Contributions to Biomechanical Finite Element Analysis for 3D Medical Images with Open-Source Software

by

Adrienne Monique Madison

(Under the direction of Mark A. Haidekker)

Abstract

Finite-element modeling (FEM), well-established to predict the mechanical behavior of mechanical systems, enjoys growing popularity in the study of the biomechanical behavior of biological tissues. This technique uses numerical methods to simulate material behavior under defined spatial constraints and load conditions. For this purpose, a system is decomposed into a large number of interacting volume elements, which can be described by a set of partial differential equations. Numerical methods are then employed to solve the equation system for the unknown quantities, such as deformation and stress. Contrary to mechanical systems, which can often be described analytically or with geometric primitives, biomedical objects require discretization of their often irregular shape. In noninvasive studies, imaging methods are used to obtain the geometry. Image-based finite-element models have a highly complex geometry, and the assignment of material properties to individual elements from image information is difficult. In this work, the four steps to obtain an image-based finite-element-based material simulation (segmentation, meshing, simulation, and post-processing) are described in detail, and strategies to overcome the specific challenges of image-based finite-element models are discussed. A completely open-source FEM

toolchain was established which includes a custom meshing module. In comparison to results obtained from commercial systems, the open-source feature gives users the ability to modify and extend the code, and thus offers additional flexibility over commercial systems. A fully open-source toolchain is feasible, but the critical element is the meshing module. Lastly, a novel approach which implements FEM and medical imaging to approximate intracranial pressure using open-source software in settings where conventional monitoring techniques are unavailable is proposed. Emphasizing on the relationship between the cerebral perfusion pressure (CPP) and ICP, patterns of non-linear biomechanical behavior in biphasic analysis of normal and abnormal canine brains are observed and identified. The method presents a framework which can use material response to increased ICP as a diagnostic, treatment, or preventative method to assess levels of brain injury in clinical veterinary settings noninvasively while simultaneously introducing a free open-source software toolchain that can be used in any biomedical application, including analysis of bone, tissue and implants.

INDEX WORDS: Finite Element Analysis, FEA, Finite Element Modeling, FEM, Open-Source Software, Medical Image Segmentation, Mesh Generation, Tissue Biomechanics, Vascular Grafts, Intracranial Pressure, Brain Parenchyma, Cerebrospinal Fluid, CSF, Noninvasive, Biphasic, Non-Linear

Contributions to Biomechanical Finite Element Analysis for 3D Medical Images with Open-Source Software

by

Adrienne Monique Madison

B.S., University of Tennessee-Knoxville, 2006

A Dissertation Submitted to the Graduate Faculty
of The University of Georgia in Partial Fulfillment
of the
Requirements for the Degree

Doctor of Philosophy

Athens, Georgia

2013

Contributions to Biomechanical Finite Element Analysis for 3D Medical Images with Open-Source Software

by

Adrienne Monique Madison

Approved: August 2013

Major Professor: Mark A. Haidekker

Committee: Steven D. Holladay
William S. Kisaalita
Ajay Sharma

Electronic Version Approved:

Maureen Grasso
Dean of the Graduate School
The University of Georgia
August 2013

Contributions to Biomechanical Finite Element Analysis for 3D Medical Images with Open-Source Software

Adrienne Monique Madison

August 1, 2013

Dedication

"The difference between dreams and success is, dreams need effortless sleep and success needs sleepless efforts." –Akon

This work is dedicated to the memory of my maternal grandparents, Aaron and Cliffie Cotton and paternal grandfather Harry Madison, who were instrumental in fostering and engaging my early educational and social development. Watching helplessly as they battled complications from diabetes, cancer, and congenital heart disease ignited my interests in medical devices, prosthetics, and ultimately biomedical engineering.

To my parents, Larry and Vickie Madison, my sister Candace, the Cotton and Madison families, and hosts of friends both near and far, this dissertation is a result of your undying support and encouragement throughout the pursuit of this degree. Words cannot adequately express my feelings of gratitude. Because of your presence, I was successfully able conquer my fears, maintain a positive attitude, and continue to persevere despite the obstacles and setbacks encountered.

Mrs. Helen Fagan, since the tender age of two you have gone above and beyond the traditional teacher role to nurture and mold me into the student I am today. Throughout each academic endeavor, you have celebrated my successes and achievements.

Lastly, this research is dedicated to those who dream big. The innovations of tomorrow begin with the ideas of today.

Acknowledgments

First and foremost, I would like to acknowledge my Lord and Savior Jesus Christ for blessing me with the academic talents necessary to complete this doctoral research.

To my major professor, Dr. Mark A. Haidekker, thank you for selecting me based on my passion for biomedical research. Your continued confidence, patience, motivation, and assistance over the years have significantly and directly contributed to my evolution as a both a person and biomedical engineer. To Dr. William S. Kisaalita, you were instrumental in my transition to UGA and have always been there for me in whatever capacity required. Because of the tireless dedication and commitment to teaching and mentoring displayed by you both, I was privy to two outstanding templates in the construction, development, embracing, and enhancing of my own instructional desires and abilities.

I am additionally grateful to Drs. Steven D. Holladay and Ajay Sharma in the UGA School of Veterinary Medicine, for agreeing to serve as members of my doctoral research committee. Your research contributions and partnerships helped me to accomplish my dream of leading a cross-functional biomedical team responsible for the research, development, design, and testing of medical devices and diagnostic tools using novel computational approaches. Because of your guidance, I learned to not allow time and other external constraints to diminish or hinder my research potential and objectives.

Special thanks to the University of Georgia Graduate School and (former Faculty) College of Engineering faculty and staff for the opportunity and providing the financial support that allowed me to fulfill my dream of obtaining this doctoral degree. I would not have been exposed to the Bulldog tradition and experience I am so fond of without you.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Finite Element Modeling (FEM or alternatively FEA for finite-element analysis) is a mainstream computational tool for the prediction of mechanical behavior in engineering design processes. Within the last three decades, this numerical methods-based technique has been implemented to evaluate the biophysical behavior exhibited in biomedical contexts. The incorporation of medical imaging (*e.g.* CT and MRI) data in these engineering applications results in robust three-dimensional models which capture and highlight the intricate geometrical details of complex anatomical pathologies commonly observed in organs, bone, muscle, and soft tissues. Coupled with finite element theory, researchers can create realistic simulations that provide both qualitative and quantitative patient-specific results associated with biomechanical response. Such data is vital in the development of medical devices, biomaterials, prosthetics, as well as in the improvement of diagnostic, treatment, and prevention protocols currently utilized in clinical environments. Medical imaging-based FEM is comprised of four steps and begins with biomedical imaging data sources as the input. This image data is then pre-processed such that the desired geometry is isolated (segmentation) and transformed into a model ready for finite element simulation (meshing). Next, analyt-

ical solutions characterizing material response behavior are obtained through finite element simulation and are then presented for visualization or other forms of analysis in the final step.

A variety of combination software options designed to perform mesh generation, finite element analysis, and visualization of simulation data exist, and are ideal for the analysis of classical load and stress problems in mechanical engineering. The geometric models subjected to simulation in these instances are solid model representatives constructed from primitive shapes. Such software poses a challenge for anatomical models since the geometry is extracted from medical images via segmentation; therefore, external programs are required to perform the pre-processing steps.

The major obstacle in the execution of a medical image-based FEM chain is the lack of a uniform, straightforward method capable of segmenting and meshing all types of anatomical geometry. Differences in image modality data presentation, material types or properties, geometrical complexity, and biomechanical behavior are just a few of the contributing factors which have fueled the development of biomedical application-specific computational methods for pre-processing. As a result of these complications, a strong community-driven effort to develop and make available free, open-source software (FOSS) has gained increasing popularity from which interested researchers can immensely benefit. Since its emergence in the late 1970s and early 1980s, the platform based on free (unlimited) exchange, enhancement, and modification of programming code promotes a cyclic learning and development environment from which high-value software options for the segmentation, meshing, simulation, and visualization of biomedical models have emerged. Therefore, the compilation of software to perform FEA in biomedical applications can be achieved with minimal or no charge to the user. The absence of license fees also permits researchers to download and install software without risk in order to select the programs best-suited to the intended application.

Biomedical research groups are interdisciplinary and composed of medical professionals, scientists, engineers, computer specialists, and statisticians. A general familiarity and understanding of each stage in the medical imaging FEM process should be possessed by all members within these cross-functional teams. Unfortunately, the non-existence of literature which highlights all aspects of the use of biomedical images with the finite element theory is also a hindrance to the utilization and success of this modeling process. The work presented in this dissertation presents a unique contribution to the research progression of medical imaging used in conjunction with the finite element theory. Chapter 2 provides a comprehensive literature compilation of the theoretical background, the governing principles, methods of implementation, software options, and future research trends which assist in completion the pre-processing, analysis, and visualization components. Using a non-traditional approach, the open-source software concept is introduced and this chapter places emphasis on the key fundamental choice of whether to utilize often inflexible, out-of-box-software software tools or a collection of disjunct packages that can be linked together. Additionally, if the latter is the preferred option, general information is presented to assist in the compilation of software options to efficiently construct toolchain to perform medical image-based FEA.

In Chapter 3, a medical image-based FEA pipeline consisting entirely of disjunct open-source software is presented for the observation and prediction of mechanical behavior in tissue-engineered blood vessels and tubular constructs. The pre-processing, meshing and segmentation steps are completed using a customized module that can be used in conjunction with medical imaging data. While this is only a phantom study to verify the feasibility of the proposed toolchain software options and highlight the overall flexibility of the open-source philosophy, the results additionally demonstrate the usefulness of FEA in the development of biomaterials.

The fourth chapter is a progression of the phantom study in the previous chapter. Here, actual patient-specific clinical data is utilized as input for a proposed open-source software toolchain. While the construction of an adequate software pipeline is the"pulse" of the research objective, the purpose of this work is to examine the potential for medical imaging based FEA to be employed as a noninvasive diagnostic and treatment apparatus in a veterinary clinical setting. Changes and variability in biomechanical properties during instances of increased intracranial pressure and decreased cerebral perfusion in canine patients are the focus of this study. Additionally, the relationships between the biomechanical properties and pressure and volume are examined. The final chapter discusses the achievements, advantages, limitations, next steps, and long-term possibilities related to the overall research based on each chapter's contribution.

Chapter 2

MEDICAL IMAGING IN FINITE ELEMENT ANALYSIS: OVERVIEW AND APPLICATIONS[1]

[1]Madison, A.M. and M.A. Haidekker. 2013. Submitted to Guigen Zhang (Editor), *Computational Bioengineering.* CRC/Taylor & Francis, 2013.

Abstract

Finite-element analysis is a numerical method to simulate material behavior under defined spatial constraints and load conditions. For this purpose, a system is decomposed into a large number of interacting volume elements, which can be described by a set of partial differential equations. Numerical methods are then employed to solve the equation system for the unknown quantities, such as deformation and stress. Finite-element models have been employed observe the mechanical response behavior of biological tissue as well as implant materials. Unlike in mechanical systems, where a relatively simple geometrical description is available, patient-specific models of organs or implants are often obtained from volumetric images. Image-based finite-element models have a highly complex geometry, and the assignment of material properties to individual elements from image information is difficult. In this chapter, the four steps to obtain an image-based finite-element-based material simulation (segmentation, meshing, simulation, and post- processing) are described in detail, and strategies to overcome the specific challenges of image-based finite-element models are discussed. A focus of this chapter lies on available software to perform the four steps with aspects of the practical realization of a finite-element modeling chain.

2.1 Overview

Finite element analysis (known under the acronym FEA, or FEM for finite-element modeling) is a numerical modeling technique used to solve engineering problems by obtaining approximate solutions from a large system of algebraic equations. The underlying principle requires the entire geometry of interest to be subdivided into a large number of small regions or *elements* that are considered homogeneous. These small elements have a variety of possible shape patterns (e.g. triangular, rectangular, tetrahedral, hexagonal, and cubic), depending on the number of geometric dimensions present, and they share their edges and vertices with neighboring elements. The vertices (referred to as *nodes*) are rigidly connected, therefore each element has boundary conditions imposed upon it by its neighbors (internal boundary conditions) and by the environment (external boundary conditions). Internal boundary conditions are obtained from the balance of forces and from the equal displacement of a node that is common to multiple elements. External boundary conditions, such as pressure, forces, or spatial constraints, can be prescribed to the model as a whole. The actual finite-element *model* is represented by a series of partial differential equations that describe material properties, conservation of mass and energy, the laws of motion, and deformation of an element by forces acting upon its nodes. Ultimately, solutions are found for each region by the calculation of unknowns within the volume by only taking into account the regions located next to them.

Originally developed to analyze the twisting behavior of cylindrical objects in the early 1940s [2], finite element modeling has been intensely used in biomedical contexts to examine, simulate, and predict the material behavior and non-linear biomechanical properties of soft tissues [3,4], organs [5,6], bone [7,8], and joints [9,10]. It is also highly useful in applications of modeling, testing, and verification of medical device designs, such as vascular implants and stents [11–13], dental implants [14], and most recently in accident analysis and prevention

[15]. The number of reported studies which utilize finite element modeling and related computational numerical methods in the advancement of biomedical and clinical research, development, diagnosis, and treatment applications has constantly increased since 1980, and currently surpasses 10,000 [16].

Classical finite-element models are based on geometric descriptions of the object under examination: A motor, for example, can be composed of cylinders, pistons, rods, the crank shaft, and other interacting parts that can be represented by their exact geometrical description. In biomedical applications, two challenges appear. First, a tissue or organ can have a fairly complex geometry that necessitates an equally complex analytical description, sometimes with approximations made to simplify the model. Second, to obtain a patient-specific description, the geometry is often extracted from volumetric images, such as computed tomography (CT) or magnetic resonance (MR) images. Finite element modeling of biomedical structures involves four main steps (Figure 2.1) and begins with a volumetric image. In the first step, the object of interest is *segmented*, i.e., separated from image elements not related to the object of interest. At this point, the object is still represented by individual voxels. The *meshing* step follows, in which the volumetric arrangement of voxels is parametrized (that is, approximated by an analytical description of the surface with straight-line or curved segments), and during which boundary conditions and material properties are applied. The third step involves the actual finite element simulation of the time-variable behavior of the created model under the established parameters. Once this step is completed, the output of the simulation process is visualized or post-processed in the final step.

The objective of this chapter is twofold: (i). to explore the basic principles, theories, and techniques behind the execution of each component within this modeling chain, and (ii). to introduce the reader to software options along with trends and developments in which these methods and approaches could be better used in biomedical applications.

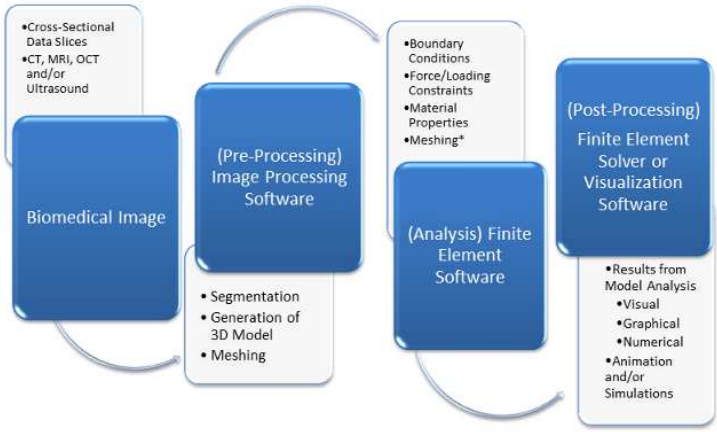

Figure 2.1: *Flow process diagram for finite-element modeling of anatomical geometry extracted from medical imaging data. With a volumetric image as starting point, the first crucial step is the extraction of the geometry of the object of interest. For this purpose, a segmentation step is followed by the meshing step, in which an analytical description of the geometry is obtained. For this geometry, the constituent mechanical partial differential equations are then solved (i.e., the actual modeling step). Lastly, the modeling results are visualized or further examined with respect to the property of interest, such as deformation, stress, or failure.*

2.2 Segmentation

The pre-processing phase, which consists of the segmentation and meshing steps, is the most fundamental and challenging component within the biomedical modeling chain. The segmentation process divides an image into meaningful regions in an attempt to delineate and extract objects or regions of interest (these are often referred to as *features* in the image). No uniform solution exists for the segmentation problem. The segmentation strategy strongly depends on the imaging modality, the object itself, and the varying image interpretations amongst different modalities. For example, CT allows for a relatively easy intensity-based segmentation of bone or the lungs due to the excellent bone-tissue and air-tissue contrast. Conversely, MRI can provide excellent contrast between tissues, but is not usually used for

imaging bone due to the low water content [17]. Frequently, image voxels are classified according to visual characteristics (e.g. intensity, texture, or color). Prerequisite is a form of contrast between healthy and abnormal tissue and between the tissue of interest and adjoining regions (the latter are often referred to as *background* in a generalized sense).

A second goal of the segmentation process is the assignment of tissue properties, most often in inhomogeneous tissue regions. An assumed relationship between image intensities and material types is frequently the guiding standard in medical imaging segmentation application [6,18–22]. For example, the CT number of a bone voxel is related to its mineral content, and it can be argued that mineral content and stiffness are related [23,24]. Therefore, some empirical material properties are assigned to the voxel based on its CT number.

The segmentation step can be performed manually, with limited manual assistance, or in an automated fashion. In a manual segmentation step, a trained observer (such as the radiologist) delineates the boundary of the object. This process is often performed in 2D on a slice-by-slice basis. Manual segmentation suffers from variability due to subjective influences. Computer-aided segmentation or fully automated segmentation are desirable to reduce intra- or inter-observer variability [25,26] and to accelerate the segmentation process. A brief overview of some commonly-used segmentation approaches is given in the following section. For a more comprehensive overview or for the underlying mathematical theories, the reader is referred to the pertinent literature [25,27–30].

2.2.1 First Generation Processes

First generation processes solely rely on information available within an individual image (i.e., its voxel values) and therefore are purely image based. Manual segmentation largely relies on the same contrast properties that enables these first generation methods.

Automatic-multi step segmentation approaches (discussed later) usually incorporate a combination of these first-generation methods or use them as a foundation for higher-level steps. First-generation processes can be partitioned into categories:

Intensity thresholding (Figure 2.2B)

- Intensity-based segmentation method that uses contrasting intensity levels to identify distinctive regions.
- Each image pixel is compared to threshold value such that all values:
  - Above this range are labeled as *feature*
  - Equal to or below the indicated value are classified as *background*
- The ideal image case would contain pixels with nonoverlapping intensity values between feature and background
- Often, a suitable transformation (for example, local texture filters) can provide additional contrast
- Multidimensional thresholding is an alternative option when a voxel is described by multiple criteria, such as intensity and one or more local neighborhood properties (e.g., smoothness or proximity to a gradient). When a voxel is associated with multiple orthogonal criteria, these are referred to as *feature vector*.

Region-based methods (Figure 2.2C)

- Extension of the intensity-based thresholding method under the assumption that the object forms a contiguous region
- Assumption of connectivity allows the separation of image features even if those have similar intensity

Edge/boundary based (Figure 2.3)

- Relies on the identification of intensity gradients or sharp transitions of intensity levels
- Anticipated result is that all of the pixels located near the gradient boundaries have higher intensity values of than those lying outside or inside of the object
- The resulting image highlights the object surface rather than the volume; intensity-based thresholding is possible (Figure 2.3 C)

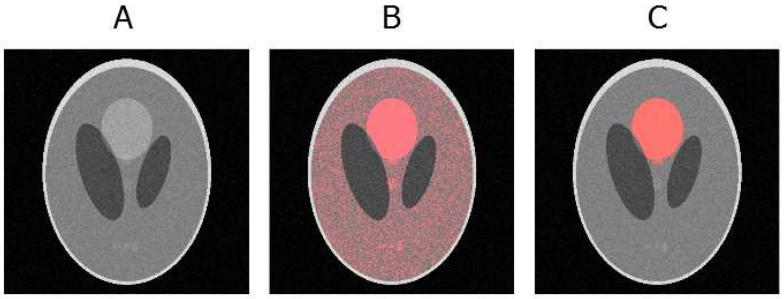

Figure 2.2: *Intensity-based segmentation in a noisy variant of the Shepp-Logan head phantom. The additive noise causes major overlap of the image values between the "gray matter" and the large central "tumor" (A). Pure intensity-based thresholding (red pixels) cannot separate the tumor region from the gray matter region (B). When connectivity is considered, for example, with the region-growing algorithm, the segmented pixels are constrained to a connected region, and unconnected pixels within the thresholded intensity range are excluded (C).*

2.2.2 Second Generation Processes

Second-generation processes rely on the same intensity or contrast information that first-generation processes use, but attempt to describe the image feature at a higher level of abstraction. Often, some form of numerical optimization or minimization is involved (purely discrete algorithms), or a physical model is approximated by numerical methods. Second-

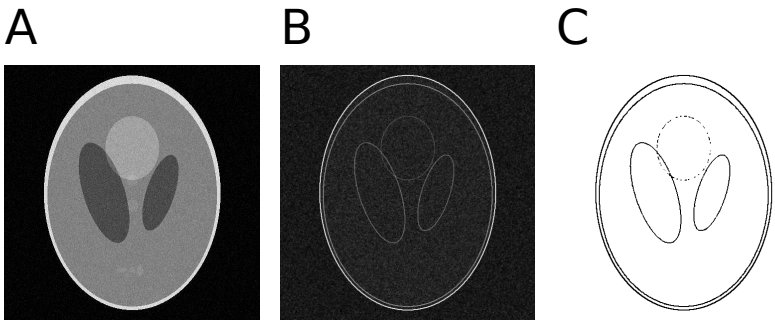

Figure 2.3: *Edge-based segmentation. The source image is the head phantom with additive noise used in Figure 2.2 (A). The application of an edge enhancement filter (Sobel operator) converts gradients into higher-intensity pixels (B). As a side effect, the noise component is amplified. After thresholding and removal of isolated noise pixels, the edges remain; however, low-contrast features do not necessarily have a closed boundary, and the features with the lowest contrast have disappeared altogether (C).*

generation methods are purely image-based derivations of first generation methods that occasionally use information gathered directly from initial first-generation segmentation results. The need for second-generation methods can be illustrated with the edge-based segmentation in Figure 2.3C, where it is desirable to obtain a closed contour for the large "tumor" in the same fashion as for the skull and the ventricles. Second-generation processes can be subdivided as follows:

Continuous Model Discretizations: Based on a physical model, such as elastic contraction or viscous flow, but with an external, image-based driving force that lets the model converge on an image feature.

- **Snakes** – Snakes are models of elastic rubber bands which are subject to a stretching and a bending force. The external force is the negative image gradient. In the snake concept, a closed path contracts until an equilibrium is reached between

the curvature-based internal force responsible for the contour's smoothness and the external force of the intensity gradient [31]. Figure 2.4 highlights the iterative progression of this process.

- **Level Set Active Contours** – Level set active contours are implicit methods (contrasted to the explicit numerical formulations underlying the snakes). Level set based active contours are derived from a numerical method designed to track an evolving contour deforming at a rate of speed based on curvature and gradient [32]. The model ceases deformation once the speed reaches an artificial preset speed assigned to desired boundary areas. The main benefit over snakes are their ability to change the topology [33]
- **Live Wire** – The live wire formulation is a semi-assisted method where a cost function is established from representative boundary points identified by the user. The live wire algorithm then connects those points along a minimum-cost path [34]

Purely Discrete Algorithms:

- **Clustering** – Automatic assignment of pixels to one of several classes (e.g., feature and background) based on minimum-distance criteria for a vector of descriptive metrics
  - k-means – A pixel belongs to exactly one class, with which it has the smallest distance to the class centroid [35]
  - Fuzzy c-means – a pixel initially belongs to a fuzzy extent to multiple classes, and final class membership is decided after the optimization process [36]
- **Graph-Based** (see below)
  - Watershed segmentation
  - Fuzzy Connected

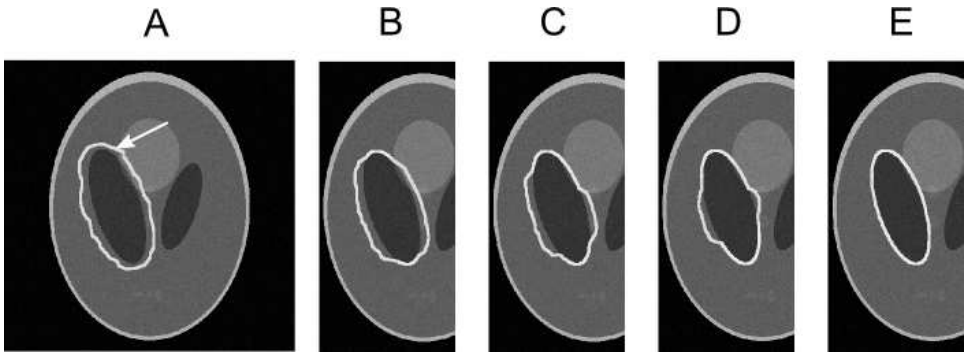

Figure 2.4: *Semi-supervised segmentation with active contours (snakes). A hand-drawn region (arrow) near one of the "ventricles" of a noisy Shepp-Logan head phantom serves as the starting point for the snake (A). The snake algorithm causes the snake to contract, but vertices are attracted by edges. Beginning contraction after five iterations (B); Some convergence can be seen after 10 iterations (C); Most of the snake has locked onto the edges after 20 iterations (D); Final convergence of the snake (E). The vertices of the snake coincide with the edge of the ventricle, and the curve defined by the vertices serves as parametrization of the contour.*

In graph-based methods [29, 37], the nodes of the graph correspond to the voxels. The graph itself is the set of all paths that connect a voxel inside the feature with a background voxel. Some criteria can be found where a path traverses a discontinuity (such as an edge) that defines the object boundary. At this point, the graph is cut. The remaining connected regions show a higher voxel homogeneity with respect to the segmentation criteria than the uncut graph. It is possible to express the links of the graph as a cost function similar to the one that drives the Live Wire, and training of the cost function is performed with some user-selected seed points.

The watershed and fuzzy-connectedness algorithms are based on the graph-cut principle. Watershed segmentation demarcates the boundaries of regions on a topological surface using "watershed" lines [38]. The topological surface is built on the interpretation of pixel intensity as elevation. The object of interest is assumed to be the set of pixels that becomes flooded, and the low-elevation regions that become flooded first serve to cut disjunct features. Thus,

watershed segmentation is used to separate features that are connected in a first-generation segmented image. Fuzzy connectedness [39] assigns a continuous voxel similarity to pairs of voxels, and the graph (in the sense introduced above) is cut at a certain level of dissimilarity.

Statistical pattern recognition and neural networks are related image-based segmentation methods that are often collaborative [40]. In statistical pattern recognition, a model to is used to assign image pixels to a known set of classes. Neural networks can be trained to segment images based on pixel values manually designated to classes. Neural networks in particular rely on high-order feature vectors that include the voxel intensity and a number of neighborhood criteria that can be generated through texture description methods (e.g., Laws' texture energy [41]).

2.2.3 Third Generation Processes

Third generation methods of geometry extraction are higher-level model-based recognition procedures that require extensive *a priori* information. A simplified explanation to third-generation processes can be given when we consider Figure 2.3. The edge contours are known to be elliptical (this is the *a priori* knowledge). The image can now be searched for voxels that belong to an ellipse, for example, with the Hough transform [42]. In this fashion, the fragments of the central ellipse in Figure 2.3 C would be recognized as belonging to one ellipse, and the segmentation would yield the completed ellipse rather than the fragments.

More generally, third-generation processes are based on the construction of a spatial statistical model for the object of interest that consists of prominent shape details and any potential variations from a set of training images. During analysis, image data is scanned to locate regions that resemble the model. *Active shape models* (ASMs) identify objects within an image set that are recognized as members of the same class by using landmark points [43]. Image patches, such as object intensity or texture are incorporated to add region-based features in *active appearance models* (AAMs) [44]. In *atlas-based segmentation*, a composite

image constructed from segmented images of many subjects. Descriptive information besides the geometry and shape, such as intensity properties, object labeling, and relationship definitions is also stored with the atlas. An image set is first matched to the atlas template via 3D mapping, followed by the atlas using the accompanying descriptive properties for statistical pattern recognition.

Hybrid segmentation methods merge the the complementary strengths of separate methods with an aim to create more complex methods that increase accuracy and precision while simultaneously overcoming some of the individual limitations. Many of the recent advances in segmentation have been contributed to the varying combinations of the aforementioned techniques. Some comprehensive surveys and reviews are based on method type [45], medical application [46,47], image type and method [48], medical application and image type [49,50], or more specifically by medical application, image, type, and method [51].

Almost all of the above segmentation methods are supervised, and some level of user interaction is required throughout the process. Hybrid methods incorporating unsupervised processes can significantly reduce the level of operator assistance. Rule-based systems attempt to achieve effective, unsupervised segmentations. In this multi-step process, first and second generation methods extract desired image features, which are interpreted and labeled by third generation knowledge-based approaches. The segmentation is carried out according to an explicit pre-defined set of rules [52]. Neural networks and fuzzy-connected filters are commonly implemented in rule-based systems and have been used in brain tumor extraction from MR images [53–55], breast cancer lumps [56], and skin cancer lesions [57].

At the end of the segmentation process, the image has been subdivided into the background and one or more features (objects) that will be analyzed in the finite-element simulation. At this point, the voxels are merely labeled as belonging to the background or to the

image feature (or features if multiple materials are present). In the next step, the shape of the objects formed by those voxels needs to be described analytically such that the proper set of governing equations can be set up.

2.3 Meshing

In conventional engineering FEM applications, the meshing step is relatively straightforward, because an analytical description of the object usually exists. Notably, solid modeling software often allows to automatically subdivide a 3D model into nodes and elements, and to assign material properties to the elements. The resulting finite-element shapes in these 3D models can be extensions of either a quadrilateral in the form of a *hexahedron* (8 vertices, 12 edges, 6 faces) or a triangle evolved into a *tetrahedron* (4 vertices, 6 edges, 4 faces). The mesh is of fundamental importance, because it directly gives rise to the set of partial differential equations that are used in the FEM solver (i.e., the simulation step) to determine the unknown quantities of the simulation. The process of discretizing a known geometrical shape can be seen as a *top-down approach* to generating a finite-element model.

Conversely, the irregular shape of biomedical objects – especially when obtained from volumetric images – requires a piece-wise analytical approximation of the shape. Usually, a large number of points on the surface is determined, and these points are connected with high-order curves, such as splines. The process of approximating a shape with discrete nodes can be seen as a *bottom-up approach*. The element shape, mesh arrangement, and algorithm selections to generate the most accurate mesh for evaluating biomechanical behavior are widely dependent on the analysis emphasis, material type, complexity of geometry, method of finite-element analysis, and other application characteristics. Adjustment or modification of one application parameter could require the use of an entirely different meshing approach.

Moreover, in biomedical applications, the material properties are often unknown or uncertain, and a rigorous match between elements and their associated material properties cannot be obtained in a straightforward manner. For these reasons, *the overall meshing concept based on patient-specific medical imaging data remains the most crucial bottleneck* in the entire finite-element analysis chain. However, a large body of literature has emerged in recent years where improvements and modifications are presented for tailored and application-specific mesh generation, and this section provides an overview of some of the most relevant approaches.

Image-based mesh generation begins with the segmented object, which is the outcome of any of the processes described in the previous section. To provide the necessary input for finite-element analysis, three steps are taken:

- Surface mesh construction: The shape of the object is discretized by a large number of nodes that are placed as close as possible to the surface of the segmented object. Ideally, these nodes are either evenly spaced or become denser in regions of higher curvature.

- Volume mesh construction: The areas between the surface boundaries can be meshed in a manner similar to the analytical top down approach, that is, the object's volume is subdivided into a large number of small hexahedral or tetrahedral elements. Volume meshing usually begins at the nodes of the meshed surface. Similar to conventional mesh generation of CAD based models, the elements are rigidly connected at the nodes and provide the boundary conditions for neighboring elements.

- Application of material properties and external boundary conditions: Material properties must be defined for all volume elements. External forces or spatial constraints can be applied to some of the nodes.

2.3.1 Surface Mesh Construction

A three-dimensional image volume is the yield, or end product, of the segmentation process. Each voxel within this volumetric scalar field is assigned a value to represent pixels that are a part of either the background or any of one or more segmented objects (including their associated material types) within the identified geometry. Two general mechanisms have been implemented to fit surfaces to data [58]: surface extraction and adaptive contours. Not coincidentally are these related to the first- and second-order processes in the segmentation step.

Isosurface extraction uses piecewise linear interpolation algorithms such as the marching cubes algorithm [59] or level sets, to subdivide (tesselate) the surface into triangular surface facets. For the isosurface extraction, the volume of interest is embedded into a regular, cuboid mesh. The assumption is that the object has higher voxel values than the background. Each node of the mesh is now checked whether it is above the isosurface value (i.e., lies within the object) or below the isosurface value (i.e., is part of the background). Edges that connect nodes inside and outside the object intersect the surface, and the intersection point is determined by linear interpolation. The intersection points then form the nodes of the surface mesh. Edges are shared by multiple cubes, and consequently, surface facets are connected. Whenever the intersection of any cuboid with the isosurface renders quadrilaterals or higher-order polygons, these are further subdivided into triangles, because triangles are always planar, and thus have a defined surface normal. Figure 2.5 demonstrates how the algorithm constructs triangular elements.

