood from the heart. Most of the drainage from the heart goes directly to the right atrium via either the coronary sinus or the anterior cardiac veins [20].

Fiber Structure

The myocardium is organized in a fiber structure which some have described as one helically-wrapped muscle [22], while others consider it to be a continuum with a continuously varying fiber orientations [23]. The fiber orientation is often described by two angles, the helical angle (in the horizontal short-axis plane) and the inclination angle (in the vertical long-axis plane). At mid-height of the ventricle, fiber inclination angles vary through the wall. Approximate values have been measured experimentally to be about +60 degrees at the endocardial surface, 0 degrees at mid-wall, and -70 degrees at the epicardial surface [24]. The steep inclination angles at the endocardial and epicardial surfaces enable the LV to shorten and twist as it contracts, which facilitates ejection. The fibers are organized in sheets, between layers of ECM. Fiber orientations in can be measured ex-vivo with a type of imaging known as DT

MRI as mentioned in S1.2.3. This has been used to create detailed images of LV fiber orientation structure [19] [**Error! Reference source not found.**].

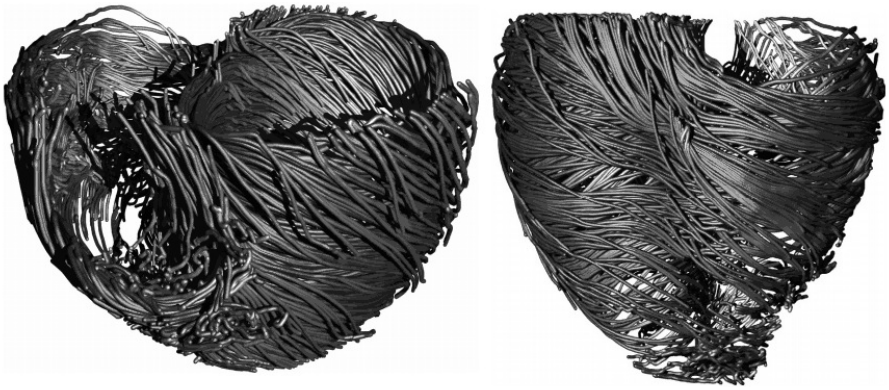

Figure 1-2: Fiber structure of the heart determined from DT-MRI

Constitutive Description of Passive Myocardial Behavior

General characteristics of cardiac soft tissue include nonlinear stress-stretch behavior, large strains, hysteresis and anisotropy. The structure of the myocardium is orthotropic, with the three orientations being the fiber, sheet, and normal directions. However, because the cross-fiber behavior is relatively similar in the sheet and normal directions, it is commonly approximated with a transversely isotropic material model. Many different variations in the formulation of the constitutive law for the passive myocardium have been proposed [23]. The material parameters used in any given model are based on decades of bi-axial testing data [25-28]. One of the models used to describe the passive behavior of the myocardium is the transversely isotropic Mooney-Rivlin model, which is described below.

The transversely isotropic Mooney-Rivlin model describes a material with oriented fibers embedded in a hyperelastic base material. The strain energy function for this material is:

$$\Psi = F_1(\tilde{I}_1, \tilde{I}_2) + F_2(\tilde{\lambda}) + \frac{K}{2}[\ln(J)]^2 \,. \qquad \textbf{Equation 1.7}$$

where (with the tilde denoting the deviatoric part) $\tilde{I}_1$ and $\tilde{I}_2$ are the first and second invariants of $\tilde{C}$, and $\tilde{\lambda}$ is the deviatoric part of the stretch along the fiber direction. The first term represents the response of the base Mooney-Rivlin material [10], while the second term represents the contribution from the fibers. The fiber strain energy us defined by:

$$\tilde{\lambda}\frac{\partial F_2}{\partial \tilde{\lambda}} = 0\,, \qquad \tilde{\lambda} \leq 1 \qquad \textbf{Equation 1.8}$$

$$\tilde{\lambda}\frac{\partial F_2}{\partial \tilde{\lambda}} = C_3(e^{C_4(\tilde{\lambda}-1)} - 1)\,, \qquad 1 < \tilde{\lambda} < \lambda_m$$

$$\tilde{\lambda}\frac{\partial F_2}{\partial \tilde{\lambda}} = C_5 + C_6\tilde{\lambda}\,, \qquad \tilde{\lambda} \geq \lambda_m$$

where C_3 scales the exponential stress, C_4 is the rate of uncramping of fibers, and C_5 is the modulus of the straightened fibers, and λ_m is the stretch at which the fibers are straightened.

Active Heart Function

Cardiac Myocytes and the Sliding Filament Theory

Cardiac myocytes are the muscle cells of the heart. They are approximately 80 to 125 micrometer long and 10 to 20 micrometers in diameter. These cells contain many myofibrils within the cytoplasm. Myofibrils are strings made up of many tiny contractile units, known as sarcomeres, in series. Each sarcomere is made of overlapping thin filaments of actin (5nm) and thick filaments of myosin (10nm) [21]. The sliding filament theory of 1954 [29, 30] states that

the myosin has many transverse “cross bridges” that connect to the actin. Contraction is initiated by the release of calcium ions, and this causes a conformation change in the individual myocin heads resulting in a smooth ratcheting action (at about 15 mm/sec) as the cross bridges release, move forward, and reattach. The force of contraction depends on the initial degree of overlap between actin and myosin filaments, which determines the initial sarcomere length. There is an “optimal” sarcomere length at which the greatest amount of force will be produced, which has been determined experimentally to be about 2 to 2.4 micrometers. There is a direct relationship between the local sarcomere stretch and the global LV stretch due to filling. Within the normal physiological range of ventricular volumes, sarcomere lengths are under or near to the optimal length. The Frank-Starling law of the heart states that greater stretch of the cardiac tissue (within physiological limits) will cause the heart to contract with greater force. This is due to the optimal sarcomere length, and also because stretch makes the sarcomeres more sensitive to calcium. In addition to the force tension relationship, cardiac myocytes actively respond to the mechanical environment in terms of progressive adaptation. By way of protein turnover, cardiac myocytes can change in size and shape *[31]*.

Electrical Activation

The electrical signals travel between cells through connections between heart cells known as gap junctions. Cardiac myocytes are connected to an average of 11.3 other cells, 5.3 on the sides and 6 on the ends [21]. The electrical signal travels into each cell via t-tubules, and causes a release of calcium ions. The calcium ions initiate contraction of the muscle.

Specialized cells in the heart control the initiation of electrical activation, which causes the heart to beat. Activation begins in the SA node, a region of spontaneously depolarizing cells

that lie between the superior vena cava and the right atrium. The wave of electrical activation propagates through the right atrium, then the left atrium. After the atria, the electrical signal reaches an area of slowly conducting cells called the AV node, then a fast conducting area called the AV bundle, where it branches into the left and right bundle branches, which run along the interventricular septum. The impulses carried through the bundle branches reach the His-Purkinje system, a subendocardial network of fast-conducting cells that serve to help synchronize ventricular activation [20]. When the heart motion is viewed with imaging modalities such as cineMRI, the atria appear to contract while the ventricles fill, and vice versa.

Constitutive Description of Active Contractile Behavior of the Myocardium

Active contraction of the myocardium is commonly described following the approach developed by Guccione and McCulloch [32]. The time-varying elastance model is a modified from the standard Hill equation by an activation curve $C(t)$. The active fiber stress is:

$$T^a = T_{max} \frac{Ca_0^2}{Ca_0^2 + ECa_{50}^2} C(t). \qquad \textbf{Equation 1.9}$$

where T_{max} = 135.7 KPa is the isometric tension at maximum activation, Ca_0 = 4.35 μM is the peak calcium concentration. The calcium sensitivity, ECa_{50}, depends on the sarcomere length, l, as:

$$ECa_{50} = \frac{(Ca_0^2)_{max}}{\sqrt{\exp[B(l - l_0)] - 1}}$$ **Equation 1.10**

where B governs the shape of the tension-sarcomere length relation, and l_0 is the sarcomere length at which no active contraction develops.

The total stress is the sum of the active and passive stresses: $\boldsymbol{\sigma} = \boldsymbol{\sigma}^p + \boldsymbol{\sigma}^a$, where $\boldsymbol{\sigma}^a = T^a \boldsymbol{a} \otimes \boldsymbol{a}$, and a is the fiber orientation [10].

The Cardiac Cycle

The cardiac cycle can be separated into two main parts: the diastolic (filling) phase and the systolic (contracting) phase [Figure 1-3]. During diastole, the LV stretches and the volume increases to the end-diastolic volume (EDV). Diastolic pressures are relatively low: normal end-diastolic pressures (EDP) are about 10 mmHg in both humans and most animal models. During systole, LV pressure increases. When the LV pressure reaches the pressure within the ascending aorta, the aortic value opens and the blood is ejected, and LV volume decreases to the end-systolic volume (ESV). The end-systolic pressure (ESP) has a large role in determining the workload on the LV.

Another graphical representation that is often used to describe LV mechanics is the *Pressure-Volume Loop (PV Loop)* [Figure 1-3], which plots the LV lumen pressure versus the LV cavity volume over the cardiac cycle. Beginning in the lower left corner, diastolic filling takes place in section (a). Then isovolumetric contraction (b) takes place: the pressure increases while the volume remains the same. Once the pressure in the LV reaches the aortic pressure, the

aortic value opens, which is then followed by the ejection phase (c) where the blood is ejected into the systemic circulation. Isovolumetric relaxation then takes place (d) and the pressure drops. The Stroke Volume (SV) is the volume of blood ejected on each beat, and is calculated as the difference between the maximum (diastole) and minimum (systole) LV cavity volumes. A commonly used measure of overall cardiac function is the Ejection Fraction (EF), which represents the fraction of blood in the LV that is ejected on each beat, and is calculated as the SV divided by the end diastolic volume. The area inside the PV loop represents the energy expended on each beat, and is known as the Stroke Work (SW). Note that increases in ESP will increase the SW.

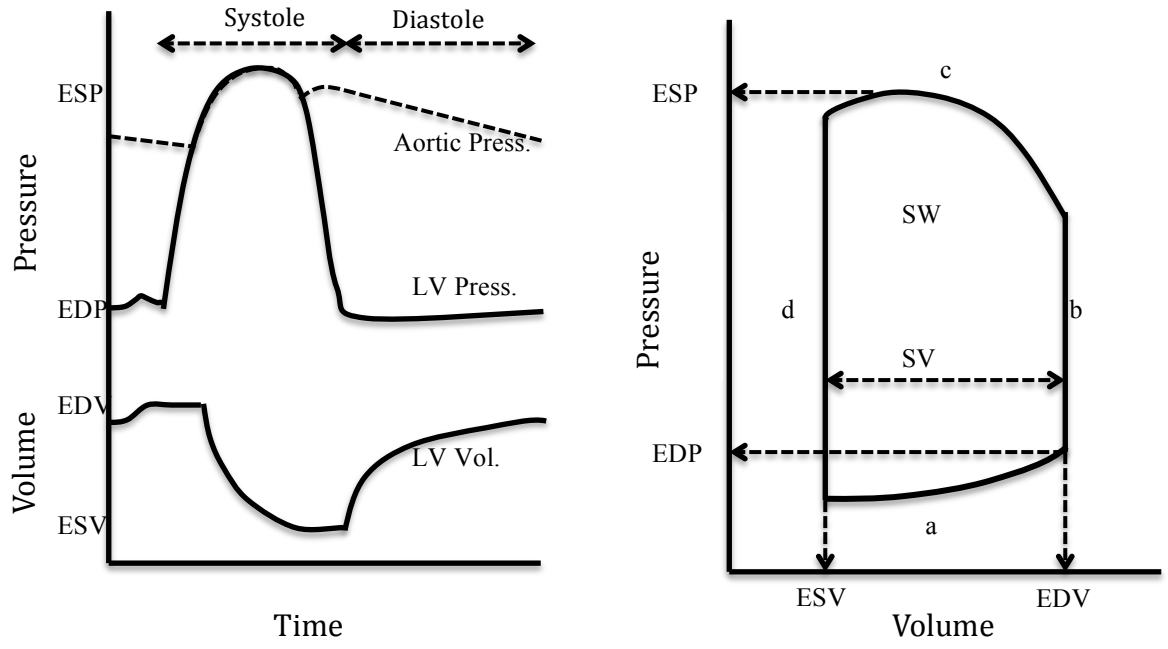

Figure 1-3: (a) Schematic illustrating the general shape of Pressure and Volume curves the LV over the Cardiac Cycle, and (b) Schematic illustrating the general shape of an LV Pressure-Volume Loop

Blood Pressures

The pressure that the left ventricle must exceed in order to pump blood to the body, which is the pressure within the ascending aorta, is known as the ventricular *afterload*, and it is affected by the mechanical properties of the vasculature. Blood pressures vary spatially throughout the vasculature, and very temporally over the cardiac cycle [33].

In the aorta, large pulses of pressure occur with each left ventricular systolic contraction. The aorta and the large arteries have a high degree of elasticity, which serves to regularize the (pulsatile) flow velocity downstream. The elastance of the aorta and large arteries is due to a layer of elastic tissue in between the inner endothelium and the smooth muscle layer [22]. Arterial elastance tends to decrease with age, and this can increase the afterload (and thus the systolic blood pressure) in the elderly. The small arteries, called arterioles, are responsible for what is known as peripheral resistance, which is responsible for a decrease in flow velocity just prior to the capillaries. This is important because ultimately, the purpose of the systemic circulation is to make a steady and slow flow of blood through the capillaries so that diffusion of oxygen and nutrients can take place throughout the tissues of the body. The peripheral resistance of the arterioles is due to the tiny diameter of these vessels relative to the size of blood cells, and because these vessels have a significant layer of vasoactive smooth muscle, leading to frictional resistance to the movement of the blood. Increased peripheral resistance will increase the ventricular afterload.

While the actual blood pressures vary significantly depending on the location in the vasculature and the time in the cardiac cycle, the term *Blood Pressure* in the clinical setting typically refers to the maximum and minimum pressures (over the cardiac cycle) that occur in

the brachial artery, as measured with a sphygmomanometer. This measure is used in the clinical setting to diagnose hypertension.

1.4: Hypertension and Hypertrophy

Definition of Hypertension

Hypertension, or high blood pressure, is defined by the JNC7 [34] according to Table 1-1. This defines categories of Prehypertension, Stage I Hypertension, and Stage II Hypertension. The ESH-ESC Guidelines (2007) {Mancia, 2007 #577} additionally defines a third stage (Stage III Hypertension) for people with systolic blood pressure exceeding 179 mmHg or a diastolic pressure over 109 mmHg.

Recent estimates suggest that hypertension affects over 30% of the adult population in the United States [1], and as much as 1 billion people worldwide [34]. Untreated long-term hypertension is one of the primary risk factors for stroke, heart attack, heart failure, aneurisms, peripheral artery disease, chronic kidney disease, and other serious diseases. It is a major economic burden on the United States health care system with an estimated annual direct and indirect cost of $50.6 billion [1]. While both awareness and treatment of hypertension have increased in recent decades, the total deaths have actually increased. From 1998 to 2008, the death rate caused by HBP increased 20.2%, and the actual number of deaths rose 49.7% [1]. In fact, high blood pressure is the number one risk factor for death throughout the world [34].

Table 1-1: Definition of Hypertension [JNC-7]

SBP/DBP	JNC 7 Category
<120/80	Normal
120-139/80-89	Prehypertension
140-159/90-99	Stage I Hypertension
>160/100	Stage II Hypertension

High blood pressure is termed essential or primary hypertension when there is no direct disease-related cause, and accounts for 95% of all hypertension cases [35]. Family history has been shown to play a role [35] in the development of hypertension, however, other factors are known to increase blood pressure including obesity, insulin resistance, high alcohol intake, high salt intake, and aging.

The serious health risks of long term hypertension are largely due to the progressive structural changes in the arteries and the left ventricle caused by the high pressure loading. Hypertensive heart disease first leads to left ventricular hypertrophy, and then eventually leads to impaired contractility [36]. Numerous studies show that increased LV mass predicts cardiovascular events [37]. The remodeling that takes place in hypertensive hypertrophy has been shown to increase the risk of cardiac ischemia [38] and ventricular arrhythmia [39].

Animal Models for Studying Hypertension

Hypertensive cardiac remodeling is a long term process that occurs over the lifespan (and can take decades in humans), it is convenient for research purposes to study this process in animal models with a relatively short lifespan. The spontaneously hypertensive rat (SHR) is a

genetically derived animal model that has been used to study hypertension and hypertrophy for decades because it closely mimics the physiological and geometric changes observed in humans [40], and its two-year lifespan makes it possible to study the effects of hypertension over the animal's entire life. Typically, the SHR is compared to a normotensive rat such as the Wistar-Kyoto (WKY). Throughout this section, experimental results for both human and animal model studies will be used to describe the general characteristics of hypertension and hypertrophy.

Blood pressure is very similar in most animal models and humans. Bing [41] measured blood pressure in SHR and WKY subjects at approximately 20 months of age (old age) and found that the mean systolic BP was 200 for the SHR with normal EF, 191 for the SHR with failing EF, and 125 for the WKY. These BP values are similar to what has been observed in humans [1, 42]. If one were to categorize these rats according to the JNC definition of hypertension (for humans) in Table 1-1, the WKY would fall just above normal, in the prehypertension category, while the SHR subjects would fall into the category for Stage II hypertension.

Changes in Ventricular Structure

Over time, the excessive workload due to hypertension causes changes in the global ventricular geometry, the size of myocytes, the proportion of interstitial collagen, and the myofiber organization.

Hypertrophic Growth

Hypertensive (pressure overload) hypertrophy is known as *concentric* hypertrophy, and is characterized by an increase in LV wall thickness while LV cavity volume remains normal. This is in contrast to another type of hypertrophy, called *eccentric* hypertrophy, which is characterized by an increase in LV cavity volume and typically occurs in response to sustained volume overload [36].

Global changes in LV geometry have been documented in both human studies and animal models of hypertension. Hypertrophy in the SHR was documented in experimental studies as early as the 1970s. Pfeffer et al. [43] measured the maximum external horizontal circumference of the left ventricle in rats at 90 weeks of age (near the end of the lifespan), and found mean values of 41 mm for SHR, compared to 35 mm for WKY, indicating hypertrophy in the SHR. This study also measured LV wall thickness at 90 weeks of age, and found values of approximately 4 mm for the WKY, and 5 mm for the SHR, indicating hypertrophy in the SHR. Tissue samples indicated that the diameter of the individual muscle fibers was about 15 micrometers in the WKY, and about 22 micrometers in the SHR. These results show that pressure overload hypertrophy is associated with increased muscle fiber thickness, which accounts for the overall thickening of the LV wall, and is consistent with the definition of concentric hypertrophy. Numerous human studies have quantified varying degrees of hypertensive hypertrophy using medical imaging [44]. Echocardiography has been used to determine LV end-diastolic wall thickness to be 12.8 +/- 1.6 mm in hypertensive patients compared with 8.5 +/- 1.1 mm for healthy volunteers [45]. Ventricular wall thicknesses of up to 20 mm have been observed [46] in severe cases of pressure induced hypertrophy.

Fibrosis

Fibrosis (additional interstitial collagen) has been observed in samples of hypertrophied myocardial tissue, and is thought to be an adaptive response to increased pressure loads. In the SHR study by Pfeffer et al. [43], fibrosis was measured at 90 weeks using a point counting technique. The SHR (55 points) had more than double the fibrosis of the WKY (24 points) [43]. In a study of the same SHR subjects that will be described in Chapter 2, trichrome stained collagen slides show noticeably more collagen (blue) in the SHR than in the WKY [Figure 1-4]. In a study on human hearts, fibrosis was measured as a percentage area of a tissue sample. The percentage area (mean (SD)) of fibrosis in the left ventricular wall in the hypertensive hearts (2.6 (1.5)%) was more than twice that of normal hearts (1.1 (0.5)%). This shows that the comparisons of fibrosis between SHR and WKY are very similar to comparisons between hypertensive and normal human hearts. The doubling of fibrosis in hypertension compared to normal suggests that high blood pressures may trigger a response that may serve to stiffen the LV wall.

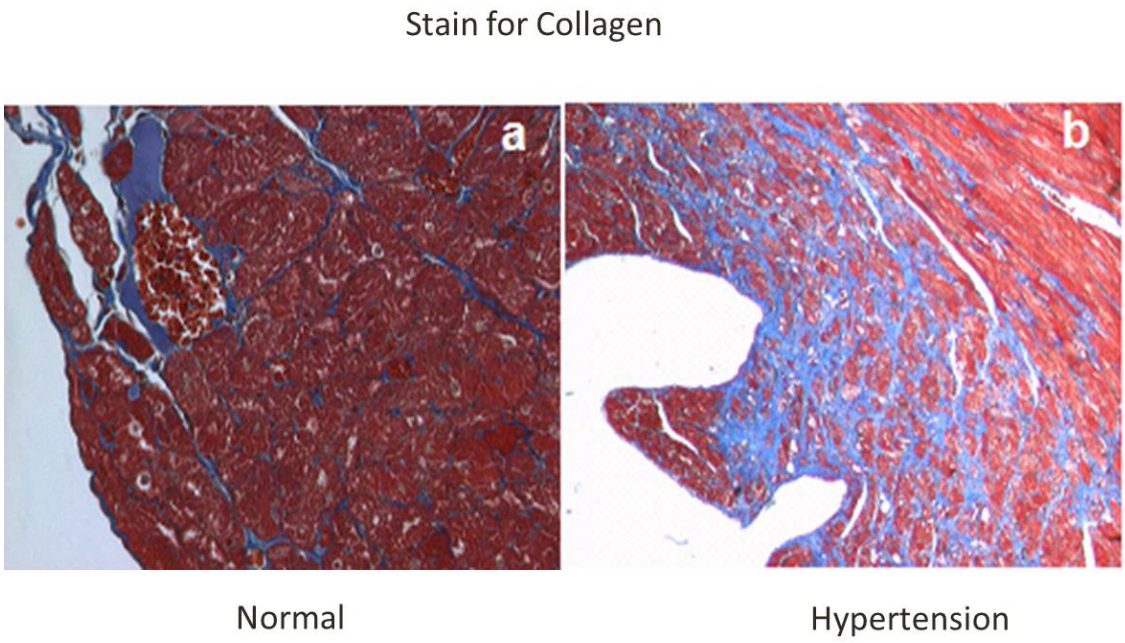

Figure 1-4: Trichrome stained collagen in the normal (a) and SHR (b) myocardium

In an SHR study by Bing [41], a measure of stiffness of tissue samples was tested in addition to histological measurements of interstitium. The mean percent interstitium of tissue samples was 8% for the WKY, 9.5% for the non-failing SHR subjects, and 20.1% for the failing SHR subjects. The central segment stiffness [47] constant was 40 for the WKY, 42 for the non-failing SHR subjects, and 96 for the failing SHR subjects. This suggests that the increase in interstitium in the failing SHR is associated with an approximately proportional increase in tissue stiffness as quantified by this experiment. Another study by Norton [48] found that increased stiffness in the SHR myocardium was actually more related to enhanced cross-linking of the collagen rather than an increase in total collagen alone.

It was also shown by Tanaka [49] that fibrosis in the ventricular septum and in the left ventricular free wall were similar, indicating a uniform pattern of fibrosis throughout the LV.

Furthermore, the percentage area of fibrosis correlated with heart weight, indicating that the degree of fibrosis was related to the degree of hypertrophy.

Fiber Disarray

Myocardial disarray is a fiber distribution that deviates away from the tightly organized, parallel alignment of myocardial fibers that characterizes the normal myocardium. It is this normal, transmural parallel organization of fibers and sheets that results in the twisting contraction of the left ventricle (LV) that is a hallmark of normal LV contraction. In contrast, myofiber disarray shows local as well as global deviations from this normal pattern with fibers having random directionality compared with the surrounding myofibers bundles. A study by Tanaka found the percent area percent area of fibrosis in tissues with and without disarray were significantly different (p less than 0.01) among hypertensive hearts (6.6 +/- 3.6 and 2.5 +/- 1.4%) compared with normal hearts (2.8 +/- 1.2 and 1.0 +/- 0.4%)[50]. In other words, the hypertensive hearts had higher levels of both fibrosis and disarray. And interestingly, the percent area of fibrosis in tissue with disarray was about 2.5 times greater than in that without disarray for both normal and hypertensive hearts, indicating that fibrosis and disarray may develop simultaneously as part of the same remodeling process of hypertrophy.

Changes in Ventricular Function

SV and EF

Measurements of Stoke Volume (SV) and Ejection Fraction (EF) have been used to quantify changes in function in the development of hypertensive disease. SV should stay

relatively constant over the lifespan in order to meet the demands of a consistent cardiac output. Pfeffer et al compared SV measurements in the SHR and WKY controls. SV for the SHR was 0.162ml at 25 weeks, and 0.134 at 90 weeks [43]. SV for the WKY was 0.207 at 25 weeks and 0.221ml at 90 weeks. This indicates that the WKY had slightly higher SV than the SHR, and that this was maintained throughout the lifespan of the animals. This result supports the notion that SV tends to be maintained over the lifespan regardless of the degree of hypertrophy.

EF is commonly used as a measure of cardiac function, and is typically measured using echocardiography. Various groups have attempted to define a cut-off value below which one might consider the EF to be indicative of heart failure. This is usually in the range of $EF < 0.45$ or $EF < 0.50$ [51-53] for both humans and the SHR. Experimental data of EF for the SHR was measured in the study by Bing et al [41]. Mean EF was 73% in the non-failing SHR, and 45% in the failing SHR. These are typical values for "normal" versus "failing" EF.

Strain

Recently there has been interest in developing methods for measuring image-based strains in cardiac tissue as a means of assessing function. More detailed information about the changes in LV function can be obtained by assessment of local tissue elongation and contraction rather than simple global volume data. One method that has been used to obtain strain measures of LV deformation is known as speckle tracking. In one such study [54], a relationship was found between LV mass index (a measure of hypertrophy) and strain (measured from end-diastole to end-systole), indicating that hypertrophic remodeling impaired systolic function. This was the case even in subjects that had a normal EF, which highlights the value of local tissue strain measures over global volume measures for assessing function. However, this method does

not give the full three-dimensional strain field. Instead, two-dimensional strains are calculated, and the values reported are scalar-valued stretch values in the radial, circumferential and longitudinal directions. Furthermore, this method does not take account of the fiber structure of the LV so it is not possible to obtain strain values relative to the fiber orientation. Measuring strain in the fiber direction is important for understanding the stretch imposed on the myofibers, which is a candidate stimulus for cardiac hypertrophy [55].

Tagged MRI has been used to calculate of 2D and 3D strains, and has been used to quantify the changes in LV function in hypertensive hypertrophy. In a study of 30 hypertensive human subjects and 10 healthy controls, circumferential and longitudinal strain was reduced in hypertensive subjects. Mean circumferential shortening at midwall was 20 +/- 6% for the hypertensive subjects versus 30 +/- 6% for the controls. Mean longitudinal shortening at mid-ventricle was 14 +/- 8% for the hypertensive subjects versus 18 +/- 3% for the controls [56]. The primary limitations of this method are that: (1) the strain distributions are affected by the relatively course tag spacing, and (2) there is no information available about the fiber orientation.

A very recent development in image-based strain measurement is known as slice followed cine displacement encoded imaging with stimulated echoes (DENSE). This enables the calculation of 3D myocardial tissue strains. [57] However, this method is very new and has yet to be applied to the study of altered mechanics in hypertensive hypertrophy.

In order to characterize the altered mechanics of hypertensive hypertrophy in detail, it will be necessary to use methods which: (1) are based on commonly used imaging modalities so that data is available, (2) measure 3D strains, and (3) take account of the fiber orientation such that the fiber-direction strains can be obtained. This is the primary motivation for the choice of methodolodgy utilized in Chapter 2.

Physiological Changes in Hypertensive Hypertrophy

Perfusion

In hypertensive hypertrophy, there is increased myocardial cell diameter, but no proportional increase in capillary vessels has been found. The hypertrophy increases the required diffusion distance of nutrients [58]. This may contribute to the observed incidence of myocardial ischemia [38] in hypertensive hypertrophy. However, microPET perfusion imaging of SHRs has shown that there is no apparent correlation between perfusion and tissue function as quantified by strain [Chapter 2].

Metabolism

Energy use and metabolism are also altered in hypertensive hypertrophy. While the primary energy substrate for normal hearts is fatty acids, studies have shown that in hypertensive hypertrophy, there is decreased use of fatty acids and increased use of glucose (which is less efficient). This has been documented in both the SHR and in humans [13] [59]. In a longitudinal PET imaging study of the SHR, at 20 months of age, the SHR had higher glucose utilization than the WKY [13]. It has also been shown that the degree of hypertrophy correlates with the degree of decrease in fatty acid oxidation in humans [59], indicating that the process of the development of hypertrophic structural changes coincide with the change in energy substrate utilization.

1.5: Typical Modeling Assumptions in Cardiac FEA

Finite Element simulations of left ventricular solid mechanics enable the calculation of stresses and strains in the myocardial wall. This has been used extensively in research related to the development of treatments [60-63], as well as in the study of the basic science of LV function in both diseased [64-66] and normal [2-4] conditions. This section briefly describes some typical modeling assumptions in FE modeling of the LV, including representations of geometry, boundary conditions, and loads. (Constitutive relations were discussed on Section 1.3.)

Geometry and Mesh

Some researchers have approximated the LV shape with a prolate-spheroidal geometry [67, 68], and have obtained realistic results. However, many researchers have focused on developing patient-specific models [60], in which the LV geometry for a specific individual is obtained through medical imaging modalities such as CT, MRI, and perfusion PET. In this case, the LV geometry may be manually segmented from the image slices [15], or an automatic segmentation software tool may be used [69]. Then, the segmented contours are used to define the LV mesh using a finite element pre-processor or meshing software. The LV geometry is usually discretized with 8-node hexahedral “brick” elements [15, 60, 67, 68].

Boundary Conditions and Loads

The heart is attached at the base via the large arteries and veins as well as some connective tissue. Some z-direction (vertical, up and down) motion of the base has been observed in medical imaging [70] during the cardiac cycle. The boundary condition prescribed at the base can be defined with some z-direction prescribed base displacement, if the goal is to match the movement observed in images. Or, if the goal is only to determine the stresses and strains in response to loads, then this motion may be neglected.

The pressure on the outside (epicardial) surface of the heart (inside the pericardium) is difficult to measure, but is believed to be relatively low, and is often neglected in cardiac FE analyses [2-4]. The pressure on the inside (endocardial) surface of the LV has been documented with catheter measurements [41] over the cardiac cycle, and is the primary source of loading on the LV.

Quasi-static vs. Dynamic Modeling

The LV mechanics over the cardiac cycle are typically modeled as quasi-static, rather than dynamic, because the inertial component has been found to change the stresses by less than 1% [71].

Neglect of Residual Stresses

The adult LV has a natural distribution of residual stresses that are formed in the process of organ morphogenesis. These stresses have been quantified in numerous experimental studies through what is known as the *opening angle* [72]. Briefly, a mid-ventricle slice (or ring) of the LV is removed and its geometry is measured. Then, a cut is made in the radial direction, such

that the ring of tissue can open. The angle it opens to is called the opening angle and is a measure of the overall residual stress within the LV wall. Experiments in which the ring was also sliced circumferentially show that there is a great degree of through-wall variation in residual stress, with the endocardial side having a much greater opening angle [72]. While these stresses are known to exist in the LV, the residual stresses are typically neglected in finite element analysis studies of the left ventricular cardiac cycle mechanics [15, 60, 67, 68].

1.6: Modeling of Growth and Remodeling

Early Attempts in Growth Modeling

Growth is a fundamental aspect of biological material that is particularly important in cardiac biomechanics. Let us begin with a history of the development of material laws that describe load-induced growth. The first work describing a relationship between the structure (size and shape) of biological tissues and their mechanical environment was Wolff's Law [73], published in 1892, and described a relationship between bone growth and the loads placed on it. However, because it was based on linearized theory of elasticity it cannot be applied to soft tissues, which undergo large deformations. Nearly a century later, the continuum theory for describing finite growth of soft tissues finally began to take shape. Skalak et al. [74] was the first to develop an approach to describing volumetric and surface growth of soft biological tissues in which the focus was placed on the kinematics involved in growth. Skalak identified measures of

finite volumetric growth, as well as rates and velocities that define surface growth. It is also worth mentioning that Skalak introduced the notion that differential growth need not be compatible, and that this would result in residual stresses.

Constitutive Law for Load-Based Finite Growth of Soft Tissues

Building on the work of Skalak, one of the most influential papers in the development of continuum mechanics theories describing load-based finite growth, was that of Rodriguez et al [75], in which a formal framework to describe kinematic growth was proposed. They described the kinematics of volumetric growth as a sequence of two deformations. (The multiplicative decomposition of the deformation gradient was originally developed in the context of plasticity theory [76] based on the notion of an intermediate stress-free configuration). The first deformation, $\boldsymbol{F_g}$, can be thought of as follows: imagine the material as many individual pieces, and allow each of these pieces undergo volumetric growth independently. The second part of the deformation, $\boldsymbol{F_e}$, serves to fit the pieces back together into a contiguous body, which requires internal loads, and as a result the grown configuration is residually stressed.

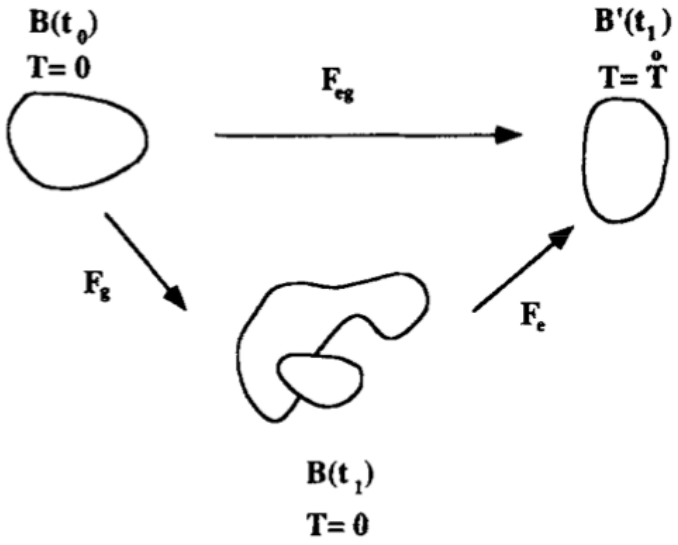

Figure 1-5: Kinematic Representation of the decomposition of the deformation gradient. The undeformed state is B(t_0). The intermediate configuration is B(t_1). The final configuration is B'(t_1).

Figure 1-5 (taken directly from Rodriguez 1994) shows three states in finite growth: (a) the original zero-stress reference state $\boldsymbol{B(t_0)}$; (b) the grown zero-stress $\boldsymbol{B(t_1)}$; (c) the observed intact grown state $\boldsymbol{B'(t_1)}$. The growth deformation gradient $\boldsymbol{F_g}$ maps $\boldsymbol{B(t_0)}$ to $\boldsymbol{B(t_1)}$. $\boldsymbol{F_g}$may not be compatible so $\boldsymbol{B(t_1)}$ is shown as a collection of discontinuous material. In order to make the overall growth deformation compatible, the elastic deformation $\boldsymbol{F_e}$ is required to map $\boldsymbol{B(t_1)}$ to the intact grown state $\boldsymbol{B'(t_1)}$. $\boldsymbol{F_e}$ gives rise to the residual stress $\boldsymbol{T°}$. The overall growth deformation is then the composition given by

$$\boldsymbol{F_{eg} = F_e F_g}. \qquad \textbf{Equation 1.11}$$

Since $\boldsymbol{F_e}$ alone is responsible for the residual stress, they suggest that the residual stresses can be calculated as $\boldsymbol{t = -pI + F_e \cdot \frac{\partial W}{\partial E_e} \cdot F_e^T}$. Neither $\boldsymbol{F_e}$ nor $\boldsymbol{F_g}$ are directly measureable since they involve a fictitious state. However, the total deformation $\boldsymbol{F_{eg}}$ can be measured, and $\boldsymbol{F_e = F_{eg} \cdot}$

$\boldsymbol{F_g^{-1}}$. With $\boldsymbol{F_g = R_g \cdot U_g}$ in general, they assume that $\boldsymbol{R_g = I}$ since all of the rigid body motion can be absorbed into $\boldsymbol{F_e}$. The authors then suggest that finding $\boldsymbol{F_g}$ essentially boils down to finding $\boldsymbol{U_g}$ based on any postulated evolution law for $\frac{d\boldsymbol{U_g}}{dt}$ which can, for example, be a function of stress.

This type of constitutive growth law has been applied to various problems in cardiovascular growth and remodeling including arterial wall growth [77-79], cardiac morphogenesis [80, 81], and cardiac hypertrophy [61, 81-84]. For a broader treatment of the study of theoretical growth laws in biomechanics see references [84, 85].

Computational Implementations of Constitutive Growth Law for Modeling Cardiac Hypertrophy

In recent years, researchers have worked on developing specific forms of the constitutive law for $\boldsymbol{F_g}$ in the context of cardiac hypertrophy. The growth tensor $\boldsymbol{F_g}$ can be been strain-driven, stress-driven, and can be either isotropic or anisotropic. Rauch et al characterized $\boldsymbol{F_g}$ for three cases: athlete's heart, cardiac dilation, and cardiac wall thickening [86, 87]. $\boldsymbol{F_g}$ was defined as isotropic stress-driven for athlete's heart, myofiber-direction strain-driven for cardiac dilation, and cross-fiber stress-driven for pressure-overload hypertrophy. These growth tensors succeed in identifying potentially important growth stimuli and define the anisotropy of growth in ways that match the observed disease. However, the computational implementation lacked a realistic material model (linear elastic, no active contraction). Kerckhoffs et al. [88] showed that single a strain-based growth law (again based on passive filling pressures only) was able to cause two

different growth patterns, concentric and eccentric hypertrophy, in response to two different hemodynamic disease cases, aortic stenosis and mitral valve regurgitation, when the model was coupled to a circulatory model.

In addition to the definition of $\boldsymbol{F_g}$, some researchers have also considered the computational framework for implementing the constitutive growth modeling approach. For example, it is known that during cardiac growth, material turnover occurs simultaneously with growth. Therefore, a continually updated reference configuration may be of interest. Kroon et al [82] used an isotropic strain-based growth law to do a comparison of an unchanging vs. an updated reference configuration for hypertrophic growth in response to passive filling pressures. The updated reference configuration model took more iterations of growth achieve the same amount of wall thickening and had a slightly different through-wall distribution of growth.

Limitations of the Constitutive Growth Law for Modeling Cardiac Hypertrophy and Future Directions in Cardiac Growth Modeling

While many others have implemented the constitutive modeling approach in the context of modeling cardiac hypertrophy [61, 81-84], there are some notable limitations. First, in this approach, only one state of mechanics can be used to drive growth. This may be appropriate for some applications in which it is known that the growth is only affected by a single state of stress or strain. The left ventricle, however, experiences a wide range of different states of stress and strain over the cardiac cycle, and the mechanisms of growth are not well characterized with respect to the actual mechanical growth stimuli. It is likely that multiple aspects of the mechanics influence the development of hypertrophy. For example, end-diastolic fiber strain, peak systolic

fiber stress, peak systolic cross-fiber shear strain, strain rates, and other mechanical measures may be candidate stimuli worth evaluating. The development of mechanical models which can incorporate mechanical information from throughout the cardiac cycle would enable the testing of competing hypotheses regarding mechanical growth stimuli. This topic will be addressed in Chapter 3.

Eventually, the future goal of this area of research is to incorporate growth and remodeling material behaviors into patient-specific predictive models. This would make it possible to predict of outcomes of competing treatment strategies.

Chapter 2. Image-Based Strain Measurements Document Progression of Hypertensive LV Dysfunction Over SHR Lifespan

2.1: Introduction

Long term hypertension causes progressive changes in LV structure and function. Thickening and stiffening of the myocardium has been shown to lead to diastolic dysfunction [89], while the increased energy demand has been shown to lead to systolic dysfunction [90]. While hypertrophic adaptations can maintain LV function in the short term, long-standing hypertension has been shown to lead to heart failure (HF) [51]. A detailed understanding of the long-term changes in both diastolic and systolic LV function that occur in hypertension is of great importance to the continued development of mechanics-based treatment approaches.

The spontaneously hypertensive rat (SHR) is a genetically derived animal model that has been used to study hypertension induced hypertrophy and heart failure for decades. Its relatively short two-year lifespan makes it possible to study the effects of hypertension over the animal's entire life. This animal model closely mimics the physiological and geometric changes associated with hypertension and hypertrophy in man [40]. Detailed temporal studies have documented the changes in the LV geometry [41, 43], associated changes in histology [48, 91], electromechanical function [41, 47], and contractility [92], as well as changes in molecular function of the myocytes [93]. Changes in hemodynamic measures such as stroke volume (SV) and ejection fractions (EF) are also well documented in the SHR [41, 43], and have been studied

in the context of the time course of the development of hypertrophy. However, detailed studies of changes in regional LV function have yet to be carried out.

The use of mechanical strain, a measure of local tissue deformation, overcomes some inherent limitations of global measures such as SV and EF for quantifying myocardial function. First, if the deformation of the LV is determined relative to an in-vivo reference configuration, then the strains calculated during diastole and systole will be separate measures. Differentiating between changes in diastolic function and changes in systolic function is valuable because changes in systolic and diastolic function may have different underlying mechanisms and different time courses providing new insights into the disease progression. A second advantage of strain as a measure of LV function is that it is a local measure of deformation. In contrast to SV and EF, which measure the overall blood volume output, strain quantifies the mechanical deformation behavior throughout the LV. This feature is necessary to determine whether certain regions of the LV are performing differently than others. Additionally, local strain can be directly related to other aspects of local disease physiology, such myocardial tissue perfusion, which could give insight into the physiology of the disease progression. Strain measurements in the LV of SHR subjects over their whole lifespan would contribute to the understanding of long-term hypertension-induced changes in LV function.

The full 3D strain field of the LV can be determined directly from clinical imaging data with an image registration technique known as Hyperelastic Warping (HW). In HW, the intensity differences between a template (reference) image data set and a target (deformed) image data set are used to deform a finite element (FE) representation of the LV depicted in the template image into alignment with the target image [16, 94]. The FE model definition can include realistic myocardial material properties as well as fiber directions [15, 70]. Regions within the image data

sets with large intensity gradients such as the epi- and endocardial walls, as well as inhomogeneities within the wall, act as fiducial points for the image registration process. Because this is a non-invasive image-based approach, it is possible to track these changes over the whole lifespan of the SHR through repeated imaging. MicroPET imaging has been used to study cardiac geometry, perfusion, and metabolism in small animal models such as the SHR [95]. In previous work, HW has been successfully applied to: (1) SHR microPET images for evaluating LV strains at end-diastole referenced to end-systole [15], (2) human PET images for evaluating LV strains at end-systole referenced to end-diastole [16], (3) human cineMRI images for evaluating LV strains at end-systole referenced to end-diastole [70], (4) human cineMRI images for evaluating LV strains at end-diastole referenced to end-systole [15]. However, there is value in differentiating between systolic and diastolic strains. If gated image data is available, and an appropriate reference configuration image is chosen, HW can be used to determine the strain distributions at all gates in the cardiac cycle relative to the reference configuration image. Then, changes in diastolic and systolic deformation can be evaluated separately.

The objective of this study was to use HW image registration to determine LV strain distributions based on gated microPET images acquired of 7 SHR and 7 Wistar-Kyoto (WKY) controls at 4 points in time over their lifespans, and assess the long-term changes in myocardial function associated with hypertension in the SHR. In contrast to the previous work which only used the end-diastolic and end-systolic images [15], the novelty of the current study was to determine strains at all eight intermediate states over the cardiac cycle in the gated microPET images, with the strain distributions calculated relative to an image gate in early diastole which was defined as the reference configuration. This made it possible to study the time course of changes in diastolic and systolic deformation behavior separately. Making use of the microPET

image-based strain data, the current study also evaluated differences in regional deformation behavior of the LV as well as a possible relationship between perfusion and deformation. This type of longitudinal study of strain in the SHR animal model contributes toward a more complete understanding of the time course of changes in LV function that occur in response to long-term hypertension.

2.2: Methods

Animal Study Design

All imaging studies were performed in accordance with Institutional Animal Care and Use Committee (IACUC) approved protocols from both UCSF and Lawrence Berkeley National Laboratory. Seven male SHRs and seven male WKY normotensive rats were purchased from Charles River Laboratories (Wilmington, MA). Imaging began at approximately six months of age and the rats were imaged throughout their life cycle at separate time points corresponding to 6, 12, 15, 18 and 20 months of age. All rats were freely fed standard Purina rat chow and water. The rats were nocturnal and mostly ate at night. During each time point, every living rat was imaged using ^{18}F-flurodihydorotenol. The rats were not fasted for the studies.

