COUPLED TRANSPORT/HYPERELASTIC MODEL FOR HIGH ENERGY DENSITY NASTIC MATERIALS

by

Christopher Joseph Homison

B. S. in Mechanical Engineering, Grove City College, 2004

Submitted to the Graduate Faculty of

School of Engineering in partial fulfillment

of the requirements for the degree of

Master of Science in Mechanical Engineering

University of Pittsburgh

2006

UNIVERSITY OF PITTSBURGH

SCHOOL OF ENGINEERING

This thesis was presented
by

Christopher Joseph Homison

It was defended on

April 26, 2006

and approved by

Lisa Mauck Weiland, PhD, Assistant Professor

Anne M. Robertson, PhD, Associate Professor

Patrick Smolinksi, PhD, Associate Professor

Thesis Advisor: Lisa Mauck Weiland, PhD, Assistant Professor

COUPLED TRANSPORT/HYPERELASTIC MODEL FOR HIGH ENERGY DENSITY NASTIC MATERIALS

Christopher Homison, M.S.

University of Pittsburgh, 2006

A new development in aerospace technology involves the creation of aircraft that can undergo large changes in the shape of their wings and control surfaces. This technology, called morphing aircraft, does away with several performance compromises by allowing the aircraft to adapt to a wide variety of speed and altitude conditions. One of the challenges associated with the development of morphing aircraft is the creation of a skin that can allow for the in-plane stretching necessary for morphing but possess enough out-of-plane flexural rigidity to handle aerodynamic forces.

A new class of high energy density active materials based upon biological processes is being developed to address this problem. These materials utilize the controlled transport of charge and fluid into micron-scale inclusions. The inclusions are phase separated from the surrounding matrix by a selectively-permeable membrane. Selective stimulation of the membrane enables bulk deformation in a process referred to in the plant kingdom as *nastic movements*. The particular material considered in this work utilizes biological transport mechanisms to generate an osmotic gradient across the membrane.

The purpose of this work is to develop a physics-based computational model of the nastic material that couples ion and solvent transporter fluxes to a finite element analysis of the surrounding matrix. This model is to act as a feedback loop for material synthesis efforts. The processes occurring in the biotransport system are complex and highly coupled to one another. Key challenges in creating a viable model are the numerical solution of the resulting transport model and its coupling with the finite element analysis. The resulting model has been compared to experiment and is capable of predicting material response over a wide range of configurations. A series of parametric studies is performed to determine the relative importance of the material parameters and provide guidance to experimental efforts.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

NOMENCLATURE

a	number of ions moved into the inclusion per exchanger cycle
a_{S}	activity of species S
A	membrane area
A_0	cross sectional area of an ion channel
A_{p}	effective pore area of an ion channel
b	number of ions moved out of the inclusion per exchanger cycle
c_{S}	spatial concentration of ion S
C	effective membrane capacitance
$\mathbf{C}$	right Cauchy-Green stress tensor
C_{01}	second Mooney-Rivlin coefficient
C_{10}	first Mooney-Rivlin coefficient
$C^{(i)}$	eigenvalues of the right Cauchy-Green stress tensor
C_p	pressure compliance of the inclusion fluid
C_T	temperature compliance of the inclusion fluid
d	length of an ion channel
d_∞	probability of activation for channels with an inactivation mechanism
D_1	third Mooney-Rivlin coefficient
D_{S}	diffusion constant for ion S
e	charge of an electron
f	probability of inactivation for channels with an inactivation mechanism
F	Faraday's constant
h	Planck's constant
$i_{\mathrm{cotransporter}}$	ion cotransporter current
i_{diff}	ion diffusion current
$i_{\mathrm{exchanger}}$	ion exchanger current
i_{pump}	ion pump current

i_{S}	S-channel current
I	total transmembrane current
J	Jacobian
J_{el}	elastic volume ratio
J_i	flow of an extensive property
J_j	energy flow
J_r	flow from a chemical reaction
J_{S}	flux of ion S
J_{V}	volumetric solvent flux
k	Boltzmann's constant
$k_{\text{cotransporter}}$	ion cotransporter conductance parameter
k_{nm}^{0}	actual reaction constant without respect to ligand concentration
k_{pump}	ion pump conductance parameter
k_{S}	S-ion channel conductance parameter
k_{SX}	ion exchanger conductance parameter
K	effective solvent permeability of the membrane
K_0	initial bulk modulus of the matrix material
K_m	Michaelis-Menten coefficient
L_{ik}	phenomenological coefficients
m	number of ions moved out of the inclusion per pump cycle
m_{S}	molal concentration of species S
M	number of ion pumps in the membrane
$\lvert\mathbf{M}\rvert$	characteristic determinant for the entire carrier system
$\lvert\mathbf{M}_j\rvert$	determinant associated with the S-binding form of the carrier
$\lvert{}^{*}\mathbf{M}\rvert$	characteristic determinant for the isotopically labeled portion of the carrier system
n	number of ions moved into the inclusion per pump cycle
n_{S}	moles of S inside the inclusion
n_{W}	number of water molecules moved into the inclusion per cotransporter cycle
N	number of ion exchangers in the membrane
p	hydrostatic pressure difference across the membrane
p_{S}	partial pressure of species S
P_{S}	permeability coefficient for species S
q	ion channel gating charge

q_{S}	total moles of ions given by one mole of electrolyte
r	absolute reaction rate
R	ideal gas constant
R_{ir}	thermodynamic cross coefficient
$[\mathrm{S}]$	concentration of species S
T	absolute temperature of the inclusion fluid
T_{m}	phase transition temperature of the lipid bilayer
u	reduced membrane potential
u_{S}	mobility of ion S
U	strain energy per unit reference volume of matrix material
v	membrane potential
v_d	half-activation potential for channels with an inactivation mechanism
v_f	half-inactivation potential for channels with an inactivation mechanism
v_x	gating potential for a channel with an activation mechanism only
v_{S}	Nernst equilibrium potential for ion S
V	inclusion volume
$\overline{V}_{\mathrm{S}}$	partial molal volume of species S
W_{S}	molecular weight of species S
x	probability that a channel with an activation mechanism only is open
X	fixed charge concentration within the membrane
X_i	thermodynamic conjugate driving force
z_{S}	valence charge of ion S
α	forward reaction rate of a transporter
β	backward reaction rate of a transporter
β_{S}	partition coefficient for species S
γ_{S}	activity coefficient for species S
δ	membrane thickness
ε	fraction of the ion channel length of the channel pore
$\varepsilon_{\mathrm{th}}$	linear thermal expansion strain
θ	rate of entropy generation due to irreversible processes
κ	transmission coefficient for absolute rate law
λ	transporter reaction rate parameter
λ_i	principle stretches

$\bar{\lambda}_i$	deviatoric stretches
μ_0	initial shear modulus of the matrix material
μ_{S}	electrochemical potential of ion S
μ_{S}^0	electrochemical potential at standard state
π_{S}	osmotic pressure difference across the membrane due to species S
ρ	effective density of the inclusion fluid
σ_{S}	osmotic reflection coefficient with respect to species S
τ_{S}	relaxation time for S-ion channels
ϕ	electric potential in space
ω	osmotic coefficient
Γ	observed mass action ratio for a chemical reaction
Γ'	apparent observed mass action ratio for a chemical reaction
ΔG	change in Gibbs free energy for a chemical reaction
ΔG°	standard change in Gibbs free energy for a chemical reaction
$\Delta G^{\circ\prime}$	apparent standard change in Gibbs free energy for a chemical reaction
$\Delta G*$	free energy of activation for a process
ΔG_{ATP}	free energy released by the hydrolysis of a single ATP molecule
$\Delta G^{\circ\prime}{}_{\mathrm{ATP}}$	apparent standard free energy release from the hydrolysis of ATP
Λ	equilibrium constant for a chemical reaction
Λ'	apparent equilibrium constant for a chemical reaction
Φ	dissipative function

ACKNOLEDGMENTS

First, I would like to thank my advisor, Dr. Lisa Weiland, for her guidance and support throughout this research. I would also like to thank my collaborators, especially Vishnu Baba Sundaresan and Dr. Donald Leo at Virginia Tech.

I wish to thank Dr. G. Bard Ermentrout from the math department and Dr. Giovanni Paolo Galdi from the mechanical engineering department for their valuable discussions on the numerical solution of the model equations.

Thanks go to Dr. Anne M. Robertson and Dr. Patrick Smolinksi for serving on my committee.

Finally, I would like to thank DARPA for providing the funding for this research.

1.0 INTRODUCTION

Biological systems tend to be well optimized, efficient, and robust; this makes them ideal models for creating many new and innovative technologies. One such technology involves replacing the discrete control surfaces of traditional aircraft with morphing wings and tails so as to mimic the way birds morph their bodies in flight. By mimicking the flight of birds, these aircraft benefit from the optimization of millions of years of evolution. This new technology is called morphing aircraft.

A morphing aircraft is an aircraft capable of controlled, gross shape changes in-flight, with the purpose of increasing efficiency, versatility, and/or mission performance [1]. This allows the morphing aircraft to do away with many of the performance compromises associated with conventional aircraft, which are designed to operate over a small range of flight conditions. A morphing aircraft can adapt to a wide variety of speed and altitude conditions ranging from high-efficiency gliding to high-speed flight and maneuvering. One of the challenges associated with the development of morphing aircraft is the creation of a skin that can allow for the in-plane stretching necessary for morphing but possess enough out-of-plane flexural rigidity so as to maintain the correct airfoil shape and avoid flutter or increased turbulence [2].

Several material systems are being developed to meet this challenge. One such development considers the incorporation of engineered biological systems into synthetic materials for the development of high energy density active materials [3, 4, 5]. These materials utilize active and passive transport of charge and fluid across semi-permeable membranes surrounding aqueous inclusions to achieve bulk deformation (Figure 1.1). This process is common in the plant kingdom and is used to produce controlled motions called *nastic movements*. These aqueous inclusions are called *vesicles* and can be placed into organized arrays in order achieve controllable expansion and contraction as well as bending and twisting actuation. The goal is to develop a "nastic material" that will serve as a "smart skin" for the wings and control surfaces of morphing aircraft and create a new class of smart materials. The engineered nastic material must be capable of generating large strains while still performing a structural function. To advance this requires a substantial collaborative effort. The work of Sundaresan *et al.* [4] represents one facet of the larger DARPA funded project to achieve this goal. The University of Pittsburgh contribution to this effort is the development of a physics-based computational model of the material to serve as an experimental and design feedback loop. This introduction serves to define the fundamental characteristics of a nastic material.

The biological transport system being considered consists of a semi-permeable membrane containing *ion pumps*, *ion exchangers*, *ion cotransporters*, and *ion channels* (Figure 1.2). These transport structures are *electrogenic* meaning that they transport an ion without a counterion resulting in an electric potential difference. An *ion pump* actively transports select ions against their electrochemical gradient while *ion exchangers* and *cotransporters* use the downhill movement of one type of ion to facilitate the uphill movement of a different type of ion. Ion exchangers transport ions in opposite directions across the membrane whereas ion cotransporters transport ions in the same direction. *Ion channels* are passive transport structures that selectively allow certain ions to travel down their electrochemical gradient. The biological transport model presented in this work includes ion pump and ion exchanger models based on reaction kinetics, and an ion channel model based upon the Nernst-Planck equation. Solvent flux through the semi-permeable membrane is modeled as a function of the hydrostatic and osmotic pressure differences across the inclusion boundary. The resulting system of ordinary differential equations is solved using an appropriate numerical method.

The actuation process (Figure 1.3) is initiated by the introduction of ATP, which activates the ion pumps and establishes an electrochemical gradient of the transported species. This, in turn, activates the channels, exchangers and/or cotransporters to further establish an osmotic gradient, which causes solvent to flow into the inclusion. The mechanical constraint of the surrounding polymer matrix causes an increase in the hydrostatic pressure of the inclusion fluid. The solvent flux will continue until the osmotic pressure is balanced by the hydrostatic pressure.

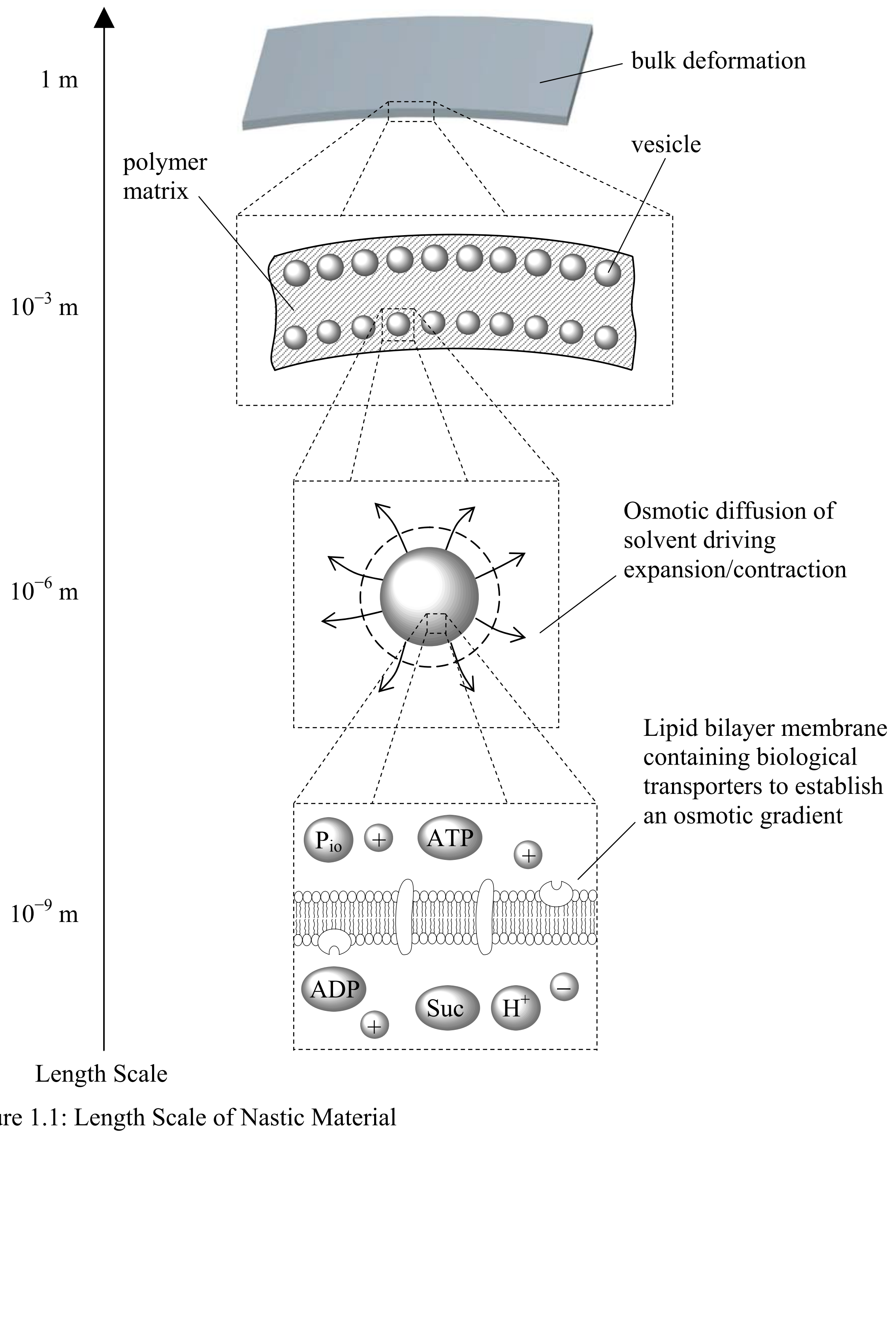

1 m
bulk deformation
polymer matrix
vesicle
10^{-3} m
10^{-6} m
Osmotic diffusion of solvent driving expansion/contraction
Lipid bilayer membrane containing biological transporters to establish an osmotic gradient
10^{-9} m
P_{io}
+
ATP
+
ADP
+
Suc
H^+
–
Length Scale

Figure 1.1: Length Scale of Nastic Material

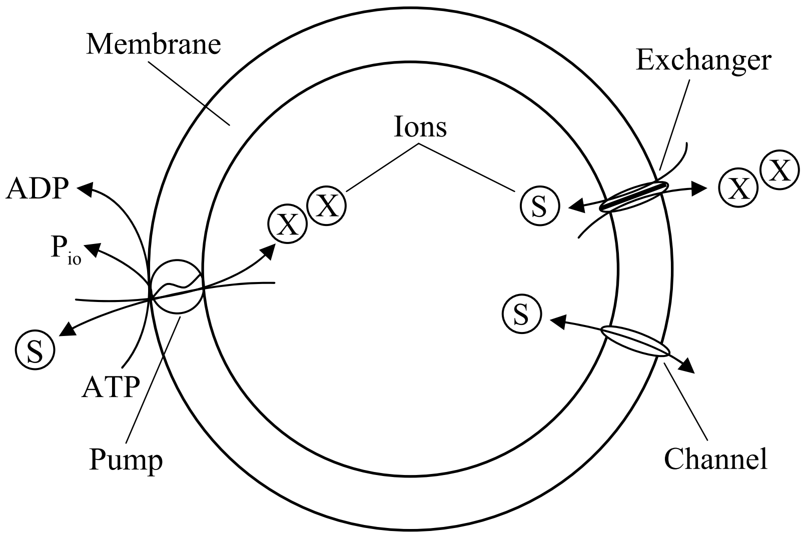


Figure 1.2: Transport Components – A schematic representation of vesicle membrane components including an ion pump, an ion exchanger, and an ion channel. Both the channel and exchanger are reversible with respect to ion flow.