The alternative approach to surface reconstruction of a segmented volume is a 3D adaptation of the local energy-minimizing snake algorithm [60]. A net wraps around, and conforms to, a specific volumetric feature based on internal energy and image force calculation between the binary surface and the parametric surface in an iterative manner. The points on the surface and other parameters, for example, the number of nodes must be pre-assigned

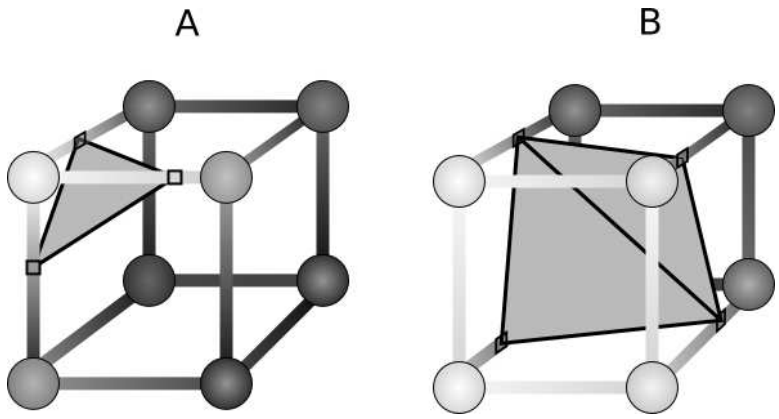

Figure 2.5: *Two examples of an isosurface intersecting a cube. In both cases, the image value of the voxel centers (spheres) is known, and the image values between voxel centers are obtained by linear interpolation (thick lines with gradients). Interpolation provides the location of the desired isosurface value between two adjoining pixels, and this location serves as one vertex of the tesselated surface (small squares). (A) Only three edges cross the isosurface value, and all other edges lie below the isosurface value. The resulting element is a triangle. (B) The front four voxels have values above the isosurface value, the four back pixels lie below . Consequently, the vertices lie on the diagonal edges and form a quadrilateral. Since tesselation requires planar (triangular) elements, the quadrilateral is subdivided once.*

to determine the net's initial geometry. The definition of such parameters contribute to the model's smoothness. A smoothed approximation of the surface by the model nodes is the output from this process, and nodes are connected to form triangles.

2.3.2 Volume Mesh Construction

The surface mesh can now be transformed into a FEM *volume mesh* by filling the interior region with the preferred element shape and arrangement. Mesh types can be broadly classified into *structured* or *unstructured* meshes according to their geometrical pattern. *Structured* mesh arrangements consist of an evenly-spaced, fixed, uniform amount of nodes and elements and are comprised entirely of hexahedral elements that are a result from grid-based surface construction algorithms. This template-based connectivity pattern was the initial attempt

at meshing automation, and it produces meshes that approximate curved surfaces in a stair-step fashion. The disadvantage of a structured mesh is the inability to adapt to the shape curvature: the node density is the same in regular regions, where fewer nodes would yield an adequate discretization, and in irregular regions, where the node density may not be sufficient to describe the surface with adequate detail.

Unstructured meshes contain either all tetrahedral elements or a combination of hexahedral, tetrahedral, and wedge-shaped elements generated automatically in arbitrary configurations. The relationship between the number of vertices, edges, faces, and regions is unknown beforehand. Unstructured meshes are better capable of adhering to curved boundaries while being adaptable to general complex shapes with less user intervention. Figure 2.6 demonstrates the difference between structured and unstructured meshes. The following algorithms have been established for the generation of unstructured mesh types:

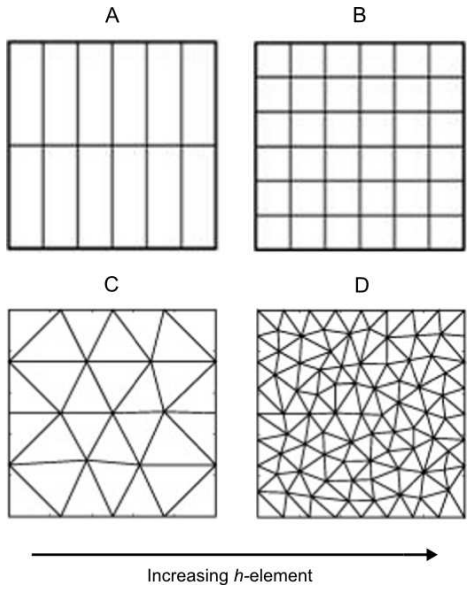

Figure 2.6: *Comparison of mesh types. Structured mesh with quadrilateral elements (corresponding to hexahedral elements in 3D (A-B), unstructured mesh with triangular elements (corresponding to tetrahedral elements in 3D) (C-D). The effects of increasing the h-element constraint can also be observed. The finer mesh patterns in (B)& (D) with smaller element size and increased element quantities are derived from the original patterns in (A)& (C) by subdivision of the original elements.*

Tetrahedral-based mesh

- **Octree** [61, 62]: Recursively partitions a cube containing the geometry into eight octants until a preferred resolution is reached. Non-uniform tetrahedral elements are formed between the intersections of these cubes.
- **Advancing Front** [63, 64]: Begins at a boundary then progresses towards empty space within a region
- **Delaunay** [65]: Creates nearly equilateral trangles (which are the faces of tetrahedrons) based on satisfaction of an "empty circle" criterion. For this method, each edge of a triangle lies along the edge of a circle, and vertices are prevented from being located within the circle's circumference. This approach minimizes the occurrence of "skinny" triangles that have very small angles or form sharp, jagged edges (see Figure 2.7)

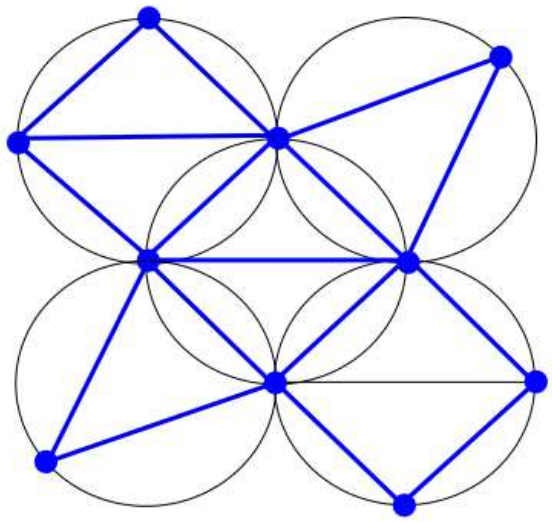

Figure 2.7: *Triangular/Tetrahedral mesh construction based on Delaunay algorithm. All points or vertices of each triangle lies along the edge of a circle, because the "empty circle" criterion prevents vertices from being placed within the circle or sphere's circumference.*

Hexahedral-based mesh

- **Plastering** [66] is a hexahedral adaptation of "advancing front" mechanism. It begins with an all-quadrilateral mesh that projects faces into a volume, thereby creating hexahedral elements in an iterative manner
- **Medial Surface** [67]: Decomposes a volume into hexahedral elements
- **Grid-Based** [68]: Imposes a 3D grid of hexahedral elements within volume interiors
- **Whisker-Weaving** [69]: Builds a dual or spatial twist continuum (STC) with bisecting intersecting surfaces which fit hexagonal elements derived from quads into the volume based on connectivity followed by the subsequent determination of nodal locations

2.3.3 Mesh Quality, Optimization, and Adaptive Refinement

Meshes must be feature-preserving in the sense that they must be as close as possible to the original surfaces and be capable of handling complex topology [70, 71]. Ideally, they also possess the adaptivity to increase node density in areas of high interest (e.g. areas of concentrated stress) and curvature to limit the surface discretization error, while balancing the number of mesh elements. A mesh composed of nearly equilateral triangles is desirable, and vertices need to be seamlessly aligned with neighboring vertices.

A frequently-employed strategy is to generate an initial mesh that is then refined until convergence is reached, that is, until further mesh refinement changes the simulation results by less than a pre-defined margin. The method to generate the initial mesh is referred to as *primal contouring method*, and the marching cubes algorithm is one example for a

primal contouring method. The resulting initial mesh approximates the implicit geometry, but generally has undesirable features that need to be suppressed. Common observations in meshed objects from biomedical images include [72, 73]:

- A uniform stair-step surface appearance that does not fully align with the natural surface curvature
- Badly shaped triangles (e.g. triangles with very large and small angles that cause sharp, jagged edges
- Excessive amounts of irrelevant nodal and faceted data
- The non-preservation of sharp features within the data

Methods for subsequent refinement are referred to as *dual contouring methods* [74]. Dual contouring methods were derived to improve the quality of meshes with respect to the accurate reproduction of sharp features [71]. In the basic dual contouring algorithm, the region to be meshed is divided into overlapping cubic grids. The surface is then analyzed at vertices of the grids, and classified as either being inside or outside the mesh. Cubes consisting of inside and outside combinations of vertices are denoted as containing surface portions. The dual contouring process creates a vertex pair per cube which straddles the surface, and the connection of each to neighboring vertex pair shapes the final mesh.

Additional refinement is based either on local remeshing through swapping of edges or faces, or on local refinement through insertion or deletion of vertices [75]. In fact, the methods described in the previous section for the generation of unstructured meshes can be used for mesh refinement as well. The criterion that the octree, Delaunay, and advancing front algorithms act upon are responsible for the swapping of edges/faces and insertion or

deletion of vertex points that lead to the final positions and number of the vertices of the mesh. It is at these locations at which the error is minimal between the planes and normals at the edge intersections [76–78].

In addition to dual methods, mesh smoothing can be applied. Smoothing occurs when vertices are relocated following some regularity criterion, but relocation does not affect the topology of the mesh. For example, several sweeps of a spatial Laplacian smoothing operator relaxes each adjustable vertex to the arithmetic average of adjacent vertices [79]. Alternatively, the smoothing could be incorporated into an optimization algorithm to allow adjustment of either individual or independent sets of vertices in parallel [80]. The amount of smoothing a vertex is subjected to could be controlled by hierarchical order of classification. Vertices are categorized by increasing rank levels, and those within the highest class are most affected by the smoothing operations.

In mesh quality analysis, there are two types of element classifications that are applicable to both hexahedral and tetrahedral shapes. The low order linear or quadratic *h-element* corresponds to the step size needed to converge the biomechanical behavior solutions or minimize the error obtained for the actual analysis. A smaller mesh size h represents a finer mesh consisting of a larger number of elements in the model. This is how a finer mesh in areas of high interest (e.g., areas of stress concentration) can be constructed. The *p-element* is a metric for the optimization of the mesh at a higher, polynomial, order: in p-element refinement, edges are no longer straight lines, but polynomial curves. The increased polynomial order p of the element allows the element edges to adapt to curved boundaries more accurately and thereby leads to minimization of discretization error and solution convergence during the analysis of the model. In contrast to h-elements, fewer elements are required to obtain convergence, and the surface discretization error can be reduced without increasing the number of elements within the mesh. Figure 2.6 demonstrates how a mesh can be refined by h-elements, while optimization using p-elements is illustrated in Figure 2.8.

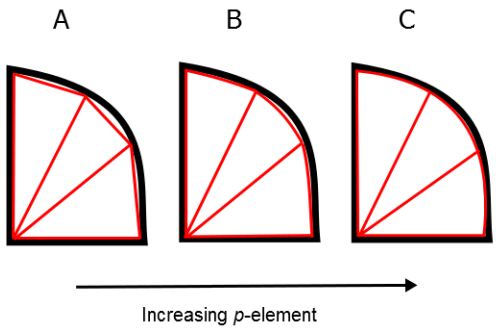

Figure 2.8: *Observation of increasing p-elements in triangular/tetrahedral mesh. First-order linear elements do not capture the curved boundaries of the model (A). The mesh improves its adaption to the curvature of the model when the order of the p-element is increased to quadratic (B). Higher order cubic element allows even better optimization as the mesh more accurately adapts to the curved regions of the model (C).*

2.3.4 Tetrahedral vs. Hexahedral Elements in Biomechanics

The preference of all-tetrahedral or all-hexahedral[2] elements in finite element modeling continues to an unsolved topic of debate. Each has advantages and disadvantages. Tetrahedral elements are widely selected due to their geometrical flexibility, and they produce acceptable displacement behavior [81]. Hexahedral elements are well suited to model stresses and strains in tissues. However, inaccurate behavior can be observed at jagged edge areas [82]. Moreover, hexahedral elements are assumed to lead to more accurate simulation results. However, hexahedral meshes are are more challenging to generate than tetrahedral meshes.

Meshes with tetrahedral elements can match simulation accuracy of hexahedral meshes when more and smaller elements are used (i.e., increased *h*-element). Four to ten times the amount of elements are needed in tetrahedral meshes in order to achieve comparable levels of accuracy provided by hexahedral elements [83–85]. Tetrahedral elements of lower order are overly stiff and lock in instances of modeling extensive stress deformations or in materials

[2]In FEM terminology, the reference is the number of facets. A hexahedral element has six sides, and a cuboid would be a hexahedral element. A *hexagonal* prism, in FEM terminology, would count as octahedral.

that are nearly incompressible. They are also more prone to a failure of the inversion problem during analysis [86]. Many of these differences can be observed and justified in Figures 2.6 and 2.8. One possible solution to this dilemma is to create hybrid meshes which are constructed with an outer surface of all-tetrahedral elements and inner volume of all-hexahedral elements [7, 87].

2.4 Simulation

Once the mesh is generated, the actual analysis can be performed. The analysis consists of the assignment of material properties and boundary conditions, followed by numerically solving the constituent partial differential equations. Overall, this step represents a numerical *simulation* of the material behavior under the given boundary conditions.

It is debatable whether the assignment of material properties and boundary conditions is part of the meshing process or part of the simulation process. In meshes obtained from medical images, material properties are often extracted from the image. However, because material properties and boundary conditions can easily be changed for simulation purposes, they are covered in this section.

2.4.1 Boundary and Loading Constraints

Some of the nodes and and elements of the mesh must be supplemented with additional information that is strongly dependent on the analysis application. Accurate simulation may require certain regions of the model to be loaded with prescribed displacements, nodal forces, pressure forces, body forces, or surface tractions, velocity, or flux. Such conditions occurring on the boundary are referred to as non-essential or *Neumann boundary conditions*. Specifi-

cations also need to be made in reference to how the model interacts with its surroundings. These are the essential or *Dirichlet boundary conditions* and refer to the rotation, support, or fixation of the region.

2.4.2 Material Properties

Material characteristic values (e.g. elastic modulus) are then assigned to the model in order to numerically distinguish it as bone, soft tissue, muscle, fluidic or a combination of materials from an engineering aspect in terms of strength, elasticity, durability, conductivity, and porosity. Material properties can be:

- Homogeneous: Same at all locations
- Non-homogeneous: Location-dependent
- Isotropic: Same in any direction
- Anisotropic: Direction dependent
- Orthotropic: Symmetric with respect to $x-y$, $y-z$, or $x-z$ planes

Examples for isotropic tissues are those in the cardiovascular system and neurological tissues. Typical anisotropic tissues are ligaments, tendons and cartilage. Both cortical and trabecular bone exhibit orthotropic behavior. Soft tissue also exhibits *creep*, that is, its stress-strain behavior is time-dependent. FEM solvers need to be capable of incorporating these different types of material descriptions for the simulation of biological tissues.

2.4.3 Governing Principle

In the basic theory of FEM, each subdivision, or element is associated with a finite number of degrees of freedom (DOF), which are the unknown function values at the nodal points which the element is composed of. The constraints, mechanical properties, and loads applied at the

nodes are condensed into discrete differential equations corresponding to the element's response. A node can be shared by several elements, therefore the deflection at the shared node is representative of deflection of the sharing elements at the node location. Shape functions use interpolation to approximate deflection values occurring at all non-nodal points within an element. The assembly of these individual element equations leads to the generation of the global equations representing the entire meshed region. Both the individual and global equations are always expressed in the form:

$$\{F\} = [K]\{u\} \tag{2.1}$$

where

$\{F\}$ is the column matrix of the applied external force or nodal loads.

$[K]$ is a stiffness matrix containing the geometric and material information which determines the element's or (mesh's) resistance to deformation when subjected to loading. Specifically, this matrix is:

- Symmetric, as a result of equal and opposite forces to ensure equilibrium
- Singular, since the equation has not been solved. Boundary conditions must be included at this point to alleviate this.

$\{u\}$ is the column matrix of nodal displacements (i.e., the degrees of freedom) resulting from the application of load $\{F\}$. These values are interdependent since the body is continuous and elastic.

Equation 2.1 is solved for the nodal displacements $\{u\}$. Once this information is obtained, the stresses (σ) can be determined from kinematic relationships, and the strain (ϵ) is calculated through material or constitutive relationships in solid mechanics analysis. The

general form Equation 2.1 governs many physical phenomena, although the parameters $\{u\}$ and $\{F\}$ have different physical significance. It should also be noted in applications in which there is no measure of deformation, the matrix $[K]$ is still referred to as the "stiffness matrix" to maintain uniformity in nomenclature and definition. Table 2.1 lists these physical phenomena and the interpretation of $\{u\}$ and $\{F\}$ for some applications.

Table 2.1: *Interpretation of the displacement matrix $\{u\}$ and load matrix $\{F\}$ in different physical problems that obey the general form of Equation 2.1*

Analysis/Application	$\{\mathbf{u}\}$	$\{\mathbf{F}\}$
Acoustic Fluid	Displacement Potential	Particle Velocity
Electrostatics	Electric Potential	Charge Density
General Flow	Velocity	Flux
Heat Conduction	Temperature	Heat Flux
Potential Flow	Pressure Gradient	Velocity Potential
Magnetostatics	Magnetic Potential	Magnetic Intensity
Structures and Solid Mechanics	Displacement	Mechanical Load

It is possible to combine physical phenomena and subject them to a joint simulation. One example is blood flow in the vasculature. Fluid flow exerts shear stress on the tissue at the interface and can cause viscoelastic deformation. Deformation can also be caused by fluid pressure, such as blood pressure. Such models are referred to as *biphasic models*. The combination of biphasic models, like the brain models presented in Chapter 4, with nonlinear properties can lead to models of appreciable complexity; however, these models still follow the general outline presented in the next section.

2.4.4 Approximate Solution Determination

Finite element analysis solvers commonly obtain the approximate solution to Equation 2.1 via an indirect formulation method based on the minimum total potential energy (TPE). The total potential energy (Π) for a model composed of n elements and n nodes is the difference between the total strain energy (Λ) and the work done by the external forces ($F \cdot u$) and is given by

$$\Pi = \sum_{e=1}^{n} \Lambda^{(e)} - \sum_{i=1}^{m} F_i u_i \tag{2.2}$$

where the strain energy per unit volume for linear elastic materials is

$$\Lambda^{(e)} = \frac{1}{2} \int \sigma \epsilon \mathrm{d}V. \tag{2.3}$$

This is ideal for models with solid materials where the strain energy of the system can be calculated. During deformation, the work done by the applied load F is stored as elastic or strain energy. The model is considered to be at equilibrium when the total potential energy is minimal with respect to displacement u. Therefore the solution is found by

$$\frac{\partial \Pi}{\partial u_i} = \frac{\partial}{\partial u_i} \sum_{e=1}^{n} \Lambda^{(e)} - \frac{\partial}{\partial u_i} \sum_{i=1}^{m} F_i u_i = 0 \quad i = 1, 2, 3, \ldots \tag{2.4}$$

in accordance to the law of conservation of energy. Each finite element will have its own total potential energy.

For simplicity, we will use the discrete connected system of springs in Figure 2.9 to illustrate the FEA procedure. There are 4 springs, 3 nodes, and 2 external loads present in the system. The total potential energy is given by

$$\Pi = \frac{1}{2} k_1 \delta_1^2 + \frac{1}{2} k_2 \delta_2^2 + \frac{1}{2} k_3 \delta_3^2 + \frac{1}{2} k_4 \delta_4^2 - F_1 u_1 - F_3 u_3 \tag{2.5}$$

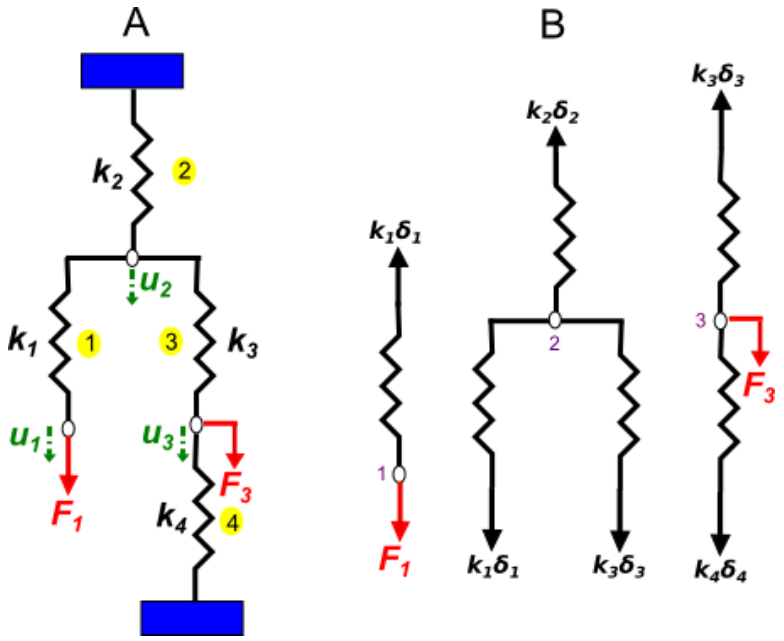

Figure 2.9: *Discrete spring system used to derive FEA theory (A). The system is composed of four springs, and is subjected to two external loads F_1 and F_3 as well as three displacements u_1, u_2, and u_3. Free-body diagram representing each spring component of system's mechanical behavior (B).*

where δ_1, δ_2, δ_3, δ_4 corresponds to the amount each spring stretches and can be rewritten as

$$\begin{aligned} \delta_1 &= u_1 - u_2 \\ \delta_2 &= u_2 \\ \delta_3 &= u_3 - u_2 \\ \delta_4 &= -u_3 \end{aligned} \tag{2.6}$$

and substitution into Equation 2.5 yields

$$\Pi = \frac{1}{2}k_1(u_1 - u_2)^2 + \frac{1}{2}k_2u_2^2 + \frac{1}{2}k_3(u_3 - u_2)^2 + \frac{1}{2}k_4q_3^2 - F_1u_1 - F_3u_3 \tag{2.7}$$

such that u_1, u_2, and u_3 are the displacements of each node.

In order for equilibrium to be reached, the total potential energy needs to be minimized with respect to the three displacements as explained in Equation 2.4. Therefore, each spring's mechanical behavior can be defined by a set of differential equations:

$$\begin{aligned}
\frac{\partial \Pi}{\partial u_1} &= k_1(u_1 - u_2) - F_1 = 0 \\
\frac{\partial \Pi}{\partial u_2} &= -k_1(u_1 - u_2) + k_2 u_2 - k_3(u_3 - u_2) = 0 \\
\frac{\partial \Pi}{\partial u_3} &= k_3(u_3 - u_2) + k_4 u_3 - F_3 = 0
\end{aligned} \tag{2.8}$$

These differential equations are converted into algebraic equations, followed by the each k value being arranged into a stiffness matrix using the general form:

$$K^{(e)} = \begin{bmatrix} k_e & -k_e \\ -k_e & k_e \end{bmatrix} \tag{2.9}$$

where e represents the 4 spring systems (k_1, k_2, k_3, and k_4). Each of these system matrices can be combined such that a global stiffness matrix for the overall system is obtained and expressed as:

$$K^{(G)} = \begin{bmatrix} k_1 & -k_1 & 0 & 0 \\ -k_1 & k_1 + k_2 + k_3 & -k_2 - k_3 & 0 \\ 0 & -k_2 - k_3 & k_2 + k_3 + k_4 & -k_4 \\ 0 & 0 & -k_4 & k_4 \end{bmatrix}. \tag{2.10}$$

Based on the algebraic equation from each node,

$$
\begin{aligned}
k_1\delta_1 &= F_1 \\
k_2\delta_2 - k_1\delta_1 - k_3\delta_3 &= 0 \\
k_3\delta_3 - k_4\delta_4 &= F_3
\end{aligned}
\tag{2.11}
$$

and the boundary conditions of our system, the k_2 components in the 3rd column as well as the 4th row and column of the matrix can be deleted due to the spatial fixation of the nodes at those locations. Now, the algebraic form can be rewritten in the general form as presented in Equation 2.1,

$$
\begin{Bmatrix} F_1 \\ 0 \\ F_3 \end{Bmatrix} = \begin{bmatrix} k_1 & -k_1 & 0 \\ -k_1 & k_1 + k_2 + k_3 & -k_3 \\ 0 & -k_3 & k_3 + k_4 \end{bmatrix} \begin{Bmatrix} u_1 \\ u_2 \\ u_3 \end{Bmatrix} \tag{2.12}
$$

and solved by inversion of the matrix $[K]$.

In order to observe the mechanical behavior of the spring system, let us assume that each spring has a uniform cross-section area A and length l when subjected to a force F. The average stress (σ) in a spring is given by

$$
\sigma = \frac{F}{A} \tag{2.13}
$$

and the average normal strain (ϵ) is the change in length Δl per unit original length of the spring

$$
\epsilon = \frac{\Delta l}{l} \tag{2.14}
$$

Hooke's law relates stress and strain in elastic regions through the equation

$$\sigma = E\epsilon \tag{2.15}$$

where E is the modulus of elasticity of the spring's material. Rearranging and substituting Equations 2.14, 2.13, and 2.15 to isolate F results in

$$F = \left(\frac{AE}{l}\right)\Delta l \tag{2.16}$$

which bears resemblance to the equation of a linear spring. Therefore we can conclude in this case that each spring has a stiffness k of

$$k_e = \frac{AE}{l}. \tag{2.17}$$

As described above, solving for the displacement values $\{u\}$ allows the stresses and strains to be determined. The stress for each spring is calculated by

$$\sigma = \frac{F}{A} = \frac{k_e(u_{i+1} - u_i)}{A} = \frac{\frac{AE}{l}(u_{i+1} - u_i)}{A} = E\left(\frac{u_{i+1} - u_i}{l}\right) \tag{2.18}$$

and the strain can be determined from Equation 2.15 or from the displacement of each spring at the nodes,

$$\epsilon = \left(\frac{u_{i+1} - u_i}{l}\right) \tag{2.19}$$

Although we presented a simplified example, the FEM simulation obeys the exact same principle, because each node of the volumetric mesh is examined in the same fashion as the nodes in Figure 2.9, with the spring constants determined by the neighboring nodes and the material properties in the respective element. It should be noted that in more complex

2D and 3D models the directional components of stress and strain are considered (stress and strain tensors), and that the stiffness k is dependent on the area of the model and the analysis application (i.e. solid mechanics, heat transfer, etc).

2.4.5 Non-Linear Analysis

Biomaterials often exhibit nonlinear behavior. A deviation from linear behavior (i.e., Hooke's law) not only occurs for large geometry changes that lead to fracture, but also include plastic deformation and creep. These situations result in non-linear and even time-dependent σ-ϵ and ϵ-u relationships, where occurrences of stiffening, hardening, softening, or buckling are possible. In such cases, the stiffness matrix $[K]$ becomes time-dependent. The solution is usually obtained by incremental or iterative methods, such as the implicit Newton-Raphson approach. In this approach, a load is applied in discrete time steps that advance the simulation from t to $t + \Delta t$. This technique addresses material non-linearity by solving for the state at time $t + \Delta t$ when displacements at u^t are known. The load increment is applied such that the state is updated to $t + \Delta t$, and $u^{t+\Delta t}$ is the main variable. The principles of the Newton-Raphson approach can be integrated into the TPE method and represented as

$$\Phi(u^{t+\Delta t}) = \int \sigma\epsilon(u^{t+\Delta t})\mathrm{d}V - F = 0 \tag{2.20}$$

where Φ represents the sum of internal and external residual forces and $\Phi = 0$ is a set of non-linear equations in $\{u\}$. Given that $\Phi(u^{t+\Delta t}) = 0$, for an i^{th} iteration:

$$u_{i+1}^{t+\Delta t} = u_i^{t+\Delta t} - \left[\frac{\partial}{\partial u}\Phi(u_i^{t+\Delta t})\right]^{-1} \Phi(u_i^{t+\Delta t}) \tag{2.21}$$

can be rearranged such that

$$\delta u_{i+1} = u_{i+1}^{t+\Delta t} - u_i^{t+\Delta t} = -\left[\frac{\partial}{\partial u}\Phi(u_i^{t+\Delta t})\right]^{-1} \Phi(u_i^{t+\Delta t}) \tag{2.22}$$

where

$$[K_T] = \left[\frac{\partial}{\partial u}\Phi(u_i^{t+\Delta t})\right]^{-1} \tag{2.23}$$

is the total tangential stiffness matrix for which holds

$$[K_T(u_i^{t+\Delta t})] = \left[K^a(u_i^{t+\Delta t}) + K^b(u_i^{t+\Delta t}) + K^c(u_i^{t+\Delta t})\right] \tag{2.24}$$

where

$[K]^a$ is a linear matrix that includes small strain and displacement model components

$[K]^b$ houses linear and quadratic strain terms and is considered the large displacement or stiffness matrix

$[K]^c$ is a non-Linear matrix composed of quadratic strains and non-linearities associated with the material.

Substitution yields

$$\delta u_{i+1} = -K_T(u_i^{t+\Delta t})^{-1}\Phi(u_i^{t+\Delta t}) \tag{2.25}$$

and therefore

$$-\{\Phi(u_i^{t+\Delta t})\} = [K_T(u_i^{t+\Delta t})]\{\delta u_{i+1}\} \tag{2.26}$$

is in the general form presented in Equation 2.1. This must be solved for each iteration i to advance the simulation by Δt and obtain the change in incremental displacements

$$u_i^{t+\Delta t} - u_i^t = \Delta u_i = \sum_{k=1}^{i} \delta u_k \tag{2.27}$$

whereby K_T and Φ are different for each iteration. A quadratic convergence rate towards a tolerance threshold ϵ_T is achieved when

$$\left|\{\Phi(u_{i+1}^{t+\Delta t})\}\right| < \epsilon_T. \tag{2.28}$$

Because Φ needs an accurate stress value (σ) to obtain a current approximation of $u^{t+\Delta t}$, the updated Lagrangian approach is implemented as the solution process moves from $t \rightarrow \Delta t$ to iteratively follow elements of the model along their configuration. All derivatives and integrals are determined with respect to spatial coordinates.

2.5 Post-Processing: Output and Visualization

The solutions to equations evaluated in the analysis are stored as raw data. In post-processing, the results are displayed in either FEA or visualization software as graphic contour plots, line plots, or tabular charts. These results represent behavioral effects (i.e. areas of high stress, strain, and pressure or shape distortion due to factors such as bending, twisting, swelling, clotting, hemorrhaging, obstruction, constriction, dilation, compression, impact, or fracture) at all points on the model resulting from the conditions imposed. Often, the original geometry is displayed, and one specific property, such as the stress magnitude, superimposed in a false-color scheme. In this fashion, high-stress regions, can be visually identified. Figure 2.10 demonstrates some of the possible FEA visualizations.

There is an almost limitless number of visualizations, and in many ways, a good visual representation of the result contains a considerable artistic element. In fact, it is recommended that a *visualization strategy* [88] is developed beforehand, which is determined by the key conclusions of the simulation.

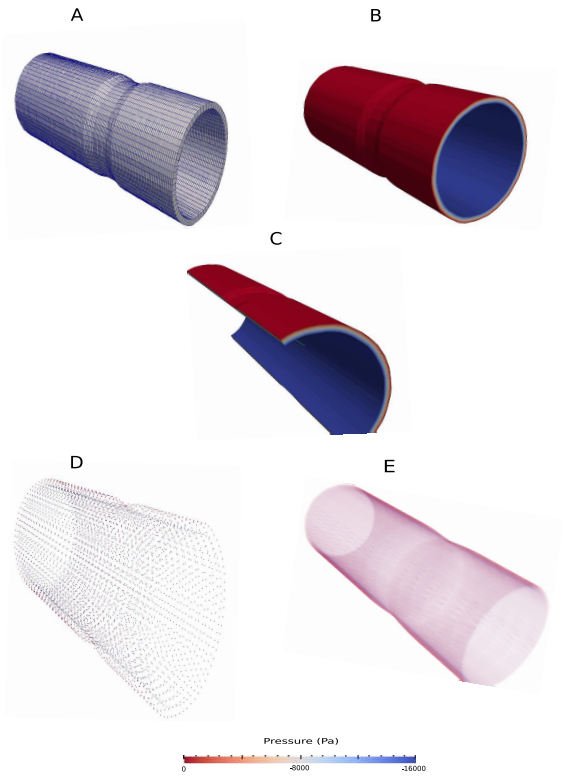

Figure 2.10: *Visualization results of a tubular phantom model using Paraview software. The meshed model of the tubular construct is presented (A). The pressure expansion and distribution behaviors throughout the entire tube (B), a section of the tube (C), the entire tube using vector dots (D), and the entire tube using a volumetric rendering (E) are also shown.*

2.6 Software

There is a large number of software options available for use in image segmentation, mesh generation, finite element analysis, and visualization applications. Specifically, review of biomechanical finite element analysis literature reveals frequently-used commercial options:

Image Segmentation/Meshing:

- MIMICS (Materialise, Leuven, Belgium)

- ScanIP+FE (Simpleware Ltd., Exeter, UK)
- 3D-DOCTOR (Able Software Corp., Lexington, MA, USA)

FEA/ Visualization:

- Abaqus (Simulia (Dassault Systèmes), Vélizy-Villacoublay Cedex, France)
- Algor (Autodesk)
- ANSYS (ANSYS, Inc., Canonsburg, PA)
- COMSOL (COMSOL, Inc.)