MicroPET Imaging

Temporal changes in geometry and deformation were documented through multiple image acquisitions that were performed on the 7 SHR and 7 WKY over the 1.5-2 year lifespan of the animals. All imaging was performed using a microPET/CT scanner (InveonTM dedicated PET docked with CT in the multimodality platform, Siemens Medical Solutions, Malvern, PA). ^{18}F-flurodihydorotenol was used to document the wall motion as it is taken up equally well in the SHR as well as normal controls. Imaging issues at the 12-month acquisition resulted in unusable images at this time point. Data sets were named by subject number and imaging acquisition number. For example, "SHR2_5" refers to the data set from the fifth imaging acquisition of SHR2.

Prior to imaging, the animals were anesthetized with 2% isoflurane and oxygen. The animals were then placed in the microPET scanner with ECG electrodes attached for gating. The animals were injected via tail vein with 1-2 mCi of ^{18}F-flurodihydorotenol. Image acquisition began at the time of injection. The microPET/CT system was used to acquire dynamic ECG-gated list mode PET data over 60 minutes. After the PET acquisition, a separate CT scan was acquired with 120 projections of continuous rotation to cover 220° with an x-ray tube operated at 80 kVp, 0.5 mA, and 200 ms exposure time. Listmode data were histogrammed into 8 gates of the cardiac cycle summed over the total acquisition after the initial few minutes for the studies documenting the wall mechanics. For the dynamic and kinetic modeling studies, multiple gates of dynamic sequences, each of which consisted of complete tomographic projections, were histogrammed at 12 intervals of 5s, followed by 6 of 10s, 4 of 30s, 6 of 60s, and finally 10 intervals of 300s to give a total of 38 time frames over the 60 minutes. Dynamic sequences of 128×128×159 matrices of 0.776×0.776×0.796 mm^3 voxels were reconstructed using a 2D ordered-subsets expectation maximization [Fourier rebinning, OSEM (4 iteration and 16

subsets)] reconstruction algorithm supplied by the Inveon^{TM} imaging system software with attenuation correction using the x-ray CT images. No scatter correction was applied in this study.

Weight, LV Volume, and Wall Thickness Measurements

Body weights were measured just prior to image acquisition for all SHR and WKY subjects. The reconstructed ^{18}F-flurodihydorotenol images were used to determine LV lumen volumes at each gate of each image data set using Xeleris [96] software. Volume data was not available for one data set (SHR5_3) due to issues with the software. The LV cavity volume data were used to calculate Stroke Volume (SV) and Ejection Fraction (EF) as well as the reference configuration volume (RCV). The RCV is the LV cavity volume at the image gate corresponding to the reference configuration. The choice of image gate for the reference configuration is discussed in the following section. The RCV was tracked over the lifespan, and was used as a measure of unloaded LV lumen volume. Additionally, a comparison was made between RCV and EF for individual subjects' data (rather than averages). In order to document the extent of hypertrophy over time, a small number of SHR and WKY subjects were sacrificed following the series of ^{18}F-flurodihydorotenol microPET image acquisitions performed at 6, 15, 18, and 20 months in order to measure the weight of the heart. Heart weight measurements were also made of animals that died between image acquisitions. Hypertrophy was also documented in LV wall thickness (WT) measurements from the images. A mid-ventricle short axis slice of the reference configuration FE mesh was used to measure WT for all data sets. Means and standard deviations of Body weight, SV, EF, RCV, and WT were calculated for each image acquisition time for the SHR and the WKY groups.

Strain Measurement by Hyperelastic Warping Image Registration

Hyperelastic Warping

Hyperelastic Warping (HW) is an image registration technique that can provide estimates of the full 3D strain field of the LV directly from clinical imaging data. In HW, the intensity differences between a template (reference) image data set and a target (deformed) image data set are used to deform a finite element (FE) representation of the LV depicted in the template image into alignment with the LV depicted in the target image. The FE model can include realistic passive and contractile material properties of the myocardium, as well as realistic fiber directions. Regions in the image data sets with steep intensity gradients such as the epi- and endocardial walls, as well as inhomogeneities within the wall, act as fiducial points for the image registration process. Complete details of HW theory and its application to nuclear imaging can be found in the following publications [15, 16, 94, 97]. The HW algorithm is integrated into a specific version of the nonlinear FE code NIKE3D [98]. In the current project, the HW solution evolved over seven intermediate images from early diastole (approximate stress-free state) through end systole. The HW analysis was completed on a total of 33 image data sets.

Creation of Finite Element Models

An FE mesh was created for each image data set based on the geometry of the image gate approximately 1/3 of the way into diastole. This time point has been shown to be the closest to an unloaded state that occurs in-vivo [99]. This reference image data set was manually segmented, and the resulting closed contours were used to define the FE mesh using TrueGrid

preprocessing program [100]. Each FE mesh consisted of 15,448 nodes and 13,700 hexahedral (8-node solid) elements with B-bar (mean volumetric strain) formulation [98]. FE meshes for the following data sets were created using HW image registration to deform the mesh from the geometry of one LV to another [101].

The passive material behavior of the myocardium was represented with the transversely isotropic hyperelastic material model described in Section 1.3. A description of the constitutive model and its FE implementation can be found in Weiss et al. [102]. Realistic fiber inclination angles [103] of -60, -30, 0, 25, 50 degrees were defined transmurally for each model from the epicardium to the endocardium. Active fiber contraction was implemented following the approach used by Guccione and McCulloch [32] and further details can be found in Veress et al. [104].

Strain Data Analysis

The strain in the fiber direction (of the transversely isotropic material model) was chosen as the appropriate strain measure to report in this study because the goal was to assess the local elongation and shortening of the myocardial tissue during diastole and systole. The fiber strain was calculated as $\boldsymbol{a} \cdot \boldsymbol{E} \cdot \boldsymbol{a}$, where $\boldsymbol{a}$ is the fiber direction vector, and $\boldsymbol{E}$ is the Green-Lagrange strain tensor. The fiber strain was recorded at all of the nodes in the FE mesh at all intermediate states of the image registration process representing the image gates. To obtain a measure of overall deformation for the whole LV, the nodal fiber strain results were averaged over all of the nodes in the FE mesh. The average fiber strain in the LV mesh was calculated for each of the 7 target image gates in the cardiac cycle. The average fiber strains at end-diastole and end-systole are of particular interest.

To assess the differences in function between the SHR and WKY groups the means and standard deviations of the average end-diastolic fiber strain (EDFS) and average end-systolic fiber strain (ESFS) were calculated at each image acquisition point over the lifespan of the subjects. Statistical significance of differences in mean EDFS and ESFS between the SHR and WKY were assessed using the Student's t-test.

Regional fiber strains were also assessed for each image data set in order to determine whether functional changes occurred uniformly throughout the LV. Each model was divided into four regions (septal, anterior, lateral, posterior) using the attachment points of the right ventricle as anatomical landmarks. Average values of regional EDFS and ESFS were determined for each image acquisition time. The statistical significance of differences in EDFS and ESFS between the regions was assessed using the Student's t-test. The statistical significance of changes in EDFS and ESFS between acquisition times was also assessed using the Student's t-test.

Comparison of Strain Data with Other Measurements

EF, RCV, WT, EDFS, and ESFS were tabulated for each image data set in order to assess how changes in strain coincided with the associated changes in geometry and hemodynamics. Cutoff values were chosen for EF, RCV, WT, EDFS, and ESFS in order to separate the data sets into those representative of "normal" states vs. those representative of "diseased" states. The cutoff value for defining a low EF in the present work was taken to be $EF < 0.45$ based on: (1) the commonly used clinical definition of systolic HF in humans (the combination of various symptoms and an EF below some threshold value usually taken to be 0.40, 0.45, or 0.50 [20, 51, 105-109]), and (2) experimental studies of SHR and WKY rats in which the SHR subjects

experiencing clinically diagnosed heart failure had a mean EF of 0.45 [41]. Cutoff values for ESFS, EDFS, RCV, and WT, the SHR data were defined based on from the deviation from the normal average of the WKY data. Following the determination of the mean and standard deviation of all of the WKY data sets, the SHR data sets were given designations based on the number of standard deviations the values were from the mean of the WKY data. This is intended to give an indication of the degree of abnormality of the SHR data. The standard deviations were based on the assumption that SHR population data is normally distributed.

Perfusion was evaluated by fitting blood and tissue time activity curves generated from the dynamic reconstructed PET data to a single-compartment irreversible model. The fine binning in the ECG gating sequence was used to select the bins corresponding to the end diastolic phase of the cardiac cycle, thus minimizing cardiac motion, partial volume effects and spillover contamination. Once these gates were identified, a new dynamic series was formed by summing the counts in each identified diastolic bin. The new dynamic series of diastolic bins were then uploaded into Inveon^{TM} Research Workplace version 3.0 (Siemens Molecular Solutions, Malvern, PA) for tracer kinetic analysis. Myocardial time-activity curves (TACs) were generated by manually drawing a volume of interest (VOI) throughout the myocardium, with care taken to avoid the myocardial border. ^{18}F-flurodihydrorotenol images displayed substantial liver uptake in both rat models. To minimize spillover of radioactivity from the liver to the heart, the entire apex and mid-cavity septal wall were not included within the VOI for any ^{18}F-fluorodihydrorotenol images. Tissue VOIs were selected by placing voxels in 10 consecutive mid ventricle slices. Only slices containing myocardium in 360° were selected. To generate blood time-activity curves, small regions of interest (10-20 mm^3) were placed within the LV blood pool with care taken to avoid the boundary of myocardium and LV cavity. The diastolic

PET images allowed for clear definition of myocardium and blood pool regions well within anatomical boundaries, alleviating the need for partial volume correction over continuous slices. The flow extraction product K_1 for ^{18}F-fluorodihydrorotenol was calculated by fitting blood and tissue time activity curves to a single-compartment irreversible model using the Inveon^{TM} software. After the strain analysis, K_1 was correlated with ESFS and EDFS in order to determine whether perfusion of the LV was related to myocardial function.

2.3: Results

Weight, LV Volume, and Wall Thickness Measurements

The SHR and WKY subjects were similar in overall body size. At 6 and 12 months, average body weight for the SHR was within one standard deviation of the WKY values. At 15, 18, and 20 months, average body weight for the SHR and was lower than one standard deviation below the average WKY values but the differences were not statistically significant due to the small number of SHR subjects alive at the later time points [Table 2-1].

Table 2-1: Body weight (BW), Heart to body weight ratio (H/BW), Wall thickness (WT), Reference configuration volume (RCV), Stroke volume (SV), Ejection fraction (EF). Values are means +/- standard deviation. NS = not

	BW (g)	H/BW (mg/g)	WT (mm)	RCV (mm^3)	SV (mm^3)	EF
6 months						
WKY	367.9 +/- 23.0	4.58 +/- 1.74	5.3 +/- 0.2	208.6 +/- 33.9	337 +/- 68.2	0.68 +/- 0.05
SHR	382.3 +/- 13.2	4.40 +/- --	5.2 +/- 0.1	253.8 +/- 22.2	275 +/- 26.9	0.56 +/- 0.04
P value	NS	--	0.064	0.009	0.031	0.0001
12 months						
WKY	429.0 +/- 19.1	-- +/- --	-- +/- --	441.5 +/- 218.3	431 +/- 210.7	0.63 +/- 0.02
SHR	428.0 +/- 25.3	-- +/- --	5.6 +/- 0.7	222.1 +/- 23.0	309 +/- 53.6	0.60 +/- 0.04
P value	NS	--	--	NS	NS	NS
15 months						
WKY	466.7 +/- 31.9	3.42 +/- --	5.4 +/- 0.4	232 +/- 55.6	363 +/- 63.2	0.67 +/- 0.04
SHR	433.0 +/- 36.0	4.62 +/- 0.30	6.2 +/- 0.2	319.6 +/- 77.2	294 +/- 67.7	0.52 +/- 0.11
P value	NS	--	0.003	NS	NS	0.056
18 months						
WKY	481.6 +/- 37.1	4.78 +/- --	5.5 +/- 0.4	232.9 +/- 101.2	391 +/- 141.2	0.70 +/- 0.03
SHR	415.0 +/- 24.0	6.54 +/- 1.45	6.2 +/- 0.4	471.3 +/- 329.4	284 +/- 103.4	0.46 +/- 0.28
P value	NS	--	0.062	NS	NS	NS
20 months						
WKY	487.0 +/- 29.0	3.42 +/- 0.44	5.2 +/- 0.1	218 +/- 42.7	335 +/- 24.4	0.71 +/- 0.09
SHR	384.0 +/- --	8.60 +/- --	6.6 +/- --	465.9 +/- --	222 +/- --	0.34 +/- --
P value	NS	--	NS	NS	NS	NS

The SHR showed progressive development of LV hypertrophy. Average wall thickness values for the SHR were similar to the WKY controls at 6 months; however, by 15 months the SHR subjects had significantly increased wall thickness ($p=0.003$). Heart to body weight ratio in the SHR increased progressively between 15 and 20 months of age, while the WKY showed relatively unchanging heart to body weight ratio.

LV volume data showed that the average RCV and the average EF changed over time in the SHR, while average SV remained relatively constant over the lifespan for both SHR and WKY groups. The average SV was greater for the WKY than SHR at all time points, but the differences were not statistically significant. The average RCV for the SHR increased progressively from 12 to 20 months. Average EF for the SHR was close to that defined as low EF at 18 months, and was below the low EF cutoff at the 20-month time point. Decreases in EF observed in the aging SHR were proportional to increases in RCV. A comparison made between RCV and EF for individual subjects' data (rather than group averages) revealed a correlation between RCV and EF ($R^2 = 0.67$).

Strain Measurement by Hyperelastic Warping Image Registration

SHR strain results at 6 months of age appeared similar to that of WKY subjects, however, at all later time points the SHR subjects showed progressive decreases in observed diastolic and systolic strains compared with the WKY controls [Figure 2-1]. Significant differences between SHR and WKY average strain data were observed at the 18-month and 20 month image

acquisitions. At 18 months, there was a significant difference between the SHR and WKY in EDFS (p=0.009), and a somewhat significant difference in ESFS (p=0.068) [Table 2-2]. At 20 months there was a significant difference between the SHR and WKY groups in ESFS (p=0.05). More variability in both EDFS and ESDS was observed for the SHR compared with the WKY subjects.

Evaluating the change in SHR strains between the 6-month image acquisition and later time points revealed that statistically significant changes in EDFS occurred before statistically significant changes in ESFS. There was a significant decrease in EDFS between 6 and 18 months (p=0.004), and a significant decrease in ESFS between 6 and 20 months (p=0.032). Changes between other time points were not statistically significant.

The regional strain data for the WKY showed that the lateral region had the greatest mean EDFS values for the first three of the four time points, and the greatest mean ESFS values for all of the time points [Figure 2-2]. In all of those cases, the data for the lateral region was significantly ($p < 0.001$) greater than the other regions. At all of the time points, the difference between the highest and lowest regional strains was greater for ESFS than for EDFS, indicating more variation between regions in systole than in diastole.

The regional strain data for the SHR showed that the lateral region had the greatest mean EDFS values at all time points, and the greatest mean ESFS values for the first three of the four time points [Figure 2-2]. In all of those cases, the data for the lateral region was significantly ($p < 0.001$) greater than the other regions. At all time points, the difference between the highest and lowest regional strains was only slightly greater for ESFS than for EDFS (less difference than was observed in the WKY).

The SHR EDFS decreased significantly ($p < 0.001$) for all regions between 6 and 15 months, and again between 15 and 18 months. The SHR ESFS decreased significantly ($p < 0.001$) for all regions between 18 and 20 months. The largest difference between consecutive regional strains for the SHR occurred in the lateral region between 18 and 20 months. This indicated that heart failure in the SHR2 subject appeared to involve decreased systolic function in the lateral region of the LV.

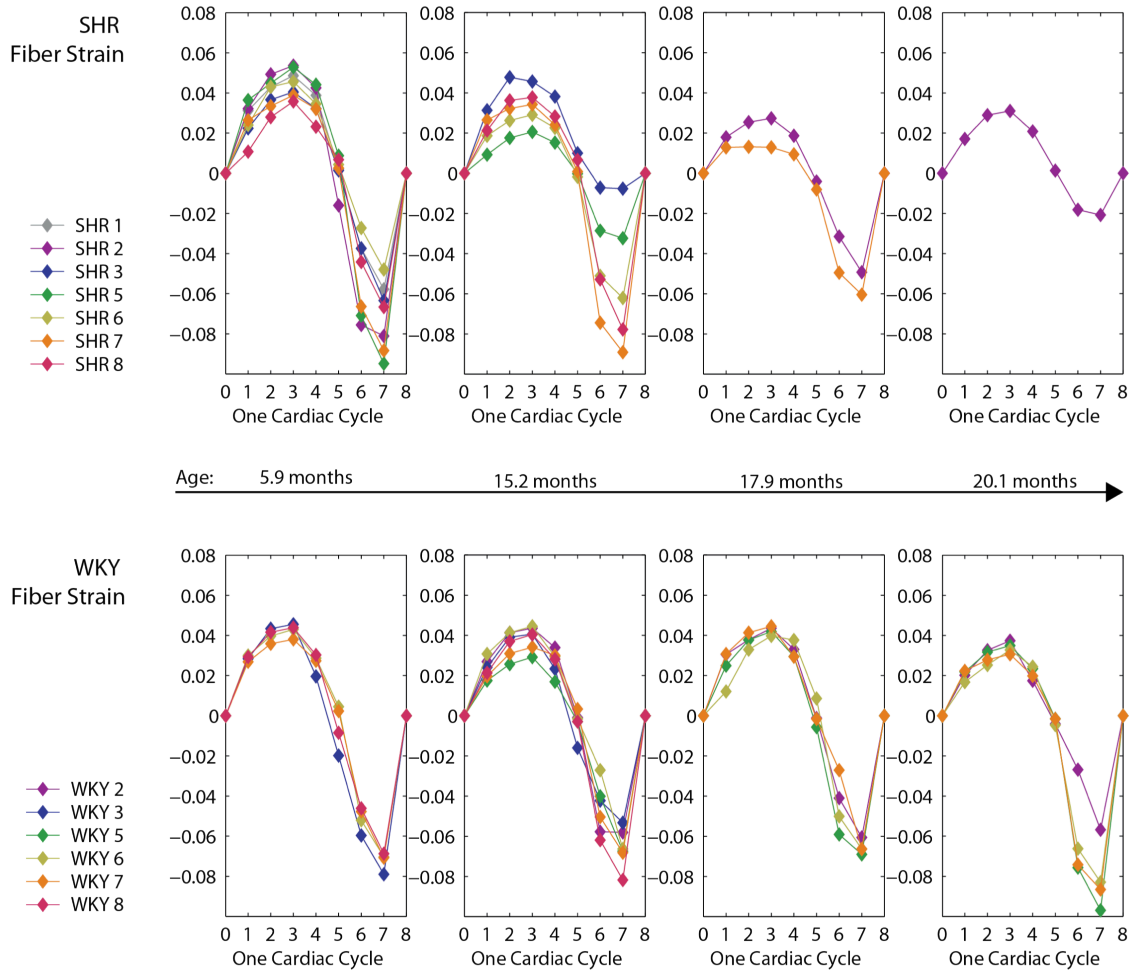

Figure 2-1: Cardiac cycle fiber strain beginning in early diastole. The top row shows SHR results and the lower row shows WKY results. Each of the plots from left to right depicts the results for a given age in the lifespan of the animals.

Table 2-2: Average end-diastolic fiber strain (EDFS) and end-systolic fiber strain (ESFS). Strain data is averaged for all datasets at a given image acquisition time. Values are means +/- standard deviation. NS = not significant.

	EDFS			ESFS		
6 months						
WKY	0.044	+/-	0.004	-0.070	+/-	0.007
SHR	0.045	+/-	0.007	-0.071	+/-	0.017
P value		NS			NS	
15 months						
WKY	0.039	+/-	0.006	-0.066	+/-	0.010
SHR	0.037	+/-	0.007	-0.059	+/-	0.036
P value	NS			NS		
18 months						
WKY	0.046	+/-	0.009	-0.067	+/-	0.003
SHR	0.020	+/-	0.010	-0.055	+/-	0.008
P value		0.009			0.068	
20 months						
WKY	0.034	+/-	0.003	-0.081	+/-	0.017
SHR	0.031	+/-	NA	-0.021	+/-	NA
P value		NS			0.051	

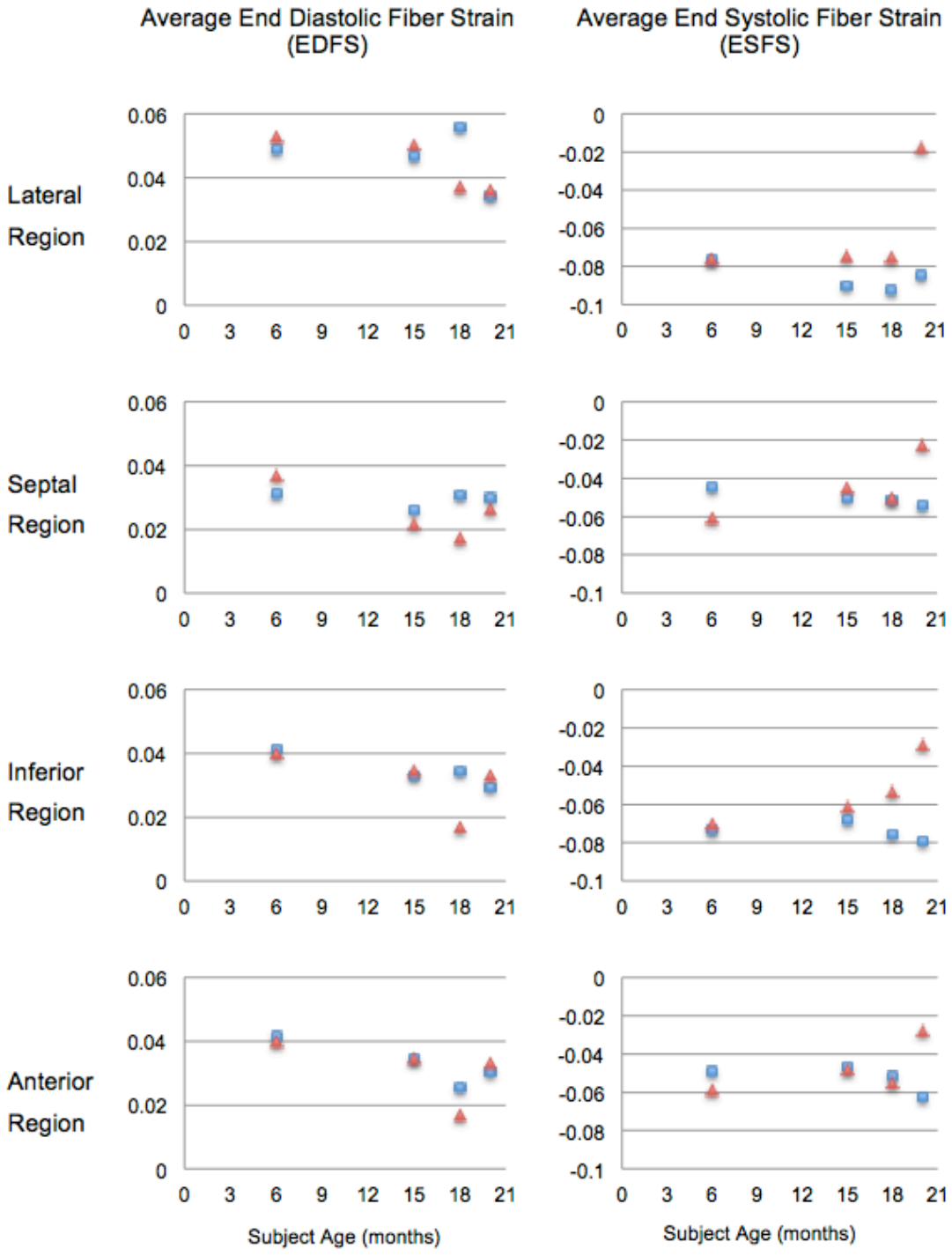

Figure 2-2: Plots of regional EDFS (Left Column) and regional ESFS (Right Column) vs. time of image acquisition (subject age). Rows correspond to Lateral, Septal, Inferior and Anterior Regions. Triangles represent SHR data and Squares represent WKY data. The lateral region tends to have the greatest values of both EDFS and ESFS. Changes in

SHR EDFS do not vary much by region. The decrease in SHR ESFS between 18 and 20 months was greater for the lateral region than the other regions.

Comparison of Strain Data with Other Measurements

The SHR showed differences from the WKY in all measurements of LV structure and function. The EF, RCV, WT, EDFS and ESFS results showed at least one SHR data set with a 3-sigma difference from normal function, with many 2-sigma and 1-sigma differences being observed [Table 2-3]. The primary results of the data comparison were:

(1) All image data sets with decreased diastolic strain showed LV hypertrophy, and the degree of decrease in EDFS appeared to match the degree of hypertrophy. The one case of 3-sigma decreased EDFS (SHR7_4) had 3-sigma increased WT, and all of the 2-sigma and 1-sigma decreased EDFS cases also had 1-sigma or 2-sigma increased WT.

(2) Decreased systolic strain coincided with LV hypertrophy only in severe cases. The two cases of 3-sigma decreased ESFS (SHR2_5 and SHR3_3) also had 3-sigma increased WT. However, only some of the 1-sigma and 2-sigma decreased ESFS cases showed hypertrophy. Furthermore, there were three cases of 2-sigma increased wall thickness (SHR6_3, SHR7_3, SHR8_3) that did not show decreased ESFS.

(3) Subjects with severely decreased systolic strain had low or nearly low EF, but not all subjects with low EF had decreased ESFS. The two data sets with 3-sigma decreased ESFS had low or nearly low EF. However, there was one data set with low EF (SHR6_3) that did not have a low ESFS.

(4) Hypertrophy was associated with other evidence of disease. Congestive heart failure with fluid in the lungs was observed at the time of death in the same three data sets (SHR3_3, SHR7_4, and SHR2_5) that had 3-sigma increased wall thickness.

(5) No correlation was found between the perfusion rate constant K_1 and ESFS (R^2 = 0.00016) [Figure 2-3]. There was also no correlation between K_1 and EDFS.

Table 2-3: Ejection Fraction (EF), Reference Configuration Volume (RCV), Wall Thickness (WT), End-Diastolic Fiber Strain (EDFS), End-Systolic Fiber Strain (ESFS), Presence of congestive heart failure (CHF) for the SHR data sets. Windicates EF<0.5. *, **, * indicate 1-sigma, 2-sigma, 3-sigma difference from "normal" values. + or – indicates presence or absence of CHF. Data sets are named by subject number and acquisition number, e.g. SHR2_5 is the data set from the fifth acquisition of SHR2.**

	EF	RCV (mm^3)	WT (mm)	EDFS	ESFS	CHF
SHR1_1	0.58	271	5.2	0.049	-0.058 *	-
SHR 2_1	0.58	222	5.2	0.054	-0.081	-
SHR2_4	0.66	238	5.9 *	0.027 **	-0.049 **	-
SHR2_5	0.34 w	466 **	6.6 ***	0.031 *	-0.021 ***	+
SHR3_1	0.58	238	5.1	0.041	-0.063	-
SHR3_3	0.47	423 *	6.6 ***	0.046	-0.008 ***	+
SHR5_1	0.59	260	5.3	0.053	-0.095	-
SHR6_1	0.55	238	5.1	0.046	-0.048 **	-
SHR6_3	0.39 w	330	6.0 **	0.029 *	-0.062	-
SHR7_1	0.58	260	5.3	0.039	-0.088	-
SHR7_3	0.61	282	6.1 **	0.034 *	-0.089	-
SHR7_4	0.26 w	704 ***	6.5 ***	0.013 ***	-0.060 *	+
SHR8_1	0.47	287	5.0	0.036	-0.067	-
SHR8_3	0.60	244	6.2 **	0.038	-0.078	-

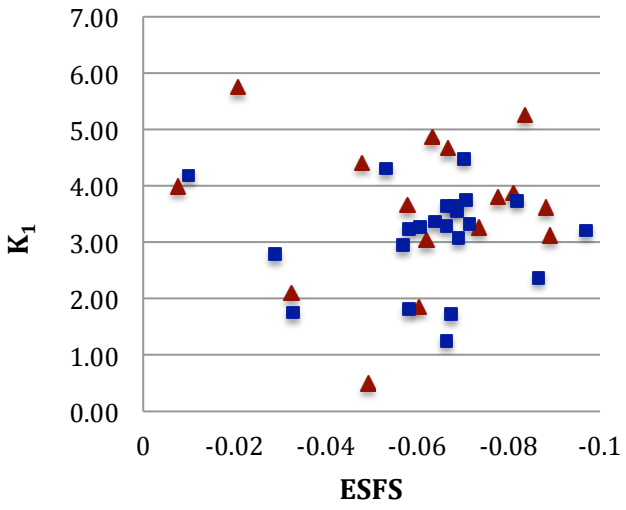

Figure 2-3: Perfusion value K_1 vs. ESFS. Squares are WKY data and triangles are SHR data. No correlation was found ($R^2 = 0.00016$).

2.4: Discussion

The current study combined strain analysis of LV deformation behavior with traditional measures of geometry, weight, and EF in order to provide a more complete picture of the changes in structure and function that occur over the lifespan of the SHR. Traditional measures of EF and wall thickness provide information about the overall level of function, and degree of hypertrophy. Strain provides a detailed picture of the changes in regional deformation behavior of the LV over the lifespan, and enables the study of the different time courses of diastolic and systolic function separately. Comparisons between measures of structure (such as wall thickness), and function (such as strain) provide clues about the underlying disease mechanisms.

Weight, LV Volume, and Wall Thickness Measurements

The time course of the development of myocardial hypertrophy in this study is consistent with the literature on the SHR when considering heart to body weight ratios [40], as well as wall thickness [43]. The relatively unchanging SV and noticeably decreasing EF in the aging SHR is consistent with previously reported results [43]. The relatively unchanging RCV over much of the lifespan followed by an increase at the end of life is consistent with the literature suggesting that hypertension causes stable concentric hypertrophy over a long period of time followed by decompensation [20]. The proportionality between RCV and EF suggests that LV lumen volume increase tends to coincide with failing pump function during the late stages of life in the SHR.

Strain Estimation by Hyperelastic Warping Image Registration

The time courses of decreases in LV function in the aging SHR observed in this study are also consistent with previous studies that measured EF [41]. This confirms that the images used in this study were representative of typical SHR LV function. The strain data, based on the same images, can thus be assumed to be a reasonable representation of SHR LV function.

The observation of decreases in diastolic strain before decreases in systolic strain suggests that stiffening of the heart wall, associated with decreased diastolic filling, may precede the changes in systolic function that ultimately lead to heart failure. Increased stiffness of the myocardium may actually contribute to causing systolic contractile dysfunction by increasing the workload on the LV. In other words, in order to maintain SV with decreased diastolic volume, smaller systolic volume is required, which increases the contractile workload. This interpretation

of the time course of changes involved in the development of hypertension induced hypertrophy is consistent with that reported previously [20], however, this is the first study to quantify the separate time courses of changes in diastolic and systolic strain. Changes in diastolic and systolic function due to long-term hypertension may have different underlying mechanisms. Separating these effects in future studies of long term hypertension induced changes in LV function may be important for understanding the underlying disease.

The regional strain data showed that the lateral region tended to have the greatest strains, in both diastole and systole, for both WKY and SHR groups. This result makes sense intuitively because the septal region of the LV is constrained in its movement by the attachment of the RV and thus the largest strains could be expected to occur on the lateral wall. Spatial variation in strain was greater in systole than diastole. In other words, the difference between the lateral region strain and the other regional strains was greater in systole than diastole. This spatial variation in systolic deformation behavior was much less pronounced in the aging SHR than in the WKY, indicating that the aging SHR may have pattern of altered LV mechanics in which the lateral region shows decreased systolic function. While the notion that pressure overload causes relatively uniform change in *passive* material properties of the myocardium is well supported in the literature [40, 93], changes in spatial variation in contractile function is not well understood in the SHR.

Comparison of Strain Data with Other Measurements

The relationship observed between the degree of wall thickness increase and the degree of reduction in EDFS suggests that hypertrophic adaptation reduces the LV's ability to stretch in

diastole. It is generally understood that prolonged hypertension causes thickening and stiffening of the myocardium, which initially serves a compensatory role in response to pressure overload by both preventing overstretch during diastole and enabling adequate contractile function in systole, but which ultimately contributes to HF when the LV becomes too stiff to allow for adequate filling volumes. The relationship observed between severe hypertrophy and congestive heart failure with fluid in the lungs at the time of death is consistent with generally understood trends in hypertensive disease.

There was no direct relationship between EF and either ESFS or EDFS because EF is a measure of blood volume output, whereas EDFS and ESFS are measures of LV tissue deformation relative to an intermediate reference configuration. A decreased EF can occur as a result of *either* decreased diastolic filling (low EDFS) or decreased systolic contraction (low ESFS), as was observed in this study. An additional difference between EF and strain is that, because EF is a *relative* measure of volume output, it is not sensitive to simultaneous changes in end-systolic and end-diastolic volumes. In contrast, strain is measured relative to a reference (zero strain) state. This highlights some of the advantages of strain as a measure of LV function.

The lack of correlative relationship between the LV perfusion rate constants and the average fiber strains indicate that perfusion was maintained regardless of the functional state of the LV. This lack of a relationship persisted even in those subjects having HF. Therefore, perfusion of the LV appears to be independent of the passive and contractile mechanics of the LV.

Limitations

There are some limitations to the work presented in this study. First, there was a limited number of SHR data sets at the later stages in life because fewer of them lived until the end of the study. This limited the ability to achieve statistical significance in the measurements of interest at the later stages of life for the SHR. Future work will address this limitation by using a larger number of subjects.

Secondly, blood pressure measurements were not successfully made in this study due to experimental difficulties with the tail cuff method. In future work, additional means of obtaining blood pressure measurements will be pursued, as this may explain why some SHR subjects tended to show altered LV mechanics sooner than others. Left ventricular pressure measurements via catheterization were intentionally not performed in this study because of the risks of infection and early death that often accompany repeated catheterizations [109-112]. In this study, the focus was on the progressive changes in LV deformation behavior over the whole lifespan, with particular interest in the altered mechanics in the aging SHR, and repeated catheterizations may have decreased the already small number of animals that lived until the later image acquisitions. Ideally however, LV pressure measurements would be useful in understanding the relationship between decreased diastolic strain and hypertrophy. Changes in diastolic strain and wall thickness could be compared to changes in diastolic filling pressures in order to gain insight into whether the decreased strains are more due to myocardial thickening or material stiffening. Future work will consider other means of obtaining pressure measurements.

Next, because the strains are based on image registration, the accuracy of the strains is limited by the resolution of the images. While this type of microPET imaging has the advantage of being able to evaluate both perfusion and geometry at once, it has the limitation of having a lower spatial resolution than some other imaging modalities. In addition to spatial resolution,

temporal resolution over the cardiac cycle was also limited to 8 image gates. It is possible that the images depicting the most contracted LV may not have represented the exact moment of end systole, and so there could be error in the end-systolic strain measurements. More image gates over the cardiac cycle would make it possible to choose the exact moments of end diastole and end systole.

Lastly, imaging issues at the second acquisition resulted in a series of images that were not usable. Image-based strain measurements at more time points over the lifespan in the 12 to 18 month range would provide more detailed information about the changes in myocardial function that occur during the transition to heart failure.

Conclusions and Future Work

The in-vivo strain measurements reported in this study, and the quantitative comparisons of myocardial deformation and geometry, are in agreement with generally understood trends in hypertensive disease and give insight into the time course of changes in diastolic and systolic function in the SHR model. Changes in diastolic function may be an early effect in the progression of the disease. This supports the notion that diastolic dysfunction precedes systolic dysfunction in long-term hypertension and highlights the importance of separating the diastolic and systolic effects when studying this disease. The association between HF in the aging SHR and decreased systolic function in the lateral region of the LV suggests that spatial variation in function may be important to consider in the study of the transition to HF. The type of image-based strain measurements used in this study are a useful means of gaining insight into the

differences in the time courses in diastolic and systolic function, and also enable the assessment of spatial variation in function due to long term hypertension.

Local strain measurements can also be related to other local measures of physiologically relevant quantities. In future work, this type of longitudinal population study on the SHR will be extended to include a direct comparison of local systolic strains with local metabolic substrate utilization, as well as local measures of myofiber disarray. These comparisons may explain the changes in spatial variation in contractile function observed in the SHR, and may provide further insight into the underlying disease.

Chapter 3. A Method for Computational Modeling of Cardiac Growth Applied to Pressure Overload Hypertrophy

3.1: Introduction

Modeling of left ventricular (LV) hypertrophy has the potential to make significant contributions to the understanding of cardiac disease, and influence the development of treatment strategies. However, the cyclic nature of LV mechanics complicates the modeling of LV hypertrophy because the growth and remodeling (G&R) can be affected by many aspects of the full cardiac cycle mechanics. Methods for biomechanical modeling of cardiac hypertrophic growth have yet to incorporate growth stimuli from multiple points in the cardiac cycle (or growth stimuli that fully represent the full cardiac cycle mechanics). This work aims to address this problem(s); a novel growth modeling approach is presented.

Load-induced hypertrophy of left ventricular myocardial tissue can occur in a wide variety of disease states including hypertension, aortic stenosis, myocardial infarction, and left bundle branch block. In this project, the efficacy of the LV hypertrophy modeling method will be illustrated for the case of hypertension, because this disease is well researched in terms of its morphological effects, and because it is the most widespread cardiovascular disorder in the world. In spite of increased pharmacological treatment in recent decades, this condition continues to lead to heart failure for tens of thousands of people per year [1], largely due to the

progressive damage to the left ventricle that is caused by load-induced hypertrophic structural adaptation.

In systemic hypertension, the left ventricle (LV) must work against increased pressure load to pump blood to the body. Over time, this excessive workload causes thickening and stiffening of the myocardium. Myocytes are known to react to their mechanical environment by thickening when loads and/or deformations are greater than normal. The thickening of these muscle cells, known as hypertrophy, results in an overall increase the LV wall thickness and LV mass. Hypertensive (pressure overload) hypertrophy is known as *concentric* hypertrophy, and is characterized by an increase in LV wall thickness while LV cavity volume remains normal [36]. Global changes in LV geometry have been documented in both human studies and animal models of hypertension. In humans untreated hypertensive hypertrophy progresses over the subjects entire lifespan, and so the use of animal models is convenient for studying the long term disease progression. The spontaneously hypertensive rat is the most commonly studied animal model of hypertensive hypertrophy because it closely mimics the changes in morphology and physiology that is observed in humans, and because its two year lifespan makes it possible to study the longitudinal disease progression. LV wall thickness in SHR and WKY subjects was measured at 90 weeks of age (old age) by Pfeffer et al [43] to be 4 mm for the WKY, and 5 mm for the SHR, indicating hypertrophy in the SHR. Numerous human studies have quantified varying degrees of hypertensive hypertrophy using medical imaging [44]. Echocardiography has been used to determine LV end-diastolic wall thickness to be 12.8 +/- 1.6 mm in hypertensive patients compared with 8.5 +/- 1.1 mm for healthy volunteers [45]. In severe cases of hypertensive hypertrophy, ventricular wall thicknesses of up to 20 mm have been observed [46]. While initially a compensatory mechanism, this adaptation can eventually lead to heart failure (HF) [2].

An understanding of the mechanisms of hypertrophic growth, and the mechanical stimuli that cause it, is essential to the development of treatments strategies (which might seek to interrupt those mechanisms in the interest of preventing excessive growth). For this reason, several potential growth stimuli have been investigated. In a study of volume-overload hypertrophy, Holmes et al [55] suggest that because myocytes normally experience cyclic variations in both stress and strain, candidate hypertrophic stimuli might include minima, maxima, mean values, amplitudes, and frequencies in the stress and strain signals.

In the case of hypertension (pressure-overload), the LV must exert more energy than normal in order to pump against the increased systemic pressure. Changes in energy use and metabolism have been documented in hypertensive hypertrophy. Changes in energy substrate utilization have been found to directly correlate with LV mass [59]. The connection between energy use and hypertensive hypertrophy suggests that the total energy expended during systole (total systolic work) may be a potential growth stimulus in the case of pressure overload. In general, it is likely that mechanical stimuli from multiple points in the cardiac cycle (i.e. diastolic strain, and systolic work or stress) may be important in the development of hypertrophy.

Computational models of tissue growth and remodeling have emerged as an effective means to test hypotheses related to growth stimuli and mechanisms of adaptation [84, 113]. Much work in recent decades has been devoted to the development of theoretical and computational frameworks for modeling of cardiac G&R.

One approach to modeling of G&R due to a mechanical stimulus has focused on the development of constitutive laws that cause volumetric expansion of the material in response to loads or deformations. The strength of this approach is that it is possible to obtain a distribution of growth that is based on the local tissue mechanics. It is believed that myocytes respond to

overload individually, and so the whole organ response is thought to have a spatial distribution of growth that is based on the local mechanics. In the constitutive growth modeling approach the intention is to define a material that will have an instantaneous load-growth response in addition to the usual load-deformation response. This can be accomplished with the decomposition of the deformation gradient into an elastic part $\boldsymbol{F_e}$, which governs the load-deformation response, and a growth part $\boldsymbol{F_g}$, which governs the load-growth response [75]. The overall deformation gradient is then $\boldsymbol{F_{eg} = F_e F_g}$. The definition of the growth portion, $\boldsymbol{F_g}$ is then chosen based on an assumed mechanical growth stimulus, such as stress or strain, and it can cause either isotropic or anisotropic growth patterns.

The strength of the constitutive modeling approach is that it is possible to obtain a through-wall distribution of growth that is based on the local tissue mechanics. The primary limitation of the constitutive modeling approach is that only a single moment in the cardiac cycle mechanics can be used to drive growth. In order to better reproduce the in vivo growth mechanisms, mechanical stimuli from throughout the cardiac cycle should be utilized to drive the growth. A second limitation is that it is not possible to run a simulation of the cardiac cycle following the growth of the LV. When the growth is implemented as a constitutive law, the LV has to be in a loaded state in order to be in a grown state, and so it is not possible to obtain a grown reference configuration from which to begin a cardiac cycle analysis. In order to understand the ways that the growth is affecting the full cardiac cycle mechanics, the growth should be applied to a reference configuration geometry.

The purpose of this study was to create a framework for modeling cardiac G&R in which (1) multiple mechanical stimuli from throughout the cardiac cycle can be used to drive growth, (2) the growth is applied to the reference configuration geometry, and (3) the spatial distribution

of growth throughout the LV is based on the local tissue mechanics. The intention is to leverage the advantage of the constitutive modeling approach's ability to create a through wall distribution of growth that reflects the local mechanics, while incorporating local mechanical data from throughout the cardiac cycle, and ultimately creating a thickened reference configuration geometry from which to assess the new cardiac mechanics. Addressing these key issues is a necessary step toward the goal of testing hypotheses about growth stimuli. In other words, in order to test hypotheses about growth mechanism, it is necessary to have a modeling framework that can incorporate multiple growth stimuli, and that can evaluate the affect of the resulting growth on the full cardiac cycle mechanics. Eventually, once growth stimuli have been studied extensively, the ultimate goal of this area of research is to develop models that incorporate both the mechanical properties as well as the G&R properties of cardiac tissue in order to actually predict medical outcomes of treatments.

3.2: Methods

Method for Modeling Cardiac Hypertrophy

Overall Modeling Framework

This section describes the overall flow of the modeling procedure that was developed for this study [Figure 3-1]. More detail on each of the steps in the process is given following this section.

