

12-2018

Computational Approaches to Understanding Structure-Function Relationships at the Intersection of Cellular Organization, Mechanics, and Electrophysiology

Tyler George Harvey
Clemson University, tgharve@gmail.com

Follow this and additional works at: https://tigerprints.clemson.edu/all_dissertations

Recommended Citation

Harvey, Tyler George, "Computational Approaches to Understanding Structure-Function Relationships at the Intersection of Cellular Organization, Mechanics, and Electrophysiology" (2018). *All Dissertations*. 2249.
https://tigerprints.clemson.edu/all_dissertations/2249

This Dissertation is brought to you for free and open access by the Dissertations at TigerPrints. It has been accepted for inclusion in All Dissertations by an authorized administrator of TigerPrints. For more information, please contact kokeefe@clemson.edu.

COMPUTATIONAL APPROACHES TO UNDERSTANDING STRUCTURE-
FUNCTION RELATIONSHIPS AT THE INTERSECTION OF CELLULAR
ORGANIZATION, MECHANICS, AND ELECTROPHYSIOLOGY

A Dissertation
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy
Bioengineering

by
Tyler George Harvey
December 2018

Accepted by:
Dr. Delphine Dean, Committee Chair
Dr. Bruce Z. Gao
Dr. Jiro Nagatomi
Dr. William Richardson

ABSTRACT

The heart is a complex mechanical and electrical environment and small changes at the cellular and subcellular scale can have profound impacts at the tissue, organ, and organ system levels. The goal of this research is to better understand structure-function relationships at these cellular and subcellular levels of the cardiac environment. This improved understanding may prove increasingly important as medicine begins shifting toward engineered replacement tissues and organs. Specifically, we work towards this goal by presenting a framework to automatically create finite element models of cells based on optical images. This framework can be customized to model the effects of subcellular structure and organization on mechanical and electrophysiological properties at the cellular level and has the potential for extension to the tissue level and beyond.

In part one of this work, we present a novel algorithm is presented that can generate physiologically relevant distributions of myofibrils within adult cardiomyocytes from confocal microscopy images. This is achieved by modelling these distributions as directed acyclic graphs, assigning a cost to each node based on observations of cardiac structure and function, and determining to minimum-cost flow through the network. This resulting flow represents the optimal distribution of myofibrils within the cell. In part two, these generated geometries are used as inputs to a finite element model (FEM) to determine the role the myofibrillar organization plays in the axial and transverse mechanics of the whole cell. The cardiomyocytes are modeled as a composite of fiber trusses within an elastic solid matrix. The behavior of the model is validated by

comparison to data from combined Atomic Force Microscopy (AFM) and Carbon Fiber manipulation. Recommendations for extending the FEM framework are also explored.

A secondary goal, discussed in part three of this work, is to make computational models and simulation tools more accessible to novice learners. Doing so allows active learning of complicated course materials to take place. Working towards this goal, we present CellSpark: a simulation tool developed for teaching cellular electrophysiology and modelling to undergraduate bioengineering students. We discuss the details of its implementation and implications for improved student learning outcomes when used as part of a discovery learning assignment.

DEDICATION

This work is dedicated to my family. To my wife Julia, whose love, support, and encouragement has been unwavering. To my mother Heike, for instilling a love of learning, encouraging my curiosity, and always believing in me. And to Polly, for keeping me company during many of the long hours it took to make this dissertation a reality.

ACKNOWLEDGMENTS

First and foremost, I would like to thank my advisor and mentor, Dr. Delphine Dean. Words cannot express the gratitude I have for the nearly a decade of advice, encouragement and wisdom I've gained while working alongside you. I would also like to thank my committee members: Dr. Bruce Gao, Dr. Jiro Nagatomi, and Dr. Will Richardson who helped guide the course of my research. Additionally, I would like to thank Dr. Brian Dean, whose ability to make even the most daunting problems seem simple made much of this work possible and Dr. Scott Wood, whose doctoral work inspired my own.

I would like to thank Dr. Aesha Desai, Nardine Ghobrial, and all past and present members of the Multiscale Bioelectromechanics Lab for their contributions to my work, Dr. Martine LaBerge and the Department of Bioengineering for the opportunity and support to complete my degree, and especially Maria Torres, for her unending kindness and support.

I would like to acknowledge Dr. Peter Kohl and Lucas Schmidt for providing the confocal images and experimental data used in my work, MOSEK Inc, for allowing free-use of the MOSEK Optimization Toolbox, and the National Science Foundation for their financial support of this project (NSF CAREER CBET 1254609). I would also like to acknowledge the Summer Program for Research Interns (SPRI) initiative between the Calhoun Honors College and the South Carolina Governor's School for Science and Mathematics (SCGSSM) for their supplemental funding and Emily Fast, the research intern who worked on this project.

TABLE OF CONTENTS

	Page
TITLE PAGE	i
ABSTRACT	ii
DEDICATION	iii
ACKNOWLEDGMENTS	iv
LIST OF TABLES	vii
LIST OF FIGURES	viii
CHAPTER	
1. INTRODUCTION AND SPECIFIC AIMS.....	1
1.1 Motivation.....	2
1.2 Specific Aims.....	3
1.3 Significance.....	4
1.4 Background.....	5
1.5 References.....	8
2. ALGORITHMIC ESTIMATION OF MYOFIBRIL DISTRIBUTIONS IN ADULT CARDIOMYOCYTES	9
2.1 Literature Review.....	10
2.2 Methods.....	16
2.3 Results.....	24
2.4 Discussion.....	29
2.5 Conclusions.....	31
2.6 References.....	32
3. FINITE ELEMENT MODELLING TO UNDERSTAND THE ROLE OF SUBCELLULAR STRUCTURES ON WHOLE-CELL BEHAVIOR.....	34
3.1 Literature Review.....	35
3.2 Methods.....	41
3.3 Results.....	47
3.4 Discussion.....	58
3.5 Conclusions.....	60
3.6 References.....	60

Table of Contents (Continued)

	Page
4. SINGLE-CELL ELECTROPHYSIOLOGICAL MODELS AS TOOLS IN ENGINEERING EDUCATION	70
4.1 Introduction and Background	70
4.2 Methods.....	78
4.3 Results.....	84
4.4 Discussion.....	90
4.5 Conclusions.....	94
4.6 References.....	95
5. CONCLUSIONS AND RECOMMENDATIONS FOR FUTURE WORK.....	97
5.1 Conclusions.....	97
5.2 Recommendations for Future Work.....	98
APPENDICES	101
A: MATLAB Code for Generating Myofibril Distributions	102
A.1: cell6.m.....	102
A.2: Import_Confocal_Stack_sep.m.....	105
A.3: estimateBinaryVolume_2.m	105
A.4: estimateBinaryArea_2.m	105
A.5: regImage.m	106
A.6: regImage2.m	106
A.7: paths_nosweep.m	107
A.8: fitcylinder.m.....	113
A.9: generateNodes3.m.....	114
A.10: findAttachments2.m.....	116
A.11: scaleDemand.m.....	117
A.12: generateEdges3.m	117
A.13: generateSideView.m	118
A.14: matrix2ft.m.....	118
B: MATLAB Code for CellSpark.....	119
B.1: CellSpark.m.....	119
B.2: settings.m	137
B.3: run_simulation.m	142
B.4: Step.m.....	153
B.5: StepN.m.....	156

Table of Contents (Continued)

	Page
B.6: Variables.m	160
B.7: VariablesN.m	161
C: CellSpark Related Course Materials	162
C.1: Electrophysiology Lecture Slides	162
C.2: Laboratory Tutorial Exercise	166
C.3: Midterm Lab Assignment Prompt.....	170
C.4: Midterm Lab Assignment Grading Rubric	171
D: CellSpark Survey and Informed Consent Document.....	172
D.1: CellSpark Survey	172
D.2: Informed Consent Document.....	173
D.3: Participant Recruitment Prompt.....	175

LIST OF TABLES

Table		Page
2.1	Algorithm results for each sample cell and node spacing case.....	29
3.1	Material properties for each component of the COMSOL models generated	49
3.2	Summary of model complexity for the four COMSOL models generated	54
4.1	Selected ABET criteria for accredited programs in engineering.....	77
4.2	Statistical analysis of assignment grades (analyzed papers vs all papers)	94

LIST OF FIGURES

Figure	Page
1.1 Cardiac Excitation-Contraction coupling mechanism	6
1.2 Intercalated disc composition and mechanical coupling mechanism	7
2.1 Structure of striated muscle	10
2.2 A Directed Acyclic Graph (DAG)	14
2.3 Schematic representation of the myofibril generation model	19
2.4 Optimal circle in circle packing of $N = 2,3,4,5$ and 7 unit circles	22
2.5 Interpolation curve of optimal packing densities based on unit and enclosing circle radii ratio	23
2.6 Orthogonal views of isolated membrane and nuclei isosurfaces	24
2.7 Visual representation of cylinder approximation method.....	25
2.8 Visual representation of the node network for a sample cell.....	26
2.9 Vizualization of the myofibril generation process.....	27
2.10 Comparison of original image and generated fiber distributions.....	28
3.1 Simple element models of viscoelasticity.....	37
3.2 The Generalized Maxwell model of viscoelasticity.....	38
3.3 Schematic of the principle of Tensegrity	39
3.4 A unidirectional composite exhibites transverse isotropy	40
3.5 Overview of cell mechanics measurement techniques	43
3.6 Schematic representation of AFM method	44

List of Figures (Continued)

Figure	Page
3.7 Overview of the CF manipulation method.....	46
3.8 Visualization of combination CF/AFM technique.....	51
3.9 Model geometries visualized in COMSOL.....	53
3.10 Visualization of axial stress in models with variable fiber thickness	55
3.11 Comparison of axial force and elastic moduli for models with variable fiber thickness.....	56
3.12 Deformed mesh and axial stress in truss elements for each applied stretch.....	58
3.13 Axial force vs sarcomere length – experimental data.....	60
3.14 Axial force vs sarcomere length – simulation data.....	60
3.15 Axial force vs sarcomere length (1% binning) – experimental	61
3.16 Axial force vs sarcomere length (1% binning) – simulation	61
3.17 Apparent elastic modulus vs sarcomere length - experimental	63
3.18 Approximated elastic modulus vs sarcomere length - simulation	63
4.1 Circuit diagram of the Hodgkin-Huxley model.....	71
4.2 Action potential at various cardiac cycle frequencies.....	73
4.3 Learning objectives for using simulations in teaching engineering.....	76
4.4 Overview of CellSpark software dependencies	80
4.5 CellSpark software interface.....	81
4.6 Combined responses to survey items.....	87

CHAPTER ONE

INTRODUCTION AND SPECIFIC AIMS

1.1 Motivation

My long-term research goal is to improve the understanding of how changes in cardiovascular structure at the cellular and subcellular can impact the mechanical and electrical function at the tissue level. In pursuit of this, the objective of this dissertation is to develop a framework through which to model the unique electromechanical properties of cardiac cells and tissues. To this end, the research developed methods to observe, measure, and estimate information about the geometry, mechanics, and electrical characteristics of cells at the tissue, cellular, and subcellular level and incorporate this data into a Finite Element Modelling (FEM) multi-physics package to build useful models with varying levels of complexity. The research employs various microscopy and image processing techniques to observe and measure geometries at the cellular and tissue levels, but electromechanical function is inherently linked to the subcellular structure as well. Novel computational methods were employed to estimate geometries at this scale that are below the practical resolution of the microscopy. Validation of these models was by performed by comparison to experimental cell mechanics methods. The framework developed allows a simple method to create cardiac cellular models that can be used to assess the effects of subcellular structure changes on the cellular and tissue level properties. Additionally, the models can be used to see how tissue level structural changes affect electromechanical function in environments which are difficult to study *in vivo* and difficult to recreate *in vitro*, such as the post-infarct heart.

Following myocardial infarction, damaged cardiac tissue undergoes remodeling in an attempt to repair and strengthen the heart. As a part of this remodeling process, myocytes change size and shape, and there is an infiltration of fibroblasts into the myocardial space¹. This drastic change in the tissue structure leads to changes in the electrical and mechanical function, the extent of which are not fully understood.²⁻⁴ Because of the difficulty of studying this environment *in vivo* or adequately recreating it in tissue culture, the long term goal of this project is to develop a computational platform to model the mechanics of this, and other unique cardiac environments at the tissue level scale.

My long-term educational goals are to increase undergraduate bioengineering students' exposure to topics in computer science, electrical engineering, and engineering design to improve breadth of knowledge and sense of engineering identity.

1.2 Specific Aims

1.2.1 Aim 1: Algorithmic Estimation of Myofibril Distributions in Adult Cardiomyocytes

The goal of this aim is to create a novel algorithmic method to estimate myofibrils in adult cardiomyocytes based on confocal images of the cell. This is achieved by modelling these distributions as directed acyclic graphs, assigning a cost to each node based on observations of cardiac structure and function, and determining to minimum-cost flow through the network using the Network Simplex algorithm. This resulting flow represents the optimal distribution of myofibrils within the cell.

1.2.2 Aim 2: Finite Element Modeling to Understand the Role of Subcellular Structures on Whole-Cell Behavior

The goal of this aim is to use the geometries generated in aim 1 as inputs to a finite element model (FEM) to determine the role the myofibrillar organization plays in the axial and transverse mechanics of the whole cell. The cardiomyocytes are modeled in COMSOL Multiphysics as a composite of fiber trusses within an elastic solid matrix. The behavior of the model is validated by comparison to data from combined Atomic Force Microscopy (AFM) and Carbon Fiber manipulation.

1.2.3 Aim 3: Single-Cell Electrophysiological Models as Tools in Engineering Education

The goal of this aim is to develop and implement CellSpark: a simulation tool developed for teaching cellular electrophysiology and modelling to undergraduate bioengineering students. We discuss the details of its implementation and implications for improved student learning outcomes when used as part of a discovery learning assignment. This software makes these computational more accessible to novice learners and allows active learning of these complicated course materials to take place

1.3 Significance

Two of the grand challenges of engineering, as identified by the National Academy of Engineering, are to engineer better medicines and to engineer the tools of scientific discovery. As technology and computing advances, a powerful tool is at our disposal as engineers to help make the transition from benchtop to bedside much faster through the use of computational biology. This work will help to develop a computational

framework, one of these tools of scientific discovery that will help bridge gaps in knowledge between cardiac properties at the single cell and tissue levels, 2D to 3D levels, and in vitro and in vivo levels. This framework makes it possible to predict mechanical and electrical behavior of tissue constructs, of vital importance for tissue and organ engineering technologies.

Medicine is progressively moving to smaller scales, with treatments shifting from systemic macroscale approaches to targeted, cellular and molecular ones. At this small scale, not only is observation and assessment more difficult, but the effects of changes at the higher, macroscopic levels is not as well studied or understood. Development of this computational platform and its use to assess the impact of the small scale structural and functional changes will allow future research endeavors to progress faster, and with less time and capital investment than traditional in vitro preliminary testing. This multidisciplinary research also allows for improved collaboration between computer scientists, who possess these powerful computational tools but lack the background to even realize the extent of problems they could be solving, and bioengineers – especially those still early in their training – who often have a reluctance to expand outside their comfort zones into the realms of computing and simulation. The educational goals of this proposal seek to start bridging this gap, to help bioengineering undergraduates become well rounded and confident engineers.

1.4 Background

Anatomy and Physiology:

At the cellular level, cardiac tissue is composed of specialized muscle cells referred to as cardiomyocytes. Cardiac muscle, while possessing qualities similar to both skeletal muscle (striations due to bands of actin and myosin) and smooth muscle (involuntary activation), it has two main characteristics which make it both structurally and electrophysiologically unique within the body: t-tubules and intercalated discs.⁵

T-tubules are the primary structures responsible for excitation-contraction coupling. As the cell is excited following an electrical stimulus, a wave of depolarization travels across the membrane and into t-tubules. Here, the depolarization causes voltage gated calcium ion channels to open allowing extracellular Ca^{2+} to diffuse into the cell and initiate a positive feedback loop to release additional Ca^{2+} stored in the sarcoplasmic reticulum into the cytoplasm.⁶ Cytoplasmic calcium, when in sufficient quantities binds to the Troponin C, exposing the active site of actin. With actin now exposed, the classical “sliding filament” model of contraction between the actin and myosin filaments takes place, contracting the entire muscle. Following contraction, ATP binding releases the calcium from Troponin-C and ion pumps restore the previous levels of cytoplasmic calcium, thus repolarizing the cell.⁷

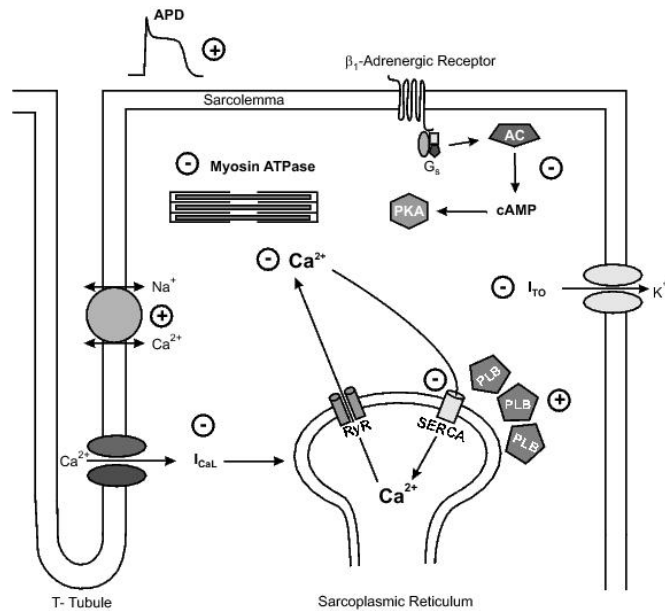


Figure 1.1: Mechanism of Excitation-Contraction coupling of cardiomyocytes.⁶

Intercalated discs are the “glue” which hold the heart together – at least electrically and mechanically. These discs are the site of connection between adjacent cardiomyocytes and allows them to form a functional syncytium.⁵ Discs are composed of three parts: actin anchor points (fascia adherens) which allow internal stresses to be transferred to the sarcomeres of the adjacent cell, intermediate filament anchor points (desmosomes) which physically attach the cells together and let external stresses be transferred between them, and gap junctions which allow intersarcoplasmic ion flow and enable action potential propagation down the syncytium.

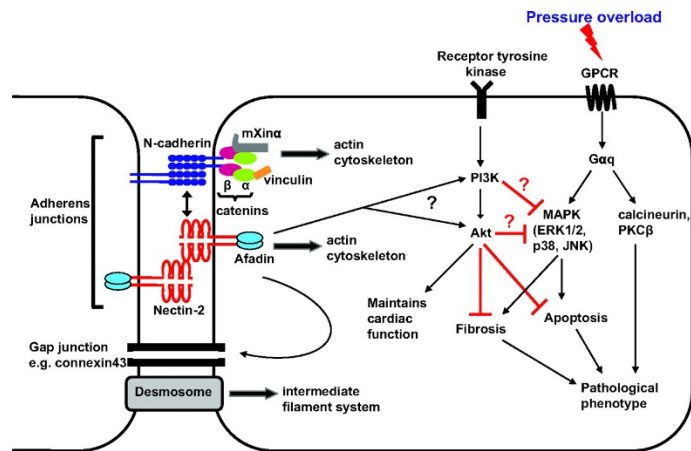


Figure 1.2: Intercalated disc composition and mechanism of mechanical coupling.⁷

Pathology:

Abnormal electromechanical behavior of the heart can be generally be broken down into two main problems: pathology of the conduction circuit itself and necrosis or other tissue damage with results in mechanical failure of the functional syncytium. The former cause usually manifests as one of several arrhythmias, interrupting the normal cardiac cycle.⁸ Common arrhythmias can include extra beats, which are usually fairly benign, tachycardias (elevated pacing) and fibrillation, the fast irregular pacing which leads to both stroke and heart failure.⁸

The latter cause, damage to the cardiomyocytes themselves, can be caused by a number of things including trauma, chronic or prolonged ischemia, and myocardial infarction. The resulting damage can cause mechanical weakening of the heart muscle, reduced circulatory capacity, arrhythmias, and enlarging/remodeling of the heart – in general an overall loss of proper mechanical performance.⁹

1.5 References

1. Turner NA, Porter KE. Function and fate of myofibroblasts after myocardial infarction. *Fibrogenes Tissue Repair*. 2013;6(1):1. doi:10.1186/1755-1536-6-5.
2. Van Breemen VL. Intercalated discs in heart muscle studied with the electron microscope. *Anat Rec*. 1953;117(1):49-63. doi:10.1002/ar.1091170106.
3. Barr L. Propagation of Action Potentials and the Structure of the Nexus in Cardiac Muscle. *J Gen Physiol*. 1965;48(5):797-823. doi:10.1085/jgp.48.5.797.
4. Goshima K, Tonomura Y. Synchronized beating of embryonic mouse myocardial cells mediated by FL cells in monolayer culture. *Exp Cell Res*. 1969;56(2-3):387-392. doi:10.1016/0014-4827(69)90029-9.
5. Martini, F & Nath J. *Anatomy and Physiology*. New York: Pearson; 2009.
6. Fabiato A. Calcium-induced release of calcium from the cardiac sarcoplasmic reticulum. *Am J Physiol Physiol*. 1983;245(1):C1-C14. doi:10.1152/ajpcell.1983.245.1.C1.
7. Bu G, Adams H, Berbari EJ, Rubart M. Uniform action potential repolarization within the sarcolemma of in situ ventricular cardiomyocytes. *Biophys J*. 2009;96(6):2532-2546. doi:10.1016/j.bpj.2008.12.3896.
8. National Heart, Lung and BI-N. Arrhythmia. nhlbi.nih.gov/health/health-topics/topics/arr/.
9. Kovanen PT. Mast cells and degradation of pericellular and extracellular matrices: potential contributions to erosion, rupture and intraplaque haemorrhage of atherosclerotic plaques: Figure 1. *Biochem Soc Trans*. 2007;35(5):857-861. doi:10.1042/BST0350857.

CHAPTER TWO

ALGORITHMIC ESTIMATION OF MYOFIBRIL DISTRIBUTIONS IN ADULT CARDIOMYOCYTES

Cellular mechanics are often simplified to basic constitutive models, such as elastic¹⁻³ or poroelastic⁴ solids. While these models can often predict mechanical properties or behavior of normal cells relatively accurately, they do not account for cytoskeletal structure. Since cellular architecture is constantly remodeling⁵, whether the result of pathology or changes in the needs of the organism, these models are insufficient to predict properties and behaviors during or after remodeling and may be insufficient at different levels of model complexity.

Cytoskeletal organization itself is difficult to measure, especially at the tissue level and above, where imaging techniques do not have sufficient resolution to visualize subcellular components. To solve this problem, we are interested in instead using cellular images as an input and using computational methods to create models of cytoskeletal architecture that can then improve cellular and tissue mechanics models.

In this chapter, we present one such technique to estimate likely distributions of myofibrils in cardiomyocytes by approximating the scenario as a thick non-crossing paths problem. We present an approximate solution by representing the cell volume as a directed acyclic graph and computing the minimum cost-flow through the network. The collection of paths that results approximates the distribution of myofibrils within the cell.

2.1 Literature Review

2.1.1 Myocyte Structure and Function

Cardiomyocytes are the cell that acts as the building block of cardiac tissue, accounting for nearly 75% of healthy myocardium⁶. Compared to other cells present in the heart, such as cardiac fibroblasts, myocytes are large and they account for this volume despite only being 30-40% of the cells present^{5,7}. Myocytes are roughly cylindrical, about 100 μm long, with diameters in the range of 10-25 μm , though branching and other non-cylindrical shapes can be observed. Myocytes contain the same cytoskeletal elements as other mammalian cells (F-actin, microtubules)^{8,9} with the notable addition of bundles of myofibrils, the primary contractile organelles¹⁰.

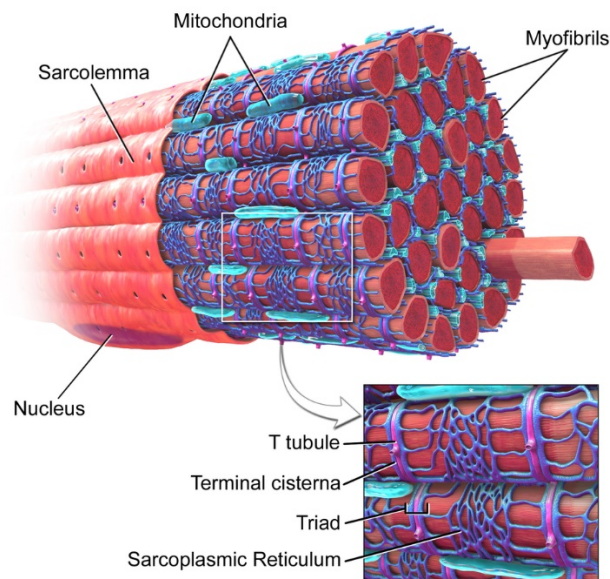


Figure 2.1: Structure of striated muscle. The myocyte is essentially a bundle of smaller contractile fibers called myofibrils.²³

Myofibrils are chains of sarcomeres, the smallest contractile unit of striated muscle. Each sarcomere consists of two groups of actin filaments bound to Z-disks at opposite ends of the sarcomere.⁸ Between each pair of actin filaments is a myosin filament, which bridges the gap (called the H-zone) and attaches to both Z-disks through the giant protein titin. Each side of the Z-disk contains a mirror image of these filament attachments, which is how the sarcomeres are chained together into a myofibril. The prevailing theory of muscle contraction, the sliding filament theory, posits that the interweaved actin and myosin filaments slide past each as ATP hydrolysis of the myosin causes reversible binding of the myosin heads to the actin filaments.⁴

2.1.2 Myofibril Distribution

As myofibrils are the contractile element of the myocyte, their distributions within the cell dictate the overall contractility of the cell. The mechanisms that determine these distributions have been extensively studied. There has also been interest in modeling these distributions for reasons similar to our own, but most of these models¹¹⁻¹³ are based upon reaction kinetics of the contractile subunits. In this way they are more accurately modelling the formation and development of muscles.

At least one image-based approach to automatic modelling these distributions in myocytes has been attempted¹⁴ by segmenting out individual A-bands from high resolution phase contrast images of striated muscle and developing a path growing algorithm to chain them together into myofibrils. Though successful, this approach is limited to high resolution images and a two-dimensional image which make it non-ideal as an approach for our goal of extending our model to the tissue and organ level.

2.1.3 Packing Problems

This question of determining a specific organization of subcellular components within the finite volume of the cell is a member of a broader class of problems in mathematics known as packing problems.

In the simplest form the question of their distribution within the cellular volume is given by the Pencil-Packing Problem¹⁵. A three-dimensional grid of voxels, some of which are occupied with obstacles, is packed with “pencils” – unions of adjacent voxels that form an axis-parallel strip. Only one pencil can be packed into a given voxel and cannot be packed into voxels occupied by obstacles. The length of the pencils can be fixed or variable. In this simplification, myofibrils are assumed to have a square cross-sectional area equal to one voxel by one voxel, and bending is restricted. The other limitation is that myofibrils can only be oriented along the major cartesian directions.

The next logical extension of the problem is to remove the bending restriction and add endpoints in the domain that should be connected to each other. The problem can now be considered a member of a separate class: shortest path problems¹⁶. The simplest of these problems seeks to find the shortest path between two points in a polygonal domain which also contains obstacles. This problem can be extended to the non-crossing paths problem by finding a collection of paths between multiple sets of points such that the paths do not intersect each other or the obstacles and that the distribution of paths is optimized in some way. In this problem specifically, the paths are “thick” since the myofibrillar cross section is non-zero. Thick non-crossing paths problems, where thick

paths are defined as the Minkowski sum of a curve and a disk, are common in other disciplines such as integrated circuit design and air traffic management.^{17,18}

2.1.4 Nodes, Networks, and Flows

Our proposed approach to solving the thick non-crossing path problem of myofibril distribution within cardiomyocytes is by representing the internal volume of the cell as a graph. A graph is a collection of objects and the interconnections among them. The objects in the graph are typically called vertices or nodes and the connections between nodes are called edges or arcs. Both the nodes and edges can be described by any number of characteristics (such as coordinate position) but at minimum are described by their node/edge relationships.¹⁹

All edges connect two nodes, but this connection can be either undirected, where information can move between nodes in either direction, or directed where it is only possible in one direction. Directed graphs are commonly abbreviated to digraphs. The other important characteristic of graphs is cyclicity. A cyclic graph is one where one can trace from a specific node along edges to other nodes and eventually arrive back at the starting node. An acyclic graph is one where this tracing is not possible. Cyclicity is independent of direction and in this work we will specifically model the cellular volume with graphs that are both acyclic and directed – so called Directed Acyclic Graphs or DAGs.¹⁹

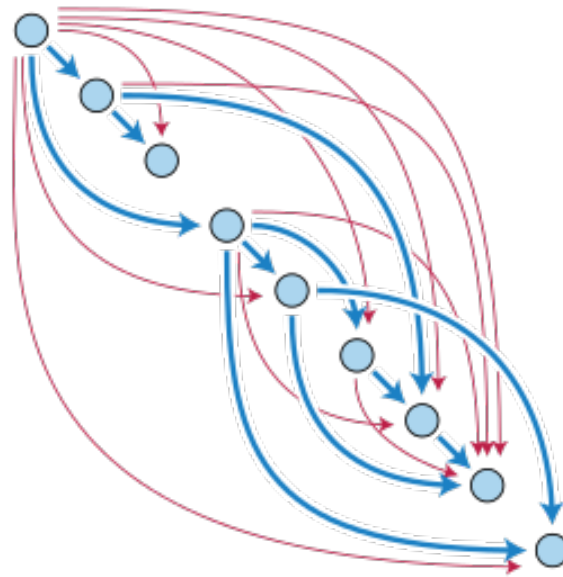


Figure 2.2: A Directed Acyclic Graph (DAG). Nodes are represented by the blue circles and the edges are represented as arrows.²⁴

A flow graph is a specific digraph whose edges represent the coefficients of algebraic or differential equations of the nodes and a flow network is a specific flow graph whose edges represent gains or losses of flow. In more concrete terms, each edge of the flow network has a capacity for the amount of flow it is capable of receiving and every node in the network must conserve flow – that is the flow into and out of the network must be equal – with the exception of sources and sinks, where flow originates and terminates, respectively. The most common problem associated with flow networks are ones in which the maximum flow through the network is determined. This has applications in transportation, logistics, and communications. If an additional cost constraint is placed on edges, analogous to a resistance to flow, then another class of Minimum-Cost Maximum-Flow problems is created. These types of problems allow the resulting node disjoint paths through the network to be optimized by not only

connectivity (like a Maximum-Flow problem) but also by any additional quality, such as, in our case, position.²⁰

2.1.5 The Primal-Simplex Algorithm

Minimum-Cost Maximum-Flow problems are NP-hard but, as linear programs, are solvable by multiple different optimization algorithms, including the Primal Simplex Algorithm. Let the Minimum-Cost Flow problem is linearized as:

$$\begin{aligned} \mathbf{Minimize} \quad & z = \mathbf{c}x \\ \mathbf{subject\ to} \quad & \mathbf{A}x = \mathbf{b} \\ & \mathbf{0} \leq x \leq \mathbf{u} \end{aligned}$$

where \mathbf{A} is the (sparse) incidence matrix defining permissible edges, \mathbf{b} is a vector giving the demand of each node, \mathbf{c} is a matrix defining the cost of each edge, and \mathbf{u} is capacity constraint, defining the maximum permissible flow on each edge (in our formulation, this is equal to 1 for every permissible edge.)

Then the solution can be found by first computing the basic solutions of the form:

$$\mathbf{x}_B = \mathbf{B}^{-1}[\mathbf{b} - \mathbf{N}_1\mathbf{u}_1].$$

where \mathbf{B} is the basis vector and \mathbf{N}_1 is a matrix containing the columns of \mathbf{A} associated with the non-basis variable set \mathbf{n}_1 such that:

$$\mathbf{B}\mathbf{x}_B + \mathbf{N}_0\mathbf{x}_0 + \mathbf{N}_1\mathbf{x}_1 = \mathbf{b}.$$

Next, the algorithm optimizes a solution by iteratively solving the computation

$$\mathbf{y}_k = \mathbf{B}^{-1}\mathbf{a}_k = \mathbf{p}_i - \mathbf{p}_j$$

where \mathbf{p}_i and \mathbf{p}_j are the columns of \mathbf{B}^{-1} describing the paths from the source to nodes i and j respectively, and \mathbf{a}_k is the column of \mathbf{A} for the entering node.²⁰

2.2 Methods

2.2.1 Image processing

The images provided for use in this study were of primary ventricular cardiac myocytes obtained from female Sprague Dawley rats. The myocytes were fixed using 4% paraformaldehyde, permeabilized with Triton-X, and fluorescently stained with phalloidin and DAPI to visualize actin and the nuclei, respectively. Cells were imaged using an Olympus PLAPON60XO 60x oil immersion objective (NA=1.42) on an Olympus IX81 inverted microscope with a DSU spinning confocal disc and a Hamamatsu ImagEM CCD camera (Hamamatsu Photonics K.K., Hamamatsu City, Japan). Additionally, three images of Human ventricular myocytes, obtained using the same protocol on comparable equipment were included as a comparison.

The red, green, and blue channels of each image were imported into MATLAB r2016b (The MathWorks Inc., Natick, MA, USA) as separate 3-D grayscale matrices. The empty green channel was discarded and each of the red and blue matrices was converted to a binary (black-and-white) matrix using an empirically determined 15% intensity threshold. The matrices were rescaled using bicubic interpolation such that each element of the matrix represented a $1\mu\text{m} \times 1\mu\text{m} \times 1\mu\text{m}$ cellular volume.

After being rescaled, the centroid and major axis of the cell was determined by finding Q1, Q2 (centroid) and Q3 of the (x,y,z) coordinates of the red channel binary volume matrix. Both matrices were rotated using nearest-neighbor interpolation about the centroid such that the major axis of the cell was aligned with the y-axis (second dimension of the matrix.)

“Holes” in the interior of the binary volume were filled and surface artifacts introduced by the thresholding, rescaling, and registration processes were mitigated by smoothing using a gaussian filter with size 3 pixels and standard deviation 1 pixel followed by image dilation with a spherical structuring element of radius 1 pixel. The membrane surface was captured by extracting the 1% isosurface using the marching-cubes algorithm²¹ from the red channel matrix and exporting it as a stereolithography (STL) file containing vertices and normal vectors of triangular faces which define the surface. The same method was used to determine the nuclei surfaces using the blue channel matrix.

2.2.2 Generation of Representative Myofibrils: A Subcellular Min-Cost Flow Problem

In this section, we describe a novel algorithmic technique to estimate the fiber geometries, based on the geometric information known about the cell membrane and nuclei. We make the following assumptions about myofibril distribution within a myocyte:

- 1) The cell membrane mainly serves to contain all the myofibrils, so the space between the membrane and the outermost fibers is minimal.
- 2) The main function of the cardiomyocyte is mechanical contraction, so myofibrils are efficiently packed within the cell and excess sarcoplasmic space is minimal.
- 3) For a combination of the previous reasons, fibers will terminate at or close to the cell membrane, and the average fiber length will be maximized.
- 4) Muscle tissue in general, and cardiac tissue specifically, is transversely isotropic so deviations in fiber direction from the major axis of the cell will be minimal.