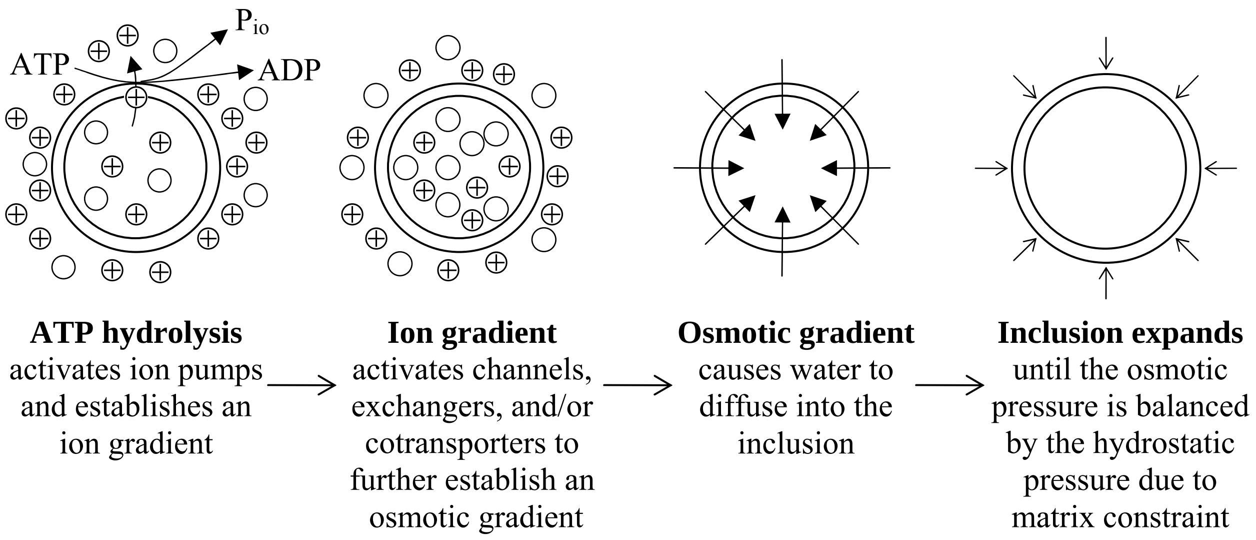

P_{io}
ATP
ADP
ATP hydrolysis activates ion pumps and establishes an ion gradient
Ion gradient activates channels, exchangers, and/or cotransporters to further establish an osmotic gradient
Osmotic gradient causes water to diffuse into the inclusion
Inclusion expands until the osmotic pressure is balanced by the hydrostatic pressure due to matrix constraint

Figure 1.3: Actuation Process – Processes involved in using the energy stored in ATP to achieve actuation for a single inclusion.

In the engineered nastic material, the surrounding matrix is a hyperelastic polymer. Matrix response due to vesicle expansion/contraction is modeled using ABAQUS/Standard. Hydrostatic fluid elements are used to couple the mechanical deformation of the polymer matrix with the hydrostatic pressure of the inclusion fluid. The hyperelastic model, solved using finite element analysis, is coupled to the transport model through a user-defined fluid constitutive model subroutine. Because this modeling approach is too computationally intensive to be applied at the structural level, macroscopic constitutive response equations must be developed once a viable material design has been achieved. This computational model will be used to generate a large number of loading curves from which phenomenological expressions may be derived. Chapter 2.0 of the following thesis provides a detailed background and literature review while Chapter 3.0 provides an overview of the model development. Chapter 4.0 details the validation and implementation of the nastic model. The results from a series of parametric studies aimed at determining the relative importance of the model parameters are presented in Chapter 5.0. Conclusions and future work are presented in Chapters 6.0 and 7.0 respectively. Detailed derivations of several of the model equations as well as the source codes used to implement the model are presented in the Appendices.

2.0 BACKGROUND AND LITERATURE REVIEW

The concept of smart materials and structures was developed in the 1980s and describes a material that can transform energy from one domain (i.e. electrical) to another (i.e. mechanical). Examples of smart materials include: piezoelectrics, shape-memory alloys, magnetorheological fluids, polyelectrolyte gels, pyroelectrics, photostrictive materials, magneto-optical materials, and superconducting materials [6]. The characteristics of smart materials can be divided into two categories: *passive smartness* and *active smartness* [7]. A passive smart material is one that can respond to its environment in a useful manner without external fields, forces, or feedback systems whereas an active smart material can make use of external stimulus. Smart materials that are based on biological systems are called *biomimetic*. Two terms used to describe the actuation properties of smart materials are *blocking force* which is the maximum load that an active material can support with no displacement and *free displacement* which is the maximum displacement that can be generated at zero load [8].

The goal of this work is to develop a physics-based computational model of a "nastic material" utilizing biological transport mechanisms that will serve as a "smart skin" for the wings and control surfaces of morphing aircraft and create a new class of active smart materials. To accomplish this, a model must be created that can act as a feedback loop for material synthesis efforts. There are multiple available approaches to create the model including thermodynamic, enzyme kinetics, and electrodiffusion strategies. Because of their relative simplicity, various reversible thermodynamic approaches have been explored for nastic material modeling, including efforts by Sundaresan *et al.* [4, 9] and Giurgiutiu *et al.* [10]. The main drawback to these approaches is that they do not capture the complex nature of either the transport or matrix response and cannot give information on transient response. Several pure chemical reaction kinetics models of alternate systems exist [11, 12, 13, 14, 15], and are presumably more accurate; however they require the determination of several rate constants which cannot be directly measured. An approach combining thermodynamic, enzyme kinetics, and electrodiffusion balances application of a physics-based approach with development of a method readily compatible with experiment and thus forms the basis of the approach applied here. As outlined in Homison and Weiland [16, 17], this approach is capable of providing detailed insights into the transport processes, matrix response, and transient response.

This chapter provides an overview of nastic material components as well as information on the various processes and models that govern them. A literature review giving an overview of current biological transport models and theories provides insight into vesicle properties. This chapter ends with an overview of the various numerical algorithms that exist to integrate the resulting system of nonlinear ODEs.

2.1 THE LIPID BILAYER MEMBRANE

The biological transport system being considered consists of a lipid bilayer membrane containing biological transport proteins. A lipid bilayer is the main structural component in biological membranes [18, 19, 20]. It is composed of phospholipids which have a phosphate head that is hydrophilic and a lipid tail that is hydrophobic. When in contact with water, this amphiphatic structure causes phospholipids to group together into a two-molecule thick sheet with the hydrophobic tails facing each other (Figure 2.1). When this sheet is planar it is called a lipid bilayer. Lipid bilayers are thin, having a total thickness of 50 Å with a 30 Å-thick hydrocarbon interior. They are flexible, have the ability to reseal if broken, and are selectively permeable [21]. Also, lipid bilayers can contain transporter proteins which allow certain species to cross the membrane even if the lipid bilayer itself is not permeable to them.

When a synthetic lipid bilayer is formed on a supporting substrate, it is called a *black lipid membrane* (BLM). This membrane was first described by Mueller *et al.* (1962) [22] and consists of three components: lipid bilayers, a thicker annulus that forms at the interface between the bilayer and substrate, and *microlenses*, or pockets of decane trapped within the membrane [23]. The lipid bilayer membrane of the nastic program is supported on a synthetic substrate to improve mechanical integrity [5, 24].

Lipid bilayers can exist in three states, or phases; the first phase is the crystalline state in which the phospholipids are in ordered arrays [18, 25]. The second phase occurs when the temperature of the lipid bilayer is raised above its melting transition temperature, T_{m}, and is referred to as the gel state, L_{β}. The gel state serves as an intermediate state between the crystalline and the disordered crystal state, or fluid phase, L_{α}. The melting transition temperature is a function of the type of lipids that form the membrane and the hydrostatic pressure. The permeability of the membrane and kinetics of the transporter proteins are dependent on the phase of the lipid bilayer. The permeability of the membrane is higher in the fluid phase than in the gel phase, and is significantly higher during the transition between the two phases due to the highly permeable interface regions between the gel and fluid phases [18]. The membrane is generally less permeable in the gel state and the more ordered structure can slow the rate of ion pumps and ion exchangers and cotransporters that must translocate species across the membrane. Ion channels are relatively unaffected by the phase of the membrane. The pressures in the nastic material may be high enough to cause the membrane to transition from the fluid phase to the crystalline phase. A detailed review of lipid bilayer membranes including experimental methods, permeability, and electrical properties is given by Tien [26].

Under biological conditions, most lipid bilayers are in the fluid phase; this is the assumed state for the nastic model. This assumption may need to be readdressed in the future; additional information on the effects of hydrostatic pressure on bilayer phase is provided in the Chapter on Future Work (Section 7.4).

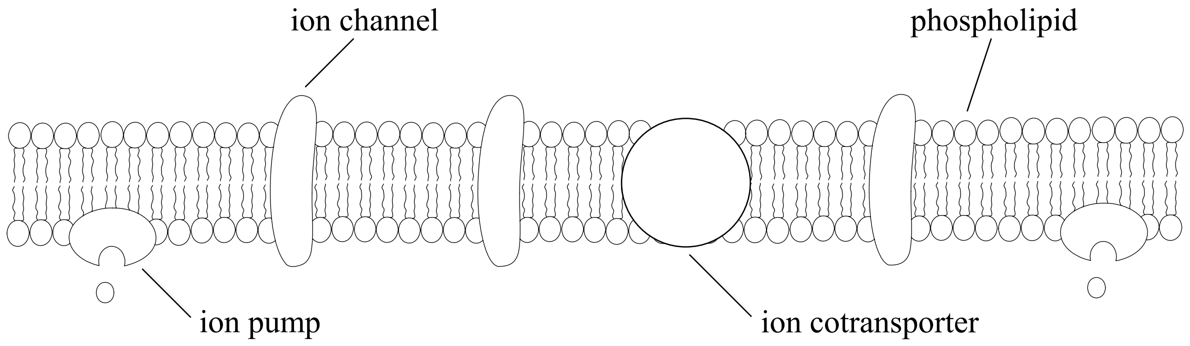


Figure 2.1: Lipid Bilayer Membrane – A planar lipid bilayer containing transport proteins.

2.1.1 Electrochemical and Nernst Equilibrium Potentials

In order to model transport processes, it is necessary to track the work required to introduce additional species to that system. The electrochemical potential, or partial molal free energy, represents the change in free energy of the system resulting from the addition of 1 mole of S when temperature, pressure, and the amounts of all other substances are constant [27]. The electrochemical potential, μ_S, is defined by [28],

$$\mu_\mathrm{S} = \left(\frac{\partial G}{\partial n_\mathrm{S}}\right)_{\substack{T,p,n_\mathrm{B} \\ \mathrm{B}\neq\mathrm{S}}} = \mu_\mathrm{S}^0 + RT\ln[\mathrm{S}] + z_\mathrm{S} F v + \overline{V}_\mathrm{S} p + m_\mathrm{S} g h \tag{2.1.1}$$

where G is the Gibbs free energy of the system, n_B is the amount of substance B in the system, μ_S^0 is the electrochemical potential at standard state, R is the ideal gas constant, T is the absolute temperature, [S] is the concentration of species S, z_S is the valence charge of species S, F is Faraday's constant, v is the electric potential difference across the boundary of the system, $\overline{V}_\mathrm{S}$ is the partial molal volume of S, p is the pressure difference across the boundary of the system, m_S, is the molar mass of species S, g is the local gravitational acceleration, and h is the height above sea level. The partial molal volume, $\overline{V}_\mathrm{S}$, is defined as the change in the volume of the solution per mole of S added to the solution at concentration [S]. It is typically close to the actual molal volume of S which is defined as the atomic weight divided by density. In modeling biological systems, the pressure and gravitational contributions to the electrochemical potential are often ignored. Under these assumptions, the electric potential required for equilibrium between two chambers denoted by the subscripts 'i' and 'e' is,

$$v_\mathrm{S} = v_\mathrm{i} - v_\mathrm{e} = \frac{RT}{z_\mathrm{S} F}\ln\frac{[\mathrm{S}]_\mathrm{e}}{[\mathrm{S}]_\mathrm{i}} \tag{2.1.2}$$

This potential is used extensively in electrochemistry calculations and is called the Nernst equilibrium potential.

2.1.2 Classification of Transport Processes

The terms *primary active*, *secondary active* and *passive transport* have specific thermodynamic definitions. The definitions come from the linear phenomenological equations as presented by Schultz [27], and first defined by Kedem [29], to read,

$$J_i = (X_i / R_{ii}) - \sum_j (R_{ij} J_j / R_{ii}) - (R_{ir} J_r / R_{ii}) \tag{2.1.3}$$

where J_i is the flow of an extensive property, J_r is the flow of an extensive property resulting from a chemical reaction, R_{ir} is the cross coefficient that quantifies the coupling between J_i and J_r, and J_j represents all other flows. The term X_i is called the *conjugate driving force* and is defined as the difference in the conjugate intensive property between the exterior and interior spaces separated by the membrane that drives the flow. Flows against their conjugate driving forces in which $R_{ir} \neq 0$ are referred to as *primary active*, or *active*, transport, also known as *chemi-osmotic* transport. These flows are thermodynamically unfavorable, or endergonic, and hence must be coupled to an exergonic process such as a downhill chemical reaction. Flows against their conjugate driving force that are driven by interactions with other flows are referred to as *secondary active* transport, also referred to as *osmo-osmotic* transport. The solute whose electrochemical gradient drives the secondary flux is called the *cosolute* of the secondary active transport process [30]. Secondary active transport differs from active transport in that the energy for transport comes from the downhill motion of the cosolute rather than a chemical reaction. Flows that are driven by their conjugate driving forces are referred to as *passive* transport. The terms *active*, *secondary active*, and *passive* will be used to describe transport processes throughout this document. In the nastic system, ion pumps are active, channels are passive, and exchangers and cotransporters are secondary active.

Membrane transporters can be divided into two categories: catalytic systems and stoichiometric systems [31]. *Catalytic systems* are systems in which the amount of substance transported is not related to the reaction sequence of the transporter, if there is one. They only allow for downhill transport and their reaction sequences consist of open and closed states. An example of a catalytic system is an ion channel. *Stoichiometric systems* are systems in which the amount of substance transported is related numerically to the cycle of the reaction sequence.

They can carry out active transport and their reaction sequences have at least two states. Ion pumps, exchangers, and cotransporters are stoichiometric systems.

The theories governing transport processes can be divided into three main groups [32]: (I) those based on the Nernst-Planck equation, (II) those based on the concept of the absolute reaction rate, and (III) those using the principle of nonequilibrium thermodynamics. All three contribute to the nastic material model.

(I) The *Nernst-Planck equation* represents the sum of the flux due to drift in an electric field and that due to diffusion along a concentration gradient. In one-dimensional form, the Nernst-Planck equation for ion S is [27, 32],

$$J_{\mathrm{S}} = u_{\mathrm{S}} c_{\mathrm{S}} \left(-\frac{d\mu_{\mathrm{S}}}{dx} \right) = -u_{\mathrm{S}} \left[RT \frac{dc_{\mathrm{S}}}{dx} + c_{\mathrm{S}} z_{\mathrm{S}} F \frac{d\phi}{dx} \right] = -u_{\mathrm{S}} c_{\mathrm{S}} \left[RT \frac{d \ln c_{\mathrm{S}}}{dx} + z_{\mathrm{S}} F \frac{d\phi}{dx} \right] \tag{2.1.4}$$

where u_{S} is the mobility of S, $c_{\mathrm{S}} = c_{\mathrm{S}}(x)$ is the concentration of S, $\phi = \phi(x)$ is the electric potential, and the remaining symbols have previously been defined. For nonelectrolytes, the Nernst-Planck equation becomes Fick's First Law of diffusion when the Einstein diffusion relationship, $D_{\mathrm{S}} = u_{\mathrm{S}} RT$, which relates the diffusion coefficient, D_{S}, to the mobility, is employed. In fact, the Nernst-Planck equation is equivalent to adding together Fick's Law and Ohm's Law.

Nernst-Planck theory is used foremost in the modeling of ion channels and electrodiffusion across the membrane itself. The Nernst-Planck channel model can be expanded to include ion-ion interactions and single-file ion movement; however, the resulting solutions are complex [33].

(II) The second class of transport theory is the *theory of absolute reaction rates* which assumes that the diffusing species must cross the membrane in a series of jumps. According to the theory of absolute reaction rates, a reaction rate constant for any process, *r*, is given by [32],

$$r = \kappa \frac{kT}{h} \exp\left(-\frac{\Delta G^*}{RT} \right) \tag{2.1.5}$$

where κ is the transmission coefficient, *k* is Boltzmann's constant, *h* is Planck's constant, and ΔG^* is the free energy of activation for the process. In order to apply the theory of absolute reaction rates to a membrane, rate constants must be calculated not only for the jump across the membrane but also for the jumps between the membrane and the bulk solutions. The theory of absolute reaction rates is employed in the development of transporter kinetics equations for all membrane transporters (pumps, channels, exchangers, and cotransporters, Section 3.1.1).

The theory of absolute reaction rates provides alternate approaches to modeling ion transport through the Eyring rate theory and Brownian dynamics [33]. Eyring Rate Theory transport models apply the theory of absolute reaction rates and are based on the assumption that the diffusing species is surrounded by neighboring molecules. Diffusion through a channel is modeled as a series of jumps resulting in a set of first-order reactions. As in the case of the Nernst-Planck Theory, ion-ion interaction and single-file formulation can be derived. Another application of Erying Rate Theory is the derivation of the exponential dependence of the translocation step of transport structures on membrane potential. The Brownian dynamics transport models solve the equations of motion, such as Newton's laws, for all particles in the system. The main drawback to the Eyring Rate Theory and Brownian dynamics models is their complexity.