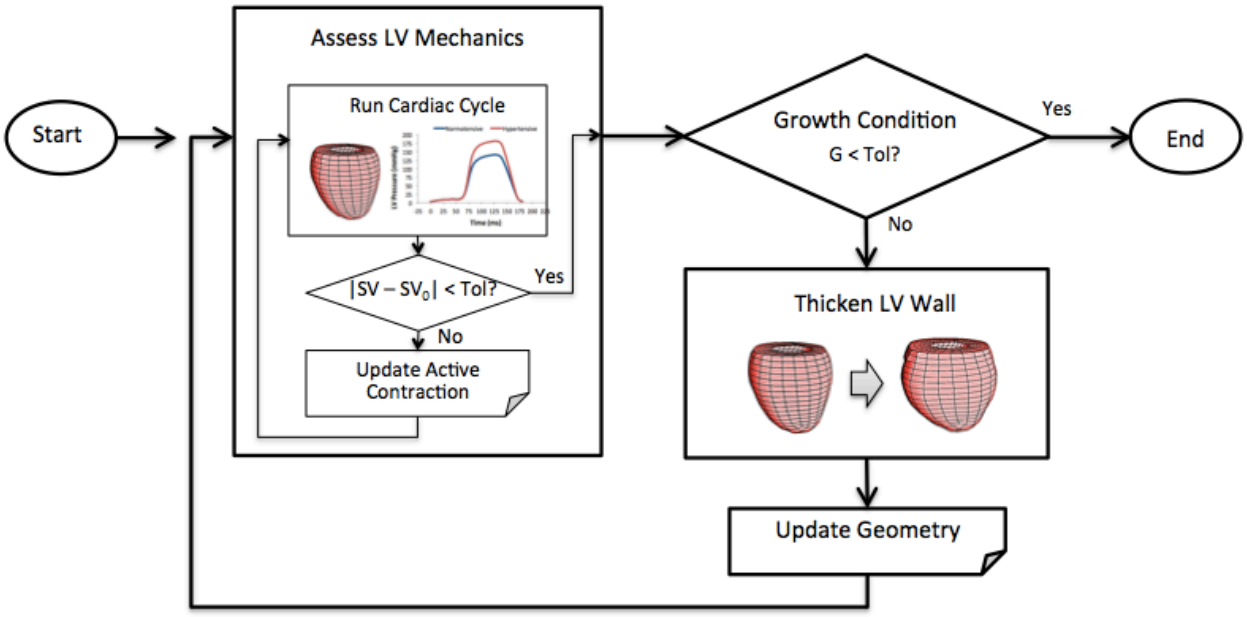

Figure 3-1: Flowchart representation of overall growth modeling method

The initial step in the modeling process is to run a finite element analysis of the cardiac cycle for the case of normal mechanics. The nodal stresses and strains are recorded throughout the cardiac cycle, and the EDV and ESV are also calculated. These data are saved for future comparison with the altered mechanics in the hypertrophy model.

The growth model begins by running one cardiac cycle of an LV finite element model subjected to diseased mechanics, as characterized through the application of hypertensive pressure loading. The nodal stresses and strains are recorded throughout the cardiac cycle, and the EDV and ESV are also calculated. These data are used to determine whether the LV must thicken in response to the diseased mechanical state. The growth condition can be based on any measure, or combination of measures, from the full cardiac cycle mechanics that was stored from the previous analysis. If the value of the (global average) mechanical growth stimuli, G, is

greater than a small, user defined tolerance, then the model continues to a procedure to thicken the LV wall. The procedure for thickening the LV wall uses the stored data from the cardiac cycle, in combination with a growth constitutive model, to create a distribution of growth throughout the LV that is related to the full cardiac cycle mechanics. Once the LV has been thickened by some incremental amount in the wall thickening procedure, the new geometry is then used to reassess the mechanical measures of interest by running a new cardiac cycle analysis. The fiber orientations are re-calculated for the new geometry in the same manner as they were for the original mesh. Each time the model re-assesses the mechanics after the LV has been thickened, it will be necessary to determine a new value for the applied active contraction of the material specified in the mechanical model. This is achieved with a secant method approach that conserves the Stroke Volume. The model continues in this manner until the growth condition is no longer met.

Growth Condition

The growth condition, which controls the total degree of hypertrophy, can be specified as any measure, or combination of measures, from the full cardiac cycle mechanics that was stored from the previous analysis. The term G represents this global growth stimulus, which, for example, could be based on the average nodal end-diastolic strain and the average nodal end-systolic stress. G_n denotes the "normal" value of that quantity, i.e. the value of G obtained from the initial analysis of the LV with normal mechanics. The growth condition determines whether or not it is necessary to increase the thickness of the LV. If the diseased mechanics causes G to be greater than normal by more than a small tolerance, then the model continues to a procedure that thickens the LV wall.

Wall Thickening Procedure

The procedure for thickening the LV wall uses the stored data from the previous cardiac cycle, in combination with a growth constitutive model, to create a distribution of growth throughout the LV that is related to the full cardiac cycle mechanics. The constitutive relation used for the material definition in the wall thickening procedure is a variation of the type of growth constitutive law that has been used in previous G&R models [75, 82, 87, 88]. However, its implementation differs from the previous work in that the constitutive growth condition is based on stored data from a previous cardiac cycle analysis. A general definition of the deformation gradient is:

$$\boldsymbol{F_g} = \left(g_f - g_{fc}\right)\beta_f \boldsymbol{f}\otimes\boldsymbol{f} + \left(g_s - g_{sc}\right)\beta_s \boldsymbol{s}\otimes\boldsymbol{s} + \left(g_n - g_{nc}\right)\beta_n \boldsymbol{n}\otimes\boldsymbol{n} \qquad \textbf{Equation 3.1}$$

Here, there are three terms, which determine the growth in the fiber, sheet, and sheet-normal directions of the myocardial material, which is orthotropic in nature [21], and tends to grow in an orthotropic manner in response to different types of diseased mechanics [20]. In practice, only one of these terms might be used to model any given disease process. For example, in the test case described in the Section 3.2.3, growth is only in the sheet direction. Taking one of these, $\boldsymbol{F_g}$ is scaled by $(g - g_c)$, where g is the local growth stimulus, which can be based on any measure or combination of measures from the previous analysis of the cardiac cycle. The choice of g is a user-defined input. g_c is the critical value above which growth will occur, which is a set value that is based on the initial normotensive analysis. The beta values simply allow the user to scale the degree of growth in the myofiber, sheet, and sheet-normal directions. For example, it would

be possible to define $\boldsymbol{F_g}$ with the fiber-direction growth distribution based on diastolic strain, and the sheet-direction growth distribution based on systolic stress, etc.

The wall thickening procedure begins with an FE mesh of the LV in its reference configuration geometry. Then, a nominal loading state is applied in order to achieve the appropriate change in geometry. In the Test Case of hypertensive hypertrophy this loading state was defined by fixing the nodes on the interior (endocardial) surface, and applying a negative (pulling) pressure to the outer (endocardial) surface. The value of the applied load was chosen to cause an incremental amount of wall thickening. This procedure will thicken the LV wall based on the constitutive relation. For comparison, if a simple elastic material were to undergo this type of loading, all of the elements would stretch slightly in a uniform manner. However, with the growth constitutive relation, much more volumetric growth will occur, and the degree of element growth will vary based on the local element value of g. This enables the model to create a spatial distribution of growth that is based on the spatially local mechanical states, and which can incorporate data from throughout the cardiac cycle. The final grown geometry is affected by both the load state chosen for the wall thickening procedure, and the through-wall distribution of the local growth stimuli.

Stroke-Volume Conserving Active Contraction

Each time the model re-assesses the mechanics after the LV has been thickened, it will be necessary to determine a new value for the active systolic contraction of the material. This is because, when the LV thickens, there is more material exerting active contractile stress, and so less active contractile stress is necessary to achieve the same amount of LV volume output per cardiac cycle. The LV volume output per cardiac cycle is the Stroke Volume (SV). In the

literature, stroke volume has been shown to stay constant over the lifespan regardless of the state of hypertension [43]. This indicates that LV myocardial tissue will tend to contract with as much active stress as necessary in order to meet the demand for cardiac output.

The procedure for determining the correct active contraction value for any given thickened state is based on a secant method. First, an initial guess of the active contraction value is made based on the following relation:

$$c_{start} = c_0(1 - \alpha^3), \text{ where } \alpha = (t_g - t_0)/t_0. \qquad \textbf{Equation 3.2}$$

Here, c_0 is the active contraction in the initial FE model, t_0 is the wall thickness of the initial FE model, and t_g is the grown wall thickness. Then, the cardiac cycle is run, and the SV is assessed. If the SV is within a small tolerance of the initial forward model SV, then the model analysis continues. If the SV was outside that range, then a secant method approach is used to determine the next active contraction estimate. For complete details on the implementation, see Appendix A.

Test Case: FE Model Creation

Geometry and Mesh

Gated microPET image data sets were obtained for an SHR subject at 6 months (young adult) and 20 months (old age). The 6 month image data set was used to define the initial mesh geometry. The image gate approximately 1/3 of the way into diastole was used to define the initial mesh geometry. This time point has been shown to be the closest to an unloaded state that occurs in-vivo [21]. This "reference configuration" image was manually segmented, and the resulting closed contours were used to define the FE mesh using TrueGrid preprocessing

program [22]. The FE mesh was created with 2814 nodes and 2220 hexahedral (brick) elements.

Constitutive Model for Mechanics

The LV myocardium was modeled as a transversely isotropic hyperelastic material with realistic fiber inclination angles and active fiber contraction. In particular, the transversely isotropic Mooney-Rivlin [10] model in FEBio was chosen for this study. The material property values were based on literature values [114]: C_1 = 2.325 kPa, C_2 = 0.232 kPa, C_3 = 0.3075 kPa, C_4= 0.002, C_5 = 0, K = 100 kPa, λ_{max} = 9. Active contraction is available in this material model and the active properties were based on literature values [115]: ca_0= 4.35, β = 4.75, l_0 = 1.58, l_{ref} = 2.04. Realistic fiber inclination angles were calculated using a custom code, to vary through the thickness of the LV wall. The LV mesh had five layers of elements through the wall, and the fiber inclination angles were specified to be 65, 35, 5, -25, -55 degrees in each layer, from the endocardial surface to the epicardial surface.

Boundary Conditions

The nodes on the surface of the base of the LV were fixed in the z-direction. A "tether" of very soft linear elastic material was created to encircle the top two rows of elements (LV base). This was done to provide a mild constraint that facilitates convergence of the finite element solution without fully restricting the movement of the base of the LV in the x-y plane. The material properties of the tether material were E = 10 kPa, and ν = 0.4. The diameter of the tether was 40 mm, and the nodes on the outer edge of the tether were fixed in all three directions.

Pressure Loads

The SHR is commonly compared to the normotensive Wistar-Kyoto (WKY) rat in animal studies [40, 41]. The normal systolic pressures for the WKY have been measured to be 125 mmHg [41], although measurements vary from experiment to experiment. The cutoff value defining of hypertension in humans according to the JNC7 [34] is 140 mmHg. The value of "normal" end-systolic pressure to be used in the Test Case model was chosen to be 140 mmHg.

The hypertensive systolic pressures measured in the SHR have been reported to be approximately 200 mmHg in the literature [41]. The LV end-systolic pressure used in the Test Case study was 200 mmHg.

The end-diastolic pressure (EDP) was chosen to be 8.5 mmHg for both the normotensive and hypertensive models because the goal of the Test Study was to evaluate the effect of increased systolic workload due to an increased systolic pressure. In reality, literature shows that the SHR tends to have greater EDP than the WKY [41], however, study of this additional variable was outside the scope of the Test Case model.

Test Case Growth Modeling Choices

Work Definition

Myocardial energy use has been shown to correlate with myocardial mass, suggesting that total systolic workload may influence hypertrophy in the case of hypertension [59]. The systolic work, the work done by the LV during the systolic phase of the cardiac cycle, was used in both the Growth Condition, to determine the total degree of wall thickening, and in the Wall

Thickening Procedure, to determine the through-wall distribution of growth. The total systolic fiber work for each element was calculated as the integral from end-diastole to end-systole of the element stress in the fiber direction, σ_{ff}, with respect to the element strain in the fiber direction, ε_{ff}.

$$w = \int_{EDia}^{ESys} \sigma_{ff}\, d\varepsilon_{ff} \qquad \textbf{Equation 3.3}$$

Each of the quantities in the above relation are element-specific. The element values of w were calculated after each cardiac cycle analysis. In this chapter, the values for the normal model (the initial model run with normal pressures) will be denoted as w_n with the subscript n, for normal.

The average value of the work, over all of the elements in the mesh, for the normotensive model, $\overline{w}_n$, was calculated. Here, the over-bar on $\overline{w}_n$ denotes that it is an average value rather than an element specific value.

$$\overline{w}_n = \left(\frac{1}{N_{elts}} \sum_{k=1}^{N_{elts}} w^k \right)_{Normal} \qquad \textbf{Equation 3.4}$$

This is quantity was used in the Wall Thickening Procedure of the Test Case model. The superscript on w^k denotes the element number, and N_{elts} refers to the number of elements in the LV mesh.

Growth Condition

The growth stimulus to be used in the growth condition, G, was based on the average difference in element systolic fiber work between the hypertensive and normotensive models.

$$G = \frac{1}{N_{elts}} \sum_{k=1}^{N_{elts}} (w^k - {w_n}^k) \qquad \textbf{Equation 3.5}$$

The Growth Condition was specified to state that when G is greater than the tolerance, then the model will continue to the wall thickening procedure. In the model implementation, the tolerance value was set to be 1% of w_c.

Wall Thickening Procedure

In the growth constitutive relation, the growth deformation gradient, F_g, given in 3.1 was defined such that there would only be growth in the sheet direction, and the distribution of growth would be based on element systolic work as follows:

$$\boldsymbol{F_g} = (w^k - \overline{w}_n)\, \beta\, \boldsymbol{s} \otimes \boldsymbol{s}. \qquad \textbf{Equation 3.6}$$

Here, s is the sheet direction vector, and β is the scale factor in the sheet direction, which was set to unity. Note that $\overline{w}_n$, the average normal element work, is used here, rather than an element-specific normal value. This is intended to represent a single work threshold above which tissue growth will occur.

In the wall thickening procedure, the nodes on the inner (endocardial) surface of the LV were fixed, and a pressure load was applied to the outer (epicardial) surface. The pressure value selected was 6 kPa. This pressure load was selected for the purpose of causing an incremental increase in wall thickness during each growth phase.

Test Case Data Analysis

Wall Thickness

The wall thickness of the LV was assessed in the reference configuration geometry before and after each wall thickening phase. The wall thickness was calculated as the average value of four measurements of wall thickness in the inferior, lateral, anterior, and septal quadrants of the LV. Wall thickness measurements were made in a short-axis slice of the LV. The slice was at mid-ventricle, approximately 65% of the height up from the bottom of the apex. The wall thickness was measured as the distance between nodes on the opposite sides of the LV wall. For details on the implementation of this calculation, see Appendix A.

LV Volume

The LV cavity volume was assessed in the reference configuration state, the end-diastolic state, and the end-systolic state during each cardiac cycle analysis. The volume was calculated based on the nodal locations on the endocardial surface of the LV. For details on the implementation of this calculation, see Appendix A.

Work Normalization

The value of the average element systolic work was calculated for each cardiac cycle analysis. This value was plotted for each wall thickening phase in order to show how the work decreases incrementally toward the normal value as the LV becomes thicker.

Distributions of Work and Growth

The distributions of work and growth were observed in PostView [116] post-processing software, for each cardiac cycle and wall thickening procedure, respectively. These plots were evaluated to qualitatively show the correspondence between the distributions of work and growth. The distributions were also evaluated to observe the decrease in work as the LV thickens.

Additionally, one wall thickening procedure was performed on the LV with a "non-growth" transversely isotropic Mooney-Rivlin material (no $\boldsymbol{F_g}$). The distribution of thickening in the cross section was evaluated for comparison with the growth distributions in the Test Case. This comparison was made in order to show how the growth part of the material law ($\boldsymbol{F_g}$) changes the through-wall distribution of growth observed during the wall thickening procedure.

Additional Model Functionality

Active Contraction Time Delay Feature

Some hypertrophic disease processes stem from alterations in electrical conduction in the heart. For example, Left Bundle Branch Block (LBBB) is a cardiac conduction abnormality which causes a delay along the pathway for LV activation. This time delay in activation varies spatially over the LV with the lateral wall contraction being delayed compared with the septal wall. The hypertrophic growth that occurs in response to this change in mechanics is asymmetric, and is characterized by a severe thickening of the lateral wall which works against a higher pressure load than the septal wall which contracts earlier [117]. In order to be able to

model hypertrophy that occurs in response to a spatially varying time delay in activation, a feature was added to the modeling framework. Each element in the FE model can have a time delay assigned to its active contraction. This element-specific time delay data can be specified in the input file, in the same manner as element-specific fiber orientations. The time delay values could come from patient-specific fast temporal imaging studies, and the model could be used to predict the degree and spatial variation of the hypertrophy.

Direct Spatial Growth Scaling

In some hypertrophy modeling studies, it may be useful to be able to directly specify an a priori spatial variation in the growth. This feature allows the user to directly specify element-specific growth scaling factors. For example, one might want to quickly create an asymmetrically grown geometry for use in a cardiac cycle analysis or other study. Creating a new geometry and mesh from scratch is very time consuming. Thus it may be useful to be able to simply specify a growth distribution in order to create the altered mesh geometry. Furthermore, there are some disease processes that involve non-mechanical growth stimuli, in addition to the mechanical stimuli. The ability to specify an additional element specific growth scale factor would enable the user to incorporate this additional information into the model predicted grown geometry.

Element Specific Local Growth Threshold

The ability to define a spatially varying range of growth thresholds would make it possible to define different hypertrophic responses to loads through the thickness of the myocardium or regionally within the LV. This feature enables the user to specify whether to use

a single local growth threshold value, or whether to use element-specific or region-specific values. In the case of element-specific values, local growth threshold is the element-specific value from the normal model. In the case of region-specific, the values of the elements in a given region are averaged, and that value is used for all the elements in that region.

3.3: Results

Wall Thickness

The wall thickness increased incrementally with each iteration of the wall thickening procedure as shown in Figure 3-2 with the final model-predicted wall thickness being similar to the measured value for the SHR later in life (predicted 4.9 mm and SHR measured at20 months was 5 mm).

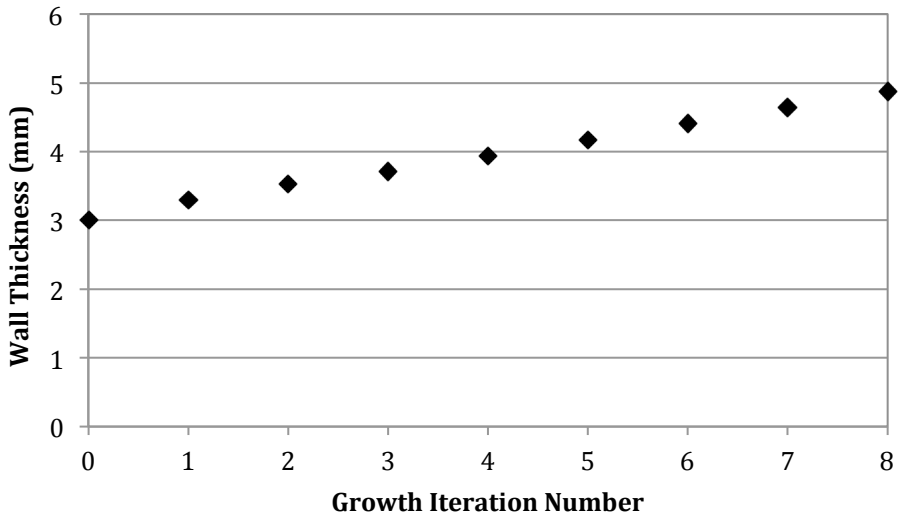

Figure 3-2: LV wall thickness vs. growth iteration number

Work Normalization

The average element systolic work plot [Figure 3-3] indicated that the average element systolic work decreased as the LV wall thickness increased for each successive iteration. After 8 iterations of the wall thickening procedure, the work decreased to within a small tolerance of the "normotensive" model.

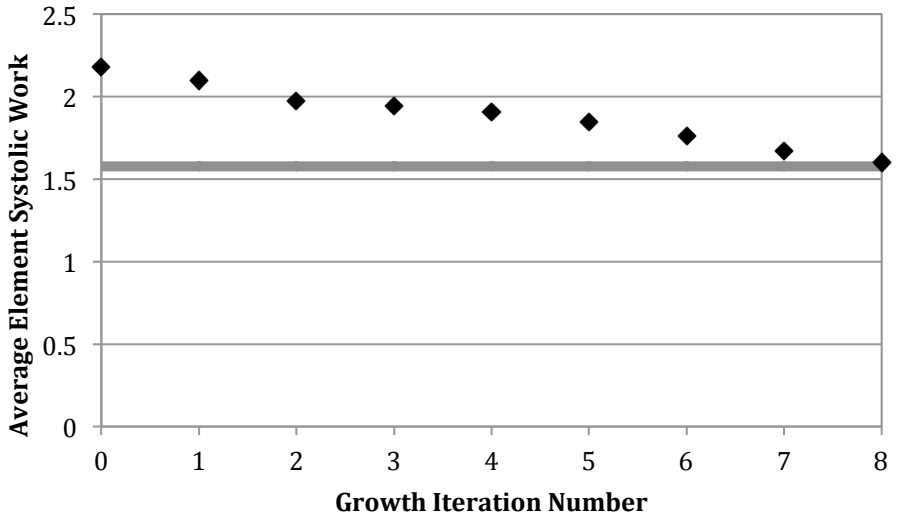

Figure 3-3: Average element systolic work vs. growth iteration number. The grey line represents the "normotensive" value of average element systolic work.

The work growth condition used to determine the total degree of wall thickening was defined as the average of the differences in element systolic work between the hypertensive model and the normotensive model, as discussed in Section 3.2. This parameter can be thought of as the "Overwork" or the degree of work overload experienced by the hypertensive model. The amount of overwork decreased as the wall thickness increased, and after 8 iterations, it decreased to within tolerance of zero [Figure 3-4]. A tolerance value of 0.05 was used, and the final value of overwork was 0.025.

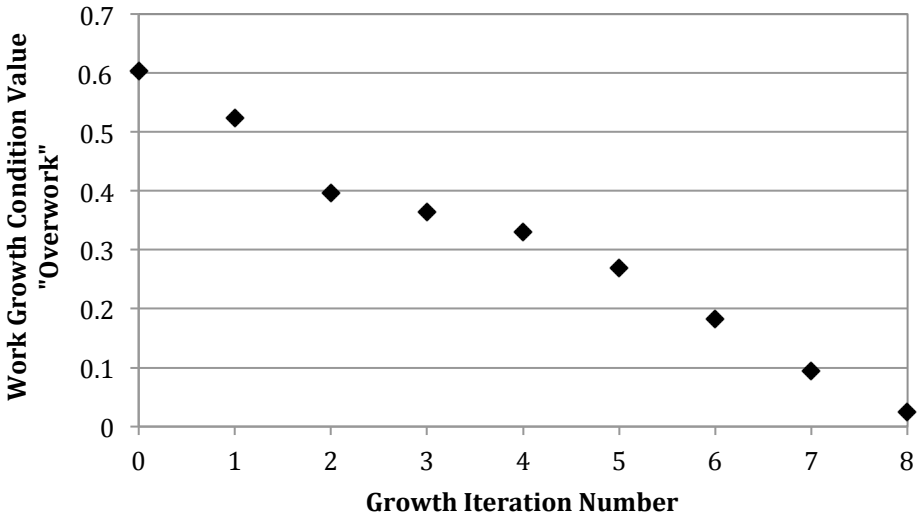

Figure 3-4: Work growth condition ("overwork") vs. growth iteration number. The degree of work overload decreases as the wall thickness increases, and the final value is within a small tolerance of zero.

Distributions of Work and Growth

Figure 3-5 illustrates the distributions of total systolic element work in a short-axis mid-ventricle slice for the cardiac cycle analysis following the wall thickening procedure for the 1st, 4th, and 8th iterations. The distributions show the decrease in work associated with the increase in LV thickness.

Figure 3-6 illustrates the distributions of volumetric strain (volumetric growth) that took place during the wall thickening iterations for the same short-axis mid-ventricle slices. The distributions of growth resemble the distributions of work, in that the regions of increased work in Figure 3-5 correspond with the regions of increased volumetric strain in Figure 3-6. The distribution of growth in the areas where the local work is lower than the growth threshold show a flat or even distribution.

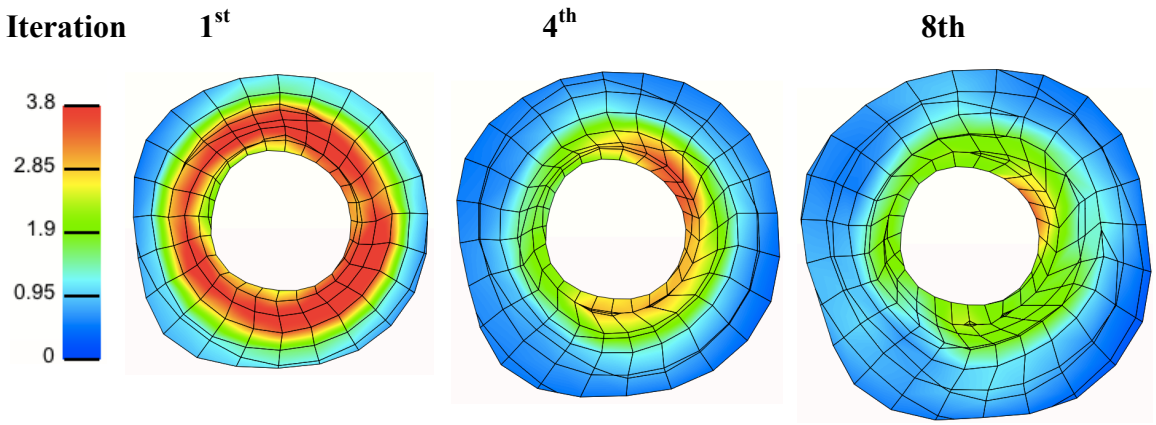

Figure 3-5: Systolic work distributions, shown in a mid-ventricle short-axis slice, from the first, fourth, and eighth iterations of the modeling procedure. The magnitudes of work decrease as the LV wall becomes thicker.

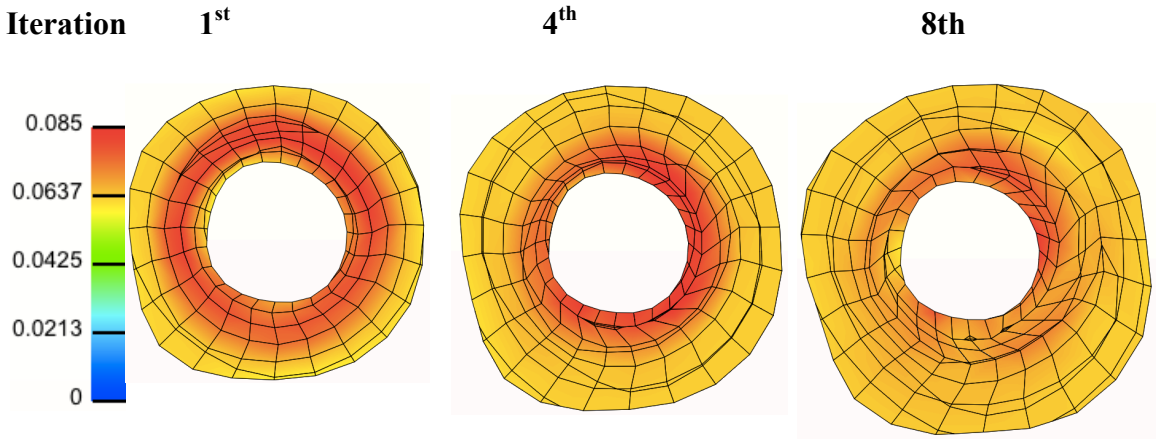

Figure 3-6: Distributions of volumetric strain (growth) from the first, fourth, and eight iterations of the wall thickening procedure. As the LV wall becomes thicker, less growth occurs. The "hot" areas are the areas where the work from the previous cardiac cycle exceeded the local work threshold for growth.

The distribution of growth in the cross section for a regular (non-growth) transversely isotropic Mooney-Rivlin material subjected to the loads of the wall thickening procedure showed

uniform and relatively small volume increase compared to the growth model distributions [Figure 3-7].

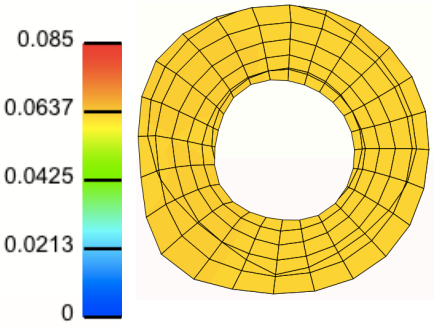

Figure 3-7: Volume strain for a transversely isotropic Mooney Rivlin (non-growth) material subjected to the loads of the wall thickening procedure. Volume increase is small and uniform compared to the growth model results.

3.4: Discussion

LV Wall Thickness and the Growth Condition

The model-predicted wall thickness (4.9 mm) was similar to the measured value (approximately 5 mm) for the same SHR subject later in life (20 months). These values are consistent with typical wall thickness measurements reported in the literature for the SHR late in life [43]. As discussed earlier, the total degree of wall thickening was determined by the growth condition, which, in the Test Case, was the difference in systolic work between the hypertensive model and a predefined non-hypertensive model. This can be thought of as the degree of systolic work *overload*, compared to a non-hypertensive model.

With increased wall thickening, there is more material exerting contraction during systole, and so less active contractile stress is necessary to achieve the same stroke volume, resulting in less tissue work. The model's adaptive active contraction loop was effective at determining the correct amount of active contraction to achieve the required stroke volume as seen in the normalization of the average work values in Figure 3-3. These results are consistent with the literature, where the prevailing thought is that increases in wall thickness serve to reduce the wall stresses to normal levels [21].

The total degree of wall thickening is affected by the value of systolic pressure used in the normal/non-hypertensive model. The choice of peak systolic pressure of 140 mmHg used in the normal model of the Test Case resulted in a realistic prediction of wall thickness. This is the cut-off value for brachial systolic blood pressure between prehypertension and stage I hypertension according to Table 1-1. Systolic pressures in the LV are known to be higher than brachial systolic blood pressures, so the application of this value as the LV peak systolic pressure corresponds to a case in which the brachial systolic blood pressure would be below this cut-off. In the literature, systolic blood pressures for normotensive rats have been measured to be 125 mmHg [41]. In the model, if a lower value were used to define the baseline mechanics measures of the non-hypertensive case, then the model-predicted (hypertensive) wall thickness would be greater. And conversely, if a higher value were defined as normal, then the final model-predicted wall thickness would be less.

In this model, a single value of systolic pressure was defined as *normal*, and the total degree of hypertrophy was essentially a linear function of the difference between the hypertensive and normal pressure. In reality, there is no one set threshold for blood pressure above which hypertrophy will occur. There is considerable variability between individuals.

Furthermore, due to additional physiological factors, the degree of hypertrophy does not necessarily scale directly with the level of hypertension[42] and that damage mechanisms from very high levels of stress or work can inhibit hypertrophy development.

In this model, the growth was driven by one variable only, the systolic work (which is a function of the systolic pressure). In reality, there are a large number of physiological influences that affect hypertrophy. In large population studies, in which results were controlled for multiple known risk factors, including systolic blood pressure, only 60-68% [118] of the variability in LV mass could be explained. Genetic risk factors are currently being studied in an effort to understand more of the influences affecting hypertrophy [119, 120]. Those caveats aside, decreases in systolic blood pressure are known to decrease the development of hypertrophy, blood pressure defined as hypertensive [34] appears to be the primary influence in the development of hypertrophy, and myocytes are known to increase in size in response to elevated loads [41].

The choice of total systolic work for the growth condition is one way to account for the hypertensive systolic pressure load, while also taking account of the deformation necessary to achieve the required stroke volume. Furthermore, it has been suggested that energy use (the total energy expended by the tissue during systole) may be a driving factor for hypertrophy. The use of a measure of local energy expended during systole to drive growth may be better at predicting growth than stress or strain at any one point in time in the cardiac cycle because it represents the behavior over the whole systolic phase.

Distributions of Work and Growth

The distributions of growth show distinct differences between the regions showing larger increases in volume versus areas that have lower and more uniform values of volume increase. The regions of high volume increase directly coincide with high work regions. These are the areas where the local systolic work exceeded the work threshold value, and thus caused the growth deformation tensor to increase the volumetric expansion. The areas where the work was below the growth threshold the growth distributions showed mid-range volumetric strain values (yellow rather than red) indicating these elements were stretching in the same manner as a material that does not have the additional growth deformation tensor. These areas appear very similar in volumetric strain values to the uniform distribution obtained when the non-growth material was subjected to the same loading for comparison.

The use of a single threshold value for the local growth stimulus has been used extensively in the application of this type of growth constitutive model [87] [81, 86, 88]. This threshold is intended to represent the observed behavior in tissue studies in which (controlling for other variables) higher-than-normal loads cause hypertrophy, while “normal” loads do not. It is believed this threshold is a property of the myocardial material [87] and does not vary spatially throughout the organ. However, this aspect of growth modeling is still in the early stages of development and further work will be required to determine whether some spatial heterogeneity in the growth response to loads should be included in this type of model.

The growth model employed in this study was successful in being able to incorporate information from throughout the cardiac cycle into the through-wall distribution of growth. In-vivo, myocytes normally experience cyclic variations in both loading and deformation. Cellular mechanisms for sensing when the mechanical state is “normal” versus “overloaded” may therefore respond to mechanical information from throughout the cardiac cycle, including

maximum and minimum stress and strain, as well as temporal information such as heart rate, strain rate, etc. The modeling method presented represents the first implementation of finite element-based growth modeling of left ventricular hypertrophy that can incorporate information from multiple points throughout the cardiac cycle. This framework can be used as a test bed for evaluating competing hypotheses about mechanical growth stimuli.

This methodology is able to incorporate multiple mechanical stimuli into the spatial distribution of growth through the definition of a growth deformation gradient that is scaled by locally stored data from the cardiac cycle analysis. This work represents a unique and novel application of the growth constitutive modeling approach. In all previous models that used a constitutive growth law, the growth deformation tensor was scaled by the current state of stress or strain, when the model was subjected to passive filling pressures. In that type of modeling scenario, the degree and distribution of growth are determined as an instantaneous load-growth response to a single state of mechanical loading. This is certainly appropriate for applications in which it is known *a priori* that only a single state of loading is involved in the growth mechanism. For example, in some locations in the vasculature, the variations in pulse pressure are small in comparison to the left ventricle, and so the overall loading might be approximated by a single "average" loaded state. In this case, the direct implementation of the constitutive growth law for obtaining the instantaneous load-growth response to a single state of loading might indeed represent the in-vivo response. In contrast, the left ventricle experiences extreme variations in pressure, and it is thought that mechanical information from multiple points in the cardiac cycle may be important determinants of the hypertrophic response. The method presented herein overcomes this limitation by using stored data from throughout the cardiac

cycle, in combination with a nominal loading state in a wall thickening procedure, to obtain a spatial distribution of growth that incorporated information from throughout the cardiac cycle.

The final distribution of growth is affected by both the growth stimuli, and the nominal loading state used in the wall thickening procedure. In the Test Case presented, the nominal loading state was shown to produce a uniform and small degree of wall thickening when the growth deformation tensor was not included. This resulted in a final distribution of growth in which the areas that exceeded the local work threshold showed larger volumetric strains that scaled in comparison to the threshold value, and the areas that did not meet this threshold grew by a small uniform amount. In hypertension, the true through-wall distribution of growth is unknown. More experimental data will be required to determine how this aspect of the model should truly behave.

Mechanical Modeling Assumptions

In this or any hypertrophy model in which the growth is a function of the local mechanics, the results of the growth will obviously be affected by the accuracy of the mechanical model (forward model of the cardiac cycle). In the Test Case presented, typical cardiac finite element modeling assumptions were made with regard to loads, material properties, etc. What follows is a discussion of some of these modeling assumptions.

The applied endocardial pressure loads are based on literature values. Ideally, a model would include subject-specific pressure measurements, however, catheter pressure measurements were not experimentally feasible in this case. And, blood pressure measurements for the experiments discussed in Chapter 2 were unsuccessful due to experimental difficulties with the

tail cuff method. The epicardial pressure is neglected. This is typical for LV FE models because in-vivo epicardial pressures are difficult to measure and are thought to be negligibly small in comparison to the endocardial loads.

The passive myocardial properties were represented with a transversely isotropic hyperelastic material model. Transversely isotropic material models are commonly used in LV FE models, however, orthotropic material models have been shown to be slightly more accurate at representing myocardial behavior [23]. In the Test Case, only the fiber-direction stresses and strains are used in the growth model, and so it is likely that the difference between transversely isotropic and orthotropic material models would have a very small effect on the final results. The fiber inclination angles were defined with a realistic through-wall distribution, as is commonly done in the literature [60]. Ideally, a model would include subject-specific fiber angles determined from DT-MRI. However, this is not possible for longitudinal studies of repeated imaging over the lifespan because DT-MRI can only be performed ex-vivo.

The residual stress/strain distribution was not included in the Test Case model. Following typical assumptions for LV FE models, the state at 1/3 diastole is assumed to be the closest thing to an unloaded state or "reference configuration" that occurs in vivo. In truth, the adult LV has a natural distribution of residual stresses that are formed in the process of organ morphogenesis, and which have been measured in experimental studies [72]. Incorporation of these stresses into the model would change the through-wall distribution of stress, and would thus have some impact on the distribution of growth, most likely resulting in a more uniform distribution of growth than was indicated in the present study.

Chapter 4. The Effect of Changes in Systolic Pressure and Material Stiffness on Growth Modeling Results

4.1: Introduction

In Chapter 3, systolic work was chosen as a potential growth stimulus for the case of hypertension because it is a mechanical variable that captures the total increase in energy expended during systole that occurs as a result of systolic pressure increases. The results of the previous chapter demonstrated that the use of systolic work as the growth stimulus yielded LV wall thickening values that were realistic, using literature values for the systolic pressure for the SHR. The model was constructed such that the greater the systolic pressure (over a non-hypertensive baseline value), the greater the hypertrophy. However, the exact relationship between systolic pressure and the resulting wall thickness had yet to be quantified. The first goal of this study was to quantify the relationship between systolic pressure and the model predicted LV wall thickness, holding all other variables constant. A second mechanical variable that is know to change in hypertensive hypertrophy, and which also affects the systolic work, is the material stiffness. A change in material stiffness will cause a change in the LV workload when volumes are maintained because the required active contraction stress will change. The second goal of this study was to quantify the relationship between the material stiffness, and the resulting systolic work and LV wall thickness. In any model study, it is necessary to quantify how changes in relevant variables, within an appropriate range, will affect the final results. In

this chapter, two important variables affecting the results will be assessed: systolic pressure, and material stiffness.

Systolic pressure increase is what defines the change in left ventricular loading in hypertension. The medical definition of hypertension is a brachial systolic pressure of over 140 and a brachial diastolic pressure of over 80. Systolic brachial pressure measurements are a good estimate of systolic left ventricular pressure. There are four different medically classified stages of hypertension which were previously described (Table 1-1). Hypertrophy has been observed experimentally in patients with all of these stages of hypertension, particularly stages II and III. There is a huge degree of variability in human studies of hypertrophy in hypertensive patients due to large number of confounding variables. For example, many people with hypertension also have other health issues that may or may not be related to the hypertension, there is huge variation in lifestyle factors which can affect outcomes, and genetic variation also plays a role. For this reason, animal models are a useful way to study the disease. In animal studies, the groups of animals are bred to have similar genetic background, and lifestyle conditions are controlled. This removes many confounding variables. The SHR is a well-studied animal model of hypertension, and will be used again in the present study. While blood pressure measurements were not made in the experimental study that provided the images used in finite element model construction in this work, other studies have measured this. Bing et al.measured systolic pressures of 191+/-11 mmHg in failing SHRs and 120+/-12 mmHg in non-failing SHRs, compared with normal control WKY rats which had pressures of 125+/-5 mmHg {Bing, 1995 #36}. If the average values for the SHR were placed in the categories for humans, they would be representative of Stage III hypertension. The range of hypertensive systolic pressure values to

chosen for this study was 160 to 200 mmHg because it encompases stage II and III hypertension in humans, and captures the average measured values observed in the SHR.

Fibrosis (collagen stiffening) is a common physiological affect associated with hypertensive growth and remodeling. Fibrosis refers to the presence of excessive interstitial collagen in the myocardium. In histological studies, the fraction of collagen per area in subjects with hypertensive hypertrophy has been shown to be double that of normal subjects [47]. This increased collagen fraction results in differences in the macroscopic mechanical properties. Experiments on excised myocardial tissue have shown that the stiffness in subjects with hypertensive hypertrophy is approximately twice that measured in normal subjects [47]. This increase in material stiffness will alter the LV mechanics. One would expect two primary changes in mechanics in order to meet the required SV with a stiffer myocardial material: (1) an increase in diastolic pressure will be necessary to maintain diastolic volume (EDV), and (2) an increase in systolic active contraction stress will be necessary to maintain systolic volume (ESV). An increase in the active contraction stress will increase the systolic work. Using systolic work as the growth stimulus, an increase in material stiffness will cause an increase in the model-predicted LV wall thickening.

The purpose of the model study presented in this chapter is to quantify the range of model predicted LV wall thickness as a function of changes in systolic pressure and material stiffness. The range of systolic pressures and material stiffness evaluated in this study were chosen to be representative of the range that have been measured experimentally in hypertensive hypertrophy.

4.2 Methods

4.2.1 Systolic Pressure

The model was evaluated for a range of peak systolic pressures from 160 to 200 mmHg. The relationship between the applied systolic pressure, the resulting systolic work, and the predicted LV wall thicknesses were assessed. In Chapter 3, the threshold value of systolic work that defines the "non-hypertensive" mechanical state was based on a systolic pressure of 140 mmHg. The same value was used in the current study for consistency. The tolerance on the SV used in the calculation of active contraction necessary to maintain SV was set at 0.1% of the SV.

4.2.2 Material Stiffening

Material Parameters

Material parameters C_1 and C_2, which represent the stiffness of the hyperelastic base material in the transversely isotropic Mooney-Rivlin constitutive model, were increased in order to approximate the material stiffening observed in hypertensive hypertrophy. In the Mooney-Rivlin base material, C_1 scales the first deviatoric invariant, and C_2 scales the second deviatoric invariant.

In the Test Case model in Chapter 3, C_1 was 2.325 kPa, and C_2 was .2325 kPa. These two parameters were scaled up by what will be called the *stiffness factor* (SF). The growth model was evaluated for SF values of 1.5, 2.0, and 2.5.

CGR Code Addition

Increasing the material stiffness will change the diastolic mechanics. In order to compare wall thickening results from growth models, the EDV and ESV should be the same for all of the

models so that SV is maintained. Furthermore, experimental studies show that SV is maintained during the time course of stable concentric hypertrophy, as was observed in Chapter 2. In Chapter 3, the code incorporated a secant method root-finding algorithm to determine the active contraction necessary to maintain the ESV. For this part of the study, a similar iterative procedure was incorporated in order to determine the EDP necessary to maintain the EDV. That way, the systolic contraction will begin at the same level of diastolic stretch, and the systolic work can be compared between models. The tolerance on the EDV was set at 0.05% of the EDV. See Appendix A for code implementation details. The LV volumes used in the model were: EDV = 460 mm^3, and ESV = 195 mm^3, which results in SV = 265 mm^3, and EF = 0.58.

Analysis

A table was constructed showing the relationship between stiffness factor and change in EDP. The change in EDP was calculated as the difference between the model EDP and EDP_0 divided by EDP_0, where EDP_0 is the EDP for the original material properties. The relationship between stiffness factor and systolic work was evaluated for a series of models with systolic pressures of 160 mmHg, as well as a series of models with systolic pressures of 200 mmHg. The relationship between stiffness factor and LV wall thickness was evaluated for a series of models with a systolic pressure of 160 mmHg.

4.3 Results

4.3.1 Systolic Pressure

Increased systolic pressure results in increased systolic work (Figure 4-1) with a trend that is linear ($R^2 = 0.9999$). Average element systolic work for pressures of 160, 180, and 200 mmHg were 13.5%, 27.4%, and 41.3% greater than the systolic work for the 140 mmHg model.

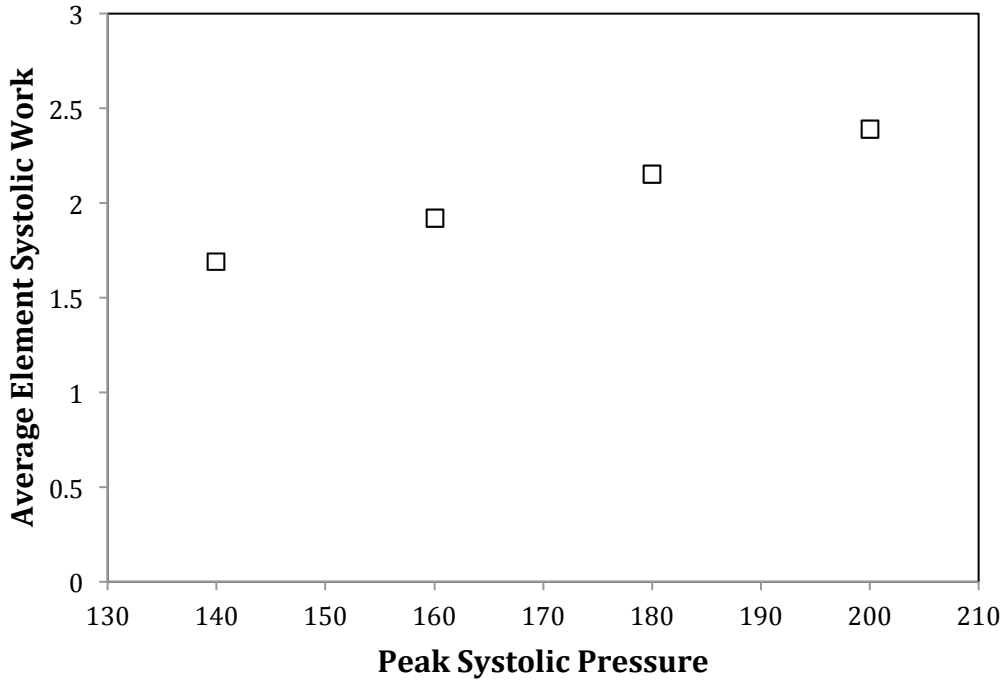

Figure 4-1: Average element systolic work vs. peak systolic pressure. Increasing pressure causes an increase in the systolic work.