As these assumptions are all linear in nature (maximizing or minimizing some constraint), the problem of fiber distribution can be modeled and solved as a linear program. Here, we represent the cellular volume as a directed acyclic graph and solve for the optimal fiber distribution by determining the Minimum-Cost Flow.

Let N denote a set of nodes located on a regularly spaced grid with spacing s fully contained within and completely filling the membrane isosurface extracted in the previous section. Let E denote a set of all edges between members of N and their orthogonally and diagonally adjacent neighbors. Each edge, E_{ij} , has a length, l_{ij} of 1 (for orthogonally adjacent nodes) or $\sqrt{2}$ (for diagonally adjacent nodes). d_{ij} , the distance from the membrane isosurface, is calculated by averaging the Euclidean distance transform of the voxels nearest the nodes N_i and N_j in the binary representation of the intracellular space. This transform assigns each voxel a number representing the distance from that voxel to the nearest non-zero voxel. If the spacing between nodes is an integer, then N_i and N_j will each represent exactly one voxel.

Each edge is assigned a cost, c_{ij} , given by:

$$c_{ij} = f(d_{ij}, l_{ij})$$

where the weighting function, f , allows customization of which assumptions are prioritized. In our implementation, f was empirically chosen to be given by:

$$f(d_{ij}, l_{ij}) = \frac{3}{2} \cdot d_{ij} \cdot l_{ij}$$

Another possible simple weighting function is given by:

$$f(d_{ij}, l_{ij}) = \alpha \cdot d_{ij} + \beta \cdot l_{ij}$$

In this form, the length and distance assumptions can be independently prioritized using the weighting parameters α and β , respectively.

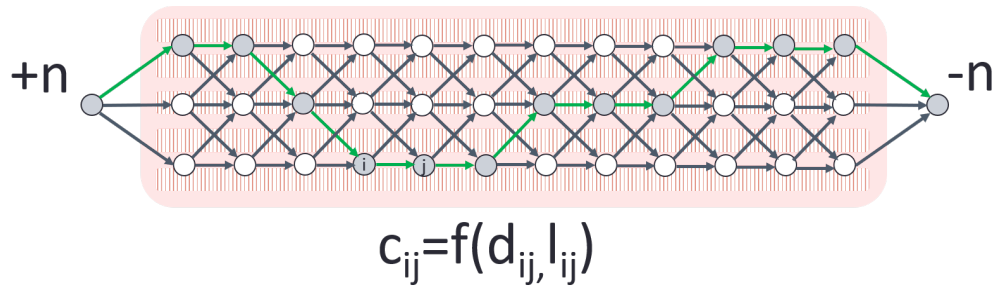


Figure 2.3: Schematic representing the model. Nodes are shown as white or grey circles (grey indicates nodes that are part of a path), and all permissible edges are shown as arrows. Terminal nodes at one end of the cell are connected to a source node with flow $+n$ and terminal nodes at the other are connected to a sink node with flow $-n$. This flow constraint, n , indicates the number of fiber paths (represented by green arrows) to be computed. The cost of including each node in a path is given by the weighting function f .

To represent our scenario as a flow problem, specific nodes representing the termination points of the fibril paths must be identified as either “sources” or “sinks” of flow. Following assumption 3, these termination points should 1) lie at or near the membrane isosurface and 2) lie at opposite ends of the major axis of the cell. To satisfy the first criteria, nodes with fewer than three non-zero neighboring voxels in the binary volume representation were eliminated as possible termination points. The second criteria required segmentation of the cell volume into sections to determine which nodes occurred near the end of the cell. This can be done in a variety of ways. In our case, this was achieved by approximating the volume as a set of end-to-end cylinders and only considering endpoints which lie the terminal cylinders.

To approximate the cellular volume as cylinders, we represent the volume as a point cloud, with each voxel represented as a single point in space. These points are grouped into two clusters using the k-means algorithm. For each cluster, the smallest cylinder which contains every point is generated. The axis of the cylinder lies along the least-squares regression line of the points. The radius is given by the maximum perpendicular distance between all the points and the regression line, and the height is given by the maximum distance between two points in the direction of the line. The fit of each cylinder is scored by dividing the number of points in the cluster by the volume of the cylinder. This score is equal to the volume of cell which lies in the cylinder since each point represents one unit voxel. This process is repeated with incrementally increasing clusters until the set of cylinders with the best fit is found.

Once the possible termination nodes have been identified, an additional edge is created between each node at one end of the cell and a master source node and each node at the other end and a master sink node. The cost of each of these edges can either be zero so that each node is equally likely to be included in a path, or some other function which allows the probability of each node being included to be further tuned. The source and sink nodes are given a demand value of positive and negative n , respectively. This demand value, n , represents the number of paths which will be placed within the cell.

The scenario presented now represents a fully defined flow problem. The minimum-cost flow through the network can be linearized in the form described in section 2.1.5. To generate our myofibril distributions, we solved this minimum-cost flow problem using the Primal Simplex algorithm as implemented in the Mosek Optimization

Toolbox (Mosek ApS, Copenhagen, Denmark) iteratively, with the number of paths to be placed incrementing until an optimal solution could no longer be found. The output of the model is a list of all the node coordinates for each path. The distributions were visualized by sweeping a circle along each path and overlaying this over the surface plot of the cell volume.

2.2.3 Validation

Performance of the model was measured with two separate criteria. The average fiber length was computed as a percentage of the total length of the cell. The percentage of cellular volume filled by the fibers was also computed. However, an ideal packing of cylinders inside a larger cylinder can never achieve a packing of 100% so in order for this metric to better represent the performance of the model, the percentage of volume filled relative to an ideal packing was determined. To compute this, each cell was approximated as the cylinder with the same length and volume of the cell and containing no nuclei. The packing of smaller cylinders, with radius s into this larger cylinder simplifies to a circle packing problem – where an integer number of unit circles are packed into a larger enclosing circle. Optimal solutions (proven or conjectured) for this problem has been found for packings up to 20 unit circles²². As the number of unit circles increases, the ratio of the enclosing circle radius to the unit circle radius also increases. An interpolation curve of this ratio vs packing density for the first 20 solutions was created and used to

roughly approximate the optimal packing density for the radius ratio of each model output.

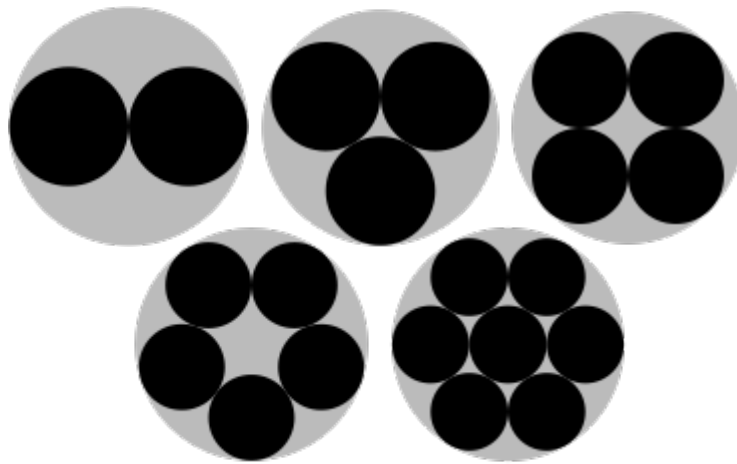


Figure 2.4: Optimal packings for the packings of $N=2,3,4,5$, and 7 unit circles into a larger enclosing circle. As the number of unit circles increases, the ratio of the radius of the enclosing circle to the ratio of the unit circle increases. For these packings, the packing density also increases, but this is not true for all N . The 7 solutions show are trivially optimal.²²

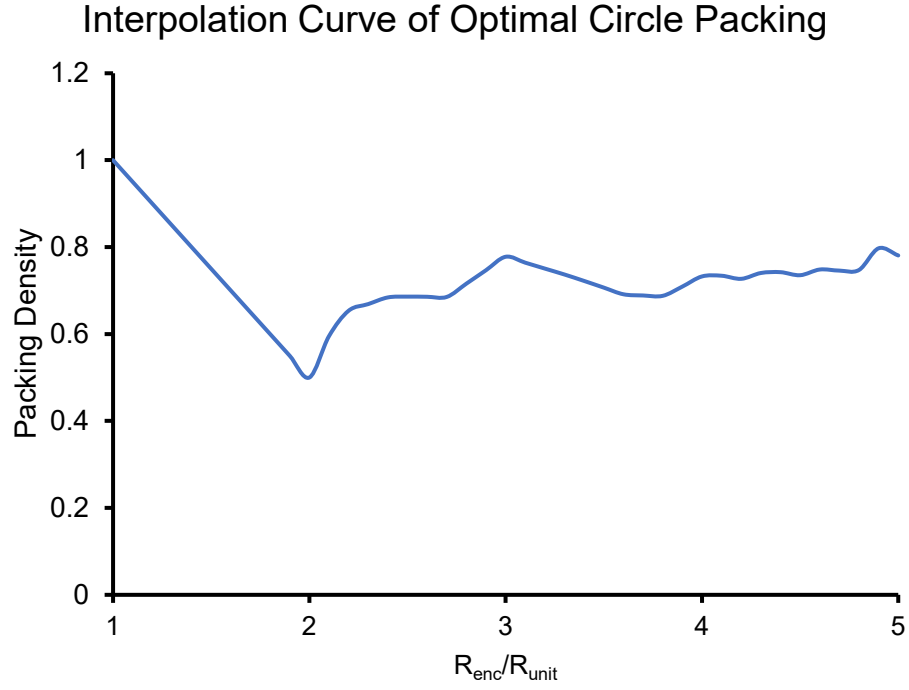


Figure 2.5: Interpolation curve of optimal packing densities for unit circles of radius R_{unit} into an enclosing circle of radius R_{enc} , based on the proven or conjectured optimal solutions for packing of up to 20 unit circles.

2.3 Results

2.3.1 Image Segmentation and Boundary Extraction

Image segmentation and boundary extraction of both the cell membrane and nuclei were successful for each sample image. The images of human cells used in this study did not contain a full z-stack all the way through the cell, due to their larger size, so the tops and bottoms of these cells are “clipped” flat. Additionally, the channel containing fluorescence of the nuclei was missing, but extraction of the nuclei was still possible from voids in the present channel. Figure 2.6 shows an orthogonal view of the extracted isosurfaces from a sample cell (Rat 3).

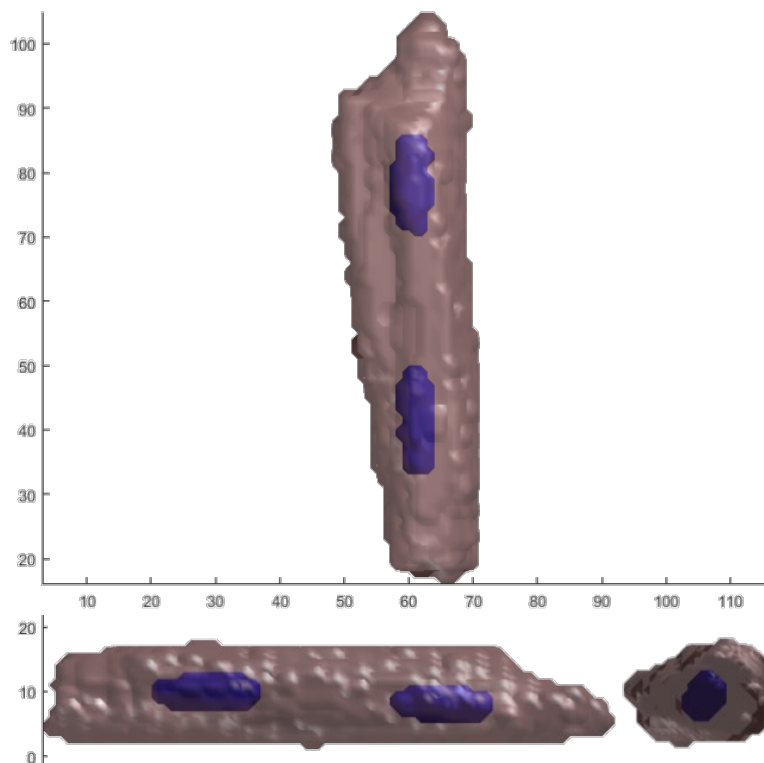


Figure 2.6: Orthogonal views of isolated membrane and nuclei isosurfaces for a sample cell (Rat 3).

2.3.2 Myofibril Distributions

Myofibril distributions were successfully generated for each cell using the algorithm outlined. Figure 2.7 shows a visualization of the cylinder approximation method used to identify the termination points for one particular cell (Rat 4). The cell geometry, point cloud clusters, and approximating cylinders are shown.

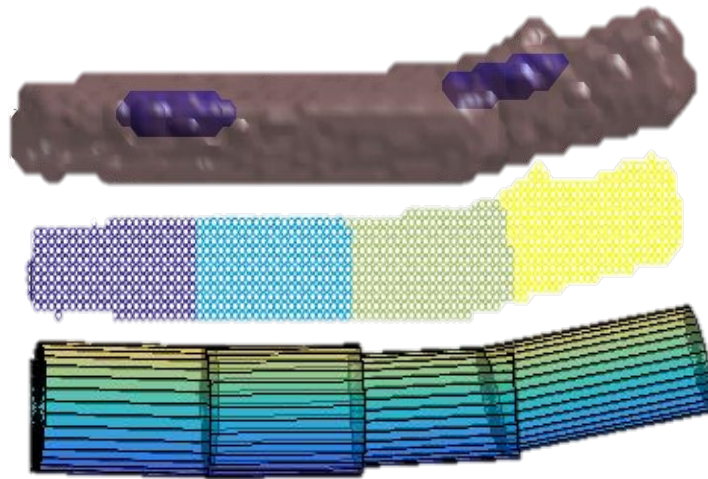


Figure 2.7: Plots of the extracted membrane and nuclei isosurfaces (top), the k-clusters of voxels used in partitioning the cell (middle), and the cylindrical approximation for identification of the fiber termination points for a sample cell (Rat 4).

Figure 2.8 shows a visualization of the algorithm's solution for a particular cell (Human 3). Overlaid on the membrane isosurface is the set of nodes generated for the cell as well as a solution of myofibril paths generated by the algorithm. Source and sink nodes (representing termination points of the myofibrils) are shown in purple and yellow respectively.

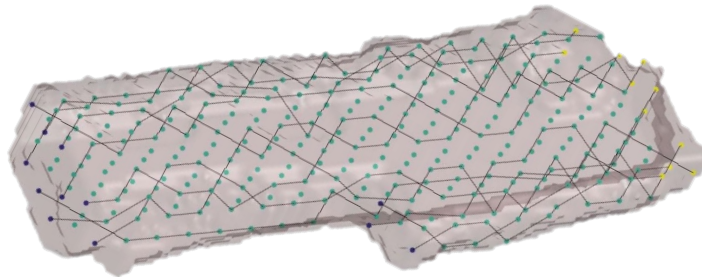


Figure 2.8: A visual representation of the node network for a sample cell (Human 3). Regular nodes are shown in teal, source nodes in purple, and sink nodes in yellow. A collection of minimum-cost flows through the network is shown as a solid edge-paths.

Figure 2.9 shows a full visualization of the algorithm's solution. The original confocal image is the input to the algorithm, and from this membrane and nuclei isosurfaces are extracted and the cellular volume is represented by a node network. The minimum cost flow through the network is determined and flow represents the paths of myofibrils through the cell. These myofibrils are visualized as thick paths with circular cross sections.

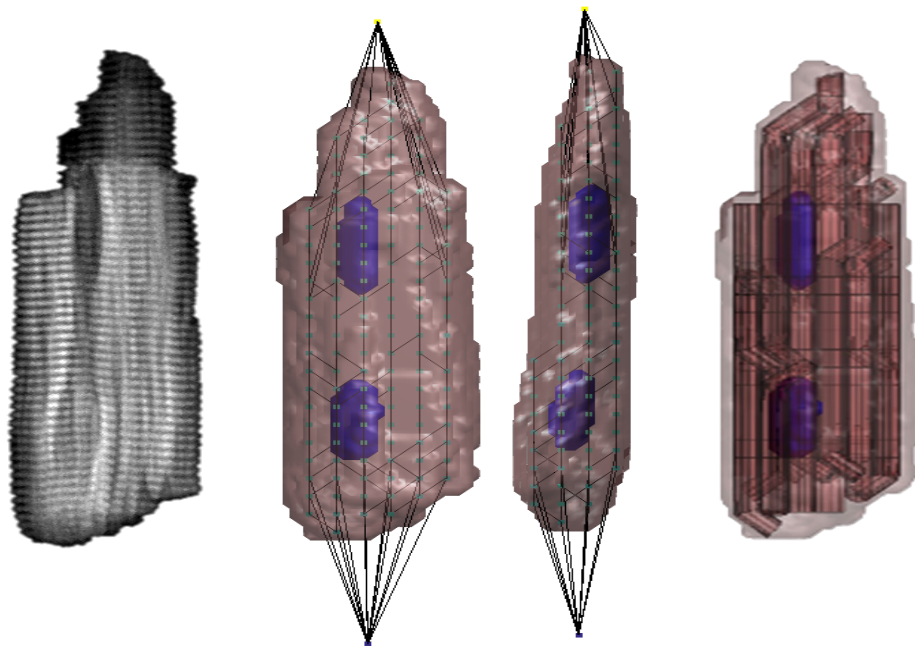


Figure 2.9: Visualization of the fiber generation process for a sample cell (Rat 2) showing the original confocal image (left), the network flow solution overlaid on the membrane and nucleus isosurfaces (middle two), and the visualized myofibrils (right). Fiber spacing = $4 \mu\text{m}$.

The sensitivity of the model to fiber thickness was determined by running the algorithm on each cell with fiber spacings of both $2 \mu\text{m}$ and $4 \mu\text{m}$. The two solutions for

a sample cell (Human 2) are shown in figure 2.10, along with the original confocal image for comparison.

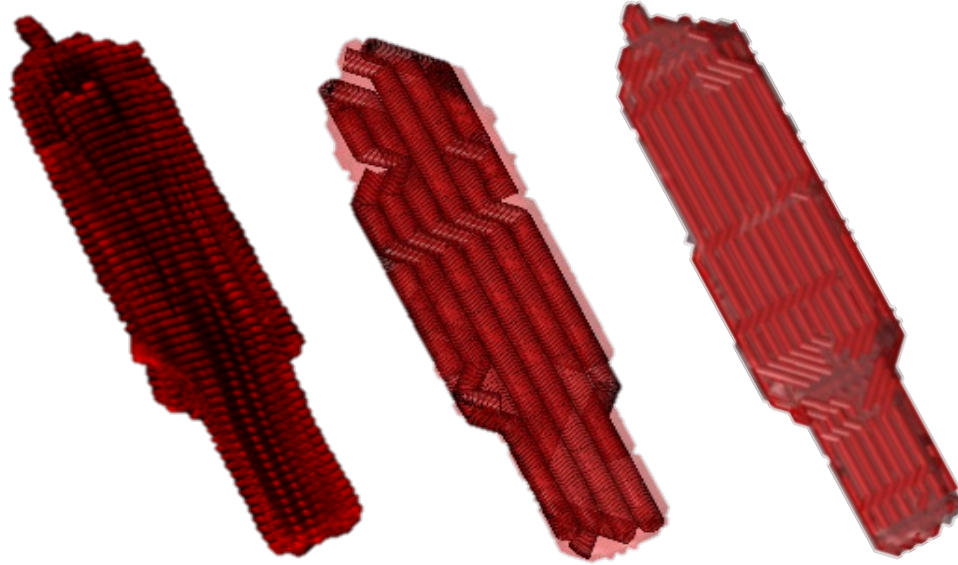


Figure 2.10: Comparison of original confocal image (left), visualized fiber distribution with spacing of $4\ \mu\text{m}$ (middle), and visualized fiber distribution with spacing of $2\ \mu\text{m}$ for a sample cell (Human 2).

2.3.3 Validation Metrics

Performance of the algorithm for each cell was determined by comparing the number of fibers placed in each cell, the average volume filled (as a percentage of “ideal packing” – see section 2.2.3 for details) and the average length of the fibers as a percentage of the cell length. These metrics, as well as the cellular volume are presented below in table 2.1 for both cases of node spacing.

Table 2.1: Algorithm results for each sample cell and node spacing case

Cell	Volume (μm^3)	Node spacing = 4 μm			Node spacing = 2 μm		
		No. of Fibers	Volume Filled*	Avg. Fiber Length	No. of Fibers	Volume Filled*	Avg. Fiber Length
Hum. 1	25,749	13	50.0%	74.0%	105	86.8%	82.7%
Hum. 2	23,715	14	61.1%	63.5%	82	91.6%	82.1%
Hum. 3	45,026	24	68.7%	80.4%	111	69.0%	94.5%
Rat 1	14,188	7	82.2%	87.9%	24	64.28%	92.7%
Rat 2	27,483	17	86.7%	72.0%	64	73.0%	78.9%
Rat 3	16,057	7	56.1%	75.3%	34	67.1%	87.2%
Rat 4	16,662	8	70.9%	78.2%	37	77.3%	86.3%

2.4 Discussion

Our results show that the choice of myofibril size is somewhat dependent on the cell used. Cells with high aspect ratios (much longer than they are wide) are better filled with thinner myofibrils. Because of the small cross-sectional areas, even a few obstacles severely limits the number of fibers that can be placed if a larger spacing is used. Filling with smaller diameter fibers tends to increase the fiber length score. This is assumed to

occur since the thinner fibers are able to penetrate farther to the terminal ends of the cell, where the cross-sectional area of the cell decreases. Conversely, the volume filled tends to drop with smaller fibers somewhat counterintuitively. While more fibers are being placed, the smaller diameters of the fibers means more wasted space between them.

Looking at figure 2.4 and considering this from the standpoint of circle packing this makes sense. As the radius of the fibers is cut in half, approximately 4 times as many fibers are packed into the same area, but these four fibers do not take up the same amount of space as a single fiber that is four times the size. The extra length gained from the smaller fibers does not appear to make up for this deficit for most cells. One other consideration is that the computational time increases exponentially with the number of fibers placed, however this can be mitigated to some degree if there is a good initial estimate of the number of fibers, since all iterating through distributions with fewer fibers is not needed.

The images of human cardiomyocytes used in this study did not capture the entirety of the cell's volume in the z-dimension. Despite this, the algorithm was still capable of generating myofibril distributions for these cells, albeit with lower volume fill scores. This lower volume filled is likely due to the cells not being very cylindrical, since the top and bottoms of the cell are not present, but the volume filled is still scored based on ideal packing in a cylindrical cell. Modifying the method of calculating the volume filled is one possible method to better capture these cells. The fact that the algorithm is still capable of working for these cells is promising that it could be implemented towards our larger goal of tissue-based models as tissue slices that are stained and imaged will likely contain many cells that are not fully imaged in the z-direction.

As a more general limitation, cells that are very non-cylindrical also require a different method to identify termination points for the fibers, since the current method also relies on approximating the cell as a set of cylinders. Some potential methods to address this include creating a probability distribution of points based on additional cell staining, such as for integrins or connexins that indicate focal adhesions and gap junctions with other cells (for tissues) or manual segmentation identification of the cell termini. While manual identification would be simple in practice for single cells, it is not ideal for studies with many cells or if the approach is scaled up for tissue constructs.

2.5 Conclusions

We have demonstrated a novel algorithmic method that allows for rapid generation of cellular geometries that can be used in other studies, such as Finite Element Analysis. This method is automatic and only based on confocal microscopy images of the cells. Since the only inputs to the algorithm are the whole cell and nuclei geometries, the method should work successfully independent of image quality, as long as the whole cell geometry can be segmented. The method can also be optimized to allow more specific control over the subcellular organization and study how organization affects other aspects of the cell such as mechanics or electrophysiological behavior. Because of the nature of the algorithm, generation of cytoskeletal structures could also be performed on custom cellular geometries, created in a Computer Assisted Design (CAD) program, for example, allowing the algorithm to be used as an unprecedented design tool for future cellular engineering studies.

2.6 References

1. Radmacher M. Measuring the elastic properties of biological samples with the AFM. *IEEE Eng Med Biol Mag.* 1997;16(2):47-57. doi:10.1109/51.582176.
2. Ding Y, Xu GK, Wang GF. On the determination of elastic moduli of cells by AFM based indentation. *Sci Rep.* 2017;7:1-8. doi:10.1038/srep45575.
3. Dokukin ME, Guz N V., Sokolov I. Quantitative study of the elastic modulus of loosely attached cells in AFM indentation experiments. *Biophys J.* 2013;104(10):2123-2131. doi:10.1016/j.bpj.2013.04.019.
4. Huxley AF, Niedergerke R. Structural Changes in Muscle During Contraction: Interference Microscopy of Living Muscle Fibres. *Nature.* 1954;173(4412):971-973. doi:10.1038/173971a0.
5. Turner NA, Porter KE. Function and fate of myofibroblasts after myocardial infarction. *Fibrogenes Tissue Repair.* 2013;6(1):1. doi:10.1186/1755-1536-6-5.
6. Camelliti P, McCulloch AD, Kohl P. Microstructured Cocultures of Cardiac Myocytes and Fibroblasts : A Two-Dimensional In Vitro Model of Cardiac Tissue. *Microsc Microanal.* 2005;11(3):249-259. doi:10.1017/S1431927605050506.
7. Camelliti P, Green CR, Kohl P. Structural and Functional Coupling of Cardiac Myocytes and Fibroblasts. *Adv Cardiol.* 2006;42:132-149. doi:10.1159/000092566.
8. Sarantitis I, Papanastasiopoulos P, Manousi M, Baikoussis NG, Apostolakis E. The cytoskeleton of the cardiac muscle cell. *Hell J Cardiol.* 2012;53(5):367-379.
9. Aquila LA, McCarthy PM, Smedira NG, Young JB, Moravec CS. Cytoskeletal structure and recovery in single human cardiac myocytes. *J Hear Lung Transplant.* 2004;23(8):954-963. doi:10.1016/j.healun.2004.05.018.
10. Severs NJ. The cardiac muscle cell. *BioEssays.* 2000;22:188-199.
11. Grosberg A, Kuo P-L, Guo C-L, et al. Self-Organization of Muscle Cell Structure and Function. Crampin EJ, ed. *PLoS Comput Biol.* 2011;7(2):e1001088. doi:10.1371/journal.pcbi.1001088.
12. Drew NK, Eagleson MA, Baldo Jr. DB, Parker KK, Grosberg A. Metrics for Assessing Cytoskeletal Orientational Correlations and Consistency. Zaman M, ed. *PLOS Comput Biol.* 2015;11(4). doi:10.1371/journal.pcbi.1004190.

13. Sanger JW, Kang S, Siebrands CC, et al. How to build a myofibril. *J Muscle Res Cell Motil.* 2006;26(6-8):343-354. doi:10.1007/s10974-005-9016-7.
14. Ningping Fan, Li CC, Fuchs F. Myofibril image processing for studying sarcomere dynamics. In: *[1988 Proceedings] 9th International Conference on Pattern Recognition.* IEEE Comput. Soc. Press; 1988:468-472. doi:10.1109/ICPR.1988.28269.
15. Arkin EM, Kim J, Mitchell JSB, Sabhnani GR. The Pencil Packing Problem. *Proc Fall Work Comput Geom.* 2009;19.
16. Bellman R. On a routing problem. *Q Appl Math.* 1958;16(1):87-90. doi:10.1090/qam/102435.
17. Polishchuk V, Mitchell JSB. Thick non-crossing paths and minimum-cost flows in polygonal domains. In: *Proceedings of the Twenty-Third Annual Symposium on Computational Geometry - SCG '07.* New York, New York, USA: ACM Press; 2007:56. doi:10.1145/1247069.1247079.
18. Arkin EM, Mitchell JSB, Polishchuk V. Maximum thick paths in static and dynamic environments. *Comput Geom.* 2010;43(3):279-294. doi:10.1016/j.comgeo.2009.02.007.
19. Chartrand G. *Introductory Graph Theory.* Courier Corporation; 1977.
20. Jenson, P, Bard J. Relation of Pure Minimum Cost Flow Model to Linear Programming.
21. Lorensen WE, Cline HE. Marching cubes: A high resolution 3D surface construction algorithm. In: *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques - SIGGRAPH '87.* Vol 91. New York, New York, USA: ACM Press; 1987:163-169. doi:10.1145/37401.37422.
22. Friedman E. Circles in Circles - Erich's Packing Center. <http://www2.stetson.edu/~efriedma/cirincir/>.
23. Blausen.com staff (2014). "Medical gallery of Blausen Medical 2014". *WikiJournal of Medicine* 1 (2). DOI:10.15347/wjm/2014.010. ISSN 2002-4436.
24. David Eppstein - Own work

CHAPTER THREE

FINITE ELEMENT MODELLING TO UNDERSTAND THE ROLE OF SUBCELLULAR STRUCTURES ON WHOLE-CELL MECHANICS

The cellular models and subcellular geometries identified by the method presented in the previous chapter are a useful input for further investigation of the mechanical and electrophysiological behavior of single cells. In this chapter we present a framework for using these (or other cellular geometries) as inputs for a multiphysics finite element model of the mechanical behavior and properties of the cells. This model framework was developed in COMSOL Multiphysics and allows fully or semi-automated generation mechanics simulations.

COMSOL Multiphysics was used as the underlying finite element solver platform upon which the model will be built as it offers several advantages over other platforms that make it suited for this project. First, is a fully documented Java API which makes integration with MATLAB, where the algorithm previously described was developed, simple. Second, COMSOL handles the coupling of the underlying physics to each other, which is much more difficult in other software packages which are not designed specifically for multiphysics problems. Lastly, COMSOL is deployed on the Palmetto Cluster, Clemson University's supercomputing platform. This would allow us to access this computing resource to run complex simulations with only minor modification, if the project grows in scale and complexity in the future.

3.1 Literature Review

3.1.1 Cell Mechanics

The earliest models of cellular mechanics were of “balloons full of molasses” – simple continuum models with no internal structures present.¹ While not representative of the actual physiological environment of the cell, these models persist in the literature as they are easy to understand, provide decent approximations of mechanical properties, and are analogous to the macroscopic mechanics that are introduced in most undergraduate engineering curricula. On the contrary, structural models of cell mechanics take into account these subcellular elements and provide a more realistic model of the mechanical behavior, at the expense of computational cost.

The simplest (yet most persistent in literature²⁻⁵) mechanical model that is applied to cellular mechanics is the Hertz model⁶ of elastic contact between spheres or infinite half-spaces. This model assumes each material is a linearly elastic, homogenous, and isotropic material and contact between them is frictionless, non-plastic, and with infinitesimally small strain⁷. If these assumptions hold, then the contact force can be given by:

$$F = \frac{4}{3} \frac{E}{(1 - \nu^2)} R^{\frac{1}{2}} \delta^{\frac{3}{2}}$$

where E is the elastic modulus of the sample and ν is the Poisson’s ration, R is the radius of the indenting sphere, and δ is the depth of indentation. This approximation is particularly useful for analysis of Atomic Force Microscopy³ (AFM) indentation data – a

technique commonly used to assess whole cell mechanical properties (described in more detail in section 3.1.4).

3.1.2 Viscoelasticity

A notable limitation of the Hertz model when applied to cell mechanics is the assumption that biological materials are elastic in nature. In reality, most biological materials exhibit some degree of viscoelasticity^{8,9} – a time-dependent response when subjected to stress or strain. While an elastic material experiences a linear increase in stress with applied strain (or vice versa) and an equally linearly decrease upon unloading, viscoelastic materials experience hysteresis between loading and unloading as well as the phenomena of creep and stress-relaxation when subjected to constant stress and strain respectively.

Several material models of viscoelastic materials have been developed which describe the various phenomena associated with viscoelasticity. The Maxwell model consists of a perfectly elastic spring in series with a purely viscous damper¹⁰ and the constitutive equation describing its mechanics is given by:

$$\sigma + \frac{\eta}{E} \frac{d\sigma}{dt} = \eta \frac{d\varepsilon}{dt}$$

where σ and ε are the stress and strain of the material, η is the viscosity and E is the elastic modulus. Because of the series relationship of this model, it predicts stress relaxation well. There is an immediate reaction to the application or removal of stress (because of the spring element) and a time dependent relaxation due to the dashpot element. However, the Maxwell material does not accurately predict creep, since the

elastic component ensures a linear increase in strain under constant stress, which is not observed in most viscoelastic materials.

Arranging the components in parallel instead of series yields the Kelvin-Voigt model¹⁰. As the inverse arrangement, this model's constitutive equation is given by:

$$\sigma = E\varepsilon + \eta \frac{d\varepsilon}{dt}$$

The Kelvin-Voigt model predicts the creep behavior well but is less accurate in accounting for relaxation.

The Standard Linear Solid model is the simplest model that accounts for both phenomena¹⁰ and consists of a spring in parallel with a series combination of spring and damper (Maxwell representation) or a spring in series with a parallel combination (Kelvin representation). It is more accurate than either of the simpler models but is still incapable of predicting the response to all loading conditions since it only accounts for a single time constant.