(III) The third transport theory, which is based on *nonequilibrium thermodynamics*, is used to (i) provide general relations that are independent of molecular models and (ii) establish certain criteria that are valid for dissipative systems that are not in equilibrium [32, 34]. It is based on the premise that the dissipative function, Φ, must be satisfied,

$$\Phi = T\theta = \sum_i J_i X_i \geq 0 \tag{2.1.6}$$

where θ is the rate of entropy generation due to irreversible processes, J_i are flows, and X_i are forces. The dissipative function can always be related to the dissipation of free energy, or the rate of decrease in the ability to perform useful work [27]. It is also assumed that the phenomenological coefficients, L_{ik}, in the linear phenomenological equations,

$$J_i = L_{ii} X_i + \sum_i L_{ik} X_k \tag{2.1.7}$$

satisfy the Onsager relations,

$$L_{ik} = L_{ki} \tag{2.1.8}$$

The phenomenological coefficients, L_{ij}, relate the flow of an extensive property, J_i, to both conjugate, L_{ii}, and nonconjugate, $L_{ij(j \neq i)}$, driving forces. The Onsager relations reduce the number of experimentally determined coefficients for a system with n flows from n^2 to $n(n+1)/2$ [27]. This approach has the advantage of being general and able to predict results for any configuration [34]. However, the ability of this approach to generate quantitative results is limited because of the inability to directly measure many of the phenomenological coefficients. Nonequilibrium thermodynamics are used primarily in the development of the Kedem-Katchalsky equations for osmotic diffusion (Section 2.1.5).

2.1.3 Biological Transporters

There are three main types of biological transporters: *ion pumps*, *ion channels*, and *ion exchangers* and *cotransporters*. An *ion pump*, or ATPase, is an active transporter that moves select ions against their respective electrochemical gradients [28]. These pumps are most often powered by the hydrolysis of adenosine triphosphate (ATP), which is a primary energy source in biological systems, resulting in adenosine diphosphate (ADP) and inorganic phosphate (P_{io}). Ion pumps can move more than one type of ion into or out of a cell in a single cycle (Figure 2.2).

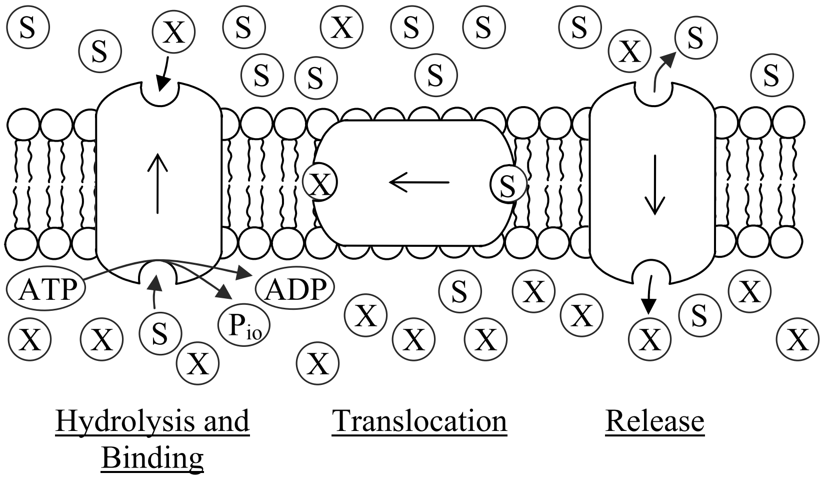


Figure 2.2: Ion Pumps – A representation of ion pumps in their three primary states: hydrolysis and binding, translocation, and release. The generic ion pump shown carries ions S and X in opposite directions across the membrane against their concentration gradients.

Ion channels are passive transport structures that selectively allow certain ions to travel down their electrochemical gradients [35]. These channels are gated so that they are not always permeable to ions and can be activated by membrane potential, concentration gradient, or stress in the membrane (Figure 2.3). Detailed discussion of ion channels including gating mechanisms and channel modeling can be found in Hille [35], Cooper *et al.* [33], and Bett and Rasmusson [36].

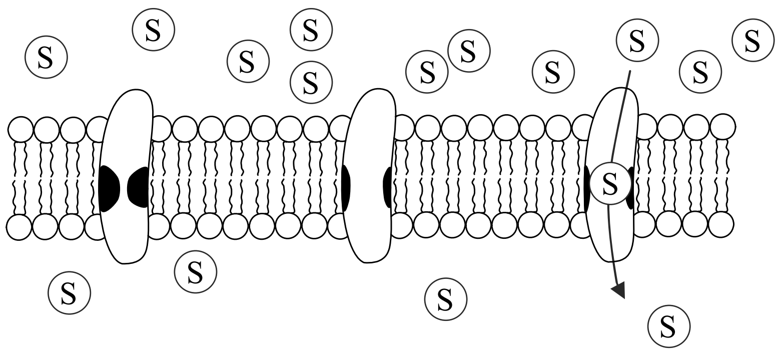


Figure 2.3: Ion Channels – A representation of ion channels in a closed state (left), an open state (middle), and allowing ion S to travel along its concentration gradient (right).

Ion exchangers and *cotransporters*, also called *antiporters* and *symporters* respectively, are secondary active transporters that use the energy from the downhill movement of one type of ion across the membrane to move a different type of ion uphill against its electrochemical gradient [28]. Ion exchangers transport ions in opposite directions whereas cotransporters move ions in the same direction (Figure 2.4). There are two classes of ion-coupled transporters: *simultaneous* and *consecutive* [37, 38]. Simultaneous transporters require the binding of all transported species to the carrier before crossing the membrane. Consecutive, or ping-pong, transporters carry one molecule across the membrane at a time. Most transporters are simultaneous and hence only simultaneous transporters will be considered. Detailed discussion of ion exchangers and cotransporters including structure, kinetics, and regulation can be found in Tanner and Caspari [38]. The terms *ion exchanger* and *ion cotransporter* will be used throughout this document.

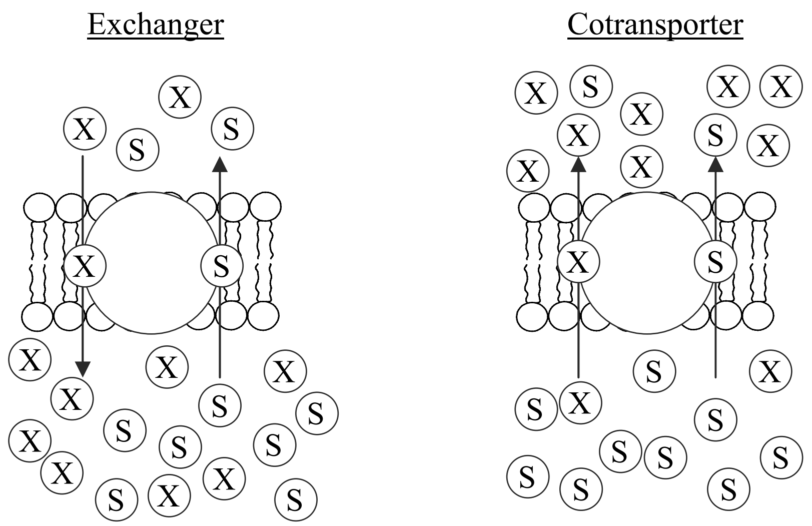


Figure 2.4: Ion Exchangers and Cotransporters – A representation of ion exchangers (left) and ion cotransporters (right) where ion S travels down its concentration gradient and carries ion X against its concentration gradient.

2.1.4 Energy Release by a Chemical Reaction

The free energy released from the hydrolysis of ATP stimulates the ion pumps in a nastic system. The actual chemical reaction for the hydrolysis of ATP is quite complex because ATP can exist in several ionized states depending on the pH and the concentration of free Mg^{2+} [21]. However, reasonable estimates may be made of the free energy released by treating the reaction as a relatively simple function of the concentrations of ATP and the hydrolysis products ADP and P_{io} [39]. This enables analysis of the free energy release based on a generic reversible reaction, where *a* moles of A and *b* moles of B react to form *c* moles of C and *d* moles of D,

$$a\mathrm{A} + b\mathrm{B} \rightleftharpoons c\mathrm{C} + d\mathrm{D} \tag{2.1.9}$$

The change in Gibbs free energy for this reaction is given by [39],

$$\Delta G = -RT \ln \Lambda/\Gamma \tag{2.1.10}$$

where Λ is the equilibrium constant determined by,

$$\Lambda = \frac{[\mathrm{C}]_{\mathrm{eq}}^{c}[\mathrm{D}]_{\mathrm{eq}}^{d}}{[\mathrm{A}]_{\mathrm{eq}}^{a}[\mathrm{B}]_{\mathrm{eq}}^{b}}\mathrm{Molar}^{(c+d-a-b)} \tag{2.1.11}$$

and Γ is the observed mass action ratio, which uses the instantaneous concentrations, and is given by,

$$\Gamma = \frac{[\mathrm{C}]_{\mathrm{obs}}^{c}[\mathrm{D}]_{\mathrm{obs}}^{d}}{[\mathrm{A}]_{\mathrm{obs}}^{a}[\mathrm{B}]_{\mathrm{obs}}^{b}}\mathrm{Molar}^{(c+d-a-b)} \tag{2.1.12}$$

The notation $\mathrm{Molar}^{(c+d-a-b)}$ is used to ensure that the ratios are dimensionless. Equation (2.1.10) is often given in terms of the standard Gibbs energy change, ΔG°, as,

$$\Delta G = \Delta G^{\circ} + RT\ln\left(\frac{[\mathrm{C}]_{\mathrm{obs}}^{c}[\mathrm{D}]_{\mathrm{obs}}^{d}}{[\mathrm{A}]_{\mathrm{obs}}^{a}[\mathrm{B}]_{\mathrm{obs}}^{b}}\right) \tag{2.1.13}$$

where ΔG° is the change in Gibbs energy when the concentration of all reactants and products is 1 M at 1 atm,

$$\Delta G^{\circ} = -RT\ln\Lambda \tag{2.1.14}$$

Because the reactants and products are typically partially ionized in biological systems and the actual equilibrium constant is difficult to measure, an apparent equilibrium constant, Λ', and corresponding $\Delta G^{\circ\prime}$ are employed. This constant is calculated from the total concentration of each reactant and product, ignoring any effects of ionization or chelation and omitting any protons that are involved. The term chelation is used to describe the forming of multiple bonds between a ligand, or the molecule or ion that surrounds the metal in a complex ion, and a metal ion. These factors are important because even at physiological pH levels ATP, ADP and P_{io} are all partially ionized and ATP and ADP can be chelated with different affinities if a metal cation is present. Therefore, the apparent equilibrium constant is not universal but depends upon all of the factors that are omitted. These omissions are cancelled out provided that the apparent observed mass action ratio, Γ', is calculated in the same manner. While not applied here, an additional adjustment can be made to include the effects of nonstandard pH [10].

2.1.5 Water Transport across a Lipid Bilayer Membrane

Because of the presence of biological transporters, water will be used as the solvent in the nastic material. The term solvent refers to the component of the solution present in the largest amount. The term solute refers to the component of the system present in the smaller amount. Water can cross the lipid bilayer by three different routes (Figure 2.5) [40]. The first is simple diffusion through the lipid bilayer. The lipid bilayer is not normally permeable to water, but water can pass through it when the chemical potential gradient for water is high enough to break the hydrogen bonds in the hydrophobic interior of the bilayer. The phosphate heads are not a significant barrier to diffusion [41]. The activation energy for water flux through the lipid bilayer is about 63 kJ/mole (15 kcal/mole) [40]. Water crosses the membrane by a solubility-diffusion mechanism where its concentration in the membrane remains very low [41]. The permeability of the membrane without any specific transport pathways is called the *basal* permeability. The second means by which water can cross the membrane is through dedicated water channels. The effective water permeability of a membrane containing water channels is significantly higher than for a lipid bilayer alone. The third manner with which water can cross the membrane is through other transporters. Many channels and cotransporters act as water channels. Also, cotransporters can actively transport water against its chemical potential gradient with a fixed stoichiometry as part of the transport cycle. All of the above are catalytic osmotic diffusion processes except cotransport which is typically a stoichiometric process.

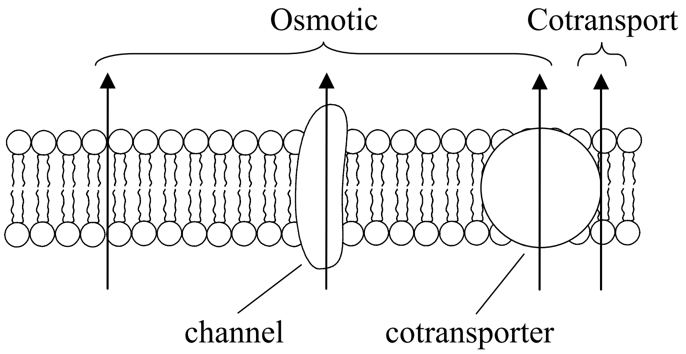


Figure 2.5: Water Transport Routes and Mechanisms – The three routes of water transport across a biological membrane: diffusion across the lipid bilayer, water channels, and cotransport.

Osmotic diffusion can be modeled using the Kedem-Katchalsky equation for solvent flux which is a modified version of Starling's law of filtration that includes an osmotic reflection coefficient. This equation is part of the set of Kedem-Katchalsky equations which are derived using irreversible thermodynamics [27]. The volumetric solvent flux, J_V, is given by,

$$J_V = K\left(\sum_S \sigma_S \pi_S - p\right) \tag{2.1.15}$$

where K is the effective permeability of the membrane with respect to solvent taking into account the contributions from transporters and is sometimes referred to as the *hydraulic conductivity* of the membrane, σ_S is an experimentally determined osmotic reflection coefficient for species S, π_S is the osmotic pressure difference across the membrane caused by species S, and p is the hydrostatic pressure difference across the membrane. Hydrostatic pressure is the actual pressure of the fluid generated by some mechanical means. The osmotic pressure is the hydrostatic pressure that would have to be applied to the concentrated solution to offset the difference in water activities across the membrane so that there is no net solvent flux. This equation is used successfully by Su *et al.* [42] to model an osmotic microactuator.

The osmotic reflection coefficient, σ_S, which is sometimes referred to as the Staverman factor, is the ratio of the flow of solvent relative to solute compared with the total volume flow under the influence of a pressure gradient alone [28] which can be expressed as,

$$\sigma_S = 1 - [\mathrm{S}]_e / [\mathrm{S}]_m \tag{2.1.16}$$

where $[\mathrm{S}]_m$ is the concentration of species S in the fluid crossing the membrane. Thus it is a measure of the *semipermeability* of the membrane to a given solute. If the membrane is perfectly selective, then $\sigma = 1$ and only pure solvent will be allowed to diffuse across the membrane. If the membrane does not distinguish between solute and solvent, i.e. the fluid crossing the membrane will have the same concentration of solute as the medium from which it is flowing, then $\sigma = 0$. *The permeability of the membrane with respect to solvent is the effective permeability of the membrane and transport components and not the basal permeability.* Therefore, the permeability of the membrane will be a function of the density of transporters that are permeable to water.

The osmotic pressure difference across the membrane, π_S, caused by species S is typically determined using the van't Hoff equation for an ideal solution, or a solution in which the number of moles of solvent is significantly greater than the number of moles of solute,

$$\pi_S = RT\left([\mathrm{S}]_i - [\mathrm{S}]_e\right) \tag{2.1.17}$$

The assumption of an infinitely dilute solution is not strictly valid for the nastic system. Robinson and Stokes [43] provide a more general expression for concentrated solutions,

$$\pi_S = \frac{RT}{\overline{V}_A} \frac{q_S m_S W_A}{1000} \omega \tag{2.1.18}$$

where $\overline{V}_A$ and W_A are the partial molar volume and molecular weight respectively of the solvent A, m_S is the molal concentration of S which is defined as the moles of S per kg of solvent, q_S is the total number of moles of ions given by one mole of electrolyte and has a value of 1 for nonelectrolytes, and ω is an experimentally determined osmotic coefficient that takes into account the non-ideal behavior of the solute in the solution.

The amount of solute carried by the volumetric solvent flux, J_V, can be determined by the second Kedem-Katchalsky equation. However, this equation does not take into account the effects of membrane potential on the solute flux and hence will not be used in the present thesis. More complex versions of the Kedem-Katchalsky equations exist [27] that include the effects of membrane potential but they require several experimentally determined parameters.

For additional detailed discussion of membrane diffusion processes, both simple and facilitated, refer to Stein [28] and Sten-Knudsen [44]. Detailed discussion of water transport across biological membranes is given by Finkelstein [41] including information on osmotic diffusion across lipid bilayer and pores, and the thermodynamics of osmotic diffusion.

2.1.6 Membrane Potential

The membrane potential is the electric potential difference between the interior and exterior faces of the membrane. Thus,

$$v = v_i - v_e \tag{2.1.19}$$

The first model to attempt to predict the potential across a biological membrane was developed by Hodgkin and Huxley (1952) [45, 46]. It assumes that the membrane is isopotential and that transporters are evenly distributed. Also, it is assumed that the number of transporters is large enough that individual gating events are averaged out and that there are no local interactions between small numbers of channels. Because the membrane is a thin insulating layer separating two conducting solutions, it acts as the dielectric for a capacitor. The membrane capacitance is charged by ion flux and the establishment of a diffuse double layer at each of the membrane faces [47]. A resistor/potential source series combination for each type of ion to which the membrane is permeable is in parallel with the membrane capacitance (Figure 2.6). The resistance is determined by the instantaneous effective permeability of the membrane including that caused by channels, pumps, exchangers and cotransporters. The magnitude of the potential source is given by the Nernst potential. Therefore, the membrane current, *I*, can be divided into a capacitance current, which represents the change in ion density at the inner and outer surfaces of the membrane, and an ionic current, which represents the movement of charged particles across the membrane. Thus a circuit diagram for a "flux capacitor" membrane may be employed.