Increased systolic pressure results in increased LV wall thickness values (Figure 4-2). Model predicted LV wall thickness for pressures of 160, 180, and 200 mmHg were 10%, 24%, and 51% greater than the original LV wall thickness for the 140 mmHg model. The slope of the curve increases with increasing pressure. In otherwords, incremental increases in systolic workload produced greater increases in LV wall thickness for higher pressure loads.

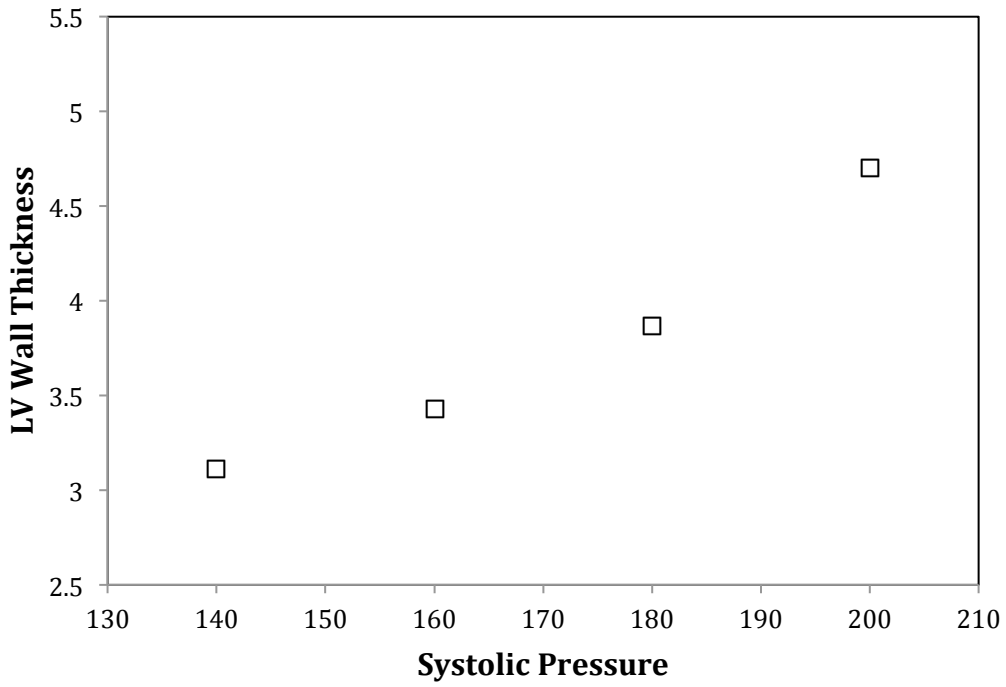

Figure 4-2: LV wall thickness vs. peak systolic pressure. The 140 mmHg model was the non-hypertensive control.

4.3.1 Material Stiffness

The change in end-diastolic pressures necessary to maintain filling volumes was determined for a range of material stiffness factors (Table 4-1). Doubling the stiffness factor caused an 86% increase in the EDP necessary to maintain filling volumes.

Table 4-1: Stiffness Factor and Percent Change in EDP

Stiffness Factor	Percent Change in EDP (mmHg)
1.5	43%
2.0	86%
2.5	128%

The effect of increasing material stiffness on average element work was evaluated for a series of models with ESP=160 mmHg as well as for a series of models with ESP=200 mmHg (Figure 4-2). In both series of models, increasing stiffness resulted in increasing systolic work, in a relationship that appears fairly linear (R^2=0.997 for the 160 series and R^2=0.987 for the 200 series). For the 160 series, stiffness factors of 1.5, 2.0, and 2.5 resulted in systolic work increases of 4.5%, 10%, and 16%. For the 200 series, stiffness factors of 1.5, 2.0, and 2.5 resulted in systolic work increases of 3.0%, 7.5%, and 12%.

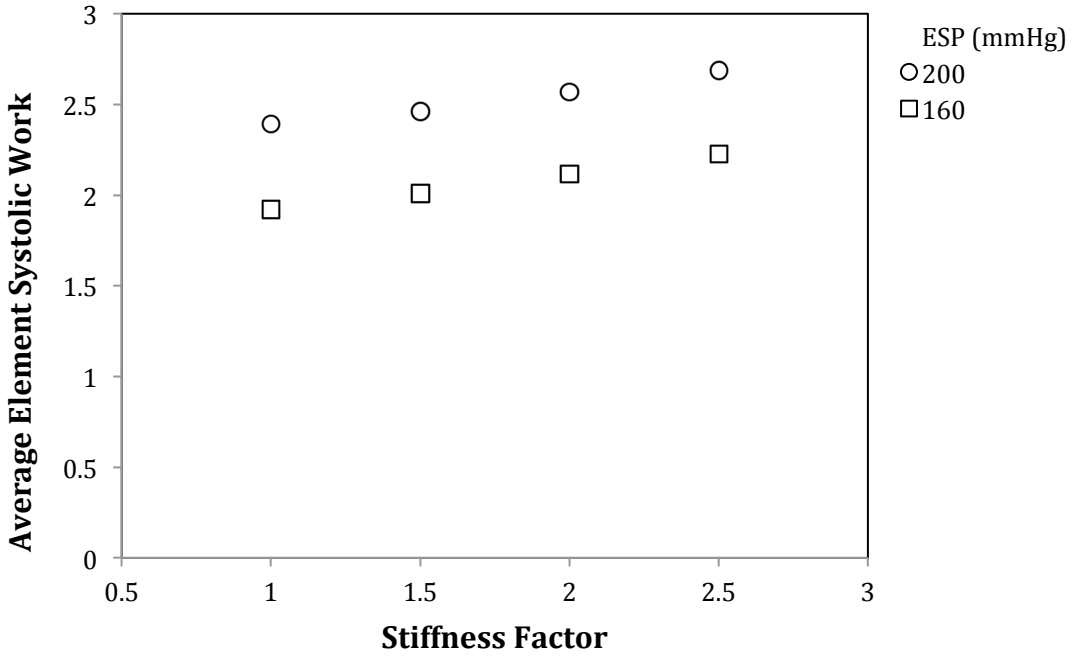

Figure 4-3: Average element systolic work vs. stiffness factor. Two series of results are shown: circles represent models with peak systolic pressure of 200 mmHg, squares represent models with peak systolic pressures of 160 mmHg.

Material stiffening caused an increase in the model predicted LV wall thickness (Figure 4-4) with a trend that increases in slope with increasing stiffness factor. Stiffness factors of 1.5, 2.0, and 2.5 resulted in wall thickness increases of 0.9%, 4.7%, and 14%.

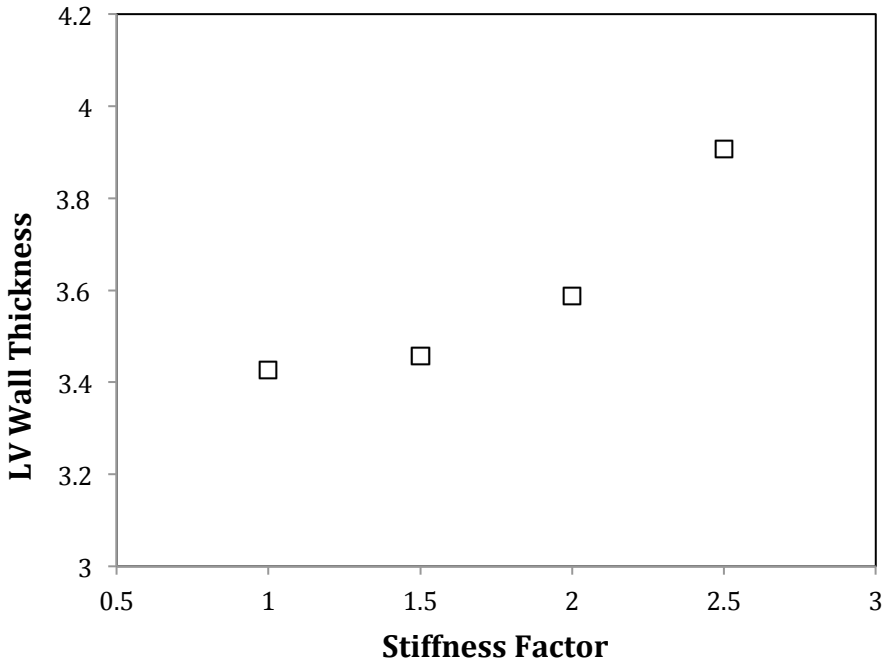

Figure 4-4: LV wall thickness vs. stiffness factor for ESP=160mmHg.

4.4 Discussion

4.4.1 Systolic Pressure

The average element systolic work in the model was intended to represent a measure of the workload being exerted by the muscle tissue in vivo, for each cardiac cycle, per gram of tissue. The linearity of average element systolic work vs. peak systolic pressure is expected because the applicable aspects of the mechanics are similar to that of the simple theory of cylindrical of pressure vessels, in which wall stress is linearly related to the applied pressure. Small amounts of error in the calculated values were due to the tolerance value on the SV, which was used in the iterative procedure that determines the active contraction necessary to maintain SV. An increase in tissue workload of 41%, as was observed in this study, between an LV with 140 mmHg peak systolic pressure and an LV with 200 mmHg peak systolic pressure represents a

very serious difference with regard to the physiology. That degree of increase in energy demand may explain why the SHR has been shown to have changes in the oxygen cost of stress development {Brooks, 1993 #103} and metabolic substrate utilization {Hernandez, 2013 #262}. Quantifying the mechanical relationship between systolic pressure increases and tissue workload is an important step in understanding the underlying mechanisms of pressure overload hypertrophy. In future studies, the increases in tissue workload should be related to studies of metabolic substrate utilization and other aspects of the physiology as this may give insight into the mechanisms of adaptation present in the disease.

The model developed in this project was created to determine the amount of LV wall thickening necessary to reduce the average element systolic work to a predetermined threshold level representing a non-hypertensive state. Because the relationship between applied peak systolic pressure and the resulting average element systolic work was linear, but the relationship between systolic pressure and growth was nonlinear, one can deduce that the relationship between workload and growth is nonlinear. The slope of the curve increases, such that incremental increases in workload will cause more LV wall thickening at higher pressures. This effect is due to the geometry of the left ventricular model. Quantifying this relationship is important for understanding the implications of choosing systolic workload as the growth stimulus.

In this model, a single value of systolic pressure served as the threshold between non-hypertensive and hypertensive states. In reality, there is no absolute threshold for pressure above which hypertrophy will occur. There is considerable variability between individuals {Jones, 2004 #391}. Additionally, the model was created such that the total degree of hypertrophy was defined as a function of the difference between the hypertensive and normotensive workload. In other

words, the model was created such that the greater the pressure, the greater the workload, and the greater the resulting LV wall thickness. While this is an appropriate first step for a model study, it does not represent all of the variables involved in the true physiology. In reality, the degree of hypertrophy does not necessarily scale directly with the systolic pressure. The incidence of hypertrophy has been found to change with blood pressure categories [42]. For pressures defined as stage I hypertension or stage II hypertension, the incidence of hypertrophy does increase with blood pressure [42]. However, patients with stage III hypertension were found to have less hypertrophy compared to those with stage II hypertension [42]. This suggests that increases in loads up to a certain point will cause hypertrophy, but beyond that, extremely high loads will injure the myocytes, preventing hypertrophy. The damage mechanisms present in the higher stages of hypertension are not considered in the current growth model implementation. Future work in models of pressure overload hypertrophy should begin to incorporate some information about the limits of hypertrophy that are observed experimentally, in order to create a more realistic relationship between the workload and the resulting LV thickness.

4.4.2 Material Stiffening

Increases in the stiffness of the hyperelastic base material used in the transversely isotropic constitutive model are intended to represent the changes in macroscopic material behavior that occur as a result of fibrosis in hypertensive hypertrophy. Increases in stiffness of 2x have been observed in animal studies of hypertensive hypertrophy. In this model study, the effects of increased stiffness on EDP, work, and wall thickness were evaluated.

EDP was increased by 86% as a result of doubling the stiffness. This represents a large change in the physiological hemodynamics. Increases in diastolic ventricular pressures have been

observed experimentally in SHR subjects. Bing et al reported pressure tracings showing failing SHRs to have an EDP of approximately 8 mmHg and non-failing SHRs to have an EDP of approximately 5 mmHg {Bing, 1995 #36}. While the study did not show EDP for the normal WKY controls, normal values of EDP are generally less than that {Kostis, 2003 #79}. This suggests that failing ventricular function is associated with increased diastolic pressures, which may be caused by increased material stiffness while the LV attempts to maintain SV.

Systolic work was increased by 10% as a result of doubling the stiffness for a model with ESP=160. A 10% increase in workload due to stiffening, in addition to the increased workload due to the systolic pressure overload, could have physiological consequences. Additionally, the affect of material stiffening on the workload was slightly different at a higher pressure. Doubling the stiffness caused an increase in systolic workload of only 7.5% for a model with ESP=200. This study quantified some salient points of the relative effects of systolic pressure and material stiffening on the LV workload. Tolerance values in this study were set small enough to minimize the potential for error. In the material stiffening study, the model results are affected by both the tolerance on SV as well as the tolerance on EDV.

Wall thickness increases with increasing stiffness showed a trend in which the slope of the curve increased with increasing stiffness. This demonstrated that a change in the mechanics that causes an increase in systolic work will cause a change in model predicted wall thickness with this trend.

4.4.3 Further Discussion and Future Directions

In this model study, the increased stiffness and the increased pressure were evaluated separately in order to assess the individual affects on the systolic work. In the physiological range of interest, changes in systolic pressure have a much greater impact on the ventricular workload. However, in vivo, the collagen deposition that causes material stiffening occurs simultaneously with the development of myocyte hypertrophy, as both the fibroblasts and the myocytes respond to the in-vivo mechanical environment of pressure overload. This means that, in addition to the extra workload caused by hypertension, material stiffening will cause an incremental increase in workload. This suggests that hypertensive hypertrophy could potentially accelerate over time due to the interplay of pressure loading and fibrosis. The combined effects of these two factors are important to understand because they likely play a role in the transition to heart failure.

Future work should consider the interplay of these variables further. It would be informative to incorporate a closed-loop model of the circulation, rather than simply forcing the model to maintain stroke volume, since this would have an impact on the relationship between material stiffening, diastolic pressures, and the resulting workload.

One key area for improvement for this type of model is to develop a growth law that captures the damage mechanisms present at the higher levels of hypertrophy. This would entail incorporating empirical information from experimental studies of muscle tissue. Doing so would enable the model to describe a wider range of behavior, not only for hypertensive hypertrophy, but also for other cardiac growth and remodeling applications.

Another aspect of the growth model that may be useful to pursue further is the capability to incorporate information from throughout the cardiac cycle into the growth law. This means that as additional experimental data becomes available, it would be possible to incorporate more of the physiological information into the growth law. For example, the growth law could cause fiber direction growth as a function of diastolic strain, in addition to the sheet direction growth as a function of systolic work.

As mentioned earlier, systolic work is an intriguing choice for the growth stimulus in cases of pressure overload because it captures the increased energy demand. Future work should further explore the connection between pressure overload induced changes in mechanical energy demand and changes in metabolism.

Hypertensive hypertrophy causes symmetric hypertrophy over the whole LV, but it would be interesting to extend the model to study cases of asymmetric remodeling. Asymmetric hypertrophy in the cases of left bundle branch block could be modeled by making use of the active contraction time delay feature that was incorporated. Asymmetric growth and remodeling caused by myocardial infarction could be modeled by defining an area with zero or decreased active contraction.

Finally, this study completely neglected the time course of the development of hypertrophy. Future work in ventricular growth modeling should begin to consider the many variables that affect the time course of the development of hypertrophy. This will be necessary in order to achieve the ultimate goal of growth and remodeling research, which is to be able to predict the long term affects of treatment strategies.

Chapter 5. Appendices

Appendix A: CGR Code

A.1: Main.cpp

```
////////////////////////////////////////////////////////////////////
//                                                                //
//      C A R D I A C   G R O W T H   &   R E M O D E L I N G     //
//                                                                //
////////////////////////////////////////////////////////////////////



#include <iostream>
#include <stdio.h>
#include <stdlib.h>
#include <string>
#include <vector>
#include <math.h>

#include "create_feb.h"
#include "create_feb1.h"
#include "create_feb2.h"
#include "create_feb3.h"
#include "get_line.h"
#include "output.h"
#include "hello.h"
#include "settings1.h"
#include "read_input.h"
#include "DataAnimal.h"


using namespace std;
```

```
/// Global variables

SETTINGS1 set;      // global settings variable
FILE* fpout;
FILE* fpoutR;       // this is for the output
//char outfile[256];// ouput filename
char nfefile[256];  // FE file
char ngfile[256];   // displacement / growth file
char nxyzfile[256];
char nwfile[256];
char runstr[256];   // FE model run command
int c;              // number of remodeling runs



/// Main entry point of application

int main(int nargs, char* argv[])
{

    // check nargs
    if ((nargs != 2) && (nargs != 3))
    {
      hello(fpout);
      printf("\n incorrect arguements \n\n\n");
      return 0;
    }



    // read input file
    if (read_input(argv[1], set) == false)
    {
      output("ERROR : An error occured reading input file\n", fpout);
      return 0;
    }



    fpout = fopen(set.logfile, "w");
    if (fpout == 0) printf(" WARNING : logfile could not be created. Output
will not be saved\n");
    char str[256];// this is a string that I'll use to print stuff to the
logfile

    fpoutR = fopen(set.resultsfile, "w");
    if (fpoutR == 0) printf(" WARNING : results file could not be created.
Output will not be saved\n");
    sprintf(str,"AvgWork Over-Work T_grown\n"); output(str, fpoutR);

    // print welcome message
    hello(fpout);
```

```
    // print out what we got from the input file
    output("Input Data:\n\n", fpout);
    sprintf(str, "\tFEBio Input Filename ....... : %s\n", set.febfile);
output(str, fpout);
     sprintf(str, "\tNumber of Nodes in Model ... : %d\n", set.n);
output(str, fpout);
     sprintf(str, "\tTimes (Edia Esys Erel Egr).. : %1.1f %1.1f %1.1f %1.1f
\n",
                    set.times[0],set.times[1],set.times[2],set.times[3]);
output(str, fpout);
     sprintf(str, "\tSystem Flag................. : %s\n", set.flag);
output(str, fpout);
     sprintf(str, "\tWall Thickness Tolerance.... : %1.2f * Original_Value\n",
set.t_tol);    output(str, fpout);
    sprintf(str, "\tStroke Volume Tolerance..... : %1.2f * Original_Value\n",
set.sv_tol);   output(str, fpout);
    sprintf(str, "\tNodal Coordinates Filename . : %s\n", set.nodefile);
output(str, fpout);
    sprintf(str, "\tDisplacement Data Filename . : %s\n", set.dispfile);
output(str, fpout);
    sprintf(str, "\tFiber Definitions Filename . : %s\n",
set.fiberfile);output(str, fpout);
    sprintf(str, "\tWork Data Filename ......... : %s\n\n", set.workfile);
output(str, fpout);



    // declare some more stuff to be used later
    float t_ref0, SV0;
    char nikefile[256];

    bool got_correct_t = false; //this becomes true when it grows big enough
    //bool overshot_t; // I'm not actually using this yet
    bool forward_only; // when this is true we know it is only a forward
model run, not growth



     // Declare stuff
     DataAnimal postprocessor;
     float t_ref, t_grown;
     float rcv, edv, esv, grv;
     float SV, EF;
     float AvgWork, maxWork, minWork;
     double SRSSdiffWork = 0; double AvgWorkDiff = 0;
     float ac_start;
     float alpha;


    // this is the remodeling loop
    for ( c=1 ; got_correct_t == false ; c++ )
    {
```

```
        output("_____\n\n",
fpout);
        sprintf(str,"\nGrowth Loop # %d \n\n",c);
      output(str, fpout);



        /// New addition of loop on diastole to obtain the EDP that creates
the "right" EDV

        bool got_correct_edp = false;
        float correctEDP;


        // set filenames for this loop
        //if (c==0 && strcmp(set.flag, "osx") == 0)
        //{
        //    sprintf(nfefile, "%s%s%s",
"/Users/genevievefarrar/Desktop/CGR1/", set.febfile, ".feb");
        //    sprintf(nxyzfile,"%s%s%s",
"/Users/genevievefarrar/Desktop/CGR1/", set.nodefile,".txt");
        //}
        //if (c!=0 && strcmp(set.flag, "osx") == 0)
        if (strcmp(set.flag, "osx") == 0)
        {
            sprintf(nfefile, "%s%s%03i%s",
"/Users/genevievefarrar/Desktop/CGR1/", set.febfile,  c, ".feb");
            sprintf(nxyzfile,"%s%s%03i%s",
"/Users/genevievefarrar/Desktop/CGR1/", set.nodefile, c, ".txt");
        }

        // declare stuff for edp-edv loop
        std::vector<float> d_edv;
        std::vector<float> d_edp;
        d_edp.push_back(0.1);
        std::vector<float> edp;

        // estimate an ititial value of edp, EDP_start
        edp.push_back((set.dstiff)*(set.EDP0)); //(proportional for thin wall
tube, lin elast. just a starting guess.)

        // This is the loop
        int j;
        for (j = 0; got_correct_edp == false; j++)
        {
            cout << "\n j=" << j << "\n";

            // create input file
            if(create_feb1( set, c, edp[j] ) == false)
            {
                output("ERROR : An error occured in create_feb1 \n", fpout);
```

```
                return 0;
            }

            // Run it
            if (strcmp(set.flag,"osx") == 0)
            {sprintf(runstr,
"/Users/genevievefarrar/Desktop/GRFebio/bin/grfebio.osx -i %s -noconfig",
nfefile);}
            system(runstr);

            cout << "\n in main, nodefile = nxyzfile =" << nxyzfile << "\n";


            // calc EDV
            if (postprocessor.read_nodefile(nxyzfile, 1.5) == false)
            {
                output("Error reading nodefile at 1.5 \n",fpout);
                fclose(fpout);
                return false;
            }
            edv = postprocessor.calc_volume(set);

            // determine whether the edp used in this loop resulted in an
acceptable edv
            d_edv.push_back( edv - set.EDV0 ); // this is d_edv[j]

            // if edv is in the acceptable range
            if( d_edv[j] < set.edv_tol*set.EDV0  &&  d_edv[j] > -
1*(set.edv_tol*set.EDV0) )
            {
                got_correct_edp = true;
                edp.push_back( edp[j] );
                correctEDP = edp[j];
            }

            // if edv is NOT in the acceptable range
            else
            {
                if (j > 0)
                {
                    d_edp.push_back( (d_edp[j-1] * d_edv[j]) / (d_edv[j-1] -
d_edv[j]) ); // this is d_edp[j]
                }
                if( d_edv[j] > set.edv_tol*set.EDV0 ) {edp.push_back( edp[j]
- d_edp[j] );}// ac[j+1]
                if( d_edv[j] < -1*(set.edv_tol*set.EDV0) ) {edp.push_back(
edp[j] + d_edp[j] );} // ac[j+1]
            }

            // Output diastole loop results
            sprintf(str,"\t\tDiastole Loop Results for c = %d, j = %d\n\n",
c,j); output(str, fpout);
            sprintf(str,"\t\t\tEDV..... : %3.1f\n", edv); output(str,
fpout);
            sprintf(str,"\t\t\tEDV - EDV0.. : %2.3f \n", d_edv[j]);
```

```
output(str, fpout);
            sprintf(str,"\t\t\tEDV_tol..... : %2.3f \n", set.edv_tol*set.EDV0
); output(str, fpout);
            sprintf(str,"\t\t\tlast EDP.... : %3.3f \n", edp[j] );
output(str, fpout);
            sprintf(str,"\t\t\tnext EDP.... : %3.3f \n\n", edp[j+1] );
output(str, fpout);


        }

        sprintf(str,"\t\tDiastole Loop Results for edp[0] = %3.3f\n",
edp[0]); output(str, fpout);
        sprintf(str,"\t\tDiastole Loop Results for edp[1] = %3.3f\n",
edp[1]); output(str, fpout);
        sprintf(str,"\t\tDiastole Loop Results for edp[2] = %3.3f\n\n",
edp[2]); output(str, fpout);


        // put the correct EDP into the original feb to be used in the
remodeling loop
        if(create_feb2( set, c, correctEDP ) == false)
        {
            output("ERROR : An error occured in create_feb2 \n", fpout);
            return 0;
        }




        // this is a flag for the AC loop
        bool got_correct_AC = false;


      // set filenames for this loop
      if (c==0 && strcmp(set.flag, "lnx") == 0)
      {
          sprintf(nfefile, "%s%s", set.febfile,".feb");
          sprintf(ngfile, "%s%s", set.dispfile,".txt");
            sprintf(nxyzfile, "%s%s", set.nodefile,".txt");
            sprintf(nwfile, "%s%s", set.workfile,".txt");
      }
      if (c!=0 && strcmp(set.flag, "lnx") == 0)
      {
            sprintf(nfefile, "%s%03i%s", set.febfile, c, ".feb");
            sprintf(ngfile, "%s%03i%s", set.dispfile, c, ".txt");
            sprintf(nxyzfile, "%s%03i%s", set.nodefile, c, ".txt");
            sprintf(nwfile, "%s%03i%s", set.workfile, c, ".txt");
      }

        if (c==0 && strcmp(set.flag, "osx") == 0)
      {
          sprintf(nfefile, "%s%s%s", "/Users/genevievefarrar/Desktop/CGR1/",
set.febfile, ".feb");
          sprintf(ngfile,  "%s%s%s", "/Users/genevievefarrar/Desktop/CGR1/",
```

```
set.dispfile,".txt");
            sprintf(nxyzfile,"%s%s%s",
"/Users/genevievefarrar/Desktop/CGR1/", set.nodefile,".txt");
            sprintf(nwfile,  "%s%s%s",
"/Users/genevievefarrar/Desktop/CGR1/", set.workfile,".txt");
      }
      if (c!=0 && strcmp(set.flag, "osx") == 0)
      {
            sprintf(nfefile, "%s%s%03i%s",
"/Users/genevievefarrar/Desktop/CGR1/", set.febfile,  c, ".feb");
            sprintf(ngfile,  "%s%s%03i%s",
"/Users/genevievefarrar/Desktop/CGR1/", set.dispfile, c, ".txt");
            sprintf(nxyzfile,"%s%s%03i%s",
"/Users/genevievefarrar/Desktop/CGR1/", set.nodefile, c, ".txt");
            sprintf(nwfile,  "%s%s%03i%s",
"/Users/genevievefarrar/Desktop/CGR1/", set.workfile, c, ".txt");
      }


        // declare stuff for the active contraction loop
        std::vector<float> dsv;
        std::vector<float> dac;
        dac.push_back(1.0);
        std::vector<float> ac;
        int i_whengotcorrectac;

        // just use specified ac for c = 0 run
        //if (c == 0) {ac.push_back(set.ac0);}
        if (c == 1) //C1//
        {
            ac_start = set.ac0;
            ac.push_back(set.ac0);
        }

        // for c > 0, calc a new ac_start and make new fiber definitions
        else
        {
            // calc new ac_start
            alpha = (t_grown - set.t_ref0)/set.t_ref0;
            ac_start = (set.ac0)*(1 - alpha*alpha*alpha);
            ac.push_back(ac_start);


            sprintf(str,"\t\t t_grown : %2.3f\n", t_grown); output(str,
fpout);
            sprintf(str,"\t\t t_ref0 : %2.3f\n", set.t_ref0); output(str,
fpout);
            sprintf(str,"\t\t alpha : %2.3f\n", alpha); output(str, fpout);
            sprintf(str,"\t\t ac_start : %2.3f\n", ac_start); output(str,
fpout);


            /// Make New Fiber Definitions

            // read the nodal locations of the grown state (t=4.0) from the
```

```
last run
            postprocessor.read_nodefile(nxyzfile, 4.0);
            // put the nodal locations into nike format because that's what
fibdeft needs as input
            if (strcmp(set.flag,"osx") == 0) sprintf(nikefile,
"%s%s","/Users/genevievefarrar/Desktop/fibdeft/", set.nikefile);
            if (strcmp(set.flag,"lnx") == 0) sprintf(nikefile,
"%s%s","/home/gfarrar/fibdeft/", set.nikefile);
            postprocessor.updateNikefile(nikefile);
            // run fibdeft
            if (strcmp(set.flag,"osx") == 0) sprintf(runstr,
"/Users/genevievefarrar/Desktop/fibdeft/fibdeft10a.osx
/Users/genevievefarrar/Desktop/fibdeft/6Gnewref.inp");
            if (strcmp(set.flag,"lnx") == 0) sprintf(runstr,
"/home/gfarrar/fibdeft/fibdeft10a.lnx /home/gfarrar/fibdeft/6Gnewref.inp");
            system(runstr);
        }



        // This is the active contraction loop
        int i;
        for (i = 0; (got_correct_AC == false || (got_correct_AC == true &&
i==i_whengotcorrectac+1 && c > 0)) && c<15; i++)
        {
            //if(c==0) {got_correct_AC = true;} //C1// commented this out


            /// Create New GRFEBio Input File

            if(c == 1) // then need to create feb. //C1// added this part
            {
                if (got_correct_AC == false) // then create_feb with end time
at 3 (forward only)
                {
                    forward_only = true;

                    if(i == 0) // then use ac_start
                    {
                        if(create_feb3( set, c, ac_start, 12 ) == false)
                        {
                            output("ERROR : An error occured in create_feb
\n", fpout);
                            return 0;
                        }
                    }
                    else if( i > 0 ) // use ac[i]
                    {
                        if(create_feb3( set, c, ac[i],  12 ) == false)
                        {
                            output("ERROR : An error occured in create_feb
\n", fpout);
                            return 0;
                        }
                    }
```

```
                }
                else // if got_correct_AC = true, and c > 0, and
got_correct_t == false,
                    // then create_feb for growth run
                {
                    forward_only = false;
                    if(create_feb3( set, c, ac[i], 14 )  == false)
                    {
                        output("ERROR : An error occured in create_feb \n",
fpout);
                        return 0;
                    }
                }
            }



            //if(c > 0) // then need to create feb.
            if(c > 1) // then need to create feb. //C1//
            {
                if (got_correct_AC == false) // then create_feb with end time
at 3 (forward only)
                {
                    forward_only = true;

                    if(i == 0) // then use ac_start
                    {
                        if(create_feb( set, c, ac_start, 12 ) == false)
                        {
                            output("ERROR : An error occured in create_feb
\n", fpout);
                            return 0;
                        }
                    }
                    else if( i > 0 ) // use ac[i]
                    {
                        if(create_feb( set, c, ac[i],  12 ) == false)
                        {
                            output("ERROR : An error occured in create_feb
\n", fpout);
                            return 0;
                        }
                    }
                }
                else // if got_correct_AC = true, and c > 0, and
got_correct_t == false,
                // then create_feb for growth run
                {
                    forward_only = false;
                    if(create_feb( set, c, ac[i], 14 )  == false)
                    {
                        output("ERROR : An error occured in create_feb \n",
fpout);
                        return 0;
                    }
```

```
            }
        }



        /// Run GRFEBio
        if (strcmp(set.flag,"lnx") == 0)
            {sprintf(runstr, "/home/gfarrar/grfebio/bin/grfebio.lnx -i
%s -noconfig", nfefile);}

        if (strcmp(set.flag,"osx") == 0)
            {sprintf(runstr,
"/Users/genevievefarrar/Desktop/GRFebio/bin/grfebio.osx -i %s -noconfig",
nfefile);}

        system(runstr);



        /// If this was ONLY a Forward Model run, then Evaluate Forward
model Results
        // (also do this if it's c=0)

        if( c==0 || forward_only==true )
        {

            /// Calculate Forward Model Results

            //EDV
            if (postprocessor.read_nodefile(nxyzfile, 1.5) == false)
            {
                output("Error reading nodefile at 1.5 \n",fpout);
                break;
            }
            edv = postprocessor.calc_volume(set);

            //ESV
            if (postprocessor.read_nodefile(nxyzfile, 3.0) == false)
            {
                output("Error reading nodefile at 3.0 \n",fpout);
                break;
            }
            esv = postprocessor.calc_volume(set);

            //Work
            if (postprocessor.read_workfile(nwfile) == false)
            {
                output("Error reading workfile \n",fpout);
                break;
            }
            AvgWork = postprocessor.calc_avgWork();
            maxWork = postprocessor.calc_maxWork();
            minWork = postprocessor.calc_minWork();
```

```
                  //AvgWorkDiff = postprocessor.calc_workAvgDiff();
                AvgWorkDiff = AvgWork-set.w0;

                  SV = edv-esv; if(c ==0) {SV0 = SV;}
                  EF = SV/edv;



                  /// Output Forward Model Results
                  sprintf(str,"\tGRFebio Forward Model Results for c = %d, i
= %d\n\n", c,i); output(str, fpout);

                  sprintf(str,"\t\tEDV.... : %3.1f\n", edv); output(str,
fpout);
                  sprintf(str,"\t\tESV.... : %3.1f\n\n", esv); output(str,
fpout);

                  sprintf(str,"\t\tSV..... : %3.1f\n", SV); output(str,
fpout);
                  sprintf(str,"\t\tEF..... : %1.3f\n\n", EF); output(str,
fpout);

                  sprintf(str,"\t\tAvgWork : %2.3f\n", AvgWork); output(str,
fpout);
                  sprintf(str,"\t\tmaxWork : %2.3f\n", maxWork); output(str,
fpout);
                  sprintf(str,"\t\tminWork : %2.3f\n", minWork); output(str,
fpout);
                  sprintf(str,"\t\tAverage over-work : %2.3f\n\n",
AvgWorkDiff); output(str, fpout);



                // do the active contraction loop when c > 0
                if( c > 0)
                {
                        /// Evaluate SV and Calculate new AC if needed

                        // determine whether the ac used in this loop
resulted in an acceptable SV
                        dsv.push_back( SV - set.SV0 ); // this is dsv[i]


                        // if SV is in the acceptable range
                        if( dsv[i] < set.sv_tol*set.SV0  &&  dsv[i] > -
1*(set.sv_tol*set.SV0) )
                        {
                              got_correct_AC = true;
                        i_whengotcorrectac = i;
                              ac.push_back( ac[i] ); // if  ac[i] was correct
then set the ac for then next run to this one. ac[i+1] =  ac[i]
                        }

                        // if SV is NOT in the acceptable range, calculate
```

```
the new AC based on the SV results
                        else
                        {
                              // dac is the incremental change in ac.
                              // for the first loop there is a small starting
dac, which is set in the .inp file
                              // if i > 0, then calculate a new dac based on
the previous data
                              if (i > 0)
                              {
                                    dac.push_back( (dac[i-1] * dsv[i]) /
(dsv[i-1] - dsv[i]) ); // this is dac[i]
                              }

                              // if ac was too much, subtract dac
                              // ( if dsv is positive, then SV > SV0, and we
want to reduce SV by reducing ac )
                              if( dsv[i] > set.sv_tol*set.SV0 )
{ac.push_back( ac[i] - dac[i] );}// ac[i+1]

                              // if ac was too little, add dac
                              if( dsv[i] < -1*(set.sv_tol*set.SV0) )
{ac.push_back( ac[i] + dac[i] );} // ac[i+1]
                        }


                        /// Output Results from Active Contraction Loop
                        output("\t\tActive Contraction (Loop) Results: \n\n",
fpout);
                        sprintf(str,"\t\t\tAC loop #.. : %d \n\n", i);
output(str, fpout);
                        sprintf(str,"\t\t\tSV - SV0... : %2.3f \n", dsv[i]);
output(str, fpout);
                        sprintf(str,"\t\t\tSV_tol..... : %2.3f \n",
set.sv_tol*set.SV0 ); output(str, fpout);
                        sprintf(str,"\t\t\tlast AC.... : %3.3f \n", ac[i] );
output(str, fpout);
                        sprintf(str,"\t\t\tnext AC.... : %3.3f \n\n", ac[i+1]
); output(str, fpout);
                  }


                  /// If we had the right SV, see if we need to grow
                  if(got_correct_AC == true)
                  {
                        /// Evaluate Growth Condition: choose either (1)wall
thickness or (2)work

                        // option (1): if it didn't get thicker in the last
growth phase, then it's done growing.
                        // (this option basically defers the growth condition
to the growth constitutive model)
                        //if( (t_grown - t_ref) < set.t_tol*t_ref )
{got_correct_t = true;}
```

```
                        // option (2): test if average local work is within
tolerance of normal


                    output("\t Growth Condition Evaluation: \n\n", fpout);
                    sprintf(str,"\t\t w - w0.. : %2.3f \n", AvgWorkDiff);
output(str, fpout);
                    sprintf(str,"\t\t w_tol... : %2.3f \n",
set.w_tol*set.w0); output(str, fpout);


                    // if the "overwork" is within +/- tolerance of zero,
then we have the correct growth and we're done
                        if( (AvgWorkDiff < set.w0*set.w_tol && AvgWorkDiff >
-set.w0*set.w_tol) || (t_grown - t_ref) < set.t_tol*t_ref )
                    {
                        got_correct_t = true;
                        sprintf(str,"\t\t w within w_tol. No need to thicken
LV. \n\n" ); output(str, fpout);

                        break;
                    }
                    // NOTE: The set.w0 is the average element work for the
normotensive (ESP=140) it is 1.591.

                    // if the overwork is negative then we grew too much, and
need to go back
                    if( AvgWorkDiff < -set.w0*set.w_tol )
                    {
                        overshot_t = true; // not actually using this yet
                        got_correct_t = true;
                        sprintf(str,"\t\t w within w_tol. No need to thicken
LV. \n\n" ); output(str, fpout);
                        break;
                    }

                    // if we need to grow more, then the model continues to
grow.
                    if( AvgWorkDiff > set.w0*set.w_tol )
                    {
                        got_correct_t = false;
                        sprintf(str,"\t\t Work is too high. Must thicken the
LV wall. \n\n" ); output(str, fpout);
                    }

                }

            } // end of forward model results evaluation



            /// If this was a Growth run, Evaluate Growth Results
```

```
            if( c==0 || forward_only==false)
            {

                // Evaluate Growth Results
                if (postprocessor.read_nodefile(nxyzfile, 3.5) == false)
                {
                    output("Error reading nodefile at 3.5 \n",fpout);
                    break;
                }
                t_ref = postprocessor.calc_wallThick(); if(c==0) {t_ref0 =
t_ref;}
                rcv = postprocessor.calc_volume(set);


                if (postprocessor.read_nodefile(nxyzfile, 4.0) == false)
                {
                    output("Error reading nodefile at 4.0 \n",fpout);
                    break;
                }
                t_grown = postprocessor.calc_wallThick();
                grv = postprocessor.calc_volume(set);


                  // Output Growth Results
                sprintf(str,"\tGRFebio Growth Results for c = %d, i =
%d\n\n", c,i); output(str, fpout);
                sprintf(str,"\t\tRCV.... : %3.1f\n", rcv); output(str,
fpout);
                sprintf(str,"\t\tGRV.... : %3.1f\n\n", grv); output(str,
fpout);
                sprintf(str,"\t\tt_ref.. : %1.4f\n", t_ref); output(str,
fpout);
                sprintf(str,"\t\tt_grown : %1.4f\n\n", t_grown); output(str,
fpout);

                //this goes to the results file
                sprintf(str,"%2.3f\t %2.3f\t %1.4f %3.3f\n", AvgWork,
AvgWorkDiff, t_grown, ac[i]);
                output(str, fpoutR);

            } // end of growth results evaluation



        } // end of i loop (active contraction loop)



    } // end of c loop (remodeling loop)


  output("\n-----------------------------\n", fpout);
  output(" Done \n", fpout);

  if (fpout) fclose(fpout);
```

```
  return 0;

}
```