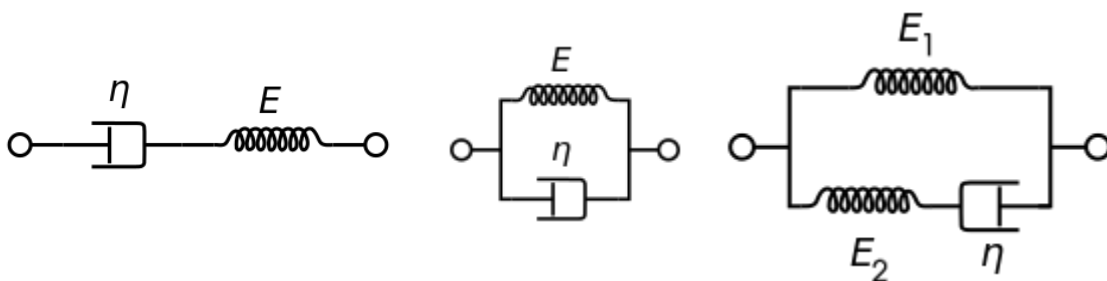


Figure 3.1: Three models of viscoelastic elements. From left to right: the Maxwell model, the Kelvin-Voigt model, and the Standard Linear Solid model (Maxwell representation)³²

This limitation is overcome in the Generalized Maxwell model¹⁰, which extends the Maxwell representation of the Standard Linear Solid Model to include as many parallel viscous branches as are necessary to fully describe the viscoelastic response of the material. While all the viscoelastic models have been applied to cell mechanics^{8,11-15}, the Generalized Maxwell model most accurately predicts cellular behavior which often exhibit multiple time-dependent rates of relaxation.¹⁶

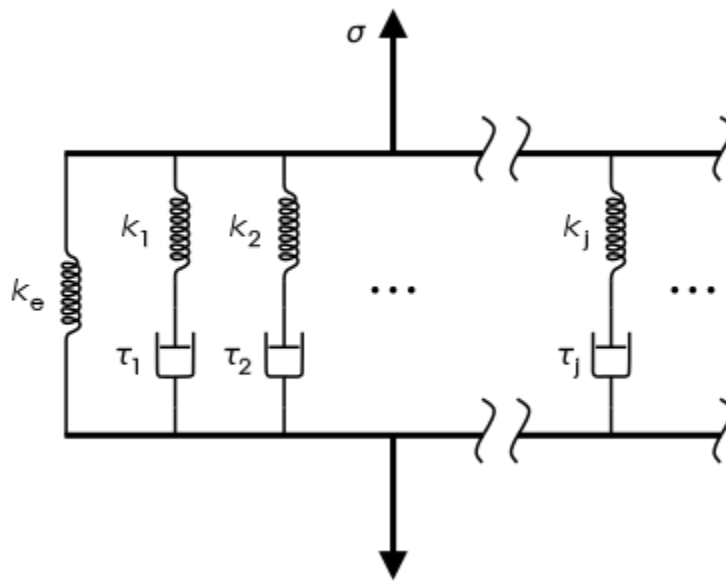


Figure 3.2: The Generalized Maxwell model of viscoelasticity.³²

3.1.3 Structure Based Models

As an alternative to the continuum-based models described in the previous sections, structural models of cellular mechanics have also been explored. The most notable of these model paradigms is based around the concept of tensegrity in which the internal structure of the cell is represented by isolated compressive elements connected to each other through a series of tensioned elements¹⁷⁻¹⁹. The members are either

undergoing pure tension or pure compression, so failure of the whole can only occur if one of the component elements is broken but otherwise the material experiences mechanical stability. This modeling paradigm is particularly interesting for the study of cell mechanics because of the implication that the structure can be rapidly remodeled in response to applied stress to maintain mechanical stability similar to the remodeling that has been observed in cells.¹⁹

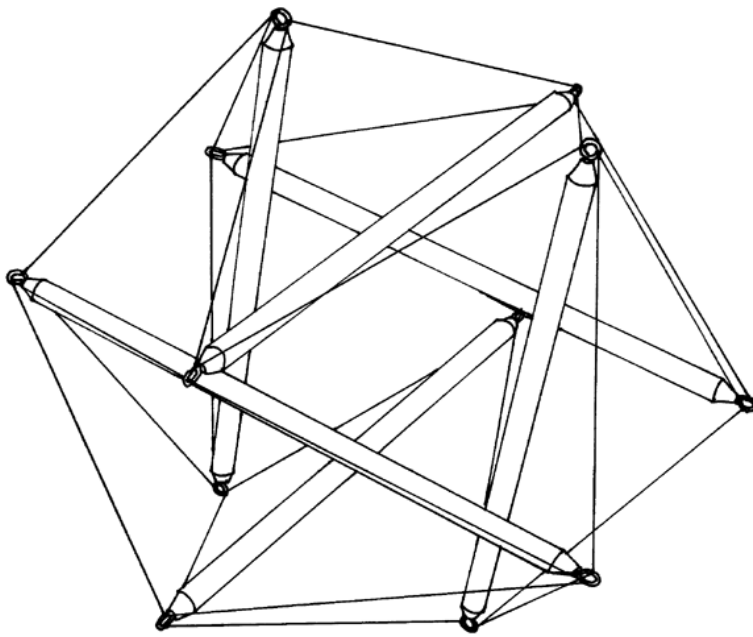


Figure 3.3: A mechanically stable combination of compressed beams and tensioned cables demonstrating the principle of Tensegrity.³³

Another property of biological materials not accounted for by the simpler continuum-based approaches is varying isotropy. While some biological cells and tissues are nearly isotropic, others exhibit anisotropy. Muscle tissue specifically exhibits transverse isotropy. This means that the properties in one direction (such as along the axis of a cylindrical cell) are vastly different than the properties in the transverse directions.

This is largely due to the organization of myofibrils as described in section 2.1.2. Because of this transverse isotropy, we hypothesize that modeling cardiac cells as a composite material of fibers embedded in an elastic continuum could improve the accuracy of the model while retaining the benefits of the continuum approach (namely computation efficiency.)

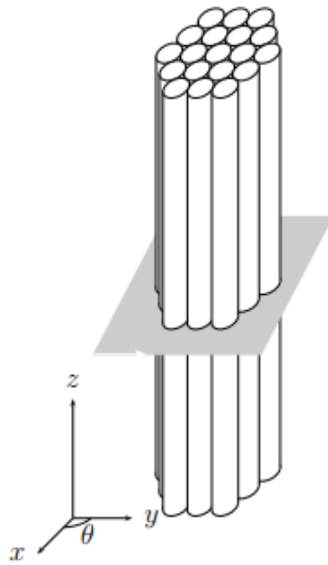


Figure 3.4: A unidirectional fiber composite exhibits transverse isotropy.³⁴

A unidirectional fiber composite is a well understood material composite consisting of uniform fibers which are aligned along a single axis. This provides excellent tensile strength to the material in the fiber direction. The mechanical properties of the material can be approximated in both the axial and transverse directions. For the axial direction, it is assumed that the only two components of the composite are the fibers and matrix. If the fibers are fixed in the matrix and an axial load is applied, it can be assumed

that the strain in the matrix is the same as the strain in the fibers. This iso-strain assumption can be represented as:

$$\varepsilon_c = \varepsilon_m = \varepsilon_f$$

where ε_c , ε_m , and ε_f are the strain in the composite, matrix, and fiber domains respectively. Since the only components are the matrix and fiber domain, all of the applied force is translated to one of the two domains, so the force in the composite can be given by:

$$F_c = F_m + F_f$$

From the definition of stress, we can also represent this as:

$$\sigma_c A_c = \sigma_m A_m + \sigma_f A_f$$

where σ is the stress and A is the cross-sectional area of each domain. If the composite and fibers are assumed to be uniform along their axes, then the cross-sectional areas and volume fractions of the domains are equivalent:

$$\sigma_c = \frac{\sigma_m A_m}{A_c} + \frac{\sigma_f A_f}{A_c} = \sigma_m \frac{V_m}{V_c} + \sigma_f \frac{V_f}{V_c}$$

From the definition of the elastic modulus, E , and the iso-strain assumption this can be rewritten as:

$$E_c \varepsilon_c = E_m \varepsilon_m \frac{V_m}{V_c} + E_f \varepsilon_f \frac{V_f}{V_c} = E_m \varepsilon_c \frac{V_m}{V_c} + E_f \varepsilon_c \frac{V_f}{V_c}$$

Dividing by strain on both sides yields:

$$E_c = E_m \frac{V_m}{V_c} + E_f \frac{V_f}{V_c}$$

And since the volume and matrix volume fractions must add up to 1:

$$E_c = E_m \left(1 - \frac{V_f}{V_c}\right) + E_f \frac{V_f}{V_c}$$

This results in a very good approximation of the axial elastic modulus of the composite dependent only on the moduli of the fibers and matrix and the volume of fibers.

A similar assumption of the properties in the transverse direction can be made if it is assumed that the stress in both domains is the same. The equivalent derivation gives:

$$\frac{1}{E_c} = \frac{1}{E_m} \left(1 - \frac{V_f}{V_c}\right) + \frac{1}{E_f} \frac{V_f}{V_c}$$

This iso-stress assumption is much weaker than the iso-strain assumption, so this is a worse approximation, but is still relatively accurate for low transverse deformations.²⁰

Soft fiber composites have been used to model muscle mechanics at the tissue level and have demonstrated similar properties to muscle.²¹

3.1.4 Experimental Cell Mechanics Techniques

Measurement of mechanical properties of any material, including cells, requires a method of applying either a constant force or displacement and another method of recording the other metric. These manipulations and measurements can be applied to one small part of the cell, or to the cell as a whole. Common techniques used with cells and other biological samples include cantilever techniques like atomic force microscopy^{22,23}, flow techniques, bead probing driven optically or magnetically²⁴, micropipette aspiration, and cell stretching²⁵. These techniques and others are categorized in figure 3.5.

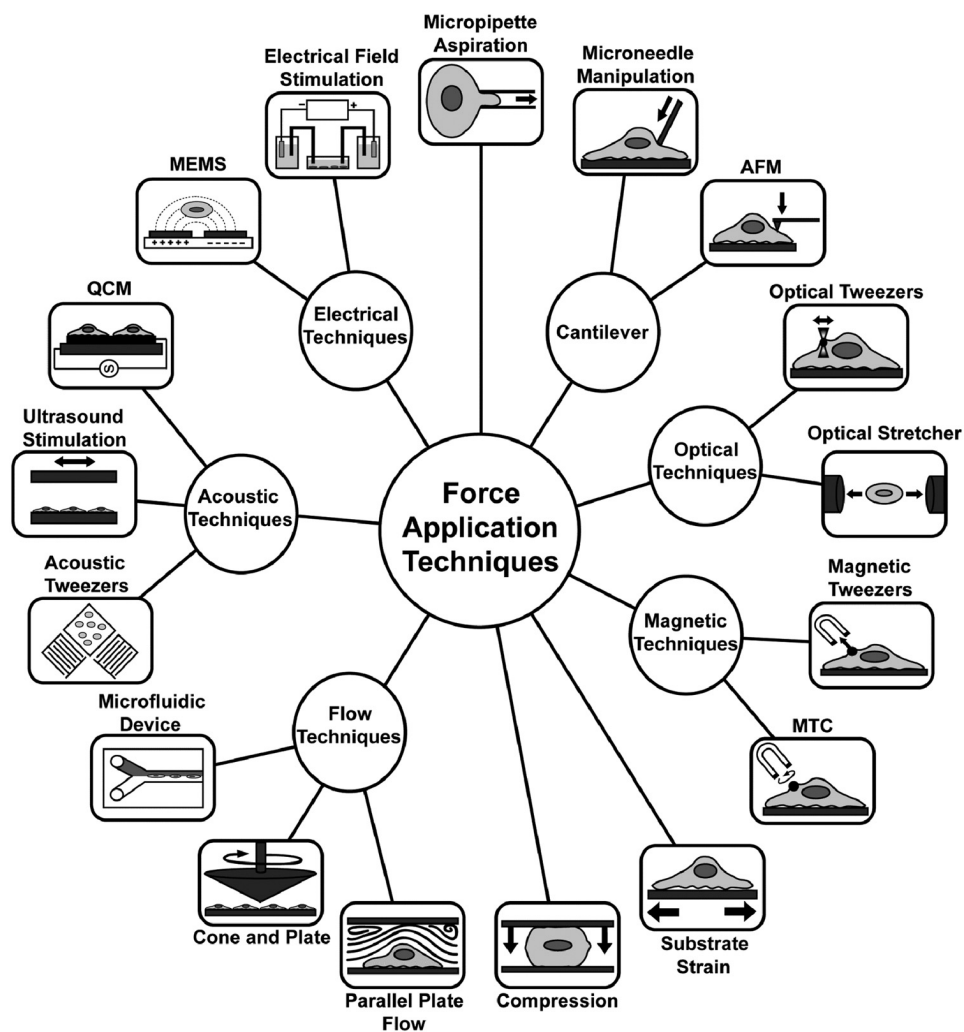


Figure 3.5: Overview of cell mechanics measurement techniques.²⁶

Atomic Force Microscopy (AFM) is a specific type of scanning probe microscopy used to characterize material properties (force mode) or sample topography (scanning mode). During force mode, a tipped cantilever is indented into a sample via a piezoelectric motor. As the tip indents the material, it experiences a resisting force which causes bending of the cantilever to occur. Simultaneously, a laser is aimed at the back of the cantilever, which is usually coated in a reflective material, and reflects into a

photodiode array, which indirectly measures the cantilever's deflection. By calibrating the stiffness of the cantilever and the deflection of the laser, the force of indentation can be measured and used along with the amount of indentation to approximate the mechanical properties of the sample.

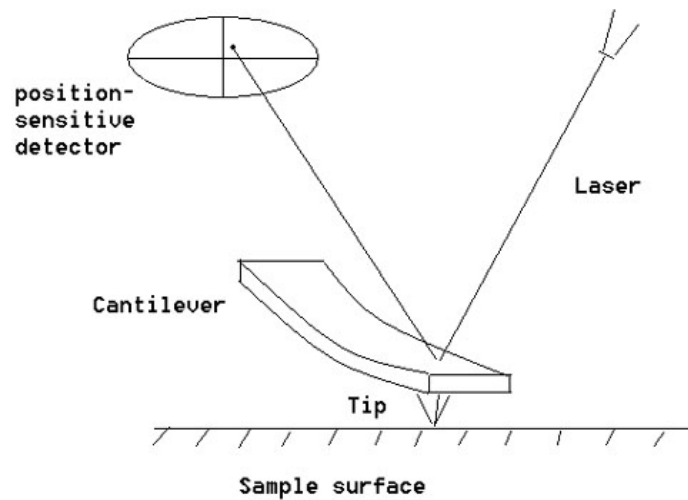


Figure 3.6: Schematic representation of AFM method.³⁵

AFM is widely used in measurements of cellular mechanical properties because of its prevalence in materials science research and the ease of analyzing the data by fitting to the Hertz contact model, as previously discussed. More complex models of contact allow relatively good measurements of cellular properties, but the major limitations of AFM are that it cannot be used to measure whole cell properties and that it can be used to perform measurements normal to or tangent to the cell surface.²⁷ This makes it a great tool for measuring transverse and shear properties, but a bad choice for axial measurements.

Carbon Fiber (CF) manipulation is a relatively new technique in which forces are applied and recorded at opposing ends of a cell through carbon fibers mounted to glass capillaries.²⁵ This allows the measurement of axial properties of cells like myocytes as well as performing contraction studies.^{25,27,28} Measurement of strains in the cells are performed via processing of optical microscopy done simultaneously.^{29,30} This processing allows the sarcomere length of cells to be monitored and the resulting signal can be used as a feedback mechanism to control the piezoelectric elements that apply forces to the carbon fibers. Recently, the combination of the CF technique with other techniques such as AFM has been shown to be successful.²⁵ This combination of AFM and CF allows simultaneous measurements of both axial and transverse properties of the cell and is the experimental setup this chapter will attempt to model.

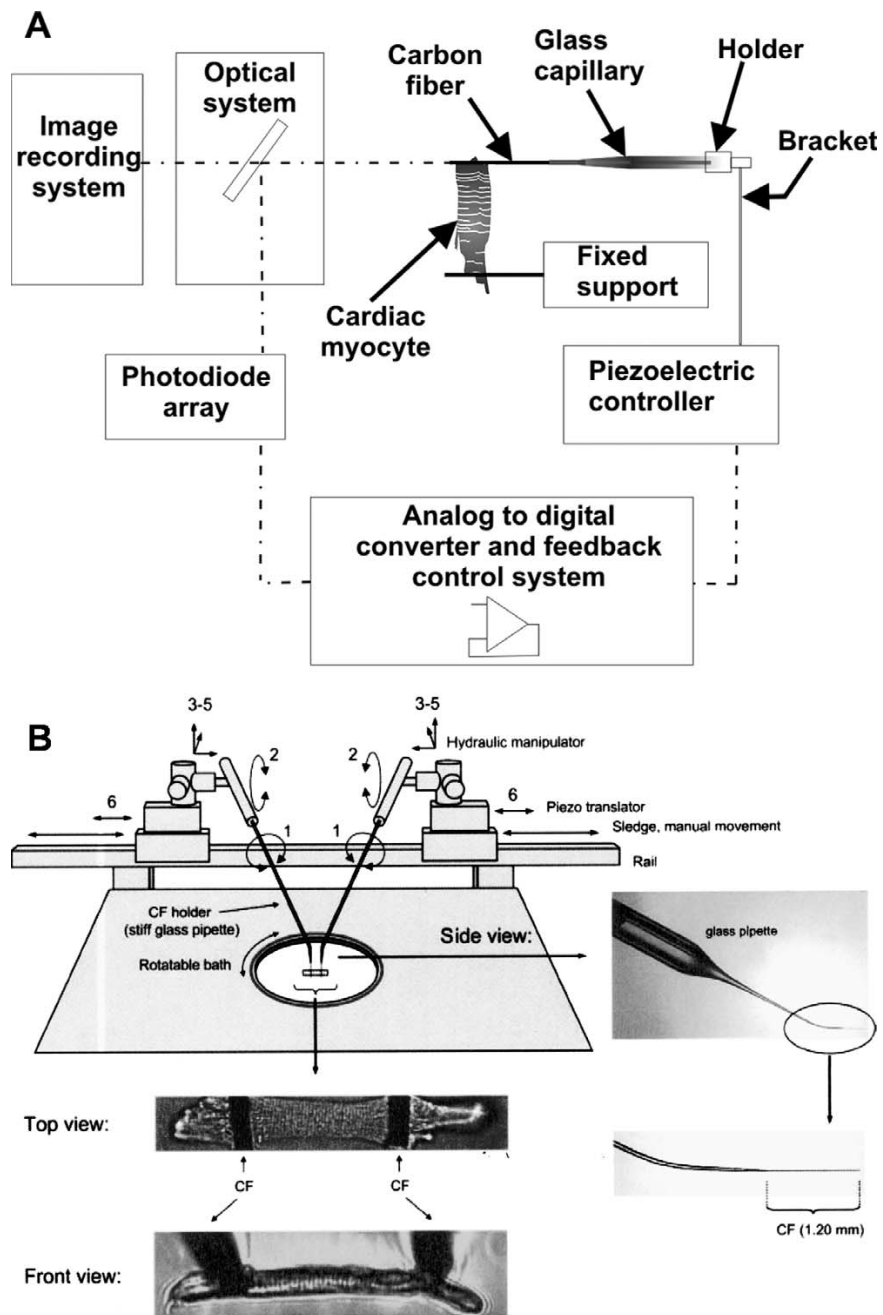


Figure 3.7: A: block diagram of a CF manipulation system²⁹ B: schematic of the experimental setup and images of a myocyte being manipulated³⁰

3.2 Methods

Geometry

The cellular membrane of each cell is represented by an isosurface of triangular faces. In the previous chapter, we described the method to generate this isosurface and the results were saved as Stereolithography (STL) format files. These STL files were imported into COMSOL Multiphysics v5.3 (COMSOL, Inc., Burlington, MA) as meshes into a separate 3-dimensional component with minimal boundary detection. This mesh sequence was then imported into the primary 3-dimensional component as a geometry sequence, utilizing mesh simplification with a relative repair tolerance of 0.01, a defect removal factor of 1, and solids not automatically generated from surface objects. Then, the sequence was converted to a solid to allow physics to be applied. While STL files can be imported as geometry sequences directly, segmenting the process into two distinct steps allows the built-in mesh simplification to be utilized, which makes later meshing of the model domains as free tetrahedral elements significantly simpler.

In the previous chapter, a collection of paths representing myofibrils within the cellular volume was identified and is represented by a list of node-coordinates. In the COMSOL model, each of these paths is given by an open Interpolation Curve, defined by the list of node coordinates. By interpolating the path between points rather than simply importing the paths as piecewise Bezier polygons, sharp angles which make meshing difficult (and which are not physiologically accurate) can be avoided. While these Interpolation Curves can be constructed by manually entering each point or semi-automatically (if the list of points for each curve is stored in a separate text file) the

process can be quite time consuming if a large number of myofibrils are being modelled. In our case, these interpolation curves were generated automatically using the MATLAB Scripting plugin for COMSOL as the paths were generated in the previous chapter.

In order to apply stretch to our model, one end of the cell must be fixed while the other end is subjected to a displacement. Ideally, this displacement will be entirely in one cartesian dimension. In our case, the initial image processing steps rotated and rescaled the images so that the major axis of the cell would be aligned with the y-axis. If this step was not performed in pre-processing, rotation of the entire geometry within COMSOL to achieve this orientation is recommended. To create domains that could be fixed and subjected to displacement, a rectangular block larger than the cell volume in the x and z dimensions and 70-80% of the length of the cell in the y dimension was created roughly in the center of the cell. This block was used as a tool in a Partition Objects sequence to divide the geometry into three distinct domains. The two resulting domains which lie outside the partitioning block are subject to a fixed constraint and a prescribed displacement, respectively, and are therefore not subject to any stress or strain. To simplify the meshing and reduce the computational load of the model, the sections of the interpolation curves which lie in these domains were deleted from the geometry. The geometry was finalized by forming a union of all remaining objects.

Materials

Distinct material properties were defined for the cytoplasmic and myofibril domains. In reality, these particular structures are nonhomogeneous and viscoelastic. Our model will consider them as homogenous (for the sake of computational efficiency) and

purely elastic since the transient responses of the cell to mechanical stress are not of interest to our investigation. The relevant mechanical properties of each material were based on values found in the literature³¹ and are summarized in Table 3.1.

Table 3.1: Material properties for each component of the COMSOL models generated.

Material	Density (kg/m ³)	Young's Modulus (kPa)	Poisson's Ratio
Cytoplasm	1.0	2.2	0.49
Myofibril	12.0	12.0	0.05

Since cytoplasm is primarily composed of water, its density was set to be 1.0 kg/m³. This could also be set slightly higher to account for the presence of other cellular components, but these were ignored in our model. Being primarily water, the cytoplasm is highly viscous, and the elasticity of the cell is primarily due to the cellular membrane. To minimize the computational complexity of the model, the decision was made to combine these into one elastic continuum with an elastic modulus consistent with the literature. If complexity is not a concern, the membrane could instead be modeled as a separate thin-shell or membrane domain with its own mechanical properties.

Physics

Two separate physics modules were applied to the model. On the cytoplasm domains, Solid Mechanics with linear elastic mechanical properties was applied. A fixed constraint was applied to the domain lying in the negative y direction (the major axis of the cell is oriented along y) and a prescribed displacement was applied to the domain

lying closest to the positive y direction. The displacement was restricted to the y direction a parameter was created to vary the amount of stretch applied to the cell.

The Truss physics module was applied to all edges representing fibers (note that the edges created as a result of the domain partitioning were not subject to this physics.) The default straight edge constraint for the trusses was disabled, and the cross section area of the trusses elements was set consistent with the node spacing of the geometry (circular cross sections with radii of either 2 or 4 μm .) To couple the two physics modules to one another, the dependent variable of the truss physics, displacement, was changed to the same displacement field of the solid mechanics domains.

Mesh

The geometry was reduced to a single free tetrahedral mesh with normal size elements using the default meshing algorithms in COMSOL. This resulted in sufficient element density of both the truss elements and solid domains for the current investigation.

Study and Results

A stationary study including geometric nonlinearity (to account for the deforming mesh) was performed using a segregated MUMPS solver with COMSOL's suggested settings. A parametric sweep was included to alter the applied stretch. The exact amounts of stretch differed for each cell, but each was subject to up to approximately 15% strain.

After the study was performed, the results were determined from the solution set with a spatial reference frame. At each value of applied stretch, the axial stress in each truss element was plotted in 3D along with the deformed mesh (represented as a wireframe) and the deformed cellular volume, deformed truss volume, average axial

strain in the truss elements, and total y reaction force at the fixed end of the cell were computed.

Assessment and Validation

Assessment of the model was performed by comparison to experimental data obtained through combined AFM and Carbon Fiber mechanical analysis of isolated mouse adult cardiomyocytes²⁵. The main metrics for comparison are total axial force vs applied stretch and apparent transverse elastic modulus vs applied stretch. In order to compare between different sized cells and account for species differences, the applied stretch was normalized by change in sarcomere length. This was measured optically in the experimental setup and calculated from the average axial strain in each fiber for the model.

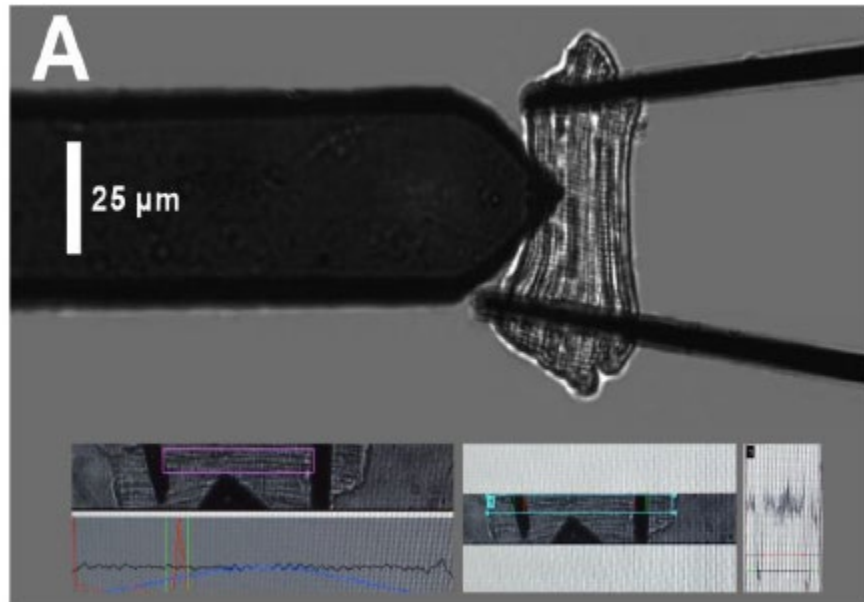


Figure 3.8: AFM/Carbon Fiber experimental setup being used to perform mechanics measurements on an isolated mouse adult cardiomyocyte.²⁵

Apparent elastic modulus was measured experimentally using an AFM cantilever with a 5 μm borosilicate glass bead conjugated. Elastic modulus was then calculated by fitting the data to the Hertz contact model. This technique was not recreated in the model due to the large computational load required to analyze contact mechanics. Additionally, there is no direct way to measure the elastic modulus in either the axial or transverse directions from our model data. Instead, the moduli were estimated by approximating the cell as a unidirectional fiber composite. In this approximation, the transverse elastic modulus can be estimated by:

$$\frac{1}{E_c} = \frac{1}{E_m} \left(1 - \frac{V_f}{V_c}\right) + \frac{1}{E_f} \frac{V_f}{V_c}$$

where E_c , E_m , and E_f are the moduli of the cell (composite), matrix, and fibers respectively and V_f and V_c are the total volumes of the fibers and cell. Similarly, the elastic modulus in the axial direction can be given by:

$$E_c = E_m \left(1 - \frac{V_f}{V_c}\right) + E_f \frac{V_f}{V_c}$$

with the variables representing the same quantities as the transverse direction. It is important to note that the axial direction requires assuming an iso-strain state and the transverse requires the iso-stress assumption. This makes the estimation of the transverse properties slightly less accurate, but still a reasonably good approximation.

3.3 Results

The resulting COMSOL geometries that were generated for the four sample cells used in this study are shown in figure 3.9. The fiber spacing for each cell is 2 μm . Table 3.2 provides a summary of the details of the geometry including cell volume and number of fibers generated for that cell in the algorithm presented in the previous chapter. The model degrees of freedom are also listed. These are dependent not only on the complexity of the geometry, but also the meshing of the geometry and the physics applied. Lastly the computation time needed to perform the stationary study at 7 different values of stretch are included.

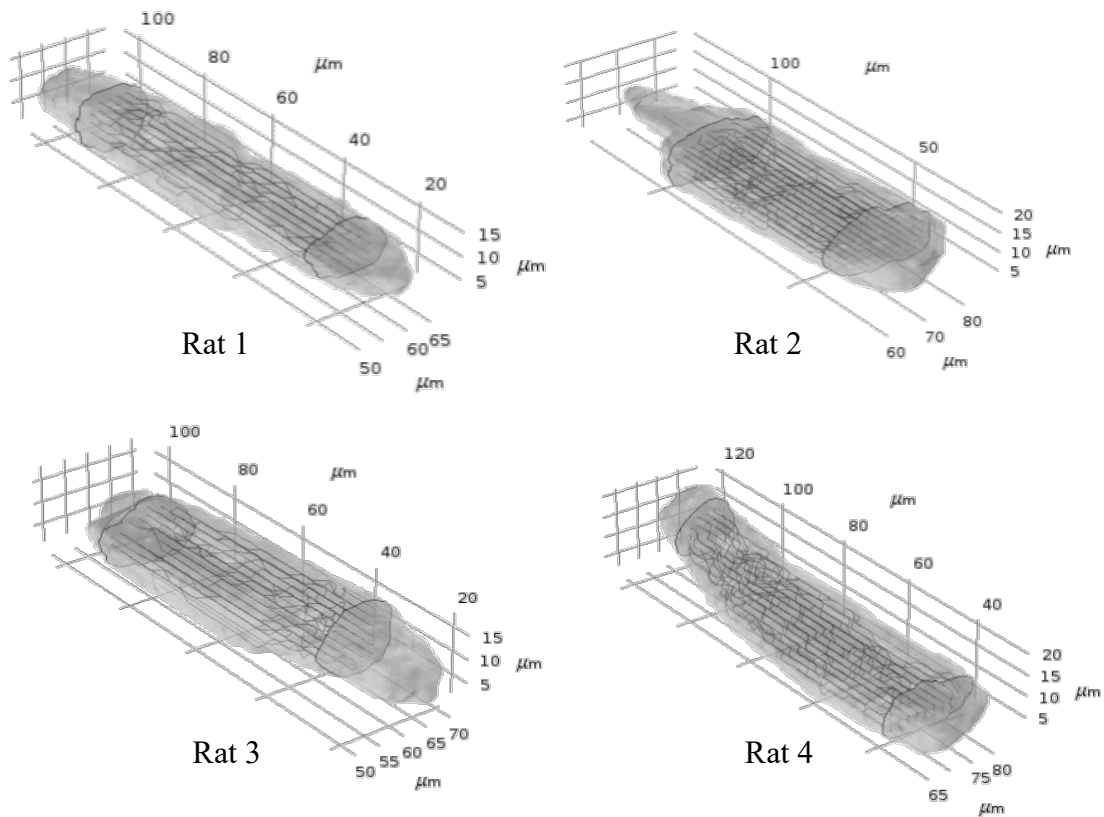


Figure 3.9: Model geometries, generated by the algorithm presented in chapter 2, visualized in COMSOL. Fiber node spacing for all geometries shown is 2 μm .

Table 3.2: Summary of model complexity for the four COMSOL models generated.

Cell	Volume (μm^3)	# of Fibers	Degrees of Freedom	Computation Time*
Rat 1	14,188	24	44,775	9 min 37 sec
Rat 2	27,483	56	92,505	21 min 6 sec
Rat 3	16,057	30	80,157	36 min 52 sec
Rat 4	16,662	37	87,696	22 min 26 sec

*All computations were performed with COMSOL v5.3 on a 64-bit Windows 10 machine with an AMD A8-7600 Radeon R7, 10 Compute Cores (4C+6G) @ 3.10 GHz and 24.0 GB of DDR3 RAM.

Sensitivity of the model to fiber spacing was assessed by performing the study on a sample cell (Rat 2) with fiber spacings of both 2 μm and 4 μm . The resulting visualization of axial stress in each fiber overlaid on deformation of the solid domains at maximum stretch is shown in figure 3.10. Fiber spacing of 4 μm is shown on the left and 2 μm on the right. As expected, the larger spacing resulted in higher maximum stresses in individual fibers. A larger decrease in the cross-sectional area of the cell can also be observed in the larger spacing case.

Figure 3.11 shows the summary graphs of the axial force vs applied stretch, transverse elastic modulus vs applied stretch, and axial elastic modulus vs applied stretch for both models. All three metrics were initially higher (at zero stretch) for the model with larger fiber spacing and also exhibited a larger increase with increased stretch in this model.

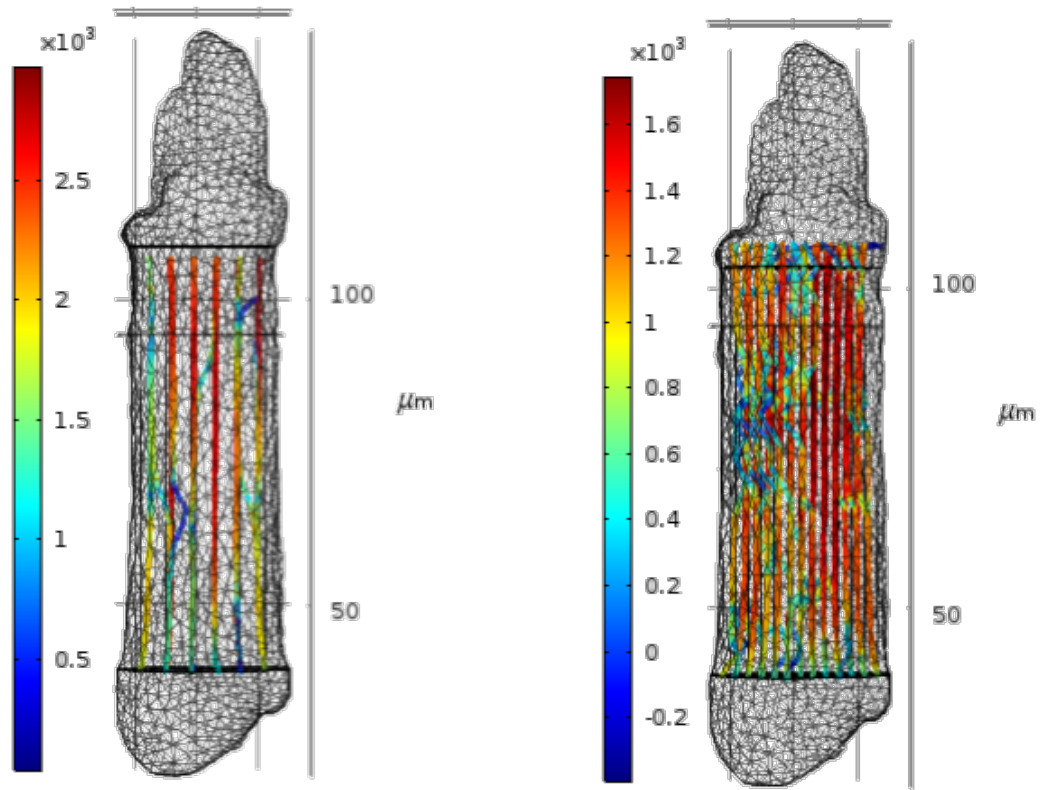


Figure 3.10: Comparison of fiber thickness on simulation results. Deformed meshes and axial stress in the truss elements are shown for a sample cell (Rat 2) with node spacing of $4 \mu\text{m}$ (left) and $2 \mu\text{m}$ (right). Units for both color legends are N/m^2 .