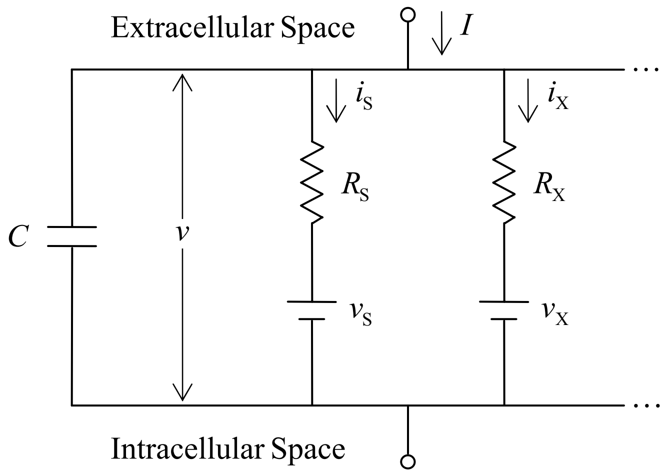


Figure 2.6: Circuit Representation of the Membrane – The membrane capacitance is in parallel with resistor/potential source series combinations for each permeable ion.

The conservation of current at the intracellular node yields,

$$I = C\frac{dv}{dt} + \sum i \tag{2.1.20}$$

where C, the membrane capacitance, is a measurable material property and assumed to be constant. The specific capacitance of biological membranes is about $1\,\mu\text{F}/\text{cm}^2$ [44] and ranges from 0.3 to $1.5\,\mu\text{F}/\text{cm}^2$ for black lipid membranes (BLMs) depending on the presence of organic solvent between the phospholipid tails [23]. In biological systems, the membrane current quickly approaches zero as a membrane potential is established and charged particles are drawn towards the membrane surfaces. Thus, the membrane potential can be determined by,

$$\frac{dv}{dt} = -\frac{1}{C}\sum i \tag{2.1.21}$$

This approach is employed in the nastic material model; the validity of this approach is tested as part of the model validation process. Other models have been developed specifically for determining the membrane potential based on the physics of the membrane-solution interface. A review of electrolytics, dielectrics, and the electrical properties of tissue is given by Grimnes and Martinsen [48]. This information forms the basis of the occurrence of membrane potential and the corresponding diffuse double layers. One of the simplest and most widely used models for the diffuse double layer is the Gouy-Chapman model [26, 48, 49, 50], which is an electrostatic model that assumes the ionic distribution is governed solely by Coulombic forces. It does a reasonably good job of predicting the ion distribution near a lipid bilayer. The Gouy-Chapman model is strictly valid in the case of an infinitely dilute bulk solution with a structureless solvent and an infinitely narrow membrane-solution interface. Several corrections have been developed that extend the Gouy-Chapman model to include ion-ion interactions and hydration effects as well as other interfacial effects [47]. A few nonequilibrium models have been developed [51, 52, 53] that approach the problem from the standpoint of electrodiffusion and typically involve some method of applying the simultaneous solution of the Poisson and Nernst-Planck equations.

2.1.7 Electrodiffusion across a Lipid Bilayer Membrane

The membrane materials currently being considered for the nastic material are permeable to small ions. To account for this passive transport of ions across the membrane, the one-dimensional Nernst-Planck equation described in Section 2.1.2 is employed,

$$J_{\mathrm{S}} = -\left(D_{\mathrm{S}} \frac{dc_{\mathrm{S}}}{dx} + u_{\mathrm{S}} z_{\mathrm{S}} F c_{\mathrm{S}} \frac{d\phi}{dx} \right) \tag{2.1.22}$$

where J_{S} is the flux of ion S per unit area of membrane, D_{S} is the diffusion constant for ion S, c_{S} is the concentration of S within the membrane as a function of x for $0 < x < \delta$, δ is the membrane thickness, and $\phi = \phi(x)$ is the electric potential within the membrane. The Nernst-Planck equation assumes that there is no convective flow and that the flow is not affected by other flows or forces. The solution of the equation requires that the functions $c_{\mathrm{S}}(x)$ and $\phi(x)$ must either be known or assumed. Since it is extremely difficult to determine the functions either experimentally or analytically in biological systems, their form is assumed based on assumptions about the membrane material, such as homogeneity. The electric field, and hence the gradient of the electric potential, is typically assumed to be constant across the membrane. It is also commonly assumed that the species concentrations at both surfaces of the membrane are related to the species concentrations in the corresponding medium by,

$$c_{\mathrm{S}}\big|_{x=0} = \beta_{\mathrm{S}} [\mathrm{S}]_{\mathrm{e}} \tag{2.1.23}$$

$$c_{\mathrm{S}}\big|_{x=\delta} = \beta_{\mathrm{S}}[\mathrm{S}]_{\mathrm{i}} \tag{2.1.24}$$

where the subscripts 'e' and 'i' refer to the extracellular and intracellular spaces respectively, β_{S} is called the partition coefficient and gives the relative solubility of ion S in the membrane versus that in the adjacent solution. The partition coefficients, β_{S}, are assumed to be independent of concentration. The flux equation in this case is known as the Goldman equation. For purposes of this analysis, it will be assumed that they are equal. The Nernst-Planck equation can be integrated with respect to x using these assumptions to yield [54],

$$J_{\mathrm{S}} = -\frac{\beta_{\mathrm{S}} u_{\mathrm{S}} z_{\mathrm{S}} F v}{\delta}\left[\frac{[\mathrm{S}]_{\mathrm{i}} - [\mathrm{S}]_{\mathrm{e}} \exp\left(-\frac{z_{\mathrm{S}} F v}{RT}\right)}{1-\exp\left(-\frac{z_{\mathrm{S}} F v}{RT}\right)}\right] \tag{2.1.25}$$

where $v = \phi(\delta) - \phi(0)$ is the membrane potential. This equation is often referred to as the Goldman-Hodgkin-Katz (GHK) equation. The above equation can also be expressed as,

$$J_{\mathrm{S}} = -P_{\mathrm{S}} \frac{z_{\mathrm{S}} F v}{RT}\left[\frac{[\mathrm{S}]_{\mathrm{i}} - [\mathrm{S}]_{\mathrm{e}} \exp\left(-\frac{z_{\mathrm{S}} F v}{RT}\right)}{1-\exp\left(-\frac{z_{\mathrm{S}} F v}{RT}\right)}\right] \tag{2.1.26}$$

where P_{S} is the permeability coefficient defined as,

$$P_{\mathrm{S}} = \frac{\beta_{\mathrm{S}}}{\delta} D_{\mathrm{S}} \tag{2.1.27}$$

A separate GHK equation will need to be employed for each permeable species in the system.

2.2 REVIEW OF APPLIED TRANSPORT MODELS

The foregoing model components can be combined to predict system response. A physics-based model must include (i) kinetics models for the various biological transport structures and the osmotic diffusion of solvent, (ii) a method of tracking species concentration and membrane potential, and (iii) a means of determining the hydrostatic pressure within the inclusion due to matrix constraint. Several models have been developed to simulate excitable cells which include kinetics models for biological transport structures and methods for tracking species concentration and membrane potential.

The earliest nerve excitation model developed in a series of papers written by A. L. Hodgkin and A. F. Huxley in 1952 [45, 46, 55, 56, 57]. In this series of papers, Hodgkin and Huxley perform extensive experimental work on the squid giant axon using voltage-clamp methods and derive a set of empirical nonlinear ordinary differential equations to model the data. In essence this approach adopts the membrane-as-capacitor approach discussed in Section 2.1.6 with the values of the resistances determined by various first-order reaction schemes. Since its introduction, several models have been developed to extend the Hodgkin-Huxley model and adapt it to other excitable cells such as those that make up the sinoatrial node [58]. This class of models, now collectively known as *Hodgkin-Huxley models*, for the most part differs, only in the number and kind of membrane current components [59]. One notable model is that of Wilders [59]. It combines features from several other models to create a comprehensive nerve excitation model. The Wilders model is based primarily on the DiFrancesco-Noble model [60] which takes into account variations in ion concentrations.

Several analyses have been performed on the mathematical aspects of Hodgkin-Huxley type models. A thorough resource outlining stability analysis as well as many other aspects of the Hodgkin-Huxley equations is given by Cronin [61]. Cronin gives an overview of the numerical studies that have been performed on the Hodgkin-Huxley equations and details several of the mathematical phenomena that occur under certain conditions. Hodgkin-Huxley type models require the integration of a stiff nonlinear system of ordinary differential equations. Much attention has been paid to the numerical solution of Hodgkin-Huxley models including the search for simple but efficient integration algorithms [62] and the comparison of several numerical integration schemes by Moore and Ramon [63] and by Victorri *et al.* [64]. A review of numerical solution techniques is provided in Section 2.4.

Moreover, several books and review papers have been written on the subject of biological transport. Notable summary works that cover diffusion, electrochemical potential and the thermodynamics of membrane transport are Schultz [27], Friedman [30], and Byrne and Schultz [65]. These books provide detailed derivations of several of the basic thermodynamic expressions used in biotransport analysis as well as discussion of the underlying physics. Of additional relevance, because the exact type and number of transporters present in a membrane are generally not known, methods have been developed to characterize and analyze the kinetics of the transport processes from experimental data. A discussion of these methods is presented by Stein [66] who illustrates how to use experimental data to determine the presence of various transport systems including simple diffusion, the presence of cooperating and competing species, and the existence of parallel transport pathways.

One of the more recent and most complete of the biotransport models is that of Endresen, *et al.* [67, 68, 69]. This model takes into account the contributions of ion pumps, channels, and exchangers as well as tracks species concentration and membrane potential. Because the approach of the nastic effort is based in part on the model of Endresen *et al.* [67], additional detail of that work is provided. The model relies upon basic physical principles as considered through empirical laws, such as Fick's law, Ohm's law, and the Nernst-Planck equation, and statistical thermodynamics, such as Boltzmann distribution, Markov assumption, and Onsager's principle of detailed balance, while applying the following simplifying assumptions: (1) all currents result in free ions within the cell, i.e. no intracellular binding, and (2) the electrical activity is influenced only by the motion of cations; anion concentrations are assumed to remain equal on both sides of the membrane.

The transport system consists of a lipid bilayer membrane that may contain ion pumps, ion exchangers, ion cotransporters, and/or ion channels (Figure 2.7); the system is modeled through the simultaneous solution of the governing equations of each of these components.

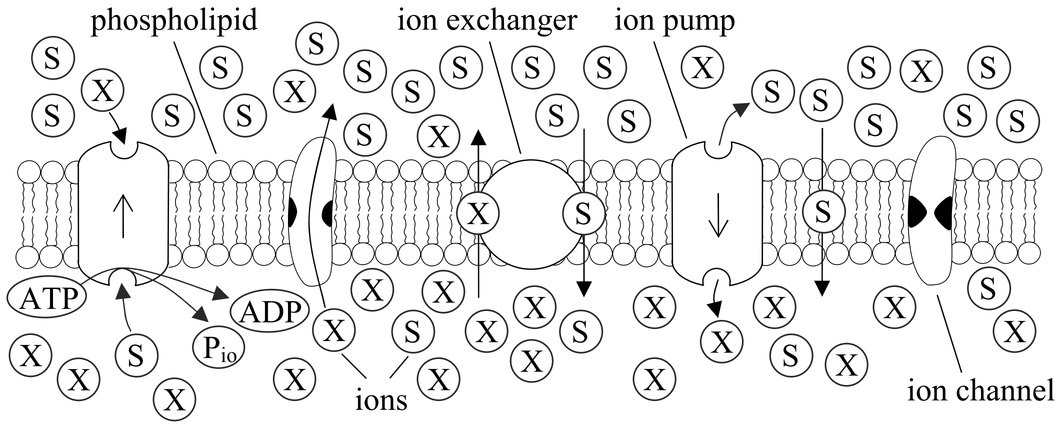


Figure 2.7: Transport Components – Membrane components including ion pumps, ion exchangers, and ion channels.

2.2.1 Ion Channels

Ion channels are modeled using the Nernst-Planck equation with channel gating modeled as a simple Markov process for first order kinetics; the gating process is stochastic with the probability of entering a certain state depending only on the last state occupied. The channels are assumed to randomly fluctuate between the completely open and the completely closed states. Ion channel gating is influenced by concentration gradient and membrane potential as well as the particular gating mechanism for the channel [35]. The ion channel current for a generic ion S, where the channel is governed by an activation mechanism alone, is determined by,

$$i_{\mathrm{S}} = k_{\mathrm{S}} x \sinh\left(\frac{z_{\mathrm{S}} e(v - v_{\mathrm{S}})}{2kT}\right) \tag{2.2.1}$$

with,

$$k_{\mathrm{S}} = 2 z_{\mathrm{S}} e u_{\mathrm{S}} kT \sqrt{[\mathrm{S}]_{\mathrm{e}} [\mathrm{S}]_{\mathrm{i}}} \frac{A_{\mathrm{p}}}{\varepsilon d} \tag{2.2.2}$$

where $x = x(t)$ is the probability that the channel is open, z_{S} is the valence charge of ion S, $e = 1.60217733 \times 10^{-19}\ \mathrm{C}$ is the charge of an electron, *v* is the membrane potential determined by Eq. (2.1.21), v_{S} is the Nernst equilibrium potential, $k = 1.380658 \times 10^{-23}\ \mathrm{J/K}$ is Boltzmann's constant, *T* is the absolute temperature, u_{S} is the ion mobility through the channel, $[\mathrm{S}]_{\mathrm{e}}$ is the concentration of ion S outside the inclusion, $[\mathrm{S}]_{\mathrm{i}} = n_{\mathrm{S}}/V(t)$ is the concentration of ion S inside the inclusion at time *t*, n_{S} is the number of moles of ion S inside the inclusion at time *t*, A_{p} is the effective pore area of the ion channel, *d* is the total channel length, and εd is the effective pore length of the ion channel where $0 \le \varepsilon \le 1$. The above equation assumes that the ion channel has a constant area A_0 except for a short narrow pore in its middle with an area A_{p} much smaller than the channel area such that A_{p}/A_0 is of the order ε^2 (Figure 2.8). This is typical for many ion channels.

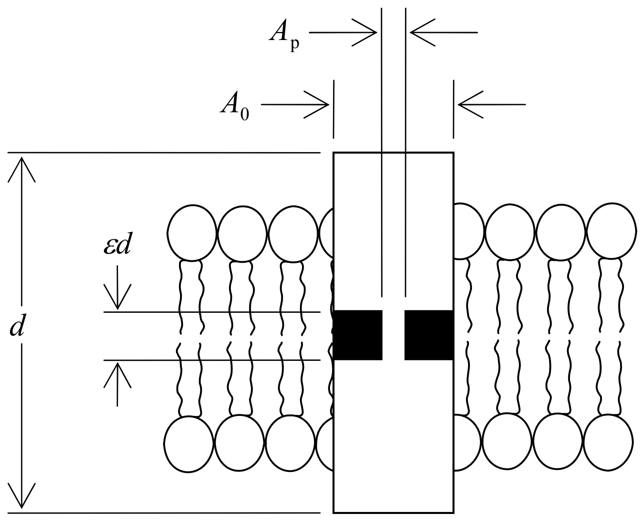


Figure 2.8: Ion Channel Geometry – Assumed geometry of an ion channel.

Also it is assumed that the electric field is constant everywhere in the channel and that the current must be uniform through all cross sections. The Nernst equilibrium potential, which represents the membrane potential required for the flux of a given ion S to be zero, is given by,

$$v_\mathrm{S} = \frac{kT}{z_\mathrm{S} e} \ln \frac{[\mathrm{S}]_\mathrm{e}}{[\mathrm{S}]_\mathrm{i}} \tag{2.2.3}$$

The probability that a channel is open, *x*, is determined by solving the differential equation,

$$\frac{dx}{dt} = \frac{1}{\tau_\mathrm{S}} \cosh\left(\frac{2e(v - v_x)}{kT}\right)\left\{\frac{1}{2}\left[1 + \tanh\left(\frac{2e(v - v_x)}{kT}\right)\right] - x\right\} \tag{2.2.4}$$

where τ_S is the ion channel relaxation time, which represents the time required for the channel to either open or close. This expression assumes that the energy difference between the open and closed states is given by,

$$\Delta G = q(v_x - v) \tag{2.2.5}$$

where q is the gating charge, qv is the change in electrical potential energy with charge redistribution, and qv_x is the change in mechanical conformational energy between the open and closed states. Based upon experimental observations for Calcium, Sodium, and Potassium channels, it is assumed that $q \approx \pm 4e$ [67].

Some channels have both an activation and an inactivation mechanism. Typically the activation mechanism is relatively fast compared to the inactivation mechanism and allows the channel to fluctuate between the permeable and impermeable states in response to membrane potential. The inactivation mechanism closes the channel during depolarization and will prevent the channel from becoming permeable until the membrane is repolarized [35]. Therefore the kinetics model for channels with an inactivation mechanism must take into account that the activation mechanism must be switched on *and* the inactivation mechanism must be switched off for the channel to be permeable. Endresen *et al.* [67] illustrates the case where the activation mechanism is very fast compared to the inactivation mechanism,

$$i_{\mathrm{S}} = k_{\mathrm{S}} f d_{\infty} \sinh\left(\frac{z_{\mathrm{S}} e(v - v_{\mathrm{S}})}{2kT}\right) \tag{2.2.6}$$

$$d_{\infty} = \frac{1}{2}\left[1 + \tanh\left(\frac{2e(v - v_d)}{kT}\right)\right] \tag{2.2.7}$$

$$\frac{df}{dt} = \frac{1}{\tau_{\mathrm{S}}}\cosh\left(\frac{2e(v - v_f)}{kT}\right)\left\{\frac{1}{2}\left[1 - \tanh\left(\frac{2e(v - v_f)}{kT}\right)\right] - f\right\} \tag{2.2.8}$$

where f is the probability of inactivation, d_{∞} represents the probability of activation, v_d is the half-activation potential, v_f is the half-inactivation potential.