A.2: create_feb

create_feb.h

```
#ifndef CREATE_FEB_H_INCLUDED
#define CREATE_FEB_H_INCLUDED

#include <iostream>
#include "get_line.h"
#include "output.h"
#include <stdio.h>
#include <string.h>
#include "settings1.h"

bool create_feb(SETTINGS1 &set, int &c, float &ac, int ts);

#endif // CREATE_FEB_H_INCLUDED
```

create_feb.cpp

```
#include "create_feb.h"

//-------------------------------------------------------------------------
-----
// create_feb : Function to create new febio input file from a template .feb;
//

bool create_feb(SETTINGS1 &set, int &c, float &ac, int ts)
{
    // declare stuff
    char szline[256], line[256], str[256];
    char o_febfile[256]; // original feb file. each time, we copy lines from
the same original file.
    char n_febfile[256]; // new feb file. this is the file i am making.

    // these are the files where i am getting the nodes and fibers to put
into the new feb
    char lastnodefile[256];
    char lastfiberfile[256];

    // these are the filenames to put into the output section of the new feb
    char n_nodefile[256];
    char n_fiberfile[256];
    char n_dispfile[256];
    char n_workfile[256];
```

```
    // set current filenames by loop number
    if ( strcmp(set.flag, "lnx") == 0 )
    {
        sprintf(o_febfile, "%s%s", set.febfile,".feb");
        sprintf(n_febfile , "%s%03i%s", set.febfile , c, ".feb");
        sprintf(n_dispfile, "%s%03i%s", set.dispfile, c, ".txt");
        sprintf(n_nodefile, "%s%03i%s", set.nodefile, c, ".txt");
        sprintf(n_workfile, "%s%03i%s", set.workfile, c, ".txt");
        sprintf(n_fiberfile, "%s%03i%s", set.fiberfile, c, ".txt");
        sprintf(lastfiberfile, "fibers.txt");
        if(c==1) {sprintf(lastnodefile, "%s%s", set.nodefile,".txt");}
        else {sprintf(lastnodefile, "%s%03i%s", set.nodefile, c-1, ".txt");}
    }

    if ( strcmp(set.flag, "osx") == 0 )
    {
        sprintf(o_febfile, "%s%s%s", "/Users/genevievefarrar/Desktop/CGR1/",
set.febfile,".feb");
        sprintf(n_febfile , "%s%s%03i%s",
"/Users/genevievefarrar/Desktop/CGR1/", set.febfile , c, ".feb");
        sprintf(n_dispfile, "%s%s%03i%s",
"/Users/genevievefarrar/Desktop/CGR1/", set.dispfile, c, ".txt");
        sprintf(n_nodefile, "%s%s%03i%s",
"/Users/genevievefarrar/Desktop/CGR1/", set.nodefile, c, ".txt");
        sprintf(n_workfile, "%s%s%03i%s",
"/Users/genevievefarrar/Desktop/CGR1/", set.workfile, c, ".txt");
        sprintf(n_fiberfile,"%s%s%03i%s",
"/Users/genevievefarrar/Desktop/CGR1/", set.fiberfile, c, ".txt");
        sprintf(lastfiberfile,
"/Users/genevievefarrar/Desktop/fibdeft/out.txt");
        if(c==1) {sprintf(lastnodefile, "%s%s%s",
"/Users/genevievefarrar/Desktop/CGR1/", set.nodefile,".txt");}
        else {sprintf(lastnodefile, "%s%s%03i%s",
"/Users/genevievefarrar/Desktop/CGR1/", set.nodefile, c-1, ".txt");}
    }



    /// this is to set some landmarks in the feb file
    // initialize landmarks
    //int cs=0; int ce=0;
    //int ms=0; int me=0;
    int ns=0; int ne=0;
    int es=0; int ee=0;
    int ls=0; //int le=0;
    int os=0; //int oe=0;
    int ss=0; //for step start

    // open original feb file
    FILE* ofeb = fopen(o_febfile, "r");
    if (ofeb == 0) return false;

    // set landmarks
```

```
for (int i=1; fgets(szline, 256, ofeb); i++)
{
    sscanf(szline, "\t%s %*s", str);

    if (!strcmp(str,"<Nodes>"))     {ns = i;} // node deck starts
    if (!strcmp(str,"</Nodes>"))    {ne = i;} // node deck ends
    if (!strcmp(str,"<ElementData>")){es = i;}
    if (!strcmp(str,"</ElementData>")){ee = i;}
    if (!strcmp(str,"<LoadData>")){ls = i;}
    //if (!strcmp(str,"</LoadData>")){le = i;}
    if (!strcmp(str,"<logfile>"))   {os = i;}
    //if (!strcmp(str,"</logfile>"))  {oe = i;}
    if (!strcmp(str,"</Output>"))
    {
        ss = i;
        break;
    }
}
fclose(ofeb);



/// this is to determine the last converged time point
float lct = 0; //the last converged time point in the previous febio run

// open nodal coordinates file
FILE * lnf = fopen(lastnodefile, "r");
if (lnf == 0) return false;

// find last time point
char label[256]; float value = 0;
for (int i=1; fgets(szline, 256, ofeb); i++)
{
    sscanf(szline, "%s %*s %f", label, &value );
  if ( strcmp(label,"*Time") == 0 && value > lct)
  {
      lct = value;
  }
}
fclose(lnf);



/// now reopen the files and get to the right place in them
 // re-open original febio file
ofeb = fopen(o_febfile, "r");

// create and open new febio file
FILE * nfeb = fopen(n_febfile, "w");
if (nfeb == 0) return false;

// open fiber definitions file
FILE * lff = fopen(lastfiberfile, "r");
if (lff == 0) return false;
```

```
    //get past the first (header) line
    fgets(szline, 256, lff);

    // open nodal coordinates file
    lnf = fopen(lastnodefile, "r");

    // get to the last time point in the nodal coordinates file
    while (fgets(szline, 256, lnf))
    {
        char label[256]; float value;
        sscanf(szline, "%s %*s %f", label, &value );
      if ( strcmp(label,"*Time") == 0 && value == lct)
        {
            fgets(szline, 256, lnf);
            break;
        }
    }



    /// make the new FEBio input file
    for (int i=1; fgets(szline, 256, ofeb); i++)
    {


        // update nodal coordinates
        if (i>ns && i<ne)
        {
            // the current node number
            int n = i-ns;

            //float xvalo = 0; float yvalo = 0; float zvalo = 0;
            //float xvaln = 0; float yvaln = 0; float zvaln = 0;
            //char str1[25] = "0"; char str2[25] = "0"; char str3[25] = "0";
            //char printstr[256] = "0";

            float xvalo, yvalo, zvalo;
            float xvaln, yvaln, zvaln;
            char str1[25], str2[25], str3[25];
            char printstr[256];

            // get what's currently on this line of the original feb
            if (n < 10) {sscanf(szline,"%5s %7s%15e,%15e,%15e
%11s",str1,str2,&xvalo,&yvalo,&zvalo,str3);}
            if (n >= 10 && n < 100) {sscanf(szline,"%5s %8s%15e,%15e,%15e
%11s",str1,str2,&xvalo,&yvalo,&zvalo,str3);}
            if (n >= 100 && n < 1000) {sscanf(szline,"%5s %9s%15e,%15e,%15e
%11s",str1,str2,&xvalo,&yvalo,&zvalo,str3);}
            if (n >= 1000 && n < 10000) {sscanf(szline,"%5s
%10s%15e,%15e,%15e %11s",str1,str2,&xvalo,&yvalo,&zvalo,str3);}

            // read new coordinates from nodefile
            fgets(line, 256, lnf);
            int nnf;
            sscanf(line,"%d %e %e %e",&nnf,&xvaln,&yvaln,&zvaln);
```

```
            if (n!=nnf) break;

            // print the modified line to the new file
            sprintf(printstr,"\t\t\t%s %s%+15.7e,%+15.7e,%+15.7e %s
\r\n",str1,str2,xvaln,yvaln,zvaln,str3);
            fputs(printstr, nfeb);
        }



        // update fiber orientations
        else if (i>es && i<ee)
        {
            // the current element number
            int e = i-es;

            float xval, yval, zval, gval, dval, wval;
            char str1[256], str2[256], str3[256], str4[256], str5[256],
str6[256], str7[256];


            if (strcmp(set.elda_flag,"yes_elda_wc") == 0)
            {
                // get what's currently on this line of the feb
                if (e < 10) {sscanf(szline,"%8s
%14s%9e,%9e,%9e%16s%8e%16s%8e%14s%e%17s",str1,str3,&xval,&yval,&zval,str4,&gv
al,str5,&dval,str6,&wval,str7);}
                if (e >= 10 && e < 100) {sscanf(szline,"%8s
%15s%9e,%9e,%9e%16s%8e%16s%8e%14s%e%17s",str1,str3,&xval,&yval,&zval,str4,&gv
al,str5,&dval,str6,&wval,str7);}
                if (e >= 100 && e < 1000) {sscanf(szline,"%8s
%16s%9e,%9e,%9e%16s%8e%16s%8e%14s%e%17s",str1,str3,&xval,&yval,&zval,str4,&gv
al,str5,&dval,str6,&wval,str7);}
                if (e >= 1000 && e < 10000) {sscanf(szline,"%8s
%17s%9e,%9e,%9e%16s%8e%16s%8e%14s%e%17s",str1,str3,&xval,&yval,&zval,str4,&gv
al,str5,&dval,str6,&wval,str7);}

                // read new fibers from fiberfile and replace the values
                get_line(lff, line, 256);
                int enf;
                sscanf(line,"%d %*e %*e %*e %e %e %e %*e %*e %*e %*e %*e
%*e",&enf,&xval,&yval,&zval);
                if (e!=enf) break;

                // print the modified line to the new file
                sprintf(str2,"\t\t\t%s
%s%+15.7e,%+15.7e,%+15.7e%16s%8f%16s%8f%14s%8f%17s
\n",str1,str3,xval,yval,zval,str4,gval,str5,dval,str6,wval,str7);
                fputs(str2, nfeb);
            }

            if (strcmp(set.elda_flag,"no_elda_wc") == 0)
            {
                // get what's currently on this line of the feb
                if (e < 10) {sscanf(szline,"%8s
```

```
%14s%9e,%9e,%9e%16s%8e%16s%8e%18s",str1,str3,&xval,&yval,&zval,str4,&gval,str
5,&dval,str6);}
                if (e >= 10 && e < 100) {sscanf(szline,"%8s
%15s%9e,%9e,%9e%16s%8e%16s%8e%18s",str1,str3,&xval,&yval,&zval,str4,&gval,str
5,&dval,str6);}
                if (e >= 100 && e < 1000) {sscanf(szline,"%8s
%16s%9e,%9e,%9e%16s%8e%16s%8e%18s",str1,str3,&xval,&yval,&zval,str4,&gval,str
5,&dval,str6);}
                if (e >= 1000 && e < 10000) {sscanf(szline,"%8s
%17s%9e,%9e,%9e%16s%8e%16s%8e%18s",str1,str3,&xval,&yval,&zval,str4,&gval,str
5,&dval,str6);}

                // read new fibers from fiberfile and replace the values
                get_line(lff, line, 256);
                int enf;
                sscanf(line,"%d %*e %*e %*e %e %e %e %*e %*e %*e %*e %*e
%*e",&enf,&xval,&yval,&zval);
                if (e!=enf) break;

                // print the modified line to the new file
                sprintf(str2,"\t\t\t%s
%s%+15.7e,%+15.7e,%+15.7e%16s%8f%16s%8f%18s
\n",str1,str3,xval,yval,zval,str4,gval,str5,dval,str6);
                fputs(str2, nfeb);
            }

        }

        /*
        //use the edp that we found in the edv loop
        else if(i == ls+8)
        {
            sprintf(str, "\t\t\t<loadpoint>1.5,%f</loadpoint> \n", edp);
            fputs(str, nfeb);
        }*/


        //modify the active contraction loadcurves
        else if(i == ls+20)
        {
            sprintf(str, "\t\t\t<loadpoint>3.0,%f</loadpoint> \n", ac);
            fputs(str, nfeb);
        }
        //else if(i == ls+45)
        //{
        //    sprintf(str, "\t\t\t<loadpoint>3.0,%f</loadpoint> \n", ac*.75);
        //    fputs(str, nfeb);
        //}


        //modify the output filenames in the logfile settings
        else if (i==os+1)
        {
            //modify the nodal coordinates output filename to contain the
model update number
```

```
            sprintf(str, "\t\t\t<node_data data=\"x;y;z\"
name=\"coordinates\" file=\"%s\"></node_data> \n", n_nodefile);
            fputs(str, nfeb);
        }
        else if (i==os+2)
        {
            //modify the nodal displacements output filename to contain the
model update number
            sprintf(str, "\t\t\t<node_data data=\"ux;uy;uz\"
name=\"displacements\" file=\"%s\"></node_data> \n", n_dispfile);
            fputs(str, nfeb);
        }
        else if (i==os+3)
        {
            //modify the nodal displacements output filename to contain the
model update number
            sprintf(str, "\t\t\t<element_data data=\"Wf\" name=\"work data\"
file=\"%s\"></element_data> \n", n_workfile);
            fputs(str, nfeb);
        }
        else if (i==os+4)
        {
            //modify the nodal displacements output filename to contain the
model update number
            sprintf(str, "\t\t\t<element_data data=\"Ax;Ay;Az\"
name=\"deformed fibers\" file=\"%s\"></element_data> \n", n_fiberfile);
            fputs(str, nfeb);
        }

        // modify the timesteps (end times) in the steps definitions
        else if (i == ss+3)
        {
            sprintf(str, "\t\t\t<time_steps>%2d</time_steps> \n", ts);
            fputs(str, nfeb);
        }
        else if (i == ss+21)
        {
            if(ts==12) { sprintf(str, "\t\t\t<time_steps>%d</time_steps>
\n",1);}

            if(ts==14) { sprintf(str, "\t\t\t<time_steps>%d</time_steps>
\n",10);}

            fputs(str, nfeb);

        }

        // just copy over all the lines that don't need to be modified
        //else {fputs(szline, nfeb);}
        else {fprintf(nfeb, "%s", szline);}
    }


    // close files
    fclose(ofeb);
```

```
    fclose(nfeb);

    return true;
}
```

A.3: create_feb1

create_feb1.h

```
#ifndef CREATE_FEB1_H_INCLUDED
#define CREATE_FEB1_H_INCLUDED

#include <iostream>
#include "get_line.h"
#include "output.h"
#include <stdio.h>
#include <string.h>
#include "settings1.h"

bool create_feb1(SETTINGS1 &set, int &c, float &edp);


#endif // CREATE_FEB1_H_INCLUDED
```

create_feb1.cpp

```
#include "create_feb1.h"

//-------------------------------------------------------------------------
-----
// create_feb1 : Function to create new febio input file from a template
.feb;
// changes the edp and makes it only run diastole

bool create_feb1(SETTINGS1 &set, int &c, float &edp)
{
    // declare stuff
    char szline[256], line[256], str[256];
    char o_febfile[256]; // original feb file. each time, we copy lines from
the same original file.
    char n_febfile[256]; // new feb file. this is the file i am making.

    // these are the filenames to put into the output section of the new feb
    char n_nodefile[256];
    char n_fiberfile[256];
    char n_dispfile[256];
    char n_workfile[256];



    // set current filenames by loop number
    if ( strcmp(set.flag, "osx") == 0 )
    {
        sprintf(o_febfile, "%s%s%s", "/Users/genevievefarrar/Desktop/CGR1/",
set.febfile,".feb");
        sprintf(n_febfile , "%s%s%03i%s",
"/Users/genevievefarrar/Desktop/CGR1/", set.febfile , c, ".feb");
```

```
        sprintf(n_nodefile, "%s%s%03i%s",
"/Users/genevievefarrar/Desktop/CGR1/", set.nodefile, c, ".txt");
        sprintf(n_dispfile, "%s%s%03i%s",
"/Users/genevievefarrar/Desktop/CGR1/", set.dispfile, c, ".txt");
        sprintf(n_workfile, "%s%s%03i%s",
"/Users/genevievefarrar/Desktop/CGR1/", set.workfile, c, ".txt");
        sprintf(n_fiberfile,"%s%s%03i%s",
"/Users/genevievefarrar/Desktop/CGR1/", set.fiberfile, c, ".txt");
    }



    /// this is to set some landmarks in the feb file
    int ls=0; //int le=0;
    int os=0; //int oe=0;
    int ss=0; //for step start

     // open original feb file
    FILE* ofeb = fopen(o_febfile, "r");
    if (ofeb == 0) return false;

    // set landmarks
    for (int i=1; fgets(szline, 256, ofeb); i++)
    {
        sscanf(szline, "\t%s %*s", str);

        if (!strcmp(str,"<LoadData>")){ls = i;}
        if (!strcmp(str,"<logfile>")){os = i;}
        if (!strcmp(str,"</Output>")){ss = i; break;}
    }
    fclose(ofeb);



    /// now reopen the old file, create the new file
    ofeb = fopen(o_febfile, "r");
    FILE * nfeb = fopen(n_febfile, "w");
    if (nfeb == 0) return false;



    /// make the new input file
    for (int i=1; fgets(szline, 256, ofeb); i++)
    {
        //modify the edp
        if(i == ls+8)
        {
            sprintf(str, "\t\t\t<loadpoint>1.5,%f</loadpoint> \n", edp);
            fputs(str, nfeb);
        }

        // modify the timesteps (end times) in the steps definitions
        else if (i == ss+3)
```

```
        {
            sprintf(str, "\t\t\t<time_steps>%2d</time_steps> \n", 6);
            fputs(str, nfeb);
        }
        else if (i == ss+21)
        {
            sprintf(str, "\t\t\t<time_steps>%d</time_steps> \n",1);
            fputs(str, nfeb);

        }



        //modify the output filenames in the logfile settings
        else if (i==os+1)
        {
            //modify the nodal coordinates output filename to contain the
model update number
            sprintf(str, "\t\t\t<node_data data=\"x;y;z\"
name=\"coordinates\" file=\"%s\"></node_data> \n", n_nodefile);
            fputs(str, nfeb);
        }
        else if (i==os+2)
        {
            //modify the nodal displacements output filename to contain the
model update number
            sprintf(str, "\t\t\t<node_data data=\"ux;uy;uz\"
name=\"displacements\" file=\"%s\"></node_data> \n", n_dispfile);
            fputs(str, nfeb);
        }
        else if (i==os+3)
        {
            //modify the nodal displacements output filename to contain the
model update number
            sprintf(str, "\t\t\t<element_data data=\"Wf\" name=\"work data\"
file=\"%s\"></element_data> \n", n_workfile);
            fputs(str, nfeb);
        }
        else if (i==os+4)
        {
            //modify the nodal displacements output filename to contain the
model update number
            sprintf(str, "\t\t\t<element_data data=\"Ax;Ay;Az\"
name=\"deformed fibers\" file=\"%s\"></element_data> \n", n_fiberfile);
            fputs(str, nfeb);
        }



        // just copy over all the lines that don't need to be modified
        else {fprintf(nfeb, "%s", szline);}
    }
```

```
    // close files
    fclose(ofeb);
    fclose(nfeb);

    return true;
}
```

A.4: create_feb2

create_feb2.h

```
#ifndef CREATE_FEB2_H_INCLUDED
#define CREATE_FEB2_H_INCLUDED

#include <iostream>
#include "get_line.h"
#include "output.h"
#include <stdio.h>
#include <string.h>
#include "settings1.h"

bool create_feb2(SETTINGS1 &set, int &c, float &edp);


#endif // CREATE_FEB2_H_INCLUDED
```

create_feb2.cpp

```
#include "create_feb2.h"

//-------------------------------------------------------------------------
-----
// create_feb2 : Function to create new febio input file from a template
.feb;
// changes the edp in the original feb

bool create_feb2(SETTINGS1 &set, int &c, float &edp)
{
    // declare stuff
    char szline[256], line[256], str[256];
    char o_febfile[256]; // original feb file. each time, we copy lines from
the same original file.
    char n_febfile[256]; // new feb file. this is the file i am making.


    // set current filenames by loop number
    if ( strcmp(set.flag, "osx") == 0 )
    {
        sprintf(o_febfile, "%s%s%s", "/Users/genevievefarrar/Desktop/CGR1/",
set.febfile,".feb");
        sprintf(n_febfile , "%s%s%03i%s",
"/Users/genevievefarrar/Desktop/CGR1/", set.febfile , 999, ".feb"); //just a
placeholder
    }
```

```
/// this is to set some landmarks in the feb file
int ls=0; //int le=0;

 // open original feb file
FILE* ofeb = fopen(o_febfile, "r");
if (ofeb == 0) return false;

// set landmarks
for (int i=1; fgets(szline, 256, ofeb); i++)
{
    sscanf(szline, "\t%s %*s", str);
    if (!strcmp(str,"<LoadData>")){ls = i;}
}
fclose(ofeb);



/// now reopen the old file, create the new file
ofeb = fopen(o_febfile, "r");
FILE * nfeb = fopen(n_febfile, "w");
if (nfeb == 0) return false;



/// make the new input file
for (int i=1; fgets(szline, 256, ofeb); i++)
{
    //modify the edp
    if(i == ls+8)
    {
        sprintf(str, "\t\t\t<loadpoint>1.5,%f</loadpoint> \n", edp);
        fputs(str, nfeb);
    }

    // just copy over all the lines that don't need to be modified
    else {fprintf(nfeb, "%s", szline);}
}

///close the files
fclose(ofeb);
fclose(nfeb);

/// now reopen the files
ofeb = fopen(o_febfile, "w");
nfeb = fopen(n_febfile, "r");


/// now copy everything back to the old feb
for (int i=1; fgets(szline, 256, nfeb); i++)
{
    // just copy over all the lines that don't need to be modified
    fprintf(ofeb, "%s", szline);
}
```

```
    // close files
    fclose(ofeb);
    fclose(nfeb);

    return true;
}
```

A.5: create_feb3

create_feb3.h

```
#ifndef CREATE_FEB3_H_INCLUDED
#define CREATE_FEB3_H_INCLUDED

#include <iostream>
#include "get_line.h"
#include "output.h"
#include <stdio.h>
#include <string.h>
#include "settings1.h"

bool create_feb3(SETTINGS1 &set, int &c, float &ac, int ts);

#endif // CREATE_FEB3_H_INCLUDED
```

create_feb3.cpp

```
#include "create_feb3.h"

//-------------------------------------------------------------------------
-----
// create_feb3 : Function to create new febio input file from a template
.feb;
//

bool create_feb3(SETTINGS1 &set, int &c, float &ac, int ts)
{
    // declare stuff
    char szline[256], line[256], str[256];
    char o_febfile[256]; // original feb file. each time, we copy lines from
the same original file.
    char n_febfile[256]; // new feb file. this is the file i am making.

    // these are the files where i am getting the nodes and fibers to put
into the new feb
    char lastnodefile[256];
    char lastfiberfile[256];

    // these are the filenames to put into the output section of the new feb
    char n_nodefile[256];
    char n_fiberfile[256];
    char n_dispfile[256];
    char n_workfile[256];
```

```
    // set current filenames by loop number
    if ( strcmp(set.flag, "lnx") == 0 )
    {
        sprintf(o_febfile, "%s%s", set.febfile,".feb");
        sprintf(n_febfile , "%s%03i%s", set.febfile , c, ".feb");
        sprintf(n_dispfile, "%s%03i%s", set.dispfile, c, ".txt");
        sprintf(n_nodefile, "%s%03i%s", set.nodefile, c, ".txt");
        sprintf(n_workfile, "%s%03i%s", set.workfile, c, ".txt");
        sprintf(n_fiberfile, "%s%03i%s", set.fiberfile, c, ".txt");
        sprintf(lastfiberfile, "fibers.txt");
        if(c==1) {sprintf(lastnodefile, "%s%s", set.nodefile,".txt");}
        else {sprintf(lastnodefile, "%s%03i%s", set.nodefile, c-1, ".txt");}
    }

    if ( strcmp(set.flag, "osx") == 0 )
    {
        sprintf(o_febfile, "%s%s%s", "/Users/genevievefarrar/Desktop/CGR1/",
set.febfile,".feb");
        sprintf(n_febfile , "%s%s%03i%s",
"/Users/genevievefarrar/Desktop/CGR1/", set.febfile , c, ".feb");
        sprintf(n_dispfile, "%s%s%03i%s",
"/Users/genevievefarrar/Desktop/CGR1/", set.dispfile, c, ".txt");
        sprintf(n_nodefile, "%s%s%03i%s",
"/Users/genevievefarrar/Desktop/CGR1/", set.nodefile, c, ".txt");
        sprintf(n_workfile, "%s%s%03i%s",
"/Users/genevievefarrar/Desktop/CGR1/", set.workfile, c, ".txt");
        sprintf(n_fiberfile,"%s%s%03i%s",
"/Users/genevievefarrar/Desktop/CGR1/", set.fiberfile, c, ".txt");
        sprintf(lastfiberfile,
"/Users/genevievefarrar/Desktop/fibdeft/out.txt");
        if(c==1) {sprintf(lastnodefile, "%s%s%s",
"/Users/genevievefarrar/Desktop/CGR1/", set.nodefile,".txt");}
        else {sprintf(lastnodefile, "%s%s%03i%s",
"/Users/genevievefarrar/Desktop/CGR1/", set.nodefile, c-1, ".txt");}
    }



    /// this is to set some landmarks in the feb file
    // initialize landmarks
    //int cs=0; int ce=0;
    //int ms=0; int me=0;
    int ns=0; int ne=0;
    int es=0; int ee=0;
    int ls=0; //int le=0;
    int os=0; //int oe=0;
    int ss=0; //for step start

    // open original feb file
    FILE* ofeb = fopen(o_febfile, "r");
    if (ofeb == 0) return false;

    // set landmarks
    for (int i=1; fgets(szline, 256, ofeb); i++)
```

```
    {
        sscanf(szline, "\t%s %*s", str);

        if (!strcmp(str,"<Nodes>"))     {ns = i;} // node deck starts
        if (!strcmp(str,"</Nodes>"))    {ne = i;} // node deck ends
        if (!strcmp(str,"<ElementData>")){es = i;}
        if (!strcmp(str,"</ElementData>")){ee = i;}
        if (!strcmp(str,"<LoadData>")){ls = i;}
        //if (!strcmp(str,"</LoadData>")){le = i;}
        if (!strcmp(str,"<logfile>"))   {os = i;}
        //if (!strcmp(str,"</logfile>"))  {oe = i;}
        if (!strcmp(str,"</Output>"))
        {
            ss = i;
            break;
        }
    }
    fclose(ofeb);



     /*
    /// this is to determine the last converged time point
    float lct = 0; //the last converged time point in the previous febio run

    // open nodal coordinates file
    FILE * lnf = fopen(lastnodefile, "r");
    if (lnf == 0) return false;

    // find last time point
    char label[256]; float value = 0;
    for (int i=1; fgets(szline, 256, ofeb); i++)
    {
        sscanf(szline, "%s %*s %f", label, &value );
      if ( strcmp(label,"*Time") == 0 && value > lct)
      {
          lct = value;
      }
    }
    fclose(lnf);
    */


    /// now reopen the files and get to the right place in them
     // re-open original febio file
    ofeb = fopen(o_febfile, "r");

    // create and open new febio file
    FILE * nfeb = fopen(n_febfile, "w");
    if (nfeb == 0) return false;
/*
    // open fiber definitions file
    FILE * lff = fopen(lastfiberfile, "r");
    if (lff == 0) return false;
    //get past the first (header) line
```

```
    fgets(szline, 256, lff);

    // open nodal coordinates file
    lnf = fopen(lastnodefile, "r");

    // get to the last time point in the nodal coordinates file
    while (fgets(szline, 256, lnf))
    {
        char label[256]; float value;
        sscanf(szline, "%s %*s %f", label, &value );
      if ( strcmp(label,"*Time") == 0 && value == lct)
        {
            fgets(szline, 256, lnf);
            break;
        }
    }
*/


    /// make the new FEBio input file
    for (int i=1; fgets(szline, 256, ofeb); i++)
    {


        /*// update nodal coordinates
        if (i>ns && i<ne)
        {
            // the current node number
            int n = i-ns;

            //float xvalo = 0; float yvalo = 0; float zvalo = 0;
            //float xvaln = 0; float yvaln = 0; float zvaln = 0;
            //char str1[25] = "0"; char str2[25] = "0"; char str3[25] = "0";
            //char printstr[256] = "0";

            float xvalo, yvalo, zvalo;
            float xvaln, yvaln, zvaln;
            char str1[25], str2[25], str3[25];
            char printstr[256];

            // get what's currently on this line of the original feb
            if (n < 10) {sscanf(szline,"%5s %7s%15e,%15e,%15e
%11s",str1,str2,&xvalo,&yvalo,&zvalo,str3);}
            if (n >= 10 && n < 100) {sscanf(szline,"%5s %8s%15e,%15e,%15e
%11s",str1,str2,&xvalo,&yvalo,&zvalo,str3);}
            if (n >= 100 && n < 1000) {sscanf(szline,"%5s %9s%15e,%15e,%15e
%11s",str1,str2,&xvalo,&yvalo,&zvalo,str3);}
            if (n >= 1000 && n < 10000) {sscanf(szline,"%5s
%10s%15e,%15e,%15e %11s",str1,str2,&xvalo,&yvalo,&zvalo,str3);}

            // read new coordinates from nodefile
            fgets(line, 256, lnf);
            int nnf;
            sscanf(line,"%d %e %e %e",&nnf,&xvaln,&yvaln,&zvaln);
            if (n!=nnf) break;
```

```
            // print the modified line to the new file
            sprintf(printstr,"\t\t\t%s %s%+15.7e,%+15.7e,%+15.7e %s
\r\n",str1,str2,xvaln,yvaln,zvaln,str3);
            fputs(printstr, nfeb);
        }



        // update fiber orientations
        else if (i>es && i<ee)
        {
            // the current element number
            int e = i-es;

            float xval, yval, zval, gval, dval, wval;
            char str1[256], str2[256], str3[256], str4[256], str5[256],
str6[256], str7[256];


            if (strcmp(set.elda_flag,"yes_elda_wc") == 0)
            {
                // get what's currently on this line of the feb
                if (e < 10) {sscanf(szline,"%8s
%14s%9e,%9e,%9e%16s%8e%16s%8e%14s%e%17s",str1,str3,&xval,&yval,&zval,str4,&gv
al,str5,&dval,str6,&wval,str7);}
                if (e >= 10 && e < 100) {sscanf(szline,"%8s
%15s%9e,%9e,%9e%16s%8e%16s%8e%14s%e%17s",str1,str3,&xval,&yval,&zval,str4,&gv
al,str5,&dval,str6,&wval,str7);}
                if (e >= 100 && e < 1000) {sscanf(szline,"%8s
%16s%9e,%9e,%9e%16s%8e%16s%8e%14s%e%17s",str1,str3,&xval,&yval,&zval,str4,&gv
al,str5,&dval,str6,&wval,str7);}
                if (e >= 1000 && e < 10000) {sscanf(szline,"%8s
%17s%9e,%9e,%9e%16s%8e%16s%8e%14s%e%17s",str1,str3,&xval,&yval,&zval,str4,&gv
al,str5,&dval,str6,&wval,str7);}

                // read new fibers from fiberfile and replace the values
                get_line(lff, line, 256);
                int enf;
                sscanf(line,"%d %*e %*e %*e %e %e %e %*e %*e %*e %*e %*e
%*e",&enf,&xval,&yval,&zval);
                if (e!=enf) break;

                // print the modified line to the new file
                sprintf(str2,"\t\t\t%s
%s%+15.7e,%+15.7e,%+15.7e%16s%8f%16s%8f%14s%8f%17s
\n",str1,str3,xval,yval,zval,str4,gval,str5,dval,str6,wval,str7);
                fputs(str2, nfeb);
            }

            if (strcmp(set.elda_flag,"no_elda_wc") == 0)
            {
                // get what's currently on this line of the feb
                if (e < 10) {sscanf(szline,"%8s
%14s%9e,%9e,%9e%16s%8e%16s%8e%18s",str1,str3,&xval,&yval,&zval,str4,&gval,str
```

```
5,&dval,str6);}
                if (e >= 10 && e < 100) {sscanf(szline,"%8s
%15s%9e,%9e,%9e%16s%8e%16s%8e%18s",str1,str3,&xval,&yval,&zval,str4,&gval,str
5,&dval,str6);}
                if (e >= 100 && e < 1000) {sscanf(szline,"%8s
%16s%9e,%9e,%9e%16s%8e%16s%8e%18s",str1,str3,&xval,&yval,&zval,str4,&gval,str
5,&dval,str6);}
                if (e >= 1000 && e < 10000) {sscanf(szline,"%8s
%17s%9e,%9e,%9e%16s%8e%16s%8e%18s",str1,str3,&xval,&yval,&zval,str4,&gval,str
5,&dval,str6);}

                // read new fibers from fiberfile and replace the values
                get_line(lff, line, 256);
                int enf;
                sscanf(line,"%d %*e %*e %*e %e %e %e %*e %*e %*e %*e %*e
%*e",&enf,&xval,&yval,&zval);
                if (e!=enf) break;

                // print the modified line to the new file
                sprintf(str2,"\t\t\t%s
%s%+15.7e,%+15.7e,%+15.7e%16s%8f%16s%8f%18s
\n",str1,str3,xval,yval,zval,str4,gval,str5,dval,str6);
                fputs(str2, nfeb);
            }

        }*/

        /*
        //use the edp that we found in the edv loop
        else if(i == ls+8)
        {
            sprintf(str, "\t\t\t<loadpoint>1.5,%f</loadpoint> \n", edp);
            fputs(str, nfeb);
        }*/


        //modify the active contraction loadcurves
        if(i == ls+20)
        {
            sprintf(str, "\t\t\t<loadpoint>3.0,%f</loadpoint> \n", ac);
            fputs(str, nfeb);
        }
        //else if(i == ls+45)
        //{
        //    sprintf(str, "\t\t\t<loadpoint>3.0,%f</loadpoint> \n", ac*.75);
        //    fputs(str, nfeb);
        //}


        //modify the output filenames in the logfile settings
        else if (i==os+1)
        {
            //modify the nodal coordinates output filename to contain the
model update number
            sprintf(str, "\t\t\t<node_data data=\"x;y;z\"
```

```
name=\"coordinates\" file=\"%s\"></node_data> \n", n_nodefile);
            fputs(str, nfeb);
        }
        else if (i==os+2)
        {
            //modify the nodal displacements output filename to contain the
model update number
            sprintf(str, "\t\t\t<node_data data=\"ux;uy;uz\"
name=\"displacements\" file=\"%s\"></node_data> \n", n_dispfile);
            fputs(str, nfeb);
        }
        else if (i==os+3)
        {
            //modify the nodal displacements output filename to contain the
model update number
            sprintf(str, "\t\t\t<element_data data=\"Wf\" name=\"work data\"
file=\"%s\"></element_data> \n", n_workfile);
            fputs(str, nfeb);
        }
        else if (i==os+4)
        {
            //modify the nodal displacements output filename to contain the
model update number
            sprintf(str, "\t\t\t<element_data data=\"Ax;Ay;Az\"
name=\"deformed fibers\" file=\"%s\"></element_data> \n", n_fiberfile);
            fputs(str, nfeb);
        }

        // modify the timesteps (end times) in the steps definitions
        else if (i == ss+3)
        {
            sprintf(str, "\t\t\t<time_steps>%2d</time_steps> \n", ts);
            fputs(str, nfeb);
        }
        else if (i == ss+21)
        {
            if(ts==12) { sprintf(str, "\t\t\t<time_steps>%d</time_steps>
\n",1);}

            if(ts==14) { sprintf(str, "\t\t\t<time_steps>%d</time_steps>
\n",10);}

            fputs(str, nfeb);

        }

        // just copy over all the lines that don't need to be modified
        //else {fputs(szline, nfeb);}
        else {fprintf(nfeb, "%s", szline);}
    }


    // close files
    fclose(ofeb);
    fclose(nfeb);
```

```
    return true;
}
```

A.6: DataAnimal

DataAnimal.h

```
//
//  DataAnimal.h
//
//
//  Created by Genevieve Farrar on 1/7/13.
//
//

#ifndef ____DataAnimal__
#define ____DataAnimal__

#include <iostream>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <vector>
#include <math.h>
#include "settings1.h"
#include "output.h"

class DataAnimal
{
public:
    int nnodes;         // number of nodes
    int nelts; // number of elements

    //float ncR[4080][3]; //nodal coordinates ref
    //float ncD[4080][3]; //nodal coordinates end diastole
    //float ncS[4080][3]; //nodal coordinates end systole
    //float ncG[4080][3]; //nodal coordinates grown ref

    float nc[2814][3]; //nodal coordinates
    float w[2220]; //element total systolic work

public:

    DataAnimal(int n);
    DataAnimal();
    ~DataAnimal();

    bool read_nodefile(const char * nodefile, float time);
    bool read_workfile(const char * workfile);
    //float calc_wallThick(int height);
    float calc_wallThick();
    float calc_volume(SETTINGS1 &set);
    float calc_avgWork();
```

```
    void updateNikefile(const char * nikefile);
    float calc_minWork();
    float calc_maxWork();
    float calc_workSRSS();
    float calc_workAvgDiff();

};

#endif /* defined(____DataAnimal__) */
```

DataAnimal.cpp

```
//
//  DataAnimal.cpp
//
//
//  Created by Genevieve Farrar on 1/7/13.
//
//

#include "DataAnimal.h"


DataAnimal::DataAnimal(int n)
{
    nnodes = n;

}

DataAnimal::DataAnimal()
{
    nnodes = 2814;
    nelts = 2220;

}

DataAnimal::~DataAnimal()
{

}

bool DataAnimal::read_nodefile(const char * nodefile, float time)
{
    char line[256]; // string to store each line

    FILE* fp = fopen(nodefile, "r");
    if (fp == 0) return false;

    char label[256];
    float value;

    while(fgets(line, 256, fp))
    {
        sscanf( line, "%5s %*1s %8f", label, &value );
      if ( strcmp(label,"*Time") == 0 && value == time )
```

```
      {
          fgets(line, 256, fp); // skip a line, then read values
          int j;
          for(j=0; j<nnodes; j++)
          {
                fgets(line, 256, fp);
                sscanf( line, "%*i %e %e %e", &nc[j][0], &nc[j][1], &nc[j][2]
);
          }
        }
    }
     fclose(fp);
    return true;

}



bool DataAnimal::read_workfile(const char * workfile)
{
    char line[256]; // string to store each line
    FILE* fp = fopen(workfile, "r");
    if (fp == 0) return false;

    char label[256];
    float value;
    float time = 3;

    while(fgets(line, 256, fp))
    {
        sscanf( line, "%5s %*1s %f", label, &value );
        //printf("this label: %s , value: %f \n",label,value);
      if ( strcmp(label,"*Time") == 0 && value == time )
      {
          //printf("correct label: %s , value: %f \n",label,value);
            fgets(line, 256, fp); // skip a line, then read values
          int j;
          for(j=0; j<nelts; j++)
          {
                fgets(line, 256, fp);
                sscanf( line, "%*i %e", &(w[j]) );

          }
        }
    }
    //printf("last label: %s , value: %f \n",label,value);

     fclose(fp);
    return true;

}



float DataAnimal::calc_wallThick()
```

```
{

    /// First, read the node numbers of the radial slices to use.
WTNodes.txt

    //declare stuff
    char line[256]; // string to store each line
    FILE* fp = 0;       // file pointer
    int enn[8][2]; // endo node numbers

    // open the file WTNodes.txt
    fp = fopen("/Users/genevievefarrar/Desktop/CGR1/WTNodes.txt", "r");
    if (fp == 0) return 0;

    int i;
    for(i=0; i<8; i++)
    {
        fgets(line, 256, fp);
        sscanf( line, "%d %d", &(enn[i][0]), &(enn[i][1]) );
    }
    fclose(fp);



    /// Next, calculate the thickness of each radial slice and average them

    float sumWT = 0;

    for(i=0; i<8; i++)
    {
        // each slice
      int p1, p2;
      float p1x, p1y, p2x, p2y;

        p1 = enn[i][0]; std::cout << "p1=" << p1;
    p2 = enn[i][1]; std::cout << "p2=" << p2;

    p1x = nc[p1-1][0]; std::cout << "p1x=" << p1x;
    p1y = nc[p1-1][1]; std::cout << "p1y=" << p1y;

    p2x = nc[p2-1][0]; std::cout << "p2x=" << p2x;
    p2y = nc[p2-1][1]; std::cout << "p2y=" << p2y;

    sumWT += sqrt( ((p1x - p2x)*(p1x - p2x)) + ((p1y - p2y)*(p1y - p2y)) );
        std::cout << "sumWT=" << sumWT;
    }
    std::cout << "Final sumWT=" << sumWT;
    std::cout << "WT=" << sumWT/8;
    return sumWT/8;

}



float DataAnimal::calc_avgWork()
```

```
{
    float sumWork = 0;
    float averageWork =0;

    int i;
    for(i=0;i<nelts;i++)
    {
        sumWork = sumWork + w[i];
        //printf("sumWork: %f \n",sumWork);
    }
    //printf("sumWork: %f \n",sumWork);
    //printf(" w[100] %f, w[200] %f, w[300] %f, w[400] %f, w[500] %f
\n",w[100],w[200],w[300],w[400],w[500] );

    float numElts = 2220.0;
    averageWork = sumWork/numElts;
    //averageWork = ( sumWork )*( 0.00045045045 );
    //printf("averageWork: %f \n",averageWork);

    return averageWork;

}

float DataAnimal::calc_maxWork()
{
    float maxWork;
    maxWork = 0.0;

    int i;
    for (i=0; i<nelts; i++)
    {
        if(w[i] > maxWork) { maxWork = w[i]; }
    }

    return maxWork;
}

float DataAnimal::calc_minWork()
{
    float minWork;
    minWork = 5.0;

    int i;
    for(i=0; i<nelts; i++)
    {
        if(w[i] < minWork) { minWork = w[i]; }
    }

    return minWork;
}


void DataAnimal::updateNikefile(const char * nikefile)
{
    // Replace the nodal coordinates in the .n file
```

```
    //------------

    //open the file and get to the right place
    char line[256];
    FILE* fp = fopen(nikefile, "r+");

    int i;
    for (i=0; i<207; i++) {fgets(line, 256, fp);}


    //print the nodal locations
    char str[256];
    int n;

    for (i=0; i<nnodes; i++)
    {
        fgets(line, 256, fp);
        sscanf( line, "%i %*f %*f %*f %*f %*f", &n);
        if (n != i+1) {break;}
        printf("node: %d \n", n);

sprintf(str,"%8d%5d%20.13E%20.13E%20.13E%5d\n",n,0,nc[i][0],nc[i][1],nc[i][2]
,7);
        fputs(str, fp);
    }
    fclose(fp);

}



float DataAnimal::calc_volume(SETTINGS1 &set)
{
    /// First, read the endo nodes from the file EndoNodes.txt

    //declare stuff
    char line[256]; // string to store each line
    FILE* fp = 0;       // file pointer
    int enn[8][24]; // endo node numbers

    // open the file EndoNodes.txt
    if ( strcmp(set.flag, "osx") == 0 ) fp =
fopen("/Users/genevievefarrar/Desktop/CGR1/EndoNodes.txt", "r");
    if ( strcmp(set.flag, "lnx") == 0 ) fp =
fopen("/home/gfarrar/CGR1/EndoNodes.txt", "r");
    if (fp == 0) return 0;

    int i;
    for(i=0; i<8; i++)
    {
        fgets(line, 256, fp);
        sscanf( line, "%d %d %d %d %d %d %d %d %d %d %d %d %d %d %d %d %d %d
%d %d %d %d %d %d",
        &(enn[i][0]), &(enn[i][1]), &(enn[i][2]), &(enn[i][3]),
        &(enn[i][4]), &(enn[i][5]), &(enn[i][6]), &(enn[i][7]),
```

```
    &(enn[i][8]), &(enn[i][9]), &(enn[i][10]),&(enn[i][11]),
    &(enn[i][12]),&(enn[i][13]),&(enn[i][14]),&(enn[i][15]),
    &(enn[i][16]),&(enn[i][17]),&(enn[i][18]),&(enn[i][19]),
    &(enn[i][20]),&(enn[i][21]),&(enn[i][22]),&(enn[i][23]) );
}
fclose(fp);



/// Next, calculate the area of each layer

float a[8][24]; // array to store areas
float cx, cy;
cx = nc[1638-1][0];
cy = nc[1638-1][1];//center point x and y coords are at node#1638


for(i=0; i<8; i++)
{
// add up the triangles around the center of the polygon, going clockwise
int j;
for(j=0; j<23; j++)
{
  float r, d;
        int p1, p2;
        float p1x, p1y, p2x, p2y;

  p1 = enn[i][j];
  p2 = enn[i][j+1];

  p1x = nc[p1-1][0];
  p1y = nc[p1-1][1];

  p2x = nc[p2-1][0];
  p2y = nc[p2-1][1];

  float area;
  area = 0.5*( cx*(p1y-p2y) + p1x*(p2y-cy) + p2x*(cy-p1y) );
        if(area<0){area=area*-1;}
  a[i][j] = area;

}

float r, d;
  int p1, p2;
  float p1x, p1y, p2x, p2y;

// this is for the last triangle
  p1 = enn[i][23];
p2 = enn[i][0];

p1x = nc[p1-1][0];
p1y = nc[p1-1][1];

p2x = nc[p2-1][0];
```

```
    p2y = nc[p2-1][1];

    r = sqrt( ((cx - p1x)*(cx - p1x)) + ((cy - p1y)*(cy - p1y)) );

    d = sqrt( ((p1x - p2x)*(p1x - p2x)) + ((p1y - p2y)*(p1y - p1y)) );

        float area;
        area = 0.5*( cx*(p1y-p2y) + p1x*(p2y-cy) + p2x*(cy-p1y) );
        if(area<0){area=area*-1;}
        a[i][j] = area;

    }


    float A[8];
    for(i=0; i<8; i++)
    {
    A[i] = a[i][0]+a[i][1]+a[i][2]+a[i][3]+a[i][4]+a[i][5]+a[i][6]+a[i][7]
       +
a[i][8]+a[i][9]+a[i][10]+a[i][11]+a[i][12]+a[i][13]+a[i][14]+a[i][15]
       +
a[i][16]+a[i][17]+a[i][18]+a[i][19]+a[i][20]+a[i][21]+a[i][22]+a[i][23];
    }


    // Next, get the distances between slices

    float dz[8];
    int n[8];
    float z[8];

    for(i=0; i<7; i++)
    {
      n[i] = enn[i][0];
    z[i] = nc[(n[i])-1][2];
    }
    n[7] = 1638;
    z[7] = nc[(n[7])-1][2];

    for(i=0; i<8; i++)
    {
    dz[i] = z[i] - z[i+1];
    }


    // Next calculate the Volume

    float V[8];

    // this is for the volumes between slices
    for(i=0; i<7; i++)
    {
      V[i] = ( 0.5 )*( (A[i]) + (A[i+1]) )*( dz[i] );
    }
```

```
    // this is for the apex volume under the lowest slice,
    V[7] = ( 0.6777 )*( A[7] )*( dz[7] );

    float Volume = V[0]+V[1]+V[2]+V[3]+V[4]+V[5]+V[6]+V[7];

    return Volume;

}
```

A.7: read_input

read_input.h

```
#ifndef READ_INPUT_H_INCLUDED
#define READ_INPUT_H_INCLUDED


#include <iostream>
#include "get_line.h"
#include "output.h"
#include "settings1.h"


bool read_input(const char* infile, SETTINGS1 &set);

#endif // READ_INPUT_H_INCLUDED
```

read_input.cpp

```
#include "read_input.h"


// read_input : Reads the input from file

bool read_input(const char* infile, SETTINGS1 &set)
{
    char szline[256];

    // open the input file
    std::cout << infile;
    FILE* fp = fopen(infile, "r");
    if (fp == 0) return false;

    // read the name of the initial FE input file
    if (get_line(fp, szline, 256) == 0) { fclose(fp); return false; }
    sscanf(szline, "%s", (set.febfile));
```

```
    // read the number of nodes
    if (get_line(fp, szline, 256) == 0) { fclose(fp); return false; }
    sscanf(szline, "%d", &(set.n));

    // times
    if (get_line(fp, szline, 256) == 0) { fclose(fp); return false; }
    sscanf(szline, "%f %f %f %f",
&(set.times[0]),&(set.times[1]),&(set.times[2]),&(set.times[3]) );

    // read the system flag, lnx/win/osx
    if (get_line(fp, szline, 256) == 0) { fclose(fp); return false; }
     sscanf(szline, "%s", (set.flag));

    // read the work tolerance
    if (get_line(fp, szline, 256) == 0) { fclose(fp); return false; }
    sscanf(szline, "%7e", &(set.w_tol));

     // read the wall thickness tolerance
    if (get_line(fp, szline, 256) == 0) { fclose(fp); return false; }
    sscanf(szline, "%7e", &(set.t_tol));

    // read the stroke volume tolerance
    if (get_line(fp, szline, 256) == 0) { fclose(fp); return false; }
    sscanf(szline, "%7e", &(set.sv_tol));

    // read the name of the nodefile
    if (get_line(fp, szline, 256) == 0) { fclose(fp); return false; }
     sscanf(szline, "%s", (set.nodefile));

    // read the name of the dispfile
    if (get_line(fp, szline, 256) == 0) { fclose(fp); return false; }
     sscanf(szline, "%s", (set.dispfile));

    // read the name of the fiberfile
    if (get_line(fp, szline, 256) == 0) { fclose(fp); return false; }
     sscanf(szline, "%s", (set.fiberfile));

    // read the name of the workfile
    if (get_line(fp, szline, 256) == 0) { fclose(fp); return false; }
     sscanf(szline, "%s", (set.workfile));

    // read the name of the associated .n
    if (get_line(fp, szline, 256) == 0) { fclose(fp); return false; }
     sscanf(szline, "%s", (set.nikefile));

    // read the starting active contraction
    if (get_line(fp, szline, 256) == 0) { fclose(fp); return false; }
     sscanf(szline, "%f", &(set.ac0));

    // read what to name the logfile
    if (get_line(fp, szline, 256) == 0) { fclose(fp); return false; }
     sscanf(szline, "%s", (set.logfile));

    // read what to name the resultsfile
    if (get_line(fp, szline, 256) == 0) { fclose(fp); return false; }
```

```
    sscanf(szline, "%s", (set.resultsfile));

    // read the elda flag,
    if (get_line(fp, szline, 256) == 0) { fclose(fp); return false; }
    sscanf(szline, "%s", (set.elda_flag));


    // read the EDV0
    if (get_line(fp, szline, 256) == 0) { fclose(fp); return false; }
    sscanf(szline, "%f", &(set.EDV0));

    // read the dstiff
    if (get_line(fp, szline, 256) == 0) { fclose(fp); return false; }
    sscanf(szline, "%f", &(set.dstiff));

    // read the EDP0
    if (get_line(fp, szline, 256) == 0) { fclose(fp); return false; }
    sscanf(szline, "%f", &(set.EDP0));

    // read the edv_tol
    if (get_line(fp, szline, 256) == 0) { fclose(fp); return false; }
    sscanf(szline, "%f", &(set.edv_tol));

    // read the SV0
    if (get_line(fp, szline, 256) == 0) { fclose(fp); return false; }
    sscanf(szline, "%f", &(set.SV0));

    // read the w0
    if (get_line(fp, szline, 256) == 0) { fclose(fp); return false; }
    sscanf(szline, "%f", &(set.w0));

    // read the t_ref0
    if (get_line(fp, szline, 256) == 0) { fclose(fp); return false; }
    sscanf(szline, "%f", &(set.t_ref0));




    // close the input file
    fclose(fp);

    return true;
}
```