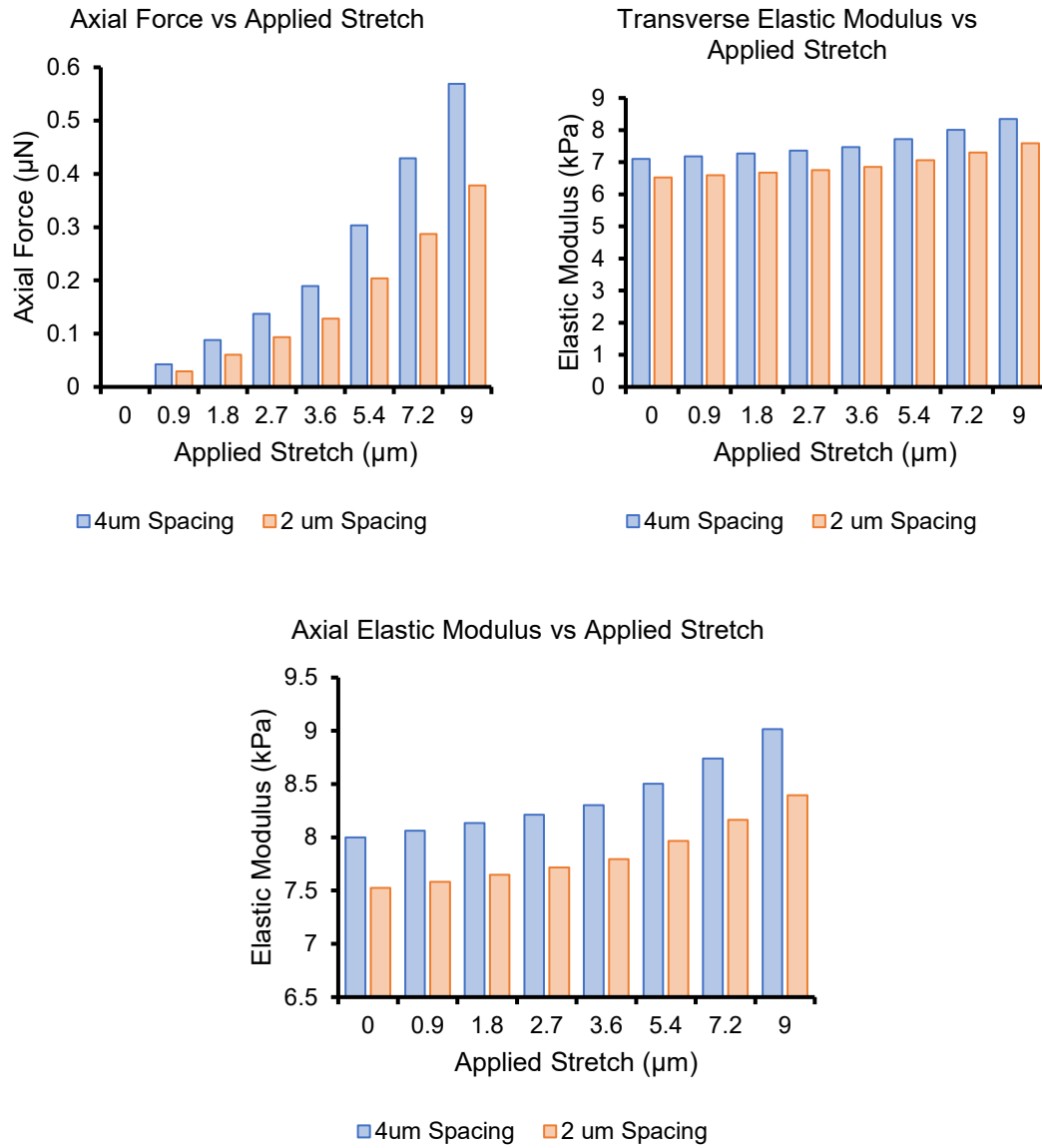


Figure 3.11: Comparison of simulation results between models with fiber spacing of 2 and 4 μm for a sample cell (Rat 2). Total axial force, transverse elastic modulus, and axial elastic modulus are plotted vs applied stretch.

The remaining cells were only modeled with a fiber spacing of 2 μm . The amount of applied stretch varied for each cell (based on its initial length) in order to achieve a similar degree of strain in each cell. Figure 3.12 shows a representative visualization of axial stress in each fiber overlaid on solid deformation for a single cell (Rat 3) at each value of stretch applied. The colored legend for each is normalized to the highest stretch case. It can be observed that axial strain remains relatively low initially as the fibers straighten out, with stress increasing rapidly after the fibers are all straight. Some degree of necking can also be observed at the higher cases of stretch.

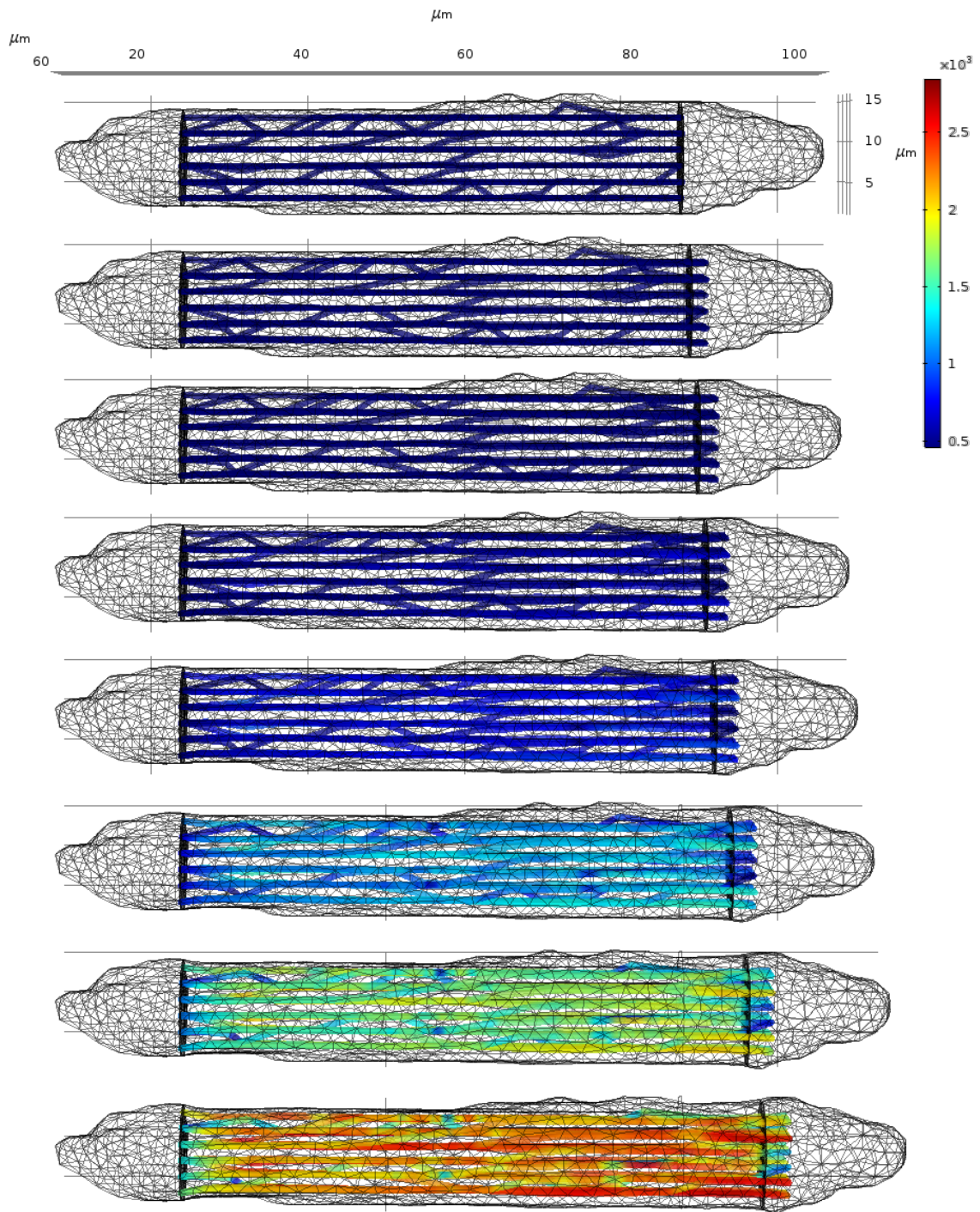


Figure 3.12: Deformed mesh and axial stress in the truss elements for a sample cell (Rat 3) for each applied stretch (0, 1.2, 2.4, 3.6, 4.8, 7.2, 9.6, 12 μm). Units of the color legend are N/m^2 .

The model data was compared to experimental data measured on a combined Carbon Fiber/AFM setup on isolated mouse adult cardiomyocytes. To account for size differences across species, the stretch data was normalized to % change in sarcomere length (experimental) or % change in axial strain (model). Figure 3.13 shows each value of axial force plotted vs change in sarcomere length for the experimental data. A total of 19 cells from 4 mice were included. Figure 3.14 shows the same metrics plotted for the model data from all 4 rat cardiomyocyte models.

Figures 3.15 and 3.16 presents this same data for the experimental setup and model, respectively. In these figures, the x-axis data is binned into groups of 1% change in sarcomere length/axial strain to show the range of axial forces experienced at each percent change in elongation. Because of the small sample size of the model, some bins only resulted in a single data point.

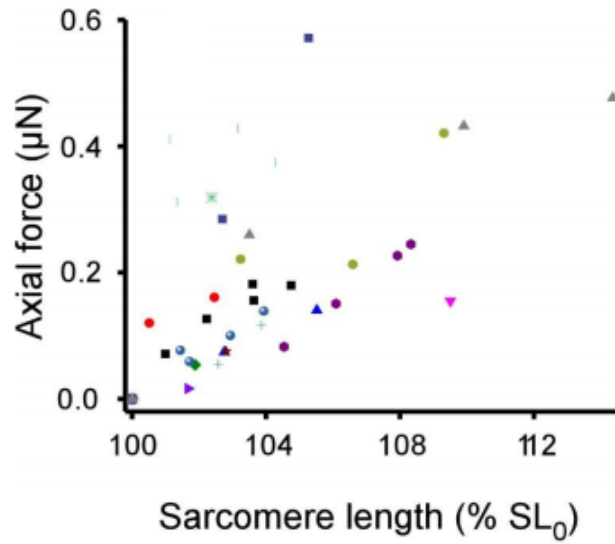


Figure 3.13: Experimental data of 19 cardiomyocytes (4 mice) stretched on a combination carbon fiber/AFM apparatus. Total axial force in the cell is plotted vs sarcomere length (as a percentage of resting length).

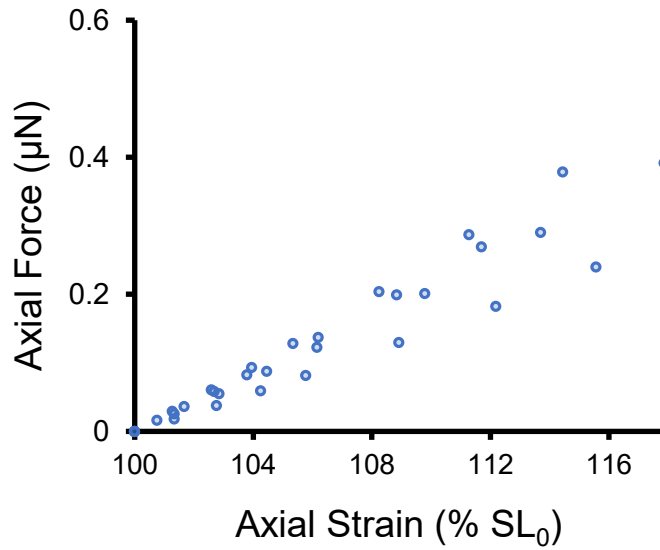


Figure 3.14: Simulated data of 4 cardiomyocytes (Rat 1-4). Total axial force in the cell is plotted vs axial strain (as a percentage of resting length). This axial strain is analogous to a change in sarcomere length.

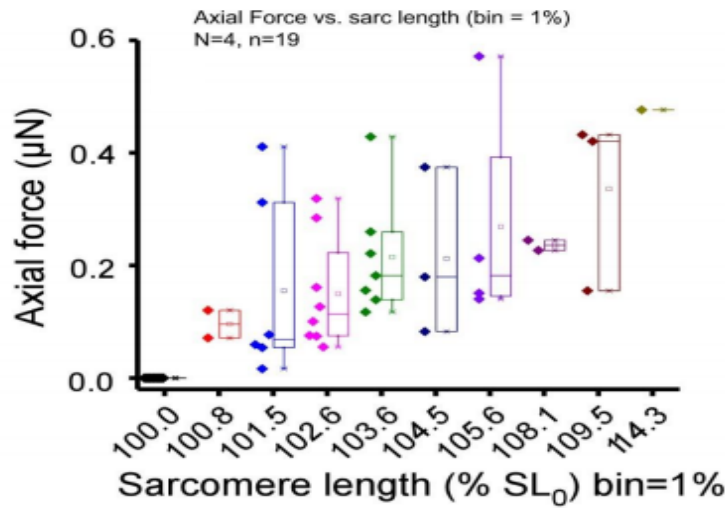


Figure 3.15: Experimental data of 19 cardiomyocytes (4 mice) stretched on a combination carbon fiber/AFM apparatus. Total axial force in the cell is plotted vs sarcomere length (as a percentage of resting length). The x-axis was grouped into bins of 1% change in sarcomere length.

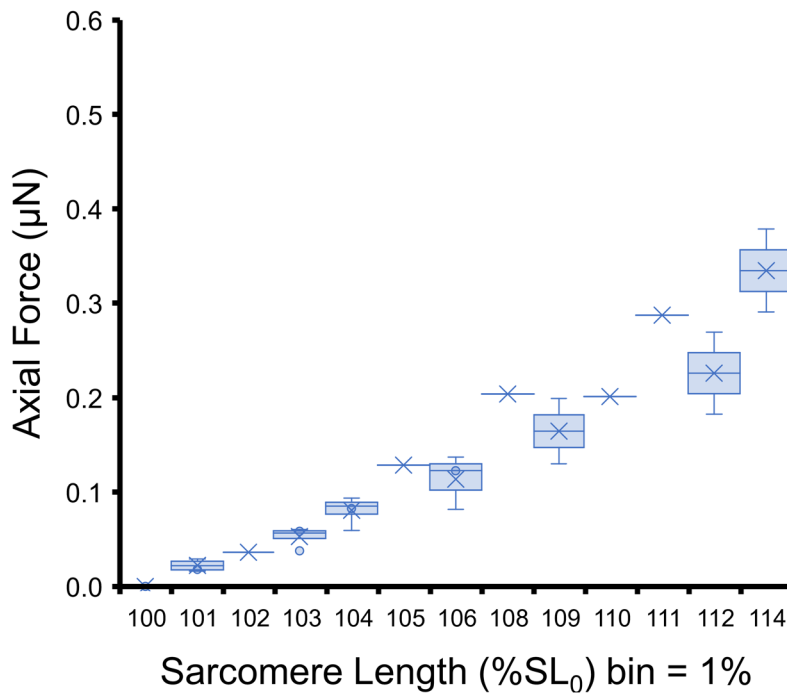


Figure 3.16: Simulated data of 4 cardiomyocytes (Rat 1-4). Total axial force in the cell is plotted vs axial strain (as a percentage of resting length). The x-axis was grouped into bins of 1% change in sarcomere length.

The second metric assessed through comparison with the experimental data is the transverse elastic modulus of the cell. This data is plotted in Figure 3.17, with similar 1% binning of the elongation as in the previous analysis. For the experimental setup, this was assessed using the AFM with a 5 μm borosilicate glass bead attached to the end of the AFM cantilever. Resulting indentation data was fit to the Hertz elastic contact model, so the data plotted is the apparent elastic modulus of the cell.

For the model, transverse elastic modulus was assessed by approximating the cell as a unidirectional composite material, so the data plotted in Figure 3.18 is the estimated elastic modulus vs elongation, with the same 1% binning. Again, due to the small sample size of the model, some bins only contained one data point.

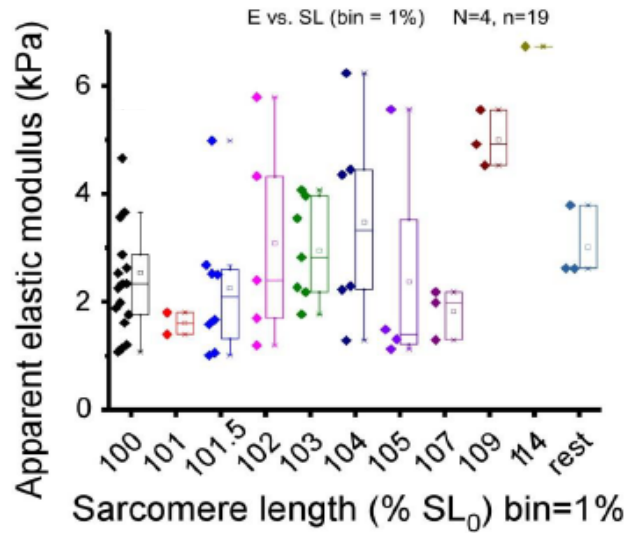


Figure 3.17: Experimental data of 19 cardiomyocytes (4 mice) stretched on a combination carbon fiber/AFM apparatus. Apparent elastic modulus of the cell is plotted vs sarcomere length (as a percentage of resting length). The x-axis was grouped into bins of 1% change in sarcomere length

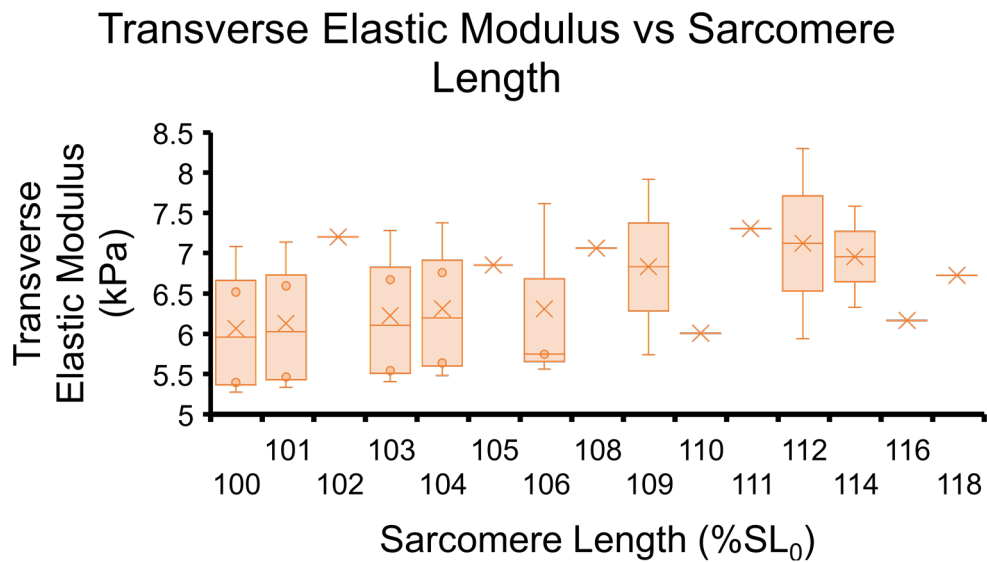


Figure 3.18: Simulated data of 4 cardiomyocytes (Rat 1-4). Estimated transverse elastic modulus on the cell is plotted vs axial strain (as a percentage of resting length). The x-axis was grouped into bins of 1% change in sarcomere length.

3.4 Discussion

As expected, larger diameter fibers result in an increase in the axial force and elastic moduli of the cell. Consistent with the findings in the previous chapter, larger fiber spacing results in an increase in the packing density of the fibers. The elastic moduli estimations are especially sensitive to the volume fraction of fibers. Both fiber spacings resulted in cell properties of similar magnitude to the experimental data, at least in the single cell model where both were analyzed. It can also be observed in figure 3.10 that the smaller fiber spacing resulted in noticeably more narrowing of the cell as it was stretched, despite an increase in the transverse elastic modulus. It is important to note that this cell had a much larger volume and lower aspect ratio than the other cells, which resulted in efficient packing despite fiber spacing. As shown in the previous chapter, high aspect ratio cells can be packed more efficiently by smaller fibers. Because of this, the remainder of the models were only generated with the smaller spacings.

Comparison of the model performance to the experimental data shows that most measured properties were consistent. In general, the model exhibited lower total axial force in the cell and lower mechanical properties. The material properties chosen for the model were based on values reported in literature. These properties could be modified so that the performance more closely matches the experimental data. It is also worthwhile to note that there was much less variability in the model data (as can be expected with a model) but the sample size for the model was also much lower than the experimental data ($n=4$ vs $n=19$). Perhaps the biggest difference between the experimental and model data is the dependence of transverse elastic modulus on applied stretch. A similar increase in

modulus was observed as the cells experienced stretch, but the magnitude of the increase was much lower in the model. The exact reasons for this are unclear but it is likely due to some other physiological mechanisms that are taking place in the experimental data that are not captured by the model, such as osmosis. It could also be due to other cytoskeletal elements of the cell which are not captured by the model.

This limited information about what is occurring in the transverse directions is one of the major limitations of the model. Trusses, by design, only experience stresses in the axial direction, so they don't contribute much to the transverse mechanics of the cell. In future studies, this limitation might be diminished by also including other cytoskeletal elements, such as actin and microtubules which are more directly involved with transverse mechanics since they are not isotropic in the cell like myofibrils. Despite the limitation of trusses, they also exhibit several advantages. Even though our paths travel through all three dimensions, once meshed each individual truss is modeled as a 1D element. This substantially reduces the computational load compared to modelling the fibers as a 3D solid. Additionally, because they are truss elements, the axial properties are easier to calculate than they would be for a different geometry type since the physics are already calculated based on their axial direction. If instead they were modelled as 3D elements, the physics would be calculated with respect to the x, y, and z directions so an extra step of computing the rotated axis that align with the axis of the fiber would be necessary to compute the axial properties. Though not performed in this study, applying stresses to each individual fiber in the axial direction is also possible using our framework. This allows further modelling of cellular contraction to be performed.

3.5 Conclusions

We have created a semi-automated framework to model the mechanics of single cardiomyocytes using COMSOL Multiphysics. It has been demonstrated that modelling these cells as composites of a linear elastic solid embedded with fiber trusses provides a reasonable approximation of cellular mechanical properties in the transverse and axial directions. This basic framework allows for easy extension to account for more cytoskeletal elements, contraction and contact mechanics studies, or coupling of additional physics such as cellular electrophysiology. By modifying the input geometries, such as by changing the optimization parameters of the algorithm presented in the previous chapter, rapid analysis of how differing cytoskeletal organization affects the mechanical properties of the cell can be performed. This method could also be readily adapted to be used on tissue-level geometries as well. We have demonstrated that differing fiber spacings results in similar mechanical behavior, so with tuning of the model's material properties computational cost can be saved if necessary when scaling up to multicellular geometries without only minimal reductions in performance.

3.6 References

1. Brookes M. Hard cell, soft cell. *New Sci.* 1999;164(2206):42-46.
2. Thomas G, Burnham NA, Camesano TA, Wen Q. Measuring the Mechanical Properties of Living Cells Using Atomic Force Microscopy. *J Vis Exp.* 2013;(76):1-7. doi:10.3791/50497.
3. Zhu X. Tutorial on Hertz Contact Stress. 2012:1-8.
4. Dokukin ME, Guz N V., Sokolov I. Quantitative study of the elastic modulus of loosely attached cells in AFM indentation experiments. *Biophys J.* 2013;104(10):2123-2131. doi:10.1016/j.bpj.2013.04.019.

5. Hayashi K, Iwata M. Stiffness of cancer cells measured with an AFM indentation method. *J Mech Behav Biomed Mater.* 2015;49:105-111. doi:10.1016/j.jmbbm.2015.04.030.
6. Hertz H. On the contact of rigid elastic solids. *J Reine und Angewandte Math.* 1896;92:156.
7. Wood ST. Computational approaches to understanding phenotypic structure and constitutive mechanics relationships of single cells. 2011.
8. Radmacher M, Fritz M, Kacher CM, Cleveland JP, Hansma PK. Measuring the viscoelastic properties of human platelets with the atomic force microscope. *Biophys J.* 1996;70(1):556-567. doi:10.1016/S0006-3495(96)79602-9.
9. Ragazzon MRP, Gravdahl JT, Vagia M. Viscoelastic properties of cells: Modeling and identification by atomic force microscopy. *Mechatronics.* 2018;50:271-281. doi:10.1016/j.mechatronics.2017.09.011.
10. Roylance D. Engineering Viscoelasticity. 2001:8-11.
11. Yu H, Li Z, Jane Wang Q. Viscoelastic-adhesive contact modeling: Application to the characterization of the viscoelastic behavior of materials. *Mech Mater.* 2013;60:55-65. doi:10.1016/j.mechmat.2013.01.004.
12. Mahaffy R, Park S, Gerde E, Kas J, Shih C. Quantitative Analysis of the Viscoelastic Properties of Thin Regions of Fibroblasts Using Atomic Force Microscopy. *Biophys J.* 2004;86(March):1777-1793. doi:10.1016/S0006-3495(04)74245-9.
13. Smith BA, Tolloczko B, Martin JG, Grütter P. Probing the Viscoelastic Behavior of Cultured Airway Smooth Muscle Cells with Atomic Force Microscopy: Stiffening Induced by Contractile Agonist. *Biophys J.* 2005;88(4):2994-3007. doi:10.1529/biophysj.104.046649.
14. Li M, Liu L, Xi N, Wang Y. Atomic force microscopy studies on cellular elastic and viscoelastic properties. *Sci China Life Sci.* 2018;61(1):57-67. doi:10.1007/s11427-016-9041-9.
15. Corbin EA, Adeniba OO, Ewoldt RH, Bashir R. Dynamic mechanical measurement of the viscoelasticity of single adherent cells. *Appl Phys Lett.* 2016;108(9):093701. doi:10.1063/1.4942364.

16. Hemmer JD, Nagatomi J, Wood ST, Vertegel AA, Dean D, LaBerge M. Role of Cytoskeletal Components in Stress-Relaxation Behavior of Adherent Vascular Smooth Muscle Cells. *J Biomech Eng.* 2009;131(4):041001. doi:10.1115/1.3049860.
17. Ingber DE. Tensegrity I. Cell structure and hierarchical systems biology. *J Cell Sci.* 2003;116(7):1157-1173.
18. Wang N, Naruse K, Stamenovic D, et al. Mechanical behavior in living cells consistent with the tensegrity model. *Proc Natl Acad Sci.* 2001;98(14):7765-7770. doi:10.1073/pnas.141199598.
19. Hoffman BD, Crocker JC. Cell Mechanics: Dissecting the Physical Responses of Cells to Force. *Annu Rev Biomed Eng.* 2009;11(1):259-288. doi:10.1146/annurev.bioeng.10.061807.160511.
20. Zinoviev PA, Smerdov AA. Ultimate properties of unidirectional fiber composites. *Compos Sci Technol.* 1999;59(5):625-634. doi:10.1016/S0266-3538(98)00108-0.
21. Chanda A, Callaway C. Tissue Anisotropy Modeling Using Soft Composite Materials. *Appl Bionics Biomech.* 2018;2018:1-9. doi:10.1155/2018/4838157.
22. Ding Y, Xu GK, Wang GF. On the determination of elastic moduli of cells by AFM based indentation. *Sci Rep.* 2017;7:1-8. doi:10.1038/srep45575.
23. Radmacher M. Measuring the elastic properties of biological samples with the AFM. *IEEE Eng Med Biol Mag.* 1997;16(2):47-57. doi:10.1109/51.582176.
24. Trepas X, Grabulosa M, Buscemi L, et al. Oscillatory magnetic tweezers based on ferromagnetic beads and simple coaxial coils. *Rev Sci Instrum.* 2003;74(9):4012-4020. doi:10.1063/1.1599062.
25. Desai AY. Manipulating Cardiovascular Cellular Interactions and Mechanics: A Multidimensional and Multimodal Approach. 2016.
26. Rodriguez ML, McGarry PJ, Sniadecki NJ. Review on Cell Mechanics: Experimental and Modeling Approaches. *Appl Mech Rev.* 2013;65(6):060801. doi:10.1115/1.4025355.
27. Addae-Mensah KA, Wikswo JP. Measurement Techniques for Cellular Biomechanics In Vitro. *Exp Biol Med.* 2008;233(7):792-809. doi:10.3181/0710-MR-278.

28. Radmacher M, Fritz M, Hansma PK. Imaging soft samples with the atomic force microscope: gelatin in water and propanol. *Biophys J*. 1995;69(1):264-270. doi:10.1016/S0006-3495(95)79897-6.
29. Yasuda S-I, Sugiura S, Kobayakawa N, et al. A novel method to study contraction characteristics of a single cardiac myocyte using carbon fibers. *Am J Physiol Circ Physiol*. 2001;281(3):H1442-H1446. doi:10.1152/ajpheart.2001.281.3.H1442.
30. Iribe G, Helmes M, Kohl P. Force-length relations in isolated intact cardiomyocytes subjected to dynamic changes in mechanical load. *Am J Physiol Circ Physiol*. 2007;292(3):H1487-H1497. doi:10.1152/ajpheart.00909.2006.
31. Ogneva I V., Lebedev D V., Shenkman BS. Transversal stiffness and young's modulus of single fibers from rat soleus muscle probed by atomic force microscopy. *Biophys J*. 2010;98(3):418-424. doi:10.1016/j.bpj.2009.10.028.
32. Wikimedia user: Pekaje – Own work.
33. Bob Burkhardt – Own work.
34. Wikimedia user: Tomeasy – Own work.
35. Mai, W. Fundamental Theory of Atomic Force Microscopy. Professor Zhong L. Wang's Nano Research Group. Georgia Institute of Technology. <http://www.nanoscience.gatech.edu/zlwang/research/afm.html>

CHAPTER FOUR

SINGLE-CELL ELECTROPHYSIOLOGICAL MODELS AS TOOLS IN ENGINEERING EDUCATION

The goal of this aim is to develop, launch, and evaluate a new software, CellSpark, which simulates experiments in electrophysiology for use in the course BIOE 3700 – Bioinstrumentation and Bioimaging. This software is used by the undergraduate students enrolled in the course to learn about models of electrophysiology and to develop and perform a short electrophysiology experiment. The experiment becomes the basis of their midterm lab report, which is a journal style article presenting and discussing the findings of their experiment.

4.1 Introduction and Background

4.1.1 Models of Electrophysiology

The earliest successful attempt at any single-cell electrophysiological modelling occurred in 1952, with the publishing of the now famous work by Alan Lloyd Hodgkin and Andrew Huxley.¹ The Hodgkin-Huxley model, also referred to as the conductance-based model, was developed primarily by studying the squid giant axon through use of the patch clamp technique. While originally developed to describe only the behavior of neurons, the model can be generalized to describe any electrically excitable cells including cardiomyocytes.

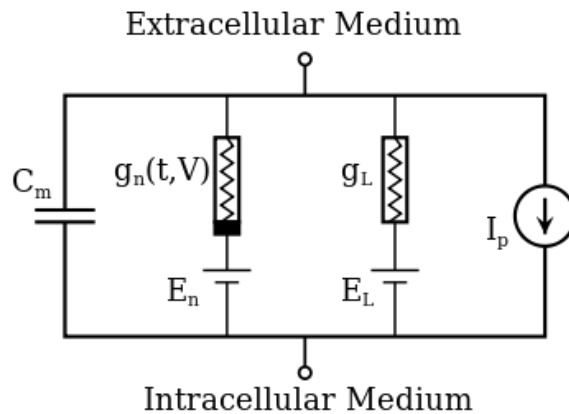


Figure 4.1: Circuit diagram of the Hodgkin-Huxley conductance-based model.²³

Fundamentally, the conductance-based model represents the cell membrane of an electrically excitable cell as the circuit shown in Figure 5.1, with the following elements in parallel:

- A capacitor, representing the membrane capacitance that arises due to the lipid bilayer.
- Nonlinear conductances in series with voltage sources, representing voltage gate ion channels and electrochemical gradients which drive ion diffusion (Nernst potentials), respectively.
- Linear conductances in series with voltage sources, representing leakage ion channels and their Nernst potentials, respectively.
- Current sources, representing the ion pumps which facilitate active transport against electrochemical gradients.

Modelling the membrane in this way results in the excitability of any cell being able to be fully described by the following four ordinary differential equations:

$$\begin{aligned}
I &= C_m \frac{dV_m}{dt} + \bar{g}_K n^4 (V_m - V_K) + \bar{g}_{Na} m^3 h (V_m - V_{Na}) + \bar{g}_l (V_m - V_l), \\
\frac{dn}{dt} &= \alpha_n(V_m)(1 - n) - \beta_n(V_m)n \\
\frac{dm}{dt} &= \alpha_m(V_m)(1 - m) - \beta_m(V_m)m \\
\frac{dh}{dt} &= \alpha_h(V_m)(1 - h) - \beta_h(V_m)h
\end{aligned}$$

where I represents the total membrane current, α and β represent species specific rate constants, and \bar{g} is the maximum value of the conductance. n , m , and h are positive constants less than 1 associated with sodium and potassium channel activation and inactivation. The rate constants, which were experimentally determined by Hodgkin and Huxley are given by:

$$\begin{aligned}
\alpha_n(V_m) &= \frac{0.01(10 - V_m)}{\exp\left(\frac{10 - V_m}{10}\right) - 1} & \alpha_m(V_m) &= \frac{0.1(25 - V_m)}{\exp\left(\frac{25 - V_m}{10}\right) - 1} & \alpha_h(V_m) &= 0.07 \exp\left(\frac{-V_m}{20}\right) \\
\beta_n(V_m) &= 0.125 \exp\left(\frac{-V_m}{80}\right) & \beta_m(V_m) &= 4 \exp\left(\frac{-V_m}{18}\right) & \beta_h(V_m) &= \frac{1}{\exp\left(\frac{30 - V_m}{10}\right) + 1}
\end{aligned}$$

Even more than 60 years later, the Hodgkin-Huxley model is still regarded as one of the most complete models of cell excitability. However, due to its nonlinearity, it is difficult to study analytically and is computationally costly. For this reason, many have sought to build on and simplify the conductance-based model.

The most famous simplification of the Hodgkin-Huxley model is the relaxation oscillator model of nerve conduction first suggested by Richard FitzHugh in 1961² and independently created by Jin-ichi Nagumo et al in 1962.³ Unlike the Hodgkin-Huxley model, which describes ion channel dynamics in great detail, the FitzHugh-Nagumo (FHN) model is described by only two variables: v , the nonlinear membrane voltage and w , a linear recovery variable. The equations of the system are:

$$\dot{v} = v - \frac{v^3}{3} - w + I_{\text{ext}}$$

$$\tau \dot{w} = v + a - bw.$$

This results in a good approximation of the Hodgkin-Huxley model while ignoring the individual ionic dynamics. As such, the comparative simplicity of the FitzHugh-Nagumo model (and derivations of it) makes an ideal choice for many computationally efficient modelling applications.

Another two-variable model, this one focused directly on the heart, was developed with the simplicity of the FitzHugh-Nagumo in mind. In 1996, Rubin Aliev and Alexander Paniflov published “A Simple Two-variable model of cardiac excitation.”⁴ This modification of the FHN differs from the original in two important aspects: the pulse shape and the restitution property of myocardium. Restitution refers to the relationship between action potential duration and length of the cardiac cycle. As can be seen in Figure 5.2, which is generated by the model, the duration of duration of the action potential is substantially shortened as cycle length decreases.

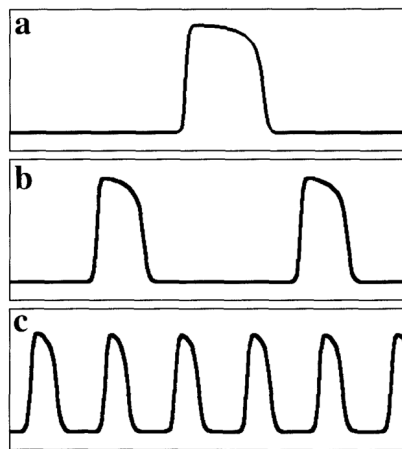


Figure 4.2: Action potential duration at various cardiac cycle frequencies⁴

More advanced models of cardiac electrophysiology have been published for various species including rabbits^{5,6}, guinea pigs, dogs, and humans. One of the most comprehensive of these models is the ten Tusscher Noble Noble Paniflow⁷ (TNNP) model published in 2004. This model of human epicardial, endocardial, and cardiac M cells is an extension of the Hodgkin-Huxley model in that it recreates ionic currents present across the membrane rather than simplifying to an over dynamics problem like the FHN model.