Software companies usually try to offer monolithic solutions. For example, many combined FEM and Visualization packages also come with a model builder tool, which allows to assemble models from relatively simple geometrical elements. In addition, these packages come with several import options to read either surface models or solid models that have been created with other packages. This collection of features makes these packages ideal for the analysis of classical load and stress problems in mechanical engineering. One advantage of the all-in-one approach of monolithic software is the ability of the individual steps to seamlessly interface with each other. Unlike solutions where multiple software modules need to use a common file format for data interchange, a monolithic package handles its data transfer internally and usually invisible to the user.

The creation of finite-element models from biomedical images is more challenging for the monolithic packages. Segmentation and surface parametrization is usually not included, and an external program, such as Mimics, is needed to perform the first steps in the analysis chain. Although the segmentation and meshing software is aimed at a broad range of applications, it may fail in some special cases, in part due to the fact that no universal segmentation and meshing approach exists. This shortcoming has led many research groups to develop application-specific modules (e.g., [70, 89–91]). Meshing-related studies often do not address

the issue of segmentation. In fact, segmentation is usually considered to be a separate step, although one pre-processing toolkit [22] specializes in combining segmentation and meshing of 3D models.

Analysis of the pertinent literature reveals that a focus of the software development lies on the meshing algorithm, because the transition from a voxel-based image to an analytical description of the object (i.e., the mesh) remains the most critical step in the entire FEM chain.

2.6.1 Image-Based Finite-Element Analysis with Open-Source Software

Frequently, research groups not only describe their methods, but also release the actual software source code for use by other interested parties. In fact, a strong community-driven effort to develop and make available free, open-source software has evolved over the last several years, from which interested researchers can immensely benefit. The philosophy of the open source community places an emphasis not only on free-of-charge availability of such software, but more importantly encourages users share software source code. As a consequence of this openness – and the associated freedoms to inspect and modify the software code – an Internet-connected community continuously works on improving the software and adding new features. Furthermore, the same community is generally available through Internet forums to respond to user questions. Free software, therefore, can be seen as a parallel model to the commercial software model with analogous characteristics of software distribution and user support through the Internet. Since the software is free of license fees, a researcher can download and install software to try without further risk whether the software is suitable for the specific application.

In many cases, wide distribution and continuous improvement has lead to the evolution of high-value software. Popular free, open-source software is available not only as program code. Rather, some members of the community create installation-ready software packages that can be conveniently installed without programming experience[3]. Specifically for medical image processing, segmentation and feature extraction, meshing, finite-element analysis and visualization, several options exist that make a fully open-source FEM toolchain feasible. Since this field of development currently exhibits a strong momentum, it is covered in the next sections in detail.

Some examples for free open-source software packages that fulfill some of the functionality laid out in Figure 2.1 are:

Image Processing and Segmentation:

- ImageJ (US National Institutes of Health, Bethesda, Maryland)
- OsiriX [92]
- Crystal Image [28]
- Seg3D (Scientific Computing and Imaging Institute, University of Utah)

Combined Segmentation/Meshing Applications:

- BiMECH [22]
- IA-MESH [93]
- Works presented in [72, 73, 94, 95]

[3]One example are the repositories of the popular *ubuntu* operating system, which contain over 20,000 installation-ready software packages ranging from office suites to scientific applications. The repositories are linked to the operating system in such a fashion that the installation of new software is no more effort than a few mouse clicks.

Meshing:

- BioMesh3D (Scientific Computing and Imaging Institute, University of Utah)
- Cleaver [96]
- Gmsh (www.geuz.org/gmsh)
- MeshExtract [97]
- MeshLab (meshlab.sourgeforce.net)
- Works presented in [7, 70, 86, 89, 97, 98]

Finite-Element Analysis

- FeBio [99]
- FreeFEM (www.freefem.org)
- Elmer (CSC, Espoo, Finland)
- Tochnog (tochnog.sourceforge.net)

Visualization:

- Paraview (Kitware, Inc., Clifton Park, NY)
- OpenDX (www.opendx.org).

Open-source software for image-based FEM has not yet reached the main-stream of the FEM community, and a careful discussion of the advantages and limitations of a free-software approach is needed. Software modules that find their way into the open-source community are often incorporated to bridge a gap or overcome challenges encountered within the execution of the flow process outlined in Figure 2.1. As such, some open-source packages are limited to special cases. However, frequently those packages are "absorbed" by the wider community and integrated into a larger system of software modules.

Before integrating open-source software into the research process, the following actions can help determine the best overall approach:

- Identify what role(s) the needed software should address
- Assess capabilities of the software options currently accessible within the research facility. First, site licenses reduce per-user fees for commercial software. Second, available software can be evaluated for suitability for a specific purpose.
- Assess capabilities of commercially-available software. If the license fees of a commercial software package are not an obstacle, monolithic software can offer a streamlined solution that promises the least investment of time by the user.
- If an open-source package is being considered, a literature search is helpful to determine if similar solutions have been reported and whether method is applicable to desired research objectives

The most frequent contributions to open-source software related to biomechanical FEM analysis are in the form of meshing modules or toolkits that discretize the geometries obtained from segmentation of medical imaging data [4] . Suitable representations of biomedical objects often lead to excessive numbers of nodes, elements, vertices, and faces that tend to overwhelm the computational specifications of commercial meshing and FE software, which are usually designed to handle CAD-based analytical geometries. As previously mentioned, commercial meshing software is also often somewhat limited in mesh functionalities and capabilities. The other major factor attributing to the recent surge in published mesh generation methods and algorithms can be attributed to the diverse field of biomedical applications, including bone, soft tissue (muscles, organs, blood vessels, implants); the medical image source, and

[4] A comprehensive list can be found at `http://www.robertschneiders.de/meshgeneration/software.html#public_domain`

the properties of the mesh (e.g., allowable discretization error, mesh complexity, and the inclusion of material properties). Moreover, reproducibility is a key factor: since open-source software allows the user to examine the actual "inner workings" of the program, a very high level of transparency is given. For scientific applications, transparency (and by association reproducibility) is of prime concern. Commercial "black-box" software does not provide this advantage: it is well possible that slightly different numerical implementations can lead to divergent results between different closed-source software packages, and even between different versions of the same package.

2.6.2 Completely Open-Source Based FEA

Some meshing modules have been developed with the sole aim of being interfaced or used in conjunction with a specific commercial FE software. To our knowledge, there are only two reported instances in which the entire analysis toolchain has been implemented exclusively with open-source software. The first instance examines patient-specific data in order to identify optimal locations for placement of Implantable Cardiac Defibrillators (ICD) [100, 101] and the second study proposes a process to evaluate the biomechanical behavior of autologous tissue-engineered blood vessels [97]. The work presented by Jolly *et.al* [100,101] features customization through the addition and modification of visualization, mesh refinement, and finite element calculation parameters within currently available open-source software (notably the software offered by the Scientific Computing and Imaging Institute, University of Utah). Our toolchain [97] revolves around the development of a hexagonal mesh-extraction module capable of adaptive mesh refinement, assignment of material properties, loading, and boundary conditions, and automatic generation of input file data needed to conduct the actual analysis performed with standalone FE and visualization software. Both studies compare the results obtained to those provided by commercial counterparts for proof of principle validation.

2.6.3 Limitations of Open-Source Software

The vast assortment of currently available open-source software options for image processing, meshing, analysis, and visualization may give rise to the misconception that construction of a FEM toolchain can be done with ease. Fortunately, the continuing adoption of open-source software into scientific computing in general leads to significant streamlining of the data analysis process. Through the organization of community-based libraries, such as the Insight Segmentation and Registration (ITK), Finite Element (FETK), and Visualization (VTK) toolkits, there exists often a central "hub" or headquarters in which developers share, debug, update, modify, and provide cross-platform support for related open-source software. To provide one example, the Scientific Computing and Imaging Institute at the University of Utah (`http://www.sci.utah.edu`) has provided a comprehensive package of interrelated modules for image-based FEA. Notable elements of this package are `Seg3D` for volumetric image segmentation, `BioMesh3D` as a mesh generator for segmented biomedical images, and `SciRUN`, termed *problem solving environment* (PSE), which allows operations with significant manual interaction, such as manual delineation and segmentation. The same research group provides `FEBio` for solving non-linear FEA problems specifically in biomedical applications [99], `PreView` and `PostView` for pre- and post-processing, `ImageVis3D` for volume rendering and visualization, and `AtlasWerks` for medical image atlas generation.

Not all software is as streamlined as this suite of packages. Frequently, use of open-source software requires some familiarity with computer programming. In some cases, the software is provided in source-code form *only*, and the user has to compile the software (that is, convert it into an executable application). The compilation process requires that development tools are installed and available. Furthermore, such software often depends on libraries (such as the ITK and VTK libraries mentioned above) that have to be installed separately.

The main challenge that the creation of a completely open-source FEM toolchain poses is the interfacing between different software applications. Since the need to interface different software programs is a known problem, some file formats are likely to emerge as universal mesh data exchange formats. For example, the STL format (STL stands for *stereo lithography*) is widely used to describe arbitrarily complex surfaces, and it is probably best known as the input format for rapid prototyping. A file format known as STEP (Standard for the Exchange of Product Data) has become an ISO standard exchange format for representing three-dimensional data. IGES (Initial Graphics Exchange Specification) is a NIST-supported file format that allows the exchange of information among Computer-aided design (CAD) software. DXF (Drawing Exchange Format) is an alternative file format, created by AutoDesk, to exchange drawing information between CAD programs. STL and DXF files contain geometry, but not material properties or boundary conditions. STEP, which can serve as input for FreeFEM, is a format under development, but the flexible format definition would allow a STEP file to carry all information needed for a FEM analysis.

In some instances special files are needed. For example, Tochnog is a versatile solver for a large number of PDE-based physical problems, including stress/strain, heat transfer, and fluid dynamics (*cf.* Table 2.1). Tochnog also contains mechanical models for time-dependent, nonlinear materials, which makes it very attractive for modeling of biological tissue. Tochnog uses a non-standard plain-text input file that contains the spatial coordinates of all mesh vertices, the list of elements and their associated vertices, material groups and the assignment of element to material groups, boundary conditions, and control instructions. Although Tochnog output is directly readable by Paraview and OpenDX, special software is needed to generate the input file from an existing mesh. It is therefore almost inevitable that Tochnog users need to write a small translation program that reads a mesh in its existing format (e.g., the widely used STL file format) and writes a Tochnog input file.

At this point of software availability, however, the creating of small filter programs to link individual applications is a small price to be paid for the advantage of source code availability. Commercial software is distributed in black-box form, and user support only reaches the extent of the existing software as provided. Given the complexity of biological geometry, the software often becomes overwhelmed and freezes or stops processing. The authors have experienced this effect with two different commercial packages that attempt to create their own volumetric mesh from a given surface mesh, yet fail with complex geometries, such as the volumetric CT image of dentures, or even an inhomogeneous tubular phantom [97]. Due to the black-box nature of commercial software, the failure reason cannot be determined beyond the help available from customer support. Conversely, an open-source package would allow to identify the point of failure and either remedy the problem (for example, by increasing the memory limits) or adjust the input file to create a workaround that avoids the point of failure.

2.7 Future Research and Trends

The need for robust, completely automatic methods for image segmentation and mesh generation of geometric models obtained from medical imaging data continues to drive the research within this realm. As the capabilities of medical imaging coupled with FEA continue to expand past engineering and computational mathematic based labs into hospitals and research groups composed of medical professionals, engineers, and statisticians, techniques that are accurate, reproducible, require minimal user intervention and can generate models within a short timeframe are in high demand. The evolving focus areas of Computer Aided Diagnosis (CAD) and Computer Aided Visualization and Analysis (CAVA) have catalyzed biomedical FEA applications and thereby are largely responsible for the formulation of such multidisciplinary teams. The former is implemented in the detection and diagnosis of disease, lesions,

and abnormalities while the latter is the study and development of computerized methods, such as the ones discussed in this chapter, to catalyze new strategies, education, and training [30, 102–104]. The biomechanical response behavior obtained from FEA could assist in clinical research trials to measure changes in condition due to drug and radiation therapies.

The creation of registration, template, and atlas based databases, rapid prototyping, and computational fluid dynamics (CFD) are a few CAVA related developments that can further facilitate the growth of CAD. Integration of a novel meshing algorithm that "morphs" itself around a pre-segmented medical image based on a template or atlas of the desired geometry provides a convenient way to evaluate clinical data for multiple patients at varying stages of disease and treatment [105–110]. Rapid prototyping allows the creation of physical models manufactured from 3D data obtained from medical images, and is ideal for anatomical instruction/education, pre-surgical planning, as well as the design, verification, and manufacturing of medical implants and prosthetics [111–113]. Lastly, the adaptation of computational fluid dynamics, a numerical methods simulation technique designed for the analysis of biphasic solid and fluidic models, to accommodate non-linear and complex flow patterns provides real-time simulations of blood flow and cerebrespinal fluid through ventricles in the vascular and neurological cavities [114, 115].

Chapter 3

A COMPLETELY OPEN-SOURCE FINITE ELEMENT MODELING CHAIN FOR TUBULAR TISSUE-ENGINEERED CONSTRUCTS[1]

[1]Madison, A.M. and M.A. Haidekker. 2012. *International Journal of Computer Science and Application (IJCSA)*. 1(2): 44-55.
Reprinted here with permission of the publisher.

Abstract

Finite-element modeling (FEM), well-established to predict the mechanical behavior of mechanical systems, enjoys growing popularity in the study of the biomechanical behavior of biological tissues. Contrary to mechanical systems, which can often be described analytically or with geometric primitives, biomedical objects require discretization of their often irregular shape. In noninvasive studies, imaging methods are used to obtain the shape. Frequently, a relationship between image intensity and biomechanical properties is assumed. Commercial FEM toolchains exist, but we failed to obtain a satisfactory discretization from simple phantom images. Driven by the application to image and characterize tissue-engineered blood vessels noninvasively, we sought to establish a completely open-source FEM toolchain. The open-source feature gives users the ability to modify and extend the code, and thus offers additional flexibility over commercial systems. We demonstrate that the combination of a custom module to discretize the geometry (meshing) combined with the open-source FEM solver *Tochnog* and free visualization software (namely, *Paraview* and *OpenDX*) completes an open-source FEM toolchain. We demonstrate its ability to analyze tubular phantoms modeled after tissue-engineered blood vessels and compare results to a commercial toolchain. We conclude that a fully open-source toolchain is feasible, but the critical element is the meshing module.

3.1 Introduction

Finite-element modeling (FEM) has been intensely used in biomedical contexts to examine, simulate, and predict the material behavior and non-linear biomechanical properties of soft tissues through integration with medical imaging modalities [3]. Areas of application include the brain [116], lungs [6], and most importantly to us, vascular stress analysis [117–120]. The underlying principle of FEM is to subdivide the object of interest into a large number of connected small volumes that are considered homogeneous. These small volumes – the finite elements – are considered to be rigidly connected, thus providing boundary conditions imposed upon the element by its neighbors (forces, displacement) and by the environment (pressure, fixed and immobile elements, or moving elements).

One key challenge in the application of FEM in the biomedical context is the extraction of the object of interest from a medical image (i.e., segmentation) and the conversion of the object into a connected set of finite elements (meshing) that not only represents the geometry of the object, but also the approximate local material properties in the tissue section that corresponds to the finite element. Only when this information is available – the geometry, discretized into finite elements, and the associated material properties – the response of the tissue to external mechanical loads and the pressure can be examined. Algorithms capable of constructing triangular and tetrahedral-shaped FEM meshes from medical images exist [70, 89], yet neither study addresses the issue of segmentation. In fact, segmentation is usually considered a separate step, although one pre-processing toolkit [22] specializes in combining segmentation and meshing of 3D models. The algorithms implemented are also capable of material property assignments. Another related software toolkit reported in [93] provides the assignment of both material properties as well as boundary/loading conditions.

The driving application behind the toolchain presented in this manuscript is the recent development of fully biological tissue-engineered blood vessels [1,121]. Blood vessels that are grown from the patient's own cells must be tested exhaustively on an individual sample basis because of the inter-sample variation. Our vision is to use an imaging method, such as optical coherence tomography (OCT), confocal imaging [122, 123], or optical trans-illumination tomography, then use the resulting image data to generate a FEM mesh that allows the prediction of its biomechanical behavior when exposed to pulsatile blood pressure. Studies exist where FEM was applied on vascular entities, such as arteries and veins [11, 12, 117–120, 124]. Commonly, model geometries, based on measurements obtained from real vessels, were constructed with commercial CAD and FEM software. However, these studies only focus on the results obtained from the analysis, and not specifically on the processes to prepare 3D image data for analysis; detail information in those publications was accordingly sparse.

Several finite element solvers are available from different companies. We initially used a departmental license of Algor (Algor, Inc., Pittsburgh, PA), a combination of FEM simulation software with limited visualization features. The generation and analysis of analytically-defined cylindrical models was straightforward, and no fundamental challenges were encountered. However, it was not possible to manually generate the complex and irregular shapes of a sample blood vessel. A separate software program was necessary to extract the mesh from the original image. Few such programs are offered, because engineering design often allows to describe mechanical objects in an analytical fashion or as a combination of geometrical primitives. One software program that allows to generate a mesh from medical images is MIMICS (Materialise, Leuven, Belgium). We prepared a simple phantom (Figure 3.1) to test the ability of MIMICS to generate a mesh that can be processed by Algor. In spite of extensive support from Materialise, we were not able to generate such a mesh. The Mimics meshes showed irregular element boundaries and shapes, and MIMICS produced a very large number of nodes that could not be reduced and that caused Algor to fail. We concluded that

the combination of MIMICS and Algor would not meet our requirements. To avoid paying the license fees for additional software, we decided to attempt the design of a FEM chain based entirely of free open-source software for use in the analysis of tubular constructs.

3.2 Materials and Methods

Finite-element modeling of tissue engineered blood vessels is generally performed using some form of volumetric image obtained, for example, by computed tomography (CT), magnetic resonance imaging (MRI), or – on a smaller scale – by optical coherence tomography (OCT) and involves four main steps. In the first step, the object of interest is separated from the image background (segmentation). The meshing step follows, during which the segmented object is subdivided into small elements, and boundary conditions and material properties are applied to the elements. The third step involves the actual finite element simulation of the time-variable behavior of the object under the established parameters. Once this is completed, the output of the simulation process is visualized or post-processed in the final step.

3.2.1 Phantom Generation

Two phantoms were used throughout this study that approximated pre-segmented blood vessels when scanned in air with computed tomography. Both phantoms were embedded in a 256 by 256 by 128 voxel matrix. Voxels were anisotropic with a size of approximately 23 microns in the x-y-plane and 92 microns in the axial (z) direction. The first phantom, shown in Figure 1, was modeled after an autologous vascular graft described by L'Heureux et al. [1]. The vascular graft has typical dimensions of 5.0 $\times 10^{-4}$ m (0.5mm) thickness, 4.0 $\times 10^{-2}$ m (40 mm) length, and 4.0 $\times 10^{-3}$ m (4 mm) diameter, which is closely matched by our phantom, except that the length was truncated to approximately 1.2 $\times 10^{-2}$ m (12mm).

The first phantom was widely homogeneous, but featured two annular inhomogeneities, one section where the wall thickness was reduced, and one section where the image intensity was lowered, while the wall thickness was held constant. The first section was created to simulate the fixation grooves (visible in Figure 3.2), and the second section represented incompletely fused tissue layers. This second section translates into different material groups. For comparison, the cross-section of a CT image of an actual tissue-engineered blood vessel is shown in Figure 3.2.

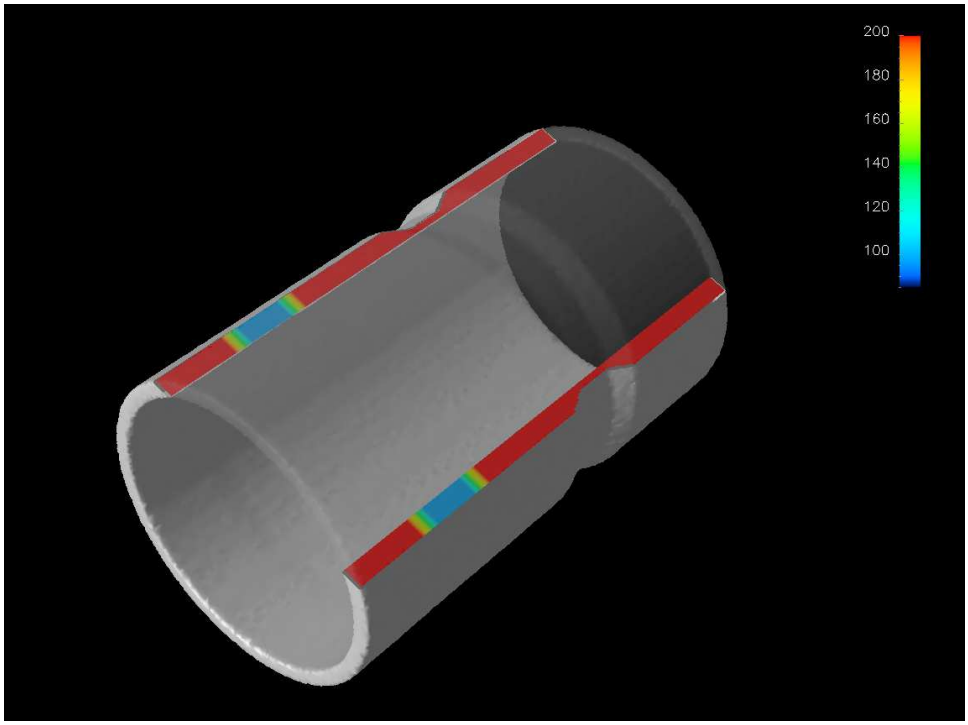

Figure 3.1: *3D Rendering of the phantom used in this study. The phantom is a tubular object that is widely homogeneous, but has two inhomogeneous regions. The first region has a thinner wall, whereas the second region has lower image values. In this rendering, the object has been clipped, and the cut surfaces have been false-colored to highlight the change in diameter and the change in image values.*

The second phantom was modeled after a fusiform aneurysm in a curved section of blood vessel. The outer dimensions, that is, 4.0 $\times 10^{-3}$ m (4 mm) diameter, 5.0 $\times 10^{-4}$ m (0.5mm) thickness, and 1.2 $\times 10^{-2}$ m (12mm) length were kept similar to the first phantom, but a 15 °

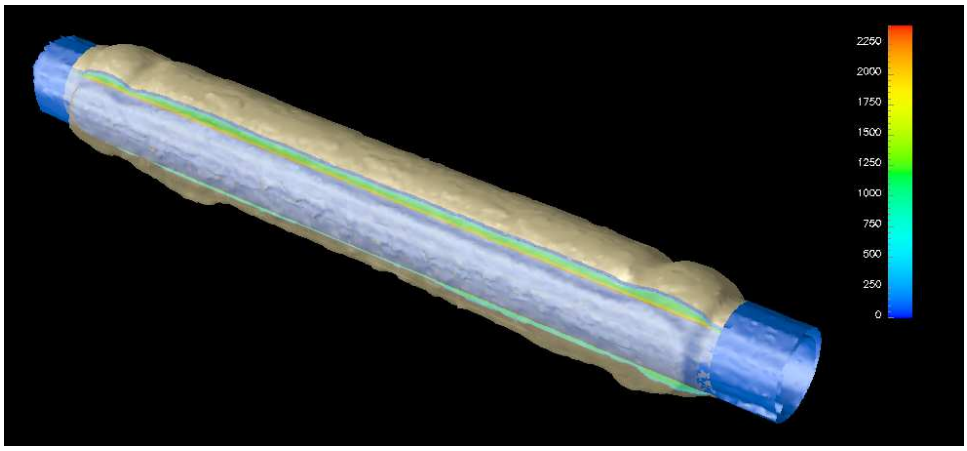

Figure 3.2: *3D Rendering of a tissue-engineered blood vessel [1] obtained by computed tomography. The tissue layer (indicated in an off-yellow color) is grown on a steel mandrel (blue) with 4 mm outer diameter. The tissue completely encloses the mandrel, but the tissue was clipped in this image to show the cross-sectional intensity distribution and the thickness irregularities. The fixation grooves show prominently at each end of the tissue section. Note that the apparent tissue density increase near the mandrel is an artifact caused by partial-volume effects.*

bend was introduced, and the central region featured an elliptical expansion, with the inner circular region filled with a simulated plaque, i.e., a region with higher material stiffness. A rendering of the second phantom is shown in Figure 3.3.

3.2.2 Mesh Extraction Module

We decided to develop a relatively simple mesh extraction module based on the principle of radial probing rays [125] that seamlessly interfaces with the chosen open source FEM software, uses the near-cylindrical geometry, and may serve as a demonstration model for more advanced meshing modules. The input of the module is assumed to be pre-segmented, that is, all background voxels have a zero value, and all voxels that belong to the vessel phantom

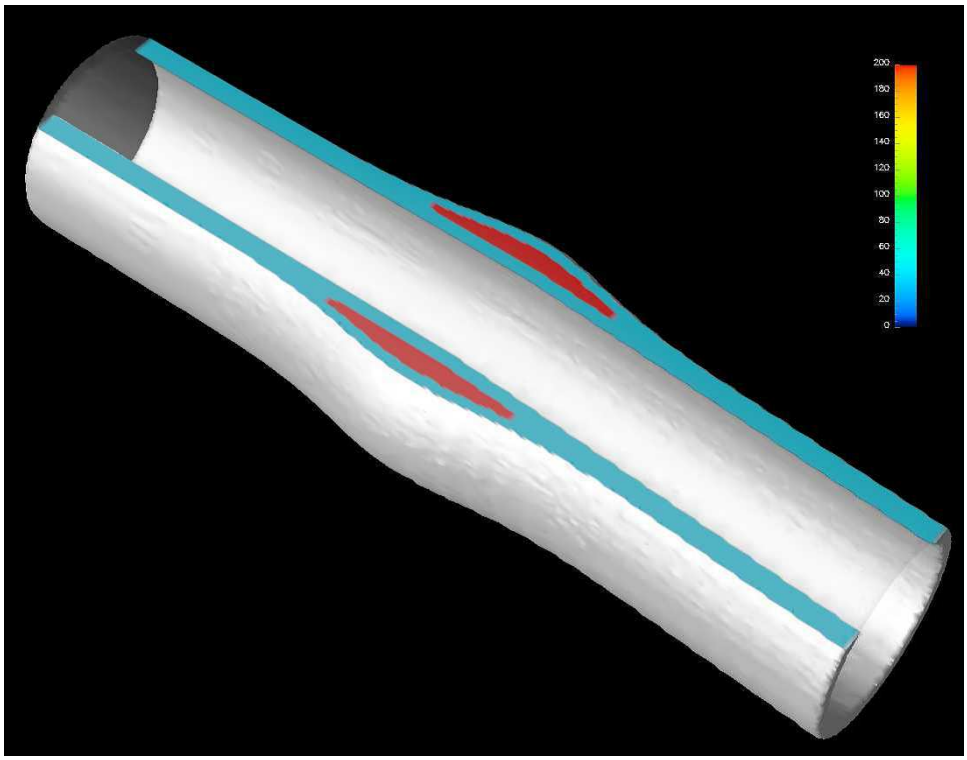

Figure 3.3: *3D Rendering of the aneurysm model for the second phantom. The model represents a fusiform aneurysm along a bent blood vessel with a 15°curve. The false-colored image values represent the material elasticity, in this case, light blue for the regular vessel and red for the more rigid plaque. The color scale bar is the same as in Figure 3.1.*

have a nonzero value. In a first pass over the image, the centroid for each slice is determined and a straight line fitted into these centroids. This line is a first-order approximation of the lumen center capable of reflecting a tubular object that is tilted or slightly curved.

In the second pass over the sample, radial lines are emitted in each slice from the point where the central line intersects the slice (Figure 3.4). Starting from this point, image values are sampled along each ray. Once a nonzero image value is encountered for the first time, this point is marked as a node on the inner wall. Tracing continues until the image values drop to zero again, and this point is marked as a node on the outer wall. Thus each ray provides one pair of nodes. Two adjoining rays together with their corresponding rays in the next

slice provide eight nodes, and these eight nodes define one finite element. The relationship of the rays and nodes to one element is shown in Figure3.5. At the same time, image values inside the approximated cuboid are averaged to provide a material index for the respective element. After user-selectable binning, the element can be assigned to a material group, and Young's discretization, mesh refinement by interpolation is possible. When interpolation is selected, a user-defined number of nodes is interpolated between two adjoining rays by means of natural cubic splines.

The meshing module alternatively outputs a STL (stereo lithography) file for immediate visualization, or a Tochnog input/control file. The Tochnog file structure, together with an explanation of the required sections for the FEM software, is provided in the Appendix.

3.2.3 FEM Software

A mature FEM solver that is available under a free open-source license exists. Tochnog, an implicit/explicit solver for a number of different problems, including linear and nonlinear solids, fluids and gases (Navier-Stokes), and for solving the wave equation, can be found at `tochnog.sourceforge.net`. The output file generated by the mesh extraction module directly serves as a Tochnog input/control file, and only minimal editing (specifically, modification of the material definition) is needed. Control statements within the Tochnog input file determine its output format. At the end of the simulation run, the output file contains the node locations, node pressure, temperature, stress, strain, and velocity for each time step. Tochnog generates output files that are directly compatible with various post-processing and visualization packages.

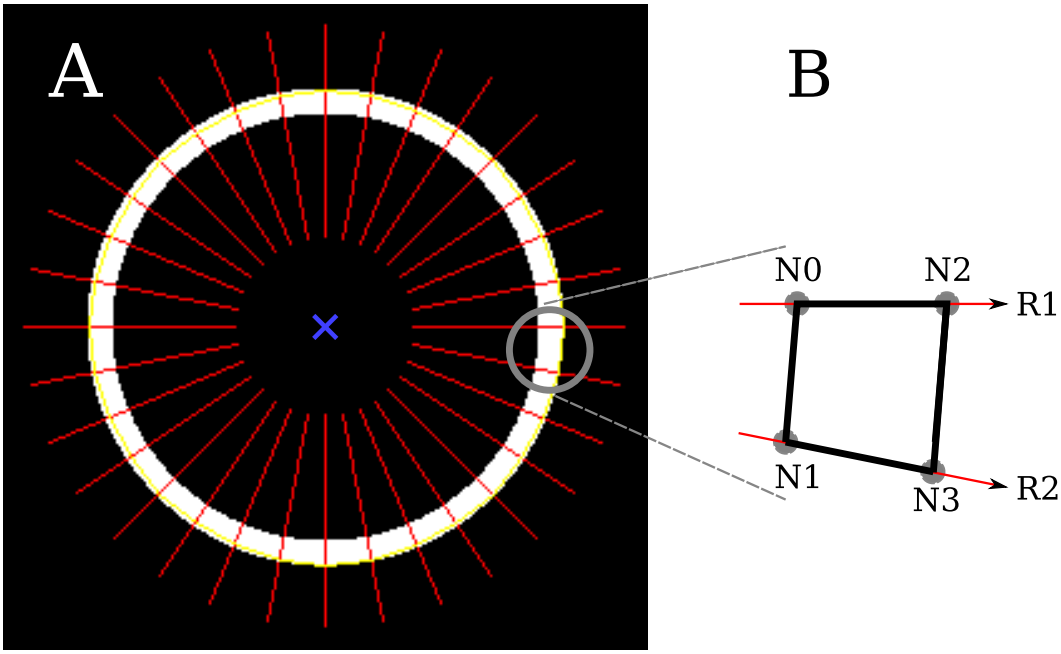

Figure 3.4: *Extraction of the boundaries of a convex, tubular object with probing rays (A). In each slice, probing rays are emitted from the centroid (blue x-mark) at regular, adjustable angular intervals. Image values are sampled along the probing rays. When a pre-selected threshold is first exceeded, an intersection (node) of the ray with the inner wall is recorded. Once the image values drop below the threshold again along the ray, the intersection of the ray with the outer wall is recorded. Two subsequent rays (R1 and R2) therefore define a quadrilateral, which is one face of an element. A magnified section (B) shows the nodes. The numbering of the nodes corresponds to Figure 3.5, and nodes need to be ordered as indicated by the node numbers.*

3.2.4 Visualization

The visualization program GiD (CIMNE, Barcelona, Spain) is recommended by Tochnog. GiD is available without a license fee, but the no-cost version comes with a time limit. We examined both Paraview (Kitware, Inc., Clifton Park, NY) and OpenDX (`www.opendx.org`) as alternative visualization modules at the end of the FEM processing chain. Paraview accepts many input formats, and we used the VTK (visualization toolkit) format to exchange data between Tochnog and Paraview. OpenDX requires a specific input format, which

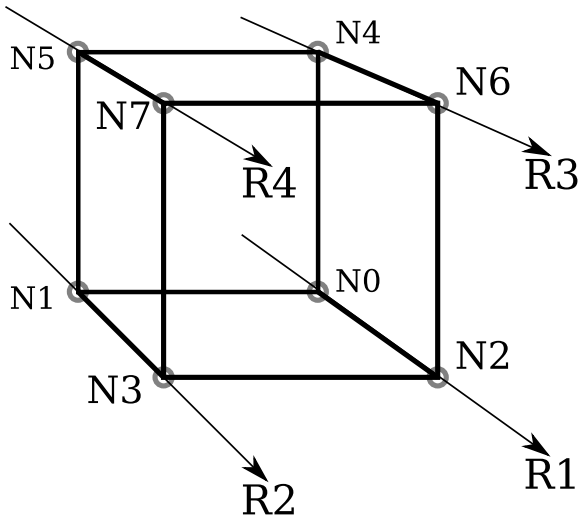

Figure 3.5: *Relationship of the probing rays (R1 through R4) to the nodes and faces of an element. The lower two rays (R1 and R2) belong to slice z, whereas the upper two rays (R3 and R4) belong to the subsequent slice at $z + 1$. From the observer's point of view, rays are processed from right to left. The nodes (indicated by gray circles) are arranged in a zigzag pattern where the first two nodes (N0 and N1) lie on the inside wall, and the next two nodes (N2 and N3) lie on the outside wall, whereby the connecting vectors $N0 \rightarrow N1$ and $N2 \rightarrow N3$ both point right-to-left. The same orientation is used for the nodes in the upper slice, N4 through N7.*

Tochnog can provide. In addition, the meshing module can produce an STL file, which is a tesselated description of the object's surface without any material properties. Many STL file viewers are available, and we used gmsh (`http://geuz.org/gmsh/`).