A.8: Small Programs

settings.h

```
#ifndef SETTINGS1_H_INCLUDED
#define SETTINGS1_H_INCLUDED

// This structure defines the settings for the problem
typedef struct
{
```

```
    char febfile[256];  // febio input filename without extension
    int n;              // number of nodes in FE model
    float times[4];     // time values of importance: end diastole, end
systole, end relaxation, end grwoth.
    char flag[256];     // lin / win / osx
    float t_tol;        // wall thickness tolerance
    float sv_tol;       // stroke volume tolerance
    float w_tol;        // tolerance of SRSS of work difference
    char nodefile[256];
    char dispfile[256];
    char fiberfile[256];
    char workfile[256];
    char nikefile[256];
    float ac0;
    char logfile[256];
    char resultsfile[256];
    char elda_flag[256];
    float EDV0;
    float dstiff;
    float EDP0;
    float edv_tol;
    float SV0;
    float w0;
    float t_ref0;

}SETTINGS1;

#endif // SETTINGS1_H_INCLUDED
```

hello.h

```
#ifndef HELLO_H_INCLUDED
#define HELLO_H_INCLUDED

#include "output.h"

void hello(FILE * fpout);

#endif // HELLO_H_INCLUDED
```

hello.cpp

```
#include "hello.h"


//-----------------------------------------------------------------------------
-----
// hello : welcome message to print at the beginning of the run
//
void hello(FILE * fpout)
{
  output("\n=======================================================\n\n",
fpout);
  output("                                                        \n\n",
```

```
fpout);
  output("                                LV G&R                          \n\n",
fpout);
  output("                                                                 \n\n",
fpout);
  output("  G.E. Farrar                                                  \n", fpout);
  output("  04/29/12                                                     \n", fpout);
  output("\n============================================================\n\n",
fpout);
}
```

output.h

```
#ifndef OUTPUT_H_INCLUDED
#define OUTPUT_H_INCLUDED

#include <stdio.h>

void output(const char* sz, FILE* fpout);


#endif // OUTPUT_H_INCLUDED
```

output.cpp

```
#include "output.h"

//-------------------------------------------------------------------------
-----
// output : writes a string to both the screen and to the log file
//
void output(const char* sz, FILE* fpout)
{

     fputs(sz, stdout); // output the string sz to the screen

     if (fpout)
     {
         fputs(sz, fpout); // output the string sz to the output file
     }

}
```

get_line.h

```
#ifndef GET_LINE_H_INCLUDED
#define GET_LINE_H_INCLUDED

#include <stdio.h>

char* get_line(FILE* fp, char* szline, int N);
```

```
#endif // GET_LINE_H_INCLUDED
```

get_line.cpp

```
#include "get_line.h"

//-----------------------------------------------------------------------------
// get_line : Gets lines out of input file, while skipping over comment lines
//
char* get_line(FILE* fp, char* szline, int N)
{
  do
    {
      fgets(szline, N, fp);

      if (feof(fp) || ferror(fp)) return 0;
    }
  while (szline[0] == '*');
  return szline; //returns (pointer to) the line of the input file
}
```

Appendix B: Modifications to FEBio Programs

Parts that were modified are indicated with the comment "///GEF"

In FEBioImport.cpp,

```
//-----------------------------------------------------------------------------
//! Reads the ElementData section from the FEBio input file

void FEBioGeometrySection::ParseElementDataSection(XMLTag& tag)
{
     int i;

     FEM& fem = *GetFEM();
     FEMesh& mesh = fem.m_mesh;

     // get the total nr of elements
     int nelems = mesh.Elements();

     //make sure we've read the element section
     if (nelems == 0) throw XMLReader::InvalidTag(tag);

     // create the pelem array
```

```
	vector<FEElement*> pelem;
	pelem.assign(nelems, static_cast<FEElement*>(0));

	for (int nd=0; nd<mesh.Domains(); ++nd)
	{
	  FEDomain& d = mesh.Domain(nd);
	  for (i=0; i<d.Elements(); ++i)
	  {
		FEElement& el = d.ElementRef(i);
		assert(pelem[el.m_nID-1] == 0);
		pelem[el.m_nID-1] = &el;
	  }
	}

	// read additional element data
	++tag;
	do
	{
	  // make sure this is an "element" tag
	  if (tag != "element") throw XMLReader::InvalidTag(tag);

	  // get the element number
	  const char* szid = tag.AttributeValue("id");
	  int n = atoi(szid)-1;

	  // make sure the number is valid
	  if ((n<0) || (n>=nelems)) throw XMLReader::InvalidAttributeValue(tag,
"id", szid);

	  // get a pointer to the element
	  FEElement* pe = pelem[n];

	  vec3d a;
	  double gr; ///GEF: for ElementData growth parameter
	  double acd; ///GEF: for ElementData active contraction delay parameter
	    double cwv; ///GEF: for ElementData critical work value
	  ++tag;
	  do
	  {
		if (tag == "fiber")
		{
			// read the fiber direction
			tag.value(a);

			// normalize fiber
			a.unit();

			// set up a orthonormal coordinate system
			vec3d b(0,1,0);
			if (fabs(fabs(a*b) - 1) < 1e-7) b = vec3d(0,0,1);
			vec3d c = a^b;
			b = c^a;

			// make sure they are unit vectors
			b.unit();
```

```
                c.unit();

                FESolidElement* pbe = dynamic_cast<FESolidElement*> (pe);
                FEShellElement* pse = dynamic_cast<FEShellElement*> (pe);
                if (pbe)
                {
                      for (int i=0; i<pbe->GaussPoints(); ++i)
                      {
                            FEElasticMaterialPoint& pt = *pbe->m_State[i]-
>ExtractData<FEElasticMaterialPoint>();
                            mat3d& m = pt.Q;
                            m.zero();
                            m[0][0] = a.x; m[0][1] = b.x; m[0][2] = c.x;
                            m[1][0] = a.y; m[1][1] = b.y; m[1][2] = c.y;
                            m[2][0] = a.z; m[2][1] = b.z; m[2][2] = c.z;
                      }
                }
                if (pse)
                {
                      for (int i=0; i<pse->GaussPoints(); ++i)
                      {
                            FEElasticMaterialPoint& pt = *pse->m_State[i]-
>ExtractData<FEElasticMaterialPoint>();
                            mat3d& m = pt.Q;
                            m.zero();
                            m[0][0] = a.x; m[0][1] = b.x; m[0][2] = c.x;
                            m[1][0] = a.y; m[1][1] = b.y; m[1][2] = c.y;
                            m[2][0] = a.z; m[2][1] = b.z; m[2][2] = c.z;
                      }
                }
          }
          else if (tag == "growth") ///GEF: for ElementData growth
parameter
          {
                // read the growth parameter
                tag.value(gr);

                FESolidElement* pbe = dynamic_cast<FESolidElement*> (pe);

                if (pbe)
                {
                      for (int i=0; i<pbe->GaussPoints(); ++i)
                      {
                            FEElasticMaterialPoint& pt = *pbe->m_State[i]-
>ExtractData<FEElasticMaterialPoint>();
                            mat3d& m = pt.g;
                            m.zero();
                            m[0][0] = gr; m[0][1] = 0; m[0][2] = 0;
                            m[1][0] = 0; m[1][1] = 0; m[1][2] = 0;
                            m[2][0] = 0; m[2][1] = 0; m[2][2] = 0;
                      }
                }
          }
          else if (tag == "delay") ///GEF: for ElementData active
contraction delay parameter
```

```
		{
			// read the value into the variable acd
			tag.value(acd);

			FESolidElement* pbe = dynamic_cast<FESolidElement*> (pe);

			if (pbe)
			{
				for (int i=0; i<pbe->GaussPoints(); ++i)
				{
					FEElasticMaterialPoint& pt = *pbe->m_State[i]->ExtractData<FEElasticMaterialPoint>();
					mat3d& m = pt.delay;
					m.zero();
					m[0][0] = acd; m[0][1] = 0; m[0][2] = 0;
					m[1][0] = 0; m[1][1] = 0; m[1][2] = 0;
					m[2][0] = 0; m[2][1] = 0; m[2][2] = 0;
				}
			}
		}
		else if (tag == "work") ///GEF: for ElementData critical work value
		{
			// read the value of the tag into the local variable cwv
			tag.value(cwv);

			FESolidElement* pbe = dynamic_cast<FESolidElement*> (pe);

			if (pbe)
			{
				//put the value of cvw into the [0][0] part of the Work_crit of all the pts in this element
			  for (int i=0; i<pbe->GaussPoints(); ++i)
				{
					FEElasticMaterialPoint& pt = *pbe->m_State[i]->ExtractData<FEElasticMaterialPoint>();
					mat3d& m = pt.WC;
					m.zero();
					m[0][0] = cwv; m[0][1] = 0; m[0][2] = 0;
					m[1][0] = 0; m[1][1] = 0; m[1][2] = 0;
					m[2][0] = 0; m[2][1] = 0; m[2][2] = 0;
				}
			}
		}
		else if (tag == "mat_axis")
		{
			vec3d a, d;

			++tag;
			do
			{
				if (tag == "a") tag.value(a);
				else if (tag == "d") tag.value(d);
				else throw XMLReader::InvalidTag(tag);
```

```
                        ++tag;
                }
                while (!tag.isend());

                vec3d c = a^d;
                vec3d b = c^a;

                // normalize
                a.unit();
                b.unit();
                c.unit();

                // assign to element
                FESolidElement* pbe = dynamic_cast<FESolidElement*> (pe);
                FEShellElement* pse = dynamic_cast<FEShellElement*> (pe);
                if (pbe)
                {
                        for (int i=0; i<pbe->GaussPoints(); ++i)
                        {
                                FEElasticMaterialPoint& pt = *pbe->m_State[i]-
>ExtractData<FEElasticMaterialPoint>();
                                mat3d& m = pt.Q;
                                m.zero();
                                m[0][0] = a.x; m[0][1] = b.x; m[0][2] = c.x;
                                m[1][0] = a.y; m[1][1] = b.y; m[1][2] = c.y;
                                m[2][0] = a.z; m[2][1] = b.z; m[2][2] = c.z;
                        }
                }
                if (pse)
                {
                        for (int i=0; i<pse->GaussPoints(); ++i)
                        {
                                FEElasticMaterialPoint& pt = *pse->m_State[i]-
>ExtractData<FEElasticMaterialPoint>();
                                mat3d& m = pt.Q;
                                m.zero();
                                m[0][0] = a.x; m[0][1] = b.x; m[0][2] = c.x;
                                m[1][0] = a.y; m[1][1] = b.y; m[1][2] = c.y;
                                m[2][0] = a.z; m[2][1] = b.z; m[2][2] = c.z;
                        }
                }
        }
        else if (tag == "thickness")
        {
                FEShellElement* pse = dynamic_cast<FEShellElement*> (pe);
                if (pse == 0) throw XMLReader::InvalidTag(tag);

                // read shell thickness
                tag.value(&pse->m_h0[0],pse->Nodes());
        }
        else if (tag == "area")
        {
                FETrussElement* pt = dynamic_cast<FETrussElement*>(pe);
                if (pt == 0) throw XMLReader::InvalidTag(tag);
```

```
                    // read truss area
                    tag.value(pt->m_a0);
                }
                else throw XMLReader::InvalidTag(tag);
                ++tag;
            }
            while (!tag.isend());

            ++tag;
        }
        while (!tag.isend());
}
```

In DataStore.h

```
//-----------------------------------------------------------------------------
--
class ElementDataRecord : public DataRecord
{
    enum {X, Y, Z, J,
      EX, EY, EZ, EXY, EYZ, EXZ,
      E1, E2, E3,
      EF,
      SX, SY, SZ, SXY, SYZ, SXZ,
      S1, S2, S3,
      SF,
      FX, FY, FZ, FYZ, FZX, FXY, FYX, FXZ, FZY,
      P, WX, WY, WZ, C, JX, JY, JZ, CRC,
      C1, J1X, J1Y, J1Z, C2, J2X, J2Y, J2Z,
      C3, J3X, J3Y, J3Z, C4, J4X, J4Y, J4Z,
      C5, J5X, J5Y, J5Z, C6, J6X, J6Y, J6Z,
      PSI, IEX, IEY, IEZ,
      WF, AX, AY, AZ}; ///GEF: added EF, SF, WF, AX, AY, AZ to this list

    struct ELEMREF
    {
      int   ndom;
      int   nid;
    };

public:
    ElementDataRecord(FEM* pfem, const char* szfile) :  DataRecord(pfem,
szfile){}
    double Evaluate(int item, int ndata);
    void Parse(const char* sz);
    void SelectAllItems();

protected:
    void BuildELT();

protected:
    vector<ELEMREF>     m_ELT;
};
```

In DataStore.cpp

```
void ElementDataRecord::Parse(const char *szexpr)
{
     char szcopy[MAX_STRING] = {0};
     strcpy(szcopy, szexpr);
     char* sz = szcopy, *ch;
     m_data.clear();
     do
     {
       ch = strchr(sz, ';');
       if (ch) *ch++ = 0;
       if      (strcmp(sz, "x"  ) == 0) m_data.push_back(X  );
       else if (strcmp(sz, "y"  ) == 0) m_data.push_back(Y  );
       else if (strcmp(sz, "z"  ) == 0) m_data.push_back(Z  );
       else if (strcmp(sz, "J"  ) == 0) m_data.push_back(J  );
       else if (strcmp(sz, "Ex" ) == 0) m_data.push_back(EX );
       else if (strcmp(sz, "Ey" ) == 0) m_data.push_back(EY );
       else if (strcmp(sz, "Ez" ) == 0) m_data.push_back(EZ );
       else if (strcmp(sz, "Exy") == 0) m_data.push_back(EXY);
       else if (strcmp(sz, "Eyz") == 0) m_data.push_back(EYZ);
       else if (strcmp(sz, "Exz") == 0) m_data.push_back(EXZ);
       else if (strcmp(sz, "E1" ) == 0) m_data.push_back(E1);
       else if (strcmp(sz, "E2" ) == 0) m_data.push_back(E2);
       else if (strcmp(sz, "E3" ) == 0) m_data.push_back(E3);
         else if (strcmp(sz, "Ef" ) == 0) m_data.push_back(EF); ///GEF: added
Ef, fiber strain, to this list
       else if (strcmp(sz, "sx" ) == 0) m_data.push_back(SX );
       else if (strcmp(sz, "sy" ) == 0) m_data.push_back(SY );
       else if (strcmp(sz, "sz" ) == 0) m_data.push_back(SZ );
       else if (strcmp(sz, "sxy") == 0) m_data.push_back(SXY);
       else if (strcmp(sz, "syz") == 0) m_data.push_back(SYZ);
       else if (strcmp(sz, "sxz") == 0) m_data.push_back(SXZ);
       else if (strcmp(sz, "s1" ) == 0) m_data.push_back(S1 );
       else if (strcmp(sz, "s2" ) == 0) m_data.push_back(S2 );
       else if (strcmp(sz, "s3" ) == 0) m_data.push_back(S3 );
         else if (strcmp(sz, "sf" ) == 0) m_data.push_back(SF); ///GEF: added
sf, fiber stress, to this list
       else if (strcmp(sz, "Fx" ) == 0) m_data.push_back(FX );
       else if (strcmp(sz, "Fy" ) == 0) m_data.push_back(FY );
       else if (strcmp(sz, "Fz" ) == 0) m_data.push_back(FZ );
       else if (strcmp(sz, "Fyz") == 0) m_data.push_back(FYZ);
       else if (strcmp(sz, "Fzx") == 0) m_data.push_back(FZX);
       else if (strcmp(sz, "Fxy") == 0) m_data.push_back(FXY);
       else if (strcmp(sz, "Fyx") == 0) m_data.push_back(FYX);
       else if (strcmp(sz, "Fxz") == 0) m_data.push_back(FXZ);
       else if (strcmp(sz, "Fzy") == 0) m_data.push_back(FZY);
       else if (strcmp(sz, "p"  ) == 0) m_data.push_back(P  );
       else if (strcmp(sz, "wx" ) == 0) m_data.push_back(WX );
       else if (strcmp(sz, "wy" ) == 0) m_data.push_back(WY );
       else if (strcmp(sz, "wz" ) == 0) m_data.push_back(WZ );
```

```
      else if (strcmp(sz, "c"  ) == 0) m_data.push_back(C  );
      else if (strcmp(sz, "jx" ) == 0) m_data.push_back(JX );
      else if (strcmp(sz, "jy" ) == 0) m_data.push_back(JY );
      else if (strcmp(sz, "jz" ) == 0) m_data.push_back(JZ );
      else if (strcmp(sz, "crc") == 0) m_data.push_back(CRC);
      else if (strcmp(sz, "c1"  ) == 0) m_data.push_back(C1 );
      else if (strcmp(sz, "c2"  ) == 0) m_data.push_back(C2 );
      else if (strcmp(sz, "c3"  ) == 0) m_data.push_back(C3 );
      else if (strcmp(sz, "c4"  ) == 0) m_data.push_back(C4 );
      else if (strcmp(sz, "c5"  ) == 0) m_data.push_back(C5 );
      else if (strcmp(sz, "c6"  ) == 0) m_data.push_back(C6 );
      else if (strcmp(sz, "j1x" ) == 0) m_data.push_back(J1X);
      else if (strcmp(sz, "j1y" ) == 0) m_data.push_back(J1Y);
      else if (strcmp(sz, "j1z" ) == 0) m_data.push_back(J1Z);
      else if (strcmp(sz, "j2x" ) == 0) m_data.push_back(J2X);
      else if (strcmp(sz, "j2y" ) == 0) m_data.push_back(J2Y);
      else if (strcmp(sz, "j2z" ) == 0) m_data.push_back(J2Z);
      else if (strcmp(sz, "j3x" ) == 0) m_data.push_back(J3X);
      else if (strcmp(sz, "j3y" ) == 0) m_data.push_back(J3Y);
      else if (strcmp(sz, "j3z" ) == 0) m_data.push_back(J3Z);
      else if (strcmp(sz, "j4x" ) == 0) m_data.push_back(J4X);
      else if (strcmp(sz, "j4y" ) == 0) m_data.push_back(J4Y);
      else if (strcmp(sz, "j4z" ) == 0) m_data.push_back(J4Z);
      else if (strcmp(sz, "j5x" ) == 0) m_data.push_back(J5X);
      else if (strcmp(sz, "j5y" ) == 0) m_data.push_back(J5Y);
      else if (strcmp(sz, "j5z" ) == 0) m_data.push_back(J5Z);
      else if (strcmp(sz, "j6x" ) == 0) m_data.push_back(J6X);
      else if (strcmp(sz, "j6y" ) == 0) m_data.push_back(J6Y);
      else if (strcmp(sz, "j6z" ) == 0) m_data.push_back(J6Z);
      else if (strcmp(sz, "psi" ) == 0) m_data.push_back(PSI);
      else if (strcmp(sz, "Iex" ) == 0) m_data.push_back(IEX);
      else if (strcmp(sz, "Iey" ) == 0) m_data.push_back(IEY);
      else if (strcmp(sz, "Iez" ) == 0) m_data.push_back(IEZ);
        else if (strcmp(sz, "Wf" ) == 0) m_data.push_back(WF); ///GEF: added
Wf, systolic work in fiber direction
      else if (strcmp(sz, "Ax" ) == 0) m_data.push_back(AX); ///GEF: added Ax
to this list
      else if (strcmp(sz, "Ay" ) == 0) m_data.push_back(AY); ///GEF: added Ay
to this list
      else if (strcmp(sz, "Az" ) == 0) m_data.push_back(AZ); ///GEF: added Az
to this list
      else throw UnknownDataField(sz);
      sz = ch;
    }
    while (ch);
}
```

In FEAnalysis.cpp

```
bool FEAnalysis::Solve()
{
    // do one time initialization of solver data
    if (m_psolver->Init() == false)
    {
```

```
      clog.printbox("FATAL ERROR","Initialization has failed.\nAborting
run.\n");
      return false;
    }

    // obtain a pointer to the console object. We'll use this to
    // set the title of the console window.
    Console* pShell = Console::GetHandle();

    // convergence flag
    // we initialize it to true so that when a restart is performed after
    // the last time step we terminate normally.
    bool bconv = true;

    // calculate end time value
    double starttime = m_fem.m_ftime0;
//  double endtime = m_fem.m_ftime0 + m_ntime*m_dt0;
    double endtime = m_tend;
    const double eps = endtime*1e-7;

    int nsteps = m_fem.m_Step.size();

    bool bdebug = m_fem.GetDebugFlag();

    if (nsteps > 1)
      pShell->SetTitle("(step %d/%d: %.f%%) %s - %s", m_fem.m_nStep+1,
nsteps, (100.f*(m_fem.m_ftime - starttime) / (endtime - starttime)),
m_fem.GetFileTitle(), (bdebug?"FEBio (debug mode)": "FEBio"));
    else
      pShell->SetTitle("(%.f%%) %s - %s", (100.f*m_fem.m_ftime/endtime),
m_fem.GetFileTitle(), (bdebug?"FEBio (debug mode)": "FEBio"));

    // keep a stack for push/pop'ing
    stack<FEM> state;

    // print initial progress bar
    if (GetPrintLevel() == FE_PRINT_PROGRESS)
    {
      printf("\nProgress:\n");
      for (int i=0; i<50; ++i) printf("\xB0"); printf("\r");
      clog.SetMode(Logfile::FILE_ONLY);
    }

    // if we restarted we need to update the timestep
    // before continuing
    if (m_ntimesteps != 0)
    {
      // update time step
      if (m_bautostep && (m_fem.m_ftime + eps < endtime))
AutoTimeStep(m_psolver->m_niter);
    }
    else
    {
      // make sure that the timestep is at least the min time step size
      if (m_bautostep) AutoTimeStep(0);
```

```
    }

    // repeat for all timesteps
    m_nretries = 0;
    while (endtime - m_fem.m_ftime > eps)
    {
      // keep a copy of the current state, in case
      // we need to retry this time step
      if (m_bautostep)
      {
            while (!state.empty()) state.pop();
            state.push(m_fem);
      }

      // update time
      m_fem.m_ftime += m_dt;

      int i;

      // evaluate load curve values at current time
      for (i=0; i<m_fem.LoadCurves(); ++i) m_fem.GetLoadCurve(i)-
>Evaluate(m_fem.m_ftime);

        ///GEF: put current time into the FEElasticMaterialPoint's
        FEMesh& mesh = m_fem.m_mesh;
        for (int i=0; i<mesh.Domains(); ++i)
        {
            for (int j=0; j<mesh.Domain(i).Elements() ; j++)
            {
                FEElementTraits* et = mesh.Domain(i).ElementRef(j).m_pT;
                for (int k=0; k<(*et).nint ; k++)
                {
                    FEMaterialPoint* mp =
mesh.Domain(i).ElementRef(j).m_State[k];
                    FEElasticMaterialPoint* pt =
(*mp).ExtractData<FEElasticMaterialPoint>();
                    (*pt).time = m_fem.m_ftime;
                }
            }
        }



      // evaluate material parameter lists
      for (i=0; i<m_fem.Materials(); ++i)
      {
            // get the material
            FEMaterial* pm = m_fem.GetMaterial(i);

            // evaluate the material parameters
            m_fem.EvaluateMaterialParameters(pm);
      }

      // evaluate body-force parameter lists
      for (i=0; i<m_fem.BodyForces(); ++i)
```

```
	{
		FEParameterList& pl = m_fem.GetBodyForce(i)->GetParameterList();
		m_fem.EvaluateParameterList(pl);
	}

	// evaluate contact interface parameter lists
	for (i=0; i<m_fem.ContactInterfaces(); ++i)
	{
		FEParameterList& pl = m_fem.m_CI[i]->GetParameterList();
		m_fem.EvaluateParameterList(pl);
	}

	// solve this timestep,
	try
	{
		int oldmode = 0;
		bconv = m_psolver->SolveStep(m_fem.m_ftime);
	}
	catch (ExitRequest)
	{
		bconv = false;
		clog.printbox("WARNING", "Early termination on user's request");
		break;
	}
	catch (ZeroDiagonal e)
	{
		bconv = false;
		// TODO: Fix this feature
		clog.printbox("FATAL ERROR", "Zero diagonal detected. Aborting
run.");
		break;
	}
	catch (NANDetected)
	{
		bconv = false;
		clog.printbox("FATAL ERROR", "NAN Detected. Run aborted.");
		break;
	}
	catch (MemException e)
	{
		bconv = false;
		if (e.m_falloc < 1024*1024)
			clog.printbox("FATAL ERROR", "Failed allocating %lg bytes",
e.m_falloc);
		else
		{
			double falloc = e.m_falloc / (1024.0*1024.0);
			clog.printbox("FATAL ERROR", "Failed allocating %lg MB",
falloc);
		}
		break;
	}
	catch (FEMultiScaleException)
	{
		bconv = false;
```

```
            clog.printbox("FATAL ERROR", "The RVE problem has failed.
Aborting macro run.");
            break;
      }
      catch (std::bad_alloc e)
      {
            bconv = false;
            clog.printbox("FATAL ERROR", "A memory allocation failure has
occured.\nThe program will now be terminated.");
            break;
      }
      catch (...)
      {
            bconv = false;
            clog.printbox("FATAL ERROR", "An unknown exception has
occured.\nThe program will now be terminated.");
            break;
      }

      // update counters
      m_ntotref  += m_psolver->m_ntotref;
      m_ntotiter += m_psolver->m_niter;
      m_ntotrhs  += m_psolver->m_nrhs;

      // see if we have converged
      if (bconv)
      {
            // Yes! We have converged!
            clog.printf("\n\n------- converged at time : %lg\n\n",
m_fem.m_ftime);

            // update nr of completed timesteps
            m_ntimesteps++;

            // output results to plot database
            if (m_nplot != FE_PLOT_NEVER)
            {
                  if ((m_nplot == FE_PLOT_MUST_POINTS) && (m_nmplc >= 0))
                  {
                        FELoadCurve& lc = *m_fem.GetLoadCurve(m_nmplc);
                        if (lc.HasPoint(m_fem.m_ftime)) m_fem.m_plot-
>Write(m_fem);
                  }
                  else m_fem.m_plot->Write(m_fem);
            }

            // Dump converged state to the archive
            if (m_bDump)
            {
                  DumpFile ar(&m_fem);
                  if (ar.Create(m_fem.GetDumpFileName()) == false)
                  {
                        clog.printf("WARNING: Failed creating restart
point.\n");
                  }
```

```
                else
                {
                        m_fem.Serialize(ar);
                        clog.printf("\nRestart point created. Archive name is
%s\n", m_fem.GetDumpFileName());
                }
            }

            ///GEF: see what time it is. two options:
            /// 1. if systole, store Eff and sff
            /// 2. both diastole and systole, store Eff and sff
            //double w_flag = m_fem.GetLoadCurve(7)->Value(m_fem.m_ftime);
            // this load curve is 1 at end diastole and 2 at end systole
            //if (w_flag >= 1 && w_flag <= 2)
            if (m_fem.m_ftime >= 1.5 && m_fem.m_ftime <= 3.0)
            ///if (m_fem.m_ftime >= 0 && m_fem.m_ftime <= 3.0)
            {
                // get the mesh
                FEMesh& mesh = m_fem.m_mesh;

                // for each domain, for each elt, for each gauss pt,
                //
                for (int i=0; i<mesh.Domains(); ++i)
                {
                    for (int j=0; j<mesh.Domain(i).Elements() ; j++)
                    {
                        for (int k=0; k<8; k++)
                        {
                            FEMaterialPoint* mp =
mesh.Domain(i).ElementRef(j).m_State[k];
                            FEElasticMaterialPoint* pt =
(*mp).ExtractData<FEElasticMaterialPoint>();

                            // get the fiber orientation
                            vec3d a0;
                            a0.x = (*pt).Q[0][0];
                            a0.y = (*pt).Q[1][0];
                            a0.z = (*pt).Q[2][0];
                            vec3d a = ((*pt).F)*a0;
                            a.unit();

                            // get the fiber strain and fiber stress
                            mat3ds C = (*pt).RightCauchyGreen();
                            mat3dd I(1.0);
                            mat3ds E = 0.5*(C-I);
                            double eachEff = a*(E*a); // green lagrange
strain in fiber direction
                            (*pt).Eff.push_back( eachEff );

                            mat3ds stress = (*pt).s;
                            double eachsff = a*(stress*a); // cauchy stress
in fiber direction
                            (*pt).sff.push_back( eachsff );

                        }
```

```
                    }
                }
            }

            ///GEF: see what time it is. if END-systole, calc systolic work
            //if (w_flag == 2)
            if (m_fem.m_ftime == 3.0)
            {
                // get the mesh
                FEMesh& mesh = m_fem.m_mesh;

                for (size_t i=0; i<mesh.Domains(); ++i)
                {
                    for (size_t j=0; j<mesh.Domain(i).Elements() ; j++)
                    {
                        FEElementTraits* et =
mesh.Domain(i).ElementRef(j).m_pT;
                        for (size_t k=0; k<(*et).nint ; k++)
                        {
                            FEMaterialPoint* mp =
mesh.Domain(i).ElementRef(j).m_State[k];
                            FEElasticMaterialPoint* pt =
(*mp).ExtractData<FEElasticMaterialPoint>();
                            double work = 0;
                            for (size_t l=1; l<(*pt).Eff.size(); l++)
                            {
                                double dE = (*pt).Eff[l] - (*pt).Eff[l-1]; if
(dE<0) {dE = -1*dE;}
                                double avgS = 0.5 * ( (*pt).sff[l-1] +
(*pt).sff[l] ); if (avgS<0) {avgS = -1*avgS;}
                                work += avgS * dE;
                            }
                            (*pt).W = work;
                        }
                    }
                }
            }


            // store additional data to the logfile
            m_fem.m_Data.Write();

            // update time step
            if (m_bautostep && (m_fem.m_ftime + eps < endtime))
AutoTimeStep(m_psolver->m_niter);

            // reset retry counter
            m_nretries = 0;

            // call callback function
            m_fem.DoCallback();
      }
      else
      {
            // Report the sad news to the user.
```

```
            clog.printf("\n\n------- failed to converge at time : %lg\n\n",
m_fem.m_ftime);

            // If we have auto time stepping, decrease time step and let's
retry
            if (m_bautostep && (m_nretries < m_maxretries))
            {
                  // restore the previous state
                  m_fem = state.top(); state.pop();

                  // let's try again
                  Retry();
            }
            else
            {
                  // can't retry, so abort
                  if (m_nretries >= m_maxretries)     clog.printf("Max. nr of
retries reached.\n\n");

                  break;
            }
      }

      // print a progress bar
      if (GetPrintLevel() == FE_PRINT_PROGRESS)
      {
            int l = (int)(50*m_fem.m_ftime / endtime);
            for (int i=0; i<l; ++i) printf("\xB2"); printf("\r");
            fflush(stdout);
      }

      // flush the m_log file, so we don't loose anything if
      // the next timestep goes wrong
      clog.flush();

      bool bdebug = m_fem.GetDebugFlag();
      if (nsteps>1)
            pShell->SetTitle("(step %d/%d: %.f%%) %s - %s", m_fem.m_nStep+1,
nsteps, (100.f*(m_fem.m_ftime - starttime) / (endtime - starttime)),
m_fem.GetFileTitle(), (bdebug?"FEBio (debug mode)": "FEBio"));
      else
            pShell->SetTitle("(%.f%%) %s - %s",
(100.f*m_fem.m_ftime/endtime), m_fem.GetFileTitle(), (bdebug?"FEBio (debug
mode)": "FEBio"));
    }

    m_fem.m_ftime0 = m_fem.m_ftime;

    if (GetPrintLevel() == FE_PRINT_PROGRESS)
    {
      clog.SetMode(Logfile::FILE_AND_SCREEN);
    }

    // output report
    clog.printf("\n\nN O N L I N E A R  I T E R A T I O N  I N F O R M A T
```

```
I O N\n\n");
    clog.printf("\tNumber of time steps completed ..................... :
%d\n\n", m_ntimesteps);
    clog.printf("\tTotal number of equilibrium iterations ............ :
%d\n\n", m_ntotiter);
    clog.printf("\tAverage number of equilibrium iterations .......... :
%lg\n\n", (double) m_ntotiter / (double) m_ntimesteps);
    clog.printf("\tTotal number of right hand evaluations ............ :
%d\n\n", m_ntotrhs);
    clog.printf("\tTotal number of stiffness reformations ............ :
%d\n\n", m_ntotref);

    // get and print elapsed time
    char sztime[64];

    m_psolver->m_SolverTime.time_str(sztime);
    clog.printf("\tTime in solver: %s\n\n", sztime);

    return bconv;
}
```

In FEElasticMaterial.cpp

```
#pragma once
#include "FEMaterial.h"

//-----------------------------------------------------------------------------
--
// This class stores material point data for elastic materials
class FEElasticMaterialPoint : public FEMaterialPoint
{
public:
    FEElasticMaterialPoint()
    {
      F.zero();
      Q.unit();
      J = 1;
      s.zero();
      g.zero(); ///GEF: for growth parameter(s)
      W = 0; ///GEF: for systolic work
        WC.zero(); ///GEF: for critical work parameter
      delay.zero(); ///GEF: for active contraction delay parameter
      time = 0;
    }

    FEMaterialPoint* Copy()
    {
      FEElasticMaterialPoint* pt = new FEElasticMaterialPoint(*this);
      if (m_pt) pt->m_pt = m_pt->Copy();
      return pt;
    }

    void Serialize(DumpFile& ar)
    {
      if (ar.IsSaving())
```

```
      {
            ar << F << J << Q << s << g << W; ///GEF: added g and W to this
list
      }
      else
      {
            ar >> F >> J >> Q >> s >> g >> W; ///GEF: added g and W to this
list
      }

      if (m_pt) m_pt->Serialize(ar);
    }

    mat3ds Strain();

    mat3ds RightCauchyGreen();
    mat3ds LeftCauchyGreen ();

    mat3ds DevRightCauchyGreen();
    mat3ds DevLeftCauchyGreen ();

    mat3ds pull_back(const mat3ds& A);
    mat3ds push_forward(const mat3ds& A);

public:
    void Init(bool bflag)
    {
      if (bflag)
      {
            F.unit();

            J = 1;

            s.zero();

//          Q.unit();
      }

      if (m_pt) m_pt->Init(bflag);
    }

public:
    // position
    vec3d   r0;   //!< material position
    vec3d   rt;   //!< spatial position

    // deformation data
    mat3d   F;    //!< deformation gradient
    double  J;                //!< determinant8 of F
    mat3d   Q;                //!< local material orientation

    ///GEF: for ElementData growth parameter
    mat3d g;

    /// GEF: fiber stress and fiber strain during systole
```

```
    // these get set in FEAnalysis.cpp
    // the vector contains the values at each time step
    vector<double> Eff;
    vector<double> sff;

    ///GEF: systolic fiber work
    // this is done in FEAnalysis.cpp
    // W is the total systolic work in the fiber direction
    // its value is calculated at end systole
    double W;

    ///GEF: critical work above which growth occurs.
    mat3d WC;

    ///GEF: active contraction delay
    mat3d delay;

    ///GEF: current time
    double time;

    // solid material data
    mat3ds  s;                //!< Cauchy stress
};
```

In FEFiberMaterial.cpp

```
#include "stdafx.h"
#include "FEFiberMaterial.h"
#include "FEElasticMaterial.h"

//-----------------------------------------------------------------------------
--
// Fiber material stress
//
mat3ds FEFiberMaterial::Stress(FEMaterialPoint &mp)
{
    FEElasticMaterialPoint& pt = *mp.ExtractData<FEElasticMaterialPoint>();

    // deformation gradient
    mat3d& F = pt.F;
    double J = pt.J;
    double Ji = 1.0 / J;
    double Jm13 = pow(J, -1.0/3.0);
    double twoJi = 2.0*Ji;
    double acd = pt.delay[0][0]; ///GEF: active contraction delay parameter


    // get the initial fiber direction
    vec3d a0;
    a0.x = pt.Q[0][0];
    a0.y = pt.Q[1][0];
    a0.z = pt.Q[2][0];

    // calculate the current material axis lam*a = F*a0;
    vec3d a = F*a0;
```

```
    // normalize material axis and store fiber stretch
    double lam, lamd;
    lam = a.unit();
    lamd = lam*Jm13; // i.e. lambda tilde

    // invariant I4
    double I4 = lamd*lamd;

    // strain energy derivative
    double W4 = 0;
    if (lamd > 1)
    {
      double lamdi = 1.0/lamd;
      double Wl;
      if (lamd < m_lam1)
      {
            Wl = lamdi*m_c3*(exp(m_c4*(lamd - 1)) - 1);
      }
      else
      {
            double c6 = m_c3*(exp(m_c4*(m_lam1-1))-1) - m_c5*m_lam1;
            Wl = lamdi*(m_c5*lamd + c6);
      }
      W4  = 0.5*lamdi*Wl;
    }
    else
    {
      W4 = 0;
    }

    // calculate dyad of a: AxA = (a x a)
    mat3ds AxA = dyad(a);

    // ---
    // calculate FdWf/dCFt = I4*W4*(a x a)
    mat3ds T = AxA*(W4*I4);

    // calculate stress:
    mat3ds s = T.dev()*twoJi;

    // --- active contraction contribution ---
    if (m_lcna >= 0)
    {
      //double ctenslm = m_plc->Value();
      double ctenslm = m_plc->Value( pt.time - acd ); ///GEF: active
contracion delay

      // current sarcomere length
      double strl = m_refl*lamd;

      // sarcomere length change
      double dl = strl - m_l0;

      if (dl >= 0)
```

```
      {
            double eca50i = (exp(m_beta*dl) - 1);

            // active fiber stress
            double saf = ctenslm*eca50i / ( eca50i + 1 );

            // add saf*(a x a)
            s += AxA*saf;
      }
    }
    //-----------------------------------------

    return s;
}
```

Appendix C: Creating <ElementData> section of FEBio input file

C.1: Formatting fiber orientations for febio input file

text2feb : main.cpp

```
/*

text2feb(fibers.txt, out.txt)

written by G.E.Farrar on 6/6/12



This program reformats fibers.txt (output from fibdeft)
into fiber definition of <ElementData> in FEBio input file.

*/


#include <iostream>
#include <stdio.h>

using namespace std;

int main(int nargs, char *argv[])
{
    if(nargs!=3){ cout << "incorrect arguments.\n"; }

    bool g = true; // true if we want to make growth tags too

    // open input file
    FILE* fp = fopen(argv[1], "r");
```

```
    if (fp == 0) return false;

    // get number of nodes from the first line of the file
    int n; // number of nodes
    char szline[256];
    fgets(szline, 256, fp);
     sscanf(szline, "%d", &n);

    // create array to store nodal fiber orientation vectors
    float data [n][3];

    // read the data from fibers.txt into data[][]
    int num;
     for(int i=0; i<n; i++)
     {
         fgets(szline, 256, fp);
         sscanf(szline, "%d %f %f %f", &num, &data[i][0], &data[i][1],
&data[i][2]);
         if (num!=(i+1)) { break; }
     }

    // open output file
    FILE* fpo = fopen(argv[2], "w");
    if (fpo == 0) return false;

    // print the fiber vectors in FEBio ElementData format
    char str[256];
    fputs("\t\t<ElementData>\n", fpo);
    for(int i=0; i<n; i++)
    {
         sprintf(str, "\t\t\t<element id=\"%d\">", i+1);
         fputs(str, fpo);

         sprintf(str, "<fiber>%f,%f,%f</fiber>", data[i][0], data[i][1],
data[i][2]);
         fputs(str, fpo);

         if (g == true)
         {
             sprintf(str, "<growth>%f</growth>", 1.0);
             fputs(str, fpo);
         }

         fputs("</element>\n", fpo);
    }
    fputs("\t\t</ElementData>", fpo);


    return 0;
}
```

C.2: Creating Whole <ElementData> section of febio input file including <fibers>, <growth>, and <delay>

```
/*
ElementData(fibers.txt, growth.txt, delay.txt, out.txt)

written by G.E.Farrar on 10/26/12



This program creates the <ElementData> section for the
 GRFEBio input file. It will create the tags for
 <fibers>, <growth>, and <delay>, from data files
 fibers.txt, growth.txt, and delay.txt. Whatever files
 are included in the command will have tags created
 in the output out.txt.


*/


#include <iostream>
#include <stdio.h>
#include <string.h>

using namespace std;

int main(int nargs, char *argv[])
{
    // some variables to use
    int fib_arg, gro_arg, del_arg;
    int n, num;
    char szline[256];

    // assess what arguments we have
    if(nargs<3){ cout << "incorrect arguments.\n"; }

    if( strcmp(argv[1],"fibers.txt") == 0 )
        {fib_arg = 1;}
    else if( strcmp(argv[2],"fibers.txt") == 0 )
        {fib_arg = 2;}
    else if( strcmp(argv[3],"fibers.txt") == 0 )
        {fib_arg = 3;}
    else
        {fib_arg = 0;}

    if( strcmp(argv[1],"growth.txt") == 0)
        {gro_arg = 1;}
    else if( strcmp(argv[2],"growth.txt") == 0)
        {gro_arg = 2;}
    else if( strcmp(argv[3],"growth.txt") == 0)
        {gro_arg = 3;}
    else
        {gro_arg = 0;}

    if( strcmp(argv[1],"delay.txt") == 0)
        {del_arg = 1;}
    else if( strcmp(argv[2],"delay.txt") == 0)
```

```
        {del_arg = 2;}
    else if( strcmp(argv[3],"delay.txt") == 0)
        {del_arg = 3;}
    else
        {del_arg = 0;}


    //--- read fibers ---
    // open input file
    FILE* fpf = fopen(argv[fib_arg], "r");
    if (fpf == 0) return false;

    // get number from the first line of the file
    fgets(szline, 256, fpf);
     sscanf(szline, "%d", &n);

    // create array to store fiber orientation vectors
    float dataf [n][3];

    // read the data from fibers.txt into dataf[][]
     for(int i=0; i<n; i++)
     {
         fgets(szline, 256, fpf);
         sscanf(szline, "%d %f %f %f", &num, &dataf[i][0], &dataf[i][1],
&dataf[i][2]);
         if (num!=(i+1)) { break; }
     }

    //--- read growth ---
    // open input file
    FILE* fpg = fopen(argv[gro_arg], "r");
    if (fpg == 0) return false;

    // get number from the first line of the file
    fgets(szline, 256, fpg);
     sscanf(szline, "%d", &n);

    // create array to store fiber orientation vectors
    float datag [n][1];

    // read the data from growth.txt into datag[][]
     for(int i=0; i<n; i++)
     {
         fgets(szline, 256, fpg);
         sscanf(szline, "%d %f", &num, &datag[i][0]);
         if (num!=(i+1)) { break; }
     }

    //--- read delay ---
    // open input file
    FILE* fpd = fopen(argv[del_arg], "r");
    if (fpd == 0) return false;

    // get number from the first line of the file
    fgets(szline, 256, fpd);
```

```
    sscanf(szline, "%d", &n);

    // create array to store fiber orientation vectors
    float datad [n][1];

    // read the data from delay.txt into datad[][]
    for(int i=0; i<n; i++)
    {
        fgets(szline, 256, fpd);
        sscanf(szline, "%d %f", &num, &datad[i][0]);
        if (num!=(i+1)) { break; }
    }


    // open output file
    FILE* fpo;
    if(nargs == 3)
        fpo = fopen(argv[2], "w");
    else if(nargs == 4)
        fpo = fopen(argv[3], "w");
    else if(nargs == 5)
        fpo = fopen(argv[4], "w");

    if (fpo == 0) return false;

    // print ElementData
    char str[256];
    fputs("\t\t<ElementData>\n", fpo);
    for(int i=0; i<n; i++)
    {
        sprintf(str, "\t\t\t<element id=\"%d\">", i+1);
        fputs(str, fpo);

        if(fib_arg != 0)
        {
            sprintf(str, "<fiber>%f,%f,%f</fiber>", dataf[i][0], dataf[i][1],
dataf[i][2]);
            fputs(str, fpo);
        }


        if(gro_arg != 0)
        {
            sprintf(str, "<growth>%f</growth>", datag[i][0]);
            fputs(str, fpo);
        }

        if(del_arg != 0)
        {
            sprintf(str, "<delay>%f</delay>", datad[i][0]);
            fputs(str, fpo);
        }

        fputs("</element>\n", fpo);
    }
```

```
    fputs("\t\t</ElementData>", fpo);


    return 0;
}
```