2.2.2 Ion Pumps

Ion pumps are modeled by determining the net reaction rate of the pumps from available ATP free energy, membrane potential and equilibrium potentials of the transported ions. For a generic ion pump with forward and backward reaction rates α and β,

$$\mathrm{ATP} + m\mathrm{S}_{\mathrm{i}}^{z_{\mathrm{S}}} + n\mathrm{X}_{\mathrm{e}}^{z_{\mathrm{X}}} \underset{\beta}{\overset{\alpha}{\rightleftharpoons}} \mathrm{ADP} + \mathrm{P}_{\mathrm{io}} + m\mathrm{S}_{\mathrm{e}}^{z_{\mathrm{S}}} + n\mathrm{X}_{\mathrm{i}}^{z_{\mathrm{X}}} \tag{2.2.9}$$

that moves *m*-S ions out of the cell and *n*-X ions into the cell in a single cycle, the pump current is given by,

$$i_{\mathrm{pump}} = k_{\mathrm{pump}} \tanh\left(\frac{e\left[(mz_{\mathrm{S}} - nz_{\mathrm{X}})v + nz_{\mathrm{X}}v_{\mathrm{X}} - mz_{\mathrm{S}}v_{\mathrm{S}}\right] - \Delta G_{\mathrm{ATP}}}{2kT}\right) \tag{2.2.10}$$

with,

$$k_{\mathrm{pump}} = (mz_{\mathrm{S}} - nz_{\mathrm{X}})Me\lambda \tag{2.2.11}$$

where k_{pump} is a measure of the maximum pump current, M is the number of pumps, λ is the net pump rate in cycles/sec, and ΔG_{ATP} is the free energy released by the hydrolysis of ATP. The above expressions assume that the single pump reaction is a complete macroscopic description of the pump reaction. See Appendix Section A.1 for derivation based on that given in Endresen *et al.* [67].

2.2.3 Ion Exchangers

Ion exchangers are modeled in a similar manner as the ion pumps by determining their net reaction rate. For a generic ideal exchanger with the reaction,

$$a\mathrm{S}_{\mathrm{e}}^{z_{\mathrm{S}}} + b\mathrm{X}_{\mathrm{i}}^{z_{\mathrm{X}}} \underset{\beta}{\overset{\alpha}{\rightleftharpoons}} a\mathrm{S}_{\mathrm{i}}^{z_{\mathrm{S}}} + b\mathrm{X}_{\mathrm{e}}^{z_{\mathrm{X}}} \tag{2.2.12}$$

that moves *a*-S ions into the cell and *b*-X ions out of the cell in a single cycle, the current is given by,

$$i_{\mathrm{exchanger}} = k_{\mathrm{SX}} \sinh\left(\frac{e\left[(bz_{\mathrm{X}} - az_{\mathrm{S}})v + az_{\mathrm{S}}v_{\mathrm{S}} - bz_{\mathrm{X}}v_{\mathrm{X}}\right]}{2kT} \right) \tag{2.2.13}$$

with,

$$k_{\mathrm{SX}} = 2(bz_{\mathrm{X}} - az_{\mathrm{S}})Ne\lambda\sqrt{[\mathrm{S}]_{\mathrm{e}}^{a}[\mathrm{S}]_{\mathrm{i}}^{a}[\mathrm{X}]_{\mathrm{e}}^{b}[\mathrm{X}]_{\mathrm{i}}^{b}} \tag{2.2.14}$$

where N is the number of exchangers and $\lambda = \alpha + \beta = \text{const.}$ is the sum of the forward and backward reaction rates. The above equations assume that there is a rigid stiochiometry between the transported ions and that either ion can activate the transporter, i.e. initiate the cotransporter reaction. See Appendix Section A.2 for derivation based on that given in Endresen *et al.* [67].

2.3 THE MOONEY-RIVLIN MATRIX MODEL

There are a number of strain energy approaches available to model matrix response. These include the Arruda-Boyce model, the Marlow model, the neo-Hookean model, the Ogden model, the Yeoh model, and the Mooney-Rivlin model. A modified version of the Mooney-Rivlin model for nearly incompressible materials is applied to capture the finite strain response of the surrounding polymer matrix. This model is employed because it is easily related to experimentally measurable material properties under varying loading conditions, and like the transport model, it is quasicontinuum in nature with some basis in statistical thermodynamics. The modified Mooney-Rivlin form of the strain energy potential is [70],

$$U = C_{10}\left(\bar{\lambda}_1^2 + \bar{\lambda}_2^2 + \bar{\lambda}_3^2 - 3\right) + C_{01}\left(\bar{\lambda}_1^{-2} + \bar{\lambda}_2^{-2} + \bar{\lambda}_3^{-2} - 3\right) + \frac{1}{D_1}\left(J_{\mathrm{el}} - 1\right)^2 \tag{2.3.1}$$

where U is the strain energy per unit reference volume and the deviatoric stretches, $\bar{\lambda}_i$, are determined by,

$$\bar{\lambda}_i = J^{-1/3}\lambda_i \tag{2.3.2}$$

where J is the Jacobian and λ_i are the principle stretches which are defined by,

$$\lambda_i = \sqrt{C^{(i)}}\,, \qquad i = 1, 2, 3 \tag{2.3.3}$$

where $C^{(1)} \geq C^{(2)} \geq C^{(3)}$ are the eigenvalues of the right Cauchy-Green strain tensor, **C**. The first two terms of Eq. (2.3.1) are the classic Mooney-Rivlin form of the strain energy potential and represent the isochoric strain energy. The third term of the strain energy potential equation represents the volumetric strain energy. Thus, the strain energy equation assumes that the isochoric and volumetric strain energy can be decoupled [71]. There are several different forms that the volumetric strain energy function can take [72]. These versions are accurate under different conditions. The version used in Eq. (2.3.1) is valid only for nearly incompressible materials. The coefficients C_{10} and C_{01} are related to the initial shear modulus, μ_0, by,

$$\mu_0 = 2\left(C_{10} + C_{01}\right) \tag{2.3.4}$$

For some rubber elastic materials the relative magnitudes of C_{10} and C_{01} vary according to the type of loading. Under equi-biaxial extension, which is approximately equivalent to uniaxial compression, the coefficient C_{01} is zero for these materials, while in all other loading cases the ratio C_{01}/C_{10} ranges from 0.3 to 1.0 [73]. Physically this corresponds to a material that is asymmetric in tension and compression. To address this, two Mooney-Rivlin models are combined: one for compression loading, in which case the coefficient C_{01} is zero, and one for tension loading where the coefficient C_{01} has some experimentally determined value. In the compression case when C_{01} is zero, the Mooney-Rivlin model takes on the neo-Hookean form. The coefficient D_1 determines the compressibility of the material and is related to the initial bulk modulus, K_0, through,

$$K_0 = \frac{2}{D_1} \tag{2.3.5}$$

The elastic volume ratio, J_{el}, is related to the linear thermal expansion strain, $\varepsilon_{\mathrm{th}}$, by [70],

$$J_{\mathrm{el}} = \frac{J}{\left(1+\varepsilon_{\mathrm{th}}\right)^3} \tag{2.3.6}$$

If $D_1 = 0$, then the material is incompressible. Note that as $D_1 \to 0$ the deformation becomes isochoric, hence $J_{\mathrm{el}} \to 1$, which for isothermal deformation becomes $J \to 1$. Because *J*, the Jacobian, approaches 1 at the same rate as $D_1 \to 0$, $\left(J_{\mathrm{el}}-1\right)^2 / D_1 \to 0$. All three material parameters C_{10}, C_{01}, and D_1 are temperature dependent.

2.4 ODE SOLVERS

Transport model development must include a means to simultaneously solve a large set of Ordinary Differential Equations (ODEs). One of the primary causes of instability and high computation time is a property called stiffness. Stiffness can be roughly defined as the presence of one or more fast decay processes in time, with a time constant that is short compared to the time span of interest [74]. The level of stiffness of the system is an important factor in determining the numerical method to employ. Although it is not intended for the integration of stiff systems the Explicit Runge-Kutta method is effective in integrating the transport model at low power levels. It is also one of the simplest and easiest to code ODE solvers. An overview of various formulations of the Explicit Runge-Kutta method as well as source codes is given by Hairer *et al.* [75].

The detection and analysis of stiff systems as well as stability analysis for several numerical algorithms is presented in Hairer and Wanner [76]. Several algorithms have been developed specifically to integrate stiff systems [76]. An overview of many of the most widely used algorithms along with source codes is available in Press *et al.* [77]. Also, source codes for several highly sophisticated solvers are available as part of the Lawrence Livermore ODEPACK [74, 78, 79]. These solvers are designed to integrate both stiff and nonsitff systems and have many advanced features such as internal Jacobian calculation and the ability to determine and use the optimum order method during integration. Several numerical methods have been developed to integrate a system of ODEs of the form,

$$\dot{\mathbf{y}} \equiv \frac{d\mathbf{y}}{dt} = \mathbf{f}(t,\mathbf{y}) \tag{2.4.1}$$

The relative stability and computational efficiency of these methods is dependent upon the size and properties of $\mathbf{f}(t,\mathbf{y})$. Two methods that are found to be effective in solving the nastic model are the Explicit Runge-Kutta Method and the Backward Differentiation Formula (BDF) Method.

2.4.1 Explicit Runge-Kutta Method

The generic formulation of an *s*-stage autonomous Runge-Kutta method is [68],

$$\mathbf{k}_i = \mathbf{f}\left(\mathbf{y}_n + h\sum_{j=1}^{s} a_{ij}\mathbf{k}_j\right), \quad i = 1\cdots s \tag{2.4.2}$$

$$\mathbf{y}_{n+1} = \mathbf{y}_n + h\sum_{j=1}^{s} b_j\mathbf{k}_j\,, \quad \hat{\mathbf{y}}_{n+1} = \hat{\mathbf{y}}_n + h\sum_{j=1}^{s} \hat{b}_j\mathbf{k}_j \tag{2.4.3}$$

where h is the step size and the vectors $\mathbf{k}_j$ are function evaluations at each stage. The method is defined by the coefficients in the matrix $\mathbf{A} = \left[a_{ij}\right]$ and the vector $\mathbf{b} = \left[b_j\right]^{\mathrm{T}}$. For an explicit method, $\mathbf{A}$ is lower triangular. For non-autonomous systems, i.e. those having explicit time-dependent terms, there is an additional vector $\mathbf{c} = \left[c_j\right]^{\mathrm{T}}$ used to integrate the explicitly time-dependent terms. In this case, the integration step becomes,

$$\mathbf{k}_i = \mathbf{f}\left(t_n + c_i h, \mathbf{y}_n + h\sum_{j=1}^{s} a_{ij}\mathbf{k}_j\right), \quad i = 1\cdots s \tag{2.4.4}$$

These coefficients can be conveniently displayed in the Butcher array [68],

$$\begin{array}{c|c} \mathbf{c} & \mathbf{A} \\ \hline & \mathbf{b}^{\mathrm{T}} \\ \hline & \hat{\mathbf{b}}^{\mathrm{T}} \end{array} \quad \Leftrightarrow \quad \begin{array}{c|ccc} c_1 & a_{11} & \cdots & a_{1s} \\ \vdots & \vdots & \ddots & \vdots \\ c_s & a_{s1} & \cdots & a_{ss} \\ \hline & & b_1\cdots b_s & \\ \hline & & \hat{b}_1\cdots\hat{b}_s & \end{array} \tag{2.4.5}$$

The 5th Order Dorman and Prince (DOPRI5) formulation of the Explicit Runge-Kutta method is an embedded Runge-Kutta method which contains, besides the numerical approximation to the solution of the system of differential equations, $\mathbf{y}_1$, of order *p*, an expression $\hat{\mathbf{y}}_1$ of order

$q = p+1$ or $q = p-1$, depending on the formulation, which can be used for error and step size control [75]. In the case of DOPRI5, the term $\mathbf{y}_1$ is of order $p = 5$ and $\hat{\mathbf{y}}_1$ is of order $q = 4$. It is best suited for relative error tolerances between 10^{-4} and 10^{-7}. The DOPRI5 method is derived so as to minimize the error terms of the higher-order result. The lower order result is computed for the purposes of step size control. The Butcher array for DOPRI5 is [75],

$$\begin{array}{c|ccccccc}
0 & & & & & & & \\
\frac{1}{5} & \frac{1}{5} & & & & & & \\
\frac{3}{10} & \frac{3}{40} & \frac{9}{40} & & & & & \\
\frac{4}{5} & \frac{44}{45} & -\frac{56}{15} & \frac{32}{9} & & & & \\
\frac{8}{9} & \frac{19372}{6561} & -\frac{25360}{2187} & \frac{64448}{6561} & -\frac{212}{729} & & & \\
1 & \frac{9017}{3168} & -\frac{355}{33} & \frac{46732}{5247} & \frac{49}{176} & -\frac{5103}{18656} & & \\
1 & \frac{35}{384} & 0 & \frac{500}{1113} & \frac{125}{192} & -\frac{2187}{6784} & \frac{11}{84} & \\
\hline
y_1 & \frac{35}{384} & 0 & \frac{500}{1113} & \frac{125}{192} & -\frac{2187}{6784} & \frac{11}{84} & 0 \\
\hline
\hat{y}_1 & \frac{5179}{57600} & 0 & \frac{7571}{16695} & \frac{393}{640} & -\frac{92097}{339200} & \frac{187}{2100} & \frac{1}{40}
\end{array} \tag{2.4.6}$$

Note that since $a_{si} = b_i$, the last derivative evaluation can be re-used in the next time step.

2.4.2 Backward Differentiation Formula Methods

Backward Differentiation Formula (BDF), or predictor-corrector, methods are well suited for the integration of stiff systems. These methods are able to maintain stability for high levels of stiffness with a significantly larger time step than conventional methods. The basic integration step of a BDF method is defined as [74, 78],

$$\begin{aligned}
\mathbf{y}_n &= \sum_{i=1}^{q} \alpha_i \mathbf{y}_{n-i} + h\beta_0 \dot{\mathbf{y}}_n \\
&= \mathbf{a}_n + h\beta_0 \mathbf{f}(t_n, \mathbf{y}_n)
\end{aligned} \tag{2.4.7}$$

where q is the order of the method, h is the stepsize in time, and α_i and β_0 are integration coefficients. Because the above integration step is implicit in **y**, it yields a nonlinear system of algebraic equations that must be solved for the current values of **y** using some iterative method. Typically the modified Newton method is employed where [74, 78],

$$-\mathbf{P}\left[\mathbf{y}_{n(m+1)}-\mathbf{y}_{n(m)}\right]=\mathbf{y}_{n(m)}-\mathbf{a}_n-h\beta_0\mathbf{f}\left(t_n,\mathbf{y}_{n(m)}\right) \tag{2.4.8}$$

and **P** is an $n\times n$ matrix approximating the Jacobian, $\mathbf{J}\equiv\partial\mathbf{f}/\partial\mathbf{y}$, which is given by,

$$\mathbf{P}=\mathbf{I}-h\beta_0\mathbf{J} \tag{2.4.9}$$

where **I** is the identity matrix.

2.4.3 Error Estimation and Time Step Control

To control error propagation and maximize computational efficiency, most ODE solvers include an algorithm for estimating the integration error and varying the time step in order to keep it below a specified tolerance. The Root Mean Square (RMS) error is estimated using the following expression [75],

$$err=\sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\frac{y_i-\hat{y}_i}{\max\left(\varepsilon,\left|y_i\right|,\left|\hat{y}_i\right|\right)}\right)^2} \tag{2.4.10}$$

where N is the number of equations and ε is a minimum value for the denominator. The optimal step size for a desired tolerance, *tol*, is,

$$h\left(tol/err\right)^{1/(p+1)} \tag{2.4.11}$$

where p is the order of the method. For a stable code, this step size estimate should be multiplied by a safety factor, *fac*, with values of 0.8, 0.9, $0.25^{1/(p+1)}$, or $0.38^{1/(p+1)}$ [75]. The value of safety factor depends upon the code and the system of equations. It is chosen so that there is a high probability that the error in the next step will be acceptable. Also, the step size should not be allowed to increase or decrease too rapidly. This is accomplished by calculating the new stepsize in time, h_{new}, using,

$$h_{\text{new}}=h\min\left(facmx,\max\left(facmn,fac\left(tol/err\right)^{1/(p+1)}\right)\right) \tag{2.4.12}$$

where *facmx* and *facmn* are the maximum and minimum fractions of h that h_{new} can assume respectively. However, the above step size control will tend to cause the step size to oscillate. This is especially the case when the system of ODEs becomes stiff. To dampen this oscillation, a PI multiplier can be added to the step size formula as,

$$h_{\text{new}} = h \min\left(facmx, \max\left(facmn, fac\left(tol/|err_n|\right)^{\alpha}\left(|err_{n-1}|/tol\right)^{\beta}\right)\right) \tag{2.4.13}$$

This multiplier is derived from control theory for a Proportional plus Integral plus Derivative (PID) controller. Optimum values of α and β are suggested by Hairer and Wanner [76] to be,

$$\alpha \approx 0.7/p+1, \quad \beta \approx 0.4/p+1 \tag{2.4.14}$$

This method will cause the step size to be somewhat conservative. A step is accepted if $err \leq tol$ else it is rejected and hence must be repeated. In either case h_{new} is the new value of the step size to be attempted. It is recommended that *facmx* be set to 1 for the step immediately following a step rejection.