3.3 Results

3.3.1 Proof of Principle

To test the overall FEM chain from the initial volumetric image to the final visualization, we assumed that our phantom model (Figure 3.1) was a thin-walled, relatively elastic blood vessel being subjected to an internal pressure that corresponds to blood pressure acting on

the inner lumen of the vessel. We also assumed that the ends of the vessel would be fixated (fixation grooves in Figure 3.2). Initially, we used 60 probing rays per slice and one set of rays every slice for a total of 15,360 nodes and 7,620 elements. The ray values were selected such that the division of the number of rays by the 360 degrees in a circle would result in a whole number multiple of 360.

Using an in-plane pixel size of 2.3x 10^{-5} m, we subjected the internal nodes to a pressure of -16 kPa (approximately 120 mmHg). The negative sign indicates outward-directed pressure. Each end of the tube was fixed by labeling them as boundary nodes and restricting movement in the x-, y-, and z-directions. We selected a Poisson ratio of 0.45, and a density of 945 kg/m^3. Based on the average intensity values calculated, five material groups were generated possessing elastic moduli values of 55, 60, 75, 85, and 100 MPa which were assigned and chosen to make the construct relatively elastic. All of the mechanical property values were selected such that our model could be characterized as having a rubber-like material composition.

The visual results of subjecting our phantom model to our modeling chain, created with GiD, are presented in Figure3.6. The images depict the generated mesh onto the 3D tubular model, the classification of material groups present, as well as an exaggeration of the deformation behavior resulting from pressure being applied to the inner wall.

3.3.2 Mesh Convergence

We examined possible convergence behavior by varying (1) the number of nodal interpolation points between rays and (2) the ray distance in the z- (axial) direction. The material stress values from each analysis were chosen as the basis of the convergence observations. The results are presented in Table 1. In the case of (1), we used spline interpolation to generate additional nodes without changing the nodes of the non-interpolated reference mesh. For these experiments, 18 probing rays per slice were implemented because the angular in-

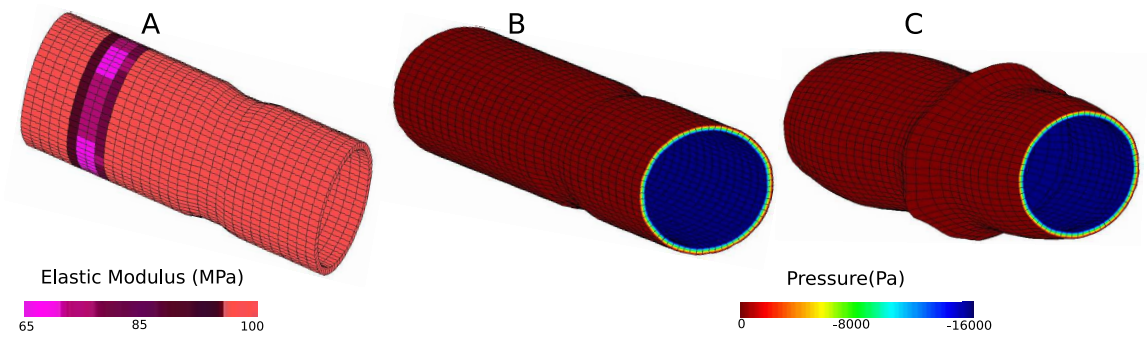

Figure 3.6: *GiD visualization results of phantom model subjected to homogeneous internal pressure. Composed of 15,360 nodes and 7,620 elements, the circular section where the wall is thinner as well as the circular section where image intensity is reflected in multiple material groups are visible (A). The internal pressure expansion and distribution behaviors of the thin wall area, lower intensity area, and ends of tube are highlighted (B). The exaggerated shape deformation resulting from the internal pressure expansion is also shown (C).*

crements of 20° provide adequate spacing for the addition of interpolated points between each pair of rays. Analyses were conducted on the tube that resulted in the addition of 0-8 nodes in between pairs of rays at each slice. The maximum stress values obtained at each interpolation level are similar, with maximum deviations of -2% and +3% from the mean value of 19.4 kPa. The axial anisotropy is the product of voxel anisotropy and the number of slices skipped for the mesh generation. For the geometry defined for the phantom, voxels are 4 times longer in the axial direction than in the x-y-plane. In the case of varying ray distance in the axial direction, two analyses were conducted at 18 rays without the addition of interpolated nodal points: (a) 18 rays per slice with one set of rays every four slices and (b) 18 rays every four slices, resulting in axial anisotropies of 4 and 16 respectively. We observed that number of nodes and the number of elements increase proportionally with the number of interpolated points; however, they decrease proportionally as the axial anisotropy increases. While the number of material groups are not affected by the number of rays and nodal and elemental amounts, increasing the axial anisotropy also decreases the number of material

Table 3.1: *Effects of axial and radial mesh refinement on the total number of nodes and elements, maximum stress values, and Number of Material Groups for the tubular phantom.*

Number of Rays	Axial Anisotropy	Interpolation	Nodes	Elements	Material Groups	Maximum Stress (kPa)	Time (min:sec)
18	4	0	4608	2286	5	19.0	0:22
18	4	1	9216	4572	5	19.1	0:45
18	4	2	13824	6858	5	19.2	1:10
18	4	3	18432	9144	5	19.8	1:32
18	4	4	23040	11430	5	19.0	1:55
18	4	5	27648	13716	5	19.2	2:24
18	4	6	32256	16002	5	20.0	2:48
18	4	7	36864	18288	5	19.6	3:17
18	4	8	41472	20574	5	19.9	3:45
18	16	0	1152	558	2	19.6	0:13

groups present. The internal wall pressure and expansion behavior remains unaffected by changes in nodal and elemental amounts. Lastly, the maximum stress values obtained at the higher axial anisotropy of 16 and in our proof-of-principle experiment using 60 rays (19.5 kPa) align with the values with presented in the table obtained using both a lower anisotropy of 4 and lower number of rays. An out-of-the-box installation of Tochnog does not have multiprocessor capabilities, but a typical file with approximately 16,000 nodes takes only a few minutes to compute on a single CPU core. The amount of computational time required increases approximately linearly with increasing nodal and elemental amounts and decreases accordingly when the axial anisotropy increases. However, Tochnog can be linked with libraries that are multi-thread-capable, and execution time is reduced correspondingly.

3.3.3 Evaluation of the Simulated Aneurysm Geometry

In addition to the cylindrical geometry of the first phantom, we examined the performance of our meshing module in cases of more complex tubular geometries. Aneurysms are balloon-like bulges that occur in blood vessels as the inner blood vessel wall weakens and eventually tears as a result of high blood pressure. Blood, and in some instances plaque, begins to pool in localized positions between these weakened vessel layers. Continued growth or expansion of the bulge increases the potential for the vessel to rupture, leading to a host of other complications including hemorrhaging, stroke, and in extreme cases, death.

We decided to simulate a fusiform aneurysm in which a bulging deformation is visible on both sides of the vessel (Figures 3.3 and 3.7A). In addition to its expanded section in the center with two different material properties, there is a height-dependent offset of the vessel geometry in the x-direction, resulting in a 15° angle between the medial axes of the two ends. The purpose of this kink is to prove that the meshing algorithm can accurately capture objects that are not fully cylindrical. With 60 probing rays per slice, the model is composed of a total of 15,360 nodes and 7,620 near-cuboid elements. The in-plane pixel size, pressure, Poisson ratio and density values were the same as outlined in Section 3.1. Based on the average intensity values calculated, eleven material groups were generated possessing elastic moduli values of 100, 105, 110, 115, 120, 125, 130, 135, 140, 145, and 150 Mpa. Elastic moduli with larger values are expected to be assigned within the bulging areas of the vessel. Different material groups occur when the element encloses different amounts of the two materials, an effect similar to partial-volume artifacts in computed tomography.

Figure 3.7B displays the material stress obtained from the Tochnog analysis. As expected, the highest material stress values are observed at the weakest areas of the inner vessel walls, where the bulging between layers is first observed. The areas of the vessel perpendicular to

the bulges are believed to also be subjected to increased stress due to the shape deformation (bulging) under high pressure and difference in material property values in comparison to the bulge.

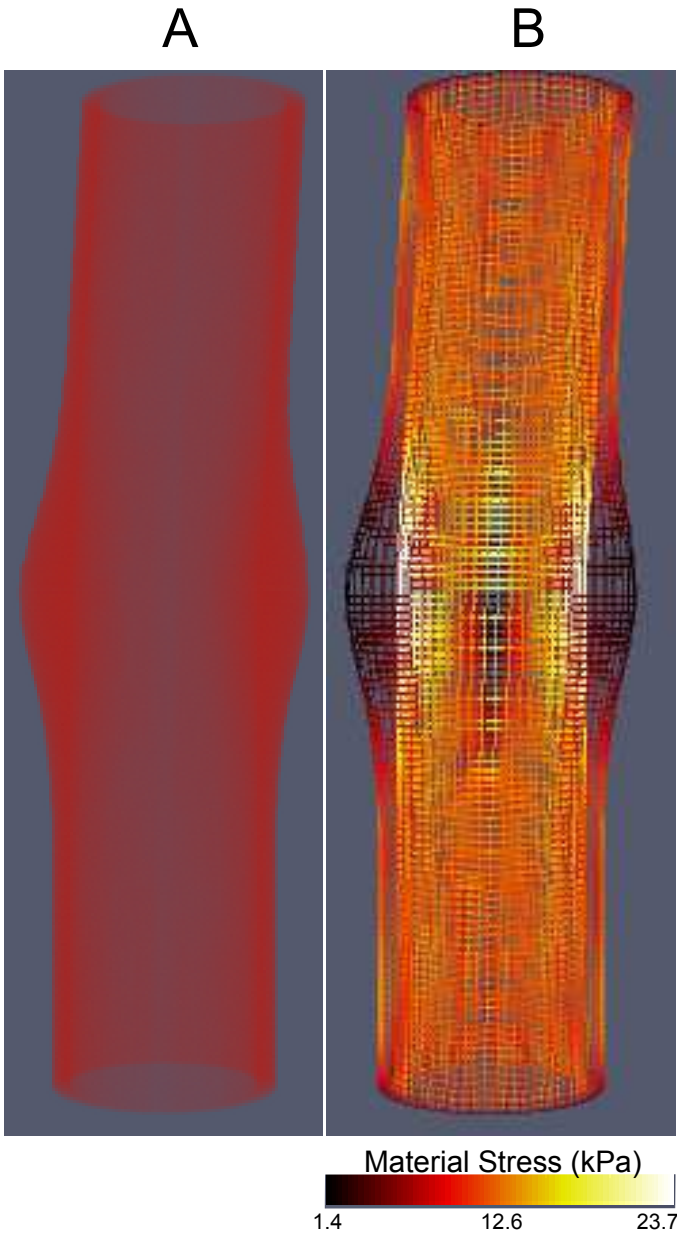

Figure 3.7: *3D Rendering of the aneurysm model for the second phantom. The model represents a fusiform aneurysm along a bent blood vessel with a 15° curve. The false-colored image values represent the material elasticity, in this case, light blue for the regular vessel wall and red for the more rigid plaque. The color scale bar is the same as in Figure 3.1.*

3.3.4 Visualization Software Comparison

The images presented in Figure 3.6 were constructed using GiD; however to adhere to our goal of presenting a completely open-source modeling chain, we compared the visualizations from GiD with those from Paraview and OpenDX. Paraview was found to provide visualizations comparable to GiD, whereby out-of-the-box visualization schemes provided immediate results with a low level of user input. A Paraview rendering example is presented in Figure 3.7. Conversely, OpenDX allowed to design complex visualizations with high flexibility. One sophisticated example is shown in Figure 3.8. However, OpenDX requires the generation of a visual program, and experience with OpenDX is a prerequisite. A unique feature of GiD is the ability to exaggerate deformation (prominently visible in Figure 3.6C). A comparable feature was not found in Paraview.

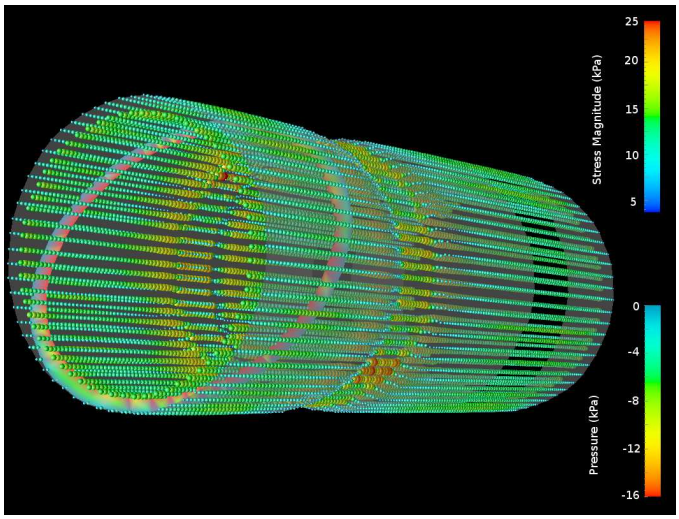

Figure 3.8: *Example 3D Rendering of the FEM simulation results with OpenDX. For this example, the inner and outer surface were rendered as gray, semi-transparent tubes. The magnitude of the shear stress tensor was superimposed for each node as a glyph, i.e, a small sphere where the size is proportional to the stress magnitude. The glyphs are false-colored with the magnitude for improved visual perception and the values representing these colors are displayed in the upper colorbar. In addition, a slanted ring is placed inside the vessel wall (near the left end of the tube) that displays the magnitude of the pressure by using the lower colorbar (negative values indicate outward-directed pressure).*

Lastly, we wanted to compare the performance of our meshing module to the mesh previously obtained by the closed-source medical image pre-processing software. We used Mimics to obtain the 3D mesh of our phantom model shown in Figure 3.1. This mesh was then imported into the commercial FEM software Algor for simulation and visualization purposes. The resulting mesh consisted of a mix of approximately 300,000 hexagonal, wedge, pyramid, and tetrahedral shaped elements and comprised approximately 100,000 nodes and 13,000 surfaces. Figure 3.9 displays the visual results obtained from the MIMICS meshing process. The simulation performed with Algor did not converge and did not produce any output.

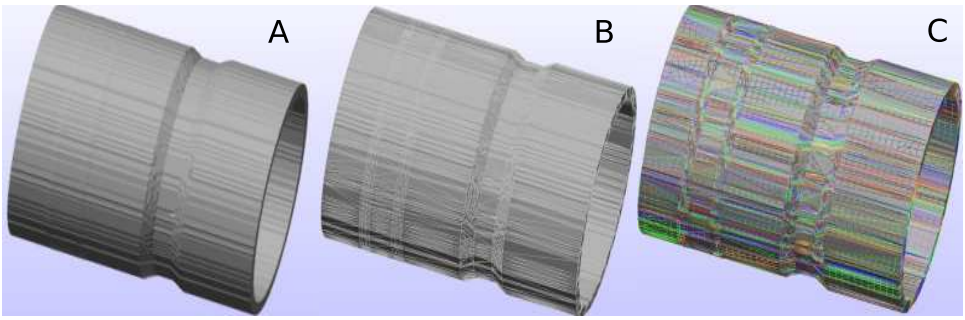

Figure 3.9: *Visualization results of phantom model subjected to open-source mesh module and closed-source image pre-processing software. The geometric rendering (A), the meshed model (B), and the colored identification of all the surfaces and elements the model is comprised of (C) are highlighted.*

3.4 Discussion

In this study, we presented a fully open-source toolchain for finite-element modeling. A number of reasons make such a toolchain an attractive alternative to commercial software packages. First and foremost, no license fees are incurred, and FEM analysis becomes possible on a low budget. Furthermore, the usability of the software for a specific purpose can be examined without obligation. The second fundamental advantage lies in the open-source

nature of the software, which means that the underlying program code can be examined or modified. In a classroom setting, where FEM is often taught with black-box software under an academic license, students can use open-source software to examine the numerical aspects of solving discrete partial differential equations. In a research setting, the exact algorithm that leads to a specific result can be determined, potential weaknesses identified, and the code amended. Access to the algorithm is particularly important when critical nonlinear cases are examined, such as turbulence or fracture. On the other hand, open-source software generally has no organized support. Rather, this type of software relies on community support, and software-related questions are usually resolved by peers in Internet forums. Software development is also driven by community efforts, although sometimes (as in the example of Paraview) a company supports development. The centerpiece of our toolchain, the FEM solver Tochnog, is offered both as a free version and as a commercial version with paid support and development (Tochnog Professional, Feat, The Netherlands).

If sufficient demand exists, community-driven software packages of similar functionality can frequently be found. The visualization toolkit VTK, for example, has led to the development of Paraview, VisIt (Lawrence Livermore National Laboratory, Livermore, CA), and Slicer (www.slicer.org), to name a few examples. Alternatives to Tochnog exist as well, for example, FreeFEM (www.freefem.org) and Elmer (elmer.sourceforge.net). However, we found no free alternative to MIMICS to create a mesh from a 3D medical image. This gap prompted us to develop a specialized meshing program for our research application of vascular constructs, and the implications of this gap are discussed in detail below.

3.4.1 Meshing Module

The most crucial step in all FEM chains that process medical images is the actual surface or volume parametrization. In simple terms, one could describe this step as converting a stack of pixels into an ordered set of nodes and elements. This step consists of two successive parts:

Image segmentation and the actual surface parametrization. Interestingly, few commercial software packages and no open-source packages exist[2] that perform this task. MIMICS by Materialise is arguably the most widely used mesh generation package for medical images. Our experiments with MIMICS led to meshes with a large number of irregular shapes, jagged edges and sharp drop off points. We observed that downstream software had difficulties processing such a mesh. Since we were unable to find the cause of those difficulties in spite of extensive help from Materialise support, we decided to favor the simplest possible geometry for our application and develop our own mesh extraction software with the key difference that, unlike MIMICS, our software will be restricted to a limited set of geometries.

A second notable difference to MIMICS is the ability of our module to automatically assign material parameters that correspond to varying image intensity values. Such a function does not exist in MIMICS, which assumes a single homogeneous material. In addition, the adjustment of probing ray density, spline-based surface smoothing, and axial anisotropy adjustment features of our meshing module do not have a known correspondence in MIMICS. These examples illustrate a typical trade-off between software aimed at solving as broad a range of problems as possible (MIMICS) and software aimed at solving specialized cases (our meshing module).

In comparison with other FEM software chains, the uniqueness of our approach lies in the completely open nature of all steps of mesh generation, solving of the equations, and visualization. Other published partly open-source chains require the generated model to be imported and analyzed with commercial closed-source software [22, 93], or, in other cases, the meshing algorithms examined require the use of commercial software to both pre-process

[2]This is no longer the case due to the development of the BioMesh3D (`http://www.sci.utah.edu`) and Cleaver [96] software.

the image slices into a 3D model and undergo FEM analysis and do not appear to be open-source [70, 89]. In addition, most of these have only been implemented on anatomical bone structures, which are by nature more easily segmented.

Our meshing software extracts geometry from the image and applies a highly regular grid of nodes to the inner and outer wall. In addition, the software is capable of applying material properties and boundary/loading conditions to the model. Using the probing ray principle, the module is able to automatically generate finite cuboids, trapezoids, or frustums in a Tochnog-ready input file. Through calculation of average intensities, we were able to distinguish between inhomogeneities along the model and assign unique material definitions on an element-by-element basis under the assumption that a relationship between image intensity and material properties exists [18–21]. This assumption has also been implemented in other FEM analyses of medical images [6, 22]. This relationship is often empirical and highly dependent on the imaging modality used. In some examples, notably bone imaged by computed tomography, a strict relationship between the CT value and bone mineral density exists. Optical modalities can relate scattering to the presence of collagen [123], which is fundamental to tissue elasticity. In other cases (e.g., T1 and T2 relaxation in MRI), the relationship is more tenuous and needs to be established beforehand in separate studies. For the proof of principle that is the focus of this study, we used somewhat arbitrary values that are common for many elastomers, and we are aware that the precise model of a tissue-engineered blood vessel requires additional research, for example, a study with optical tomography [126], where the where the blood vessel is subjected to pressure and its expansion measured.

In its present form, our meshing module is limited to approximately cylindrical, tubular objects. Concavities, for example, saccular aneurysms, would not be captured correctly by our module. Furthermore, the relatively coarse subdivision into one single element across the wall may cause some artificial rigidity. However, Tochnog performs automatic mesh

refinement based on the residuals of the governing equations. Furthermore, our algorithm can be refined in a very straightforward manner with multiple thresholds. Presently, a binary image is assumed with one object of approximate radial homogeneity. If this assumption leads to an unacceptable simplification, a node along the ray can be created whenever one of multiple thresholds is crossed. This process may generate new shapes, most notably, triangular prisms and hexahedrons, for which different element definitions exist in Tochnog.

A consistent observation of radial inhomogeneity was made, which becomes most prominently visible in Figure 6A. More specifically, larger stresses and larger apparent deformations were observed at angles of 45° , 135° , 225° , and 315° with respect to the x- and y-axes than coincident with the axes. This inhomogeneity is the consequence of the discretization of a cylindrical object on a polar grid in the phantom bitmap image: Parallel to any axis, the phantom is exactly 10 pixels (2.3×10^{-4} m) thick, whereas rays at odd multiples of 45° detect the inner and outer boundaries approximately 8 pixels (1.84×10^{-4} m) apart. This apparent inhomogeneity is correctly recognized by the meshing algorithm and correctly visualized by the toolchain. However, it should be noted that this apparent inhomogeneity indicates the critical role of the object discretization and points at a potential source of error for all FEM toolchains when objects are extracted from medical images with low resolution.

In spite of its relative inflexibility, we believe that the meshing module is useful in its source form, because it demonstrates in detail how a Tochnog input file is composed. Therefore, the meshing module may readily be modified with other extraction algorithms (such as, e.g., the marching cubes algorithm) to accommodate a larger variety of shapes. For example, a simple extension towards spherical objects can be envisioned if the probing rays are emitted radially in all directions from the centroid. We did not further pursue this path, because the ray-based algorithm proved to be sufficient for our application, and because generalized algorithms based on the marching-cubes algorithm exist [59,127–129]. By nature, familiarity

with computer programming and the C/C++ language is helpful in understanding the algorithms and methods employed in the meshing module when a modification of our meshing algorithm is desired.

3.4.2 Visualization Software Options

Both of the open-source visualization software options evaluated are fully capable of providing suitable visualizations of FEM output data. Paraview is similarly intuitive as its commercial competitor, GiD. However, Paraview allowed us to obtain visualizations of the magnitude for stress, strain, and deformation values. This feature was not found in GiD; vector values are presented and must be calculated in order to obtain magnitude values. Conversely, Paraview does not allow us to to visualize the varying material groups along the inhomogeneous section containing lower image intensity values. In addition, we were unable to visualize the exaggerated expansion behavior of the vessel when subjected to internal pressure as presented in Figure 3.6. OpenDX is advantageous in the sense that it allowed us to superimpose multiple aspects of the simulation onto one image. For instance, in the visualization presented in Figure 3.8, we were able to view the volumetric rendering of the tube (in light gray), the stress magnitude (as points in space), the pressure distribution throughout the tube (diagonal circle at left end of tube), and the inhomogenous section of the tube containing lower image intensity values (black). Tochnog output files created for OpenDX are typically very large, because all data for each point in time is recorded for processing in OpenDX. To extract specific data from the Tochnog output file, for example, the shear stress magnitude, specific visual programs must be created. Unlike Paraview and GiD, which are very intuitive, creation of a visualization with OpenDX requires experience with OpenDX. The key advantage of OpenDX is the unmatched flexibility with which visualizations can be designed.

3.5 Conclusion

The open-source philosophy allows the presented FEM chain to be available for use, modifications, improvements, and free distribution by our biomedical research peers. Our proposed modeling chain exhibits versatility as it can be applied to any tubular biomedical object that has been subjected to biomedical imaging, including medical device tubing, stents, or long bones. The meshing module is our contribution to the open-source community, and it can be freely downloaded at `http://haidekker.org/cimage/`. In this study, our module was applied only to phantom models of tubular constructs, but we have provided the framework which allows us to subject tissue-engineered vascular grafts to FEM analysis to examine biomechanical behavior. Future research will revolve around continuing to apply our modeling chain to objects of varying geometrical shapes and types of materials to test the specificity and efficiency of the probing ray principle.

Chapter 4

NONINVASIVE INTRACRANIAL PRESSURE ASSESSMENT IN CANINES VIA BIOMECHANICAL RESPONSE BEHAVIOR, MEDICAL IMAGING, AND FINITE ELEMENT ANALYSIS: A PILOT STUDY[1]

[1]Madison, A.M., A. Sharma, and M.A. Haidekker. 2013. To be submitted to *Veterinary Radiology and Ultrasound*

Abstract

Intracranial pressure (ICP) represents the pressure exerted on the brain tissue by cerebrospinal fluid (CSF) within the brain cavity, a fixed volume compartment in a state of equilibrium. Governed by the basic principle of the Monro-Kellie doctrine, a volume increase in either the brain parenchyma, blood, or CSF must be compensated by the decrease in another component. Maintenance of adequate cerebral perfusion pressure (CPP) is essential in the successful management of patients with intracranial disease. Increases in ICP can cause detrimental reductions in CPP and irreversible brain damage which could be fatal if left untreated. Invasive measurement techniques involve surgical procedures associated with high risk of infections. Therefore rapid, noninvasive, and accurate assessment of ICP is important for guiding therapeutic decisions. Finite-element modeling (FEM) enjoys growing popularity in the study of the biomechanical behavior of biological tissues. In noninvasive studies, imaging methods (e.g. computed tomography (CT) and magnetic resonance (MR)) are used to obtain the anatomical geometry. We propose a novel approach which implements FEM and medical imaging to approximate ICP in settings where conventional monitoring techniques are unavailable. Building on the relationships between the CPP and ICP, we observe patterns of non-linear biomechanical behavior in biphasic analysis of normal and abnormal canine brains. Our method presents a framework which can use material response behavior resulting from increased ICP as a diagnostic, treatment, or preventative method to assess or classify levels of brain injury in clinical veterinary settings noninvasively. The feasibility of applying FEM to brain geometry obtained MRI data from clinical patients for determining differences in ICP is successfully demonstrated in this study.

4.1 Introduction

The total volume in the canine cranial cavity is composed of brain tissue or parenchyma (1400 ml $\approx$ 80%), arterial and venous blood (150ml $\approx$ 10%), and the cerebrospinal fluid (CSF) (150ml $\approx$ 10%). Governed by the basic principle of the Monro-Kellie doctrine [130, 131], a volume increase in either the brain parenchyma, blood, or CSF must be compensated by the decrease in another component. The intracranial pressure (ICP) represents the pressure within this volume and is exerted on the brain tissue by CSF. A volume increase in any of the three brain components raises the ICP. The standard classifications and values of ICP are presented in Table 4.1.

Table 4.1: *General classifications of ICP levels and corresponding value ranges.*

ICP Value	**Ranges** (mmHg)
Normal	5-12
Increased	> 17
Sustained	> 17 for longer than 5 mins
Severe	> 35

Maintenance of brain function is dependent on brain blood flow and is affected regardless of whether the increased ICP is a result of brain tissue, blood, or CSF volume increase. This is attributed to fluctuations in the cerebral perfusion pressure (CPP), a conceptual pressure balance between the systemic mean arterial pressure (MAP) and ICP, defined by the equation

$$CPP = MAP - ICP \tag{4.1}$$

and the mean arterial pressure (MAP) is:

$$MAP = \frac{2BP_d + BP_s}{3} \tag{4.2}$$

where

BP_d**:** Diastolic blood pressure which equates to the minimum pressure required such that the heart's left ventricle can sustain its ejection phase (mmHg)

BP_s**:** Systolic blood pressure representative of both the left ventricle's ejection pressure and arterial system elasticity (mmHg).

Instances of increased ICP, especially in the sustained and severe states, is a serious medical problem which can lead to brain and spinal cord damage due to pressure on important brain structures and restriction of blood flow to brain. Complications of elevated ICP can lead to permanent neurological problems such as, seizure, stroke, and, in extreme cases, death.

In clinical diagnostic and treatment settings, ICP values are obtained via invasive intracranial measurements from the epidural, intraparenchymal, intraventricular, subarachnoidal, and subdural locations within the skull cavity by means of a surgically inserted pressure sensor. However, insertion of the devices can only be performed by highly-trained medical specialists. This restricts the monitoring and measuring of ICP within specialized critical care hospital units. Additionally, risks of infection, hemorrhage, as well as iatrogenic brain and spinal cord damage are associated with these procedures [132].

A host of noninvasive techniques have been proposed in attempts to avoid or improve invasive ICP measurement [132–136]. In this instance, medical imaging modalities are invaluable assets since they permit the visually qualitative identification and assessment of

phenomena that are known contributors to increased ICP. These non-surgical, low-infection monitoring procedures are generally classified under two categories based on how the ICP value is determined [133]:

- ICP estimation based on anatomical features and functional characteristics of incranial constituents
- ICP inference based on anatomical features and functional characteristics of organs and tissues that are not contained within the cranium, but are associated or connected to the intracranial constituents

however no noninvasive method has been successfully integrated into clinical practices of ICP monitoring. This can be attributed to uncertainties related to the accuracy and validity or the measurements obtained.

Biomechanical models could assist in the improved noninvasive estimation of ICP. The earliest simulations of the cranial cavities used simplified mathematical models to derive solutions which expressed the non-linear relationships between pressure, volume, volume-pressure responses, and compliance [137–143]. Finite-element modeling (FEM) enjoys growing popularity in the study of the biomechanical behavior of biological tissues. In noninvasive studies, imaging methods, such as computed tomography (CT) or magnetic resonance (MR), are used to obtain the anatomical geometry. Non-linear biphasic 3D FEM models incorporating brain parenchyma and ventricular geometry obtained from medical imageshave been implemented to analyze the deformation and constitutive behavior of brain tissue [5,115,144,145], however, no reported method seeks to assess, derive, or estimate an ICP level based on biomechanical response behavior.

We present a noninvasive method to approximate ICP and other critical values using finite element analysis, mathematical relationships between pressure, volume, stress, strain, and 3D brain models obtained from medical imaging data. Building on the relationships between the

cerebral perfusion pressure (CPP) and ICP, we observe patterns of non-linear biomechanical behavior in biphasic analysis of normal and abnormal canine brains. Our method presents a framework which can use material response behavior resulting from increased ICP as a diagnostic, treatment, or preventative method to assess or classify levels of brain injury in clinical veterinary settings noninvasively. We also intend to be observant of the brain/skull interface locations since the pressure of CSF in the ventricles and subarachnoid section corresponds to ICP. As the first research of its kind, our goal is to establish a proof of principle framework that will eventually implement noninvasive patient-specific ICP assessment based on biomechanical property changes obtained from utilizing medical imaging data and FEA in clinical settings.

4.2 Material and Methods

The experimental process utilized in the finite element analysis of brain geometry obtain from medical images is outlined in Figure 4.1.

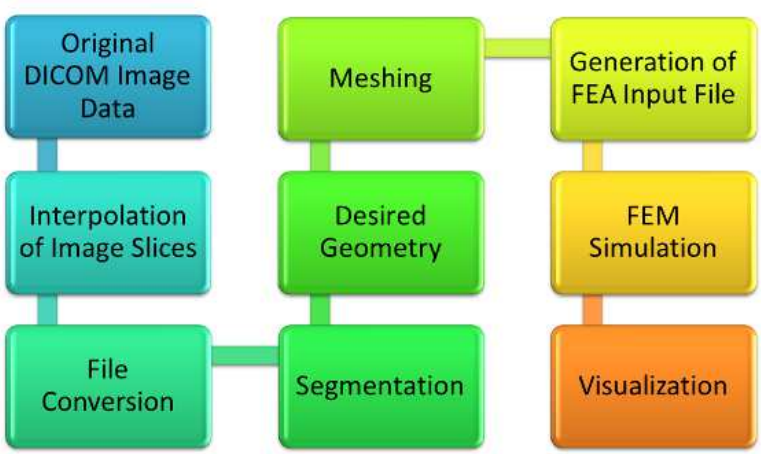

Figure 4.1: *Schematic of the experimental process implemented to conduct the research outlined in this work.*

4.2.1 Brain Geometry

Image Acquisition

The process of preparing a geometrical description of the anatomy from medical images to undergo FEA began with a volumetric image composed of cross-sectional slices obtained via magnetic resonance imaging (MRI). Two-dimensional sagittal T2 spin-echo sequence MR (General Electric 3T, 16 channel fixed site Signa HDx Magnet) brain data with slice thickness of 2.00 or 3.00mm and 2.20 or 3.20mm spacing between slices was provided courtesy of The University of Georgia School of Veterinary Medicine Five canine patient scenarios were considered for this study and the conditions of each are listed in Table 4.2 and are visually represented in Figure 4.2.