4.1.2 Experiential Education Tools

Over the past 15 years, educators and industry partners alike have repeatedly warned of a growing gap in US competitiveness⁸, a declining interest in STEM, and lagging innovation⁹. In order to address this, a considerable effort has been made to incorporate more hands-on engineering practice into engineering curricula, with much of the push being at the request of industry partners, who warn of graduates unprepared for immediate success in the “real world”¹⁰. One specific method used to this end is Design Based Learning.

Design Based Learning (DBL) is a specific type of Problem Based Learning that “involves students engaged in the process of developing, building, and evaluating a product they have designed.”¹¹ The general process of DBL is that instructor will generally pose an open-ended, loosely defined problem to students. Students then work, typically in small groups, to design, build, and test solutions to the problem¹¹. This method allows students to have a hands-on application of what they learn and because students work in groups to design unique solutions, DBL encourages teamwork and

interpersonal skills, fosters independence from the instructor and allows students to reinforce the curricula and develop problem solving and critical thinking skills¹²

While shown to be successful, DBL requires access to time, space, and physical resources that make it infeasible in some settings. An alternative approach to traditional DBL is the use of simulations as tools for engineering education. Sophisticated computer simulations can allow undergraduate engineering students exposure to “real world” engineering activities in which they would otherwise not be able to participate¹³ or simulated activities that would be time or resource intensive to do physically¹⁴ This latter example is specifically the situation for electrophysiology – the content area of interest for this study. The “gold standard” technique in electrophysiology, the patch clamp, is expensive and time intensive to perform, on top of the hours of practice needed to develop competency, making it a perfect candidate for replacement with a simulation based learning tool.

4.1.3 Instructor Motivation

While many faculty are hesitant to change curricula in order to accommodate simulation tools (especially those designed for research or industry use) into their course content, it has been demonstrated that these tools can be successfully incorporated into courses, showing that students improve their use of these tools and their mastery of course content, without requiring a large amount of lecture time.¹⁵

In a 2012 study, Magana et al¹⁶ explored the various learning objectives, both explicit and implicit, that professors identified when deciding the use computational simulations in their classrooms. This determination was carried out by conducting interviews of 14

engineering professors who implemented computational simulations developed by the Network for Computational Nanotechnology (NCN) into both graduate and undergraduate courses, mostly in the fields of Electrical, Computer, and Material Science Engineering. The interviews were analyzed through the theoretical framework of phenomenography and identified eight distinct categories of learning objectives, shown below in Figure 1.

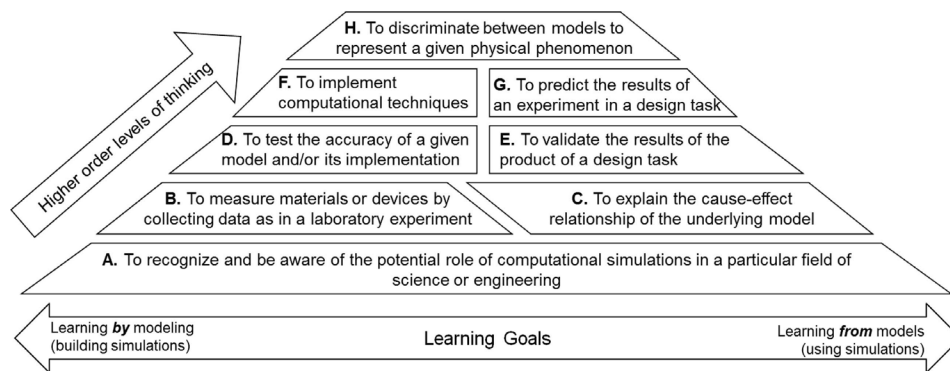


Figure 4.3: Learning Objectives for using simulations in teaching engineering¹⁶

This study initiated critical thinking about the implementation of the simulation tool in our study and lead to identification of 5 learning objectives to be accomplished using the software outlined later which roughly align to categories A-E in Figure 1, and additional objective based on the write up portion of the assignment. These learning objectives are:

- 1) The student should be able to identify and distinguish between various models of cell electrophysiology
- 2) The student should be able to use simulation tools to design and carry out an experiment
- 3) The student should be able to critically analyze the results of a simulated experiment in the context of the underlying model
- 4) The student should be able to critically assess the validity of simulated results based on their understanding of the underlying physiology
- 5) The student should be able to corroborate or contradict simulated results or their own critical analysis through external sources
- 6) The student should be able to effectively communicate the results and analysis of their experiment

These identified learning outcomes also roughly align to the requirements of the Accreditation Board of Engineering and Technology, Inc.¹⁷ criteria 3 a,b,e,g and k, listed below and this assignment is routinely used as an artifact to show compliance with these criteria.

Table 4.1: Selected ABET criteria for accredited programs in engineering¹⁷

Criteria	
(a)	an ability to apply knowledge of mathematics, science, and engineering
(b)	an ability to design and conduct experiments, as well as to analyze and interpret data
(e)	an ability to identify, formulate, and solve engineering problems
(g)	an ability to communicate effectively
(k)	an ability to use the techniques, skills, and modern engineering tools necessary for engineering practice.

4.1.4 Theoretical Framework

The theoretical framework chosen for the analysis in this study is the that of discovery learning. Discovery learning is a constructivist approach to education similar to (and in some applications synonymous with) Problem-Based Learning. While the literature presents a very broad range of definitions for discovery learning, Alfieri et al suggest the most important quality in defining it is that the learner is not directly given the information to be learned and must discover it himself through investigation within the confines of a specific task and given material¹⁸.

While its' effectiveness as an instructional method has been called into question¹⁹ especially among younger learners²⁰ or when pure discovery is relied upon²¹, recent work describing “enhanced discovery learning” in which necessary information and assistance needed to complete the task are provided by the instructor has led to renewed advocacy of the approach²².

4.2 Methods

4.2.1 Software

CellSpark is implemented as a MATLAB App. It is compatible with the current (at the time of this publication) version R2018a and backwards compatibility has been tested back to version R2014a. The full code is included in Appendix B but the software hierarchy and some implementation details will be included in this section.

The main program is contained in CellSpark.m. This serves as the bridge between the user interface (CellSpark.fig) and the script that actually runs the simulations

(run_simulation.m). CellSpark.m contains the code which initializes the user interface objects, defines their callbacks, and updates the user interface as the program is used interactively. Four different cell types can be chose: neurons, epicardial cells, endocardial cells, and cardiac M cells. Two different models are implemented in run_simulation.m: the Hodgkin-Huxley¹ model which controls the simulation if neuron is chosen, and the ten-Tusscher Noble Noble Paniflov⁷ (TNNP) model, which controls the simulation if any of the three cardiac cells are chosen. Any parameters (for either model) that can be controlled by the user are set in the user interface. The default settings for these parameters are the published values. Some additional advanced settings can be changed in Settings.m, which is the backend for the Settings.fig user interface.

Once the “Run Simulation” button of the user interface is pressed, run_simulation.m is called to begin the numerical estimate. Depending on the cell type chosen, initial values for the computed variables are pulled from the user interface (in CellSpark.m) or from a separate variables file. Variables for the Hodgkin-Huxley model (neuron) are stored in VariablesN.m and variables for the TNNP model (cardiac cells) are stored in Variables.m. The non-linear differential equation which describes each model is approximated using the forward Euler method. The time step is one of the advanced settings that can edited by the user. Each step of the approximation is performed by Step.m (TNNP) or StepN.m (Hodgkin-Huxley) and returned to run_simulation.m and CellSpark.m (to update the plots in realtime).

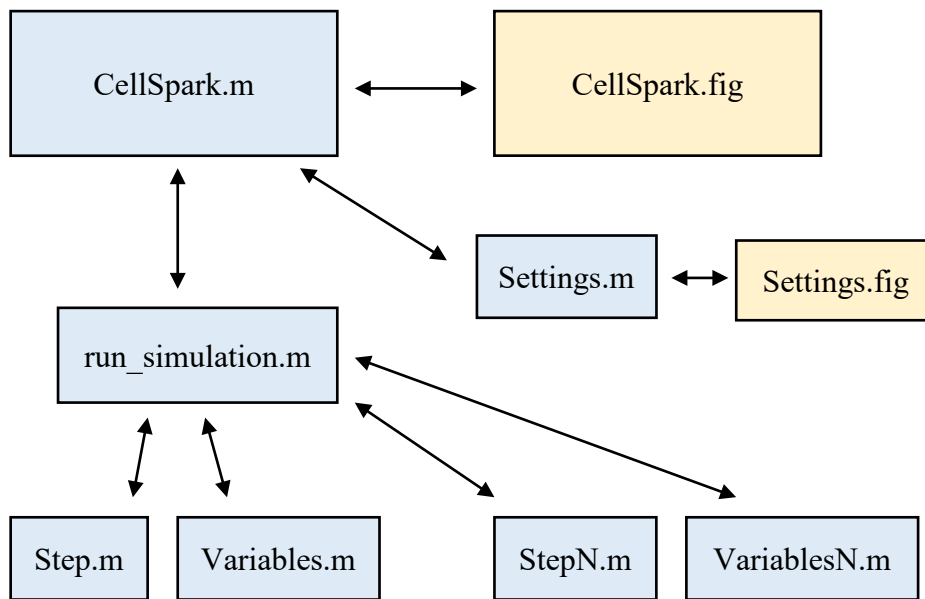


Figure 4.4: Overview of the CellSpark software dependencies. The full code is available in Appendix B.

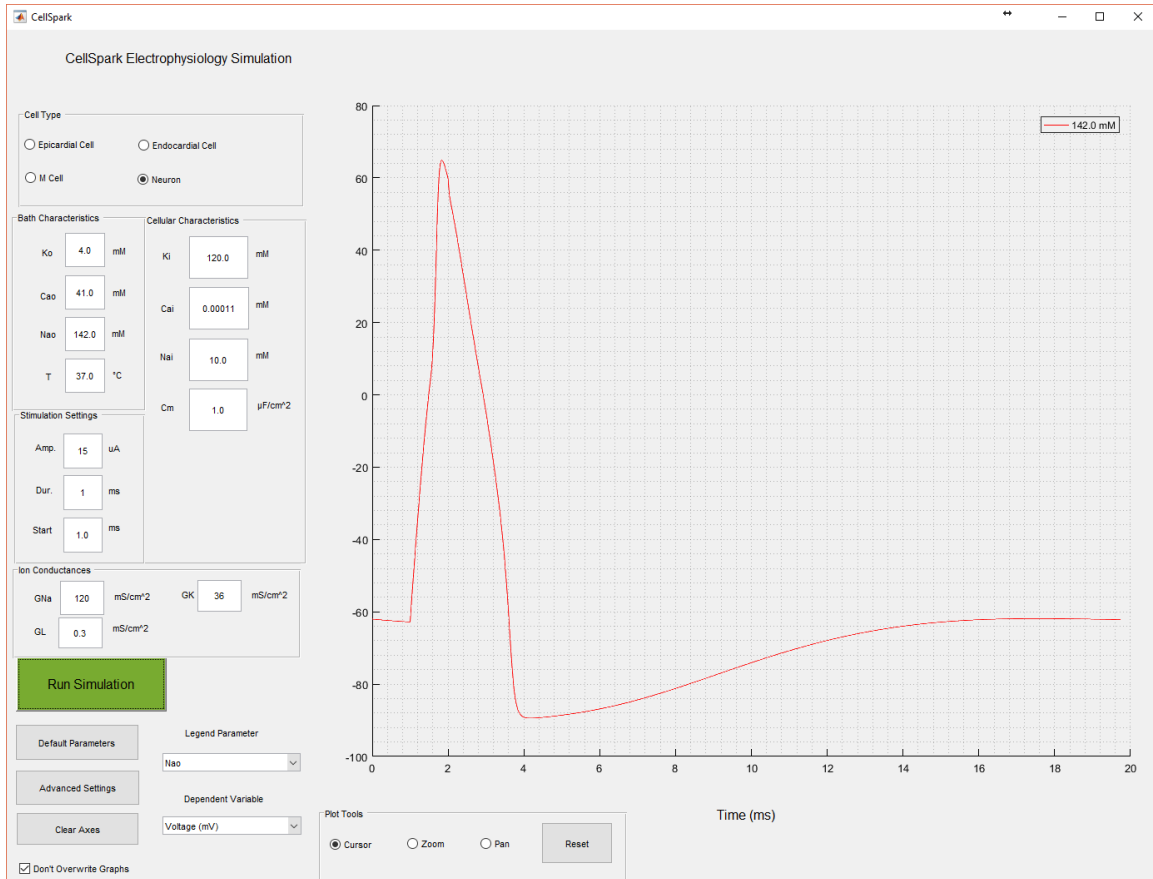


Figure 4.5: Screenshot of the CellSpark software interface. The cell type is currently set to “Neuron” and the characteristic Hodgkin-Huxley action potential is plotted. Other dependent variables that can be plotted include ionic currents, the values of gating parameters, and ion concentrations.

4.2.2 Study Parameters

CellSpark was developed for use in the laboratory portion of an undergraduate bioengineering course, Bioinstrumentation and Imaging, at a large public research institution in the southeastern United States. The course focuses on teaching the basic principles of physics, electronics, and physiology necessary to acquire, analyze, and interpret signals of biological origin. One third of the course is spent on electronics, one third on electrophysiology at the cell-organ levels, and one third on biomedical imaging modalities. The corresponding laboratory portion of the course closely follows the lecture section and covers the same topics. Enrollment in the course is typically 50-75 students per semester and is divided nearly evenly among males and females and among juniors and seniors, though class standing is often more skewed to seniors in the fall semester and juniors in the spring.

The course is part of the required curriculum for bioengineering students and is an elective for electrical engineering students. The breakdown by major is approximately 80% bioengineering students with a biomaterials concentration, 15% bioengineering with a bioelectrical concentration, and 5% electrical engineering students. All students have taken calculus through differential equations, physics II (electricity and magnetism), and some form of preparatory electrical engineering course (basic EE for non-majors or DC circuit analysis.)

This study focuses primarily on students' completion of the midterm lab assignment, which constitutes 15% of the final course grade. After brief exposure to the basics of cellular electrophysiology in lecture and a short tutorial exercise in lab, students

design and simulate an electrophysiology experiment on either a neuron or cardiac cell using the CellSpark software. After collecting data from the simulator, students write up their findings in a journal style article and present a critical discussion of the results in the context of physiology and the mathematical models upon which the simulation are based.

4.2.3 Assignments

A short introductory lecture presentation (slides are included in Appendix C.1) and tutorial lab exercise (included in Appendix C.2) were developed to introduce students to the field of electrophysiology, the mathematical models upon which CellSpark is based, and to introduce students to the software interface. The goals of the presentation were mainly to review the topics already discussed in lecture about action potentials and the ionic currents which initiate them, to familiarize students with the proper terminology to use when discussing electrophysiology concepts, and to give some helpful tips about what the teaching staff looks for when grading the assignment. The steps of the tutorial mainly served to introduce all of the parameters of the model, give an example of an “experiment” performed using the software, and to cause students to begin analyzing results of the software both quantitatively and qualitatively.

Two weeks after initial exposure to the software in lab, students had to submit their initial hypotheses for approval by the instructor and teaching assistants. Along with the hypothesis (generally of the form “if I increase/decrease X, I expect an increase/decrease in Y” - which essentially serves as their “plan” for conducting the experiment) the students had to give a logical argument to back up the hypothesis. Approval of the hypotheses was not based on correctness (or soundness of the argument)

but on suitability for testing with the software and for evidence of thinking critically about how the two parameters of the hypothesis relate, either in the context of physiology or the mathematics of the model.

Two weeks after having their hypotheses approved, students submitted their full lab reports written in the style of a short journal article. (The full assignment prompt and guidelines are presented in Appendix C.3.) While students were not forbidden from asking for help from the instructor or Teaching Assistants, only 3 students out of 65 sought additional help in interpreting the results of their experiments. Since the goal of this study is individual discovery learning, only explanations of previously taught topics were given, with the students encouraged to logically develop their own interpretation of the results.

4.2.4 Data Collection

Following submission and grading of their midterm lab reports, students were invited via email to participate in an anonymous survey about their use of the CellSpark software. The survey consisted of seven Likert-type scale questions to determine the ease of use of the software, the quality of the presentation and tutorial, the students' understanding of electrophysiology before and after completing the assignment, and preference to using the software over a traditional lecture based learning environment. The survey also featured a free response question for students to give additional comments or suggestions for improving the software.

The primary method to assess the identified learning outcomes was through content analysis of the submitted lab reports. The content of students' reports (primarily

the discussion sections) was analyzed with the intent of finding specific evidence of 1) the students' understanding of the basic process of action potential generation in a cell, 2) the students' ability to identify and interpret key elements of the mathematical models which influence their results, 3) the students' ability to think critically about the experiment and not simply rely on explanations given in lecture/lab and 4) the students' ability to find and evaluate external sources in support of or contradiction to their reasoning, without specifically being asked to.

4.3 Results

Authors Note: Three students were found to have plagiarized the assignment being examined in this study. Papers from these students were excluded from the analysis and since two of the students unenrolled in the course, they were not asked to complete the post assignment survey. It is unknown whether the third student who remained enrolled completed the anonymous survey.

4.3.1 Survey Questions

Students were solicited via email and during lecture to complete an optional online assessment concerning their use of the CellSpark software. Of the 141 students remaining in the course, 61 students participated in the survey. The survey asked students to indicate their level of agreement with 7 statements. The responses were quantized by assigning a value of 1-5 and normal distributions of responses were generated. These summary of response for each statement are presented below in figure 4.6.

An additional space was added to the survey for students to give any other comments about the software or suggestions to improve it. 16 of the 61 students who completed the survey gave a response and these are included here:

“I liked the interaction of the software but, coming from someone who was relatively new at learning about electrophysiology, I think that a key for what the abbreviations for things meant would make it easier to follow”

“I really liked using the software as a learning tool. When I got an interesting result for my experiment, it made me look into it more and I actually learned a lot.”

“Maybe account for more parameters, like denaturation or membrane composition!”

“I think there should be more of a clear way to export data from the graphs”

“Maybe have different colors to better distinguish lines if there are multiple (i.e. darker colors for the lighter background)”

“It [would] be cool if it had sound effects, like whooshes and Zaps!”

“It was very user-friendly and helped me to understand how certain parameters affect the action potential of different cell types.”

“There seem to be certain values in a relatively normal range that result in modelling errors. For example, using a neuron's default values but setting C_m to 1.1 or 1.7 results in discontinuities.”

“Could you be able to choose the colors you use for the graph? I am colorblind and some colors were, therefore, difficult to see and determine the shape of the graph. Literally couldn't see the yellow line at all and had to get someone with normal color vision to help me.”

“Add the ability to edit line colors to prevent to lines of similar colors being adjacent to each other”

“The software was great, clean interface and easy to use, and allowed me to better visualize the relationships between depolarization, repolarization, and the flux of ions.”

“I thought the software was interesting. I also thought it was pretty intuitive. I didn't know about the software beforehand and I appreciated getting to use it.”

“I thought the program was great. I was able to visualize and learn a lot that I don't feel I would have learned simply from lecture.”

“Great software! The only thing I would suggest to improve is to add more colors. I ran the program for 12 different values, graphing the curves for all values on one plot for direct comparison. There were only 6 colors (red-yellow) and then the colors repeated. This made the legend show the same color for 2 different values, which could be confusing for anyone other than the person who ran the simulation, themselves. I would say 20 colors would be sufficient.”

“Entering decimal values for variables sometimes caused weird graphs to be made, e.g. entering values like 40.111 deg C caused an extra spike to appear on an action potential that did not occur at 40 or 41 deg C. This might be due to calculator rounding errors.”

“It would be cool to see the code that was used in the program”

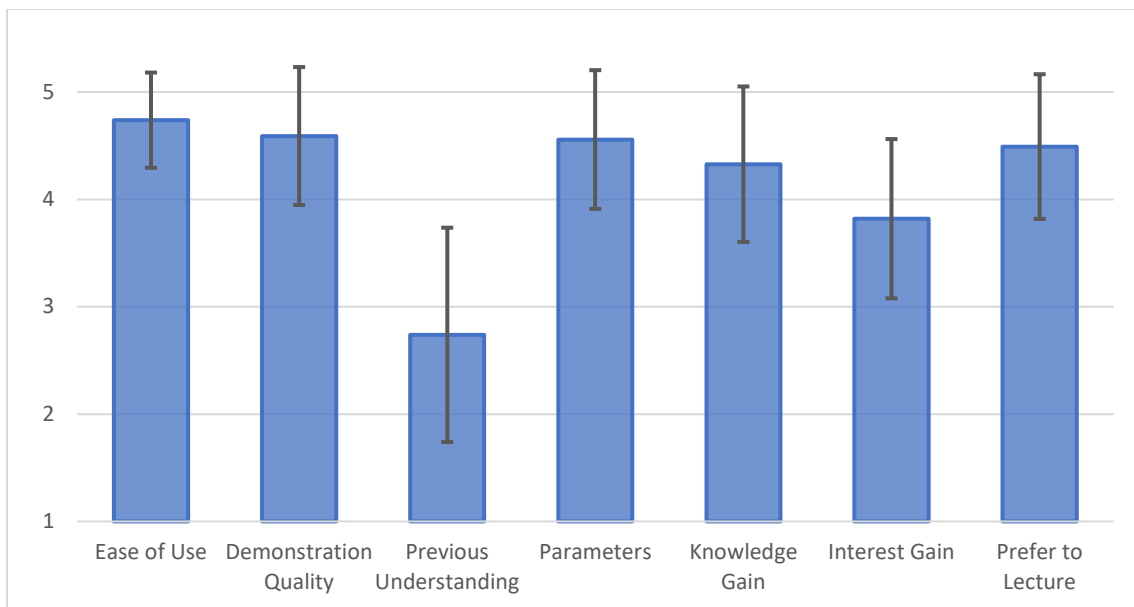


Figure 4.6: Combined responses to survey items (n=61)

4.3.2 Content Analysis – Learning Objectives

Of the 62 assignments completed by students, 13 of them were randomly selected to be used for the content analysis portion of this preliminary study. While there may be

overlap with the 61 survey responses, these papers were selected at random and do not represent the same set of students.

Learning Objective 1: The first learning objective identified in the implementation of this project was that students be able to identify and distinguish between various models of cellular electrophysiology. The specific models implemented in the software are the Hodgkin-Huxley model of a neuron and the TNNP model of cardiomyocytes. Of the 13 reports analyzed so far, 7 chose to perform experiments on cardiac cells and the remaining 6 chose neurons.

Only 1 of the 7 cardiac reports correctly identified that the TNNP model was being used, and this student reached out for extra assistance during the preparation of their reports and was confused about the distinction at that time. 5 of the students incorrectly identified the model as the Hodgkin-Huxley (which doesn't model cardiac cells) and 1 student failed to identify the model as TNNP, but did correctly state that the Hodgkin-Huxley only applies to neurons. The last student did not discuss either model in their paper.

However, the 6 students who chose to perform their experiments on neurons were all able to correctly identify the model used in their experiments as the Hodgkin-Huxley. It is important to note that this model is covered more extensively in the lecture portion of the course, whereas the only introduction to the TNNP model was in the short lecture on electrophysiology given in lab before introducing the software for the first time.

Learning Objectives 2 & 3: Since all students were able to complete the assignment, the second identified learning objective of students being able to use simulation tools to

design and carry out an experiment was completely achieved. However the third objective, that students should be able to critically analyze the results of a simulated experiment in the context of the underlying model, was not achieved by all students. Disregarding the student who made no reference to either model, students' papers tended to fall into one of three categories:

- Group I: 5 students demonstrated a good understanding of the model and were able to assess their results in the context of this understanding (fulfilling the objective)
- Group II: 4 students demonstrated an understanding of the model but only partially explained their results in the context of it, or made key errors in the analysis
- Group III: 3 students demonstrated an understanding of the model, but failed to discuss their results in the context of it at all

Learning Objective 4: The alternative approach to discussing the results in the context of the mathematical model is to discuss them in the context of what is happening physiologically at the cellular and ionic levels. Due to this, it was expected that students would primarily take one approach or the other – resulting in three similarly sized groups with the opposite trend – so the fourth identified learning objective was that student should be able to critically assess the validity of their simulated results based on their understanding of the underlying physiology.

However, all 5 students in Group I, who showed the best understanding of and ability to interpret the model *also* showed the best understanding of what happens at the physiological level, with only a few minor incorrect details. In Group II, two of the students showed a good understanding of the physiology while the third failed to discuss it and the last student had significant mistakes in their understanding. In Group III, only one student showed a partial understanding.

4.3.3 Content Analysis – The Case of the Black Mamba

Here we will highlight one specific case study which demonstrates what we aimed to achieve by creating the CellSpark software. In the third semester of the software's use in BIOE 3700 (Fall 2018), one student posed the following hypothesis:

“I hypothesize that the venom of the Black Mamba snake will increase the duration of the action potential of the neuron cell.”

The CellSpark software does not contain an option directly to alter this parameter, but this student, through their own outside research found that the primary component of Black Mamba venom is dendrotoxin. This student learned that dendrotoxin functions as a potassium channel blocker. In order to model this, the student proposed altering the maximum potassium conductance. At the time, this was not a changeable parameter in the software, but it was added at the student's request. This example highlights the type of active learning that CellSpark was designed to encourage. The student had some creative interest, engaged with the mathematical model enough to understand how their hypothesis could be tested, and requested a feature be added to the software. Once added, the student successfully tested their hypothesis and was able to relate the physiological response they observed back to the symptoms of a Black Mamba bite.

4.4 Discussion

4.4.1 Implications

In order for discovery learning to take place, students must begin the assignment with relatively little content knowledge of the material being covered or there is nothing for them to ‘discover’ by performing the exercise. Responses to survey item S3 showed that this criteria was true for the average student, who felt they did not have a strong

understanding of electrophysiology prior to using the CellSpark software, though it is worthwhile noting this item had the highest variation of the survey, showing that students come in with a variety of skill levels. Another key component of a discovery learning framework is that students are given only the basic tools required to complete the task so that they succeed in discovering the content knowledge through completion of the task. A key design criteria of the CellSpark software was that be simple and intuitive to use, as to minimize the barrier to entry, while containing enough complexity for meaningful scientific inquiry to take place. Affirmative responses to survey items S1, S2 and S5 indicate that this goal was successfully met, though one of the students expressed a desire for more detail in the interface in their free response.

Survey item S5 assessed the utility of the software for improving student understanding of the material (or at least their self-assessment of understanding) and the overall affirmative response is promising. Moreover, the affirmative responses to survey item S6 show that the exposure to the software also increased students interest in the content, though to a lesser extent. Surprisingly, students indicated in survey item S7 that they strongly prefer the learning activity to a traditional lecture environment.

The learning objectives identified for this study can be roughly assigned into a hierarchy similar to the one presented by Magana et al or to Bloom's taxonomy. Because of this, it was assumed that learning objective 1 would be the easiest for students to achieve, being the lowest in the hierarchy. However, as the results show this was clearly not the case as most students failed to correctly identify the model being used. One potential implication of this finding is that the timing and setting of content delivery

significantly impacts the ability of students to correctly learn the content. The students in this study pretty overwhelmingly failed to process the information that was presented immediately before introduction of the software/assignment and were unable to recall the content when preparing their reports 2-4 weeks later. On the other hand, content which was introduced prior to introducing the assignment and then expanded upon in more detail at a later date was able to be recalled and correctly related to the assignment. This is further evidenced by the fact that all five students who misidentified the model did so by choosing the one that was covered in lecture (which presumably they were more familiar with.)

Since the students who make up this class come from primarily two backgrounds – bioelectrical engineering and biomaterials engineering – it was expected that two different groups of students would emerge from the content analysis: those explaining their results in the context of physiology vs those explaining in the context of the mathematical model. As the results indicated, instead the groups consisted of students who were successfully able to implement *both* explanations, those who could partially explain one or the other, and those who really failed to show mastery of either explanation. Coupled with the survey data which showed most students did not consider themselves as having strong prior knowledge but that their knowledge improved after intervention, this finding lends strong support to the CellSpark platform as a learning tool. The implication of this finding is that the mathematical models are not just important research tools in electrophysiology, but are effective for teaching it as well. At this point it is unclear whether students developed a better understanding of the

physiology based on their understanding of the model or the other way around, but it is evident that students with strong understandings of both were able to better think critically about the experiments they performed.

Lastly, as the case study we highlighted shows, a discovery learning task requires students to engage with course content in a more active way. It allows students some degree of creativity and control over their learning which enhances their perceived knowledge and interest gains in the material, as demonstrated by the survey responses. Further, students strongly prefer learning difficult concepts, like electrophysiology, through experimentation rather than a traditional lecture environment.

4.4.2 Limitations

One potential limitation of this study is that content analysis of student reports may not be sufficient to measure the impact of the software on development of content knowledge as exposure to the content in other portions of the course may also influence its mastery. Additionally, as the three students caught plagiarizing on this assignment demonstrate, today's university student has access to many resources, including the work of past students so critical insights in their writing, even if presented as original thought, may not necessarily demonstrate content mastery. In an expanded follow up study this could be addressed by incorporating additional methods of assessing understanding of the content such as concept mapping, performance on related exam questions, and individual interviews with students about their experience with the assignment.

Another possible limitation of the current approach is that the students who chose to respond to the survey as well as those whose reports were randomly selected for

analysis may not give an accurate representation of the overall class population. While the survey response concern can't be easily controlled for (aside from making participation a mandatory part of the assignment) the latter can be validated by comparing the assignment grades of students whose papers were analyzed to the grades of the class as a whole. The assignment was graded by splitting the 65 submitted papers into 5 groups of 13, with each group being graded by a separate teaching assistant according to a uniform rubric (Appendix C). All 65 papers were then read separately by the course instructor who adjusted the grades to account for any variance between the teaching assistants. The statistical comparison of the analyzed papers to the whole class is presented below, showing no statistical difference between the two groups.

Table 4.2: Statistical analysis of assignment grades (analyzed papers vs all papers)

	Analyzed papers (n=13)	Entire class (n=62)
Mean score (out of 20)	18.269	18.066
Standard deviation	1.235	1.706
2-Tailed T-Test (unequal variance)	p=0.622	

4.5 Conclusions

This pilot study offers clear evidence that the CellSpark application has the potential to be a powerful tool for electrophysiology education and demonstrates the feasibility of using simulation tools primarily designed for research and a discovery learning framework as effective strategies for undergraduate engineering education.

4.6 References

1. Huxley, A. L., Hodgkin AF. A Quantitative Description of Membrane Current and its Application to Conduction and Excitation in Nerve. *J Physiol*. 1952;(117):500-544. doi:10.1007/BF02459568.
2. Fitzhugh R. Impulses and Physiological States in Theoretical Models of Nerve Membrane. *Biophys J*. 1961;1(1948):445-466. doi:10.1016/S0006-3495(61)86902-6.
3. Nagumo J, Arimoto S, Yoshizawa S. An Active Pulse Transmission Line Simulating Nerve Axon. *Proc IRE*. 1962;50(10):2061-2070. doi:10.1109/JRPROC.1962.288235.
4. Aliev RR, Panfilov A V. A Simple Two-variable Model of Cardiac Excitation. *Chaos, Solitons and Fractals*. 1996;7(3):293-301. doi:10.1016/0960-0779(95)00089-5.
5. Luo CH, Rudy Y. A model of the ventricular cardiac action potential. Depolarization, repolarization, and their interaction. *Circ Res*. 1991;68(6):1501-1526. doi:10.1161/01.RES.68.6.1501.
6. Tung L, Borderies J. Analysis of electric field stimulation of single cardiac muscle cells. *Biophys J*. 1992;63:371-386. <http://www.sciencedirect.com/science/article/pii/S0006349592816326>. Accessed August 21, 2014.
7. ten Tusscher KH, Noble D, Noble P., Panfilov A. A model for human ventricular tissue. *Am J Physiol - Hear Circ Physiol*. 2004;286(4):H1573-H1589.
8. National Acadmemy of Sciences, National Academy of Engineering I of M. *Rising Above the Gathering Storm*. Washington, D.C.: National Academies Press; 2007. doi:10.17226/11463.
9. Duderstadt JJ. *Engineering for a Changing World: A Roadmap to the Future of Engineering Practice, Research, and Education*. Ann Arbor, MI: The Millennium Project; 2007.
10. Silk EM, Schunn CD, Strand Cary M. The impact of an engineering design curriculum on science reasoning in an Urban setting. *J Sci Educ Technol*. 2009;18(3):209-223. doi:10.1007/s10956-009-9144-8.

11. Barron, Brigid JS. BARRON_1998-Doing with understanding_Lessons from research on problem and project based learning.pdf. *J Learn Sci.* 1998;7(3&4):271-311.
12. Johnson, T, Chen, H, Suh, EK, Kim P. Innovative design through creative thinking in design based learning. *Forthcoming*.
13. Chung GKWK, Harmon TC, Baker EL. The impact of a simulation-based learning design project on student learning. *IEEE Trans Educ.* 2001;44(4):390-398. doi:10.1109/13.965789.
14. Campbell JO, Bourne JR, Mosterman PJ, Brodersen AJ. The effectiveness of learning simulations for electronic laboratories. *J Eng Educ.* 2002;91(1):81-87. doi:10.1002/j.2168-9830.2002.tb00675.x.
15. Wankat PC. Integrating the Use of Commercial. *J Eng Educ.* 2002;91(1):19-23.
16. Magana AJ, Brophy SP, Bodner AM. Instructors' intended learning outcomes for using computational simulations as learning tools. *J Eng Educ.* 2012;101(2):220-243. doi:10.1002/j.2168-9830.2012.tb00049.x.
17. Commission AEA. Criteria for Accrediting Engineering Programs.
18. Alfieri L, Brooks PJ, Aldrich NJ, Tenenbaum HR. Does Discovery-Based Instruction Enhance Learning? *J Educ Psychol.* 2011;103(1):1-18. doi:10.1037/a0021017.
19. Adelson R. Instruction versus exploration in science learning. *Monit Psychology APA.* 2004;35(6):34. <https://www.apa.org/monitor/jun04/instruct.aspx>.
20. Stokke A. What to Do About Canada's Declining Math Scores? *Comment CD Howe Inst.* 2015;(427). doi:10.2139/ssrn.2613146.
21. Mayer RE. Should There Be a Three-Strikes Rule against Pure Discovery Learning? The Case for Guided Methods of Instruction. *Am Psychol.* 2004;59(1):14-19. doi:10.1037/0003-066X.59.1.14.
22. Marzano RJ. Art & Science of Teaching: The Perils and Promises of Discovery Learning. *Educ Leadersh Promot Respectful Sch.* 2011;69(1):86-87. <http://www.ascd.org/publications/educational-leadership/sept11/vol69/num01/The-Perils-and-Promises-of-Discovery-Learning.aspx>.
23. Wikimedia user: Krishnavedala - Own work.

CHAPTER FIVE

CONCLUSIONS AND RECOMMENDATIONS FOR FUTURE WORK

5.1 Conclusions

The main motivation for this work is a need to increase understanding of the structure-function relationships of cells and the impacts those have on cellular and tissue level mechanics and electrophysiology. As we move into a new era of medicine, where tissues and organs are engineered and grown, having a wide variety of methods at our disposal to investigate the properties of those constructs will be vital. This work represents a step toward developing those tools.