3.0 MODEL DEVLOPMENT

The model presented in this chapter is more general than its current application and is capable of modeling a wide range of biological transport components and inclusion configurations. While not being considered in this case, thermal effects can also be incorporated into the model. Information on thermal effects and their incorporation into the model is provided in Section 7.5. The model is based upon the coupling of a biological transport model of the inclusion and a finite element analysis of the surrounding matrix material. The model is subdivided into 3 distinctive components as illustrated in Figure 3.1. Further, Figure 3.1 provides an encapsulated view of the structure of this chapter.

The components of the following modeling effort are (1) the transport model for ion/solvent flux into/out of the inclusion, (2) the hyperelastic model of the polymer matrix, and (3) the coupling of the hyperelastic model with the transport model. The resulting model has predictive capability for both existing and hypothetical system configurations. This functionality allows the model to act as a fabrication feedback loop. Once the material has been optimized, the model may be used to generate loading curves for the development of less computationally intensive, structural length scale models. Parameters of interest at all stages of model development include blocked force and free displacement of the combined system as well as internal parameters such as the hydrostatic pressure and volume of the inclusion fluid resulting from the system loading.

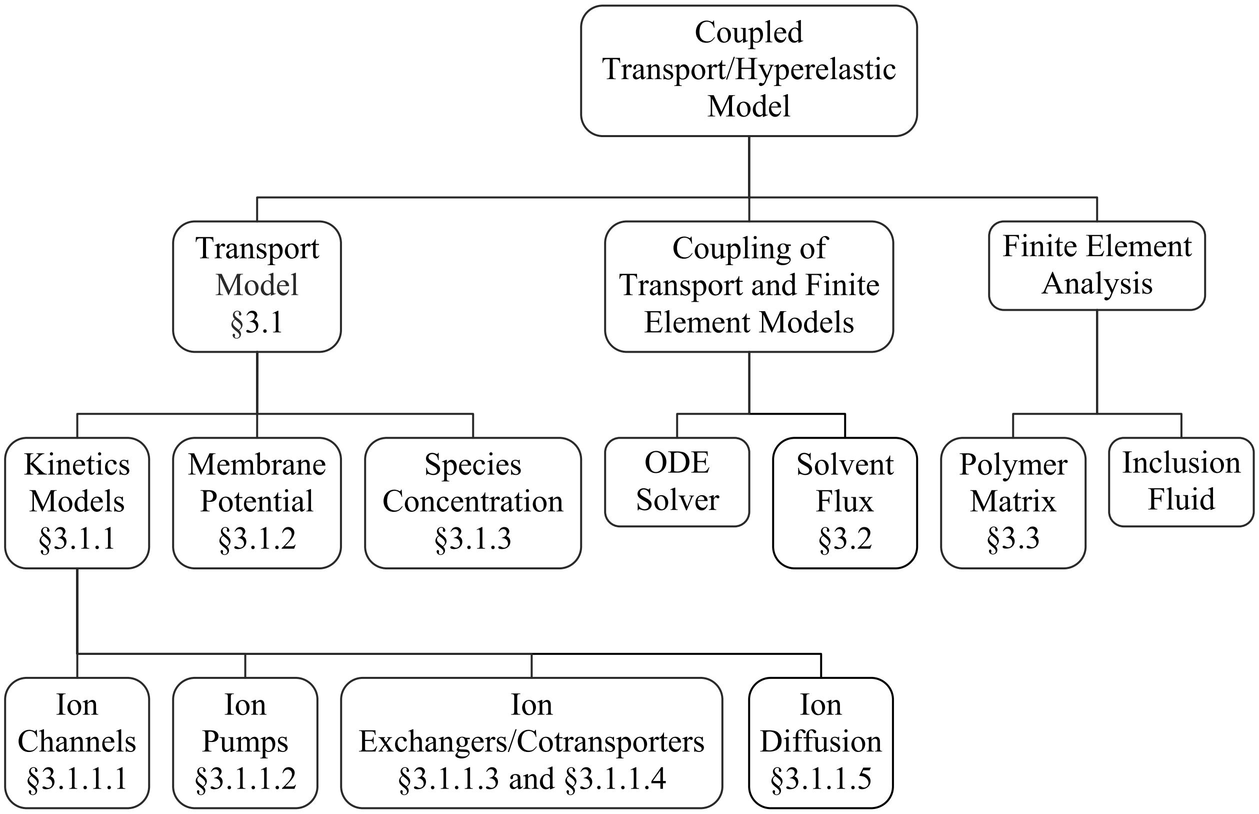

Coupled Transport/Hyperelastic Model
Transport Model §3.1
Coupling of Transport and Finite Element Models
Finite Element Analysis
Kinetics Models §3.1.1
Membrane Potential §3.1.2
Species Concentration §3.1.3
ODE Solver
Solvent Flux §3.2
Polymer Matrix §3.3
Inclusion Fluid
Ion Channels §3.1.1.1
Ion Pumps §3.1.1.2
Ion Exchangers/Cotransporters §3.1.1.3 and §3.1.1.4
Ion Diffusion §3.1.1.5

Figure 3.1: Model Organization

3.1 TRANSPORT MODEL

3.1.1 Kinetics Models

The transport model presented here is based upon the works of Endresen *et al.* [67, 68, 69]. These works provide a means to model the membrane potential of a single living cell and are readily adapted to a vesicle, or engineered cell. The following is a discussion of the adaptation of this model to a nastic material.

The transport system consists of a lipid bilayer membrane that may contain ion pumps, ion exchangers, ion cotransporters, and/or ion channels (Figure 2.7); the system is modeled through the simultaneous solution of the governing equations of each of these components. Pumps, channels, and exchangers are modeled using the methods employed by Endresen *et al.*

[67] and discussed in Section 2.2. Unlike the Endresen focus, however, the assumption of negligible hydrostatic pressure gradient is not reasonable. Thus, the biological transport model must be modified to account for the hydrostatic pressure gradient. The hydrostatic pressure gradient is introduced into the biological transport model through the general expression for the electrochemical potential [28],

$$\mu_{\mathrm{S}} = \left(\frac{\partial G}{\partial n_{\mathrm{S}}}\right)_{\substack{T,p,n_{\mathrm{B}} \\ \mathrm{B}\neq\mathrm{S}}} = \mu_{\mathrm{S}}^{0} + RT\ln[\mathrm{S}] + z_{\mathrm{S}}Fv + \overline{V}_{\mathrm{S}}p + m_{\mathrm{S}}gh \tag{3.1.1}$$

where μ_{S}^{0} is the electrochemical potential at standard state, $\overline{V}_{\mathrm{S}}$ is the partial molal volume of S, g is the local gravitational acceleration, and h is the height above sea level. For a membrane separating two solutions at equilibrium, the electrochemical potentials are equal as are the standard electrochemical potentials. Thus,

$$RT\ln[\mathrm{S}]_{\mathrm{e}} + z_{\mathrm{S}}Fv_{\mathrm{e}} + \overline{V}_{\mathrm{S}}p_{\mathrm{e}} = RT\ln[\mathrm{S}]_{\mathrm{i}} + z_{\mathrm{S}}Fv_{\mathrm{i}} + \overline{V}_{\mathrm{S}}p_{\mathrm{i}} \tag{3.1.2}$$

Rearranging the above equation yields the equilibrium potential,

$$\begin{aligned} v_{\mathrm{S}} = v_{\mathrm{i}} - v_{\mathrm{e}} &= \frac{RT}{z_{\mathrm{S}}F}\ln\frac{[\mathrm{S}]_{\mathrm{e}}}{[\mathrm{S}]_{\mathrm{i}}} + \frac{\overline{V}_{\mathrm{S}}}{z_{\mathrm{S}}F}(p_{\mathrm{e}} - p_{\mathrm{i}}) \\ &= \frac{kT}{z_{\mathrm{S}}e}\ln\frac{[\mathrm{S}]_{\mathrm{e}}}{[\mathrm{S}]_{\mathrm{i}}} - \frac{\overline{V}_{\mathrm{S}}}{z_{\mathrm{S}}F}p \end{aligned} \tag{3.1.3}$$

The above expression replaces the Nernst equilibrium potential, which is applied in the modeling of each of the transport structures. Hydrostatic pressure effects on the currents generated by the transport components are taken into account by substituting the equilibrium potential given by Eq. (3.1.3) into the current equations in place of the Nernst potential. In the case of H^+, the volume of a proton is extremely small compared with most other molecules and hence hydrostatic pressure effects can be ignored. The partial molal volume of sucrose in water on an unhydrated basis is $\overline{V}_{\mathrm{sucrose}} = 0.000212\ \mathrm{m}^3/\mathrm{mol}$ [43].

3.1.1.1 Ion Channels

The model for ion channel current is a direct adaptation of Endresen's works as discussed in Section 2.2.1. The model summary of Section 3.4 restates these equations.

3.1.1.2 Ion Pumps

In order to implement the ion pump equations developed by Endresen (Section 2.2.2), the free energy release from the hydrolysis of ATP, ΔG_{ATP}, must be determined. The simplified reaction for this process, as noted in Section 2.1.4, is given by,

$$\mathrm{ATP} + \mathrm{H_2O} \rightleftharpoons \mathrm{ADP} + \mathrm{P_{io}} \tag{3.1.4}$$

The free energy released by this reaction can be stated mathematically by applying Eq. (2.1.13) to give,

$$\Delta G_{\mathrm{ATP}} = \Delta G^{\circ\prime}{}_{\mathrm{ATP}} + RT\ln\left(\frac{[\mathrm{ADP}][\mathrm{P_{io}}]}{[\mathrm{ATP}]}\mathrm{Molar}^{-1}\right) \tag{3.1.5}$$

where,

$$\Delta G^{\circ\prime}{}_{\mathrm{ATP}} = -RT\ln\left(\frac{[\mathrm{ADP}]^{\circ\prime}[\mathrm{P_{io}}]^{\circ\prime}}{[\mathrm{ATP}]^{\circ\prime}}\mathrm{Molar}^{-1}\right) \tag{3.1.6}$$

The above expressions are in terms of J/mol. Since the ATP free energy term in the pump equation is for the hydrolysis of a single ATP molecule, Eq. (3.1.5) must be modified to read,

$$\Delta G_{\mathrm{ATP}} = \frac{e}{F}\left(\Delta G^{\circ\prime}{}_{\mathrm{ATP}} + RT\ln\left(\frac{n_{\mathrm{ADP}}n_{\mathrm{P_{io}}}}{1000 n_{\mathrm{ATP}}V}\mathrm{m^3/mole}\right)\right) \tag{3.1.7}$$

where V is the inclusion volume in m^3 and the number of moles of each species in the inclusion are applied rather than their concentrations. The number of moles of ATP, ADP, and $\mathrm{P_{io}}$ are determined from the pump current using,

$$\frac{dn_{\mathrm{ATP}}}{dt} = \frac{-i_{\mathrm{pump}}}{(mz_{\mathrm{S}} - nz_{\mathrm{X}})F} \tag{3.1.8}$$

$$\frac{dn_{\mathrm{ADP}}}{dt} = \frac{i_{\mathrm{pump}}}{(mz_{\mathrm{S}} - nz_{\mathrm{X}})F} \tag{3.1.9}$$

$$\frac{dn_{\mathrm{P_{io}}}}{dt} = \frac{i_{\mathrm{pump}}}{(mz_{\mathrm{S}} - nz_{\mathrm{X}})F} \tag{3.1.10}$$

Since the three ODEs above are identical, the ADP and $\mathrm{P_{io}}$ ODEs can be replaced by,

$$n_{\mathrm{ADP}}(t) = n_{\mathrm{ADP}}(0) + n_{\mathrm{ATP}}(0) - n_{\mathrm{ATP}}(t) \tag{3.1.11}$$

$$n_{\mathrm{P_{io}}}(t) = n_{\mathrm{P_{io}}}(0) + n_{\mathrm{ATP}}(0) - n_{\mathrm{ATP}}(t) \tag{3.1.12}$$

A typical value for the apparent equilibrium constant and standard free energy change of the ATP hydrolysis reaction at a pH of 7 with 10 mM $\mathrm{Mg^{2+}}$ and 10 mM $\mathrm{P_{io}}$ present in the solution is

$K' = 10^5$ M and $\Delta G^{\circ\prime} = -28{,}000$ J/mol [39]. The actual standard free energy change for the ATP hydrolysis reaction is $\Delta G^{\circ\prime} = -30{,}500$ J/mol.

3.1.1.3 Ion Exchangers

The model for the ion exchangers is a direct adaptation of Endresen's works as works as discussed in Section 2.2.3. The model summary of Section 3.4 restates these equations.

3.1.1.4 H⁺/Sucrose Cotransport

The nastic material currently being developed will incorporate a secondary active transport mechanism for the cotransport of H^+, Sucrose, and water. The H^+/Sucrose cotransporter has the reaction,

$$\mathrm{H_e^+} + \mathrm{Suc_e} + n_\mathrm{W}\mathrm{H_2O_e} \underset{\beta}{\overset{\alpha}{\rightleftharpoons}} \mathrm{H_i^+} + \mathrm{Suc_i} + n_\mathrm{W}\mathrm{H_2O_i} \tag{3.1.13}$$

By applying the methodology of Endresen *et al.* [67] and information on determining how to relate the forward and backward reaction rates to the energy balance from Mullins [80, 81], it can be shown that the current is given by,

$$i_\mathrm{cotransporter} = k_\mathrm{cotransporter} \sinh\left(\frac{e\left(v - v_\mathrm{H} - v_\mathrm{Suc} - n_\mathrm{W} v_\mathrm{H_2O}\right)}{2kT}\right) \tag{3.1.14}$$

where,

$$k_\mathrm{cotransporter} = 2Ne\lambda\sqrt{[\mathrm{H^+}]_\mathrm{e}[\mathrm{H^+}]_\mathrm{i}[\mathrm{Suc}]_\mathrm{e}[\mathrm{Suc}]_\mathrm{i}[\mathrm{H_2O}]_\mathrm{e}^{n_\mathrm{W}}[\mathrm{H_2O}]_\mathrm{i}^{n_\mathrm{W}}} \tag{3.1.15}$$

and,

$$v_\mathrm{Suc} = \frac{kT}{e}\ln\frac{[\mathrm{Suc}]_\mathrm{e}}{[\mathrm{Suc}]_\mathrm{i}} \tag{3.1.16}$$

$$v_\mathrm{H_2O} = \frac{kT}{e}\ln\frac{[\mathrm{H_2O}]_\mathrm{e}}{[\mathrm{H_2O}]_\mathrm{i}} = \frac{\overline{V}_\mathrm{H_2O}}{F}\sum_\mathrm{S}\pi_\mathrm{S} \tag{3.1.17}$$

where $\overline{V}_\mathrm{H_2O}$ is the partial molal volume of water and $\sum\pi_\mathrm{S}$ is the total osmotic pressure difference across the membrane. A detailed derivation is provided for Eqs. (3.1.14) and (3.1.15) in Appendix Section A.3. Note that the "sucrose potential" and "water potential", v_Suc and $v_\mathrm{H_2O}$, given by Eqs. (3.1.16) and (3.1.17) respectively are used only to simplify the cotransporter current equation and, unlike the Nernst potential, have no physical significance since both sucrose and water are neutral. Also, note the similarity between the above expressions for H^+/Sucrose cotransport and the ion/ion exchanger equations in Section 2.2.3 due to the similar energetics in both cases; i.e. the only difference from an energetics standpoint is the change in sign of the electrochemical energy gradient of the sucrose. The sign of the cotransporter current in the concentration equations must reflect the fact there is influx of both H^+ and Sucrose. If either the H^+ or sucrose concentrations driving the cotransporter are large enough that saturation effects become significant, then the above cotransporter current equation can be replaced by,

$$i_{\text{cotransporter}} = \frac{NF\lambda\gamma K\left(\exp\left(\frac{ev}{2kT}\right)\left[\text{H}^+\right]_{\text{e}}\left[\text{Suc}\right]_{\text{e}} - \exp\left(-\frac{ev}{2kT}\right)\left[\text{H}^+\right]_{\text{i}}\left[\text{Suc}\right]_{\text{i}}\right)}{2 + K\left(\left[\text{H}^+\right]_{\text{e}} + \left[\text{H}^+\right]_{\text{i}}\right) + \gamma K\left[\left[\text{H}^+\right]_{\text{e}} - \left[\text{H}^+\right]_{\text{i}} + \lambda\left(\left[\text{H}^+\right]_{\text{e}}\left[\text{Suc}\right]_{\text{e}} - \left[\text{H}^+\right]_{\text{i}}\left[\text{Suc}\right]_{\text{i}}\right)\right]} \quad (3.1.18)$$

where K and γ determine the saturation effects for H^+ and sucrose respectively. A detailed derivation based on that of Mullins [80] is provided for the above equation in Appendix Section A.4.

The particular H^+/Sucrose cotransporter that is chosen for use in the current set of experiments is from the subfamily SUT4 [82, 83, 84]. Experimental data collected by members of the nastic team shows that water can pass through SUT4 [5, 85]. It has been shown that SUT4 facilitates the passive transport of water along its osmotic and hydrostatic pressure gradients. In addition, it has been shown that water is actively transported by SUT4 against an osmotic and hydrostatic pressure gradient. It will be assumed that the active transport of water occurs according to a fixed stoichiometry with respect to H^+/Sucrose cotransport and that it is independent of the passive flux. This is consistent with the behavior of several other families of cotransporters [40, 86].