Table 4.2: *Medical conditions of subjects used in this study.*

Subject	**Image Plane Dimension** (mm)	**Medical Condition**
		Normal
1	512 x 512 x 23	No abnormalities detected
		Abnormal
2	512 x 512 x 9	Congenital occipital malformation syndrome with hydrocephalus and syringohydromyelia
3	512 x 512 x 23	Cystic meningioma in ventral aspect of cranial cavity
4	512 x 512 x 23	Cerebellar mass with cerebellar herniation
5	512 x 512 x 25	Meningioma in rostral cranial cavity

Image slices in each case were interpolated such that new two slices were added between every two original data slices using quantitative image analysis software (Crystal Image [28]). This procedure increased the number of acquisitions (slices) and decreased the spacing between slices for each patient case and resulted in updated image dimensions of 512 x 512 x 58, 512 x 512 x 25 , 512 x 512 x 67, 512 x 512 x 67, and 512 x 512 x 52mm for Subjects 1-5 respectively. Each slice from the newly generated data set in all patient cases was exported from Crystal Image as an individual image and subsequently converted from a 3D image into to a multiple-image tiff file by ImageJ (US National Institutes of Health, Bethesda, Maryland).

Image Pre-Processing: Segmentation and Meshing

Medical images must be pre-processed, or converted into a CAD-like 3D solid model. This involves the segmentation, or separation of the geometry of interest from the background. Brain parcheyma and CSF fluid were extracted from the MRI data with the Seg3D (Scientific Computing and Imaging Institute, University of Utah (`www.seg3d.org`)) volumetric image segmentation and visualization package. Manual segmentation was assisted by repetitive applications of intensity based thresholding, Otsu's Threshold, Boolean operators, and Arithmetic filters and resulted in the separation of the following material types which can be observed in Figure 4.2 (B, D, F, H, and J):

- Brain Tissue
  - Grey Matter
  - White Matter
  - Tumor (In Subjects 4&5)
  - Cyst (In Subject 3)

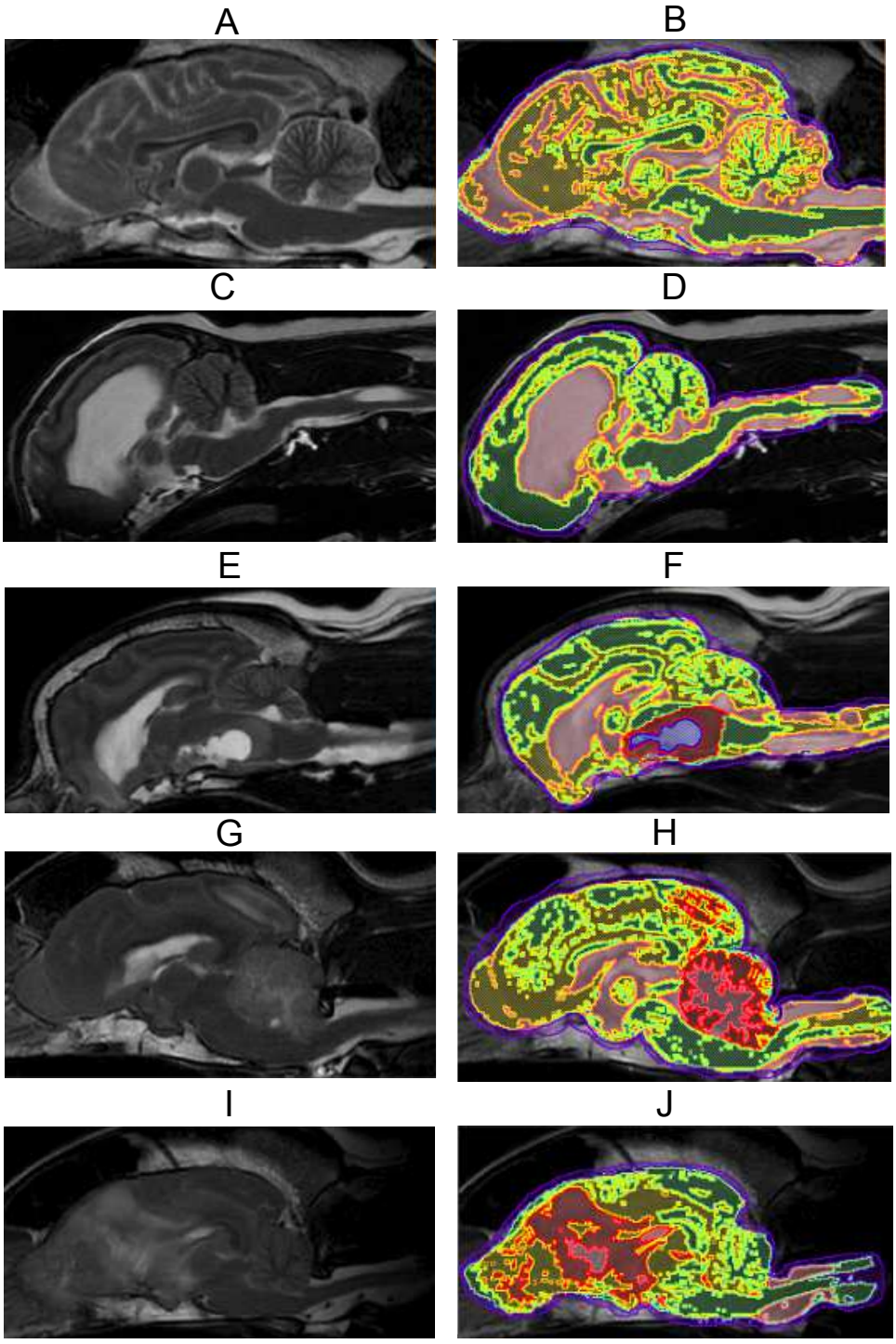

Figure 4.2: *Mid-sagittal slices of Subjects 1-5 at image acquisition (A,C, E, G, and I respectively). Segmentation masks (B, D, F, H, and J) are created to distinguish material types in all scenarios, and consist of brain tissue (yellow and green), CSF (pink), and subarachnoid space (purple) (B, D, F, H, J) . Additionally, the presence of abnorma brain tissue or mass (red) is catergorized in Subjects 3-5 (F, H, J) along with abnormal fluid (blue) in Subject 3 (F).*

- Cerebrospinal Fluid (CSF)
- Cystic Fluid (In Subject 3)
- Sub-Arachnoid Space (SAS)

Once the geometry was extracted, it was divided into elements in a procedure known as meshing because it appears as if the boundaries of the finite elements form a mesh-like structure that encompasses the segmented object. The stack of image slices containing the

segmented 3D brain geometry was exported and converted into a tetrahedral volume mesh by BioMesh3D (Scientific Computing and Imaging Institute, University of Utah (`www.biomesh3d.org`)) and Cleaver software [96]. Figure 4.3 displays the mesh of the normal brain model.

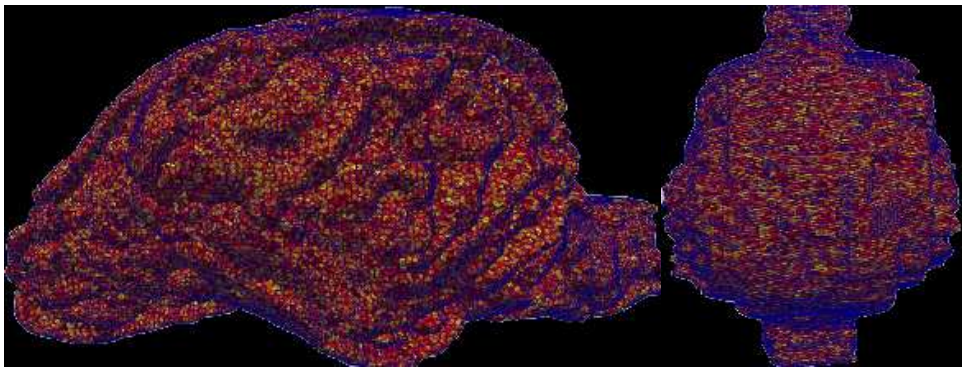

Figure 4.3: *Lateral (left) and Superior (right) views of normal brain model mesh.*

4.2.2 Finite Element Analysis

Tochnog (`tochnog.sourceforge.net`), the FE software chosen to the conduct the analysis of our models, is an implicit/explicit finite element solver for a number of different problems including linear and nonlinear solids, fluids, gases (Navier-Stokes), as well as biphasic interaction between solids and fluids. The output file provided by the meshing software was used in the generation of a Tochnog input/control file, where the analysis type, loading and

boundary conditions, and material property definitions have been specified. The input file created for the simulation of each subject can be found in the Appendix. Control statements within the Tochnog input file determine its output format. At the end of the simulation run, the output file contained the node locations, node pressure, temperature, stress, strain, and velocity for each time step. Tochnog generates output files that are directly compatible with various post-processing and visualization packages.

Simulation of Brain Parenchyma and CSF

The brain was modeled as a non-linear, isotropic, elastic deformable porous media, or biphasic material composed of a mixture of both incompressible permeable parenchyma and non-viscous CSF as the interstitial fluid. Tochnog is capable of automatically generating and calculating friction and other related contact forces in fluid-structure interaction scenarios. This model is ideal in the simulation of materials exhibiting flow-dependent viscoelastic behavior resulting from the frictional interactions of the fluid and solid. Both materials are incompressible with potential for the mixture's volume to change as the CSF enters and leaves the brain tissue. The amount of CSF present varies between image slice, patient, and condition of brain (normal vs. abnormal). In Tochnog, the strain energy (Total Potential Energy, TPE) method for material deformation containing both distortional (deviatoric) and volumetric components of the deformation gradient is defined by:

$$\rho v_i = \frac{\partial \sigma_{ij}}{\partial x_{ij}} + (1 - \beta T)\, \rho g_i - d\frac{\partial v_i}{\partial x_i} + \mathbf{f_i} \tag{4.3}$$

where

ρ: Density of the material

v_i: Material velocity in the i-direction

σ_{ij}: Material stress matrix

x: Space coordinate

β: Material expansion volume, or measurement of the volume change of a fluid or solid in response to a change in pressure or stress

T: Temperature of the material (negligible in our case)

g_i: Gravity

d: Damping coefficient of material (negligible since CSF is non-viscous)

$\mathbf{f_i}$: The pressure tensor source (In our case, a pressure was applied at all nodes within the CSF.)

Boundary/ Loading Conditions

The geometrical approximation provided by the mesh was then supplemented with additional information. For example, this includes specifying if the model is static or rotating, whether or not to account for fluid flow or temperature factors, or whether any forces are acting on, against, and possibly exerted from the model. The following conditions were implemented:

- All nodes of the SAS were fixed in the x, y and z directions to mimic its connectivity to the skull.
- All nodes of CSF were fixed in the x, y and z directions because we are not interested in the flow or movement of CSF, just the location of CSF in each model at the time of image acquisition.
- A constant pressure was applied to all nodes within the CSF since the pressure in CSF is the ICP. This pressure was assigned a negative value to simulate a radial force directed outward from the CSF and onto surrounding brain tissue (Figure 4.4). Normal

ICP values in dogs have been estimated to range in values from 0.67-1.60 kPa (5-12 mmHg), therefore we assigned a initial pressure gradient ICP value of 1.13 kPa, which is the average normal pressure value in dogs. Reported research data has suggested that mortality increases approximately 20% for each 10 mmHg (1.33 kPa) decrease in CPP [146, 147]. Low values of CPP can result in brain oxygen level depletion, and this lack of perfusion in conjunction with increased ICP is anticipated to affect the biomechanical behavior within both the brain parenchyma and CSF.

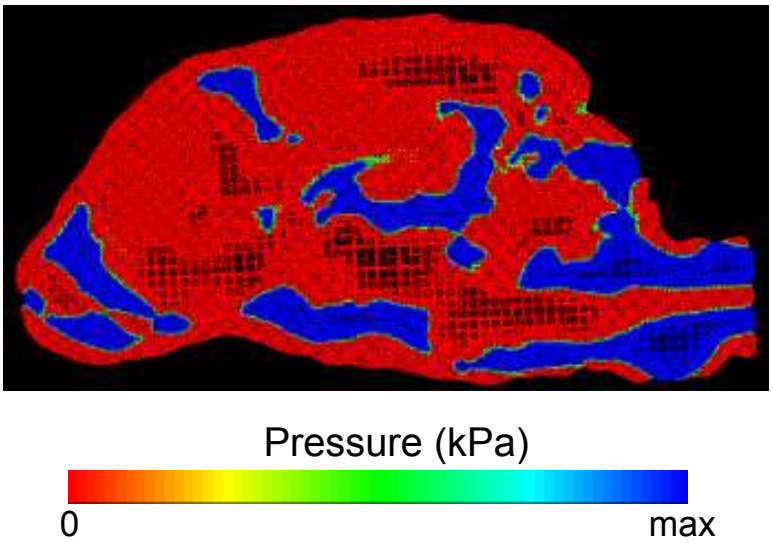

Figure 4.4: *Pressure distribution behavior resulting from pressure exerted by CSF. Locations of high pressure application (blue) correspond to CSF and boundaries where brain tissue and CSF are in direct contact with each other. The remaining brain tissue and subarachnoid space are regions of low pressure activity (red) since no pressure is being directly applied. Intermediate colors correspond the pressure gradient between high and low pressure regions within the brain tissue.*

Material Properties

Property characteristic values were then assigned to the model in order to numerically distinguish it as bone, soft tissue, fluid, or a combination of materials from an engineering aspect in terms of strength, elasticity, and durability. The model's material composition

is significant in this instance because the behavior exhibited is dependent on the type of material under examination. The material property input parameters and units for both the normal and abnormal brain models are provided in Table 4.3. All material properties for the subarachnoid space (SAS) were estimated for simulation purposes.

Table 4.3: *Material property parameters used in FE simulation.*

Property (Units)	**Brain**	**CSF**	**SAS**	**Reference**
Density, ρ $\left(\frac{kg}{m^3}\right)$	1.04 x 10^3	1.00 x 10^3	1.04 x 10^3	[148]
Bulk Modulus, K (kPa)	–	2.19 x 10^6	–	[149]
Poisson ratio	4.85 x 10^{-1}	4.99 x 10^{-1}	4.50 x 10^{-1}	[150–153]
Compressibility, β (kPa^{-1})	–	4.57 x 10^{-7}	–	$\beta = \frac{1}{K}$
Elastic Modulus, E (kPa)	558.1	–	650	[150–153]

4.2.3 Visualization

The output or visualization step presents the solutions and behavioral effects (i.e. areas of high stress, strain, and pressure and/or shape distortion due to factors such as: bending and twisting of the brain stem, swelling and hemorrhaging in brain tissue, blood clotting, obstruction, constriction, dilation, or compression of vessels and ventricles within the brain, or skull impact and fracture due to traumatic injury) at all points on the model resulting from the conditions imposed. These results are displayed in the original geometry with one specific biomechanical property (i.e. stress magnitude) superimposed in a false-color scheme based on a ranking scale of minimum to maximum numerical values, as well as tabular charts or line plots of quantitative data. The Paraview (Kitware, Inc., Clifton Park, NY) software allow the visualization of the data output by Tochnog. Paraview accepts many input formats,

and we used the VTK (visualization toolkit) format to exchange data between Tochnog and Paraview. In this work, the geometry of the mid-line sagittal slice is used to display the visualization results.

4.3 Results

4.3.1 Biomechanical Behavior Due to Fluctuations in CPP and ICP

Mechanical stress (σ), in engineering terms, refers to an average intensity level of internal forces and in solids generalizes the concept of pressure in a fluid. Mathematically defined as the force per unit area or

$$\sigma = \frac{P}{A}. \tag{4.4}$$

The strain (ϵ), or deformation of a solid due to stress is defined as

$$\epsilon = \frac{\Delta V}{V} \tag{4.5}$$

and is related to stress by

$$\sigma = E\epsilon \tag{4.6}$$

where E is the elastic modulus and is an expression of material elasticity.

In our initial experiment, we wanted to analyze changes in the biomechanical properties of brain tissue in instances of increased ICP. Therefore, we used the normal brain model to simulate an increasing ICP scenario in which there is also decreasing cerebral perfusion

pressure. FEM analysis was conducted to observe the changes in stress and strain as the CPP decreases in increments of 10mmHg ($\approx$ 1.33 kPa). Because this is also indicative of increased ICP, the ICP value is increased by approximately 1.33 kPa in each simulation to see how this affects the stress and strain distribution behaviors in the model. This incremental pressure increase value was based on two assumptions: (1) a constant MAP of 90mmHg ($\approx$ 12 kPa) and (2) the ICP increases the more CPP deviates from MAP due the mathematical relationship evident in Equation 4.1. Table 4.4 and Figure4.5 list the changes in maximum stress and strain values as ICP increases and CPP decreases.

Table 4.4: *Changes in stress and strain due to decreased CPP or increased ICP.*

CPP (kPa)	**ICP** (kPa)	$\sigma_{\mathbf{max}}$ (kPa)	$\epsilon_{\mathbf{max}}$
10.87	1.13	1.6	0.2
9.33	2.67	3.7	0.48
8.00	4.00	5.5	0.72
6.67	5.33	7.4	0.96
5.33	6.67	9.2	1.2

4.3.2 Biomechanical Behavior: Normal vs Abnormal Brain

Next, we wanted to observe how the presence of brain abnormalities and disorders at varying levels of severity affect the biomechanical behavior within the brain cavity. Therefore, the experiments from the previous sections were repeated using the abnormal brain subjects from Table 4.2 and Figure 4.2. The results were then compared to those obtained from the normal brain simulation. Tumor or abnormal masses of tissue are believed to less elastic than normal tissue, so the elastic modulus for this material was assigned a value of 1150

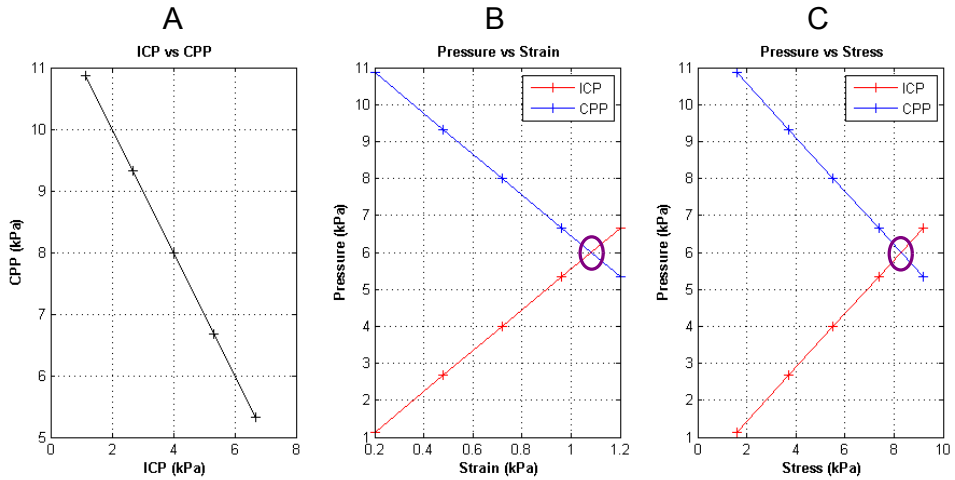

Figure 4.5: *Relationships of CPP, ICP, stress, and strain in normal brain model. As a result of the mathematical relationship expressed in Equation 4.1, CPP decreases as ICP increases (A). The changes in respect stress and strain as a result of decreasing CPP or increasing ICP are linear. Intersection of the CPP and ICP plots (purple) (B & C) represents the pressure value in where CPP=ICP, and is indicative of a MAP of zero. At a MAP of zero, there is minimal to no blood flow to the brain.*

kPa, which is twice the value of the normal parenchyma. Additionally, the Poisson ratio of 0.35 differs from that of the normal brain tissue. In the case of Subject 3, the cystic fluid was assumed to be more compressible than the CSF, and was assigned a Poisson ration of 0.47 in addition to a compressibility(β) of 9.14 x $10^{-7}(kPa^{-1})$ corresponding to twice the value of the CSF. Although more compressible than CSF, the cystic fluid is still considered relatively incompressible since its value for β is very close to zero. The maximum stress and strain values for each patient are presented in Table 4.5, while Figure 4.6 provides FEA visualizations of the stress and strain distribution behavior of each model when subjected to normal pressure (1.13 kPa).

Table 4.5: *Changes in biomechanical behavior due to increasing ICP values in research subjects.*

	Subject									
	1		**2**		**3**		**4**		**5**	
ICP (kPa)	σ_{max} (kPa)	ϵ_{max}	σ_{max} (kPa)	ϵ_{max}	σ_{max} (kPa)	ϵ_{max}	σ_{max} (kPa)	ϵ_{max}	σ_{max} (kPa)	ϵ_{max}
1.13	1.6	0.2	1.7	0.18	1.8	0.33	1.8	0.49	1.7	0.41
%Diff	–	–	**6.06**	**10.53**	**11.76**	**49.06**	**11.76**	**84.06**	**6.06**	**68.85**
2.67	3.7	0.48	4.1	0.43	4.2	0.78	4.2	1.2	3.9	1.0
%Diff	–	–	**10.26**	**10.99**	**12.66**	**47.62**	**12.664**	**85.71**	**5.26**	**70.27**
4.00	5.5	0.72	6.1	0.65	6.3	1.2	6.3	1.7	5.9	1.5
%Diff	–	–	**10.34**	**10.22**	**13.56**	**50.00**	**13.56**	**80.99**	**7.02**	**70.27**
5.33	7.4	0.96	8.1	0.86	8.4	1.6	8.4	2.3	7.8	.2.0
%Diff	–	–	**9.03**	**10.99**	**12.66**	**50.00**	**12.66**	**82.21**	**5.26**	**70.27**
6.67	9.2	1.2	10	1.1	11	2.0	10	2.9	9.8	2.5
%Diff	–	–	**8.33**	**8.70**	**17.82**	**50.00**	**8.33**	**82.93**	**6.32**	**70.27**

4.3.3 Identification of Critical Values

The brain responds physiologically to episodes of raised ICP and the response can be classified in four levels, each of which are highlighted in Table 4.6. The compensation behavior evident in Levels I and II are a result of the brain's auto-regulatory mechanism that attempts to maintain a steady, equilibrium CPP. Prolonged or steep, rapid increases of ICP overwhelm the compensatory responses and lead to severe decompensation observed in the

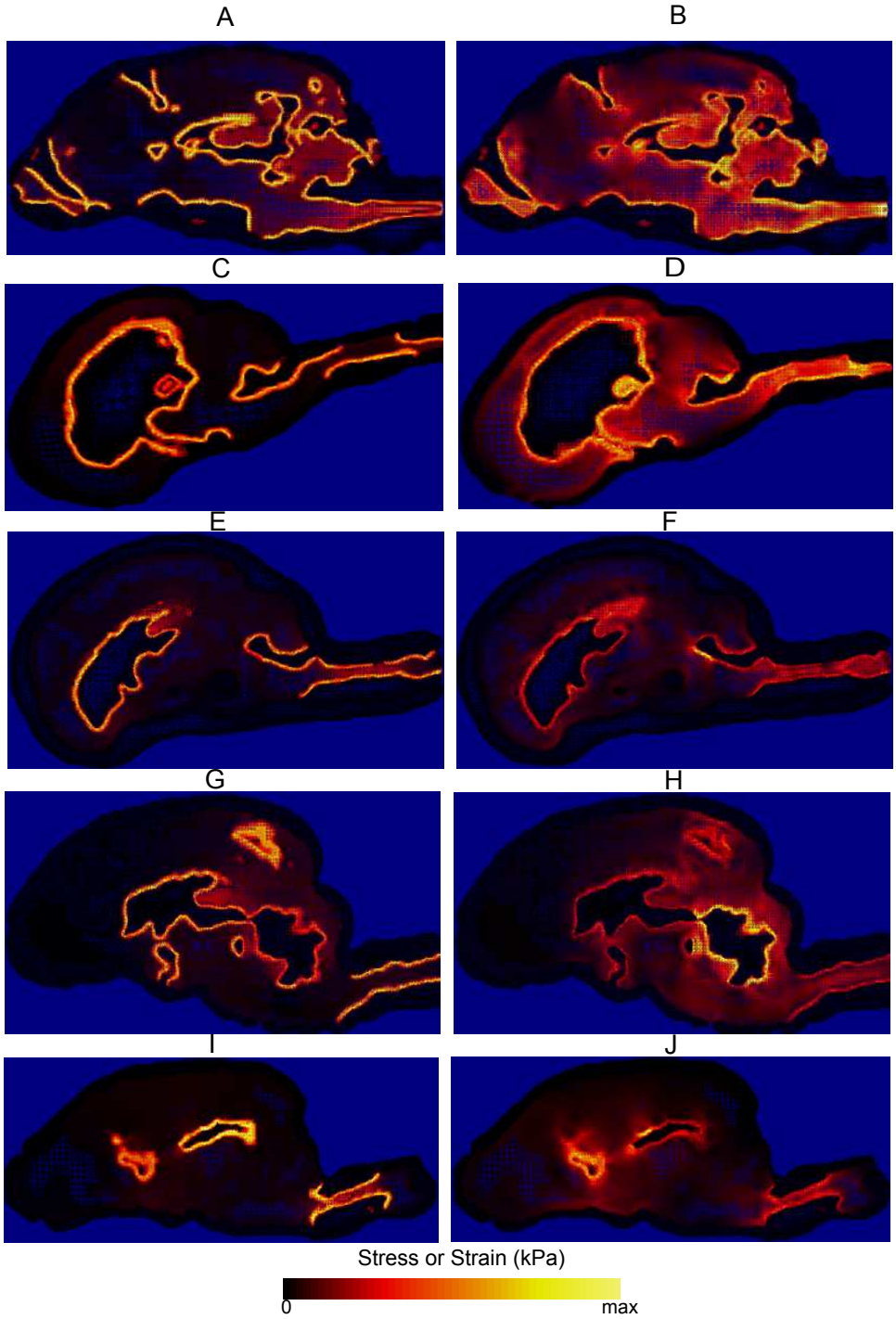

Figure 4.6: *Distribution of biomechanical stress (A,C, E, G, and I)and strain (B, D, F, H, and J) response behavior in Subjects 1-5 based on FE simulation at normal pressure (1.13 kPa). Based on the color scale, yellow represents the locations of maximum stress and strain and correspond to boundaries between tissue and fluid.*

Table 4.6: *Characterization behavior at various stages of increased ICP.*

ICP Level	**Behavior**
Compensation	
Level I	- Reduction in amount of CSF by increasing reabsorption rate - Vasoconstriction
Level II	Increase blood pressure through systemic constriction of arteries in efforts to overcome the high ICP and ultimately increase the CPP
Decompensation	
Level III	- Severe depletion of oxygen in brain - Increased blood flow in brain
Level IV	Herniation or displacement of brain tissue, blood vessels, and CSF through the opening at base of the skull

third and fourth levels and failure to maintain adequate CPP. Permanent brain injury and death are nearly inevitable at the onset of herniation experienced in Level IV ICP, therefore effective monitoring and treatment strategies are required to prevent this occurrence.

Figure 4.7 displays the general pressure-volume behavior within the skull cavity, measure of compliance, and the relationship between compliance and the compensation and decompensation levels of ICP. The non-linear relationship allows the determination of the compliance, or measure of volume distensibility for all constituents housed in the cranial cavity [137, 138]. The compliance coefficient (β), given as

$$\beta = \frac{\Delta V}{\Delta P} \tag{4.7}$$

is the reciprocal slope of the curve whose value is dependent on which cranial cavity component's volume is increasing. Steeper slopes are observed in blood and CSF in comparison to brain tissue. In addition, the onset of the curve's non-linearity corresponds to the decompensation levels in which ICP is high and the brain's auto-regulation behavior ceases.

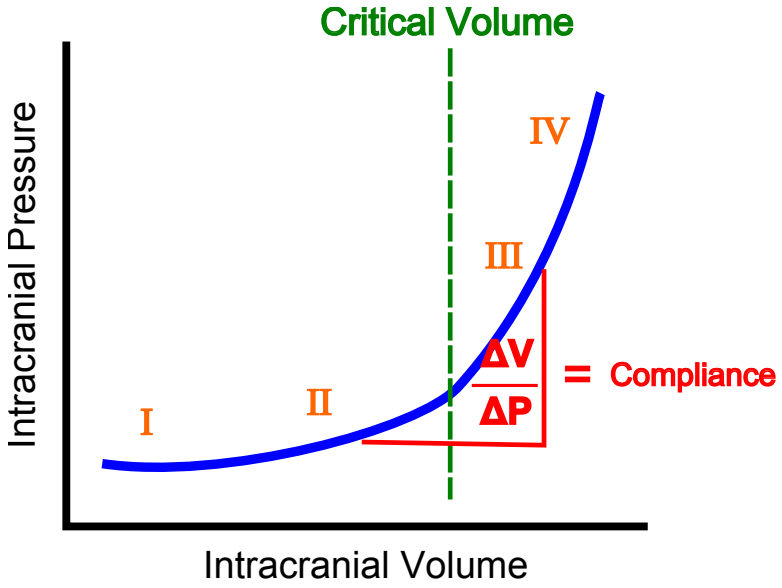

Figure 4.7: *Intracranial volume-pressure curve dynamics. The volume-pressure relation is linear in Levels I and II due to the brain's auto regulation compensation mechanisms to maintain equilibrium. These mechanisms become overwhelmed at the critical volume resulting in a non-linear relationship. Decompensation responses observed at Levels III and IV are the result of significant increases of intracranial pressure. The compliance coefficient* $\frac{\Delta V}{\Delta P}$*, is the reciprocal slope of the pressure-volume curve and measures volume distensibility for all constituents housed in the cranial cavity.*

In the attempt to identify the critical values of ICP, stress and strain that indicate onset of non-linearity or decompensation, the maximum stress and strain values at each pressure were plotted for each model (Figure 4.8). Here, changes in deformation patterns leading up to a strain value of 1 can be observed, such that they can be used as a standard for estimation or approximation of ICP, and in extreme cases herniation, in clinical settings using medical images. A strain value of one indicates that ΔV and V are equal, or that the change in volume is equal to the original volume. This tells us that brain has expanded to twice its original volume, which we believe is more than what the brain cavity permits. Using this information we can approximate the critical values at which herniation occurs. Each subject case was therefore be given a severity ranking based on the pressure at herniation.

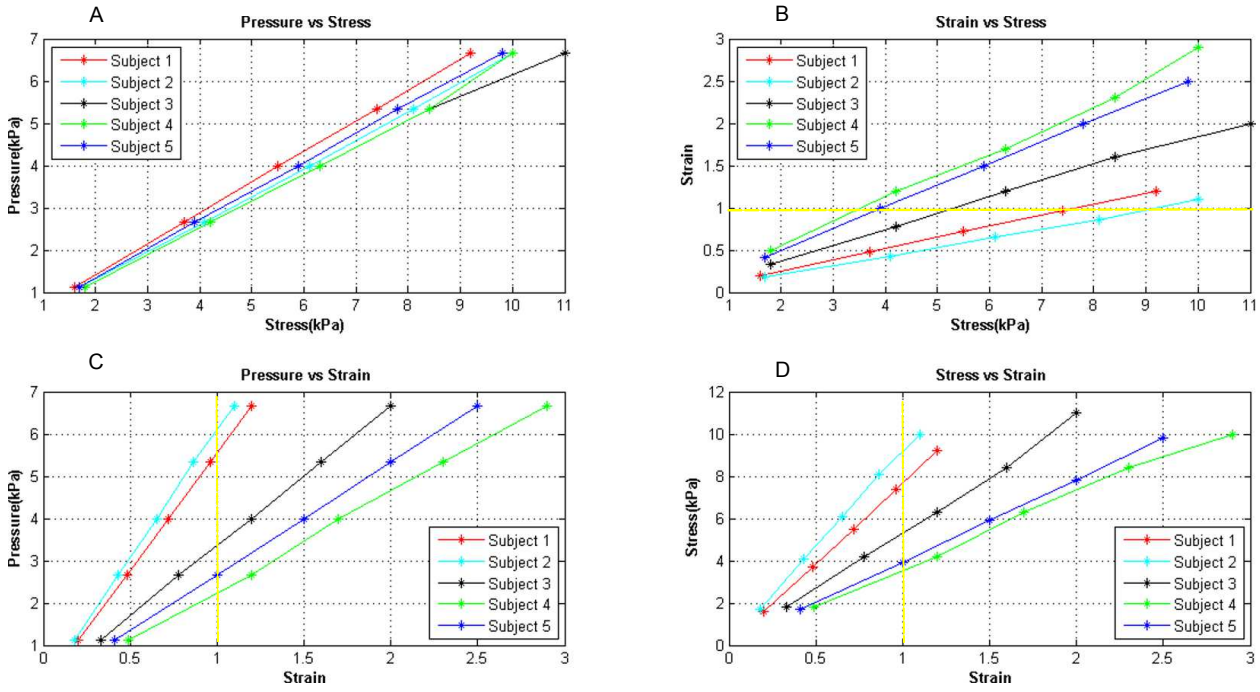

Figure 4.8: *Intracranial stress, strain, and pressure curve dynamics. In all subjects, the stress increases linearly as the pressure increases (A). At pressures of 4 kPa and greater, the stress levels in Subjects 2-5 are greater due to their varying levels of abnormality. The stress-strain relationship is relatively linear in each subject due to the brain tissue's deformation as it is subjected to increasing levels of pressure (B). The strain-pressure curve (C) displays behavior nearly identical to the strain-stress curve (D) which is the inverse of the stress-strain curve (top right). The strain value (yellow) is equal to* $\frac{\Delta V}{V}$ *and suggests that the change in volume (expansion/deformation) in the brain is equal to the original brain volume. Given the tight space of the brain cavity along with strain value evaluation, it can be concluded that Subject 4 reaches the strain value first, followed by Subjects 5 and 3, while Subjects 1 and 2 appear to arrive simultaneously. The pressure and strain at which the brain reaches a strain value of 1 can be used as a critical value or point of herniation.*

4.4 Discussion

FEM intracranial models have been implemented to analyze the deformation and constitutive behavior of brain tissue in the following contexts:

- Conditions of increased intracranial pressure due traumatic brain injury and medical disorders such as hydrocephalus [145, 149, 154–160]

- Prediction of tissue motion during surgical intervention [5, 150–153, 161, 162]

however, this is the first report of medical imaging based FE analysis being performed implementing non-human brain anatomy to our knowledge. Additionally, we subject the 3D models of the complete brain geometry to analysis instead of using symmetric inferences to justify using fractions or sections of brain or creating models based on single slices of medical imaging data. Our reasoning is that analysis of a single MRI slice or isolated section of the brain is insufficient for simulation, evaluation, and observation of changes in global biomechanical behavior and effects due to localized disruptions or abnormalities such as impact injuries, seizures, hematomas, stroke, tumor, thrombosis, and obstruction or occlusion of ventricles resulting in CSF buildup. Using the Paraview visualization software package, our data is presented using mid-line sagittal cross-sections to provide views of the biomechanical responses in subarachnoid space, cerebellum, and brain stem, which are critical in assessment of ICP levels since herniation, shifts, and compression in these areas can be fatal. However, the flexibility of Paraview allows data presentation in all sagittal, coronal, and transverse locations.