Appendix D: Post-processing Code for SHR Study

D.1: For Calculating Average Strains

jan312013NONregionalstrainsall.m

```
clear all; close all; clc;


format long
load names.mat


W_t1 = 0;
W_t3 = 0;
W_t4 = 0;
W_t5 = 0;

S_t1 = 0;
S_t3 = 0;
S_t4 = 0;
S_t5 = 0;

for i = [1:17,21:34,36:37,39:length(names)]

    % Construct the path to look for text files relating to this data set
    if (names{i}(1:3) == 'shr')
        types(i) = 1;
        path = strcat('C:\Documents and Settings\Genevieve\Desktop\RATS',...
                      '\SHR',names{i}(4:6),'\');
    elseif (names{i}(1:3) == 'wky')
        types(i) = 2;
        path = strcat('C:\Documents and Settings\Genevieve\Desktop\RATS',...
                      '\WKY',names{i}(4:6),'\');
    end

    times(i) = str2num(names{i}(6));

    FS = zeros(15448,9); %Initialize fiber strain array for each data set.
```

```
Columns 1 an 9 will remain zeros.

    for j = 8 % 4 for end diastole, 8 for end systole

        strainfile = strcat(path,'fs',num2str(j-1),'.txt' );
        FS(:,j) = dlmread(strainfile, ' ', [5 1 15452 1]);
        FS(:,j) = FS(:,j)-1; %Remember to subtract one so that zero strain is
0 not 1


        c1 = 0;
        c2 = 0;
        c3 = 0;
        c4 = 0;
        if( types(i) == 2 )
            if( times(i) == 1)
                for k = 1:15448;
                    temp(k) = FS(k,j);
                end
                if( W_t1 == 0) W_t1 = temp;
                else W_t1 = [W_t1,temp];
                end
            elseif( times(i) == 3)
                for k = 1:15448;
                    temp(k) = FS(k,j);
                end
                if( W_t3 == 0) W_t3 = temp;
                else W_t3 = [W_t3,temp];
                end
            elseif( times(i) == 4)
                for k = 1:15448;
                    temp(k) = FS(k,j);
                end
                if( W_t4 == 0) W_t4 = temp;
                else W_t4 = [W_t4,temp];
                end
            elseif( times(i) == 5)
                for k = 1:15448;
                    temp(k) = FS(k,j);
                end
                if( W_t5 == 0) W_t5 = temp;
                else W_t5 = [W_t5,temp];
                end
            end

        elseif( types(i) == 1 )
            if( times(i) == 1)
                for k = 1:15448;
                    temp(k) = FS(k,j);
                end
                if( S_t1 == 0) S_t1 = temp;
                else S_t1 = [S_t1,temp];
                end
            elseif( times(i) == 3)
                for k = 1:15448;
```

```
                    temp(k) = FS(k,j);
                end
                if( S_t3 == 0) S_t3 = temp;
                else S_t3 = [S_t3,temp];
                end
            elseif( times(i) == 4)
                for k = 1:15448;
                    temp(k) = FS(k,j);
                end
                if( S_t4 == 0) S_t4 = temp;
                else S_t4 = [S_t4,temp];
                end
            elseif( times(i) == 5)
                for k = 1:15448;
                    temp(k) = FS(k,j);
                end
                if( S_t5 == 0) S_t5 = temp;
                else S_t5 = [S_t5,temp];
                end
            end

        end


    end
end

save('AVGstrainsHUGE.mat');
```

TTESTSnonregional.m

```
W_m =  [mean(W_t1);
        mean(W_t3);
        mean(W_t4);
        mean(W_t5)]

W_sd = [std(W_t1);
        std(W_t3);
        std(W_t4);
        std(W_t5)]

allWm =  mean([W_t1,W_t3,W_t4,W_t5]')
allWs =   std([W_t1,W_t3,W_t4,W_t5]')

S_m =  [mean(S_t1);
        mean(S_t3);
        mean(S_t4);
        mean(S_t5)]

S_sd = [std(S_t1);
        std(S_t3);
        std(S_t4);
        std(S_t5)]
```

```
format long

% [h,p_W_1] = ttest2(W_t1_r1, W_t3_r1);
% [h,p_W_2] = ttest2(W_t3_r1, W_t4_r1);
% [h,p_W_3] = ttest2(W_t4_r1, W_t5_r1);
%
% p_W = [p_W_1;
%        p_W_2;
%        p_W_3]
%
% [h,p_S_r1_1] = ttest2(S_t1_r1, S_t3_r1);
% [h,p_S_r1_2] = ttest2(S_t3_r1, S_t4_r1);
% [h,p_S_r1_3] = ttest2(S_t4_r1, S_t5_r1);
%
% [h,p_S_r2_1] = ttest2(S_t1_r2, S_t3_r2);
% [h,p_S_r2_2] = ttest2(S_t3_r2, S_t4_r2);
% [h,p_S_r2_3] = ttest2(S_t4_r2, S_t5_r2);
%
% [h,p_S_r3_1] = ttest2(S_t1_r3, S_t3_r3);
% [h,p_S_r3_2] = ttest2(S_t3_r3, S_t4_r3);
% [h,p_S_r3_3] = ttest2(S_t4_r3, S_t5_r3);
%
% [h,p_S_r4_1] = ttest2(S_t1_r4, S_t3_r4);
% [h,p_S_r4_2] = ttest2(S_t3_r4, S_t4_r4);
% [h,p_S_r4_3] = ttest2(S_t4_r4, S_t5_r4);
% p_S = [p_S_r1_1;
%        p_S_r2_1;
%        p_S_r3_1;
%        p_S_r4_1;
%        p_S_r1_2;
%        p_S_r2_2;
%        p_S_r3_2;
%        p_S_r4_2;
%        p_S_r1_3;
%        p_S_r2_3;
%        p_S_r3_3;
%        p_S_r4_3]
```

jan312013regionalstrainsall.m

```
clear all; close all; clc;


format long
load names.mat


W_t1 = 0;
W_t3 = 0;
W_t4 = 0;
W_t5 = 0;
```

```
S_t1 = 0;
S_t3 = 0;
S_t4 = 0;
S_t5 = 0;

for i = [1:17,21:34,36:37,39:length(names)]

    % Construct the path to look for text files relating to this data set
    if (names{i}(1:3) == 'shr')
        types(i) = 1;
        path = strcat('C:\Documents and Settings\Genevieve\Desktop\RATS',...
                      '\SHR',names{i}(4:6),'\');
    elseif (names{i}(1:3) == 'wky')
        types(i) = 2;
        path = strcat('C:\Documents and Settings\Genevieve\Desktop\RATS',...
                      '\WKY',names{i}(4:6),'\');
    end

    times(i) = str2num(names{i}(6));

    FS = zeros(15448,9); %Initialize fiber strain array for each data set.
Columns 1 an 9 will remain zeros.

    for j = 4 % 4 for end diastole, 8 for end systole

        strainfile = strcat(path,'fs',num2str(j-1),'.txt' );
        FS(:,j) = dlmread(strainfile, ' ', [5 1 15452 1]);
        FS(:,j) = FS(:,j)-1; %Remember to subtract one so that zero strain is
0 not 1


        c1 = 0;
        c2 = 0;
        c3 = 0;
        c4 = 0;
        if( types(i) == 2 )
            if( times(i) == 1)
                for k = 1:15448;
                    temp(k) = FS(k,j);
                end
                if( W_t1 == 0) W_t1 = temp;
                else W_t1 = [W_t1,temp];
                end
            elseif( times(i) == 3)
                for k = 1:15448;
                    temp(k) = FS(k,j);
                end
                if( W_t3 == 0) W_t3 = temp;
                else W_t3 = [W_t3,temp];
                end
            elseif( times(i) == 4)
                for k = 1:15448;
                    temp(k) = FS(k,j);
                end
                if( W_t4 == 0) W_t4 = temp;
```

```
                else W_t4 = [W_t4,temp];
                end
            elseif( times(i) == 5)
                for k = 1:15448;
                    temp(k) = FS(k,j);
                end
                if( W_t5 == 0) W_t5 = temp;
                else W_t5 = [W_t5,temp];
                end

        elseif( types(i) == 1 )
            if( times(i) == 1)
                for k = 1:15448;
                    temp(k) = FS(k,j);
                end
                if( S_t1 == 0) S_t1 = temp;
                else S_t1 = [S_t1,temp];
                end
            elseif( times(i) == 3)
                for k = 1:15448;
                    temp(k) = FS(k,j);
                end
                if( S_t3 == 0) S_t3 = temp;
                else S_t3 = [S_t3,temp];
                end
            elseif( times(i) == 4)
                for k = 1:15448;
                    temp(k) = FS(k,j);
                end
                if( S_t4 == 0) S_t4 = temp;
                else S_t4 = [S_t4,temp];
                end
            elseif( times(i) == 5)
                for k = 1:15448;
                    temp(k) = FS(k,j);
                end
                if( S_t5 == 0) S_t5 = temp;
                else S_t5 = [S_t5,temp];
                end
            end

        end


    end
end

save('AVGstrainsHUGE.mat');
```

TTESTSregional.m

```
%
% W_m =  [mean(W_t1_r1);
%         mean(W_t1_r2);
```

```
%          mean(W_t1_r3);
%          mean(W_t1_r4);
%          mean(W_t3_r1);
%          mean(W_t3_r2);
%          mean(W_t3_r3);
%          mean(W_t3_r4);
%          mean(W_t4_r1);
%          mean(W_t4_r2);
%          mean(W_t4_r3);
%          mean(W_t4_r4);
%          mean(W_t5_r1);
%          mean(W_t5_r2);
%          mean(W_t5_r3);
%          mean(W_t5_r4)]
%
% W_sd = [std(W_t1_r1);
%          std(W_t1_r2);
%          std(W_t1_r3);
%          std(W_t1_r4);
%          std(W_t3_r1);
%          std(W_t3_r2);
%          std(W_t3_r3);
%          std(W_t3_r4);
%          std(W_t4_r1);
%          std(W_t4_r2);
%          std(W_t4_r3);
%          std(W_t4_r4);
%          std(W_t5_r1);
%          std(W_t5_r2);
%          std(W_t5_r3);
%          std(W_t5_r4)]
%
% S_m =  [mean(S_t1_r1);
%          mean(S_t1_r2);
%          mean(S_t1_r3);
%          mean(S_t1_r4);
%          mean(S_t3_r1);
%          mean(S_t3_r2);
%          mean(S_t3_r3);
%          mean(S_t3_r4);
%          mean(S_t4_r1);
%          mean(S_t4_r2);
%          mean(S_t4_r3);
%          mean(S_t4_r4);
%          mean(S_t5_r1);
%          mean(S_t5_r2);
%          mean(S_t5_r3);
%          mean(S_t5_r4)]
%
% S_sd = [std(S_t1_r1);
%          std(S_t1_r2);
%          std(S_t1_r3);
%          std(S_t1_r4);
%          std(S_t3_r1);
%          std(S_t3_r2);
```

```
%         std(S_t3_r3);
%         std(S_t3_r4);
%         std(S_t4_r1);
%         std(S_t4_r2);
%         std(S_t4_r3);
%         std(S_t4_r4);
%         std(S_t5_r1);
%         std(S_t5_r2);
%         std(S_t5_r3);
%         std(S_t5_r4)]

format long
[h,p_W_r1_1] = ttest2(W_t1_r1, W_t3_r1);
[h,p_W_r1_2] = ttest2(W_t3_r1, W_t4_r1);
[h,p_W_r1_3] = ttest2(W_t4_r1, W_t5_r1);

[h,p_W_r2_1] = ttest2(W_t1_r2, W_t3_r2);
[h,p_W_r2_2] = ttest2(W_t3_r2, W_t4_r2);
[h,p_W_r2_3] = ttest2(W_t4_r2, W_t5_r2);

[h,p_W_r3_1] = ttest2(W_t1_r3, W_t3_r3);
[h,p_W_r3_2] = ttest2(W_t3_r3, W_t4_r3);
[h,p_W_r3_3] = ttest2(W_t4_r3, W_t5_r3);

[h,p_W_r4_1] = ttest2(W_t1_r4, W_t3_r4);
[h,p_W_r4_2] = ttest2(W_t3_r4, W_t4_r4);
[h,p_W_r4_3] = ttest2(W_t4_r4, W_t5_r4);
p_W = [p_W_r1_1;
       p_W_r2_1;
       p_W_r3_1;
       p_W_r4_1;
       p_W_r1_2;
       p_W_r2_2;
       p_W_r3_2;
       p_W_r4_2;
       p_W_r1_3;
       p_W_r2_3;
       p_W_r3_3;
       p_W_r4_3]

[h,p_S_r1_1] = ttest2(S_t1_r1, S_t3_r1);
[h,p_S_r1_2] = ttest2(S_t3_r1, S_t4_r1);
[h,p_S_r1_3] = ttest2(S_t4_r1, S_t5_r1);

[h,p_S_r2_1] = ttest2(S_t1_r2, S_t3_r2);
[h,p_S_r2_2] = ttest2(S_t3_r2, S_t4_r2);
[h,p_S_r2_3] = ttest2(S_t4_r2, S_t5_r2);

[h,p_S_r3_1] = ttest2(S_t1_r3, S_t3_r3);
[h,p_S_r3_2] = ttest2(S_t3_r3, S_t4_r3);
[h,p_S_r3_3] = ttest2(S_t4_r3, S_t5_r3);

[h,p_S_r4_1] = ttest2(S_t1_r4, S_t3_r4);
[h,p_S_r4_2] = ttest2(S_t3_r4, S_t4_r4);
[h,p_S_r4_3] = ttest2(S_t4_r4, S_t5_r4);
```

```
p_S = [p_S_r1_1;
       p_S_r2_1;
       p_S_r3_1;
       p_S_r4_1;
       p_S_r1_2;
       p_S_r2_2;
       p_S_r3_2;
       p_S_r4_2;
       p_S_r1_3;
       p_S_r2_3;
       p_S_r3_3;
       p_S_r4_3]
```

D.2: For Calculating the Thickness and Diameter of The LV Mesh

ThicknessDiameter.m

```
clear all; close all; clc;

format long
load names.mat

n_z_r = 20;

t = zeros(42,1); % lv thickness
d = zeros(42,1); % lv diameter
a = zeros(42,1); % age/acquisition number

for i = [1:17,21:34,35,36,37,38,39:length(names)]

    % Construct the path to look for text files relating to this data set
    if (names{i}(1:3) == 'shr')
        path = strcat('C:\Documents and Settings\Genevieve\Desktop\RATS',...
                      '\SHR',names{i}(4:6),'\');
        sw(i,1) = 1;
    elseif (names{i}(1:3) == 'wky')
        path = strcat('C:\Documents and Settings\Genevieve\Desktop\RATS',...
                      '\WKY',names{i}(4:6),'\');
        sw(i,1) = 2;
    end

    nodefile = strcat( path, names{i}(1:4), names{i}(6), 'nodes.txt');

    nodes = GetNodes(nodefile,5,15448); %these are ref config nodes

    [data] = GetTD( nodes , n_z_r );

    % use max thickness and diameter values along the height of the LV
    %t(i,1) = max(data(:,1));
    %d(i,1) = max(data(:,2));
```

```
    %use average of the values in upper-middle part, slices 4-7 of 20.
    t(i,1) = mean(data(4:7,1));
    d(i,1) = mean(data(4:7,2));

    a(i,1) = str2num(names{i}(6)); % this is the acquisition number

    % use this later to look at lifespan changes in thickness and diameter
between es/ed/ref

%     FS = zeros(15448,9); %Initialize fiber strain array for each data set.
%                          %Columns 1 and 9 will remain zeros for ref config.
%
%     for j = 2:8
%         strainfile = strcat(path,'fs',num2str(j-1),'.txt' );
%         FS(:,j) = dlmread(strainfile, ' ', [5 1 15452 1]);
%         FS(:,j) = FS(:,j)-1; %Remember to subtract one so that zero strain
is 0 not 1
%         for r = 1:nr
%             c=0;
%             for k = 1:15448
%                 if R(k,1) == r
%                     c = c+1;
%                     fsr(c) = FS(k,j);
%                 end
%             end
%             FSR_all(r,j,i) = mean(fsr);
%             clear fsr;
%         end
%    end

end


%This just puts a '3' instead of a '0' in the rows of sw that haven't been
%set to 1 or 2 for shr or wky. Matlab doesn't like zeros in there.
for i=1:42
    if sw(i,1)==0,
        sw(i,1) = 3;
    end
end


%These are my beautiful colors
col = [ [.6 .1 .6];[.1 .1 .7];[.2 .6 .7];[.2 .6 .1];[.5 .5 .5];...
    [.7 .7 .3];[.9 .5 .1];[.8 .2 .4];[.8 .1 .6];[.1 .1 .1] ];


%%% make some basic plots to see the relationship. This just puts all of
%%% the points on there. no averages per age group.
figure % thickness
for i = 1:42
plot( a(i), t(i), ...
    'Color', col(sw(i,1),:),...
    'Marker','d',...
    'MarkerFaceColor',col(sw(i,1),:)), hold on,
```

```
end
hold off

figure % diameter
for i = 1:42
plot( a(i), d(i), ...
    'Color', col(sw(i,1),:),...
    'Marker','d',...
    'MarkerFaceColor',col(sw(i,1),:)), hold on,
end
hold off


%%% Average and standard dev of all of the WKY data
% (use to get "normal")
Avg_t_wky = mean( t(21:42,1) );
Std_t_wky =  std( t(21:42,1) );

%%% Now get the average at each time point

%This is to organize t and d by age for SHR
c3 = zeros(2,5);
for j = 1:5
    for i = 1:17
        if a(i)==j
            c3(1,j) = c3(1,j)+1;
            t_Age_shr(c3(1,j),j) = t(i);
            d_Age_shr(c3(1,j),j) = d(i);
        end
    end
end
%This is to organize t and d by age for WKY
for j = 1:5
    for i = 18:42
        if a(i)==j
            c3(2,j) = c3(2,j)+1;
            t_Age_wky(c3(2,j),j) = t(i);
            d_Age_wky(c3(2,j),j) = d(i);
        end
    end
end

% This calculates the average and std err of t and d by age group

for j = 1:5
    %WKY
    Avg_t_Age_shr(j) = mean( t_Age_shr( 1:c3(1,j),j ) );
    Avg_d_Age_shr(j) = mean( d_Age_shr( 1:c3(1,j),j ) );
    StdE_t_Age_shr(j) =  std( t_Age_shr( 1:c3(1,j),j ) ) / sqrt( c3(1,j) );
    StdE_d_Age_shr(j) =  std( d_Age_shr( 1:c3(1,j),j ) ) / sqrt( c3(1,j) );
    Std_t_Age_shr(j) =  std( t_Age_shr( 1:c3(1,j),j ) );
    Std_d_Age_shr(j) =  std( d_Age_shr( 1:c3(1,j),j ) );

    %SHR
    Avg_t_Age_wky(j) = mean( t_Age_wky( 1:c3(2,j),j ) );
```

```
    Avg_d_Age_wky(j) = mean( d_Age_wky( 1:c3(2,j),j ) );
    StdEr_t_Age_wky(j) =  std( t_Age_wky( 1:c3(2,j),j ) ) / sqrt( c3(2,j) );
    StdEr_d_Age_wky(j) =  std( d_Age_wky( 1:c3(2,j),j ) ) / sqrt( c3(2,j) );
    Std_t_Age_wky(j) =  std( t_Age_wky( 1:c3(2,j),j ) );
    Std_d_Age_wky(j) =  std( d_Age_wky( 1:c3(2,j),j ) );
    %check statistical significance
    [h,p] = ttest2( t_Age_shr( 1:c3(1,j),j ), t_Age_wky( 1:c3(2,j),j ), 0.05
);
    HPt(j,1:2) = [h,p];
    [h,p] = ttest2( d_Age_shr( 1:c3(1,j),j ), d_Age_wky( 1:c3(2,j),j ), 0.05
);
    HPd(j,1:2) = [h,p];
end




figure
plot([6,12,15,18,20], Avg_t_Age_wky, 'bs-'),hold on
plot([6,12,15,18,20], Avg_t_Age_shr, 'ro-'),hold off

figure
plot([6,12,15,18,20], Avg_d_Age_wky, 'bs-'),hold on
plot([6,12,15,18,20], Avg_d_Age_shr, 'ro-'),hold off


figure
plot(t(1:17),d(1:17),'bs'), hold on
plot(t(18:42),d(18:42),'r^'),
axis([5 7 10 16])
```

GetTD.m

```
function [out] = GetTD( C, n_z_r )


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%
% function [T D] = GetTD( C, z )
%
% inputs:   C: Nodal Coordinates. Nx3 array where each row contains the
%               x, y, and z coordinates of the N'th node.
%           n_z_r: number of z regions to slice it into
%
% output:   T: 1 x n_z_r. thickness of LV at heights z. average of
thicknesses of the 4
%               quadrants.
%           D: 1 x n_z_r. diameter of LV at heights z. average of outer
diameters on x
%               and y axes.
%
% e.g.: [T D] = GetTD( C, 20 )
%      note: Z_td = 11.94 corresponds to slice 16, approximately the widest
%      part of the LV.
```

```
%
% G.E.FARRAR
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%

%n_z_r = 20
delta = 0.5;

format long
N = size(C,1); %Number of nodes

%%%%  Initialize Data Arrays  %%%%
R = zeros(5000,3,n_z_r);
D = zeros(n_z_r,1);
T = zeros(n_z_r,1);
ct = zeros(n_z_r,1);
A = zeros(100,3,4);
ct2 = zeros(4,1);

%%%%  Assign the nodes to a z-region.  %%%%
z_max = max(C(:,3));
z_min = min(C(:,3));
z_chunk = (z_max - z_min)/n_z_r;

for i = 1:(n_z_r + 1)
    z_slices(i) = z_max - (i-1)*z_chunk;
end

for j = 1:n_z_r
    for i = 1:N
        z = C(i,3);
        if z <= z_slices(j) && z > z_slices(j+1)
            ct(j,1) = ct(j,1) + 1;
            R(ct(j,1),:,j) = (C(i,:));
        end
    end
end


%%%%  Calculate Average Diameter and Thickness.  %%%%
for j = 1:n_z_r

    maxX = max(R(1:ct(j,1),1,j));
    minX = min(R(1:ct(j,1),1,j));
    maxY = max(R(1:ct(j,1),2,j));
    minY = min(R(1:ct(j,1),2,j));

    cenX = mean([maxX,minX],2);
    cenY = mean([maxY,minY],2);

    DiaX = maxX-minX;
    DiaY = maxY-minY;

    AvgDia = mean([DiaX,DiaY],2);
    D(j,1) = AvgDia;
```

```
    for i = 1:ct(j,1)
        if( (R(i,2,j) < (cenY+delta)) && (R(i,2,j) > (cenY-delta)) )
            if (R(i,1,j) > (cenX))
                ct2(1,1) = ct2(1,1) + 1;
                A(ct2(1,1),:,1) = R(i,:,j);
            elseif (R(i,1,j) < (cenX))
                ct2(2,1) = ct2(2,1) + 1;
                A(ct2(2,1),:,2) = R(i,:,j);
            end
        elseif( (R(i,1,j) < (cenX+delta)) && (R(i,1,j) > (cenX-delta)) )
            if (R(i,2,j) > (cenY))
                ct2(3,1) = ct2(3,1) + 1;
                A(ct2(3,1),:,3) = R(i,:,j);
            elseif (R(i,2,j) < (cenY))
                ct2(4,1) = ct2(4,1) + 1;
                A(ct2(4,1),:,4) = R(i,:,j);
            end
        end
    end

    t1 = maxX - min(A(1:ct2(1,1),1,1));
    t2 = maxY - min(A(1:ct2(2,1),2,2));
    t3 = abs(minX - max(A(1:ct2(3,1),1,3)));
    t4 = abs(minY - max(A(1:ct2(4,1),2,4)));

    AvgT = mean([t1, t2, t3, t4],2);
    T(j,1) = AvgT;

end

out = [T D];
```

D.3: Organizing and Presenting Strain Data

DataProcessing.m

```
clear all, close all, clc

 load fiberstrains.mat

% %% Look at progression of fiber strains over time for each rat %%
% % SHR's
% figure,
% subplot(1,7,1), bar(shr1_1'), title('shr1'), legend('1'),
%     axis([0 8 -.1 0.08]),
%     set(gca,'YTick',[-0.08 -0.06 -0.04 -0.02 0 0.02 0.04 0.06 0.08]),
%     set(gca,'XTick',[0 1 2 3 4 5 6 7 8]),
%     set(gca,'YGrid', 'on' ),
% subplot(1,7,2), bar(shr2_1245'), title('shr2'), legend('1','2','4','5'),
%     axis([0 8 -.1 0.08]),
%     set(gca,'YTick',[-0.08 -0.06 -0.04 -0.02 0 0.02 0.04 0.06 0.08]),
%     set(gca,'XTick',[0 1 2 3 4 5 6 7 8]),
```

```
% subplot(1,7,3), bar(shr3_13'), title('shr3'), legend('1','3'),
%     axis([0 8 -.1 0.08]),
%     set(gca,'YTick',[-0.08 -0.06 -0.04 -0.02 0 0.02 0.04 0.06 0.08]),
%     set(gca,'XTick',[0 1 2 3 4 5 6 7 8]),
% subplot(1,7,4), bar(shr5_123'), title('shr5'), legend('1','2','3'),
%     axis([0 8 -.1 0.08]),
%     set(gca,'YTick',[-0.08 -0.06 -0.04 -0.02 0 0.02 0.04 0.06 0.08]),
%     set(gca,'XTick',[0 1 2 3 4 5 6 7 8]),
% subplot(1,7,5), bar(shr6_13'), title('shr6'), legend('1','3'),
%     axis([0 8 -.1 0.08]),
%     set(gca,'YTick',[-0.08 -0.06 -0.04 -0.02 0 0.02 0.04 0.06 0.08]),
%     set(gca,'XTick',[0 1 2 3 4 5 6 7 8]),
% subplot(1,7,6), bar(shr7_134'), title('shr7'), legend('1','3','4'),
%     axis([0 8 -.1 0.08]),
%     set(gca,'YTick',[-0.08 -0.06 -0.04 -0.02 0 0.02 0.04 0.06 0.08]),
%     set(gca,'XTick',[0 1 2 3 4 5 6 7 8]),
% subplot(1,7,7), bar(shr8_13'), title('shr8'), legend('1','3'),
%     axis([0 8 -.1 0.08]),
%     set(gca,'YTick',[-0.08 -0.06 -0.04 -0.02 0 0.02 0.04 0.06 0.08]),
%     set(gca,'XTick',[0 1 2 3 4 5 6 7 8]),
%
% % WKY's
% figure,
% subplot(1,7,1), bar(wky2_345'), title('wky2'), legend('3','4','5'),
%     axis([0 8 -.1 0.08]),
%     set(gca,'YTick',[-0.08 -0.06 -0.04 -0.02 0 0.02 0.04 0.06 0.08]),
%     set(gca,'XTick',[0 1 2 3 4 5 6 7 8]),
%     set(gca,'YGrid', 'on' ),
% subplot(1,7,2), bar(wky3_1234'), title('wky3'), legend('1','2','3','4'),
%     axis([0 8 -.1 0.08]),
%     set(gca,'YTick',[-0.08 -0.06 -0.04 -0.02 0 0.02 0.04 0.06 0.08]),
%     set(gca,'XTick',[0 1 2 3 4 5 6 7 8]),
% subplot(1,7,3), bar(wky4_1'), title('wky4'), legend('1','3'),
%     axis([0 8 -.1 0.08]),
%     set(gca,'YTick',[-0.08 -0.06 -0.04 -0.02 0 0.02 0.04 0.06 0.08]),
%     set(gca,'XTick',[0 1 2 3 4 5 6 7 8]),
% subplot(1,7,4), bar(wky5_1345'), title('wky5'), legend('1','3','4','5'),
%     axis([0 8 -.1 0.08]),
%     set(gca,'YTick',[-0.08 -0.06 -0.04 -0.02 0 0.02 0.04 0.06 0.08]),
%     set(gca,'XTick',[0 1 2 3 4 5 6 7 8]),
% subplot(1,7,5), bar(wky6_135'), title('wky6'), legend('1','3','5'),
%     axis([0 8 -.1 0.08]),
%     set(gca,'YTick',[-0.08 -0.06 -0.04 -0.02 0 0.02 0.04 0.06 0.08]),
%     set(gca,'XTick',[0 1 2 3 4 5 6 7 8]),
% subplot(1,7,6), bar(wky7_145'), title('wky7'), legend('1','4','5'),
%     axis([0 8 -.1 0.08]),
%     set(gca,'YTick',[-0.08 -0.06 -0.04 -0.02 0 0.02 0.04 0.06 0.08]),
%     set(gca,'XTick',[0 1 2 3 4 5 6 7 8]),
% subplot(1,7,7), bar(wky8_13'), title('wky8'), legend('1','3'),
%     axis([0 8 -.1 0.08]),
%     set(gca,'YTick',[-0.08 -0.06 -0.04 -0.02 0 0.02 0.04 0.06 0.08]),
%     set(gca,'XTick',[0 1 2 3 4 5 6 7 8]),


%% Organize fiber strains by time %%
```

```
% SHR's
shr_fs1 = [shr1_1; shr2_1245(1,:); shr3_13(1,:); shr5_123(1,:); ...
           shr6_13(1,:); shr7_134(1,:); shr8_13(1,:)];
shr_fs2 = [shr2_1245(2,:); shr5_123(2,:)];
shr_fs3 = [shr3_13(2,:); shr5_123(3,:); shr6_13(2,:); shr7_134(2,:);
shr8_13(2,:)];
shr_fs4 = [shr2_1245(3,:); shr7_134(3,:)];
shr_fs5 = shr2_1245(4,:);

%WKY's
wky_fs1 = [wky3_1234(1,:); ...wky4_1(1,:); ...
           ...wky5_1345(1,:); ...
           wky6_1345(1,:); wky7_1345(1,:); wky8_13(1,:)];
wky_fs2 = [wky3_1234(2,:)];
wky_fs3 = [wky2_345(1,:);wky3_1234(3,:); wky5_1345(2,:); ...
           wky6_1345(2,:); wky7_1345(2,:); wky8_13(2,:)];
wky_fs4 = [wky2_345(2,:); ...wky3_1234(4,:);
            wky5_1345(3,:); ...
            wky6_1345(3,:); wky7_1345(3,:)];
wky_fs5 = [wky2_345(3,:); ...
           wky5_1345(4,:); ...
           wky6_1345(4,:); wky7_1345(4,:)];

% get the means
shr_fsm(1,1:7) = mean(shr_fs1,1);
shr_fsm(2,1:7) = mean(shr_fs2,1);
shr_fsm(3,1:7) = mean(shr_fs3,1);
shr_fsm(4,1:7) = mean(shr_fs4,1);
shr_fsm(5,1:7) = mean(shr_fs5,1);
shr_fsm = [zeros(5,1), shr_fsm, zeros(5,1)];

wky_fsm(1,1:7) = mean(wky_fs1,1);
wky_fsm(2,1:7) = mean(wky_fs2,1);
wky_fsm(3,1:7) = mean(wky_fs3,1);
wky_fsm(4,1:7) = mean(wky_fs4,1);
wky_fsm(5,1:7) = mean(wky_fs5,1);
wky_fsm = [zeros(5,1), wky_fsm, zeros(5,1)];


% get the std deviations
shr_fs_std(1,1:7) = std(shr_fs1);
shr_fs_std(2,1:7) = std(shr_fs2);
shr_fs_std(3,1:7) = std(shr_fs3);
shr_fs_std(4,1:7) = std(shr_fs4);
shr_fs_std(5,1:7) = std(shr_fs5);
shr_fs_std = [zeros(5,1), shr_fs_std, zeros(5,1)];

wky_fs_std(1,1:7) = std(wky_fs1);
wky_fs_std(2,1:7) = std(wky_fs2);
wky_fs_std(3,1:7) = std(wky_fs3);
wky_fs_std(4,1:7) = std(wky_fs4);
wky_fs_std(5,1:7) = std(wky_fs5);
wky_fs_std = [zeros(5,1), wky_fs_std, zeros(5,1)];

% get the p values
```

```
[hd(1) pd(1)] = ttest2(shr_fs1(:,3),wky_fs1(:,3),0.05,0);
[hd(2) pd(2)] = ttest2(shr_fs2(:,3),wky_fs2(:,3),0.05,0);
[hd(3) pd(3)] = ttest2(shr_fs3(:,3),wky_fs3(:,3),0.05,0);
[hd(4) pd(4)] = ttest2(shr_fs4(:,3),wky_fs4(:,3),0.05,0);
[hd(5) pd(5)] = ttest2(shr_fs5(:,3),wky_fs5(:,3),0.05,0);

[hs(1) ps(1)] = ttest2(shr_fs1(:,7),wky_fs1(:,7),0.05,0);
[hs(2) ps(2)] = ttest2(shr_fs2(:,7),wky_fs2(:,7),0.05,0);
[hs(3) ps(3)] = ttest2(shr_fs3(:,7),wky_fs3(:,7),0.05,0);
[hs(4) ps(4)] = ttest2(shr_fs4(:,7),wky_fs4(:,7),0.05,0);
[hs(5) ps(5)] = ttest2(shr_fs5(:,7),wky_fs5(:,7),0.05,0);


% Plot average (for a given time point) of average fiber strains
% with error bars at +/- 1 std dev
% top row is SHR, bottom row is WKY
% left to right is Acquisitions 1 to 5
figure,
for i=[1,3,4,5]
    subplot(2,5,i), plot([0:8],[shr_fsm(i,:)], 'ks-', 'LineWidth',1,...
        'MarkerEdgeColor','k','MarkerFaceColor',[1 1 1], 'MarkerSize',5),
    axis([0 8 -.1 0.08]), hold on,
    errorbar([0:8],shr_fsm(i,:),shr_fs_std(i,:)),
    set(gca,'YTick',[-0.08 -0.06 -0.04 -0.02 0 0.02 0.04 0.06 0.08]),
    set(gca,'XTick',[0 1 2 3 4 5 6 7 8]),
    %set(gca,'YGrid', 'on' ),
    hold off
end

for i=[1,3,4,5]
    subplot(2,5,i+5), plot([0:8],[wky_fsm(i,:)], 'ks-', 'LineWidth',1,...
        'MarkerEdgeColor','k','MarkerFaceColor',[1 1 1], 'MarkerSize',5),
    axis([0 8 -.1 0.08]), hold on,
    errorbar([0:8],wky_fsm(i,:),wky_fs_std(i,:)),
    set(gca,'YTick',[-0.08 -0.06 -0.04 -0.02 0 0.02 0.04 0.06 0.08]),
    set(gca,'XTick',[0 1 2 3 4 5 6 7 8]),
    %set(gca,'YGrid', 'on' ),
    hold off
end


%% NEW - Organize fiber strains by time with zero-padding for empty spaces%%
% SHR's
shr_fs1z = [shr1_1; shr2_1245(1,:); shr3_13(1,:); zeros(1,7); ...
            shr5_123(1,:); shr6_13(1,:); shr7_134(1,:); shr8_13(1,:)];
shr_fs2z = [zeros(1,7); shr2_1245(2,:); zeros(1,7); zeros(1,7); ...
            shr5_123(2,:); zeros(1,7);zeros(1,7);zeros(1,7)];
shr_fs3z = [zeros(1,7);zeros(1,7);shr3_13(2,:);zeros(1,7); ...
            shr5_123(3,:); shr6_13(2,:); shr7_134(2,:); shr8_13(2,:)];
shr_fs4z = [zeros(1,7); shr2_1245(3,:); zeros(1,7); zeros(1,7); ...
            zeros(1,7); zeros(1,7); shr7_134(3,:); zeros(1,7)];
shr_fs5z = [zeros(1,7); shr2_1245(4,:); zeros(1,7); zeros(1,7); ...
            zeros(1,7); zeros(1,7); zeros(1,7); zeros(1,7)];

%WKY's
```

```
wky_fs1z = [zeros(2,7);  wky3_1234(1,:); zeros(1,7);; ...
           wky5_1345(1,:); wky6_1345(1,:); wky7_1345(1,:); wky8_13(1,:)];
wky_fs2z = [zeros(2,7);  wky3_1234(2,:); zeros(1,7); ...
            zeros(1,7); zeros(1,7); zeros(1,7); zeros(1,7)];
wky_fs3z = [zeros(1,7); wky2_345(1,:); wky3_1234(3,:); zeros(1,7); ...
            wky5_1345(2,:); wky6_1345(2,:); wky73; wky8_13(2,:)];
wky_fs4z = [zeros(1,7); wky2_345(2,:); zeros(1,7);; zeros(1,7); ...
            wky5_1345(3,:); wky6_1345(3,:); wky7_1345(3,:); zeros(1,7)];
wky_fs5z = [zeros(1,7); wky2_345(3,:); zeros(1,7); zeros(1,7); ...
           wky5_1345(4,:); wky6_1345(4,:); wky7_1345(4,:); zeros(1,7)];

%-------------------------------------------------------------%
%%% Look at strains for a given time point to see variation %%%

%c = {'bo-','rs-','k^-','k^-','m*-','gd-','cp-','y+-'};
%c = {'ko-','kx-','k+-','k*-','ks-','kd-','kv-','k^-'};
c = {'bo-','ro-','co-','ko-','mo-','go-','ko-','yo-'};


colvec = [ [.8 .1 .1];[.5 .5 .1];[.1 .7 .1];[.1 .1 .8];[.6 .1 .6] ];

colvec2 = [ [.5 .5 .5];[.6 .1 .6];[.1 .1 .7];[.2 .6 .7];[.2 .6 .1];...
            [.7 .7 .3];[.9 .5 .1];[.8 .2 .4];[.8 .1 .6];[.1 .1 .1] ];

figure,
subplot(1,4,1),
    for i=[1,2,3,5,6,7,8]
        plot([0:8],[0, shr_fs1z(i,:),
0],'Color',colvec2(i,:),'Marker','d',...
            'MarkerFaceColor',colvec2(i,:), ...
            'markers',5),hold on,
    end, hold off,
    ylabel('Fiber Strain'), xlabel('One Cardiac Cycle'), title('Acquisition
1'),
    legend('SHR 1','SHR 2','SHR 3','SHR 5','SHR 6','SHR 7','SHR 8',3),
    axis([0 8 -.1 0.08]),
    set(gca,'YTick',[-0.08 -0.06 -0.04 -0.02 0 0.02 0.04 0.06 0.08]),
    set(gca,'XTick',[0 1 2 3 4 5 6 7 8]),
    set(gca,'PlotBoxAspectRatio',[1 2 1]),
subplot(1,4,2),
    for i=[3,5,6:8]
        plot([0:8],[0, shr_fs3z(i,:),
0],'Color',colvec2(i,:),'Marker','d',...
            'MarkerFaceColor',colvec2(i,:), ...
            'markers',5),hold on,
    end, hold off,
    title('Acquisition 3'),
    %legend('SHR 3','SHR 5','SHR 6','SHR 7','SHR 8',3),
    axis([0 8 -.1 0.08]),xlabel('One Cardiac Cycle'),
    set(gca,'YTick',[-0.08 -0.06 -0.04 -0.02 0 0.02 0.04 0.06 0.08]),
    set(gca,'XTick',[0 1 2 3 4 5 6 7 8]),
    set(gca,'PlotBoxAspectRatio',[1 2 1]),
subplot(1,4,3),
    for i=[2,7]
        plot([0:8],[0, shr_fs4z(i,:),
```

```
0],'Color',colvec2(i,:),'Marker','d',...
            'MarkerFaceColor',colvec2(i,:), ...
            'markers',5),hold on,
    end, hold off,
    title('Acquisition 4'),
    %legend('SHR 2','SHR 7',3),
    axis([0 8 -.1 0.08]), xlabel('One Cardiac Cycle'),
    set(gca,'YTick',[-0.08 -0.06 -0.04 -0.02 0 0.02 0.04 0.06 0.08]),
    set(gca,'XTick',[0 1 2 3 4 5 6 7 8]),
    set(gca,'PlotBoxAspectRatio',[1 2 1]),
subplot(1,4,4),
    for i=[2]
        plot([0:8],[0, shr_fs5z(i,:),
0],'Color',colvec2(i,:),'Marker','d',...
            'MarkerFaceColor',colvec2(i,:), ...
            'markers',5),hold on,
    end, hold off,
    title('Acquisition 5'),
    %legend('SHR 2',3),
    xlabel('One Cardiac Cycle'),
    axis([0 8 -.1 0.08]),
    set(gca,'YTick',[-0.08 -0.06 -0.04 -0.02 0 0.02 0.04 0.06 0.08]),
    set(gca,'XTick',[0 1 2 3 4 5 6 7 8]),
    set(gca,'PlotBoxAspectRatio',[1 2 1]),


%WKY's
figure,
subplot(1,4,1),
    for i=[3,6:8]
        plot([0:8],[0, wky_fs1z(i,:),
0],'Color',colvec2(i,:),'Marker','d',...
            'MarkerFaceColor',colvec2(i,:), ...
            'markers',5),hold on,
    end, hold off,
    title('Acquisition 1'), xlabel('One Cardiac Cycle'), ylabel('Fiber
Strain'),
    %legend('WKY 3','WKY 5','WKY 6','WKY 7','WKY 8',3),
    axis([0 8 -.1 0.08]),
    set(gca,'YTick',[-0.08 -0.06 -0.04 -0.02 0 0.02 0.04 0.06 0.08]),
    set(gca,'XTick',[0 1 2 3 4 5 6 7 8]),
    set(gca,'PlotBoxAspectRatio',[1 2 1]),
    %set(gca,'YGrid', 'on' ),
subplot(1,4,2),
    for i=[2,3,5,6,7,8]
        plot([0:8],[0, wky_fs3z(i,:),
0],'Color',colvec2(i,:),'Marker','d',...
            'MarkerFaceColor',colvec2(i,:), ...
            'markers',5),hold on,
    end, hold off,
    title('Acquisition 3'), legend('WKY 2','WKY 3','WKY 5','WKY 6','WKY
7','WKY 8',3),
    axis([0 8 -.1 0.08]),  xlabel('One Cardiac Cycle'),
    set(gca,'YTick',[-0.08 -0.06 -0.04 -0.02 0 0.02 0.04 0.06 0.08]),
    set(gca,'XTick',[0 1 2 3 4 5 6 7 8]),
```

```
    set(gca,'PlotBoxAspectRatio',[1 2 1]),
subplot(1,4,3),
    for i=[2,5,6,7]
        plot([0:8],[0, wky_fs4z(i,:),
0],'Color',colvec2(i,:),'Marker','d',...
            'MarkerFaceColor',colvec2(i,:), ...
            'markers',5),hold on,
    end, hold off,
    title('Acquisition 4'),
    %legend('WKY 2','WKY 5','WKY 6','WKY 7',3),
    axis([0 8 -.1 0.08]), xlabel('One Cardiac Cycle'),
    set(gca,'YTick',[-0.08 -0.06 -0.04 -0.02 0 0.02 0.04 0.06 0.08]),
    set(gca,'XTick',[0 1 2 3 4 5 6 7 8]),
    set(gca,'PlotBoxAspectRatio',[1 2 1]),
subplot(1,4,4),
    for i=[2,5,6,7]
        plot([0:8],[0, wky_fs5z(i,:),
0],'Color',colvec2(i,:),'Marker','d',...
            'MarkerFaceColor',colvec2(i,:), ...
            'markers',5),hold on,
    end, hold off,
    title('Acquisition 5'),
    %legend('WKY 2','WKY 5','WKY 6','WKY 7',3),
    axis([0 8 -.1 0.08]),  xlabel('One Cardiac Cycle'),
    set(gca,'YTick',[-0.08 -0.06 -0.04 -0.02 0 0.02 0.04 0.06 0.08]),
    set(gca,'XTick',[0 1 2 3 4 5 6 7 8]),
    set(gca,'PlotBoxAspectRatio',[1 2 1]),
```