In Chapter 2, we demonstrated a novel, fully automated algorithm to develop geometries of cardiomyocytes for use in finite element modelling (FEM) studies. This algorithm can be used with confocal images as inputs or with cellular geometries designed using CAD tools. We demonstrated the customizability of the algorithm and showed that it can estimate subcellular geometries regardless of image resolution.

In Chapter 3, we used those geometries to create scalable and customizable FEM simulations of single cell mechanics. These simulations, developed automatically, were capable of replicating experimental cell mechanics measurements by modeling cardiomyocytes as fiber composites, which resulted in low computational complexity. We also demonstrated the sensitivity of the model to myofibril diameter/spacing to show the viability of scaling the model to tissue and organ levels.

In Chapter 4, we developed a software simulation tool that implements common models of cellular electrophysiology. This tool was designed for use in an undergraduate bioengineering course to allow discovery learning of electrophysiology to take place. We successfully launched in the software in the course BIOE 3700: Bioinstrumentation and Imaging and demonstrated the effectiveness of simulation tools in undergraduate engineering education.

5.2 Recommendations for Future Work

1. Extend the model to include additional subcellular components. While nuclear membranes were segmented in the method presented in Chapter 2, these were excluded from the model geometries in Chapter 3 for the sake of computational efficiency. Additional subcellular geometries, such as F-actin, microtubules and cell organelles could also be included. The model could also be improved by using clues from imaging to refine the approach. For example, staining for connexins or integrins could be used to develop a more sophisticated method of determining fiber termination points. Some preliminary work was also performed that shows the presence of the glycocalyx has a significant impact on whole cell mechanics measurements and additional modelling to explore this effect could yield interesting insights.
2. Extend the model to replicate more physical experiments. A contact simulation would provide more information about the transverse mechanical properties than our current simplifications can provide at the expense of additional computational cost. Contraction studies could be performed without much modification of the current framework.

3. Incorporate electrophysiology and excitation-contraction coupling into the finite element model. As you move from the cell to tissue level in the heart, not only are their implications for the mechanics of the tissue, but also the electrophysiology. In our additional work, we show that present framework could easily be extended to include simple models of cardiac electrophysiology without increasing the computational complexity too much. Coupling this electrophysiology to the mechanics by modeling the excitation-contraction coupling is the next step towards creating a representative model of the cardiac environment.

4. Model the transient mechanical behavior of cells. The modelling framework as it's currently presented only looks at the steady-state behavior of cells under stretch, but as the heart is one of the most dynamic environments in the body, an understanding of the transient behavior is also necessary. A minor change to the model to include viscoelastic properties to the cell would allow these time-dependent studies to be performed. However, this analysis is much more computationally intensive, which is the primary reason it was not performed in the current study. A high-throughput computing environment, such as Clemson's Palmetto Cluster, may be necessary to perform these simulations.

5. Extend the model to higher scales of complexity. While the structure-function relationships and mechanics of single cells are interesting, our primary goal is to determine the implications of these relationships at the tissue and organ level. Extending the model to the tissue level is feasible with only minor modification (and an increase in computational complexity) and would allow us to work towards this goal.

6. Extend the model to additional cell types. With modifications, the approach presented here could be applied to other cell types (particularly skeletal and smooth muscle) of interest to mechanobiology.

7. Refine the CellSpark software and create other simulation tools for engineering education. While we saw great success with implementing the CellSpark software into the curricula, there are still improvements that can be made to the software, mostly with the interface and data handling. We would also like to incorporate additional models of electrophysiology and different numerical techniques which can help to better illustrate some of the difficulties of simulation tools. Documentation of the software should be developed to increase its effectiveness as a software tool. Other simulation tools for engineering education can also be developed to allow active learning of concepts. Some potential concepts that could be explored include fluid dynamics, diffusion and heat transfer, and biomechanics.

APPENDICES

APPENDIX A

MATLAB CODE FOR GENERATING MYOFIBRIL DISTRIBUTIONS

A.1: cell6.m

This file must be manually created for each cell to be analyzed. It points to the directory containing images of the cell, assigns an identifier to the cell (the name of the file), performs the segmentation of the geometry, and writes the STL files of the cell and nuclei meshes.

```
%import sequential TIFF images into one stacked matrix, and generate a
%black and white stack of the images (25% intensity threshold)
z = 52;
[im bw] = Import_Confocal_Stack_sep...
('C:\Users\tgharve\OneDrive\Research\Image Processing\Confocal
Images\typical picture\ACM 3\Isolated Adult
CM_1.lif_Series017_z','_ch0.tif',z,0.15);
[nuc nbw] = Import_Confocal_Stack_sep...
('C:\Users\tgharve\OneDrive\Research\Image Processing\Confocal
Images\typical picture\ACM 3\Isolated Adult
CM_1.lif_Series017_z','_ch2.tif',z,0.15);
%estimate the volume and generate a binary matrix(1 inside/0 outside).
Use
[a b c] = size(im);

im_scaledxy = imresize(im,0.19);
nuc_scaledxy = imresize(nuc,0.19);

mrows = size(im_scaledxy,2); %the second dimension is already the right
size
mcols = round(0.346*size(im_scaledxy,3)); %we want to rescale the third
dimension

for i = 1:size(im_scaledxy,1)
    B(:, :) = im_scaledxy(i, :, :); %make a 2D array with the last two
dimensions of A1
    B1 = imresize(B, [mrows, mcols]);
    im_scaled(i, :, :) = B1;
end

for i = 1:size(nuc_scaledxy,1)
    B(:, :) = nuc_scaledxy(i, :, :); %make a 2D array with the last two
dimensions of A1
    B1 = imresize(B, [mrows, mcols]);
    nuc_scaled(i, :, :) = B1;
end
```

```

vol_nuc = estimateBinaryVolume_2(nuc_scaled,10,.02);
vol_nuc = bwareaopen(vol_nuc,100);

mem = estimateBinaryVolume_2(im_scaled,10,.02);
mem = bwareaopen(mem,200);

vol = mem - vol_nuc;

[im_rot,ang,CoM] = regImage(im_scaled);
vol_rot = regImage2(vol,-ang,CoM);
nuc_rot = regImage2(vol_nuc,-ang,CoM);
mem_rot = regImage2(mem,-ang,CoM);

gauss = fspecial('Gaussian',3,1);
[a b c] = size(vol_rot);
for k = 1:c
    vol_smoothed(:,:,k) =
imfilter(imfill(vol_rot(:,:,k),'holes'),gauss);
    nuc_smoothed(:,:,k) =
imfilter(imfill(nuc_rot(:,:,k),'holes'),gauss);
end

vol_bound = vol_rot;

%SE = strel('sphere',1);
%membrane = imdilate(vol_smoothed,SE);

[faces,vertices] = isosurface(vol_smoothed,0.01);
stlwrite('3D Model Files/cell6.stl',faces,vertices,'mode','ascii');

%[faces_m,vertices_m] = isosurface(membrane,0.01);
%stlwrite('3D Model
Files/membrane6.stl',faces,vertices,'mode','ascii');

[faces_n, vertices_n] = isosurface(nuc_smoothed,0.01);
stlwrite('3D Model Files/nucleus6.stl',faces_n,
vertices_n,'mode','ascii');

FVc.vertices = vertices; FVc.faces = faces;
FVn.vertices = vertices_n; FVn.faces = faces_n;
pc = patch(FVc);
pn = patch(FVn);
set(pc, 'facecolor', [0.6 0.4 0.4]);
set(pc, 'facealpha', 0.6); %translucency
set(pc, 'linestyle', 'none'); % uncomment to hide mesh
set(pn, 'facecolor', [0 0 1]);
set(pn, 'linestyle', 'none');

axis equal

```

```
grid off          % undo by  grid off
axis vis3d

daspect([1 1 1]);
%axis xy;
camlight;
lighting phong;

hold on
```

A.2: Import_Confocal_Stack_sep.m

```
function [A,B]= Import_Confocal_Stack_sep(froot,froot2,n,th)
if n>9
    for k = 0:9
        A(:,:,k+1) = imread(strcat(froot,'00',num2str(k),froot2));
        B(:,:,k+1) = im2bw(A(:,:,k+1),th);
        % ... Do something with image A ...
    end
    for k=10:n
        A(:,:,k+1) = imread(strcat(froot,'0',num2str(k),froot2));
        B(:,:,k+1) = im2bw(A(:,:,k+1),th);
    end
else
    for k = 0:n
        A(:,:,k+1) = imread(strcat(froot,'00',num2str(k),froot2));
        B(:,:,k+1) = im2bw(A(:,:,k+1),th);
    end
end
end
```

A.3: estimateBinaryVolume_2.m

```
function [vol] = estimateBinaryVolume_2(im,r,th)
%imf = imgaussfilt3(im);
[a,b,z] = size(im);
for stack=1:1:z
vol(:,:,stack) = estimateBinaryArea_2(im(:,:,stack),r,th);
end
```

A.4: estimateBinaryArea_2.m

```
function [mask] = estimateBinaryArea_2(im,r,th)
SE = strel('disk',4);
mask = im2bw(im,th);
mask = imdilate(mask,SE);
mask = imfill(mask,'holes');
mask = imerode(mask,SE);
```

A.5: regImage.m

```
function [imout2, ang, CenterOfMass] = regImage(im)
%bw = im2bw(im,.05);
[a b c] = size(im);
bw = im(:,:,floor(c/2));
[x y] = find(bw);
CenterOfMass = floor([mean(x) mean(y)]);
CenterOfImage = floor([a b]./2);
x1 = prctile(x,25);
y1 = prctile(y,25);
x2 = prctile(x,75);
y2 = prctile(y,75);
ang = 57.2958*atan((y2-y1)/(x2-x1));
for k =1:c
    imout(:,:,k) = imtranslate(im(:,:,k),CenterOfImage - CenterOfMass);
    imout2(:,:,k) = imrotate(imout(:,:,k),ang);
end
```

A.6: regImage2.m

```
function [imout] = regImage2(im, ang, CenterOfMass)
[a b c] = size(im);
CenterOfImage = floor([a b]./2);
for k =1:c
    temp(:,:,k) = imtranslate(im(:,:,k),CenterOfImage - CenterOfMass);
    imout(:,:,k) = imrotate(temp(:,:,k),ang);
end
```

A.7: paths_nosweep.m

This script initiates the algorithm for a single cell and must be modified to point to the particular cell being studied. Additionally this is where the cell spacing parameter is specified and where an initial guess of the number of paths to be placed can be provided to speed up computation time. This script requires the Mosek Optimization Toolbox to be installed and for COMSOL with MATLAB to be running and the functions kmeans, stlwrite, and extrude, all of which are available for download on the Mathworks User Community File Exchange.

```
clear
clc
close all
recon = figure;
addpath 'C:/Program Files/Mosek/7/toolbox/r2013a'
import com.comsol.model.*
import com.comsol.model.util.*

model = ModelUtil.create('Model');
geom1 = model.geom.create('geom1', 3);
model.geom('geom1').lengthUnit('um');

cell15
filename = 'cell15_th';
sp = 4; %c3=5 c4=4 c6=4 c7=4
maxpaths = 2;
found = true;

%break for cylinder fitting

[Vx Vy Vz] = ind2sub(size(vol_bound), find(imfill(vol_bound, 'holes')));

flag = true;
bestscore = 1e20;
k = 3;
while flag
    [lab eng m] = kmeans([Vx'; Vy'; Vz'], k);
    [f1 f2] = sortrows(m.means', 1);
    for i=1:length(f2)
        lab2(find(lab==f2(i)))=i;
    end
end
```

```

end
for i=1:max(lab)
    cluster{i} = zeros(size(vol_bound));
end
clustered_vol = vol_bound;
imfill(clustered_vol, 'holes');
clustered_vol(clustered_vol == 1) = -1;

for i =1:length(lab2)
    cl = lab2(i);
    cluster{cl}(Vx(i),Vy(i),Vz(i)) = 1;
    clustered_vol(Vx(i),Vy(i),Vz(i)) = cl;
end
for i=1:length(cluster)
    [cx{i} cy{i} cz{i}] =
ind2sub(size(cluster{i}),find(cluster{i}));
end
for i=1:length(cx)
    col{i} = cx{i};
    col{i}(:) = i;
end

for i=1:length(cluster)
    [ax{i} r{i}] = fitcylinder(cluster{i});
end

for i=1:length(cluster)
    %score based on average distance to cylinder surface
    %    D = zeros(a,b,c);
    %    for x=1:a
    %        for y=1:b
    %            for z=1:c
    %                if cluster{i}(x,y,z) == 1
    %                    D(x,y,z) =
distanceToCylSurf(ax{i},r{i},[x,y,z]);
    %                end
    %            end
    %        end
    %    end
    %    d(i) = mean(nonzeros(D));

    %score based on volume occupied
    v(i) = r{i}^2*pi()*norm(ax{i}(:,2)-ax{i}(:,1)) -
nnz(cluster{i});

end

%score = mean(d)
score = sum(v)

if score < bestscore

```



```

        best.cluster = cluster;
        best.ax = ax;
        best.r = r;
        best.clustered_vol = clustered_vol;
        %best.d = d;
        best.v = v;
        best.cx = cx;
        best.cy = cy;
        best.cz = cz;
        best.col = col;
        bestscore = score;
        k = k+1;
    else
        flag = false;
    end
    %k = k+1;
end

figure(recon);

%end break for cylinder fitting

while found
    clear temp;

    [N, demand] =
generateNodes3(vol_bound,best.clustered_vol,sp,maxpaths);
    demand2 = scaleDemand(demand);
    [edges, c] = generateEdges3(N,sp);
    [from, to, costs, upper] = matrix2ft(edges, c);
    lower = zeros(size(upper));
    G = digraph(c);
    I = incidence(G);
    param.MSK_IPAR_OPTIMIZER = 'MSK_OPTIMIZER_NETWORK_PRIMAL_SIMPLEX';
    param.MSK_IPAR_INFEAS_REPORT_AUTO = 'MSK_OFF';
    demand_l = zeros(size(demand2));
    sol = msklpopt(costs,I,demand2,demand2,lower,upper,param);

    %scatter3(N(:,2),N(:,1),N(:,3),20,N(:,5),'filled')

    temp(:,3) = sol.sol.bas.xx;
    temp(:,1) = from';
    temp(:,2) = to';
    [s1 s2] = size(temp);
    [nNodes s2] = size(N);

    for i = 1:s1
        if temp(i,2) <= nNodes
            temp(i,1) = temp(i,1) - nNodes;
        end
    end
    TF = temp(:,2) > nNodes;

```

```

temp(TF,:) = [];
TF = temp(:,3) == 0;
temp(TF,:) = [];

[s1 s2] = size(temp);
count = 1;
count2 = 1;
for i=1:s1
    if temp(i,1) ~= 1 && temp(i,2) ~= nNodes
        temp2(count,:) = temp(i,:);
        count = count+1;
    end
    if temp(i,1) == 1
        start(count2,:) = temp(i,2);
        count2 = count2+1;
    end
end
[s1 s2] = size(temp);
%scatter3(N(:,2),N(:,1),N(:,3),20,N(:,5),'filled')
axis equal
hold on
for k = 1:s1

%plot3([N(temp(k,1),2),N(temp(k,2),2)], [N(temp(k,1),1),N(temp(k,2),1)],
[N(temp(k,1),3),N(temp(k,2),3)], '-k')
end

%     start = N(:,5)';
%     start(start ~= -1) = 0;
%     start = find(start);
%
%     for i = 1:length(start)
%         if ~ismember(start(i),temp2(:,1))
%             start(i) = 0;
%         end
%     end
%     start = start(start~=0);

S=digraph(temp2(:,1),temp2(:,2));

myo_vol = zeros(size(start));
myo_len = zeros(size(start));
found = strcmp(sol.sol.bas.solsta,'OPTIMAL');
if found
    maxpaths = maxpaths+1;
    for i = 1:length(start)
        paths{i} = shortestpathtree(S,start(i));
    end
end
end
maxpaths = maxpaths - 1;
disp(strcat(num2str(maxpaths),' myofibrils generated. '));

```

```

% figure
count = 1;
for i = 1:length(paths)
    asdf = table2array(paths{i}.Edges);
    [s1 s2] = size(asdf);
    asdf(s1+1,1) = asdf(s1,2);
    for k = 1:s1+1
        asdf(k,3) = N(asdf(k,1),1);
        asdf(k,4) = N(asdf(k,1),2);
        asdf(k,5) = N(asdf(k,1),3);
    end

    q = linspace(0,2*pi,33);
    base = [(sp/2)*cos(q); (sp/2)*sin(q)];
    traj = [asdf(:,3)'; asdf(:,4)'; asdf(:,5)'];
    hull_nodes(count,:) = traj(:,1)';
    count=count+1;
    hull_nodes(count,:) = traj(:,size(traj,2))';
    count = count+1;

    ic = strcat('ic',num2str(i));

    num = size(traj,2)-1;
    model.geom('geom1').feature().create(ic, 'InterpolationCurve');
    traj_sw = [traj(2,:); traj(1,:); traj(3,:)];
    model.geom('geom1').feature(ic).set('table',traj_sw);
    points{i} = traj';

    [X,Y,Z] = extrude(base, traj,1);
    myo = surf(Y,X,Z);
    set(myo, 'facecolor', [.8 0 0]);
    set(myo, 'facealpha', .5);
    for j = 1:size(traj,2)-1
        seg_len = sqrt((traj(1,j+1)-traj(1,j))^2 + (traj(2,j+1)-
traj(2,j))^2 + (traj(3,j+1)-traj(3,j))^2);
        seg_vol = pi*sp^2/4 * seg_len;
        myo_vol(i) = myo_vol(i) + seg_vol;
        myo_len(i) = myo_len(i) + seg_len;
    end
end
geom1.feature('fin').name('Form Assembly');
geom1.feature('fin').set('action','assembly');
geom1.feature('fin').set('imprint',true);
geom1.feature('fin').set('createpairs',false);
geom1.run
model.save(strcat('C:/Users/tgharve/Desktop/COMSOL Models/',filename));

sv = generateSideView(vol_bound);
c = size(sv,3);
for i = 1:c
    csa(i) = nnz(sv(:, :, i));
end
d_ideal = 2*sqrt((sum(csa)/nnz(csa))/pi);

```

```

rat = d_ideal/sp;
f = 0.5919*exp(.0509*rat)-(7.041e+12)*exp(-15.72*rat);
%fillscore = (sum(myo_vol)/nnz(vol_smoothed))/f * 100;
fillscore = (sum(myo_vol)/nnz(vol_smoothed)) * 100;
lengthscore = mean(myo_len)/nnz(csa) * 100;

figure
hold on
for i=1:length(best.cluster)
    q = linspace(0,2*pi,33);
    base = [best.r{i}*cos(q);best.r{i}*sin(q)];
    [X,Y,Z] = extrude(base,best.ax{i},1);
    cyl{i} = surf(X,Y,Z);
    %set(cyl{i}, 'facecolor', [0.6 0.4 0.4]);
    set(cyl{i}, 'facealpha', 0.6);
end
axis equal

```

A.8: fitcylinder.m

```
function [ax r] = fitcylinder(vol)
[a b c] = size(vol);
[Vx Vy Vz] = ind2sub(size(vol),find(vol));
MA = Skeleton3D(imfill(vol,'holes'));
[MAx MAy MAz] = ind2sub(size(vol),find(MA));
dist = bwdistsc(~imfill(vol,'holes'));
% r = max(nonzeros(MA.*dist));
%r = ceil(max(nonzeros(dist)));

%[m p s] = best_fit_line(MAx, MAy, MAz);
[m p s] = best_fit_line(Vx, Vy, Vz);

bnd1 = (min(Vx)-m(1))/p(1);
bnd2 = (max(Vx)-m(1))/p(1);

ax = [m(1)+p(1)*bnd1, m(2)+p(2)*bnd1, m(3)+p(3)*bnd1;...
      m(1)+p(1)*bnd2, m(2)+p(2)*bnd2, m(3)+p(3)*bnd2]';

D = zeros(a,b,c);

for x=1:a
    for y=1:b
        for z=1:c
            if vol(x,y,z) == 1
                D(x,y,z) = distanceFromAxis(ax,[x,y,z]);
            end
        end
    end
end
r = max(nonzeros(D))+1;

bnd1 = ((min(Vx)-r*sin(acos(p(1))))-m(1))/p(1);
bnd2 = ((max(Vx)+r*sin(acos(p(1))))-m(1))/p(1);

ax = [m(1)+p(1)*bnd1, m(2)+p(2)*bnd1, m(3)+p(3)*bnd1;...
      m(1)+p(1)*bnd2, m(2)+p(2)*bnd2, m(3)+p(3)*bnd2]';
```

A.9: generateNodes3.m

```
function [N, demand, cent] = generateNodes3(vol, cl_vol, d, s)
%given a binary cell volume (rotated so that the long axis of the cell
is
%in the x direction) and a distance between nodes (d), this function
%generates a regular grid of nodes (N) within the volume.
%Each row of N is a node given by Xcoord, Ycoord, Zcoord, node cost,
%and node demand (-1 for source, +1 for sink). The function also
returns a
%vector containing only the demand for each node.

weights = bwdistsc(imcomplement(vol),[1,1,1]); %each voxel given a
price based on distance to the outside of the cell
%weights = weights / max(max(max(weights))); %weights are normalized
[attach_pos, attach_neg] = findAttachments2(cl_vol); %finds possible
fibril attachment sites to determine node demand

%offset grid so it is approximately centered on the entire volume
matrix
[a b c] = size(vol);
xoffset = mod(a,d)/2;
yoffset = mod(b,d)/2;
zoffset = mod(c,d)/2;

%generate a regular grid of nodes over the entire volume matrix
N1(1,1:3) = [0, b/2, c/2];
N1(1,4) = 0;
N1(1,5) = -s;
count = 2;
for i = max(floor(xoffset),1):d:a
    for j = max(floor(yoffset),1):d:b
        for k = max(floor(zoffset),1):d:c
            N1(count,1:3) = [i,j,k];
            N1(count,4) = weights(i,j,k);
            if attach_pos(i,j,k) == 1 %if a node falls in positive
attachment site, assign it as a sink
                N1(count,5) = 1;
            elseif attach_neg(i,j,k) == 1 %if a node falls in a
negative attachment site, assign it as a source
                N1(count,5) = -1;
            else
                N1(count,5) = 0;
            end
            count = count + 1;
        end
    end
end

%delete all nodes that do not fall within the volume
[s1 s2] = size(N1);
```

```

count = 2;
N(1,:) = N1(1,:);

for i=2:s1
    if vol(N1(i,1),N1(i,2),N1(i,3)) == 1
        %if N1(i,4) >= d/2 || N1(i,5) ~= 0
            N(count,1:5) = N1(i,1:5);
            count = count + 1;
        end
    end
end
[s1 s2] = size(N);

cent = [mean(N(:,1)), mean(N(:,2)), mean(N(:,3))];
N1(1,1:3) = [0, cent(2), cent(3)];
N(s1+1,1:3) = [a , cent(2), cent(3)];
N(s1+1,4) = 0;
N(s1+1,5) = s;

%generate demand vector
demand = N(:,5)';
demand(abs(demand)~=s) = 0;

```

A.10: findAttachments2.m

```
function [attach_pos, attach_neg] = findAttachments2(clustered_vol)
[a b c] = size(clustered_vol);
c1 = max(max(max(clustered_vol)));
attach_pos = zeros(size(clustered_vol));
attach_neg = zeros(size(clustered_vol));

for i = 2:a-1
    for j = 2:b-1
        for k = 2:c-1
            if clustered_vol(i,j,k) == c1 && nnz(clustered_vol(i-
1:i+1,j-1:j+1,k-1:k+1)) < 24
                attach_pos(i,j,k) = 1;
            elseif clustered_vol(i,j,k) == 1 && nnz(clustered_vol(i-
1:i+1,j-1:j+1,k-1:k+1)) < 24
                attach_neg(i,j,k) = 1;
            end
        end
    end
end
%attach_pos = bwareaopen(attach_pos,15);
%attach_neg = bwareaopen(attach_neg,15);
```


A.11: scaleDemand.m

```
function [scaled_demand] = scaleDemand(demand)
s1 = length(demand);
scaled_demand = zeros([1 2*s1]);
for i = 1:s1
    if demand(i) < 0
        scaled_demand(i) = demand(i);
    end
    if demand(i) > 0
        scaled_demand(i+s1) = demand(i);
    end
end
end
```

A.12: generateEdges3.m

```
function [E, c] = generateEdges3(N, sp)
[s1 s2] = size(N);
E = zeros(2*s1);
c = zeros(2*s1);
E(1, s1+1) = 999;
c(1, s1+1) = 0.01;
E(s1, s1+s1) = 999;
c(s1, s1+s1) = 0.01;

for i=2:s1-1
    E(i,i+s1) = 1;
    c(i,i+s1) = 0.01;
    if N(i,5) == -1
        E(1+s1,i) = 1;
        c(1+s1,i) = 100*(norm(N(i,1:3) - N(1,1:3)))^2;
    end
    if N(i,5) == 1
        E(i+s1,s1) = 1;
        c(i+s1,s1) = 100*(norm(N(s1,1:3)-N(i,1:3)))^2;
    end
end

    for j=2:s1-1
        if i ~= j
            d = sqrt((N(i,1)-N(j,1))^2 + (N(i,2)-N(j,2))^2 + (N(i,3)-
N(j,3))^2);
            if d <= sqrt(3*sp^2) && N(j,1) > N(i,1)
                %if N(j,1) > N(i,1)
                    E(i+s1,j) = 1;
                    c(i+s1,j) = d^2*(N(i,4) + N(j,4))/2;
                end
            end
        end
    end
end
end
```

A13: generateSideView.m

```
function [im_sv] = generateSideView(im)
%[im_rot, ang, CoM] = regImage(im);
%vol_rot = regImage2(vol, ang, CoM);
[a b c] = size(im);
for k = 1:c
    for i=1:b
        im_sv(k,:,i) = im(i,:,k)';
    end
end
```

A14: matrix2ft.m

```
function [from, to, costs, upper] = matrix2ft(E,c)
[a b]=size(E);
count = 1;
for i = 1:a
    for j = 1:b
        if E(i,j) ~= 0
            from(count) = i;
            to(count) = j;
            upper(count) = E(i,j);
            costs(count) = c(i,j);
            count = count+1;
        end
    end
end
```

APPENDIX B

MATLAB CODE FOR CELLSPARK

The original CellSpark application files and the MATLAB installer are available for download at: <http://github.com/tgharve/CellSpark>

B.1: CellSpark.m

```
function varargout = CellSpark(varargin)
global State currents
% CELLSPARK MATLAB code for CellSpark.fig
%   CELLSPARK, by itself, creates a new CELLSPARK or raises the
existing
%   singleton*.
%
%   H = CELLSPARK returns the handle to a new CELLSPARK or the
handle to
%   the existing singleton*.
%
%   CELLSPARK('CALLBACK',hObject,eventData,handles,...) calls the
local
%   function named CALLBACK in CELLSPARK.M with the given input
arguments.
%
%   CELLSPARK('Property','Value',...) creates a new CELLSPARK or
raises the
%   existing singleton*. Starting from the left, property value
pairs are
%   applied to the GUI before CellSpark_OpeningFcn gets called. An
%   unrecognized property name or invalid value makes property
application
%   stop. All inputs are passed to CellSpark_OpeningFcn via
varargin.
%
%   *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only
one
%   instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help CellSpark

% Last Modified by GUIDE v2.5 25-Sep-2018 09:32:57

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',      mfilename, ...
```

```

    'gui_Singleton', gui_Singleton, ...
    'gui_OpeningFcn', @CellSpark_OpeningFcn, ...
    'gui_OutputFcn', @CellSpark_OutputFcn, ...
    'gui_LayoutFcn', [] , ...
    'gui_Callback', []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if narginout
    [varargout{1:narginout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before CellSpark is made visible.
function CellSpark_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to CellSpark (see VARARGIN)
handles.count = 0;
handles.labels={};
%handles.P = plot(handles.axes1,0,0,'k');
xlim([0,600])
ylim([-100,80])
grid minor
datacursormode on
hold on
setappdata(0,'HT',0.02);
setappdata(0,'STOPTIME',600);
setappdata(0,'bcl',1000);
setappdata(0,'protocol','DYNREST');
pushbutton3_Callback(hObject,eventdata,handles);
% Choose default command line output for CellSpark
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes CellSpark wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = CellSpark_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

```

```

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes during object creation, after setting all properties.
function axes1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to axes1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: place code in OpeningFcn to populate axes1

% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
type =
find([get(handles.radiobutton1, 'Value'), get(handles.radiobutton2, 'Value'
'), get(handles.radiobutton3, 'Value'), get(handles.radiobutton7, 'Value')]
);
switch type
    case 1
        Args.type = 'EPI';
    case 2
        Args.type = 'ENDO';
    case 3
        Args.type = 'MCELL';
    case 4
        Args.type = 'NEURON';
end

Args.Ko = str2double(get(handles.edit1, 'String'));
Args.Cao = str2double(get(handles.edit2, 'String'));
Args.Nao = str2double(get(handles.edit3, 'String'));
Args.Tc = str2double(get(handles.edit4, 'String'));
Args.Ki = str2double(get(handles.edit5, 'String'));
Args.Cai = str2double(get(handles.edit6, 'String'));
Args.Nai = str2double(get(handles.edit7, 'String'));
Args.Cm = str2double(get(handles.edit8, 'String'));
Args.Vc = str2double(get(handles.edit13, 'String'));
Args.Vsr = str2double(get(handles.edit14, 'String'));
Args.amp = str2double(get(handles.edit9, 'String'));
Args.dur = str2double(get(handles.edit10, 'String'));
Args.tbegin = str2double(get(handles.edit11, 'String'));
Args.ow = get(handles.checkbox1, 'Value');
Args.HT = getappdata(0, 'HT');
Args.STOPTIME = getappdata(0, 'STOPTIME');
Args.bcl = getappdata(0, 'bcl');
Args.protocol = getappdata(0, 'protocol');
Args.GNa = str2double(get(handles.edit17, 'String'));
Args.GK = str2double(get(handles.edit16, 'String'));

```

```

Args.GL = str2double(get(handles.edit15, 'String'));

xlim([0,getappdata(0,'STOPTIME')]);
[hObject,handles] = run_simulation(Args,hObject,handles);
guidata(hObject,handles);

function edit11_Callback(hObject, eventdata, handles)
% hObject      handle to edit11 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit11 as text
%         str2double(get(hObject,'String')) returns contents of edit11
as a double
if isempty(str2num(get(hObject,'String')))
    set(hObject,'String','50');
end
set(hObject,'String',sprintf('%d',round(str2num(get(hObject,'String'))
));

% --- Executes during object creation, after setting all properties.
function edit11_CreateFcn(hObject, eventdata, handles)
% hObject      handle to edit11 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUiControlBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit10_Callback(hObject, eventdata, handles)
% hObject      handle to edit10 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit10 as text
%         str2double(get(hObject,'String')) returns contents of edit10
as a double
if isempty(str2num(get(hObject,'String')))
    set(hObject,'String','1');
end
set(hObject,'String',sprintf('%d',round(str2num(get(hObject,'String'))
));

% --- Executes during object creation, after setting all properties.
function edit10_CreateFcn(hObject, eventdata, handles)
% hObject      handle to edit10 (see GCBO)

```

```

% eventdata reserved - to be defined in a future version of MATLAB
% handles empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
% See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit9_Callback(hObject, eventdata, handles)
% hObject handle to edit9 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit9 as text
% str2double(get(hObject,'String')) returns contents of edit9 as
a double
if isempty(str2num(get(hObject,'String')))
    set(hObject,'String','52');
end
set(hObject,'String',sprintf('%0.1f',str2num(get(hObject,'String'))));

% --- Executes during object creation, after setting all properties.
function edit9_CreateFcn(hObject, eventdata, handles)
% hObject handle to edit9 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
% See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit8_Callback(hObject, eventdata, handles)
% hObject handle to edit8 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit8 as text
% str2double(get(hObject,'String')) returns contents of edit8 as
a double
if isempty(str2num(get(hObject,'String')))
    set(hObject,'String','0.185');
end
set(hObject,'String',sprintf('%0.3f',str2num(get(hObject,'String'))));

% --- Executes during object creation, after setting all properties.