4.4.1 Relationships between CPP, ICP, and Biomechanical Behavior

The data in Table 4.4 provides a biomechanical depiction or validation of the effects of decreasing CPP and increasing ICP within the brain cavity. We observe that as the ICP increases, the corresponding stress and strain values increases are proportionally linear. This data, compared with the established relationships between pressure and volume presented in Figure 4.7, verify that the auto-regulatory mechanism attempts to maintain equilibrium

during instances of increased ICP or decreasing pressure gradient CPP. Both the ICP and differences in CPP are highlighted to demonstrate that the FEA method is useful in noninvasive ICP and CPP assessment.

4.4.2 Comparison of Biomechanical Behavior in Normal and Abnormal Brain Models

Although there are minor variations between the stress values present in the abnormal and normal brain models at each pressure, the large variations in the strain values emphasize how the presence of tumorous tissue contributes to deformation of the brain tissue at any pressure. The visualization of biomechanical behaviors in Figure 4.6 display noticeable differences in the distribution patterns of stress and strain values. However, these distribution behaviors correspond to the contact locations of CSF and tissue as well as the abnormalities in each subject presented in Figure 4.2; stress and strain are concentrated in these areas. In contrast to our initial assumption that the brain/SAS interface and boundaries would be the exclusive sites at which the largest amounts of stress and strain are visible, the data in Figure 4.6 concludes that the biomechanical behavior of any area within the cranial cavity is affected when a pressure is exerted on it.

4.4.3 Assessment of Critical Values and Injury Level

The causes of rising ICP levels fall within two general categories in which illnesses, disorders, and impact injuries can contribute to [163]. Brain hemorrhaging, a vascular cause of increased ICP, is classified as active in the presence of arterial distention or dilation and passive in the presence of venous obstruction. Non-vascular causes of increased ICP can be attributed to brain bulk increase (edema), mass effects such as a tumor, abscess, or hematoma, and accumulations of CSF due to flow or absorption obstructions.

The assessment of stress-strain relationships has also provided insight in the determination of critical values in which non-linearity, triggered by cessation of auto-regulation mechanisms, begins and therefore allows us to estimate the cranial stress, strain, and ICP values at which herniation occurs. Anatomical geometry obtained from medical images permits this evaluation to be patient-specific as well as the opportunity to assess whether stress and strain distribution levels and behavior characteristics can be classified according the origination or cause of increased ICP or brain volume. This is advantageous since other works implement FEM with medical imaging to solely focus on either traumatic impact injuries or a specific medical condition (i.e. hydrocephalus, tumor, stroke, aneurysm) [5, 145, 149–162].

As aforementioned and displayed in Figure 4.8A, The stress increases linearly as the pressure increases. At pressures of 4 kPa and greater, the stress levels in Subjects 2-5 are greater due to their varying levels of abnormality. A value of 1 was chosen as the critical strain (yellow line in Figures 4.8B-D). This means that $\frac{\Delta V}{V}$ from Equation 4 is equal to 1, and suggests that the change in volume (expansion/deformation) in the brain is equal to the original brain volume. Given the restricted space of the brain cavity along with strain factor evaluation, it can be concluded that Subject 4 would reach the critical strain value first, followed by Subject 5 and Subject 3, while Subjects 1 and 2 appear to arrive simultaneously. This is valid based on the fact that in Figure 4.2H, the presence of less elastic abnormal tissue in the cerebellum has placed the brain on the verge of herniation and is evident by the ventral and caudal orientation of the cerebellum in comparison to the normal brain in Figure 4.2B. The presence of the large also less elastic tumor tissue in Subject 5 distributed throughout the brain tissue causes additional stress and brain deformation in the instances of increased ICP. In Subject 3, the presence of the less elastic abnormal tissue and more compressible fluid disrupts the normal brain volume and results in larger deformation than in Subject 1, but at a slower rate than Subject 4. This could be attributed to the difference in geographical locations of abnormalities. Subject 2, which suffers from congenital hydrocephalus, has less

brain mass due to the large accumulation of CSF in the lateral ventricle. Because of the autoregulation mechanisms in the brain compensating for the increased volume of CSF along with the absence of abnormal tissue, the biomechanical behavor pattern is very similar to that of Subject 1. The pressure and strain at which the brain reaches a strain factor of 1 can be used as a critical value or as a method to predict the onset of herniation.

From the mathematical equations used to define each mechanical property's significance, we see that stress is an intensity level measurement of internal forces acting on the brain parenchyma and SAS due to the gradient pressure (CPP) or the pressure associated with CSF (ICP). Stress, or dilation, measures the reaction of the brain parenchyma when pressure from the CSF or blood is exerted upon it. Strain is the amount of deformation or volume change the brain undergoes in response to stress. The similar behavior between the strain-pressure and strain-stress curve in Figures 4.8 C-D demonstrates the relationship between stress and pressure. Additionally a pressure vs log strain plot (not shown) would bear resemblance to the pressure-volume plot in Figure 4.7. Furthermore, the slope of the strain-stress curve is the compressibility, which measures the relative volume change of a fluid or solid as a response to a mean stress change and corresponds to the definition of compliance obtained by the inverse slope of pressure-volume curve. These relationships are exemplary of why the total potential energy (TPE) finite element concept is ideal for analysis of the cranial cavity.

ICP deviation values have been used to classify stages of brain injury [164]. When the ICP deviation value is 2.5 kPa, the brain is assumed to be in a state of mild injury. At deviations of approximately 3.5 kPa it enters the moderate injury threshold, and deviations $\geq$ 5 kPa are indicative of severe injury. Coincidently, all subjects have been analyzed at ICP approximately equal to the injury threshold values. Distinguishable changes in biomechanical property behavior in the form of numerical values can now be associated with each individual classification. This further highlights the potential for FEA, medical imaging, and biomechanical distribution response to be used as diagnostic, treatment, and preven-

tion guidelines in clinical settings. Using this information, a diagnostic standard can be implemented to approximate ICP values and define state of brain injury based on behavior observed. The ability to noninvasively assign ICP values and associate those with a state of brain injury could be an invaluable diagnosis, prevention, or treatment assessment tool in clinical veterinary settings where invasive methods of ICP measurement are rarely employed. The stress-strain conceptual relationships can be easily understood by medical and clinical personnel since their behavior mimics that of pressure and volume.

4.4.4 Hindrances and Limitations

The proposed method of ICP evaluation does have limitations. The manual segmentation of the brain geometry from the MR images is a tedious task, which requires an excessive time commitment that temporarily hinders its integration into clinical settings. Varying levels of pixel intensity based on the patient, slice artifacts (e.g. noise), or imaging modality used may result in minor misclassification of materials. In addition, the computational time and memory required for analysis completion (use of 3D model) may also be a factor, however, Tochnog can be linked with libraries that are multi-thread capable, and execution time is reduced correspondingly. In our case, each model had to be scaled to 45% of the original size to reduce the number of nodes and elements present such that Tochnog could operate within the memory constraints of our computers. The resizing does not compromise the integrity of our data for several reasons. Experiments were conducted to establish proof of principle and the data obtained at the model's full size would be proportional to that presented here. This is attributed to the fact that stress and strain values are substantially more dependent on the materials present than the size of the model.

Additionally, the pressure values reported represent instantaneous mean ICP values and not the triphasic sinusoidal waveform measurements commonly obtained in ICP measurements since mean ICP values are used in the calculation of CPP. During our evaluations,

the simulation of CSF flow was not considered; results are based on the static location and amounts of CSF present in each slice of data at the time of image acquisition. It should also be noted that no statistical analysis has been performed given only a single case of normal and each abnormality have been examined.

4.5 Conclusion

We have presented a framework for noninvasive estimation of ICP and CPP utilizing medical imaging data and three-dimensional consolidation of fluid-solid interaction analysis via the finite element method. Our initial aim was to specifically focus on the brain-skull interface, since volume expansion leads to compression of brain parenchyma against the skull. With this method, we rely on the biomechanical properties of the brain (i.e. stress and strain) to identify and in some cases, verify, areas of the brain which are immediately and most affected by the onset and progression of increased ICP and decreased CPP. Each patient scenario is different. However, the data proves our approach's ability to accommodate varying shapes and volumes of cranial cavities associated with medical conditions, age, and breed differences through the use of canine brain models. Future research will revolve around objectives which assist or allow for rapid, automated evaluation of patients in clinical settings such as :

- Identification of quicker, automated methods of segmentation and adjustment of mesh size
- Exploration of registration techniques to ultimately create atlas databases containing:
  - Anatomic brain models of assorted species and breeds for segmentation of brain geometry from medical image data

  - Anatomic brain models of varying injuries and medical conditions, such as tumors and hydrocephalus, for segmentation of brain geometry from medical image data [165]

- Investigation of techniques if available, (i.e. algorithms) that are capable of predicting, obtaining, and assigning mechanical properties based on the geometry and intensity in the medical image in order to improve patient-specificity in analysis

Chapter 5

Conclusion

Comprised of an amalgamation of book chapter and journal publications which present general information and novel applications of integrating medical image segmentation and mesh computation strategies, the objective of this work was to highlight the interdependence of Computer Aided Diagnosis (CAD) and Computer Aided Visualization and Analyisis (CAVA) in the continued construction and advancement of image-based computational analysis of biological tissue and medical device research, design, development, and testing techniques. A thorough understanding of the procedures and concepts for modeling geometry, applying loading/boundary conditions, as well as awareness of the capabilities and limitations is directly correlated to becoming a skillful user of FEA and obtaining consistently accurate results. The concept review provided in Chapter 2 fills a literature gap vital to the mastery of implementing image-based finite element biomedical modeling applications while providing rare insight about the evolving open source movement, advantages, challenges, and available software options ideal for use in such contexts.

Open-source software for image-based FEM has not yet reached the main-stream of the FEM community; however, its free-of-charge, unlimited distribution and modification, transparency, and reproducibility features make it an attractive alternative to commercial, mono-

lithic software. While the interfacing or linking of multiple software option in the construction of a toolchain usually requires familiarity of computer programming, the open source philosophy shows promise as an invaluable contributing factor in the integration of computational mathematic methods like FEA into hospitals and research settings.

The work presented in Chapters 3 and 4 verify the successful execution of two variations of the image-based FEA toolchain using exclusively open-source software in patient-specific biomedical applications, a feat that has been accomplished by only one other research group to date. While the objective in each analysis was to examine the non-linear biomechanical responses of soft tissues when subjected to pressures exerted by fluidic contact, the varying degrees of geometry complexity required the development of application-specific approaches. The differences in the methodologies lie in the medical image segmentation and meshing strategies. Simultaneously, these chapters presented methods to overcome the most challenging step in medical image-based FEA, namely pre-processing (i.e., segmentation and meshing). In Chapter 3, a custom segmentation and meshing module was designed while in Chapter 4 Seg3D (`http://www.sci.utah.edu`), a FOSS alternative to a very popular commercial image segmentation and meshing software MIMICS (Materialise, Leuven, Belgium), was used. The toolchain presented in Chapter 4 is the first reported FOSS uniform solution capable of performing medical imaging FEA in all biomedical applications, including bone, soft tissue, and implants since there are no geometrical or material property limitations or restrictions.

Although limited to geometries of hollow, tubular structure such as vascular grafts, the toolchain presented in Chapter 3 revolves around the development of a hexagonal mesh-extraction module that segments geometry from biomedical images. The module is additionally capable of adaptive mesh refinement, assignment of material properties, loading and boundary conditions, and automatic generation of input file data needed to conduct the actual analysis performed with standalone FE and visualization software. The execution

of the pipeline was time efficient and results obtained met or exceeded those provided my commercial "black box" software equivalents. This provided the desired validation that a completely open-source medical imaging FEA toolchain was indeed attainable.

Building upon the established CAVA-based contributions of the previous chapter, Chapter 4 additionally demonstrated how the CAVA principles and developments achieved by interdisciplinary collaborations are vital and highly influential in the expansion and progression of CAD techniques. 3D tetrahedral mesh representations of detailed anatomical brain geometry were achieved via the use of currently available open-source segmentation and meshing software. In addition to noninvasive assessment of ICP and CPP, the observed changes in biomechanical properties of brain tissues associated with brain injury or abnormalities could assist in the measurement of changes in condition due to drug and radiation therapies in clinical research trials. However, the extensive time constraints associated with obtaining accurate and reproducible geometrical models through manual segmentation reaffirm the need for completely automated methods for medical image segmentation.

True to the open-source principles, future research should immediately focus on the improvement, expansion, and adaptation of the segmentation and meshing strategies implemented in the scope of this work so that they are more automated. More specifically, steps should be taken to integrate the toolchain introduced in Chapter 4 into a single computer system to prepare it for clinical use. The versatility of the universal pipeline should be further validated and verified through the testing of biomedical anatomy of varying material types (e.g. bone and muscle) acquired from multiple imaging modalities. These continual modifications to the source code are examples of intermittent progressions necessary to ultimately achieve the development of application-specific standalone software options capable of automatically performing each of the steps implemented in the presented toolchains with minimal user intervention for patient-specific analysis in clinical settings.

The implementation of rapid prototyping bridges the gap between 3D simulation and physical anatomical models obtained from medical images. In conjunction with biomechanical response behavior, this method could catalyze the creation of new and innovative education and training strategies for medical professionals. Using this technology, interdisciplinary research teams could potentially transition from the research and design (R&D) to manufacturing phases of development such that artificial tissues, medical devices, implants, and diagnostic techniques can advance from conception to production more efficiently while increasing patient specificity.

Bibliography

[1] N. L'Heureux, N. Dusserre, G. Konig, B. Victor, P. Keire, T. N. Wight, N. A. F. Chronos, A. E. Kyles, C. R. Gregory, G. Hoyt, Human tissue-engineered blood vessels for adult arterial revascularization, Nature medicine 12 (3) (2006) 361–365.

[2] R. Courant, Variational methods for the solution of problems of equilibrium and vibrations, Bull. Amer. Math. Soc 49 (1) (1943) 1–23.

[3] D. J. Hawkes, D. Barratt, J. M. Blackall, C. Chan, P. J. Edwards, K. Rhode, G. P. Penney, J. McClelland, D. L. G. Hill, Tissue deformation and shape models in image-guided interventions: a discussion paper, Medical Image Analysis 9 (2) (2005) 163–175.

[4] T. Hopp, M. Dietzel, P. Baltzer, P. Kreisel, W. Kaiser, H. Gemmeke, N. Ruiter, Automatic multimodal 2d/3d breast image registration using biomechanical fem models and intensity-based optimization, Medical Image Analysis 17 (2) (2013) 209–218.

[5] K. Miller, A. Wittek, G. Joldes, A. Horton, T. Dutta-Roy, J. Berger, L. Morriss, Modelling brain deformations for computer-integrated neurosurgery, International Journal for Numerical Methods in Biomedical Engineering 26 (1) (2010) 117–138.

[6] T. A. Sundaram, J. C. Gee, Towards a model of lung biomechanics: pulmonary kinematics via registration of serial lung images, Medical Image Analysis 9 (6) (2005) 524–537.

[7] J. Teo, C. Chui, Z. Wang, S. Ong, C. Yan, S. Wang, H. Wong, S. Teoh, Heterogeneous meshing and biomechanical modeling of human spine, Medical Engineering & Physics 29 (2) (2007) 277 – 290.

[8] W. Parr, U. Chamoli, A. Jones, W. Walsh, S. Wroe, Finite element micro-modelling of a human ankle bone reveals the importance of the trabecular network to mechanical performance: New methods for the generation and comparison of 3d models, Journal of Biomechanics 46 (1) (2013) 200 – 205.

[9] M. Mononen, M. Mikkola, P. Julkunen, R. Ojala, M. Nieminen, J. Jurvelin, R. Korhonen, Effect of superficial collagen patterns and fibrillation of femoral articular cartilage on knee joint mechanicsa 3d finite element analysis, Journal of biomechanics 45 (3) (2012) 579–587.

[10] M. Bajuri, M. R. A. Kadir, M. M. Raman, T. Kamarul, Mechanical and functional assessment of the wrist affected by rheumatoid arthritis: a finite element analysis, Medical engineering & physics 34 (9) (2012) 1294–1302.

[11] C. Lally, F. Dolan, P. J. Prendergast, Cardiovascular stent design and vessel stresses: a finite element analysis, Journal of Biomechanics 38 (8) (2005) 1574–1581.

[12] P. J. Prendergast, C. Lally, S. Daly, A. J. Reid, T. C. Lee, D. Quinn, F. Dolan, Analysis of prolapse in cardiovascular stents: a constitutive equation for vascular tissue and finite-element modelling, J Biomech Eng 125 (5) (2003) 692–699.

[13] P. Mortier, G. Holzapfel, M. De Beule, D. Van Loo, Y. Taeymans, P. Segers, P. Verdonck, B. Verhegghe, A novel simulation strategy for stent insertion and deployment in curved coronary bifurcations: comparison of three drug-eluting stents, Annals of biomedical engineering 38 (1) (2010) 88–99.

[14] H. Huang, J. Hsu, L. Fuh, M. Tu, C. Ko, Y. Shen, Bone stress and interfacial sliding analysis of implant designs on an immediately loaded maxillary implant: A non-linear finite element study, Journal of dentistry 36 (6) (2008) 409–417.

[15] M. Ghajari, S. Peldschus, U. Galvanetto, L. Iannucci, Effects of the presence of the body in helmet oblique impacts, Accident Analysis and Prevention 50 (0) (2013) 263 – 271.

[16] A. Erdemir, T. M. Guess, J. Halloran, S. C. Tadepalli, T. M. Morrison, Considerations for reporting finite element analysis studies in biomechanics, Journal of biomechanics 45 (4) (2012) 625–633.

[17] M. A. Haidekker, Medical Imaging Technology, SpringerBriefs Series in Physics, Springer, 2013.

[18] L. Allard, G. Cloutier, L. Durand, 3d power doppler ultrasound imaging of an in vitro arterial stenosis, Acoustical Imaging 23 (1997) 267–272.

[19] Z. Guo, A. Fenster, Three-dimensional power doppler imaging: A phantom study to quantify vessel stenosis, Ultrasound in Medicine and Biology 22 (8) (1996) 1059–1069.

[20] Z. Guo, L.-G. Durand, L. Allard, G. Cloutier, A. Fenster, In vitro evaluation of multiple arterial stenoses using three-dimensional power doppler angiography, Journal of Vascular Surgery 27 (4) (1998) 681–688.

[21] G. Cloutier, Z. Qin, D. Garcia, G. Soulez, V. Oliva, L.-G. Durand, Assessment of arterial stenosis in a flow model with power doppler angiography: accuracy and observations on blood echogenicity, Ultrasound in Medicine and Biology 26 (9) (2000) 1489–1501.

[22] C.-K. Chui, Z. Wang, J. Zhang, J. S.-K. Ong, L. Bian, J. C.-M. Teo, C.-H. Yan, S.-H. Ong, S.-C. Wang, H.-K. Wong, S.-H. Teoh, A component-oriented software toolkit for patient-specific finite element model generation, Advances in Engineering Software 40 (3) (2009) 184–192.

[23] R. Andresen, H. J. Werner, H. C. Schober, Contribution of the cortical shell of vertebrae to mechanical behaviour of the lumbar vertebrae with implications for predicting fracture risk., British Journal of Radiology 71 (847) (1998) 759–765.

[24] C.-S. Kuo, H.-T. Hu, R.-M. Lin, K.-Y. Huang, P.-C. Lin, Z.-C. Zhong, M.-L. Hseih, Biomechanical analysis of the lumbar spine on facet joint force and intradiscal pressure-a finite element study, BMC Musculoskeletal Disorders 11 (1) (2010) 1–13.

[25] D. Withey, Z. Koles, Medical image segmentation: Methods and software, in: Noninvasive Functional Source Imaging of the Brain and Heart and the International Conference on Functional Biomedical Imaging, 2007. NFSI-ICFBI 2007. Joint Meeting of the 6th International Symposium on, IEEE, 2007, pp. 140–143.

[26] M. Gao, J. Huang, X. Huang, S. Zhang, D. Metaxas, Simplified labeling process for medical image segmentation, Medical Image Computing and Computer-Assisted Intervention–MICCAI 2012 (2012) 387–394.

[27] G. Dougherty, Digital image processing for medical applications, Vol. 1, Cambridge University Press Cambridge, United Kingdom, 2009.

[28] M. Haidekker, Advanced Biomedical Image Analysis, John Wiley & Sons, 2011.

[29] K. Ciesielski, J. Udupa, A framework for comparing different image segmentation methods and its use in studying equivalences between level set and fuzzy connectedness frameworks, Computer Vision and Image Understanding 115 (6) (2011) 721–734.

[30] X. Chen, J. K. Udupa, A. Alavi, D. A. Torigian, Gc-asm: Synergistic integration of graph-cut and active shape model strategies for medical image segmentation, Computer Vision and Image Understanding 117 (5) (2013) 513–524.

[31] M. Kass, A. Witkin, D. Terzopoulos, Snakes: Active contour models, International journal of computer vision 1 (4) (1988) 321–331.

[32] R. Malladi, J. Sethian, B. Vemuri, Shape modeling with front propagation: A level set approach, Pattern Analysis and Machine Intelligence, IEEE Transactions on 17 (2) (1995) 158–175.

[33] S. Osher, R. P. Fedkiw, Level set methods: an overview and some recent results, Journal of Computational physics 169 (2) (2001) 463–502.

[34] A. X. Falcão, J. K. Udupa, S. Samarasekera, S. Sharma, B. E. Hirsch, R. d. A. Lotufo, User-steered image segmentation paradigms: Live wire and live lane, Graphical models and image processing 60 (4) (1998) 233–260.

[35] S. Lloyd, Least squares quantization in pcm, Information Theory, IEEE Transactions on 28 (2) (1982) 129–137.

[36] J. C. Bezdek, Pattern Recognition with Fuzzy Objective Function Algorithms, Kluwer Academic Publishers, Norwell, MA, USA, 1981.

[37] Y. Boykov, G. Funka-Lea, Graph cuts and efficient nd image segmentation, International Journal of Computer Vision 70 (2) (2006) 109–131.

[38] L. Vincent, P. Soille, Watersheds in digital spaces: an efficient algorithm based on immersion simulations, IEEE transactions on pattern analysis and machine intelligence 13 (6) (1991) 583–598.

[39] J. K. Udupa, S. Samarasekera, Fuzzy connectedness and object definition: Theory, algorithms, and applications in image segmentation, Graphical Models and Image Processing 58 (3) (1996) 246 – 261.

[40] A. K. Jain, R. P. W. Duin, J. Mao, Statistical pattern recognition: A review, Pattern Analysis and Machine Intelligence, IEEE Transactions on 22 (1) (2000) 4–37.

[41] K. I. Laws, Texture energy measures., Proc. DARPA Image Understanging Workshop (1979) 47–51.

[42] R. K. Yip, P. K. Tam, D. N. Leung, Modification of hough transform for circles and ellipses detection using a 2-dimensional array, Pattern Recognition 25 (9) (1992) 1007–1022.

[43] T. Cootes, C. Taylor, D. Cooper, J. Graham, et al., Active shape models-their training and application, Computer vision and image understanding 61 (1) (1995) 38–59.

[44] T. Cootes, G. Edwards, C. Taylor, Active appearance models, Computer VisionECCV98 (1998) 484–498.

[45] T. Heimann, H.-P. Meinzer, Statistical shape models for 3d medical image segmentation: A review, Medical Image Analysis 13 (4) (2009) 543 – 563.

[46] D. Lesage, E. D. Angelini, I. Bloch, G. Funka-Lea, A review of 3d vessel lumen segmentation techniques: Models, features and extraction schemes, Medical Image Analysis 13 (6) (2009) 819 – 845.

[47] S. Ghose, A. Oliver, R. Martí, X. Lladó, J. C. Vilanova, J. Freixenet, J. Mitra, D. Sidibé, F. Meriaudeau, A survey of prostate segmentation methodologies in ultrasound, magnetic resonance and computed tomography images, Computer Methods and Programs in Biomedicine 108 (1) (2012) 262–287.

[48] Z. Dokur, T. Ölmez, Segmentation of ultrasound images by using a hybrid neural network, Pattern recognition letters 23 (14) (2002) 1825–1836.

[49] C. Petitjean, J.-N. Dacher, A review of segmentation methods in short axis cardiac mr images, Medical Image Analysis 15 (2) (2011) 169 – 184.

[50] M. M. Fraz, P. Remagnino, A. Hoppe, B. Uyyanonvara, A. R. Rudnicka, C. G. Owen, S. A. Barman, Blood vessel segmentation methodologies in retinal images–a survey, Computer methods and programs in biomedicine 108 (1) (2012) 407–433.

[51] L. Szilágyi, S. M. Szilágyi, B. Benyó, Z. Benyó, Intensity inhomogeneity compensation and segmentation of mr brain images using hybrid c-means clustering models, Biomedical Signal Processing and Control 6 (1) (2011) 3–12.

[52] A. P. Dhawan, S. Juvvadi, Knowledge-based analysis and understanding of medical images, Computer Methods and Programs in Biomedicine 33 (4) (1990) 221 – 239.

[53] F. Masulli, A. Schenone, A fuzzy clustering based segmentation system as support to diagnosis in medical imaging, Artificial Intelligence in Medicine 16 (2) (1999) 129–147.

[54] M. Clark, L. Hall, D. Goldgof, R. Velthuizen, F. Murtagh, M. Silbiger, Automatic tumor segmentation using knowledge-based techniques, Medical Imaging, IEEE Transactions on 17 (2) (1998) 187–201.

[55] D. Zhang, S. Chen, A novel kernelized fuzzy c-means algorithm with application in medical image segmentation, artificial intelligence in medicine 32 (1) (2004) 37–50.

[56] A. Papadopoulos, D. Fotiadis, A. Likas, An automatic microcalcification detection system based on a hybrid neural network classifier, Artificial Intelligence in Medicine 25 (2) (2002) 149–167.

[57] F. Xie, A. C. Bovik, Automatic segmentation of dermoscopy images using self-generating neural networks seeded by genetic algorithm, Pattern Recognition 46 (3) (2013) 1012 – 1019.

[58] S. Gibson, Constrained elastic surface nets: Generating smooth surfaces from binary segmented data, Medical Image Computing and Computer-Assisted Interventation-MICCAI98 (1998) 888–898.

[59] W. Lorensen, H. Cline, Marching cubes: a high resolution 3d surface construction algorithm, Computer Graphics 21 (4) (1987) 163–169.

[60] I. Takanashi, S. Muraki, A. Doi, A. Kaufman, 3d active net for volume extraction, in: Proc. SPIE, Vol. 3298, 1998, pp. 184–193.

[61] M. A. Yerry, M. S. Shephard, Automatic three-dimensional mesh generation by the modified-octree technique, International Journal for Numerical Methods in Engineering 20 (11) (1984) 1965–1990.

[62] M. S. Shephard, M. K. Georges, Automatic three-dimensional mesh generation by the finite octree technique, International Journal for Numerical Methods in Engineering 32 (4) (1991) 709–749.

[63] S. Lo, Volume discretization into tetrahedrai. verification and orientation of boundary surfaces, Computers & Structures 39 (5) (1991) 493–500.

[64] S. Lo, Volume discretization into tetrahedraii. 3d triangulation by advancing front approach, Computers & Structures 39 (5) (1991) 501–511.

[65] B. Delaunay, Sur la sphère vide, Izvestia Akademii Nauk SSSR: Otdelenie Matematicheskikh i Estestvennykh Nauk 7 (1934) 793–800.

[66] T. D. Blacker, R. J. Meyers, Seams and wedges in plastering: a 3-d hexahedral mesh generation algorithm, Engineering with computers 9 (2) (1993) 83–93.

[67] T. Li, R. McKeag, C. Armstrong, Hexahedral meshing using midpoint subdivision and integer programming, Computer Methods in Applied Mechanics and Engineering 124 (1) (1995) 171–193.

[68] R. Schneiders, A grid-based algorithm for the generation of hexahedral element meshes, Engineering with Computers 12 (3) (1996) 168–177.

[69] T. J. Tautges, T. Blacker, S. A. Mitchell, The whisker weaving algorithm: A connectivity-based method for constructing all-hexahedral finite element meshes, International Journal for Numerical Methods in Engineering 39 (19) (1996) 3327–3349.

[70] Z. Yu, M. J. Holst, J. Andrew McCammon, High-fidelity geometric modeling for biomedical applications, Finite Elements in Analysis and Design 44 (11) (2008) 715–723.

[71] S. Azernikov, A. Miropolsky, A. Fischer, Surface reconstruction of freeform objects based on multiresolution volumetric method, in: Proceedings of the eighth ACM symposium on Solid modeling and applications, ACM, 2003, pp. 115–126.

[72] Y. Zhang, C. Bajaj, B.-S. Sohn, 3d finite element meshing from imaging data, Computer methods in applied mechanics and engineering 194 (48) (2005) 5083–5106.

[73] N. Ribeiro, P. Fernandes, D. Lopes, J. Folgado, P. Fernandes, 3-d solid and finite element modeling of biomechanical structures–a software pipeline, in: J. Ambrò sio (Ed.), Proceedings of the 7th EUROMECH Solid Mechanics Conference, Lisbon, Portugal, 2009.

[74] G. B. Dantzig, L. R. Ford, D. R. Fulkerson, A primal-dual algorithm for linear programs, Linear inequalities and related systems (38) (1956) 171–181.

[75] L. A. Freitag, C. Ollivier-Gooch, Tetrahedral mesh improvement using swapping and smoothing, International Journal for Numerical Methods in Engineering 40 (21) (1997) 3979–4002.

[76] T. Ju, F. Losasso, S. Schaefer, J. Warren, Dual contouring of hermite data, ACM Transactions on Graphics (TOG) 21 (3) (2002) 339–346.

[77] S. Schaefer, J. Warren, Dual marching cubes: Primal contouring of dual grids, in: Computer Graphics Forum, Vol. 24, Wiley Online Library, 2005, pp. 195–201.

[78] S. Schaefer, T. Ju, J. Warren, Manifold dual contouring, Visualization and Computer Graphics, IEEE Transactions on 13 (3) (2007) 610–619.

[79] L. R. Herrmann, Laplacian-isoparametric grid generation scheme, Journal of the Engineering Mechanics Division 102 (5) (1976) 749–907.

[80] L. Freitag, M. Jones, P. Plassmann, A parallel algorithm for mesh smoothing, SIAM Journal on Scientific Computing 20 (6) (1999) 2023–2040.

[81] M. Puso, J. Solberg, A stabilized nodally integrated tetrahedral, International Journal for Numerical Methods in Engineering 67 (6) (2006) 841–867.

[82] D. L. Camacho, R. H. Hopper, G. M. Lin, B. S. Myers, An improved method for finite element mesh generation of geometrically complex structures with application to the skullbase, Journal of biomechanics 30 (10) (1997) 1067–1070.

[83] A. Cifuentes, A. Kalbag, A performance study of tetrahedral and hexahedral elements in 3-d finite element structural analysis, Finite Elements in Analysis and Design 12 (3) (1992) 313–318.

[84] V. I. Weingarten, The controversy over hex or tet meshing, Machine design 66 (8) (1994) 74–76.

[85] A. Ramos, J. Simoes, Tetrahedral versus hexahedral finite elements in numerical modelling of the proximal femur, Medical engineering & physics 28 (9) (2006) 916–924.

[86] K. H. Shivanna, S. C. Tadepalli, N. M. Grosland, Feature-based multiblock finite element mesh generation, Computer-Aided Design 42 (12) (2010) 1108–1116.

[87] Z. Wang, J. Teo, C. Chui, S. Ong, C. Yan, S. Wang, H. Wong, S. Teoh, Computational biomechanical modelling of the lumbar spine using marching-cubes surface smoothened finite element voxel meshing, Computer Methods and Programs in Biomedicine 80 (1) (2005) 25 – 35.

[88] D. Thompson, J. Braun, R. Ford, OpenDX – Paths to Visualization, Visualization and Imagery Solutions, Inc., 2001.

[89] G.-H. Kwon, S.-W. Chae, K.-J. Lee, Automatic generation of tetrahedral meshes from medical images, Computers and Structures 81 (2003) 765–775.