D.4 : Calculating Regional Strains

CalcRegionalStrains6.m

```
clear all; close all; clc;

%THIS SCRIPT WAS USED TO CALUCULATE THE 4-REGION STRAINS FOR ALL OF THE
%RATS.
%NOTE: THIS CODE WILL TAKE THE STRAINS FROM THE EQUATORIAL SLICE ONLY (2ND
%SLICE FROM THE BASE OUT OF 4 SLICES).
%IT SAVES THIS DATA TO THE FILE FSR_all_4reg_equator, WHICH CONTAINS
%ONE VARIABLE: FSR_all


format long
load names.mat

nr = 4;
FSR_all = zeros(nr,9,42); %Regional fiber strain array to contain values from
all data sets.

for i = [1:17,19,21:34,36:37,39:length(names)]
```

```
    % Construct the path to look for text files relating to this data set
    if (names{i}(1:3) == 'shr')
        path = strcat('C:\Documents and Settings\Genevieve\Desktop\RATS',...
                      '\SHR',names{i}(4:6),'\');
    elseif (names{i}(1:3) == 'wky')
        path = strcat('C:\Documents and Settings\Genevieve\Desktop\RATS',...
                      '\WKY',names{i}(4:6),'\');
    end

    [L] = GetLandmarks( names{i} );

    nodefile = strcat( path, names{i}(1:4), names{i}(6), 'nodes.txt');

    nodes = GetNodes(nodefile,5,15448);

    R = AssignRegions3( nodes, 4, [4,4,4,4], L);

    FS = zeros(15448,9); %Initialize fiber strain array for each data set.
Columns 1 an 9 will remain zeros.

    for j = 2:8
        strainfile = strcat(path,'fs',num2str(j-1),'.txt' );
        FS(:,j) = dlmread(strainfile, ' ', [5 1 15452 1]);
        FS(:,j) = FS(:,j)-1; %Remember to subtract one so that zero strain is
0 not 1
        for r = 1:nr
            c=0;
            for k = 1:15448
                if (R(k,1) == r && R(k,2) == 2)
                    c = c+1;
                    fsr(c) = FS(k,j);
                end
            end
            FSR_all(r,j,i) = mean(fsr);
            clear fsr;
        end
    end




%    %%%% PLOT %%%%
%
%    col = [ [.5 .5 .5];[.6 .1 .6];[.1 .1 .7];[.2 .6 .7];[.2 .6 .1];...
%              [.7 .7 .3];[.9 .5 .1];[.8 .2 .4];[.8 .1 .6];[.1 .1 .1];...
%              [.5 .5 .5];[.6 .1 .6];[.1 .1 .7];[.2 .6 .7];[.2 .6 .1];...
%              [.7 .7 .3];[.9 .5 .1];[.8 .2 .4];[.8 .1 .6];[.1 .1 .1];...
%              [.5 .5 .5];[.6 .1 .6];[.1 .1 .7];[.2 .6 .7];[.2 .6 .1];...
%              [.7 .7 .3];[.9 .5 .1];[.8 .2 .4];[.8 .1 .6];[.1 .1 .1] ];
%
%
%    figure,
%    for k = 1:nr
%         plot( 1:9, FSR(k,:), ...
%             'Color', col(k,:),...
```

```
%               'Marker','d',...
%               'MarkerFaceColor',col(k,:) ), hold on,
%    end
%    ylabel('Fiber Strain'),
%    xlabel('One Cardiac Cycle'),
%    title(strcat('Regional Fiber Strains for',names{i})),
%    legend('anterior','septal','inferior','lateral'),
%    axis([1 9 -.1 0.1]),
%    %set(gca,'YTick',[-0.08 -0.06 -0.04 -0.02 0 0.02 0.04 0.06 0.08]),
%    set(gca,'XTick',[0 1 2 3 4 5 6 7 8]),
%    set(gca,'YGrid', 'on' ),
%    hold off


end

save FSR_all_4reg_equator;
```

AssignRegions3.m

```
function [R] = AssignRegions3( C, n_z_r, n_xy_r, L )


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%
%function R = AssignRegions( C, n_z_r, n_xy_r, xy_as, xy_is )
%
% inputs:   C: Nodal Coordinates. Nx3 array where each row contains the
%               x, y, and z coordinates of the N'th node.
%           n_z_r: Number of Z Regions. scalar-valued. any number ok.)
%           n_xy_r: Number of XY Regions (for each z-region).
%                 vector of length n_z_r, can contain values of 1, 4, or 6
ONLY.
%               e.g.: n_xy_r = [6; 6; 4; 1] for n_z_r = 4 is AHA
Recommendation.
%           [L] = [ anteroseptal x-coord,  anteroseptal y-coord;
%                   inferoseptal x-coord,  inferoseptal y-coord;
%                   center x-coord,        center y-coord];
%
% output:   R: Nx3 array, where each row contains
%               (1) the x-y region number, (2) the z region number, and
%               (3) the AHA region number, for each of the N nodes
%
%e.g.: R = AssignRegions( C, 4, [6; 6; 4; 1], L)
%      R_shr25 = AssignRegions2( shr25nodes, 1, [4], L);
%
%G.E.FARRAR
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%
```

```
format long
N = size(C,1); %Number of nodes

%%%%  Initialize the array that will hold the region information.  %%%%
R = zeros(N,3);


%%%%  Assign the nodes to a z-region.  %%%%
%This based on each node's z-coordinate.
%The z-region number for each node is stored in the second column of R.
%Z-regions are numbered from base to apex.
%Currently, the code divides the mesh into equal-sized z-regions. Uneven
%spacing could be added.
if n_z_r < 1
    fprintf('\n Error: n_z_r cannot be negative.');

elseif n_z_r == 1
    R(:,2) = 1;

elseif n_z_r > 1

    z_max = max(C(:,3));
    z_min = min(C(:,3));
    z_chunk = (z_max - z_min)/n_z_r;

    for i = 1:(n_z_r + 1)
        z_slices(i) = z_max - (i-1)*z_chunk;
    end

    for i = 1:N
        z = C(i,3);
        for j = 1:n_z_r
            if z <= z_slices(j) && z >= z_slices(j+1)
                R(i,2) = j;
            end
        end
        if z == z_min
                R(i,2) = n_z_r;
        end

    end
end


%%%%  Check the z-region assignments  %%%%
%% Check that all nodes were assigned a z region
for i = 1:N
    if(R(i,2) == 0)
        fprintf(strcat(' \n Error: no z component for node#',num2str(i)));
    end
end


% %%%%%%%% plot x-z to see z-regions
% figure,
```

```
% for i = 1:N
%     if R(i,2) == 1
%         plot(C(i,1),C(i,3),'ko'), hold on,
%     elseif R(i,2) == 2
%         plot(C(i,1),C(i,3),'bo'), hold on,
%     elseif R(i,2) == 3
%         plot(C(i,1),C(i,3),'ro'), hold on,
%     elseif R(i,2) == 4
%         plot(C(i,1),C(i,3),'go'), hold on,
%     end
% end
% hold off
%
% %%%%%%%% plot y-z to see z-regions
% figure,
% for i = 1:N
%     if R(i,2) == 1
%         plot(C(i,2),C(i,3),'ko'), hold on,
%     elseif R(i,2) == 2
%         plot(C(i,2),C(i,3),'bo'), hold on,
%     elseif R(i,2) == 3
%         plot(C(i,2),C(i,3),'ro'), hold on,
%     elseif R(i,2) == 4
%         plot(C(i,2),C(i,3),'go'), hold on,
%     end
% end
% hold off



%%%% Assign the nodes to an xy-region %%%%

%% Find the slopes of the lines from the center to the anteroseptal
% and inferoseptal points
m_as = (L(1,2) - L(3,2))/(L(1,1) - L(3,1));
m_is = (L(2,2) - L(3,2))/(L(2,1) - L(3,1));

for i = 1:N

    %% Procedure for dividing the xy slice into 6 regions
    % if( R(i,2) == 3 || R(i,2) == 4) %assign directly to z-regions
    if n_xy_r(R(i,2)) == 6    %if this node has been assigned to a
                                  %z-region for which n_xy_r is 6
        x = C(i,1);
        y = C(i,2);
        y_as = m_as*(x-L(3,1)) + L(3,2);    % y=mx+b
        y_is = m_is*(x-L(3,1)) + L(3,2);
        y_cs = L(3,2);

        if y <= y_as
            if y <= y_is
                R(i,1) = 1;     %anterior region
            elseif y > y_is
                if y < y_cs
                    R(i,1) = 6; %antero-lateral region
```

```
                else
                    R(i,1) = 5; %infero-lateral region
                end
            end
        elseif y > y_as
            if y > y_is
                R(i,1) = 4;     %inferior region
            elseif y <= y_is
                if y > y_cs
                    R(i,1) = 3; %infero-septal region
                else
                    R(i,1) = 2; %antero-septal region
                end
            end
        end

    %% Procedure for dividing the xy slice into 4 regions
     % elseif( R(i,2) == 2 ) %directly assign to z-region numbers
    elseif n_xy_r(R(i,2)) == 4      %if this node has been assigned to a
                                          %z-region for which n_xy_r is 4
      x = C(i,1);
      y = C(i,2);
      y_as = m_as*(x-L(3,1)) + L(3,2);
      y_is = m_is*(x-L(3,1)) + L(3,2);
      if y <= y_as
            if y <= y_is
                  R(i,1) = 1;
            elseif y > y_is
                  R(i,1) = 4;
            end
      elseif y > y_as
            if y > y_is
                  R(i,1) = 3;
            elseif y <= y_is
                  R(i,1) = 2;
            end
      end

    %% This option just assigs the whole xy-plane to one region
     % elseif( R(i,2) == 1 ) %directly assign to z-region numbers
    elseif n_xy_r(R(i,2)) == 1
        R(i,1) = 1;
    else
        fprintf('\n Error: Invalid n_xy_r');
    end
end

%%%%  Check the xy-region assignments  %%%%
%% Check that all nodes were assigned an xy region
for i = 1:N
    if(R(i,1) == 0)
        fprintf(strcat(' \n Error: No xy region assigned for
i=',num2str(i)));
    end
end
```

```
% %% Check the code by plotting the regions
% for j = 1:n_z_r
%   figure,
%   for i = 1:N
%         if R(i,2) == j
%             if R(i,1) == 1
%                 plot(C(i,1),C(i,2),'ko'), hold on
%             elseif R(i,1) == 2
%                 plot(C(i,1),C(i,2),'bo'), hold on
%             elseif R(i,1) == 3
%                 plot(C(i,1),C(i,2),'ro'), hold on
%             elseif R(i,1) == 4
%                 plot(C(i,1),C(i,2),'go'), hold on
%             elseif R(i,1) == 5
%                 plot(C(i,1),C(i,2),'co'), hold on
%             elseif R(i,1) == 6
%                 plot(C(i,1),C(i,2),'mo'), hold on
%             end
%         end
%   end
%   plot(L(3,1),L(3,2),'rs'), hold on
% end

%%% Set up the third column of AHA region numbers
if n_z_r == 4
    for i=1:N
        if R(i,2) == 1
            R(i,3) = R(i,1);
        elseif R(i,2) == 2
            R(i,3) = R(i,1) + 6;
        elseif R(i,2) == 3
            R(i,3) = R(i,1) + 12;
        elseif R(i,2) == 4
            R(i,3) = 17;
        end
    end
end
```

GetLandmarks.m

```
function L = GetLandmarks( name )

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Gets anatomical landmark coordinates from text file
%
% Input:
%   name = name of rat data set. (e.g. 'shr2_5')
%
% Output:
%   [L] = [ anteroseptal x-coord,  anteroseptal y-coord;
%           inferoseptal x-coord,  inferoseptal y-coord;
%           center x-coord,        center y-coord];
```

```
%
%G.E.FARRAR
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%



in = fopen('ImageSectionPoints.txt','r');
line = fgetl(in);
line_no = 0;
while(line ~= -1)
    line_no = line_no + 1;
    if(strcmp(line(1:6),name))
        break
    end
    line = fgetl(in);
end

Data = dlmread('ImageSectionPoints.txt', ' ', [line_no-1 1 line_no-1 6]);
L = [Data(1:2);
     Data(3:4);
     Data(5:6)];
```

Chapter 6. Bibliography

[1] Roger, V. L., Go, A. S., Lloyd-Jones, D. M., Benjamin, E. J., Berry, J. D., Borden, W. B., Bravata, D. M., Dai, S., Ford, E. S., Fox, C. S., Fullerton, H. J., Gillespie, C., Hailpern, S. M., Heit, J. A., Howard, V. J., Kissela, B. M., Kittner, S. J., Lackland, D. T., Lichtman, J. H., Lisabeth, L. D., Makuc, D. M., Marcus, G. M., Marelli, A., Matchar, D. B., Moy, C. S., Mozaffarian, D., Mussolino, M. E., Nichol, G., Paynter, N. P., Soliman, E. Z., Sorlie, P. D., Sotoodehnia, N., Turan, T. N., Virani, S. S., Wong, N. D., Woo, D., and Turner, M. B., 2012, "Heart disease and stroke statistics--2012 update: a report from the American Heart Association," Circulation, 125(1), pp. e2-e220.

[2] Janz, R. F., and Grimm, A. F., 1972, "Finite-element model for the mechanical behavior of the left ventricle. Prediction of deformation in the potassium-arrested rat heart," Circulation research, 30(2), pp. 244-252.

[3] Guccione, J. M., Costa, K. D., and McCulloch, A. D., 1995, "Finite element stress analysis of left ventricular mechanics in the beating dog heart," J Biomech, 28(10), pp. 1167-1177.

[4] Hunter, P. J., Nielsen, P. M., Smaill, B. H., LeGrice, I. J., and Hunter, I. W., 1992, "An anatomical heart model with applications to myocardial activation and ventricular mechanics," Crit Rev Biomed Eng, 20(5-6), pp. 403-426.

[5] Levy, D., Garrison, R. J., Savage, D. D., Kannel, W. B., and Castelli, W. P., 1990, "Prognostic implications of echocardiographically determined left ventricular mass in the Framingham Heart Study," The New England journal of medicine, 322(22), pp. 1561-1566.

[6] Holzapfel, G. A., 2000, Nonlinear solid mechanics : a continuum approach for engineering, Wiley, Chichester ; New York.

[7] Malvern, L. E., 1969, Introduction to the mechanics of a continuous medium, Prentice-Hall, Englewood Cliffs, N.J.,.

[8] Zienkiewicz, O. C., and Taylor, R. L., 2000, The finite element method, Butterworth-Heinemann, Oxford ; Boston.

[9] Logan, D. L., 2007, A first course in the finite element method, Thomson, United States.

[10] Steve Maas, D. R., Jeffrey Weiss, Gerard Ateshian, 2012, FEBio Theory Manual, Musculoskeletal Research Laboratories, University of Utah.

[11] Bradley Maker, R. F., John Halquist, 1995, NIKE3D A Nonlinear Implicit Three-Dimensional Finite Element Code for Solid and Structural Mechanics User's Manual, Methods Development Group, Lawrence Livermore National Laboratory.

[12] Gullberg, G. T., Reutter, B. W., Sitek, A., Maltz, J. S., and Budinger, T. F., "Dynamic single photon emission computed tomography--basic principles and cardiac applications," Physics in medicine and biology, 55(20), pp. R111-191.

[13] Hernandez, A. M., Huber, J. S., Murphy, S. T., Janabi, M., Zeng, G. L., Brennan, K. M., O'Neil, J. P., Seo, Y., and Gullberg, G. T., 2013, "Longitudinal Evaluation of Left Ventricular Substrate Metabolism, Perfusion, and Dysfunction in the Spontaneously Hypertensive Rat Model of Hypertrophy Using microPET/CT Imaging," J Nucl Med.

[14] Reutter, B. W., Huesman, R. H., Brennan, K. M., Boutchko, R., Hanrahan, S. M., and Gullberg, G. T., 2011, "Longitudinal Evaluation of Fatty Acid Metabolism in Normal and Spontaneously Hypertensive Rat Hearts with Dynamic MicroSPECT Imaging," International journal of molecular imaging, 2011, p. 893129.
[15] Veress, A. I., Weiss, J. A., Huesman, R. H., Reutter, B. W., Taylor, S. E., Sitek, A., Feng, B., Yang, Y., and Gullberg, G. T., 2008, "Measuring regional changes in the diastolic deformation of the left ventricle of SHR rats using microPET technology and hyperelastic warping," Ann Biomed Eng, 36(7), pp. 1104-1117.
[16] Veress AI, K. G., and Gullberg GT, 2013, "A Comparison of Hyperelastic Warping of PET Images with Tagged MRI for the Analysis of Cardiac Deformation," International Journal of Biomedical Imaging, 2013.
[17] Cherry, S. R., Shao, Y., Silverman, R. W., Siegel, S., Meadors, K., Mumcuoglu, E., Young, J., Jones, W. F., Moyers, C., Andreaco, M., Paulus, M., Binkley, D., Nutt, R., and Phelps, M. E., 1996, "Micropet: A dedicated PET scanner for small animal imaging.," J Nucl Med, 37(5), pp. 334-334.
[18] Novelline, R. A., and Squire, L. F., 1997, Squire's fundamentals of radiology, Harvard University Press, Cambridge, Mass.
[19] Rohmer, D., Sitek, A., and Gullberg, G. T., 2007, "Reconstruction and visualization of fiber and laminar structure in the normal human heart from ex vivo diffusion tensor magnetic resonance imaging (DTMRI) data," Investigative radiology, 42(11), pp. 777-789.
[20] Katz, A. M., 2011, Physiology of the heart, Wolters Kluwer Health/Lippincott Williams & Wilkins Health, Philadelphia, PA.
[21] Humphrey, J. D., 2002, Cardiovascular Solid Mechanics: Cells, Tissues, and Organs, Springer %@ 0387951687 %L RC669.9 .H85 2002, New York.
[22] Berne, R. M., and Levy, M. N., 1993, Physiology, Mosby Year Book, St. Louis.
[23] Holzapfel, G. A., and Ogden, R. W., 2009, "Constitutive modelling of passive myocardium: a structurally based framework for material characterization," Philosophical Transactions: Mathematical, Physical & Engineering Sciences, 367(1902), pp. 3445-3475.
[24] Tezuka, F., 1975, "Muscle-Fiber Orientation in Normal and Hypertrophied Hearts," Tohoku J Exp Med, 117(3), pp. 289-297.
[25] Demer, L., and Yin, F. C. P., 1981, "Passive Biaxial Stress-Strain Properties of Excised Canine Myocardium," Ieee T Bio-Med Eng, 28(8), pp. 584-584.
[26] Sacks, M. S., and Chuong, C. J., 1993, "Biaxial Mechanical-Properties of Passive Right-Ventricular Free Wall Myocardium," J Biomech Eng-T Asme, 115(2), pp. 202-205.
[27] Lin, D. H., and Yin, F. C. P., 1995, "Biaxial Mechanical-Properties of Barium-Activated Left-Ventricular Myocardium," Faseb J, 9(3), pp. A559-A559.
[28] Emery, J. L., Omens, J. H., and McCulloch, A. D., 1997, "Biaxial mechanics of the passively overstretched left ventricle," Am J Physiol-Heart C, 272(5), pp. H2299-H2305.
[29] Huxley, A. F., and Niedergerke, R., 1954, "Structural changes in muscle during contraction; interference microscopy of living muscle fibres," Nature, 173(4412), pp. 971-973.
[30] Huxley, H., and Hanson, J., 1954, "Changes in the cross-striations of muscle during contraction and stretch and their structural interpretation," Nature, 173(4412), pp. 973-976.
[31] Humphrey, J. D., and Rajagopal, K. R., 2002, "A constrained mixture model for growth and remodeling of soft tissues," Math Mod Meth Appl S, 12(3), pp. 407-430.

[32] Guccione, J. M., and McCulloch, A. D., 1993, "Mechanics of active contraction in cardiac muscle: Part I--Constitutive relations for fiber stress that describe deactivation," Journal of biomechanical engineering, 115(1), pp. 72-81.
[33] Michael F. O'Rourke, A. P. A., Wilmer W. Nichols, 1987, "Left Ventricular-Systemic Arterial Coupling in Humans and Strategies to Improve Coupling in Disease States," Ventricular/Vascular Coupling, F. C. P. Yin, ed., Springer New York, pp. 1-19.
[34] Jones, D. W., and Hall, J. E., 2004, "Seventh Report of the Joint National Committee on Prevention, Detection, Evaluation, and Treatment of High Blood Pressure and evidence from new hypertension trials," Hypertension, 43(1), pp. 1-3.
[35] Carretero, O. A., and Oparil, S., 2000, "Essential hypertension. Part I: definition and etiology," Circulation, 101(3), pp. 329-335.
[36] Frohlich, E. D., Apstein, C., Chobanian, A. V., Devereux, R. B., Dustan, H. P., Dzau, V., Fauadtarazi, F., Horan, M. J., Marcus, M., Massie, B., Pfeffer, M. A., Re, R. N., Roccella, E. J., Savage, D., and Shub, C., 1992, "Medical Progress - the Heart in Hypertension," New Engl J Med, 327(14), pp. 998-1008.
[37] Devereux, R. B., and Roman, M. J., 1999, "Left ventricular hypertrophy in hypertension: stimuli, patterns, and consequences," Hypertens Res, 22(1), pp. 1-9.
[38] Treasure, C. B., Klein, J. L., Vita, J. A., Manoukian, S. V., Renwick, G. H., Selwyn, A. P., Ganz, P., and Alexander, R. W., 1993, "Hypertension and Left-Ventricular Hypertrophy Are Associated with Impaired Endothelium-Mediated Relaxation in Human Coronary Resistance Vessels," Circulation, 87(1), pp. 86-93.
[39] Sultana, R., Sultana, N., Rashid, A., Rasheed, S. Z., Ahmed, M., Ishaq, M., and Samad, A., 2010, "Cardiac arrhythmias and left ventricular hypertrophy in systemic hypertension," Journal of Ayub Medical College, Abbottabad : JAMC, 22(4), pp. 155-158.
[40] Bing, O. H., Conrad, C. H., Boluyt, M. O., Robinson, K. G., and Brooks, W. W., 2002, "Studies of prevention, treatment and mechanisms of heart failure in the aging spontaneously hypertensive rat," Heart failure reviews, 7(1), pp. 71-88.
[41] Bing, O. H., Brooks, W. W., Robinson, K. G., Slawsky, M. T., Hayes, J. A., Litwin, S. E., Sen, S., and Conrad, C. H., 1995, "The spontaneously hypertensive rat as a model of the transition from compensated left ventricular hypertrophy to failure," Journal of molecular and cellular cardiology, 27(1), pp. 383-396.
[42] Cunha, D. M., da Cunha, A. B., Martins, W. A., Pinheiro, L. A., Romeo, L. J., de Moraes, A. V., and Morcerf, F. P., 2001, "Echocardiographic assessment of the different left ventricular geometric patterns in hypertensive patients," Arquivos brasileiros de cardiologia, 76(1), pp. 15-28.
[43] Pfeffer, J. M., Pfeffer, M. A., Fishbein, M. C., and Frohlich, E. D., 1979, "Cardiac function and morphology with aging in the spontaneously hypertensive rat," Am J Physiol, 237(4), pp. H461-468.
[44] Shimbo, D., Muntner, P., Mann, D., Barr, R. G., Tang, W., Post, W., Lima, J., Burke, G., Bluemke, D., and Shea, S., 2011, "Association of left ventricular hypertrophy with incident hypertension: the multi-ethnic study of atherosclerosis," American journal of epidemiology, 173(8), pp. 898-905.
[45] Strotmann, J. M., Lengenfelder, B., Blondelot, J., Voelker, W., Herrmann, S., Ertl, G., and Weidemann, F., 2008, "Functional differences of left ventricular hypertrophy induced by either arterial hypertension or aortic valve stenosis," The American journal of cardiology, 101(10), pp. 1493-1497.

[46] Belenkov, Y., Vikhert, O. A., Belichenko, O. I., and Arabidze, G. G., 1992, "Magnetic resonance imaging of cardiac hypertrophy in malignant arterial hypertension," American journal of hypertension, 5(6 Pt 2), pp. 195S-199S.
[47] Wiegner, A. W., and Bing, O. H., 1979, "Laser scanner measurement of central segment performance in isolated cardiac muscle preparations," Am J Physiol, 237(2), pp. H260-264.
[48] Norton, G. R., Tsotetsi, J., Trifunovic, B., Hartford, C., Candy, G. P., and Woodiwiss, A. J., 1997, "Myocardial stiffness is attributed to alterations in cross-linked collagen rather than total collagen or phenotypes in spontaneously hypertensive rats," Circulation, 96(6), pp. 1991-1998.
[49] Tanaka, M., Fujiwara, H., Onodera, T., Wu, D. J., Hamashima, Y., and Kawai, C., 1986, "Quantitative analysis of myocardial fibrosis in normals, hypertensive hearts, and hypertrophic cardiomyopathy," British heart journal, 55(6), pp. 575-581.
[50] Tanaka, M., Fujiwara, H., Onodera, T., Wu, D. J., Matsuda, M., Hamashima, Y., and Kawai, C., 1987, "Pathogenetic role of myocardial fiber disarray in the progression of cardiac fibrosis in normal hearts, hypertensive hearts and hearts with hypertrophic cardiomyopathy," Japanese circulation journal, 51(6), pp. 624-630.
[51] Aronow, W. S., 2006, "Epidemiology, pathophysiology, prognosis, and treatment of systolic and diastolic heart failure," Cardiology in review, 14(3), pp. 108-124.
[52] Borlaug, B. A., and Redfield, M. M., "Diastolic and systolic heart failure are distinct phenotypes within the heart failure spectrum," Circulation, 123(18), pp. 2006-2013; discussion 2014.
[53] Chatterjee, K., and Massie, B., 2007, "Systolic and diastolic heart failure: Differences and similarities," Journal of cardiac failure, 13(7), pp. 569-576.
[54] Kouzu, H., Yuda, S., Muranaka, A., Doi, T., Yamamoto, H., Shimoshige, S., Hase, M., Hashimoto, A., Saitoh, S., Tsuchihashi, K., Miura, T., Watanabe, N., and Shimamoto, K., 2011, "Left ventricular hypertrophy causes different changes in longitudinal, radial, and circumferential mechanics in patients with hypertension: a two-dimensional speckle tracking study," J Am Soc Echocardiogr, 24(2), pp. 192-199.
[55] Holmes, J. W., 2004, "Candidate mechanical stimuli for hypertrophy during volume overload," J Appl Physiol (1985), 97(4), pp. 1453-1460.
[56] Palmon, L. C., Reichek, N., Yeon, S. B., Clark, N. R., Brownson, D., Hoffman, E., and Axel, L., 1994, "Intramural myocardial shortening in hypertensive left ventricular hypertrophy with normal pump function," Circulation, 89(1), pp. 122-131.
[57] Spottiswoode, B. S., Zhong, X., Lorenz, C. H., Mayosi, B. M., Meintjes, E. M., and Epstein, F. H., 2008, "3D myocardial tissue tracking with slice followed cine DENSE MRI," J Magn Reson Imaging, 27(5), pp. 1019-1027.
[58] Kahan, T., and Bergfeldt, L., 2005, "Left ventricular hypertrophy in hypertension: Its arrhythmogenic potential," Heart (British Cardiac Society), 91(2), pp. 250-256.
[59] de las Fuentes, L., Soto, P. F., Cupps, B. P., Pasque, M. K., Herrero, P., Gropler, R. J., Waggoner, A. D., and Davila-Roman, V. G., 2006, "Hypertensive left ventricular hypertrophy is associated with abnormal myocardial fatty acid metabolism and myocardial efficiency," J Nucl Cardiol, 13(3), pp. 369-377.
[60] Jhun, C. S., Wenk, J. F., Zhang, Z., Wall, S. T., Sun, K., Sabbah, H. N., Ratcliffe, M. B., and Guccione, J. M., "Effect of adjustable passive constraint on the failing left ventricle: a finite-element model study," Annals of Thoracic Surgery, 89(1), pp. 132-137.

[61] Klepach, D., Lee, L. C., Wenk, J. F., Ratcliffe, M. B., Zohdi, T. I., Navia, J. L., Kassab, G. S., Kuhl, E., and Guccione, J. M., 2012, "Growth and remodeling of the left ventricle: A case study of myocardial infarction and surgical ventricular restoration," Mech Res Commun, 42, pp. 134-141.
[62] Guccione, J. M., Walker, J. C., Beitler, J. R., Moonly, S. M., Zhang, P., Guttman, M. A., Ozturk, C., McVeigh, E. R., Wallace, A. W., Saloner, D. A., and Ratcliffe, M. B., 2006, "The effect of anteroapical aneurysm plication on end-systolic three-dimensional strain in the sheep: a magnetic resonance imaging tagging study," The Journal of thoracic and cardiovascular surgery, 131(3), pp. 579-586 e573.
[63] Walker, J. C., Ratcliffe, M. B., Zhang, P., Wallace, A. W., Hsu, E. W., Saloner, D. A., and Guccione, J. M., 2008, "Magnetic resonance imaging-based finite element stress analysis after linear repair of left ventricular aneurysm," The Journal of thoracic and cardiovascular surgery, 135(5), pp. 1094-1102, 1102 e1091-1092.
[64] Wenk, J. F., Zhang, Z., Cheng, G., Malhotra, D., Acevedo-Bolton, G., Burger, M., Suzuki, T., Saloner, D. A., Wallace, A. W., Guccione, J. M., and Ratcliffe, M. B., 2010, "First finite element model of the left ventricle with mitral valve: insights into ischemic mitral regurgitation," Annals of Thoracic Surgery, 89(5), pp. 1546-1553.
[65] Carrick, R., Ge, L., Lee, L. C., Zhang, Z., Mishra, R., Axel, L., Guccione, J. M., Grossi, E. A., and Ratcliffe, M. B., 2012, "Patient-specific finite element-based analysis of ventricular myofiber stress after Coapsys: importance of residual stress," The Annals of thoracic surgery, 93(6), pp. 1964-1971.
[66] Walker, J. C., Ratcliffe, M. B., Zhang, P., Wallace, A. W., Fata, B., Hsu, E. W., Saloner, D., and Guccione, J. M., 2005, "MRI-based finite-element analysis of left ventricular aneurysm," Am J Physiol Heart Circ Physiol, 289(2), pp. H692-700.
[67] Usyk, T. P., Omens, J. H., and McCulloch, A. D., 2001, "Regional septal dysfunction in a three-dimensional computational model of focal myofiber disarray," Am J Physiol-Heart C, 281(2), pp. H506-H514.
[68] Kroon, W., Delhaas, T., Bovendeerd, P., and Arts, T., 2009, "Computational analysis of the myocardial structure: adaptation of cardiac myofiber orientations through deformation," Medical image analysis, 13(2), pp. 346-353.
[69] Petitjean, C., and Dacher, J. N., 2011, "A review of segmentation methods in short axis cardiac MR images," Medical image analysis, 15(2), pp. 169-184.
[70] Phatak, N. S., Maas, S. A., Veress, A. I., Pack, N. A., Di Bella, E. V., and Weiss, J. A., 2009, "Strain measurement in the left ventricle during systole with deformable image registration," Medical image analysis, 13(2), pp. 354-361.
[71] Chaudhry, H. R., Bukiet, B., and Regan, T., 1997, "Dynamic stresses and strains in the left ventricular wall based on large deformation theory," Int J Nonlinear Mech, 32(3), pp. 621-631.
[72] Omens, J. H., McCulloch, A. D., and Criscione, J. C., 2003, "Complex distributions of residual stress and strain in the mouse left ventricle: experimental and theoretical models," Biomechanics and modeling in mechanobiology, 1(4), pp. 267-277.
[73] Wolff, J., 1892, Das Gesetz der Transformation der Knochen, August Hirschwald, Berlin,.
[74] Skalak, R., Dasgupta, G., Moss, M., Otten, E., Dullemeijer, P., and Vilmann, H., 1982, "Analytical Description of Growth," J Theor Biol, 94(3), pp. 555-577.

[75] Rodriguez, E. K., Hoger, A., and McCulloch, A. D., 1994, "Stress-dependent finite growth in soft elastic tissues," J Biomech, 27(4), pp. 455-467.
[76] Lee, E. H., 1969, "Elastic-Plastic Deformations at Finite Strains," Journal of Applied Mechanics, 36, pp. 1-6.
[77] Taber, L. A., and Eggers, D. W., 1996, "Theoretical study of stress-modulated growth in the aorta," J Theor Biol, 180(4), pp. 343-357.
[78] Kuhl, E., Maas, R., Himpel, G., and Menzel, A., 2007, "Computational modeling of arterial wall growth. Attempts towards patient-specific simulations based on computer tomography," Biomechanics and modeling in mechanobiology, 6(5), pp. 321-331.
[79] Taber, L. A., and Humphrey, J. D., 2001, "Stress-modulated growth, residual stress, and vascular heterogeneity," J Biomech Eng-T Asme, 123(6), pp. 528-535.
[80] Lin, I. E., and Taber, L. A., 1995, "A Model for Stress-Induced Growth in the Developing Heart," J Biomech Eng-T Asme, 117(3), pp. 343-349.
[81] Kerckhoffs, R. C., 2012, "Computational modeling of cardiac growth in the post-natal rat with a strain-based growth law," J Biomech, 45(5), pp. 865-871.
[82] Kroon, W., Delhaas, T., Arts, T., and Bovendeerd, P., 2009, "Computational modeling of volumetric soft tissue growth: application to the cardiac left ventricle," Biomechanics and modeling in mechanobiology, 8(4), pp. 301-309.
[83] Tsamis, A., Cheng, A., Nguyen, T. C., Langer, F., Miller, D. C., and Kuhl, E., 2012, "Kinematics of cardiac growth: In vivo characterization of growth tensors and strains," J Mech Behav Biomed, 8, pp. 165-177.
[84] Ambrosi, D., Ateshian, G. A., Arruda, E. M., Cowin, S. C., Dumais, J., Goriely, A., Holzapfel, G. A., Humphrey, J. D., Kemkemer, R., Kuhl, E., Olberding, J. E., Taber, L. A., and Garikipati, K., 2011, "Perspectives on biological growth and remodeling," Journal of the mechanics and physics of solids, 59(4), pp. 863-883.
[85] Menzel, A., and Kuhl, E., 2012, "Frontiers in growth and remodeling," Mech Res Commun, 42, pp. 1-14.
[86] Goktepe, S., Abilez, O. J., and Kuhl, E., 2010, "A generic approach towards finite growth with examples of athlete's heart, cardiac dilation, and cardiac wall thickening," Journal of the mechanics and physics of solids, 58(10), pp. 1661-1680.
[87] Rausch, M. K., Dam, A., Goktepe, S., Abilez, O. J., and Kuhl, E., 2011, "Computational modeling of growth: systemic and pulmonary hypertension in the heart," Biomechanics and modeling in mechanobiology, 10(6), pp. 799-811.
[88] Kerckhoffs, R. C. P., Omens, J. H., and McCulloch, A. D., 2012, "A single strain-based growth law predicts concentric and eccentric cardiac growth during pressure and volume overload," Mech Res Commun, 42, pp. 40-50.
[89] Kokubo, M., Uemura, A., Matsubara, T., and Murohara, T., 2005, "Noninvasive evaluation of the time course of change in cardiac function in spontaneously hypertensive rats by echocardiography," Hypertension Research, 28(7), pp. 601-609.
[90] Brooks, W. W., Healey, N. A., Sen, S., Conrad, C. H., and Bing, O. H., 1993, "Oxygen cost of stress development in hypertrophied and failing hearts from the spontaneously hypertensive rat," Hypertension, 21(1), pp. 56-64.
[91] Yang, C. M., Kandaswamy, V., Young, D., and Sen, S., 1997, "Changes in collagen phenotypes during progression and regression of cardiac hypertrophy," Cardiovascular research, 36(2), pp. 236-245.

[92] Bell, D., Kelso, E. J., Argent, C. C., Lee, G. R., Allen, A. R., and McDermott, B. J., 2004, "Temporal characteristics of cardiomyocyte hypertrophy in the spontaneously hypertensive rat," Cardiovasc Pathol, 13(2), pp. 71-78.
[93] Brooks, W. W., Shen, S. S., Conrad, C. H., Goldstein, R. H., and Bing, O. H., 2010, "Transition from compensated hypertrophy to systolic heart failure in the spontaneously hypertensive rat: Structure, function, and transcript analysis," Genomics, 95(2), pp. 84-92.
[94] Veress, A. I., Phatak, N., Weiss, J.A., 2005, "Deformable image registration with Hyperelastic Warping " Handbook of biomedical image analysis, J. S. Suri, Wilson, D., Laxminarayan, S., ed., Marcel Dekker Inc., New York.
[95] Cherry, S. R., Shao, Y., Silverman, R. W., Meadors, K., Siegel, S., Chatziioannou, A., Young, J. W., Jones, W. F., Moyers, J. C., Newport, D., Boutefnouchet, A., Farquhar, T. H., Andreaco, M., Paulus, M. J., Binkley, D. M., Nutt, R., and Phelps, M. E., 1997, "MicroPET: A high resolution PET scanner for imaging small animals," Ieee T Nucl Sci, 44(3), pp. 1161-1166.
[96] "Xeleris," MedX-Inc, Arlington Heights, IL.
[97] Veress, A. I., Gullberg, G. T., and Weiss, J. A., 2005, "Measurement of strain in the left ventricle during diastole with cine-MRI and deformable image registration," Journal of biomechanical engineering, 127(7), pp. 1195-1207.
[98] Maker, B. N., Ferencz, R.M., Hallquist, J.O. , 1990, "NIKE3D: A Nonlinear, Implicit, Three-dimensional Finite Element Code for Solid and Structural Mechanics," Lawrence Livermore National Laboratory Technical Report, UCRL-MA #105268.
[99] Kerckhoffs, R. C., McCulloch, A. D., Omens, J. H., and Mulligan, L. J., 2009, "Effects of biventricular pacing and scar size in a computational model of the failing heart with left bundle branch block," Medical image analysis, 13(2), pp. 362-369.
[100] "TrueGrid," XYZ Scientific Applications, Inc, Livermore, CA, pp. A Mesh Generator and Pre-Processor for FEA and CFD Analysis.
[101] Coleman, B., 2011, "Deformable image registration between cardiac PET images encompassing a range of physical heart sizes," ASME Summer Bioengineering ConferenceFarmington, Pennsylvania.
[102] Weiss, J. A., Maker, B. N., and Govindjee, S., 1996, "Finite element implementation of incompressible, transversely isotropic hyperelasticity," Computer Methods in Applied Mechanics and Engineering, 135, pp. 107-128.
[103] Hautemann, D. J., 2007, "Fiber architecture of the post-mortem rat heart obtained with Diffusion Tensor Imaging," Eindhoven University of Technology, Eindhoven, Netherlands.
[104] Veress, A. I., Segars, W. P., Weiss, J. A., Tsui, B. M., and Gullberg, G. T., 2006, "Normal and pathological NCAT image and phantom data based on physiologically realistic left ventricle finite-element models," IEEE transactions on medical imaging, 25(12), pp. 1604-1616.
[105] Fitzgibbons, T. P., Meyer, T. E., and Aurigemma, G. P., 2009, "Mortality in diastolic heart failure: an update," Cardiology in review, 17(2), pp. 51-55.
[106] Kostis, J. B., 2003, "From hypertension to heart failure: update on the management of systolic and diastolic dysfunction," American journal of hypertension, 16(9 Pt 2), pp. 18S-22S.
[107] Paulus, W. J., Flachskampf, F. A., Smiseth, O. A., and Fraser, A. G., 2007, "How to diagnose diastolic heart failure: a consensus statement on the diagnosis of heart failure

with normal left ventricular ejection fraction by the Heart Failure and Echocardiography Associations of the European Society of Cardiology: reply," Eur Heart J, 28(21), pp. 2686-2687.
[108] Wang, J., and Nagueh, S. F., 2009, "Current perspectives on cardiac function in patients with diastolic heart failure," Circulation, 119(8), pp. 1146-1157.
[109] Midol-Monnet, M., Heimburger, M., Davy, M., Beslot, F., and Cohen, Y., 1987, "[Autonomic cardiovascular regulation and permanent catheterization in normotensive conscious (WKY) and spontaneously hypertensive (SHR) rats]," Comptes rendus des seances de la Societe de biologie et de ses filiales, 181(5), pp. 560-566.
[110] Bertrand, M. E., 2011, "What are the current risks of cardiac catheterization?," Methodist DeBakey cardiovascular journal, 7(1), pp. 35-39.
[111] Conti, C. R., and Ross, R. S., 1969, "The risks of cardiac catheterization," American heart journal, 78(3), pp. 289-291.
[112] Park, P., Garton, H. J. L., Kocan, M. J., and Thompson, B. G., 2004, "Risk of infection with prolonged ventricular catheterization," Neurosurgery, 55(3), pp. 594-599.
[113] Valentin, A., and Holzapfel, G. A., 2012, "Constrained Mixture Models as Tools for Testing Competing Hypotheses in Arterial Biomechanics: A Brief Survey," Mech Res Commun, 42, pp. 126-133.
[114] Humphrey, J. D., Strumpf, R. K., and Yin, F. C., 1990, "Determination of a constitutive relation for passive myocardium: II. Parameter estimation," Journal of biomechanical engineering, 112(3), pp. 340-346.
[115] Guccione, J. M., Waldman, L. K., and McCulloch, A. D., 1993, "Mechanics of active contraction in cardiac muscle: Part II--Cylindrical models of the systolic left ventricle," Journal of biomechanical engineering, 115(1), pp. 82-90.
[116] Steve Maas, D. R., Jeffrey Weiss, Gerard Ateshian, 2012, "PostView,"Musculoskeletal Research Laboratories, University of Utah.
[117] Mahrholdt, H., Zhydkov, A., Hager, S., Meinhardt, G., Vogelsberg, H., Wagner, A., and Sechtem, U., 2005, "Left ventricular wall motion abnormalities as well as reduced wall thickness can cause false positive results of routine SPECT perfusion imaging for detection of myocardial infarction," Eur Heart J, 26(20), pp. 2127-2135.
[118] Heckbert, S. R., Post, W., Pearson, G. D., Arnett, D. K., Gomes, A. S., Jerosch-Herold, M., Hundley, W. G., Lima, J. A., and Bluemke, D. A., 2006, "Traditional cardiovascular risk factors in relation to left ventricular mass, volume, and systolic function by cardiac magnetic resonance imaging: the Multiethnic Study of Atherosclerosis," J Am Coll Cardiol, 48(11), pp. 2285-2292.
[119] Post, W. S., Larson, M. G., Myers, R. H., Galderisi, M., and Levy, D., 1997, "Heritability of left ventricular mass: the Framingham Heart Study," Hypertension, 30(5), pp. 1025-1028.
[120] Vasan, R. S., Glazer, N. L., Felix, J. F., Lieb, W., Wild, P. S., Felix, S. B., Watzinger, N., Larson, M. G., Smith, N. L., Dehghan, A., Grosshennig, A., Schillert, A., Teumer, A., Schmidt, R., Kathiresan, S., Lumley, T., Aulchenko, Y. S., Konig, I. R., Zeller, T., Homuth, G., Struchalin, M., Aragam, J., Bis, J. C., Rivadeneira, F., Erdmann, J., Schnabel, R. B., Dorr, M., Zweiker, R., Lind, L., Rodeheffer, R. J., Greiser, K. H., Levy, D., Haritunians, T., Deckers, J. W., Stritzke, J., Lackner, K. J., Volker, U., Ingelsson, E., Kullo, I., Haerting, J., O'Donnell, C. J., Heckbert, S. R., Stricker, B. H., Ziegler, A., Reffelmann, T., Redfield, M. M., Werdan, K., Mitchell, G. F., Rice, K., Arnett, D. K., Hofman, A., Gottdiener, J. S., Uitterlinden, A. G., Meitinger, T., Blettner, M.,

Friedrich, N., Wang, T. J., Psaty, B. M., van Duijn, C. M., Wichmann, H. E., Munzel, T. F., Kroemer, H. K., Benjamin, E. J., Rotter, J. I., Witteman, J. C., Schunkert, H., Schmidt, H., Volzke, H., and Blankenberg, S., 2009, "Genetic variants associated with cardiac structure and function: a meta-analysis and replication of genome-wide association data," JAMA : the journal of the American Medical Association, 302(2), pp. 168-178.