3.1.1.5 Ion Diffusion

Ion diffusion is modeled using a modified version of the Goldman equation that takes into account the effects of hydrostatic pressure. The expression is developed by substituting the definition of the electrochemical potential, μ_{S}, (Eq. 2.1.1) and Einstein's diffusion relationship, $D_{\text{S}} = u_{\text{S}}RT$, into the one-dimensional Nernst-Planck equation (Eq. 2.1.4). Ignoring gravitational effects, the result is,

$$J_{\mathrm{S}} = -D_{\mathrm{S}}\frac{dc_{\mathrm{S}}}{dx} - u_{\mathrm{S}}c_{\mathrm{S}}z_{\mathrm{S}}F\frac{d\phi}{dx} - u_{\mathrm{S}}c_{\mathrm{S}}\overline{V}_{\mathrm{S}}\frac{dp}{dx} \tag{3.1.19}$$

In addition to the assumptions about the linear forms of dc_{S}/dx and $d\phi/dx$ applied in Section 2.1.7, it will also be assumed that dp/dx is constant, i.e. the pressure varies linearly across the membrane. Integrating Eq. (3.1.19) across the membrane thickness and converting to a current yields,

$$i_{\mathrm{diff}} = z_{\mathrm{S}}FAJ_{\mathrm{S}} = -P_{\mathrm{S}}A\frac{z_{\mathrm{S}}F\left(z_{\mathrm{S}}Fv + \overline{V}_{\mathrm{S}}p\right)}{RT}\left[\frac{[\mathrm{S}]_{\mathrm{i}} - [\mathrm{S}]_{\mathrm{e}}\exp\left(-\frac{z_{\mathrm{S}}Fv + \overline{V}_{\mathrm{S}}p}{RT}\right)}{1 - \exp\left(-\frac{z_{\mathrm{S}}Fv + \overline{V}_{\mathrm{S}}p}{RT}\right)}\right] \tag{3.1.20}$$

where A is the area over which diffusion is occurring, $v = \phi(\delta) - \phi(0)$ is the membrane potential, and P_{S} is the permeability coefficient defined as,

$$P_{\mathrm{S}} = \frac{\beta_{\mathrm{S}}}{\delta}D_{\mathrm{S}} \tag{3.1.21}$$

where β_{S} is the partition coefficient, δ is the membrane thickness, and D_{S} is the diffusion constant for ion S. Note that as $\left(z_{\mathrm{S}}Fv + \overline{V}_{\mathrm{S}}p\right) \rightarrow 0$,

$$i_{\mathrm{diff}} \rightarrow P_{\mathrm{S}}Az_{\mathrm{S}}F\left([\mathrm{S}]_{\mathrm{e}} - [\mathrm{S}]_{\mathrm{i}}\right) \tag{3.1.22}$$

3.1.2 Membrane Potential

The membrane potential is determined using the Hodgkin-Huxley equation outlined in Section 2.1.6,

$$\frac{dv}{dt} = -\frac{1}{C}\sum i \tag{3.1.23}$$

The assumption of zero net membrane current is taken to hold. This assumption is not strictly valid, but does enable prediction of reasonable response trends. Further illustration and discussion of this point are provided in Chapters 5.0 and 7.0.

3.1.3 Species Concentration

The ion concentration within the inclusion is determined from the channel, pump, exchanger, and passive ion diffusion currents by,

$$\frac{d}{dt}[\mathrm{S}]_{\mathrm{i}} = \frac{-mz_{\mathrm{S}}i_{\mathrm{pump}}/(mz_{\mathrm{S}} - nz_{\mathrm{X}}) + az_{\mathrm{S}}i_{\mathrm{exchanger}}/(bz_{\mathrm{X}} - az_{\mathrm{S}}) + i_{\mathrm{cotransporter}} - i_{\mathrm{S}} - i_{\mathrm{diff}}}{z_{\mathrm{S}}FV} \qquad (3.1.24)$$

where *m* is the number of S ions pumped out of the cell by an ion pump cycle, *a* is the number of S ions pumped into the cell by an exchanger cycle, *F* is Faraday's constant, and $V = V(t)$ is the inclusion volume. This equation is simply the integral form of the conservation of mass that converts the transporter currents into mass flow rates. The signs of the pump and exchanger terms are determined by whether S is transported into or out of the cell in the transport reaction. Also, note that if the membrane does not have the same permeability to S as the solvent, then the number of moles of S will have to be tracked rather than the concentration, in which case the above equation becomes,

$$\frac{d}{dt}(n_{\mathrm{S}})_{\mathrm{i}} = \frac{-mz_{\mathrm{S}}i_{\mathrm{pump}}/(mz_{\mathrm{S}} - nz_{\mathrm{X}}) + az_{\mathrm{S}}i_{\mathrm{exchanger}}/(bz_{\mathrm{X}} - az_{\mathrm{S}}) + i_{\mathrm{cotransporter}} - i_{\mathrm{S}} - i_{\mathrm{diff}}}{z_{\mathrm{S}}F} \qquad (3.1.25)$$

where $(n_{\mathrm{S}})_{\mathrm{i}}$ is the number of moles of S in the intracellular space. The current model uses Eq. (3.1.25). Equation (3.1.25) is solved simultaneously with membrane potential equation, Eq. (3.1.23), and the governing equations for the transport components present in the membrane, as well as the solvent flux equation discussed in Section 3.2.

The above model assumes that all solutions are sufficiently dilute that they obey the perfect gas law, i.e. the individual particles do not interact with one another or those of other components [27],

$$[\mathrm{S}] = p_{\mathrm{S}}/RT \qquad (3.1.26)$$

where p_{S} is the partial pressure of S. In the case of protons, this expression holds because of their low concentration and small volume. However, in the case of other species, such as sucrose, concentrations in the nastic material are expected to be large enough that the perfect gas law will induce significant error. In order to take into account the effects of high concentration, activity will be used in place of concentration. Activity, which is the actual result from the perfect gas law, is given as,

$$a_{\mathrm{S}} = p_{\mathrm{S}}/RT \tag{3.1.27}$$

where activity is related to concentration by,

$$a_{\mathrm{S}} = \gamma_{\mathrm{S}}[\mathrm{S}] \tag{3.1.28}$$

and γ_{S} is an experimentally determined activity coefficient. The term $\gamma_{\mathrm{S}}[\mathrm{S}]$, the activity, is used in place of concentration, which is determined from the number of moles of S, in all transport equations in order to compensate for the nonideal behavior of the solute, S.

3.2 SOLVENT FLUX

The transport model is coupled to the hyperelastic model of the polymer matrix through the solvent transport equation. An overview of solvent (water) flux across biological membranes is given in Section 2.1.5. Solvent flux is modeled as a combination of osmotic diffusion and stoichiometric active transport,

$$\frac{dV}{dt} = J_{\mathrm{V}} A + \sum_{\mathrm{S}} \overline{V}_{\mathrm{S}} \frac{d}{dt}(n_{\mathrm{S}})_{\mathrm{i}} = \underbrace{KA\left(\sum_{\mathrm{S}} \sigma_{\mathrm{S}}\pi_{\mathrm{S}} - p\right)}_{\text{osmotic diffusion}} + \underbrace{\sum_{\mathrm{S}} \overline{V}_{\mathrm{S}} \frac{d}{dt}(n_{\mathrm{S}})_{\mathrm{i}}}_{\text{active transport}} \tag{3.2.1}$$

where K is the effective permeability of the membrane with respect to solvent taking into account the contributions from transporters, A is the effective surface area of the membrane over which solvent flux can occur, σ_{S} is an experimentally determined osmotic reflection coefficient for species S, π_{S} is the osmotic pressure difference across the membrane due to species S, p is the hydrostatic pressure difference across the membrane resulting from matrix constraint, $\overline{V}_{\mathrm{S}}$ is the partial molal volume of S, and $d(n_{\mathrm{S}})_{\mathrm{i}}/dt$ is given by Eq. (3.1.25). The first term in the above equation represents osmotic diffusion as defined by Eq. (2.1.15) and the second term represents stoichiometric active transport.

To account for the effects of non-ideal solution behavior, the van't Hoff equation for osmotic pressure, which is only valid at low concentrations, will be replaced by the more general expression [43],

$$\pi_{\mathrm{S}} = \frac{RT}{\overline{V}_{\mathrm{A}}} \frac{q_{\mathrm{S}} m_{\mathrm{S}} W_{\mathrm{A}}}{1000} \omega \tag{3.2.2}$$

where $\overline{V}_{\mathrm{A}}$ and W_{A} are the partial molar volume and molecular weight respectively of the solvent A, m_{S} is the molal concentration of S, q_{S} is the total number of moles of ions given by one mole of electrolyte and has a value of 1 for nonelectrolytes, and ω is an experimentally determined osmotic coefficient that takes into account the non-ideal behavior of the solute in the solution. The partial molal volume of water, $\overline{V}_{\mathrm{A}}$, has a value of $18.01\,\mathrm{ml/mol}$ for pure water but varies with solute concentration [43]. The osmotic coefficient is defined by the relation,

$$\ln a_{\mathrm{A}} = -\frac{q_{\mathrm{S}} m_{\mathrm{S}} W_{\mathrm{A}}}{1000}\omega \tag{3.2.3}$$

where a_{A} is the activity of the solvent in the solution as experimentally determined via Eq. (3.1.27).

The osmotic pressure difference across the membrane is determined using Eq. (3.2.2) as modified to read,

$$\pi_{\mathrm{S}} = \frac{RT}{\overline{V}_{\mathrm{A}}}\frac{q_{\mathrm{S}} W_{\mathrm{A}}}{1000}\left((m_{\mathrm{S}}\omega_{\mathrm{S}})_{\mathrm{i}} - (m_{\mathrm{S}}\omega_{\mathrm{S}})_{\mathrm{e}}\right) \tag{3.2.4}$$

where the subscripts 'i' and 'e' denote the intracellular and extracellular spaces respectively. The value of $\overline{V}_{\mathrm{A}}$ does not change appreciably enough over the range of concentrations encountered in the nastic material to cause significant error in the osmotic pressure calculation and hence will be assumed to be constant and that of pure solvent.

3.3 HYPERELASTIC MATRIX MODEL

The hyperelastic behavior of the matrix material is modeled using the Mooney-Rivlin form of the strain energy potential (Section 2.3) within a finite element analysis created using the ABAQUS/Standard 6.5 package. The Mooney-Rivlin form of the strain energy potential is [70],

$$U = C_{10}\left(\overline{\lambda}_1^{2} + \overline{\lambda}_2^{2} + \overline{\lambda}_3^{2} - 3\right) + C_{01}\left(\overline{\lambda}_1^{-2} + \overline{\lambda}_2^{-2} + \overline{\lambda}_3^{-2} - 3\right) + \frac{1}{D_1}\left(J_{\mathrm{el}} - 1\right)^2 \tag{3.3.1}$$

where U is the strain energy per unit reference volume, $\overline{\lambda}_i$ are the deviatoric stretches, J_{el} is the elastic volume ratio, and the coefficients C_{10} and C_{01} are determined by the type of loading. In the present effort, thermal effects are not considered causing $J_{\mathrm{el}} \to J$. Because this may ultimately prove to be a nontrivial contribution to material response, this term is included for completeness. The coefficients C_{10} and D_1 are determined from published data for common polymers. For simplicity, C_{01} is set to zero which corresponds to a state of either equi-biaxial tension or uniaxial compression. Although this is not actually the case for the cylindrical actuator, it will, for the most part, hold for the spherical inclusion, which is of primary interest in the current modeling effort.

3.4 MODEL SUMMARY

The model, as a whole, has three components (Figure 3.1); the hyperelastic model of the polymer matrix, the transport model, and the coupling of the preceding. The governing equation for the hyperelastic model is the Mooney-Rivlin form of the strain energy potential,

$$\begin{array}{c}\text{strain energy}\\\text{potential}\end{array}\left\{ U = C_{10}\left(\bar{\lambda}_1^2 + \bar{\lambda}_2^2 + \bar{\lambda}_3^2 - 3\right) + C_{01}\left(\bar{\lambda}_1^{-2} + \bar{\lambda}_2^{-2} + \bar{\lambda}_3^{-2} - 3\right) + \frac{1}{D_1}\left(J_{\mathrm{el}} - 1\right)^2 \right. \tag{3.4.1}$$

The transport and coupling equations form a set of first-order nonlinear ODEs. The transport equations consist of the governing equations for the transporters and membrane. They can be summarized as follows,

$$\begin{array}{c}\text{ion channels}\\\text{w/ activation}\end{array}\left\{\begin{array}{l} i_{\mathrm{S}} = k_{\mathrm{S}} x \sinh\left(\dfrac{z_{\mathrm{S}} e\left(v - v_{\mathrm{S}}\right)}{2kT}\right) \\ k_{\mathrm{S}} = 2 z_{\mathrm{S}} e u_{\mathrm{S}} kT \sqrt{[\mathrm{S}]_{\mathrm{e}}[\mathrm{S}]_{\mathrm{i}}}\,\dfrac{A_{\mathrm{p}}}{\varepsilon d} \end{array}\right. \tag{3.4.2}$$

$$\begin{array}{c}\text{ion channels}\\\text{w/ inactivation}\end{array}\left\{\begin{array}{l} i_{\mathrm{S}} = k_{\mathrm{S}} f d_{\infty} \sinh\left(\dfrac{z_{\mathrm{S}} e\left(v - v_{\mathrm{S}}\right)}{2kT}\right) \\ d_{\infty} = \dfrac{1}{2}\left[1 + \tanh\left(\dfrac{2e\left(v - v_d\right)}{kT}\right)\right] \\ \dfrac{df}{dt} = \dfrac{1}{\tau_{\mathrm{S}}}\cosh\left(\dfrac{2e\left(v - v_f\right)}{kT}\right)\left\{\dfrac{1}{2}\left[1 - \tanh\left(\dfrac{2e\left(v - v_f\right)}{kT}\right)\right] - f\right\} \end{array}\right. \tag{3.4.3}$$

$$\text{ion pumps}\begin{cases} i_{\text{pump}} = k_{\text{pump}} \tanh\left(\dfrac{e\left[\left(mz_{\text{S}} - nz_{\text{X}} \right)v + nz_{\text{X}}v_{\text{X}} - mz_{\text{S}}v_{\text{S}} \right] - \Delta G_{\text{ATP}}}{2kT} \right) \\ k_{\text{pump}} = \left(mz_{\text{S}} - nz_{\text{X}} \right) Me\lambda \\ \Delta G_{\text{ATP}} = \dfrac{e}{F}\left(\Delta G^{\circ\prime}_{\text{ATP}} + RT\ln\left(\dfrac{n_{\text{ADP}}n_{\text{P}_{\text{io}}}}{1000 n_{\text{ATP}}V}\,\text{m}^3/\text{mole} \right) \right) \\ \dfrac{dn_{\text{ATP}}}{dt} = \dfrac{-i_{\text{pump}}}{\left(mz_{\text{S}} - nz_{\text{X}} \right)F} \\ n_{\text{ADP}}(t) = n_{\text{ADP}}(0) + n_{\text{ATP}}(0) - n_{\text{ATP}}(t) \\ n_{\text{P}_{\text{io}}}(t) = n_{\text{P}_{\text{io}}}(0) + n_{\text{ATP}}(0) - n_{\text{ATP}}(t) \end{cases} \tag{3.4.4}$$

$$\text{ion exchangers}\begin{cases} i_{\text{exchanger}} = k_{\text{SX}} \sinh\left(\dfrac{e\left[\left(bz_{\text{X}} - az_{\text{S}} \right)v + az_{\text{S}}v_{\text{S}} - bz_{\text{X}}v_{\text{X}} \right]}{2kT} \right) \\ k_{\text{SX}} = 2\left(bz_{\text{X}} - az_{\text{S}} \right) Ne\lambda\sqrt{[\text{S}]_{\text{e}}^{a}[\text{S}]_{\text{i}}^{a}[\text{X}]_{\text{e}}^{b}[\text{X}]_{\text{i}}^{b}} \end{cases} \tag{3.4.5}$$

$$\begin{matrix}\text{H}^+/\text{sucrose} \\ \text{cotransporters}\end{matrix}\begin{cases} i_{\text{cotransporter}} = k_{\text{cotransporter}} \sinh\left(\dfrac{e\left(v - v_{\text{H}} - v_{\text{Suc}} - n_{\text{W}}v_{\text{H}_2\text{O}} \right)}{2kT} \right) \\ k_{\text{cotransporter}} = 2Ne\lambda\sqrt{\left[\text{H}^+\right]_{\text{e}}\left[\text{H}^+\right]_{\text{i}}[\text{Suc}]_{\text{e}}[\text{Suc}]_{\text{i}}[\text{H}_2\text{O}]_{\text{e}}^{n_{\text{W}}}[\text{H}_2\text{O}]_{\text{i}}^{n_{\text{W}}}} \\ v_{\text{Suc}} = \dfrac{kT}{e}\ln\dfrac{[\text{Suc}]_{\text{e}}}{[\text{Suc}]_{\text{i}}} \\ v_{\text{H}_2\text{O}} = \dfrac{\overline{V}_{\text{H}_2\text{O}}}{F}\sum_{\text{S}}\pi_{\text{S}} \end{cases} \tag{3.4.6}$$

$$\text{ion diffusion}\begin{cases} i_{\text{diff}} = -P_{\text{S}}A\dfrac{z_{\text{S}}F\left(z_{\text{S}}Fv + \overline{V}_{\text{S}}p \right)}{RT}\left[\dfrac{[\text{S}]_{\text{i}} - [\text{S}]_{\text{e}}\exp\left(-\dfrac{z_{\text{S}}Fv + \overline{V}_{\text{S}}p}{RT} \right)}{1 - \exp\left(-\dfrac{z_{\text{S}}Fv + \overline{V}_{\text{S}}p}{RT} \right)} \right] \end{cases} \tag{3.4.7}$$

$$\begin{matrix}\text{membrane} \\ \text{potential}\end{matrix}\begin{cases} \dfrac{dv}{dt} = -\dfrac{1}{C}\sum i \end{cases} \tag{3.4.8}$$

$$\begin{matrix}\text{species} \\ \text{concentration}\end{matrix}\begin{cases} \dfrac{d}{dt}\left(n_{\text{S}} \right)_{\text{i}} = \dfrac{\begin{pmatrix} -mz_{\text{S}}i_{\text{pump}}/\left(mz_{\text{S}} - nz_{\text{X}} \right) + az_{\text{S}}i_{\text{exchanger}}/\left(bz_{\text{X}} - az_{\text{S}} \right) \\ +\, i_{\text{cotransporter}} - i_{\text{S}} - i_{\text{diff}} \end{pmatrix}}{z_{\text{S}}F} \end{cases} \tag{3.4.9}$$

The coupling equation is where the osmotic pressure determined by the transport model is checked by the hydrostatic pressure due to matrix constraint. This occurs in the osmotic diffusion portion of the solvent flux equation,

$$\text{solvent flux}\begin{cases}\dfrac{1}{KA}\left(\dfrac{dV}{dt}-\sum_{\mathrm{S}}\overline{V}_{\mathrm{S}}\dfrac{d}{dt}\left(n_{\mathrm{S}}\right)_{\mathrm{i}}\right)=\overbrace{\sum_{\mathrm{S}}\sigma_{\mathrm{S}}\pi_{\mathrm{S}}}^{\text{transport model}}-\overbrace{p}^{\text{finite element analysis}} \\ \pi_{\mathrm{S}}=\dfrac{RT}{\overline{V}_{\mathrm{A}}}\dfrac{q_{\mathrm{S}}W_{\mathrm{A}}}{1000}\left(\left(m_{\mathrm{S}}\omega_{\mathrm{S}}\right)_{\mathrm{i}}-\left(m_{\mathrm{S}}\omega_{\mathrm{S}}\right)_{\mathrm{e}}\right)\end{cases} \tag{3.4.10}$$

The solution of the above set of ODEs must be carried out in parallel with the finite element analysis of the surrounding matrix. This combination is both highly coupled and highly nonlinear, resulting in a stiff system. As will be addressed in Section 4.5, the solution technique must therefore be carefully defined.