[90] W. Lee, T.-S. Kim, M. Cho, Y. Ahn, S. Lee, Methods and evaluations of mri content-adaptive finite element mesh generation for bioelectromagnetic problems, Physics in Medicine and Biology 51 (2006) 6173–6186.

[91] P. Perré, Meshpore: a software able to apply image-based meshing techniques to anisotropic and heterogeneous porous media, Drying technology 23 (2005) 1993–2006.

[92] A. Rosset, L. Spadola, O. Ratib, Osirix: an open-source software for navigating in multidimensional dicom images, Journal of Digital Imaging 17 (3) (2004) 205–216.

[93] N. M. Grosland, K. H. Shivanna, V. A. Magnotta, N. A. Kallemeyn, N. A. DeVries, S. C. Tadepalli, C. Lisle, Ia-femesh: An open-source, interactive, multiblock approach to anatomic finite element model development, Computer Methods and Programs in Biomedicine 94 (1) (2009) 96–107.

[94] L. Antiga, M. Piccinelli, L. Botti, B. Ene-Iordache, A. Remuzzi, D. A. Steinman, An image-based modeling framework for patient-specific computational hemodynamics, Medical and Biological Engineering and Computing 46 (11) (2008) 1097–1112.

[95] Q. Fang, D. A. Boas, Tetrahedral mesh generation from volumetric binary and gray-scale images, in: Proceedings of the Sixth IEEE international conference on Symposium on Biomedical Imaging: From Nano to Macro, IEEE Press, 2009, pp. 1142–1145.

[96] J. R. Bronson, J. A. Levine, R. T. Whitaker, Lattice cleaving: Conforming tetrahedral meshes of multimaterial domains with bounded quality, in: Proceedings of the 21st International Meshing Roundtable, Springer, 2013, pp. 191–209.

[97] A. Madison, M. A. Haidekker, A completely open-source finite element modeling chain for tubular tissue-engineered constructs, International Journal of Computer Science and Application (IJCSA) 1 (2) (2012) 44–55.

[98] S. K. Boyd, R. Müller, Smooth surface meshing for automated finite element model generation from 3d image data, Journal of biomechanics 39 (7) (2006) 1287–1295.

[99] S. A. Maas, B. J. Ellis, G. A. Ateshian, J. A. Weiss, Febio: finite elements for biomechanics, Journal of biomechanical engineering 134 (1) (2012) 011005–011005–10.

[100] M. Jolley, J. Stinstra, D. Weinstein, S. Pieper, R. Estepar, G. Kindlmann, R. MacLeod, D. Brooks, J. Triedman, Open-source environment for interactive finite element modeling of optimal icd electrode placement, Functional Imaging and Modeling of the Heart (2007) 373–382.

[101] M. Jolley, J. Stinstra, S. Pieper, R. MacLeod, D. H. Brooks, F. Cecchin, J. K. Triedman, A computer modeling tool for comparing novel icd electrode orientations in children and adults, Heart Rhythm 5 (4) (2008) 565–572.

[102] J. K. Udupa, V. R. LeBlanc, Y. Zhuge, C. Imielinska, H. Schmidt, L. M. Currie, B. E. Hirsch, J. Woodburn, A framework for evaluating image segmentation algorithms, Computerized Medical Imaging and Graphics 30 (2) (2006) 75 – 87.

[103] K. Doi, Computer-aided diagnosis in medical imaging: Historical review, current status and future potential, Computerized Medical Imaging and Graphics 31 (4-5) (2007) 198 – 211.

[104] H. Kobatake, Future cad in multi-dimensional medical images: Project on multi-organ, multi-disease cad system, Computerized Medical Imaging and Graphics 31 (4) (2007) 258–266.

[105] L. Baghdadi, D. A. Steinman, H. M. Ladak, Template-based finite-element mesh generation from medical images, Computer methods and programs in biomedicine 77 (1) (2005) 11–21.

[106] I. A. Sigal, M. R. Hardisty, C. M. Whyne, Mesh-morphing algorithms for specimen-specific finite element modeling, Journal of Biomechanics 41 (7) (2008) 1381–1389.

[107] M. A. Baldwin, J. E. Langenderfer, P. J. Rullkoetter, P. J. Laz, Development of subject-specific and statistical shape models of the knee using an efficient segmentation and mesh-morphing approach, Computer Methods and Programs in Biomedicine 97 (3) (2010) 232 – 240.

[108] M. Bucki, C. Lobos, Y. Payan, A fast and robust patient specific finite element mesh registration technique: Application to 60 clinical cases, Medical Image Analysis 14 (3) (2010) 303 – 317.

[109] G. D. Santis, M. D. Beule, K. V. Canneyt, P. Segers, P. Verdonck, B. Verhegghe, Full-hexahedral structured meshing for image-based computational vascular modeling, Medical Engineering & Physics 33 (10) (2011) 1318 – 1325.

[110] C. Lederman, A. Joshi, I. Dinov, L. Vese, A. Toga, J. D. V. Horn, The generation of tetrahedral mesh models for neuroanatomical mri, NeuroImage 55 (1) (2011) 153 – 164.

[111] R. Petzold, H.-F. Zeilhofer, W. Kalender, Rapid prototyping technology in medicine-basics and applications, Computerized Medical Imaging and Graphics 23 (5) (1999) 277–284.

[112] R. Bibb, J. Winder, A review of the issues surrounding three-dimensional computed tomography for medical modelling using rapid prototyping techniques, Radiography 16 (1) (2010) 78–83.

[113] T. Boehler, D. van Straaten, S. Wirtz, H.-O. Peitgen, A robust and extendible framework for medical image registration focused on rapid clinical application deployment, Computers in biology and medicine 41 (6) (2011) 340–349.

[114] K. B. Chandran, S. C. Vigmostad, Patient-specific bicuspid valve dynamics: Overview of methods and challenges, Journal of Biomechanics 46 (2) (2013) 208 – 216.

[115] V. Kurtcuoglu, D. Poulikakos, Y. Ventikos, et al., Computational modeling of the mechanical behavior of the cerebrospinal fluid system, Transactions of the ASME-K-Journal of Biomechanical Engineering 127 (2) (2005) 264–269.

[116] A. Hagemann, K. Rohr, H. S. Stiehl, Coupling of fluid and elastic models for biomechanical simulations of brain deformations using fem, Medical Image Analysis 6 (4) (2002) 375–388.

[117] J. Al-Sukhun, C. Lindqvist, N. Ashammakhi, H. Penttilä, Microvascular stress analysis. part i: simulation of microvascular anastomoses using finite element analysis, Br J Oral Maxillofac Surg 45 (2) (2007) 130–137.

[118] C. J. Beller, M. R. Labrosse, M. J. Thubrikar, F. Robicsek, Role of aortic root motion in the pathogenesis of aortic dissection, Circulation 109 (6) (2004) 763–769.

[119] C. J. Beller, M. R. Labrosse, M. J. Thubrikar, G. Szabo, F. Robicsek, S. Hagl, Increased aortic wall stress in aortic insufficiency: clinical data and computer model, Eur J Cardiothorac Surg 27 (2) (2005) 270–275.

[120] D. N. Ghista, A. S. Kobayashi, N. Davids, Analyses of some biomechanical structures and flows by computer-based finite element method, Computers in Biology and Medicine 5 (1975) 119–122, IN7, 123–161.

[121] N. L'Heureux, S. Pâquet, R. Labbé, L. Germain, F. A. Auger, A completely biological tissue-engineered human blood vessel, The FASEB Journal 12 (1) (1998) 47–56.

[122] J. T. LaCroix, J. Xia, M. A. Haidekker, A fully automated approach to quantitatively determine thickness of tissue-engineered cell sheets, Annals of Biomedical Engineering 37 (7) (2009) 1348–1357.

[123] J. T. LaCroix, M. A. Haidekker, Quantifying light scattering with single-mode fiber-optic confocal microscopy, BMC Medical Imaging 9 (1) (2009) 1–10.

[124] W. Wu, W.-Q. Wang, D.-Z. Yang, M. Qi, Stent expansion in curved vessel and their interactions: A finite element analysis, Journal of Biomechanics 40 (11) (2007) 2580–2585.

[125] M. Haidekker, R. Andresen, C. Evertsz, D. Banzer, H. Peitgen, Assessing the degree of osteoporosis in the axial skeleton using the dependence of the fractal dimension on the grey level threshold., The British Journal of Radiology 70 (834) (1997) 586–93.

[126] H. M. Huang, J. Xia, M. A. Haidekker, Fast optical transillumination tomography with large-size projection acquisition, Annals of Biomedical Engineering 36 (10) (2008) 1699–1707.

[127] T. Elvins, A survey of algorithms for volume visualization., ACM SIGGRAPH Computer Graphics 26 (3) (1992) 194–201.

[128] C. Montani, R. Scateni, R. Scopigno, Discretized marching cubes, in: IEEE Conferenvce Visualization, 1994, pp. 281–287.

[129] T. Nishimara, T. Fugimoto, Fast contour line extraction algorithm with selective thresholding according to line continuity, Systems and Computers in Japan 25 (3) (1994) 101–110.

[130] A. Monro, Observations on the structure and functions of the nervous system, Edinburgh, 1783.

[131] G. Kellie, On death from cold, and, on congestion of the brain: An account of the appearances observed in the dissection of two of three individuals presumed to have perished in the storm of 3rd november 1821; with some reflections on the pathology of the brain, Transaction Medico Chirurgical Society Edinb 1 (1824) 84–169.

[132] D. Popovic, M. Khoo, S. Lee, Noninvasive monitoring of intracranial pressure, Recent Patents on Biomedical Engineering 2 (2009) 165–179.

[133] J. Zhong, M. Dujovny, H. Park, E. Perez, A. Perlin, F. Diaz, Advances in icp monitoring techniques, Neurological research 25 (4) (2003) 339–350.

[134] K. Pattinson, G. Wynne-Jones, C. H. Imray, Monitoring intracranial pressure, perfusion and metabolism, Continuing Education in Anaesthesia, Critical Care & Pain 5 (4) (August 2005) 130–133.

[135] M. Smith, Monitoring intracranial pressure in traumatic brain injury, Anesthesia & Analgesia 106 (1) (2008) 240–248.

[136] P. Raboel, J. Bartek, M. Andresen, B. Bellander, B. Romner, Intracranial pressure monitoring: Invasive versus non-invasive methods a review, Critical Care Research and Practice 2012 (2012) Article ID 950393, 14 pages.

[137] H. W. Ryder, F. F. Espey, F. V. Kristoff, J. P. Evans, Observations on the interrelationships of intracranial pressure and cerebral blood flow., Journal of neurosurgery 8 (1) (1951) 46–58.

[138] A. Marmarou, et al., A theoretical model and experimental evaluation of the cerebrospinal fluid system, Ph.D. thesis, Drexel University Philadelphia (1973).

[139] A. Marmarou, K. Shulman, J. LaMorgese, Compartmental analysis of compliance and outflow resistance of the cerebrospinal fluid system, Journal of neurosurgery 43 (5) (1975) 523–534.

[140] A. Marmarou, K. Shulman, R. M. Rosende, A nonlinear analysis of the cerebrospinal fluid system and intracranial pressure dynamics, Journal of neurosurgery 48 (3) (1978) 332–344.

[141] S. Sivaloganathan, G. Tenti, J. Drake, Mathematical pressure volume models of the cerebrospinal fluid, Applied Mathematics and Computation 94 (2) (1998) 243–266.

[142] J. Miller, J. Garibi, Intracranial volume/pressure relationships during continuous monitoring of ventricular fluid pressure, in: Intracranial pressure, Springer, 1972, pp. 270–274.

[143] J. D. Miller, J. Garibi, J. D. Pickard, Induced changes of cerebrospinal fluid volume: effects during continuous monitoring of ventricular fluid pressure, Archives of neurology 28 (4) (1973) 265–269.

[144] A. Wittek, K. Miller, R. Kikinis, S. K. Warfield, Patient-specific model of brain deformation: Application to medical image registration, Journal of biomechanics 40 (4) (2007) 919–929.

[145] T. Dutta-Roy, A. Wittek, K. Miller, Biomechanical modelling of normal pressure hydrocephalus, Journal of Biomechanics 41 (10) (2008) 2263–2271.

[146] D. G. Changaris, C. P. McGraw, R. J. David, H. D. Garretson, E. J. Arpin, C. B. Shields, Correlation of cerebral perfusion pressure and glasgow coma scale to outcome, The Journal of Trauma and Acute Care Surgery 27 (9) (1987) 1007–1013.

[147] C. McGraw, A cerebral perfusion pressure greater than 80 mm hg is more beneficial, in: Intracranial pressure VII, Springer, 1989, pp. 839–841.

[148] S. Lin, S. Shieh, M. Grimm, Ultrasonic measurements of brain tissue properties, in: Proceedings of the Symposium of the Centers for Disease Control and Prevention, 1997, pp. 27–31.

[149] K. Omori, L. Zhang, K. H. Yang, A. I. King, Effect of cerebral vasculatures on the mechanical response of brain tissue: a preliminary study, ASME Applied Mechanics Division 246 (2000) 167–174.

[150] G. Dassios, M. Kiriakopoulos, V. Kostopoulos, On the sensitivity of the vibrational response of the human head, Computational mechanics 21 (4) (1998) 382–388.

[151] M. Gilchrist, D. O'Donoghue, Simulation of the development of frontal head impact injury, Computational mechanics 26 (3) (2000) 229–235.

[152] S. Gong, H. Lee, C. Lu, Computational simulation of the human head response to non-contact impact, Computers & Structures 86 (7) (2008) 758–770.

[153] Z. Li, Y. Luo, Finite element study of correlation between intracranial pressure and external vibration responses of human head, Adv. Theor. Appl. Mech 3 (3) (2010) 139–149.

[154] T. Nagashima, N. Tamaki, S. Matsumoto, B. Horwitz, Y. Seguchi, et al., Biomechanics of hydrocephalus: a new theoretical model., Neurosurgery 21 (6) (1987) 898–903.

[155] Y. Tada, R. Matsumoto, Y. Nishimura, Mechanical modelings of the brain and simulation of the biomechanism of hydrocephalus, JSME international journal. Ser. 1, Solid mechanics, strength of materials 33 (2) (1990) 269–275.

[156] E. Jacobson, D. Fletcher, M. Morgan, I. Johnston, Fluid dynamics of the cerebral aqueduct, Pediatric neurosurgery 24 (5) (1996) 229–236.

[157] A. Peña, M. Bolton, H. Whitehouse, J. Pickard, Effects of brain ventricular shape on periventricular biomechanics: a finite-element analysis., Neurosurgery 45 (1) (1999) 107–116.

[158] Z. Taylor, K. Miller, Reassessment of brain elasticity for analysis of biomechanisms of hydrocephalus, Journal of biomechanics 37 (8) (2004) 1263–1269.

[159] A. A. Linninger, M. Xenos, D. C. Zhu, M. R. Somayaji, S. Kondapalli, R. D. Penn, Cerebrospinal fluid flow in the normal and hydrocephalic human brain, Biomedical Engineering, IEEE Transactions on 54 (2) (2007) 291–302.

[160] M. Kaczmarek, R. P. Subramaniam, S. R. Neff, The hydromechanics of hydrocephalus: steady-state solutions for cylindrical geometry, Bulletin of mathematical biology 59 (2) (1997) 295–323.

[161] K. Miller, Constitutive model of brain tissue suitable for finite element analysis of surgical procedures, Journal of Biomechanics 32 (5) (1999) 531–537.

[162] A. Valencia, B. Blas, J. H. Ortega, Modeling of brain shift phenomenon for different craniotomies and solid models, Journal of Applied Mathematics 2012 (2012) Article ID 409127, 20 pages.

[163] I. Piper, Intracranial pressure and elastance, in: P. Reilly, R. Bullock (Eds.), Head Injury: Physiology and Management of Severe Closed Injury, Chapman & Hall, London, 2004, pp. 101–120.

[164] X. Yue, L. Wang, R. Wang, Tissue modeling and analyzing with finite element method: A review for cranium brain imaging, International Journal of Biomedical Imaging 2013 (2013) Article ID 781603, 12 pages.

[165] S. K. Kyriacou, C. Davatzikos, S. J. Zinreich, R. N. Bryan, Nonlinear elastic registration of brain images with tumor pathology using a biomechanical model [mri], Medical Imaging, IEEE Transactions on 18 (7) (1999) 580–592.

Appendix

Interface Between Meshing Module and Tochnog

Summary of key elements in the Tochnog control file. Any meshing program needs to produce a file that contains the components described in this section.

- The header section. The header contains general control elements, for example the number of spatial dimensions, and the simulated values we want to observe. In our case, we chose to observe the velocity, stresses, the total and elastic strains of the 3-D tubular construct when subjected to homogeneous pressure.

- The nodes section. This section lists all vertices (nodes) and their spatial location. Therefore, this section describes the geometry of the object. Nodes must be shared between adjoining elements. An example of lines that describe a group of nodes follows:

```
node 0    4.000000  0.000000  0.000000
node 1    3.695518  1.530734  0.000000
node 2    5.000000  0.000000  0.000000
node 3    4.619398  1.913417  0.000000
node 4    4.000000  0.000000  1.000000
node 5    3.695518  1.530734  1.000000
node 6    5.000000  0.000000  1.000000
node 7    4.619398  1.913417  1.000000
```

  Nodes are numbered consecutively (the number after the keyword "node", and the spatial coordinates in a Cartesian system follow. It can be seen in this example that the first four nodes belong to the lowest slice at $z = 0$, and the next four nodes belong to the slice at $z = 1$.

- The elements section. In this section, nodes are grouped to form a cubic element. Several element geometries are available in the selected FEM software, but we made use of a cube, which is referred to as *hex8* element in the software's terminology. A sample element definition follows:

```
element 0    -hex8      0  1  2  3  4  5  6  7
element 1    -hex8      1  8  3  9  5  10  7  11
element 2    -hex8      8  12  9  13  10  14  11  15
element 3    -hex8      12  16  13  17  14  18  15  19
```

  The number after the keyword "element" is the element number, used in material assignments. The instruction "-hex8" indicates the geometry, and the following eight integer numbers are the numbers of the nodes for this element. It can be seen that nodes are shared between adjoining elements. The order of the nodes is crucial. Figure 3.5 is a sketch that shows the required order of the nodes.

- The element grouping section. In this section, each element is assigned to one material group. In one extreme case, only one material group exists, and each element is assigned to element group 0. In the other extreme case, variability is so high that each element has its own material group. Depending on material property binning, the number of material groups can be reduced substantially.

- The material properties section. In this section, material properties (among them, Young's modulus, Poisson elasticity, and material density) are defined. This is also the section where nonlinear material properties are introduced in the FEM software.

- The node boundary condition definitions. In this section nodes may be subjected to boundary constraints. In the special case that is considered in this study, all inner-wall nodes are subjected to a constant pressure, and all nodes in the lowest and uppermost slice are held fixed in space.

- The control section. In this section, the evolution in time of the simulation can be controlled. Factors include time steps, iteration limits, time intervals after which a snapshot is saved, and a control parameter of how the FEM software may subdivide the cubes if they become too inhomogeneous.

Generation of Tochnog Input File for Subjects 1 and 2

Computer code used to create Tochnog input/control file contents described in previous section for Subjects 1 and 2.

```
%TOCHNOG INPUT FILE FOR BRAIN/ICP ANALYSIS
%Subjects 1 and 2

%DATA MANAGEMENT

%First we need to read in Cleaver files

%Opens Element File from Cleaver
name= input('Please enter name for element file including extension: ', 's');
fida=fopen(name);

%Reads data into Cell Array
A=textscan(fida,' %d %d %d %d %d %d ', 'delimiter', ' ');


%Counts number of lines in the file
eL=countLines(name);

%Close File
fclose(fida);

%Place each column of your cell array into a vector array
%This tells what info is in each column.

enum = A{1}; %Element Number
node1= A{2}; %First node in element
node2= A{3}; %Second node in element
node3= A{4}; %Third node in element
node4= A{5}; %Fourth node in element
mat= A{6};   %Material group of element

%Opens Nodes File
%Prompt user to enter name of Nodes file
name2= input('Please enter name for node file including extension: ', 's');
fidb=fopen(name2);

%Reads data into Cell Array
B=textscan(fidb,'%f %f %f %f ', 'delimiter',' ');

%Counts number of lines in the file
nL=countLines(name2);

%Close File
fclose(fidb);
```

```
45
46 %Place each column of your cell array into a vector array.
47 %This tells what info is in each column.
48 nnum = B{1}; %Node Number
49 nodex= B{2}; %x-coordinate of node
50 nodey= B{3}; %y-coordinate of node
51 nodez= B{4}; %z-coordinate of node
52
53 %Now we want to update the element and nodes file such that none of the
54 %elements or nodes in material 5 ( Cleaver padding) are included.
55
56 %ELEMENTS
57 %Create an array(or matrix) which combines all of the element vector
58 %arrays.
59
60 elearray=double(horzcat(enum, node1, node2, node3, node4, mat));
61
62 %Creates new array based on previous that does not include material 5 or
63 %air. Instructs program to keep only the rows where the mat group
64 %(Column 6) are not equal to 5
65
66 elearrayairdump= elearray(elearray(:,6)~=1,:);
67
68 %Creates new array based on previous that does not include material 6 or
69 %padding. Instructs program to keep only the rows where the mat group
70 %(Column 6) are not equal to 6
71
72 elearray1= elearrayairdump(elearrayairdump(:,6)~=6,:);
73
74 %Creates new array of our "padded-less" array w/o element number (A2-A6
75 %from above)
76
77 elearray2=elearray1(:,2:6);
78 %length of array
79 eL=length(elearray2);
80
81
82 %Creates new array of elearray2 containing only node position columns (A2-A5
83 %earlier, but most recently columns 1-4 in elearray2;
84
85 JEL=elearray2(:,1:4);
86 %length of array (should be the same as eL)
87 jL=length(JEL);
88
89 %NODES
90 %Create an array(or matrix) which combines all of the node vector arrays.
91
92 nodearray=horzcat(nnum, nodex, nodey, nodez);
93
94 %COMPARISON:  Test to see what node numbers remain now that padded elements
95 %and nodes have been dumped
```

```
96
97  %Unique function extracts each node value in the JEL array once, so
98  %that instances of repetition are not considered.
99
100 re_nodes= unique(JEL);
101
102 %Now we can use the intersection function to reduce the size of the first
103 % column in nodearray file such that only the nodes found in re_nodes remain.
104 % This eliminates all unused node numbers in the node file.
105
106 %Reduces nnum array to only the nodes in re_nodes
107
108 RN= intersect(nnum,re_nodes);
109 re=length(re_nodes);
110
111 %Creates new nodearray1 that get rid of all rows in nodearray that do not
112 %have node numbers found in RN. (Dumps all node # and coordinates for
113 %unused nodes.
114
115 % determines what values in column 1 of nodearray that are also in RN
116
117 p = ismember(nodearray(:,1),RN);
118
119 %Creates nodearray1 that only contains rows where the values in column 1
120 %of nodearray that are a member of RN
121
122 nodearray1 = nodearray(p,:);
123 na1=length(nodearray1);
124
125
126 %Separate the node numbers coordinates in to column arrays. We will need
127 % this for Tochnog file. Multiply by pixel measurement conversion factors
128 % to get accurate, realistic geometric size.
129
130 new_nodes=double(nodearray1(:,1));
131 uxn=0.71.*(nodearray1(:,2));
132 uyn=0.71.*(nodearray1(:,3));
133 uzn=2.38.*(nodearray1(:,4));
134
135 %Now that all unused nodes, elements, and material groups have been
136 %discarded, the last step before writing the Tochnog input file is the
137 %renumbering of the nodes an elements such that each begins with 0.
138
139 %Start with the replace function process to renumber nodes in JEL.
140
141 %REPLACE - Replace Elements ( by Jos van der Geest)
142 %  B = REPLACE(A,S1,S2) returns a matrix B in which the elements in
143 % A that are in S1 are replaced by those in S2. In general, S1 and S2
144 % should have an equal number of elements. If S2 has one element,
145 % it is expanded to match the size of S1.
146
```

```
147 %In our case the "B" matrix will be our renumbered matrix, "A" is JEL
148 % ( elearray containing only the columns with node positions),
149 % "S1" is new_nodes( array containing only the used node numbers,
150
151 % and "S2" is simply a matrix that is the same size and length of new_nodes
152  %numbered from %0:length(newnodes)-1; (Ex: If new_nodes is 1:10, S2=0:9).
153
154 NUM=length(new_nodes)-1;
155
156 %Creates row vector.
157 %Can also be used for node count in during Tochnog file writing
158
159 S2=(0:NUM)';
160
161 %This function is perfect!!! Works very fast!
162
163 RENUM=replace(JEL,new_nodes,S2);
164 renL=length(RENUM); %to verify if #output lines = input lines
165
166
167
168 %Separate the renum vectors
169
170 fen1=RENUM(:,1);
171 fen2=RENUM(:,2);
172 fen3=RENUM(:,3);
173 fen4=RENUM(:,4);
174
175 % material groups (5th column of elearray2 from earlier)
176
177 m=elearray2(:,5);
178 mL=length(m);
179
180 %Preallocate Arrays prior to for loopto eliminate incremental increases of
181 %the data structures.  This helps performance and memory use
182
183 count=zeros(size(na1));
184 uxx=zeros(size(uxn));
185 uyy=zeros(size(uyn));
186 uzz=zeros(size(uzn));
187 m1=zeros(renL,1);
188 FEN1=zeros(renL,1);
189 FEN2=zeros(renL,1);
190 FEN3=zeros(renL,1);
191 FEN4=zeros(renL,1);
192 %File Data extraction, filtering, and renumbering complete.
193
194 %TOCHNOG INPUT FILE CONSTRUCTION
195  %Now that the data is extracted from the Cleaver files, we are ready to
196  %format it such that it is ready for Tochnog.
197
```

```
%Prompt user to enter desired name for Tochnog Input File
name3= input('Enter name for Tochnog Input File and add ".txt": ', 's');

%Open up file named after user's input and prepare for writing data
fid=fopen(name3,'wt');

%HEADER
 % The header section contains general control elements, for example the
 % number of spatial dimensions, and the simulated values we want to observe.

fprintf(fid,'echo -yes \n');
fprintf(fid,'number_of_space_dimensions 3 \n');
fprintf(fid,'groundflow_pressure \n');
fprintf(fid,'materi_velocity \n');
fprintf(fid,'materi_stress \n');
fprintf(fid,'materi_strain_elasti \n');
fprintf(fid,'materi_strain_total \n');
fprintf(fid,'end_initia \n');
fprintf(fid,'\n');
fprintf(fid,'\n');
fprintf(fid,'\n');

%The elements section. In this section, nodes are grouped to form a
%tetrahedral element. Several element geometries are available in Tochnog,
%but we made use of 4 node tetrahedrals, which is referred to as
-tet4
%element in the software's terminology. A sample element definition
%follows:
%element 0   t e t 4   1 2 3 4
%where the elements are numbered sequentially and in this example nodes 1-4
%make up the first element.

elno=-1;
for compare2=1:renL %increments through each row of renL
        elno=elno+1; %starts numbering at 0

       %Recall individual columns of REN created earlier

       FEN1=fen1(compare2);
       FEN2=fen2(compare2);
       FEN3=fen3(compare2);
       FEN4=fen4(compare2);

     %FEN1-FEN4 represent the renumbered nodes(starting from 0)
     %Now we have created final element and node configurations.

%To view in command window
%fprintf('element %d -tet4 %d %d %d %d \n', elno, FEN1, FEN2, FEN3, FEN4,m)

%Write to file
fprintf(fid,'element %d -tet4 %d %d %d %d\n', elno, FEN1, FEN2, FEN3, FEN4);
```

```
248
249 end
250 fprintf(fid,'\n');
251 fprintf(fid,'\n');
252 fprintf(fid,'\n');
253
254 % The nodes section lists all vertices (nodes) and their spatial location.
255 % Therefore, this section describes the geometry of the object. Nodes must
256 % be shared between adjoining elements.  Nodes are numbered consecutively
257 % (the number after the keyword "node", and the spatial coordinates in a
258 % Cartesian system follow. An example of lines that describe a group of
259 % nodes follows:
260 %node 0  x   y   z
261 %node 1  x   y   z
262 %node 2  x   y   z
263
264 for order=1:na1  %increments through all rows each of the new_node array
265      format long
266
267     %Prepares node numbers to start at(or with)zero
268     %Reuse S2 array since it starts at zero and same length as
    new_node
269     count= S2(order);
270
271     uxx=double(uxn(order)); %node coordinates of used nodes
272     uyy=double(uyn(order));
273     uzz=double(uzn(order));
274
275
276 %FINALIZED RENUMBERED NODE FILE WITHOUT PADDING
277 %To view in command window
278 %fprintf('node %f %f %f %f\n', count,uxx,uyy,uzz)
279
280 %Write to file
281 fprintf(fid,'node %d %e %e %e\n',count,uxx,uyy,uzz);
282
283 end
284 fprintf(fid,'\n');
285 fprintf(fid,'\n');
286 fprintf(fid,'\n');
287
288 %Now we want to ensure that the  CSF and SAScsf compartment (material 4)
289 %is fixed in space (i.e. no translation or rotation) so we will restrict
290 %velocity in all directions at all nodes within the materials.
291
292 %Create an array that contains FEN1, FEN2, FEN3, FEN4, and m
293
294 matsort=horzcat(fen1, fen2, fen3, fen4, m);
295
296 %Now we want to only keep rows of matsort that are Materials 4 and 5
297 %(the CSF and SAScsf).
```

```
298
299 matsort4= matsort(matsort(:,5)==5 | matsort(:,5)==4,:);
300
301 %Eliminate repeating nodes
302 %Create an array of just the nodes in mat 4
303
304 mat4nodes=matsort4(:,1:4);
305
306 %Unique function extracts each node value in the matsort array once, so
307 %that instances of repetition are not considered. It also arranges them in
308 %ascending order, which is exactly what Tochnog prefers.
309
310 NR=unique(mat4nodes);
311 nrL=length(NR);
312
313 %Preallocate memory for node array
314
315 SASnodes=zeros(size(NR));
316
317 index=-1;
318 for SAS=1:nrL
319      %Will subtract 1 from order value so that count starts at zero
320     index= index+1;
321     SASnodes=NR(SAS);%node at line (row) SAS
322
323     fprintf(fid,'bounda_unknown %d %d -velx -vely -velz \n',index,SASnodes);
324 end
325
326 fprintf(fid,'\n');
327 fprintf(fid,'\n');
328 fprintf(fid,'\n');
329
330 %The final constraint applies a pressure force of 1.13 kPa (average ICP in
331 %dogs) at each node of CSF (material 4). For example, if we are interested
332 %in applying this force at nodes 99-102:
333
334 %Now we want to only keep rows of matsort that are material 4 (the CSF).
335
336 matsort3= matsort(matsort(:,5)==4,:);
337
338 %Create an array of just the nodes in mat 3
339
340 CSFnodes1=matsort3(:,1:4);
341 m3=unique(CSFnodes1);
342 m3L=length(m3);
343
344 %Preallocate memory for node array
345 CSFnodes=zeros(size(m3));
346
347 %Starts count at +1 of the last referenced index of boundary unknown so that
348  %counts won't be repeated
```

```
index2=index+1;
for Comb=1:m3L
    index2=index2+1; % so that count  is consecutively numbered
    CSFnodes=m3(Comb); %node at line (row) Comb

    fprintf(fid,'bounda_unknown %d %d -pres \n',index2,CSFnodes);
    fprintf(fid,'bounda_time %d 0.0 -6.67e-03 1.0 -6.67e-03\n',index2);

end
fprintf(fid,'\n');
fprintf(fid,'\n');
fprintf(fid,'\n');

% The element grouping section. In this section, each element is assigned
% to one material group.  In the case of the normal brain, there are 3
% material groups (1. brain tissue 2. CSF, and 3. SAScsf) so all of our
% elements should be classified according to these.
%element_group 0 3
%element_group 1 2
%In this example, element 0 is classified to Group 3 or brain tissue 2
%while element 1 is in Group 4.

elno1=-1;
for group=1:mL
        elno1=elno1+1; % starts numbering at 0
       %m=material array (5th column of elearray2 from earlier)
         m1=m(group);

       %Merges Group 2 and 3 (The Brain Tissue into one group
         if m1==2
             m1=3;
         end

%To view in command window
%fprintf('element_group %d %d \n', elno1, m1);

%Write to file
fprintf(fid,'element_group %d %d \n',elno1, m1);
end

fprintf(fid,'\n');
fprintf(fid,'\n');
fprintf(fid,'\n');

% In the material definition section, material properties (among them,
% Young's modulus, Poisson elasticity, and material density) are defined.
% This is also the section where nonlinear material properties are
% introduced in the FEM software.  We have 4 groups (not considering the
% background) consisting of both fluids and solids so there should be
% material information for each.