```

```

function edit8_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUiControlBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit7_Callback(hObject, eventdata, handles)
% hObject    handle to edit7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit7 as text
%         str2double(get(hObject,'String')) returns contents of edit7 as
a double
if isempty(str2num(get(hObject,'String')))
    set(hObject,'String','11.6');
end
set(hObject,'String',sprintf('%0.1f',str2num(get(hObject,'String'))));

% --- Executes during object creation, after setting all properties.
function edit7_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUiControlBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit6_Callback(hObject, eventdata, handles)
% hObject    handle to edit6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit6 as text
%         str2double(get(hObject,'String')) returns contents of edit6 as
a double
if isempty(str2num(get(hObject,'String')))
    set(hObject,'String','0.2');
end
set(hObject,'String',sprintf('%0.5f',str2num(get(hObject,'String'))));

```



```

% --- Executes during object creation, after setting all properties.
function edit6_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit5_Callback(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit5 as text
%         str2double(get(hObject,'String')) returns contents of edit5 as
a double
if isempty(str2num(get(hObject,'String')))
    set(hObject,'String','138.8');
end
set(hObject,'String',sprintf('%0.1f',str2num(get(hObject,'String'))));

% --- Executes during object creation, after setting all properties.
function edit5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit13_Callback(hObject, eventdata, handles)
% hObject    handle to edit13 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit13 as text
%         str2double(get(hObject,'String')) returns contents of edit13
as a double
if isempty(str2num(get(hObject,'String')))
    set(hObject,'String','0.0164');

```

```

end
set(hObject, 'String', sprintf('%0.4f', str2num(get(hObject, 'String'))));

% --- Executes during object creation, after setting all properties.
function edit13_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit13 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject, 'BackgroundColor'),
get(0, 'defaultUiControlBackgroundColor'))
    set(hObject, 'BackgroundColor', 'white');
end

function edit14_Callback(hObject, eventdata, handles)
% hObject    handle to edit14 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject, 'String') returns contents of edit14 as text
%         str2double(get(hObject, 'String')) returns contents of edit14
as a double
if isempty(str2num(get(hObject, 'String')))
    set(hObject, 'String', '0.0011');
end
set(hObject, 'String', sprintf('%0.4f', str2num(get(hObject, 'String'))));

% --- Executes during object creation, after setting all properties.
function edit14_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit14 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject, 'BackgroundColor'),
get(0, 'defaultUiControlBackgroundColor'))
    set(hObject, 'BackgroundColor', 'white');
end

function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject, 'String') returns contents of edit1 as text
%         str2double(get(hObject, 'String')) returns contents of edit1 as
a double

```

```

if isempty(str2num(get(hObject,'String')))
    set(hObject,'String','5.4');
end
set(hObject,'String',sprintf('%0.1f',str2num(get(hObject,'String'))));

% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUiControlBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
%       str2double(get(hObject,'String')) returns contents of edit2 as
a double
if isempty(str2num(get(hObject,'String')))
    set(hObject,'String','2.0');
end
set(hObject,'String',sprintf('%0.1f',str2num(get(hObject,'String'))));

% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUiControlBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

```

```

% Hints: get(hObject,'String') returns contents of edit3 as text
%         str2double(get(hObject,'String')) returns contents of edit3 as
a double
if isempty(str2num(get(hObject,'String')))
    set(hObject,'String','140.0');
end
set(hObject,'String',sprintf('%0.1f',str2num(get(hObject,'String'))));

% --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUiControlBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit4_Callback(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit4 as text
%         str2double(get(hObject,'String')) returns contents of edit4 as
a double
if isempty(str2num(get(hObject,'String')))
    set(hObject,'String','37.0');
end
set(hObject,'String',sprintf('%0.1f',str2num(get(hObject,'String'))));

% --- Executes during object creation, after setting all properties.
function edit4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUiControlBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in radiobutton1.
function radiobutton1_Callback(hObject, eventdata, handles)
% hObject    handle to radiobutton1 (see GCBO)

```

```

% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of radiobutton1
set(handles.edit2,'String','2.0');
set(handles.text17,'String','CaSR');
set(handles.edit6,'String','0.2');
set(handles.text32,'Visible','on');
set(handles.edit14,'Visible','on');
set(handles.text31,'Visible','on');
set(handles.text29,'Visible','on');
set(handles.text30,'Visible','on');
set(handles.edit13,'Visible','on');
set(handles.edit4,'String','37.0');
set(handles.text23,'String','mA');
set(handles.uibuttongroup4,'Visible','off');
setappdata(0,'STOPTIME',600);
setappdata(0,'HT',.02);
s={'Nao','Ko','Cao','T','Nai','Ki','CaSR','Cm','Vc','Vsr','Amplitude','
Duration','Start Time'};
set(handles.popupmenu2,'String',s);
s = {'Voltage (mV)','Cai (mM)','INa (mA)','ICaL (mA)','Ito (mA)','IKs
(mA)',...
'IKr (mA)','IK1 (mA)','INaCa (mA)','INaK (mA)','IbNa (mA)','IbCa
(mA)','Irel (mA)'};
set(handles.popupmenu3,'String',s);
pushbutton2_Callback(hObject,eventdata,handles);
pushbutton3_Callback(hObject,eventdata, handles);

% --- Executes on button press in radiobutton2.
function radiobutton2_Callback(hObject, eventdata, handles)
% hObject handle to radiobutton2 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of radiobutton2
set(handles.edit2,'String','2.0');
set(handles.text17,'String','CaSR');
set(handles.edit6,'String','0.2');
set(handles.text32,'Visible','on');
set(handles.edit14,'Visible','on');
set(handles.text31,'Visible','on');
set(handles.uibuttongroup4,'Visible','off');
set(handles.text29,'Visible','on');
set(handles.text30,'Visible','on');
set(handles.edit13,'Visible','on');
set(handles.edit4,'String','37.0');
set(handles.text23,'String','mA');

setappdata(0,'STOPTIME',600);
setappdata(0,'HT',.02);
s={'Nao','Ko','Cao','T','Nai','Ki','CaSR','Cm','Vc','Vsr','Amplitude','
Duration','Start Time'};

```

```

set(handles.popupmenu2,'String',s);
s = {'Voltage (mV)', 'Cai (mM)', 'INa (mA)', 'ICaL (mA)', 'Ito (mA)', 'IKs
(mA)', ...
     'IKr (mA)', 'IK1 (mA)', 'INaCa (mA)', 'INaK (mA)', 'IbNa (mA)', 'IbCa
(mA)', 'Irel (mA)'};
set(handles.popupmenu3,'String',s);
pushbutton2_Callback(hObject,eventdata,handles);
pushbutton3_Callback(hObject,eventdata, handles);

% --- Executes on button press in radiobutton3.
function radiobutton3_Callback(hObject, eventdata, handles)
% hObject    handle to radiobutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of radiobutton3
set(handles.edit2,'String','2.0');
set(handles.text17,'String','CaSR');
set(handles.edit6,'String','0.2');
set(handles.text32,'Visible','on');
set(handles.edit14,'Visible','on');
set(handles.text31,'Visible','on');
set(handles.uibuttongroup4,'Visible','off');
set(handles.text29,'Visible','on');
set(handles.text30,'Visible','on');
set(handles.edit13,'Visible','on');
set(handles.edit4,'String','37.0');
set(handles.text23,'String','mA');

setappdata(0,'STOPTIME',600);
setappdata(0,'HT',.02);
s={'Nao','Ko','Cao','T','Nai','Ki','CaSR','Cm','Vc','Vsr','Amplitude','
Duration','Start Time'};
set(handles.popupmenu2,'String',s);
s = {'Voltage (mV)', 'Cai (mM)', 'INa (mA)', 'ICaL (mA)', 'Ito (mA)', 'IKs
(mA)', ...
     'IKr (mA)', 'IK1 (mA)', 'INaCa (mA)', 'INaK (mA)', 'IbNa (mA)', 'IbCa
(mA)', 'Irel (mA)'};
set(handles.popupmenu3,'String',s);
pushbutton2_Callback(hObject,eventdata,handles);
pushbutton3_Callback(hObject,eventdata, handles);

% --- Executes during object creation, after setting all properties.
function uibuttongroup2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to uibuttongroup2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% --- Executes on button press in checkbox1.
function checkbox1_Callback(hObject, eventdata, handles)
% hObject    handle to checkbox1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB

```

```

% handles      structure with handles and user data (see GUIDATA)
handles.count = 1;

% Hint: get(hObject,'Value') returns toggle state of checkbox1

% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject      handle to pushbutton2 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)
cla(handles.axes1);
handles.labels = {};
handles.count = 0;
legend(handles.axes1, 'off');
guidata(hObject,handles);

% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject      handle to pushbutton3 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)
set(handles.edit4, 'String', '37.0');
if get(handles.radiobutton7, 'Value') == 1
    set(handles.edit1, 'String', '4.0');
    set(handles.edit2, 'String', '41.0');
    set(handles.edit3, 'String', '142.0');
    set(handles.edit5, 'String', '120.0');
    set(handles.edit7, 'String', '10.0');
    set(handles.edit6, 'String', '0.00011');
    set(handles.edit11, 'String', '1.0');
    set(handles.edit9, 'String', '15');
    set(handles.edit8, 'String', '1.0');
    set(handles.edit17, 'String', '120');
    set(handles.edit16, 'String', '36');
    set(handles.edit15, 'String', '0.3')
else
    set(handles.edit1, 'String', '5.4');
    set(handles.edit2, 'String', '2.0');
    set(handles.edit3, 'String', '140.0');
    set(handles.edit4, 'String', '37.0');
    set(handles.edit5, 'String', '138.8');
    set(handles.edit6, 'String', '0.2');
    set(handles.edit7, 'String', '11.6');
    set(handles.edit8, 'String', '0.185');
    set(handles.edit13, 'String', '0.0164');
    set(handles.edit14, 'String', '0.0011');
    set(handles.edit9, 'String', '52');
    set(handles.edit10, 'String', '1');
    set(handles.edit11, 'String', '50');
end

% --- Executes on selection change in popupmenu1.

```

```

function popupmenu1_Callback(hObject, eventdata, handles)
% hObject      handle to popupmenu1 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns popupmenu1
contents as cell array
%           contents{get(hObject,'Value')} returns selected item from
popupmenu1

% --- Executes during object creation, after setting all properties.
function popupmenu1_CreateFcn(hObject, eventdata, handles)
% hObject      handle to popupmenu1 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns
called

% Hint: popupmenu controls usually have a white background on Windows.
%           See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
set(hObject,'String',['Legend
Parameter';'Ko';'Cao';'Nao';'T';'Ki';'CaSR';'Nai';'Cm';'Vc';'Vsr';'Ampl
itude';'Duration';'Start Time']);
guidata(hObject,handles);

% --- Executes on selection change in popupmenu2.
function popupmenu2_Callback(hObject, eventdata, handles)
% hObject      handle to popupmenu2 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)
cla(handles.axes1);
handles.labels = {};
handles.count = 0;
legend(handles.axes1,'off');
guidata(hObject,handles);

% Hints: contents = cellstr(get(hObject,'String')) returns popupmenu2
contents as cell array
%           contents{get(hObject,'Value')} returns selected item from
popupmenu2

% --- Executes during object creation, after setting all properties.
function popupmenu2_CreateFcn(hObject, eventdata, handles)
% hObject      handle to popupmenu2 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns
called

% Hint: popupmenu controls usually have a white background on Windows.

```



```

%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in popupmenu3.
function popupmenu3_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
cla(handles.axes1);
handles.labels = {};
handles.count = 0;
legend(handles.axes1,'off');
guidata(hObject,handles);
% Hints: contents = cellstr(get(hObject,'String')) returns popupmenu3
contents as cell array
%       contents{get(hObject,'Value')} returns selected item from
popupmenu3

% --- Executes during object creation, after setting all properties.
function popupmenu3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
advsettings();

% --- Executes on button press in radiobutton4.
function radiobutton4_Callback(hObject, eventdata, handles)
% hObject    handle to radiobutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of radiobutton4

zoom off
pan off

```

```

datacursormode on

% --- Executes on button press in radiobutton5.
function radiobutton5_Callback(hObject, eventdata, handles)
% hObject    handle to radiobutton5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of radiobutton5

pan off
datacursormode off
zoom on

% --- Executes on button press in radiobutton6.
function radiobutton6_Callback(hObject, eventdata, handles)
% hObject    handle to radiobutton6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of radiobutton6

datacursormode off
zoom off
pan on

% --- Executes on button press in pushbutton5.
function pushbutton5_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%reset(handles.axes1);
xlim(handles.axes1,[0,getappdata(0,'STOPTIME')])
ylim(handles.axes1,'auto')
%grid minor
datacursormode on
set(handles.radiobutton4,'Value',1);
set(handles.radiobutton5,'Value',0);
set(handles.radiobutton6,'Value',0);
xlim([0,getappdata(0,'STOPTIME')]);

% --- Executes on button press in radiobutton7.
function radiobutton7_Callback(hObject, eventdata, handles)
% hObject    handle to radiobutton7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of radiobutton7
set(handles.edit1,'String','20.11');
set(handles.edit2,'String','44.0');
set(handles.edit3,'String','491.0');

```

```

set(handles.edit5,'String','400.0');
set(handles.edit7,'String','50,0');
set(handles.text17,'String','Cai');
set(handles.edit6,'String','0.1');
set(handles.text32,'Visible','off');
set(handles.edit14,'Visible','off');
set(handles.text31,'Visible','off');
set(handles.text29,'Visible','off');
set(handles.text30,'Visible','off');
set(handles.edit13,'Visible','off');
set(handles.edit9,'String','15');
set(handles.text23,'String','uA');
set(handles.uibuttongroup4,'Visible','on');
set(handles.edit4,'String','37.0');
setappdata(0,'STOPTIME',20);
setappdata(0,'HT',.001);
pushbutton2_Callback(hObject,eventdata,handles);
pushbutton3_Callback(hObject,eventdata,handles);
s={'Nao','Ko','Cao','T','Nai','Ki','Cai','Cm','Amplitude','Duration','S
tart Time','GNa','GK','GL'};
set(handles.popupmenu2,'String',s);

s = {'Voltage (mV)','INa (mA)','IK (mA)','Ileak (mA)','m','h','n'};
set(handles.popupmenu3,'String',s);

function edit15_Callback(hObject, eventdata, handles)
% hObject    handle to edit15 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit15 as text
%        str2double(get(hObject,'String')) returns contents of edit15
as a double

% --- Executes during object creation, after setting all properties.
function edit15_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit15 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%        See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUiControlBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit16_Callback(hObject, eventdata, handles)
% hObject    handle to edit16 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

```

```

% Hints: get(hObject,'String') returns contents of edit16 as text
%         str2double(get(hObject,'String')) returns contents of edit16
as a double

% --- Executes during object creation, after setting all properties.
function edit16_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit16 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit17_Callback(hObject, eventdata, handles)
% hObject    handle to edit17 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit17 as text
%         str2double(get(hObject,'String')) returns contents of edit17
as a double

% --- Executes during object creation, after setting all properties.
function edit17_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit17 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

```

B.2: settings.m

```
function varargout = settings(varargin)
% settings MATLAB code for settings.fig
% settings, by itself, creates a new settings or raises the
existing
% singleton*.
%
% H = settings returns the handle to a new settings or the handle
to
% the existing singleton*.
%
% settings('CALLBACK',hObject,eventData,handles,...) calls the
local
% function named CALLBACK in settings.M with the given input
arguments.
%
% settings('Property','Value',...) creates a new settings or
raises the
% existing singleton*. Starting from the left, property value
pairs are
% applied to the GUI before settings_OpeningFcn gets called. An
% unrecognized property name or invalid value makes property
application
% stop. All inputs are passed to settings_OpeningFcn via
varargin.
%
% *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only
one
% instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help settings

% Last Modified by GUIDE v2.5 23-Aug-2017 10:30:54

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                  'gui_Singleton',  gui_Singleton, ...
                  'gui_OpeningFcn', @settings_OpeningFcn, ...
                  'gui_OutputFcn',  @settings_OutputFcn, ...
                  'gui_LayoutFcn',  [] , ...
                  'gui_Callback',    []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
```

```

end
% End initialization code - DO NOT EDIT

% --- Executes just before settings is made visible.
function settings_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles     structure with handles and user data (see GUIDATA)
% varargin   command line arguments to settings (see VARARGIN)

% Choose default command line output for settings
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes settings wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = settings_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles     structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles     structure with handles and user data (see GUIDATA)
setappdata(0, 'HT', 0.2);
setappdata(0, 'STOPTIME', 600);
setappdata(0, 'BCL', 1000);
setappdata(0, 'protocol', 'DYNREST');
set(handles.edit1, 'String', '0.02');
set(handles.edit2, 'String', '600');
set(handles.edit3, 'String', '1000');
set(handles.radiobutton1, 'Value', 1);

% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)

```

```

% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
setappdata(0, 'HT', str2num(get(handles.edit1, 'String')));
setappdata(0, 'STOPTIME', str2num(get(handles.edit2, 'String')));
setappdata(0, 'bcl', str2num(get(handles.edit3, 'String')));
if (get(handles.radiobutton1, 'Value'))
    setappdata(0, 'protocol', 'DYNREST');
else
    setappdata(0, 'protocol', 'S1S2REST');
end
delete(handles.figure1);

function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject, 'String') returns contents of edit1 as text
%        str2double(get(hObject, 'String')) returns contents of edit1 as
a double

% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%        See ISPC and COMPUTER.
set(hObject, 'String', getappdata(0, 'HT'));
if ispc && isequal(get(hObject, 'BackgroundColor'),
get(0, 'defaultUiControlBackgroundColor'))
    set(hObject, 'BackgroundColor', 'white');
end

function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject, 'String') returns contents of edit2 as text
%        str2double(get(hObject, 'String')) returns contents of edit2 as
a double

```

```

% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
set(hObject, 'String', getappdata(0, 'STOPTIME'));
if ispc && isequal(get(hObject, 'BackgroundColor'),
get(0, 'defaultUicontrolBackgroundColor'))
    set(hObject, 'BackgroundColor', 'white');
end

function edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject, 'String') returns contents of edit3 as text
%         str2double(get(hObject, 'String')) returns contents of edit3 as
a double

% --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
set(hObject, 'String', getappdata(0, 'bcl'));
if ispc && isequal(get(hObject, 'BackgroundColor'),
get(0, 'defaultUicontrolBackgroundColor'))
    set(hObject, 'BackgroundColor', 'white');
end

% --- Executes during object creation, after setting all properties.
function uibuttongroup1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to uibuttongroup1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

```



```

% --- Executes during object creation, after setting all properties.
function radiobutton1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to radiobutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called
protocol = getappdata(0, 'protocol');
if isequal(protocol, 'DYNREST')
    set(hObject, 'Value', 1);
else
    set(hObject, 'Value', 0);
end

```

```

% --- Executes during object creation, after setting all properties.
function radiobutton2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to radiobutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called
protocol = getappdata(0, 'protocol');
if isequal(protocol, 'S1S2REST')
    set(hObject, 'Value', 1);
else
    set(hObject, 'Value', 0);
end

```

B.3: run_simulation.m

```
function [hObject_new, handles_new] =
run_simulation(Args,hObject,handles)

%clc
%close all

%External concentrations
global Ko Cao Nao Vc Vsr Bufc Kbufc Bufsr Kbufsr taufca taug Vmaxup Kup
R F T RTONF CAPACITANCE ...
    Gkr pKNa Gto GKs GKl GNa GbNa KmK KmNa knak GCaL GbCa knaca KmNai
KmCa ksat n GpCa KpCa GpK ...
    i_low i_high j_low j_high stimduration stimstrength period currents
kT Tc type protocol State GK GL VL

handles.count = handles.count + 1;
ow = Args.ow;
type = Args.type; %EPI, ENDO, MCELL, or NEURON

%Ko=5.4;
%Cao=2.0;
%Nao=140.0;
if isequal(type,'NEURON')
    Ko = Args.Ko;
    Cao = Args.Cao;
    Nao = Args.Nao;

    Vc = Args.Vc;

    %Constants
    R=8314.472;
    F=96485.3415;
    %Tc = 37;
    Tc = Args.Tc;
    T=Tc+273.0;
    kT = 3^((Tc-37.0)/10);
    RTONF=(R*T)/F;

    %Conductances
    %GNa = 120;
    %GK = 36;
    %GL = 0.3;
    VL = -49;

    GNa = Args.GNa;
    GK = Args.GK;
    GL = Args.GL;
```

```

CAPACITANCE = Args.Cm;

HT = Args.HT;

%Initial values of state variables

Cai_init=Args.Cai;
Nai_init=Args.Nai;
Ki_init=Args.Ki;

%V_init=(5/115)*RTONF*(log((Nao/Nai_init))) + (100/115)*
RTONF*(log((Ko/Ki_init))) + (10/115)*VL;
V_init = -62;

%duration of the simulation
STOPTIME = Args.STOPTIME;

stimduration=Args.dur;
stimstrength=-10* Args.amp;
tbegin=Args.tbegin;
tend=tbegin+stimduration;

time = 0;
step = 0;
Istim = 0;
Var = VariablesN(V_init, Cai_init,Nai_init,Ki_init);
State =
[0,Var.Volt,Var.Volt2,Var.Cai,Var.Nai,Var.Ki,Var.M,Var.H,Var.N,Var.Itot
];
currents = [0 0 0 0];

leg = get(handles.popupmenu2,'Value');

switch leg
case 1
    label = [get(handles.edit3,'String'), ' mM'];
case 2
    label = [get(handles.edit1,'String'), ' mM'];
case 3
    label = [get(handles.edit2,'String'), ' mM'];
case 4
    label = [get(handles.edit4,'String') ' ' char(176) 'C'];
case 5
    label = [get(handles.edit7,'String'), ' mM'];
case 6
    label = [get(handles.edit5,'String'), ' mM'];
case 7
    label = [get(handles.edit6,'String'), ' mM'];
case 8
    label = [get(handles.edit8,'String'), ' \muF/cm^{2}'];
case 9

```

```

        label = [get(handles.edit9,'String'), ' mA'];
    case 10
        label = [get(handles.edit10,'String'), ' ms'];
    case 11
        label = [get(handles.edit11,'String'), ' ms'];
    case 12
        label = [get(handles.edit17,'String'), ' mS/cm^{2}'];
    case 13
        label = [get(handles.edit16,'String'), ' mS/cm^{2}'];
    case 14
        label = [get(handles.edit15,'String'), ' mS/cm^{2}'];
end

if ow == 1
    genvarname('handles.P',num2str(handles.count));
    eval(['handles.P' num2str(handles.count)
    '=plot(handles.axes1,0,0);']);
    handles.labels{handles.count,1} = label;
    legend(handles.labels);
else
    cla(handles.axes1);
    clear handles.labels;
    handles.labels = {};
    handles.P=plot(handles.axes1,0,0);
    handles.count = 1;
    handles.labels{handles.count,1} = label;
    legend(handles.labels);
end

while time<=STOPTIME
    time = time+HT;
    if(time>=tbegin && time<=tend)

        Istim=stimstrength;
    end

    if(time>tend)
        Istim=0.;
    end

    Var = StepN(Var,HT,time,step,Istim);
    if(mod(step,10)==0)
        State = [State; time,
Var.Volt,Var.Volt2,Var.Cai,Var.Nai,Var.Ki,Var.M,Var.H,Var.N,Var.Itot];
        if(mod(step,250)==0)

            xvals = State(:,1);
            yvar = get(handles.popupmenu3,'Value');
            switch yvar
                case 1
                    yvals = State(:,2); %Voltage

```

```

        case 2
            yvals = currents(:,2); %INa
        case 3
            yvals = currents(:,3); %IK
        case 4
            yvals = currents(:,4); %IL
        case 5
            yvals = State(:,7); %M
        case 6
            yvals = State(:,8); %H
        case 7
            yvals = State(:,9); %N
        end

        if(ow == 0)
            set(handles.P, 'xdata', xvals, 'ydata', yvals);

            else
                eval(['set(handles.P'
                    num2str(handles.count) ', 'xdata', xvals, 'ydata', yvals);']);
                end
            ylim(handles.axes1, 'auto');
            drawnow('update');
            guidata(hObject, handles);
        end
    end

    step = step+1;
end
handles.count = handles.count + 1;
handles.State = State;

%ylim(handles.axes1, 'auto');

hObject_new = hObject;
handles_new = handles;

else
    protocol = Args.protocol; %DYNREST or S1S2REST

    Ko = Args.Ko;
    Cao = Args.Cao;
    Nao = Args.Nao;

    %Intracellular volumes

    %Vc=0.016404;
    %Vsr=0.001094;

    Vc = Args.Vc;

```

```

Vsr = Args.Vsr;

%Calcium dynamics
Bufc=0.15;
Kbufc=0.001;
Bufsr=10.;
Kbufsr=0.3;
taufca=kT*2.;
taug=kT*2.;
Vmaxup=0.000425;
Kup=0.00025;

%Constants
R=8314.472;
F=96485.3415;
%Tc = 37;
Tc = Args.Tc;
T=Tc+273.0;
kT = 3^((T-310)/10);
RTONF=(R*T)/F;

%Cellular capacitance
%CAPACITANCE=0.185;
CAPACITANCE = Args.Cm;

%Parameters for currents
%Parameters for IKr
Gkr=0.096;
%Parameters for Iks
pKNa=0.03;

%cell type dependent parameters for Iks and Ito
switch type
case 'EPI'
    Gto = 0.294;
    GKs = 0.245;
case 'MCELL'
    Gto = 0.294;
    GKs = 0.062;
case 'ENDO'
    Gto = 0.073;
    GKs = 0.245;
end

%Parameters for Ik1
GK1=5.405;

%Parameters for INa
GNa=14.838;
%Parameters for IbNa
GbNa=0.00029;
%Parameters for INaK

```

```

KmK=1.0;
KmNa=40.0;
knak=1.362;
%Parameters for ICaL
GCaL=0.000175;
%Parameters for IbCa
GbCa=0.000592;
%Parameters for INaCa
knaca=1000;
KmNai=87.5;
KmCa=1.38;
ksat=0.1;
n=0.35;
%Parameters for IpCa
GpCa=0.825;
KpCa=0.0005;
%Parameters for IpK;
GpK=0.0146;

%timestep (ms)
%HT =0.02;
HT = Args.HT;

%Initial values of state variables

CaSR_init=Args.Cai;
Nai_init=Args.Nai;
Ki_init=Args.Ki;

Cai_init=0.0002;
%CaSR_init=0.2;
%Nai_init=11.6;
%Ki_init=138.3;
V_init=RTONF*(log((Ko/Ki_init)));

%duration of the simulation
%STOPTIME=600;
STOPTIME = Args.STOPTIME;

switch protocol
case 'DYNREST'
    i_low=0;
    i_high=1;
    j_low=0;
    j_high=1;
    stimduration = Args.dur;
    %stimduration=1.0;
    stimstrength = -1 * Args.amp;
    %stimstrength=-52;

```

```

        %period=1000;
        period = Args.bcl;
        sum=period*1000.;
        tbegin = Args.tbegin;
        %tbegin=50;
        tend=tbegin+stimduration;
    case 'S1S2REST'
        i_low=0;
        i_high=1;
        j_low=0;
        j_high=1;
        stimduration=Args.dur;
        stimstrength=-1* Args.amp;
        tbegin=Args.tbegin;
        tend=tbegin+stimduration;
        counter=1;
        dia=5000;
        %basicperiod=1000.;
        basicperiod = Args.bcl;
        basicapd=274;
        repeats=10;
    end

    time = 0;
    step = 0;
    Istim = 0;
    Var = Variables(V_init, Cai_init,
CaSR_init,Nai_init,Ki_init);
    State =
[0,Var.Volt,Var.Volt2,Var.Cai,Var.CaSR,Var.Nai,Var.Ki,Var.M,Var.H,Var.J
,Var.Xr1,Var.Xr2,Var.Xs,Var.S,Var.R,Var.D,Var.F,Var.FCa,Var.G,Var.Itot]
;
    currents = [0 0 0 0 0 0 0 0 0 0 0 0];

    leg = get(handles.popupmenu2, 'Value');

    switch leg
        case 1
            label = [get(handles.edit3, 'String'), ' mM'];
        case 2
            label = [get(handles.edit1, 'String'), ' mM'];
        case 3
            label = [get(handles.edit2, 'String'), ' mM'];
        case 4
            label = [get(handles.edit4, 'String') ' ' char(176)
'C'];
        case 5
            label = [get(handles.edit7, 'String'), ' mM'];
        case 6
            label = [get(handles.edit5, 'String'), ' mM'];
        case 7
            label = [get(handles.edit6, 'String'), ' mM'];
    end

```



```

        case 8
            label = [get(handles.edit8,'String'), '
\mu F/cm^{2}'];
        case 9
            label = [get(handles.edit13,'String'), '
\mum^{3}'];
        case 10
            label = [get(handles.edit14,'String'), 'mum^{3}'];
        case 11
            label = [get(handles.edit9,'String'), ' mA'];
        case 12
            label = [get(handles.edit10,'String'), ' ms'];
        case 13
            label = [get(handles.edit11,'String'), ' ms'];
        case 14
            label = Args.type;
    end

    if ow == 1
        genvarname('handles.P', num2str(handles.count));
        eval(['handles.P' num2str(handles.count)
' =plot(handles.axes1,0,0);']);
        handles.labels{handles.count,1} = label;
        legend(handles.labels);
    else
        cla(handles.axes1);
        clear handles.labels;
        handles.labels = {};
        handles.P=plot(handles.axes1,0,0);
        handles.count = 1;
        handles.labels{handles.count,1} = label;
        legend(handles.labels);
    end

    while time<=STOPTIME
        time = time+HT;
        switch protocol
            case 'DYNREST'
                if(time>sum)
                    if (period>4000)
                        period=period-1000;
                        sum=sum+period*30;
                    elseif (period>3000)
                        period=period-1000;
                        sum=sum+period*30;
                    elseif (period>2000)
                        period=period-1000;
                        sum=sum+period*30;
                    elseif (period>1000)
                        period=period-1000;
                        sum=sum+period*100;
                    elseif (period>500)
                        period=period-250;
                end
            end
        end
    end

```

```

        sum=sum+period*100;
elseif(period>400)
    period=period-50;
    sum=sum+period*100;
elseif(period>300)
    period=period-10;
    sum=sum+period*100;
elseif(period>250)
    period=period-5;
    sum=sum+period*100;
elseif(period>50)
    period=period-1;
    sum=sum+period*100;
else
    %disp('Restitution protocol complete')
end
end
if(time>=tbegin && time<=tend)

    Istim=stimstrength;
end

if(time>tend)

    Istim=0.;
    tbegin=tbegin+period;
    tend=tbegin+stimduration;
end

case 'S1S2REST'
if(counter<repeats)
    if(time>=tbegin && time<=tend)
        Istim=stimstrength;
    end
    if(time>tend)
        Istim=0.;
        tbegin=tbegin+basicperiod;
        tend=tbegin+stimduration;
        counter=counter+1;

    elseif(counter==repeats)
        if(time>=tbegin && time<=tend)
            Istim=stimstrength;
        end
        if(time>tend)
            Istim=0.;
            tbegin=tbegin+basicapd+dia;
            tbeginS2=tbegin;
            tend=tbegin+stimduration;
            counter=counter+1;
        elseif(counter==repeats+1)

```



```

        case 10
            yvals = currents(:,8); %INaK
        case 11
            yvals = currents(:,7); %IbNa
        case 12
            yvals = currents(:,10); %IbCa
        case 13
            yvals = currents(:,12); %Irel
        end

        if(ow == 0)
            set(handles.P, 'xdata', xvals, 'ydata', yvals);

            else
                eval(['set(handles.P'
                    num2str(handles.count) ', 'xdata', ', xvals, 'ydata', ', yvals);']);
            end
            ylim(handles.axes1, 'auto');
            drawnow('update');
            guidata(hObject, handles);
        end
    end

    step = step+1;
end
handles.count = handles.count + 1;
handles.State = State;

%ylim(handles.axes1, 'auto');

hObject_new = hObject;
handles_new = handles;
end
end

```

B.4: Step.m

```
function [Vs] = Step(V,HT,tt,step,Istim)
global Ko Cao Nao Vc Vsr Bufc Kbufc Bufsr Kbufsr taufca taug Vmaxup Kup
R F T RTONF CAPACITANCE ...
    Gkr pKNa type Gto GKs GK1 GNa GbNa KmK KmNa knak GCaL GbCa knaca
KmNai KmCa ksat n GpCa KpCa GpK ...
    currents kT

inverseVcF2=1/(2*Vc*F);
inverseVcF=1./(Vc*F);
Kupsquare=Kup*Kup;
BufcKbufc=Bufc*Kbufc;
Kbufcsquare=Kbufc*Kbufc;
Kbufc2=2*Kbufc;
BufsrKbufsr=Bufsr*Kbufsr;
Kbufsrsquare=Kbufsr*Kbufsr;
Kbufsr2=2*Kbufsr;
exptaufca=exp(-HT./taufca);
exptaug=exp(-HT./taug);

sm = V.M;
sh = V.H;
sj = V.J;
sxr1 = V.Xr1;
sxr2 = V.Xr2;
sxs = V.Xs;
ss = V.S;
sr = V.R;
sd = V.D;
sf = V.F;
sfca = V.FCa;
sg = V.G;
svolt = V.Volt;
svolt2 = V.Volt2;
Cai = V.Cai;
CaSR = V.CaSR;
Nai = V.Nai;
Ki = V.Ki;
sItot = V.Itot;

%Needed to compute currents
Ek=RTONF*(log((Ko/Ki)));
Ena=RTONF*(log((Nao/Nai)));
Eks=RTONF*(log((Ko+pKNa*Nao)/(Ki+pKNa*Nai)));
Eca=0.5*RTONF*(log((Cao/Cai)));
Ak1=0.1/(1.+exp(0.06*(svolt-Ek-200)));
Bk1=(3.*exp(0.0002*(svolt-Ek+100))+exp(0.1*(svolt-Ek-10)))/(1.+exp(-
0.5*(svolt-Ek)));
rec_iK1=Ak1/(Ak1+Bk1);
```

```

rec_iNaK=(1./(1.+0.1245*exp(-0.1*svolt*F/(R*T))+0.0353*exp(-
svolt*F/(R*T))));
rec_ipK=1./(1.+exp((25-svolt)/5.98));

%Compute currents
INa=GNa*sm*sm*sm*sh*sj*(svolt-Ena);
ICaL=GCaL*sd*sf*sfca*4*svolt*(F*F/(R*T))*(exp(2*svolt*F/(R*T))*Cai-
0.341*Cao)/(exp(2*svolt*F/(R*T))-1.);
Ito=Gto*sr*ss*(svolt-Ek);
IKr=Gkr*sqrt(Ko/5.4)*sxr1*sxr2*(svolt-Ek);
IKs=Gks*sxs*sxs*(svolt-Eks);
IKl=GKl*rec_iKl*(svolt-Ek);
INaCa=knaca*(1./(KmNai*KmNai*KmNai+Nao*Nao*Nao))*(1./(KmCa+Cao))*(1./(1
+ksat*exp((n-1)*svolt*F/(R*T))))*(exp(n*svolt*F/(R*T))*Nai*Nai*Nai*Cao-
exp((n-1)*svolt*F/(R*T))*Nao*Nao*Nao*Cai*2.5);
INaK=knak*(Ko/(Ko+KmK))*(Nai/(Nai+KmNa))*rec_iNaK;
IpCa=GpCa*Cai/(KpCa+Cai);
IpK=GpK*rec_ipK*(svolt-Ek);
IbNa=GbNa*(svolt-Ena);
IbCa=GbCa*(svolt-Eca);

%Determine total current
sItot = IKr+IKs+IKl+Ito+INa+IbNa+ICaL+IbCa+INaK+INaCa+IpCa+IpK+Istim;

%update concentrations
Caisquare=Cai*Cai;
CaSRsquare=CaSR*CaSR;
CaCurrent=- (ICaL+IbCa+IpCa-2*INaCa)*inverseVcF2*CAPACITANCE;
A=0.016464*CaSRsquare/(0.0625+CaSRsquare)+0.008232;
Irel=A*sd*sg;
Ileak=0.00008*(CaSR-Cai);
SERCA=Vmaxup/(1.+(Kupsquare/Caisquare));
CaSRCurrent=SERCA-Irel-Ileak;
CaCSQN=Bufsr*CaSR/(CaSR+Kbufsr);
dCaSR=HT*(Vc/Vsr)*CaSRCurrent;
bjrs=Bufsr-CaCSQN-dCaSR-CaSR+Kbufsr;
cjsr=Kbufsr*(CaCSQN+dCaSR+CaSR);
CaSR=(sqrt(bjrs*bjrs+4*cjsr)-bjrs)/2;
CaBuf=Bufc*Cai/(Cai+Kbufc);
dCai=HT*(CaCurrent-CaSRCurrent);
bc=Bufc-CaBuf-dCai-Cai+Kbufc;
cc=Kbufc*(CaBuf+dCai+Cai);
Cai=(sqrt(bc*bc+4*cc)-bc)/2;

dNai=- (INa+IbNa+3*INaK+3*INaCa)*inverseVcF*CAPACITANCE;
Nai=Nai+HT*dNai;

dKi=- (Istim+IKl+Ito+IKr+IKs-2*INaK+IpK)*inverseVcF*CAPACITANCE;
Ki=Ki+HT*dKi;

if mod(step,10)==0