```

```
400 %Brain Tissue
401 fprintf(fid,'group_type 3 -materi \n');
402 fprintf(fid,'group_materi_elasti_young 3 .5581 \n');
403 fprintf(fid,'group_materi_density 3 1.04e-06 \n');
404 fprintf(fid,'group_materi_elasti_poisson 3 0.485 \n');
405 fprintf(fid,'group_materi_memory 3 -updated_without_rotation \n');
406 fprintf(fid,'\n');
407 fprintf(fid,'\n');
408 fprintf(fid,'\n');
409
410 %CSF
411 fprintf(fid,'group_type 4 -materi \n');
412 fprintf(fid,'group_materi_density 4  1.0e-06 \n');
413 fprintf(fid,'group_materi_elasti_compressibility 4 4.57e-10 \n');
414 fprintf(fid,'group_materi_elasti_poisson 4 0.499 \n');
415 fprintf(fid,'group_materi_memory 4 -updated_without_rotation \n');
416 fprintf(fid,'group_materi_stokes 4 -yes \n');
417 fprintf(fid,'\n');
418 fprintf(fid,'\n');
419 fprintf(fid,'\n');
420
421 %SAScsf
422 fprintf(fid,'group_type 5 -materi \n');
423 fprintf(fid,'group_materi_elasti_young 5 0.650 \n');
424 fprintf(fid,'group_materi_density 5 1.04e-06 \n');
425 fprintf(fid,'group_materi_elasti_poisson 5 0.450 \n');
426 fprintf(fid,'group_materi_memory 5 -updated_without_rotation \n');
427 fprintf(fid,'\n');
428 fprintf(fid,'\n');
429 fprintf(fid,'\n');
430
431 % The final component of the input file contains the control section where
432 % the evolution in time of the simulation can be controlled. Factors
433 % include time steps, iteration limits, time intervals after which a
434 % snapshot is saved, and output file formats.
435
436 fprintf(fid,'control_timestep \t 30  0.1 0.3\n');
437 fprintf(fid,'control_timestep_iterations   \t 30  1 \n');
438 fprintf(fid,'control_print    \t 30  -time_current \n');
439 fprintf(fid,'control_print_vtk \t 1545  -yes \n');
440 fprintf(fid,'end_data \n');
441
442 %Close the file. We are done here!
443 fclose(fid);
```

Generation of Tochnog Input File for Subject 3

Computer code used to create Tochnog input/control file for Subject 3.

```
% TOCHNOG INPUT FILE FOR BRAIN/ICP ANALYSIS
%Abnormal Brain (Cyst and Cystic Fluid) (Subject 3)

%DATA MANAGEMENT

%First we need to read in Cleaver files

%Opens Element File from Cleaver

name= input('Please enter name for element file including extension: ', 's');
fida=fopen(name);

%Reads data into Cell Array
A=textscan(fida,' %d %d %d %d %d %d ', 'delimiter', ' ');

%Counts number of lines in the file
eL=countLines(name);

%Close File
fclose(fida);

%Place each column of your cell array into a vector array
%This tells what info is in each column.

enum = A{1}; %Element Number
node1= A{2}; %First node in element
node2= A{3}; %Second node in element
node3= A{4}; %Third node in element
node4= A{5}; %Fourth node in element
mat= A{6};   %Material group of element

%Opens Nodes File
%Prompt user to enter name of Nodes file
name2= input('Please enter name for node file including extension: ', 's');
fidb=fopen(name2);

%Reads data into Cell Array
B=textscan(fidb,'%f %f %f %f ', 'delimiter',' ');

%Counts number of lines in the file
nL=countLines(name2);

%Close File
fclose(fidb);

```

```
%Place each column of your cell array into a vector array
%This tells what info is in each column.

nnum = B{1}; %Node Number
nodex= B{2}; %x-coordinate of node
nodey= B{3}; %y-coordinate of node
nodez= B{4}; %z-coordinate of node

%Now we want to update the element and nodes file such that none of the
%elements or nodes in material 5 ( Cleaver padding) are included.

%ELEMENTS
%Create an array(or matrix) which combines all of the element vector
%arrays.

elearray=double(horzcat(enum, node1, node2, node3, node4, mat));

%Creates new array based on previous that does not include material 1 or
%air.  Instructs program to keep only the rows where the mat group
% (Column 6) are not equal to 1

elearrayairdump= elearray(elearray(:,6)~=1,:);

%Creates new array based on previous that does not include material 8 or
%padding. Instructs program to keep only the rows where the mat group
% (Column 8) are not equal to 8

elearray1= elearrayairdump(elearrayairdump(:,6)~=8,:);

%Creates new array of our "padded-less" array w/o element number (A2-A6
%from above)

elearray2=elearray1(:,2:6);
%length of array
eL=length(elearray2);

%Creates new array of elearray2 containing only node position columns A2-A5
%earlier, but most recently columns 1-4 in elearray2;

JEL=elearray2(:,1:4);
%length of array (should be the same as eL)
jL=length(JEL);

%NODES
%Create an array(or matrix) which combines all of the node vector arrays.

nodearray=horzcat(nnum, nodex, nodey, nodez);

%COMPARISON:  Test to see what node numbers remain now that padded elements
%and nodes have been dumped

```

```
%Unique function extracts each node value in the JEL array once, so
%that instances of repetition are not considered.
re_nodes= unique(JEL);

%Now we can use the intersection function to reduce the size of the first column
% in nodearray file such that only the nodes found in re_nodes remain.
%This eliminates all unused node numbers in the node file.

%Reduces nnum array to only the nodes in re_nodes

RN= intersect(nnum,re_nodes);
re=length(re_nodes);

%Creates new nodearray1 that get rid of all rows in nodearray that do not
%have node numbers found in RN. (Dumps all node # and coordinates for
%unused nodes.

% Determines what values in column 1 of nodearray that are also in RN
p = ismember(nodearray(:,1),RN);

%Creates nodearray1 that only contains rows where the values in column 1
%of nodearray that are a member of RN

nodearray1 = nodearray(p,:);
na1=length(nodearray1);


%Separate the node numbers coordinates in to column arrays. We will need this for
%Tochnog file.

new_nodes=double(nodearray1(:,1));
uxn=0.71.*(nodearray1(:,2));
uyn=0.71.*(nodearray1(:,3));
uzn=2.38.*(nodearray1(:,4));

%Now that all unused nodes, elements, and material groups have been
%discarded, the last step before writing the Tochnog input file is the
%renumbering of the nodes an elements such that each begins with 0.

%Start with the replace function process to renumber nodes in JEL.

%REPLACE - Replace Elements ( by Jos van der Geest)
%   B = REPLACE(A,S1,S2) returns a matrix B in which the elements in A that
%   are in S1 are replaced by those in S2. In general, S1 and S2 should have
%   an equal number of elements. If S2 has one element, it is expanded to
%   match the size of S1.

%In our case the "B" matrix will be our renumbered matrix, "A" is JEL
% (elearray containing only the columns with node positions),
%"S1" is new_nodes( array %containing only the used node numbers,
% and "S2" is simply a matrix that is the same size and length of new_nodes
```

```
148 %numbered from 0:length(newnodes)-1;
149 %(Ex: If new_nodes is 1:10, S2=0:9).
150
151 NUM=length(new_nodes)-1;
152
153 %Creates row vector.
154 %Can also be used for node count in during Tochnog file writing
155
156 S2=(0:NUM)';
157 RENUM=replace(JEL,new_nodes,S2);
158 renL=length(RENUM); %to very if #output lines = input lines
159
160 %Separate the renum vectors
161
162 fen1=RENUM(:,1);
163 fen2=RENUM(:,2);
164 fen3=RENUM(:,3);
165 fen4=RENUM(:,4);
166
167 % Material groups (5th column of elearray2 from earlier)
168
169 m=elearray2(:,5);
170 mL=length(m);
171
172 %Preallocate Arrays prior to for loopto eliminate incremental increases of
173 %the data structures.  This helps performance and memory use
174
175 count=zeros(size(na1));
176 uxx=zeros(size(uxn));
177 uyy=zeros(size(uyn));
178 uzz=zeros(size(uzn));
179 m1=zeros(renL,1);
180 FEN1=zeros(renL,1);
181 FEN2=zeros(renL,1);
182 FEN3=zeros(renL,1);
183 FEN4=zeros(renL,1);
184 %File Data extraction, filtering, and renumbering complete.
185
186 %TOCHNOG INPUT FILE CONSTRUCTION
187
188 %Now that the data is extracted from the Cleaver files, we are ready to
189 %format it such that it is ready for Tochnog.
190
191 %Prompt user to enter desired name for Tochnog Input File
192 name3= input('Enter name for Tochnog Input File and add ".txt": ', 's');
193
194 %Open up file named after user's input and prepare for writing data
195 fid=fopen(name3,'wt');
196
197 %HEADER
198  % The header section contains general control elements, for example the
```

```
199  % number of spatial dimensions, and the simulated values we want to observe.
200
201 fprintf(fid,'echo -yes \n');
202 fprintf(fid,'number_of_space_dimensions 3 \n');
203 fprintf(fid,'groundflow_pressure \n');
204 fprintf(fid,'materi_velocity \n');
205 fprintf(fid,'materi_stress \n');
206 fprintf(fid,'materi_strain_elasti \n');
207 fprintf(fid,'materi_strain_total \n');
208 fprintf(fid,'end_initia \n');
209 fprintf(fid,'\n');
210 fprintf(fid,'\n');
211 fprintf(fid,'\n');
212
213 %The elements section. In this section, nodes are grouped to form a
214 %tetrahedral element. Several element geometries are available in Tochnog,
215 %but we made use of 4 node tetrahedrals, which is referred to as
   -tet4
216 %element in the software's terminology. A sample element definition
217 %follows:
218 %element 0   t e t 4   1 2 3 4
219 %where the elements are numbered sequentially and in this example nodes 1-4
220 %make up the first element.
221
222 elno=-1;
223 for compare2=1:renL %increments through each row of renL
224         elno=elno+1; %starts numbering at 0
225
226        %Recall individual columns of REN created earlier
227
228        FEN1=fen1(compare2);
229        FEN2=fen2(compare2);
230        FEN3=fen3(compare2);
231        FEN4=fen4(compare2);
232
233      %FEN1-FEN4 represent the renumbered nodes(starting from 0)
234      %Now we have created final element and node configurations.
235
236 %To view in command window
237 %fprintf('element %d -tet4 %d %d %d %d \n', elno, FEN1, FEN2, FEN3, FEN4,m)
238
239 %Write to file
240 fprintf(fid,'element %d -tet4 %d %d %d %d\n', elno, FEN1, FEN2, FEN3, FEN4);
241
242 end
243 fprintf(fid,'\n');
244 fprintf(fid,'\n');
245 fprintf(fid,'\n');
246
247 % The nodes section lists all vertices (nodes) and their spatial location.
248 % Therefore, this section describes the geometry of the object. Nodes must
```

```
% be shared between adjoining elements.  Nodes are numbered consecutively
% (the number after the keyword "node", and the spatial coordinates in a
% Cartesian system follow. An example of lines that describe a group of
% nodes follows:
%node 0  x   y   z
%node 1  x   y   z
%node 2  x   y   z


for order=1:na1  %increments through all rows each of the new_node array
     format long

    %Prepares node numbers to start at(or with)zero
    %Reuse S2 array since it starts at zero and same length as
new_node
    count= S2(order);
    uxx=double(uxn(order)); %node coordinates of used nodes
    uyy=double(uyn(order));
    uzz=double(uzn(order));


%FINALIZED RENUMBERED NODE FILE WITHOUT PADDING
%To view in command window
%fprintf('node %f %f %f %f\n', count,uxx,uyy,uzz)

%Write to file
fprintf(fid,'node %d %e %e %e\n',count,uxx,uyy,uzz);

end
fprintf(fid,'\n');
fprintf(fid,'\n');
fprintf(fid,'\n');

%We want to ensure that the CSF, SAScsf, and cystic fluid compartments
% (material 4, 5, &7) are fixed in space (i.e. no translation or rotation) so we
% will restrict velocity in all directions at all nodes within the materials.

%Create an array that contains FEN1, FEN2, FEN3, FEN4, and m

matsort=horzcat(fen1, fen2, fen3, fen4, m);




%Now we want to only keep rows of matsort that are material 4,5,& 7
%(the csf, SAScsf, and cystic fluid).

matsort4= matsort(matsort(:,5)==4 | matsort(:,5)==5 | matsort(:,5)==7,:);

%Eliminate repeating nodes
%Create an array of just the nodes in mat 4
```

```
299
300 mat4nodes=matsort4(:,1:4);
301
302 %Unique function extracts each node value in the matsort array once, so
303 %that instances of repetition are not considered. It also arranges them in
304 %ascending order, which is exactly what Tochnog prefers.
305
306 NR=unique(mat4nodes);
307 nrL=length(NR);
308
309 %Preallocate memory for node array
310
311 SASnodes=zeros(size(NR));
312
313 index=-1;
314 for SAS=1:nrL
315     %Will subtract 1 from order value so that count starts at zero
316     index= index+1;
317     SASnodes=NR(SAS);%node at line (row) SAS
318
319     fprintf(fid,'bounda_unknown %d %d -velx -vely -velz \n',index,SASnodes);
320 end
321
322 fprintf(fid,'\n');
323 fprintf(fid,'\n');
324 fprintf(fid,'\n');
325
326 %This constraint applies a pressure force of 1.13 kPa
327 %(average ICP in dogs) at each node of CSF (material 4).
328
329 %Now we want to only keep rows of matsort that are material 4 (the CSF).
330
331 matsort3= matsort(matsort(:,5)==4,:);
332
333
334 %Create an array of just the nodes in mat 4
335
336 CSFnodes1=matsort3(:,1:4);
337 m3=unique(CSFnodes1);
338 m3L=length(m3);
339
340 %Preallocate memory for node array
341 CSFnodes=zeros(size(m3));
342
343 %Starts count at +1 of the last referenced index of bounda unknown so that
344 % counts won't be repeated
345 index2=index+1;
346 for Comb=1:m3L
347     index2=index2+1; % so that count  is consecutively numbered
348     CSFnodes=m3(Comb); %node at line (row) Comb
349
```

```
350     fprintf(fid,'bounda_unknown %d %d -pres \n',index2,CSFnodes);
351     fprintf(fid,'bounda_time %d 0.0 -6.67e-03 1.0 -6.67e-03\n',index2);
352
353 end
354 fprintf(fid,'\n');
355 fprintf(fid,'\n');
356 fprintf(fid,'\n');
357
358 % The element grouping section. In this section, each element is assigned
359 % to one material group.  In the case of the normal brain, there are 3
360 % material groups (1. brain tissue 2. CSF, and 3. SAScsf) so all of our
361 % elements should be classified according to these.
362 %element_group 0 3
363 %element_group 1 2
364 %In this example, element 0 is classified to Group 3 or brain tissue 2
365 %while element 1 is in Group 4.
366
367 elno1=-1;
368 for group=1:mL
369         elno1=elno1+1; % starts numbering at 0
370
371         %m=material array (5th column of elearray2 from earlier)
372          m1=m(group);
373
374
375
376        %Merges Group 2 and 3 (The Brain Tissue into one group
377          if m1==2
378              m1=3;
379          end
380
381 %To view in command window
382 %fprintf('element_group %d %d \n', elno1, m1);
383
384 %Write to file
385 fprintf(fid,'element_group %d %d \n',elno1, m1);
386 end
387
388 fprintf(fid,'\n');
389 fprintf(fid,'\n');
390 fprintf(fid,'\n');
391
392 % In the material definition section, material properties (among them,
393 % Young's modulus, Poisson elasticity, and material density) are defined.
394 % This is also the section where nonlinear material properties are
395 % introduced in the FEM software.  We have 4 groups (not considering the
396 % background) consisting of both fluids and solids so there should be
397 % material information for each.
398
399 %Brain Tissue
400 fprintf(fid,'group_type 3 -materi \n');
```

```
401 fprintf(fid,'group_materi_elasti_young 3 .5581 \n');
402 fprintf(fid,'group_materi_density 3 1.04e-06 \n');
403 fprintf(fid,'group_materi_elasti_poisson 3 0.485 \n');
404 fprintf(fid,'group_materi_memory 3 -updated_without_rotation \n');
405 fprintf(fid,'\n');
406 fprintf(fid,'\n');
407 fprintf(fid,'\n');
408
409 %CSF
410 fprintf(fid,'group_type 4 -materi \n');
411 fprintf(fid,'group_materi_density 4  1.0e-06 \n');
412 fprintf(fid,'group_materi_elasti_compressibility 4 4.57e-10 \n');
413 fprintf(fid,'group_materi_elasti_poisson 4 0.499 \n');
414 fprintf(fid,'group_materi_memory 4 -updated_without_rotation \n');
415 fprintf(fid,'group_materi_stokes 4 -yes \n');
416 fprintf(fid,'\n');
417 fprintf(fid,'\n');
418 fprintf(fid,'\n');
419
420 %SAScsf
421 fprintf(fid,'group_type 5 -materi \n');
422 fprintf(fid,'group_materi_elasti_young 5 0.650 \n');
423 fprintf(fid,'group_materi_density 5 1.04e-06 \n');
424 fprintf(fid,'group_materi_elasti_poisson 5 0.450 \n');
425 fprintf(fid,'group_materi_memory 5 -updated_without_rotation \n');
426 fprintf(fid,'\n');
427 fprintf(fid,'\n');
428 fprintf(fid,'\n');
429
430 %Cyst
431 fprintf(fid,'group_type 6 -materi \n');
432 fprintf(fid,'group_materi_elasti_young 6 1.150 \n');
433 fprintf(fid,'group_materi_density 6 1.04e-06 \n');
434 fprintf(fid,'group_materi_elasti_poisson 6 0.350 \n');
435 fprintf(fid,'group_materi_memory 6 -updated_without_rotation \n');
436 fprintf(fid,'\n');
437 fprintf(fid,'\n');
438 fprintf(fid,'\n');
439
440 %Cystic Fluid (mult compressibility by two and change poisson to 0.47)
441 fprintf(fid,'group_type 7 -materi \n');
442 fprintf(fid,'group_materi_density 7  1.0e-06 \n');
443 fprintf(fid,'group_materi_elasti_compressibility 7 9.14e-10 \n');
444 fprintf(fid,'group_materi_elasti_poisson 7 0.470 \n');
445 fprintf(fid,'group_materi_memory 7 -updated_without_rotation \n');
446 fprintf(fid,'group_materi_stokes 7 -yes \n');
447 fprintf(fid,'\n');
448 fprintf(fid,'\n');
449 fprintf(fid,'\n');
450
451 % The final component of the input file contains the control section where
```

```
452 % the evolution in time of the simulation can be controlled. Factors
453 % include time steps, iteration limits, time intervals after which a
454 % snapshot is saved, and output file formats.
455
456 fprintf(fid,'control_timestep \t 30  0.1 0.3\n');
457 fprintf(fid,'control_timestep_iterations   \t 30  1 \n');
458 fprintf(fid,'control_print    \t 30  -time_current \n');
459 fprintf(fid,'control_print_vtk \t 3545  -yes \n');
460 fprintf(fid,'end_data \n');
461
462 %Close the file. We are done here!
463 fclose(fid);
```

Generation of Tochnog Input File for Subjects 4 and 5

Computer code used to create a Tochnog input/control file for Subjects 4 and 5.

```
% TOCHNOG INPUT FILE FOR BRAIN/ICP ANALYSIS
%Abnormal Brain Geometry (Tumor, Mass) (Subjects 4 and 5)

%DATA MANAGEMENT

%First we need to read in Cleaver files

%Opens Element File from Cleaver
name= input('Please enter name for element file including extension: ', 's');
fida=fopen(name);

%Reads data into Cell Array
A=textscan(fida,' %d %d %d %d %d %d ', 'delimiter', ' ');

%Counts number of lines in the file
eL=countLines(name);

%Close File
fclose(fida);

%Place each column of your cell array into a vector array
%This tells what info is in each column.

enum = A{1}; %Element Number
node1= A{2}; %First node in element
node2= A{3}; %Second node in element
node3= A{4}; %Third node in element
node4= A{5}; %Fourth node in element
mat= A{6};   %Material group of element

%Opens Nodes File
%Prompt user to enter name of Nodes file
name2= input('Please enter name for node file including extension: ', 's');
fidb=fopen(name2);

%Reads data into Cell Array
B=textscan(fidb,'%f %f %f %f ', 'delimiter',' ');


%Counts number of lines in the file
nL=countLines(name2);

%Close File
fclose(fidb);

```

```
%Place each column of your cell array into a vector array
%This tells what info is in each column.

nnum = B{1}; %Node Number
nodex= B{2}; %x-coordinate of node
nodey= B{3}; %y-coordinate of node
nodez= B{4}; %z-coordinate of node

%Now we want to update the element and nodes file such that none of the
%elements or nodes in material 5 ( Cleaver padding) are included.

%ELEMENTS
%Create an array(or matrix) which combines all of the element vector
%arrays.

elearray=double(horzcat(enum, node1, node2, node3, node4, mat));

%Creates new array based on previous that does not include material 1 or
%air. Instructs program to keep only the rows where the mat group
% (Column 6) are not equal to 1

elearrayairdump= elearray(elearray(:,6)~=1,:);

%Creates new array based on previous that does not include material 7 or
%padding. Instructs program to keep only the rows where the mat group
%(Column 6) %are not equal to 7

elearray1= elearrayairdump(elearrayairdump(:,6)~=7,:);

%Creates new array of our "padded-less" array w/o element number (A2-A6
%from above)

elearray2=elearray1(:,2:6);
%length of array
eL=length(elearray2);

%Creates new array of elearray2 containing only node position columns (A2-A5
%earlier, but most recently columns 1-4 in elearray2;

JEL=elearray2(:,1:4);
%length of array (should be the same as eL)
jL=length(JEL);

%NODES
%Create an array(or matrix) which combines all of the node vector arrays.

nodearray=horzcat(nnum, nodex, nodey, nodez);

%COMPARISON:  Test to see what node numbers remain now that padded elements
%and nodes have been dumped

```

```
%Unique function extracts each node value in the JEL array once, so
%that instances of repetition are not considered.

re_nodes= unique(JEL);

%Now we can use the intersection function to reduce the size of the first
% column in nodearray file such that only the nodes found in re_nodes remain.
%This eliminates all unused node numbers in the node file.

%Reduces nnum array to only the nodes in re_nodes

RN= intersect(nnum,re_nodes);
re=length(re_nodes);

%Creates new nodearray1 that get rid of all rows in nodearray that do not
%have node numbers found in RN. (Dumps all node # and coordinates for
%unused nodes.

% determines what values in column 1 of nodearray that are also in RN
p = ismember(nodearray(:,1),RN);

%Creates nodearray1 that only contains rows where the values in column 1
%of nodearray that are a member of RN

nodearray1 = nodearray(p,:);
na1=length(nodearray1);

%Separate the node numbers coordinates in to column arrays. We will need
% this for Tochnog file.

new_nodes=double(nodearray1(:,1));
uxn=0.71.*(nodearray1(:,2));
uyn=0.71.*(nodearray1(:,3));
uzn=2.38.*(nodearray1(:,4));

%Now that all unused nodes, elements, and material groups have been
%discarded, the last step before writing the Tochnog input file is the
%renumbering of the nodes an elements such that each begins with 0.

%Start with the replace function process to renumber nodes in JEL.

%REPLACE - Replace Elements ( by Jos van der Geest)
%   B = REPLACE(A,S1,S2) returns a matrix B in which the elements in A that
%   are in S1 are replaced by those in S2. In general, S1 and S2 should have
%   an equal number of elements. If S2 has one element, it is expanded to
%   match the size of S1.

%In our case the "B" matrix will be our renumbered matrix, "A" is JEL
% (elearray containing only the columns with node positions),
% "S1" is new_nodes( array containing only the used node numbers,
% and "S2" is simply a matrix that is the same size and length of new_nodes
```

```
% numbered from 0:length(newnodes)-1; (Ex: If new_nodes iss 1:10, S2=0:9).

NUM=length(new_nodes)-1;

%creates row vector.
%Can also be used for node count in during Tochnog file writing

S2=(0:NUM)';
RENUM=replace(JEL,new_nodes,S2);
renL=length(RENUM); %to verify if #output lines = input lines

%Separate the renum vectors

fen1=RENUM(:,1);
fen2=RENUM(:,2);
fen3=RENUM(:,3);
fen4=RENUM(:,4);

% Material groups (5th column of elearray2 from earlier)

m=elearray2(:,5);
mL=length(m);

%Preallocate Arrays prior to for loopto eliminate incremental increases of
%the data structures.  This helps performance and memory use.

count=zeros(size(na1));
uxx=zeros(size(uxn));
uyy=zeros(size(uyn));
uzz=zeros(size(uzn));
m1=zeros(renL,1);
FEN1=zeros(renL,1);
FEN2=zeros(renL,1);
FEN3=zeros(renL,1);
FEN4=zeros(renL,1);
%File Data extraction, filtering, and renumbering complete.

%TOCHNOG INPUT FILE CONSTRUCTION

%Now that the data is extracted from the Cleaver files, we are ready to
%format it such that it is ready for Tochnog.

%Prompt user to enter desired name for Tochnog Input File
name3= input('Enter name for Tochnog Input File and add ".txt": ', 's');

%Open up file named after user's input and prepare for writing data
fid=fopen(name3,'wt');

%HEADER
 % The header section contains general control elements, for example the
 % number of spatial dimensions, and the simulated values we want to observe.
```

```
199
200 fprintf(fid,'echo -yes \n');
201 fprintf(fid,'number_of_space_dimensions 3 \n');
202 fprintf(fid,'groundflow_pressure \n');
203 fprintf(fid,'materi_velocity \n');
204 fprintf(fid,'materi_stress \n');
205 fprintf(fid,'materi_strain_elasti \n');
206 fprintf(fid,'materi_strain_total \n');
207 fprintf(fid,'end_initia \n');
208 fprintf(fid,'\n');
209 fprintf(fid,'\n');
210 fprintf(fid,'\n');
211
212 %The elements section. In this section, nodes are grouped to form a
213 %tetrahedral element. Several element geometries are available in Tochnog,
214 %but we made use of 4 node tetrahedrals, which is referred to as  -tet4
215 %element in the software's terminology. A sample element definition
216 %follows:
217 %element 0  tet4  1 2 3 4
218 %where the elements are numbered sequentially and in this example nodes 1-4
219 %make up the first element.
220
221 elno=-1;
222 for compare2=1:renL %increments through each row of renL
223         elno=elno+1; %starts numbering at 0
224
225        %Recall individual columns of REN created earlier
226
227        FEN1=fen1(compare2);
228        FEN2=fen2(compare2);
229        FEN3=fen3(compare2);
230        FEN4=fen4(compare2);
231
232      %FEN1-FEN4 represent the renumbered nodes(starting from 0)
233      %Now we have created final element and node configurations.
234
235
236 %To view in command window
237 %fprintf('element %d -tet4 %d %d %d %d \n', elno, FEN1, FEN2, FEN3, FEN4,m)
238
239 %Write to file
240 fprintf(fid,'element %d -tet4 %d %d %d %d\n', elno, FEN1, FEN2, FEN3, FEN4);
241
242 end
243 fprintf(fid,'\n');
244 fprintf(fid,'\n');
245 fprintf(fid,'\n');
246
247 % The nodes section lists all vertices (nodes) and their spatial location.
248 % Therefore, this section describes the geometry of the object. Nodes must
249 % be shared between adjoining elements.  Nodes are numbered consecutively
```

```
% (the number after the keyword "node", and the spatial coordinates in a
% Cartesian system follow. An example of lines that describe a group of
% nodes follows:
%node 0  x   y   z
%node 1  x   y   z
%node 2  x   y   z


for order=1:na1  %increments through all rows each of the new_node array
     format long

    %Prepares node numbers to start at(or with)zero
    %Reuse S2 array since it starts at zero and same length as  new_node
    count= S2(order);
    uxx=double(uxn(order)); %node coordinates of used nodes
    uyy=double(uyn(order));
    uzz=double(uzn(order));


%FINALIZED RENUMBERED NODE FILE WITHOUT PADDING
%To view in command window
%fprintf('node %f %f %f %f\n', count,uxx,uyy,uzz)

%Write to file
fprintf(fid,'node %d %e %e %e\n',count,uxx,uyy,uzz);

end
fprintf(fid,'\n');
fprintf(fid,'\n');
fprintf(fid,'\n');

%Now we want to ensure that the CSF and sSAScsf compartment
% (material 4 & 5) is fixed in space (i.e. no translation or rotation)
%so we will restrict velocity in all directions at all nodes within the
% materials.

%Create an array that contains FEN1, FEN2, FEN3, FEN4, and m

matsort=horzcat(fen1, fen2, fen3, fen4, m);

%Now we want to only keep rows of matsort that are material  4 & 5
%(the CSF and SAScsf).
matsort4= matsort(matsort(:,5)==5 | matsort(:,5)==4,:);

%Eliminate repeating nodes
%Create an array of just the nodes in mat 4

mat4nodes=matsort4(:,1:4);

%Unique function extracts each node value in the matsort array once, so
%that instances of repetition are not considered. It also arranges them in
```

```
%ascending order, which is exactly what Tochnog prefers.
NR=unique(mat4nodes);
nrL=length(NR);

%Preallocate memory for node array

SASnodes=zeros(size(NR));

index=-1;
for SAS=1:nrL
    %Will subtract 1 from order value so that count starts at zero
    index= index+1;
    SASnodes=NR(SAS);%node at line (row) SAS

    fprintf(fid,'bounda_unknown %d %d -velx -vely -velz \n',index,SASnodes);
end

fprintf(fid,'\n');
fprintf(fid,'\n');
fprintf(fid,'\n');


%This constraint applies a pressure force of 1.13 kPa (average ICP in
%dogs) at each node of CSF (material 4).

%Now we want to only keep rows of matsort that are material 3 (the CSF).

matsort3= matsort(matsort(:,5)==4,:);

%Create an array of just the nodes in mat 3

CSFnodes1=matsort3(:,1:4);
m3=unique(CSFnodes1);
m3L=length(m3);

%Preallocate memory for node array
CSFnodes=zeros(size(m3));

%Starts count at +1 of the last referenced index of bounda unknown so that
% counts won't be repeated
index2=index+1;
for Comb=1:m3L
    index2=index2+1;% so that count  is consecutively numbered
    CSFnodes=m3(Comb);%node at line (row) Comb

    fprintf(fid,'bounda_unknown %d %d -pres \n',index2,CSFnodes);
    fprintf(fid,'bounda_time %d 0.0 -5.33e-03 1.0 -5.33e-03\n',index2);

end
fprintf(fid,'\n');
fprintf(fid,'\n');
```

```
fprintf(fid,'\n');

% The element grouping section. In this section, each element is assigned
% to one material group.  In the case of the normal brain, there are 3
% material groups (1. brain tissue 2. CSF, and 3. SAScsf) so all of our
% elements should be classified according to these.
%element_group 0 3
%element_group 1 2
%In this example, element 0 is classified to Group 3 or brain tissue 2
%while element 1 is in Group 4.

elno1=-1;
for group=1:mL
        elno1=elno1+1; % starts numbering at 0

         %m=material arra y(5th column of elearray2 from earlier)
         m1=m(group);

       %Merges Group 2 and 3 (The Brain Tissue into one group
         if m1==2
             m1=3;
         end

%To view in command window
%fprintf('element_group %d %d \n', elno1, m1);

%Write to file
fprintf(fid,'element_group %d %d \n',elno1, m1);
end

fprintf(fid,'\n');
fprintf(fid,'\n');
fprintf(fid,'\n');

% In the material definition section, material properties (among them,
% Young's modulus, Poisson elasticity, and material density) are defined.
% This is also the section where nonlinear material properties are
% introduced in the FEM software.  We have 4 groups (not considering the
% background) consisting of both fluids and solids so there should be
% material information for each.

%Brain Tissue
fprintf(fid,'group_type 3 -materi \n');
fprintf(fid,'group_materi_elasti_young 3 .5581 \n');
fprintf(fid,'group_materi_density 3 1.04e-06 \n');
fprintf(fid,'group_materi_elasti_poisson 3 0.485 \n');
fprintf(fid,'group_materi_memory 3 -updated_without_rotation \n');
fprintf(fid,'\n');
fprintf(fid,'\n');
fprintf(fid,'\n');

```

```
403 %CSF
404 fprintf(fid,'group_type 4 -materi \n');
405 fprintf(fid,'group_materi_density 4  1.0e-06 \n');
406 fprintf(fid,'group_materi_elasti_compressibility 4 4.57e-10 \n');
407 fprintf(fid,'group_materi_elasti_poisson 4 0.499 \n');
408 fprintf(fid,'group_materi_memory 4 -updated_without_rotation \n');
409 fprintf(fid,'group_materi_stokes 4 -yes \n');
410 fprintf(fid,'\n');
411 fprintf(fid,'\n');
412 fprintf(fid,'\n');
413
414 %SAScsf
415 fprintf(fid,'group_type 5 -materi \n');
416 fprintf(fid,'group_materi_elasti_young 5 0.650 \n');
417 fprintf(fid,'group_materi_density 5 1.04e-06 \n');
418 fprintf(fid,'group_materi_elasti_poisson 5 0.450 \n');
419 fprintf(fid,'group_materi_memory 5 -updated_without_rotation \n');
420 fprintf(fid,'\n');
421 fprintf(fid,'\n');
422 fprintf(fid,'\n');
423
424 %Tumor
425 fprintf(fid,'group_type 6 -materi \n');
426 fprintf(fid,'group_materi_elasti_young 6 1.150 \n');
427 fprintf(fid,'group_materi_density 6 1.04e-06 \n');
428 fprintf(fid,'group_materi_elasti_poisson 6 0.350 \n');
429 fprintf(fid,'group_materi_memory 6 -updated_without_rotation \n');
430 fprintf(fid,'\n');
431 fprintf(fid,'\n');
432 fprintf(fid,'\n');
433
434 % The final component of the input file contains the control section where
435 % the evolution in time of the simulation can be controlled. Factors
436 % include time steps, iteration limits, time intervals after which a
437 % snapshot is saved, and output file formats.
438
439 fprintf(fid,'control_timestep \t 30  0.1 0.3\n');
440 fprintf(fid,'control_timestep_iterations   \t 30  1 \n');
441 fprintf(fid,'control_print    \t 30  -time_current \n');
442 fprintf(fid,'control_print_vtk \t 2445  -yes \n');
443 fprintf(fid,'end_data \n');
444
445 %Close the file. We are done here!
446 fclose(fid);
```