```

```

    currents = [currents; tt, IKr, IKs, IKl, Ito, INa, IbNa, INaK,
ICaL, IbCa, INaCa, Irel];
end

%compute steady state values and time constants
AM=1./(1.+exp((-60.-svolt)/5.));
BM=0.1/(1.+exp((svolt+35.)/5.))+0.10/(1.+exp((svolt-50.)/200.));
TAU_M=(1/kT)*AM*BM;
M_INF=1./((1.+exp((-56.86-svolt)/9.03))*(1.+exp((-56.86-svolt)/9.03)));
if (svolt>=-40.)

    AH_1=0.;
    BH_1=(0.77/(0.13*(1.+exp(-(svolt+10.66)/11.1))));
    TAU_H= kT/((AH_1+BH_1));

else

    AH_2=(0.057*exp(-(svolt+80.)/6.8));
    BH_2=(2.7*exp(0.079*svolt)+(3.1e5)*exp(0.3485*svolt));
    TAU_H=kT/((AH_2+BH_2));
end

H_INF=1./((1.+exp((svolt+71.55)/7.43))*(1.+exp((svolt+71.55)/7.43)));

if(svolt>=-40.)

AJ_1=0.;
BJ_1=(0.6*exp((0.057)*svolt)/(1.+exp(-0.1*(svolt+32.))));
TAU_J= kT/((AJ_1+BJ_1));

else

    AJ_2=(((-2.5428e4)*exp(0.2444*svolt)-(6.948e-6)*exp(-
0.04391*svolt))*(svolt+37.78)/(1.+exp(0.311*(svolt+79.23)));
    BJ_2=(0.02424*exp(-0.01052*svolt)/(1.+exp(-0.1378*(svolt+40.14))));
    TAU_J= kT/((AJ_2+BJ_2));
end

J_INF=H_INF;

Xr1_INF=1./(1.+exp((-26.-svolt)/7.));
axr1=450./(1.+exp((-45.-svolt)/10.));
bxr1=6./(1.+exp((svolt-(-30.))/11.5));
TAU_Xr1=(1/kT)*axr1*bxr1;
Xr2_INF=1./(1.+exp((svolt-(-88.))/24.));
axr2=3./(1.+exp((-60.-svolt)/20.));
bxr2=1.12/(1.+exp((svolt-60.)/20.));
TAU_Xr2=(1/kT)*axr2*bxr2;

Xs_INF=1./(1.+exp((-5.-svolt)/14.));
Axs=1100./(sqrt(1.+exp((-10.-svolt)/6)));

```

```

Bxs=1./(1.+exp((svolt-60.)/20.));
TAU_Xs=(1/kT)*Axs*Bxs;

switch type
    case 'EPI'
        R_INF=1./(1.+exp((20-svolt)/6.));
        S_INF=1./(1.+exp((svolt+20)/5.));
        TAU_R=(1/kT)*(9.5*exp(-(svolt+40.))*(svolt+40.)/1800.+0.8);
        TAU_S=(1/kT)*(85.*exp(-
(svolt+45.)*(svolt+45.)/320.)+5./(1.+exp((svolt-20.)/5.))+3.);
    case 'ENDO'
        R_INF=1./(1.+exp((20-svolt)/6.));
        S_INF=1./(1.+exp((svolt+28)/5.));
        TAU_R=(1/kT)*(9.5*exp(-(svolt+40.))*(svolt+40.)/1800.+0.8);
        TAU_S=(1/kT)*(1000.*exp(-(svolt+67)*(svolt+67)/1000.)+8.);
    case 'MCELL'
        R_INF=1./(1.+exp((20-svolt)/6.));
        S_INF=1./(1.+exp((svolt+20)/5.));
        TAU_R=(1/kT)*(9.5*exp(-(svolt+40.))*(svolt+40.)/1800.+0.8);
        TAU_S=(1/kT)*(85.*exp(-
(svolt+45.)*(svolt+45.)/320.)+5./(1.+exp((svolt-20.)/5.))+3.);
end

D_INF=1./(1.+exp((-5-svolt)/7.5));
Ad=1.4/(1.+exp((-35-svolt)/13))+0.25;
Bd=1.4/(1.+exp((svolt+5)/5));
Cd=1./(1.+exp((50-svolt)/20));
TAU_D=(1/kT)*(Ad*Bd+Cd);
F_INF=1./(1.+exp((svolt+20)/7));
TAU_F=(1/kT)*1125*exp(-(svolt+27)*(svolt+27)/300)+80+165/(1.+exp((25-
svolt)/10));

FCa_INF=(1./(1.+power((Cai/0.000325),8))+0.1/(1.+exp((Cai-
0.0005)/0.0001))+0.20/(1.+exp((Cai-0.00075)/0.0008))+0.23 )/1.46;
if(Cai<0.00035)
    G_INF=1./(1.+power((Cai/0.00035),6));
else
    G_INF=1./(1.+power((Cai/0.00035),16));
end

%Update gates
sm = M_INF-(M_INF-sm)*exp(-HT/TAU_M);
sh = H_INF-(H_INF-sh)*exp(-HT/TAU_H);
sj = J_INF-(J_INF-sj)*exp(-HT/TAU_J);
sxr1 = Xr1_INF-(Xr1_INF-sxr1)*exp(-HT/TAU_Xr1);
sxr2 = Xr2_INF-(Xr2_INF-sxr2)*exp(-HT/TAU_Xr2);
sxs = Xs_INF-(Xs_INF-sxs)*exp(-HT/TAU_Xs);
ss= S_INF-(S_INF-ss)*exp(-HT/TAU_S);
sr= R_INF-(R_INF-sr)*exp(-HT/TAU_R);
sd = D_INF-(D_INF-sd)*exp(-HT/TAU_D);
sf =F_INF-(F_INF-sf)*exp(-HT/TAU_F);
fcaold=sfca;

```



```

sfca =FCa_INF-(FCa_INF-sfca)*exptaufca;
if(sfca>fcaold && (svolt)>-37)
    sfca=fcaold;
end
gold=sg;
sg =G_INF-(G_INF-sg)*exptaug;
if(sg>gold && (svolt)>-37)
    sg=gold;
end
%update voltage
svolt= svolt + HT*(-sItot);

Vs.M = sm;
Vs.H = sh;
Vs.J = sj;
Vs.Xr1 = sxr1;
Vs.Xr2 = sxr2;
Vs.Xs = sxs;
Vs.S = ss;
Vs.R = sr;
Vs.D = sd;
Vs.F = sf;
Vs.FCa = sfca;
Vs.G = sg;
Vs.Volt = svolt;
Vs.Volt2 = svolt2;
Vs.Cai = Cai;
Vs.CaSR = CaSR;
Vs.Nai = Nai;
Vs.Ki = Ki;
Vs.Itot = sItot;

end

```

B.5: StepN.m

```
function [Vs] = StepN(V,HT,tt,step,Istim)
global Ko Cao Nao Vc R F T RTONF CAPACITANCE ...
    type GNa GK GL VL ...
    currents kT

sm = V.M;
sh = V.H;
sn = V.N;
svolt = V.Volt;
svolt2 = V.Volt2;
Cai = V.Cai;
Nai = V.Nai;
Ki = V.Ki;
sItot = V.Itot;

%Needed to compute currents
Ek=RTONF*(log((Ko/Ki)));
Ena=RTONF*(log((Nao/Nai)));

%Compute currents
INa=GNa*sm*sm*sm*sh*(svolt-Ena);
IK=GK*sn*sn*sn*sn*(svolt-Ek);
IL=GL*(svolt-VL);

%Determine total current
sItot = INa + IK + IL + Istim;

%update concentrations
% dNai=-(INa)*inverseVcF*CAPACITANCE;
% Nai=Nai+HT*dNai;
%
% dKi=-(Istim + IK)*inverseVcF*CAPACITANCE;
% Ki=Ki+HT*dKi;

if mod(step,10)==0
    currents = [currents; tt, INa, IK, IL];
end

%compute steady state values and time constants
VCa = 0.03335*T*(log(Cao/Cai)-12.995);
AM=-0.1*kT*(35+svolt+VCa)/(exp(-0.1*(35*svolt+VCa))-1);
BM=4*exp(-(svolt+VCa+60)/18)*kT;
TAU_M=1/(AM+BM);
M_INF=AM/(AM+BM);
AH=0.07*kT*exp(-0.05*(svolt+VCa+60));
BH=kT/(1+exp(-0.1*(svolt+VCa+30)));
TAU_H=1/(AH+BH);
H_INF=AH/(AH+BH);
AN=kT*(-0.01*(svolt+VCa+50))/(exp(-0.1*(svolt+VCa+50))-1);
```

```

BN=kT*0.125*exp(-0.0125*(svolt+VCa+60));
TAU_N=1/(AN+BN);
N_INF=AN/(AN+BN);

%Update gates
%sm = M_INF-(M_INF-sm)*exp(-HT/TAU_M);
%sh = H_INF-(H_INF-sh)*exp(-HT/TAU_H);
%sn = N_INF-(N_INF-sh)*exp(-HT/TAU_N);

sm = sm + HT*(AM*(1-sm)-BM*sm);
sh = sh + HT*(AH*(1-sh)-BH*sh);
sn = sn + HT*(AN*(1-sn)-BN*sn);
%update voltage
svolt2 = svolt;
svolt= svolt - (HT/CAPACITANCE)*(sItot);

Vs.M = sm;
Vs.H = sh;
Vs.N = sn;
Vs.Volt = svolt;
Vs.Volt2 = svolt2;
Vs.Cai = Cai;
Vs.Nai = Nai;
Vs.Ki = Ki;
Vs.Itot = sItot;

end

```

B.6: Variables.m

```
function [V] = Variables(V_init, Cai_init, CaSR_init, Nai_init,  
Ki_init)  
V.Volt=V_init;  
V.Volt2=V_init;  
V.Cai=Cai_init;  
V.CaSR=CaSR_init;  
V.Nai=Nai_init;  
V.Ki=Ki_init;  
V.M= 0.;  
V.H= 0.75;  
V.J= 0.75;  
V.Xr1= 0.;  
V.Xr2= 1.;  
V.Xs= 0.;  
V.R= 0.;  
V.S= 1.;  
V.D= 0.;  
V.F= 1.;  
V.FCa= 1.;  
V.G= 1.;  
V.Itot = 0;  
end
```

B.7: VariablesN.m

```
function [V] = VariablesN(V_init, Cai_init, Nai_init, Ki_init)

global Cao T kT

V.Volt=V_init;
V.Volt2=V_init;
V.Cai=Cai_init;
V.Nai=Nai_init;
V.Ki=Ki_init;

VCa = 0.03335*T*(log(Cao/Cai_init)-12.995);

AM=-0.1*kT*(35+V_init+VCa)/(exp(-0.1*(35*V_init+VCa))-1);
BM=4*exp(-(V_init+VCa+60)/18)*kT;
TAU_M=1/(AM+BM);
M_INF=AM/(AM+BM);
AH=0.07*kT*exp(-0.05*(V_init+VCa+60));
BH=kT/(1+exp(-0.1*(V_init+VCa+30)));
TAU_H=1/(AH+BH);
H_INF=AH/(AH+BH);
AN=kT*(-0.01*(V_init+VCa+50))/(exp(-0.1*(V_init+VCa+50))-1);
BN=kT*0.125*exp(-0.0125*(V_init+VCa+60));
TAU_N=1/(AN+BN);
N_INF=AN/(AN+BN);

V.M= M_INF;
V.H= H_INF;
V.N= N_INF;
V.Itot = 0;
end
```

APPENDIX C

CELLSPARK RELATED COURSE MATERIALS

C.1: Electrophysiology Lecture Slides

Slide 1

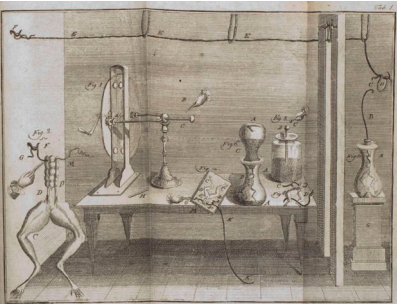
Electrophysiology

Lab 4


Slide 2

What is electrophysiology?

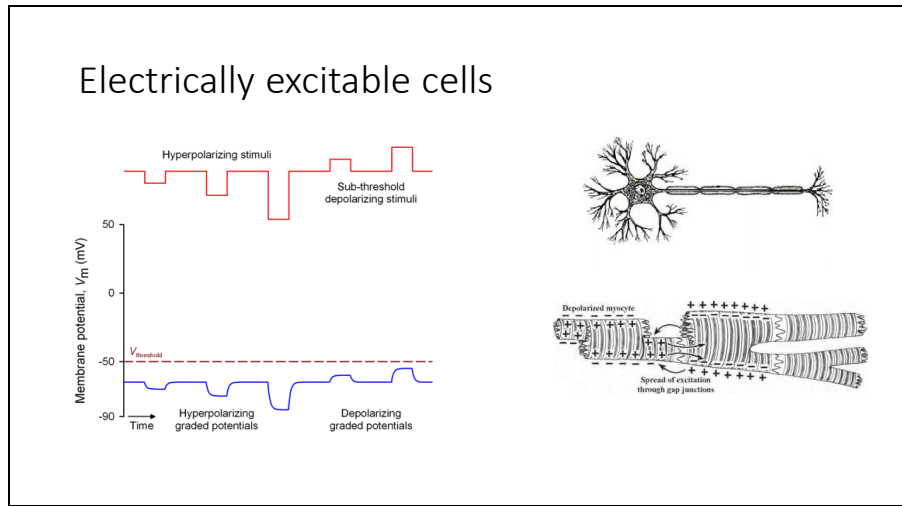
“As yet we know nothing of what goes to create or evoke the active spark of life.”
– Bram Stroker, The Jewel of Seven Stars



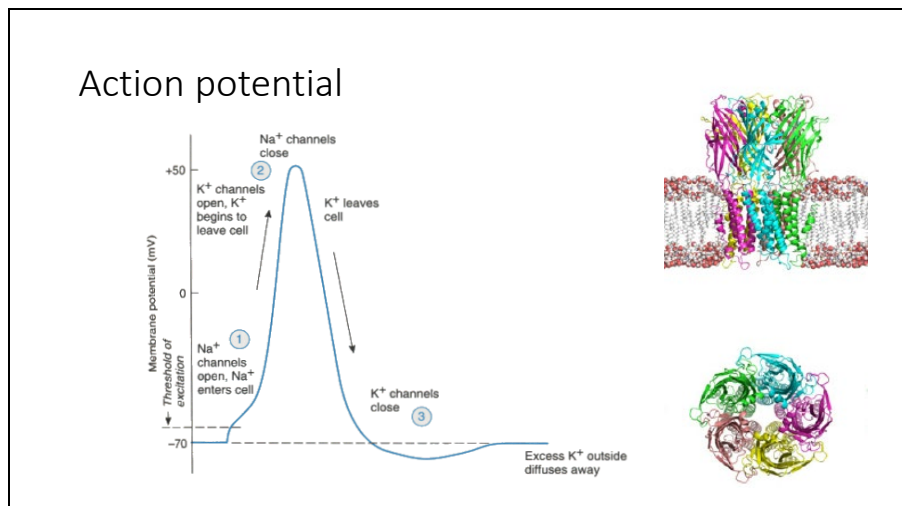
De viribus electricitatis in motu musculari – Luigi Galvani, 1780



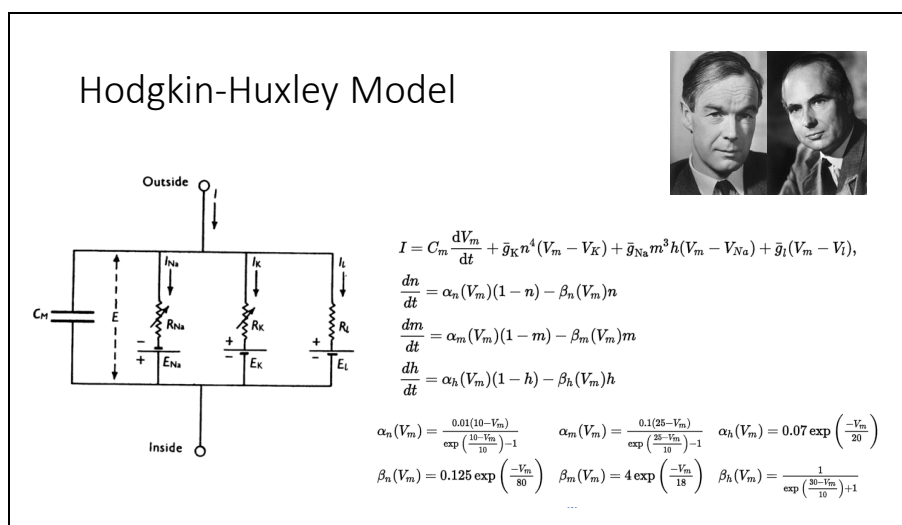
Slide 3



Slide 4



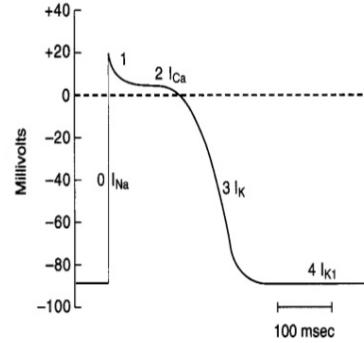
Slide 5



Slide 6

Cardiac Action Potential

- Depolarization
 - Fast Na⁺ channels
- Plateau
 - Slow Ca⁺⁺ channels
 - Slow to open
 - Slow to close
 - After depolarization, permeability to K⁺ decreases
 - Ca⁺⁺ is pumped in – excitation-contraction coupling
- Repolarization
 - Slow K⁺ channels



Slide 7

Tusscher-Noble-Noble-Panfilov (TNNP) Model

$$\frac{dV}{dt} = \frac{I_{\text{ion}} + I_{\text{stim}}}{C_m}$$

$$I_{\text{ion}} = I_{\text{Na}} + I_{\text{K1}} + I_{\text{to}} + I_{\text{Kr}} + I_{\text{Ks}} + I_{\text{CaL}} + I_{\text{NaCa}} + I_{\text{NaK}} + I_{\text{pCa}} + I_{\text{pK}} + I_{\text{hCa}} + I_{\text{hNa}}$$

$$I_{\text{Na}} = G_{\text{Na}} m^3 h^2 (V - E_{\text{Na}})$$

$$I_{\text{CaL}} = G_{\text{CaL}} d f f_{\text{Ca}} \frac{V E^2 C_{\text{CaT}} e^{2V/RT} - 0.341 C_{\text{CaT}}}{RT e^{2V/RT} - 1}$$

$$I_{\text{to}} = G_{\text{to}} F^2 (V - E_{\text{K}})$$

$$I_{\text{Kr}} = G_{\text{Kr}} s^2 (V - E_{\text{K}})$$

$$I_{\text{Ks}} = G_{\text{Ks}} \sqrt{\frac{K_s}{5.4}} s_{\text{r1}} s_{\text{r2}} (V - E_{\text{K}})$$

$$I_{\text{K1}} = G_{\text{K1}} \sqrt{\frac{K_1}{5.4}} k_{\text{K1,00}} (V - E_{\text{K}})$$

$$I_{\text{NaCa}} = I_{\text{NaCa}} \frac{e^{(V/RT) N_{\text{Na}} C_{\text{CaT}} - e^{(\tau-1)V/RT} N_{\text{Na}}^2 C_{\text{CaT}}}}{(K_{\text{mNa}}^3 + N_{\text{Na}}^3)(K_{\text{mCa}} + C_{\text{CaT}}) (1 + I_{\text{NaCa}} e^{(\tau-1)V/RT})}$$

$$I_{\text{NaK}} = I_{\text{NaK}} \frac{K_{\text{Na}}}{(K_{\text{Na}} + K_{\text{mK}})(N_{\text{Na}} + K_{\text{mNa}})(1 + 0.1245 e^{-0.1V/RT} + 0.0853 e^{-V/RT})}$$

$$I_{\text{pCa}} = G_{\text{pCa}} \frac{K_{\text{pCa}} + C_{\text{CaT}}}{V - E_{\text{K}}}$$

$$I_{\text{pK}} = G_{\text{pK}} \frac{V - E_{\text{K}}}{1 + e^{(25-V)/5.68}}$$

$$I_{\text{hNa}} = G_{\text{hNa}} (V - E_{\text{Na}})$$

$$I_{\text{hCa}} = G_{\text{hCa}} (V - E_{\text{Ca}})$$

$$I_{\text{NaK}} = V_{\text{NaK}} (C_{\text{Na}} - C_{\text{Na}}^{\text{int}})$$

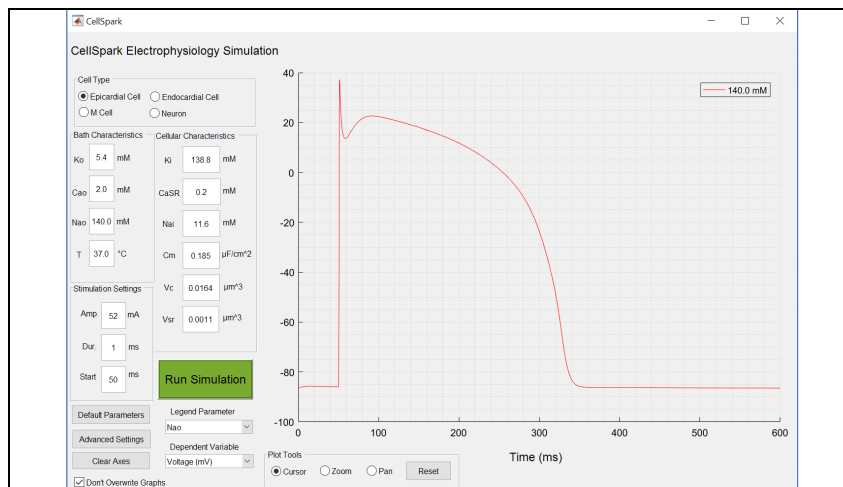
$$I_{\text{p}} = \frac{V_{\text{minusp}}}{V_{\text{minusp}} + C_{\text{CaT}}^2}$$

$$I_{\text{col}} = \left(\frac{C_{\text{CaT}}}{C_{\text{CaT}}^{\text{int}} + C_{\text{CaT}}^{\text{int}}} + C_{\text{NaT}} \right) d t$$

$$\frac{dC_{\text{CaT}}}{dt} = \frac{C_{\text{CaT}} + K_{\text{CaT}}}{C_{\text{CaT}} + K_{\text{CaT}}} \frac{2VCF}{VCF} - I_{\text{NaCa}} + I_{\text{NaK}} - I_{\text{p}} + I_{\text{col}}$$

$$\frac{dC_{\text{NaT}}}{dt} = \frac{C_{\text{NaT}} + K_{\text{NaT}}}{VCF} (-I_{\text{NaK}} + I_{\text{p}} - I_{\text{col}})$$

Slide 8



Slide 9

Midterm Lab Report

- Design and carry out an electrophysiology experiment that can be completed in CellSpark.
- Write up your findings in a journal article style lab report
 - Introduction, Methods, Results, Discussion, References
 - May include 1-2 figures
 - 4 pages max
- Submit your hypothesis to be approved for your pre-lab next week!
- **Due October 20, 2017 by 11:55pm**

Lab #4: Electrophysiology

The goal of this lab is to show you some basic cell electrophysiology techniques. In addition, you should learn how electrically excitable cells response to electrically stimuli so that you can interpret electrophysiological measurements.

Part 1: Patch Clamp vs. Microelectrode Array

We will visit the lab with the patch clamp and microelectrode array (MEA) set ups.

1. Why are both the patch clamp and MEA set ups in metal cages?

2. What are some advantages and disadvantages of doing patch clamp vs. using an MEA to measure cellular electrical responses?

Part 2: Electrically Excitable Cell Simulation

In this part of the lab, you'll be using CellSpark, a simulation software, to do a couple patch clamp type experiments virtually. Real patch clamp experiments are long and take many hours/days of practice to master the techniques.

In the software, the membrane of a neuron is modeled as the Hodgkin-Huxley circuit model below:

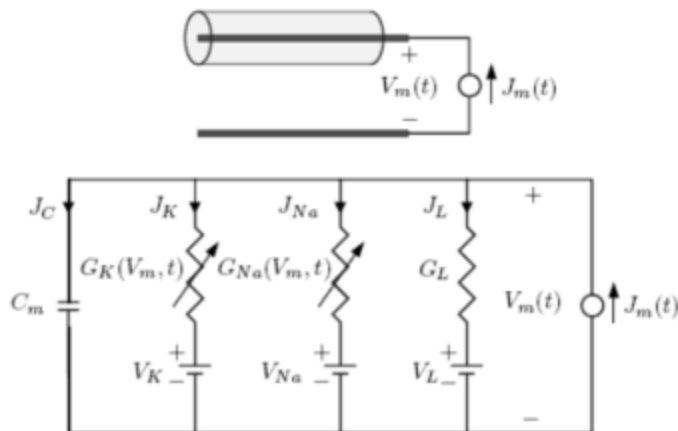


Figure 5.2: Schematic diagram and circuit model of a space-clamped axon in the current-clamp configuration.

In this simulation environment, you can interrogate a single cell in a manner similar to a real patch experiment (without all the noise, though). You can also change many of the cell and environment parameters to see the effect on the electrical response.

1. Start Matlab R2017a and click the “CellSpark” icon under apps to launch the software.
2. Under “Cell Type”, select “Neuron” and run the simulation leaving all other parameters at the default values.
3. Using the cursor, determine the peak amplitude of the action potential. _____ mV
3. Now let’s add sodium to the outside bath. Increase the box labeled “Nao” to 200 mM. What do you notice happens when you add the sodium?
4. Why do you suppose this happens? (Hint: What’s the Nernst potential for sodium with the original values and with your new increased external concentration?)

5. Now, decrease the external concentration to 100 mM (below the original value). Explain what you observe.

6. Set the simulation back to the default values by pressing the “Default Parameters” button and clear the axes by pressing the “Clear Axes” button.

7. Run the simulation again. Now change the Stimulus Amplitude to 5 μA , and run the simulation again. What do you notice about this membrane response? Would you consider this an action potential?

8. Next we will look at a model of a different electrically excitable cell, a ventricular myocyte. The model for this cell is much more complicated and includes many more ionic currents. Change the “Cell Type” to “Endocardial Cell”.

9. Run the simulation. Using the cursor, measure the peak amplitude and duration of the action potential and sketch the shape of the membrane response below. How does it differ from the neuron action potential?

Amplitude: _____ mV

Duration: _____ ms

10. Change the dependent variable of the axes to “ I_{CaL} (mA)” and run the simulation again. You should see the current through the L-Type Calcium channels plotted. How does this compare to the plot of membrane response? Explain what you observe.

C.3: Midterm Lab Assignment Prompt

Midterm Lab Assignment

Design your own Patch-Clamp cell experiment! The write-up is due by 11:55pm on the date indicated by the syllabus; turn it in on time or get a 0! (this is worth 15% of your final grade) Submit it on the TurnItIn link on Blackboard before the due date.

You can use either the axon or cardiomyocyte model. You can start now but finish later (just download the simulation installers from blackboard onto a computer.)

Ideas:

What effect does capacitance have on how susceptible myocytes are to excitation?

What's the effect of any of the ion concentrations on either of the model cells?

What effect does temperature have on the action potential duration in neurons?

Pick a good set of parameter values to test over and make sure you record your results! You should pretend you are running a real experiment (think about controls and validation for your theory for what happens).

The write-ups should be in the form of a short journal publication (~2-3 pages) and have the following: Tell me what you are planning and what your hypothesis is (Introduction and Background), then explain what you did (Methods), then what happened (Results) and then why you think it happened (Discussion). You may include one or two figures and have a maximum of 4 pages total. Please remember to cite all references you use in a bibliography. The Reference Section does not count in your page limit. Use either Times New Roman font size 12pt or Ariel/Helvetica font size 11pm with 1 in margins. For convenience, a template is included on Blackboard.

Note: references papers and journals are good references. Websites (especially Wikipedia) are NOT appropriate references. See template for bibliography format.

As an example a good hypothesis would be:

Increasing external sodium concentration increases the peak of the action potential.

You are NOT allowed to pick this hypothesis. This is the experiment you ran in lab today.

If you use this hypothesis you will get a 0.

Your hypothesis must be submitted in writing or via email to Dr. Dean or a TA before the start of lab next week. This counts as your pre-lab grade for lab #5.

C.4: Midterm Lab Assignment Grading Rubric

	Excellent (5 point)	Very Good (4 point)	Good (3 point)	Fair (2 point)	Poor (1 point)
Content	<ul style="list-style-type: none"> • Presents a <u>well-thought hypothesis</u> with <u>sufficient details</u> of reasoning behind it • Presents both qualitative and quantitative <u>results</u> with well-laid out summary graphs of the <u>relevant quantities being measured</u> • Uses <u>Hodgkin-Huxley and/or correct scientific theory to explain resulting trends</u> 	<ul style="list-style-type: none"> • Presents a hypothesis with explanation of reasoning behind it • Presents some data with calculated or summarized results • Presents detailed methods section • Uses some HH model and other scientific theory to explain reasoning 	<ul style="list-style-type: none"> • Presents a hypothesis with some background • Presents some data with graphs of raw data • Methods section contains relevant details 	<ul style="list-style-type: none"> • Presents a hypothesis; with very little details of reasoning behind hypothesis • Few scattered results • Missing details in methods section 	<ul style="list-style-type: none"> • Lacks a central hypothesis • Lacks any organized quantitative data and results
Organization / Structure	<ul style="list-style-type: none"> • Has a <u>distinct structure</u> of Introduction, Body (Development of theme), and <u>Conclusion</u> • Sentences are <u>coherent</u> • <u>Transitions</u> between paragraphs and sentences are <u>smooth and logical</u> 	<ul style="list-style-type: none"> • Has a distinct structure of Introduction, Body (Development of theme), and Conclusion • Sentences are coherent • A few of the transitions between paragraphs and sentences are jumpy 	<ul style="list-style-type: none"> • Has a distinct structure of Introduction, Body (Development of theme), and Conclusion • A few sentences are incoherent or hard to follow • Most of the transitions between paragraphs and sentences are <u>jumpy</u> 	<ul style="list-style-type: none"> • Has some structure of Introduction, Body (Development of theme), and Conclusion • Many sentences are incoherent or hard to follow • Most of the transitions between paragraphs and sentences are jumpy 	<ul style="list-style-type: none"> • Has no structure of Introduction, Body (Development of theme), and Conclusion
Discussion and Reasoning	<ul style="list-style-type: none"> • All statements are <u>accurate and concise</u> • <u>Opinions</u> (theories) and <u>facts</u> (physical evidence) are clearly <u>distinguished</u> • <u>Proper citation</u> of literature (source of information) 	<ul style="list-style-type: none"> • All statements are accurate and concise • Opinions (theories) and facts (physical evidence) are somewhat mixed up • Proper citation of literature (source of information) 	<ul style="list-style-type: none"> • All statements are accurate and concise • Opinions (theories) and facts (physical evidence) are not distinguished • Some missing citation of literature (source of information) 	<ul style="list-style-type: none"> • Some statements are inaccurate and long-winded • Opinions (theories) and facts (physical evidence) are not distinguished • Poor citation of literature (source of information) 	<ul style="list-style-type: none"> • Most statements are inaccurate • Opinions (theories) and facts (physical evidence) are not distinguished • No citation of literature (source of information)
Mechanics / Style (one select page is graded)	<ul style="list-style-type: none"> • No Grammatical error • Proper syntax • No spelling error / typographical error 	<ul style="list-style-type: none"> • No Grammatical error • Proper syntax • A few (2-3) spelling errors / typographical errors 	<ul style="list-style-type: none"> • A few (2-3) Grammatical errors • Some awkward syntax • Several (4-7) spelling errors / typographical errors 	<ul style="list-style-type: none"> • Several (4-7) Grammatical errors • Some awkward syntax • Several (4-7) spelling errors / typographical errors 	<ul style="list-style-type: none"> • Many (>7) Grammatical errors • No control of syntax • Many (>7) spelling errors / typographical errors

APPENDIX D

CELLSPARK SURVEY AND IRB DOCUMENTS

D.1: CellSpark Survey

Midterm Lab Qualitative Assessment

By completing and submitting this assessment you acknowledge that you have read the informed consent document and understand your rights as a participant in the study.

This document can be found at: <https://drive.google.com/open?id=1cRj7gFfFggDYK7eK0t0bplJ6B1ydsdUy>

Please rank the following statements based on your level of agreement:

	Strongly disagree	Disagree	Neutral	Agree	Strongly agree
The software was easy to use.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
The demonstration of the software was sufficient to understand how the software works.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
I had a strong understanding of electrophysiology before using the software.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
There were enough changeable parameters in the software for me to perform an experiment that interested me	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Performing an experiment using the software increased my understanding of electrophysiology.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
The software made me more interested in learning more about electrophysiology.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Using the software to learn about electrophysiology was preferable to a traditional lecture environment.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

If you have any other comments about the software or suggestions to improve it please list them below:

Your answer

SUBMIT

Never submit passwords through Google Forms.

D.2: Informed Consent Document

Information about Being in a Research Study Clemson University

Impact on Simulation Software on Learning Electrophysiology

Description of the Study and Your Part in It

Dr. Delphine Dean and Tyler Harvey are inviting you to take part in a research study. Dr. Delphine Dean is an Associate Professor at Clemson University. Tyler Harvey is a student at Clemson University, running this study with the help of Dr. Delphine Dean. The purpose of this research is to assess the effectiveness of a software package which simulates experiments on cells for teaching undergraduate students concepts in electrophysiology (the electrical activity of cells) as well as assessing the software's usefulness for teaching students to design and conduct scientific experiments.

Your part in the study will be to identify your level of agreement with seven statements concerning your use of the software, create a concept map of your understanding of electrophysiology by dragging and dropping concepts and relationships onto a chart, and give any additional opinion or feedback on the software.

It will take you about 20 minutes to be in this study.

Risks and Discomforts

We do not know of any risks or discomforts to you in this research study.

Possible Benefits

We do not know of any way you would benefit directly from taking part in this study. However, this research may help us to better understand the role of simulation software for engineering education and may help improve curriculum for students who take this course in the future.

Protection of Privacy and Confidentiality

No personally identifiable information will be collected in this study or will be known by any member of the research team.

Choosing to Be in the Study

You do not have to be in this study. You may choose not to take part and you may choose to stop taking part at any time. You will not be punished in any way if you decide not to

be in the study or to stop taking part in the study. If you decide not to take part or to stop taking part in this study, it will not affect your grade in any way.

Contact Information

If you have any questions or concerns about this study or if any problems arise, please contact Dr. Delphine Dean at Clemson University at 864-656-2611.

If you have any questions or concerns about your rights in this research study, please contact the Clemson University Office of Research Compliance (ORC) at 864-656-0636 or irb@clemsun.edu. If you are outside of the Upstate South Carolina area, please use the ORC's toll-free number, 866-297-3071.

Clicking on the "agree" button indicates that:

- You have read the above information
- You voluntarily agree to participate
- You are at least 18 years of age

You may print a copy of this informational letter for your files.

D.3: Participant Recruitment Prompt

Recruitment Script - Email

Subject: Optional Assessment

BIOE 3700 Students,

Now that you have completed and submitted your midterm lab report, we would like you to complete a short survey concerning the software you used to complete the assignment, CellSpark, and a short activity to gauge your understanding of cell electrophysiology.

Participation in the study is completely optional and anonymous. Your choice to participate will not affect your grade in any way.

The activity is designed to take less than 20 minutes to complete.

If you would like to participate please visit the following link and read through the informed consent document. If you understand and agree to participate, you can continue through to complete the activity. You may choose to stop participating at any time by closing the webpage.

Link: <[link to online assessment will be included here](#)>

Thank you,

Tyler Harvey

Dr. Dean

Recruitment Script – In person

Now that you have finished your midterm lab reports, you all should have received an email about participating in a survey and short activity related to the assignment. This is a study designed to help us improve the simulation software and understand whether it was effective in helping meet the objectives of the course. Participation is completely optional and anonymous, and does not affect your grade in any way.

If you would like to participate, please refer to the email for how to access and complete the assessment.