4.0 MODEL VALIDATION AND IMPLEMENTATION

4.1 SIMULATION OF THE RABBIT SINOATRIAL NODE

The transport model is validated using information presented in Endresen *et al.* [67] for the rabbit sinoatrial node. The model includes the effects of ion channels for K$^+$, Ca^{2+}, and Na$^+$, with open channel probabilities x, $f = 1 - x$, and h respectively, as well those of Na$^+$/K$^+$ pumps and Na$^+$/Ca^{2+} exchangers. The reactions for the pumps and exchangers respectively are,

$$\mathrm{ATP} + 3\mathrm{Na_i^+} + 2\mathrm{K_e^+} \underset{\beta}{\overset{\alpha}{\rightleftharpoons}} \mathrm{ADP} + \mathrm{P_{io}} + 3\mathrm{Na_e^+} + 2\mathrm{K_i^+} \tag{4.1.1}$$

$$3\mathrm{Na_e^+} + \mathrm{Ca_i^{2+}} \underset{\beta}{\overset{\alpha}{\rightleftharpoons}} 3\mathrm{Na_i^+} + \mathrm{Ca_e^{2+}} \tag{4.1.2}$$

The Ca^{2+} and Na$^+$ channels have an inactivation mechanism in addition to an activation mechanism and thus are modeled using Eqs. (3.4.3)$_{1,2,3}$ with activation probabilities d_∞ and m_∞ respectively. The K$^+$ channels have only an activation mechanism and hence will be modeled using Eqs. (3.4.2)$_{1,2}$. The oscillation of the membrane potential, and hence the pacemaker potential, is accomplished by introducing another Ca^{2+} influx. This is added to the model by modifying the Ca^{2+} channel current equation to read,

$$i_{\mathrm{Ca}} = \left[k_{\mathrm{Ca}}(1-x)d_\infty + k_{\mathrm{b,Ca}}\right]\sinh\left(\frac{e(v - v_{\mathrm{Ca}})}{kT}\right) \tag{4.1.3}$$

where $k_{\mathrm{b,Ca}}$ is the conductance parameter governing the additional Ca^{2+} influx. A program is written (Appendix B) using Microsoft Visual Basic 6.0 to solve the resulting coupled system of nonlinear ordinary differential equations using the Explicit 5th Order Dorman and Prince (DOPRI5) formulation of the Runge-Kutta method with step size control as presented in Section 2.4.1. The DOPRI5 code is adapted from the DOPRI5 program given in Hairer *et al.* [75]. The model parameters and initial conditions are taken from Tables 1 – 4 in Endresen *et al.* [67]. The resulting membrane potential is shown in Figure 4.1 below and is identical to the membrane potential plot given by Endresen when a correction factor of 1.25 is applied as set out in his paper.

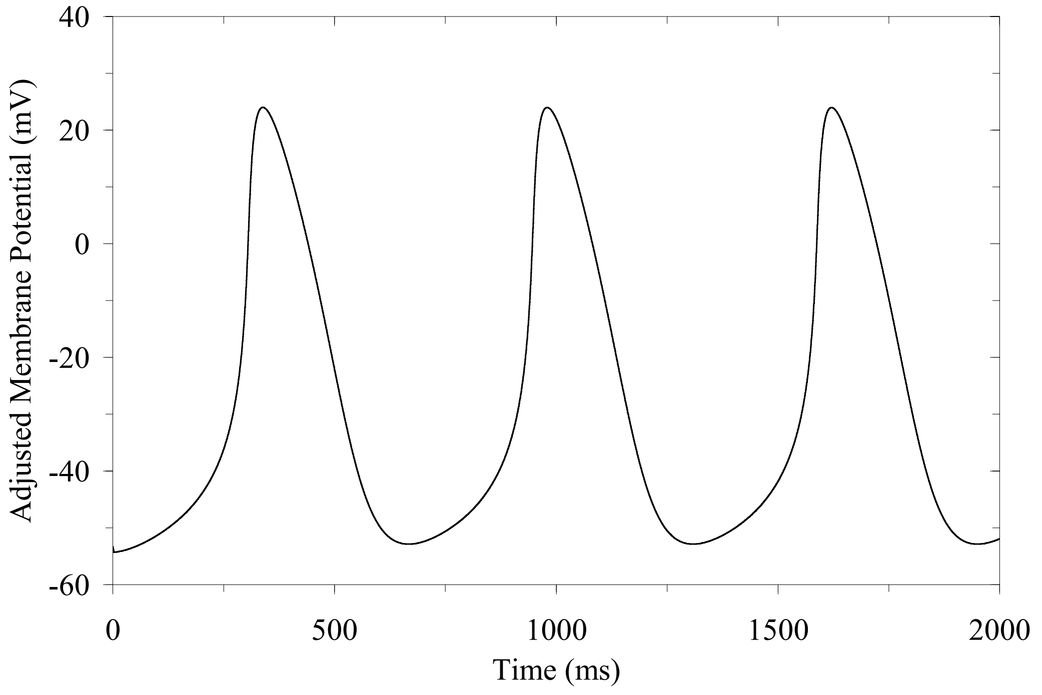


Figure 4.1: Simulated Membrane Potential for the Rabbit Sinoatrial Node

The model completed the 2 s simulation in 11159 time steps with 11 rejected time steps. The relative error tolerance is set to 1×10^{-11} and an initial time step of 0.01 ms is employed. A plot of the relative error (Figure 4.2) shows that relative stability is maintained with peaks in relative error occurring only at or near local minima or maxima on the membrane potential curve.

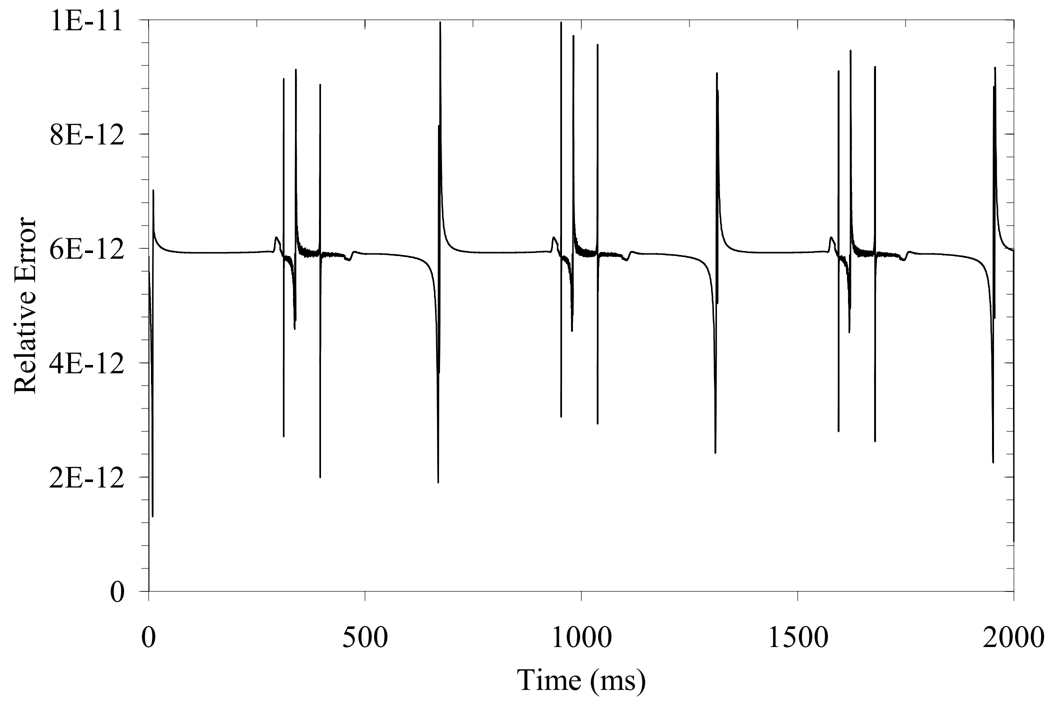


Figure 4.2: Relative Error for the Simulation of the Rabbit Sinoatrial Node

4.2 SIMULATION OF THE ASSAY CUP EXPERIMENT

As part of the nastic structures project, a series of experiments has been performed at Virginia Tech to quantify the volume flux across a lipid bilayer membrane embedded with reconstituted biological transporters [85]. The experiment consists of forming a lipid bilayer membrane containing SUT4 H^+/Sucrose cotransporters on the porous base of an assay cup that sits in a fluid reservoir. The solution in the reservoir contains a pH 4 buffer and varying concentrations of sucrose while the solution in the cup contains a pH 7 buffer and no sucrose initially. The fluid level in the cup is recorded as a function of time for 1.0, 5.0, and 10 mM sucrose concentrations in the reservoir fluid. A simulation of this experiment is carried out using the transport model and holding the H^+ concentration constant. The model implements only the ion pump and H^+/Sucrose cotransporter equations as outlined in Section 3.4. Since the number of moles of SUT4 cotransporter present in the membrane, N, is not known, it is lumped with the SUT4 rate constant, λ, and the square root of the water concentration which is assumed to represent a mole fraction of one. The resulting parameter along with the effective system capacitance, *C*, and effective hydraulic permeability of the membrane, *K*, are adjusted until a fit to the 10 mM data is achieved. The resulting input parameters are given in Table 4.1. These parameters are then used to predict the 1mM and 5 mM response (Figure 4.3).

Table 4.1: Input Parameters for Simulation of Assay Cup Experiments

Parameter	Value
$\mathrm{pH_e} = -\log_{10}\left[\mathrm{H^+}\right]_\mathrm{e}$	4.0
$\left[\mathrm{Suc}\right]_\mathrm{e}$	1.0, 5.0, 10.0 mM
$N_\mathrm{SUT4}\lambda_\mathrm{SUT4}\sqrt{\left[\mathrm{W}\right]_\mathrm{e}^{n_\mathrm{W}}\left[\mathrm{W}\right]_\mathrm{i}^{n_\mathrm{W}}}$	1.22×10^{16} Hz/(mM $\mathrm{H^+}$)·(mM Suc)
n_W	350
C	3.2 F
K	5.0×10^{-11} $\mathrm{m^3/N\cdot s}$
A	4.77×10^{5} $(\mu\mathrm{m})^2$
σ	1.0
$\mathrm{pH_{i0}}$	4.0
$\left[\mathrm{Suc}\right]_\mathrm{i0}$	0.0 mM
v_0	0.0 V
V_0	1.2×10^{11} $(\mu\mathrm{m})^3$

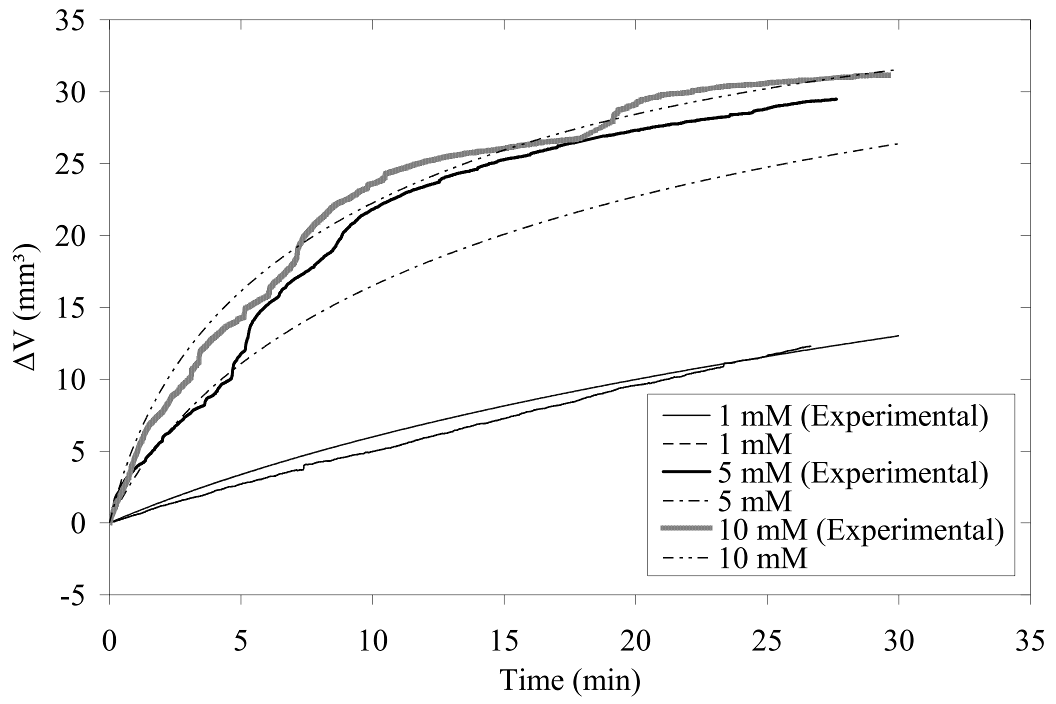


Figure 4.3: Assay Cup Volume Flux

The illustrated transport model volume flux predictions are within the bounds of experimental accuracy. The model inputs are physically reasonable to the extent that their values are known with the exception of the effective system capacitance which has a value of 3.2 F. This is several orders of magnitude higher than the actual capacitance of the lipid bilayer which has been measured at 0.456 nF without SUT4 [85]. It has been concluded that a portion of the protons transported across the membrane diffuse into the bulk solution rather than remain in the diffuse double layer as is assumed by the membrane potential equation given in Eq. (2.1.21). The larger effective capacitance has the effect of slowing the increase of the membrane potential as protons are transferred to the cup. Further discussion of the application of this parameter is provided in Chapter 7.0.

4.3 THE HYPERELASTIC MATRIX MODEL

In order to consider the hyperelastic component of the model in isolation, to serve as a benchmark in the coupled response, the change in volume expressed by Eq. (3.2.1) must be artificially represented. In this instance, this is accomplished via the substitution of an expanding ideal gas in place of the inclusion fluid in Figure 4.6. The ideal gas is given the properties of air. The properties of the matrix material are currently based on the properties of polytetrafluoroethylene (PTFE) and information found in Treloar [73]. The Mooney-Rivlin coefficients chosen are: $C_{10} = 89$, $C_{01} = 0$, and $D_1 = 9.23 \times 10^{-4}$ MPa. Results are generated for both a free displacement case and a blocked force case.

4.3.1 Spherical Actuator

A spherical actuator element is created as shown in Figure 4.4 with hydrostatic fluid elements to model the ideal gas within the inclusion. Referring to Figure 4.5, the initial volume fraction of the region A is 6.5% with initial internal pressure of 0.1 MPa and initial temperature of 293 K. The temperature of the ideal gas inside of region A is increased linearly to 60,000 K over 20 s causing the gas to expand.

For the free displacement case, the inclusion expands by 9.3% resulting in a new volume fraction of 7.1% with a final internal pressure of 18.7 MPa. Of the total radial displacement developed in the region A, 3.2% is observed at the nearest outer surface. The maximum Mises stress, of 26.2 MPa, is developed in the matrix material at the A-B interface (Figure 4.5a).

For the blocked force case, where the sample is constrained in one dimension, the inclusion expands by 9.1% resulting in a new volume fraction of 7.1% with a final internal pressure of 18.8 MPa. Of the total radial displacement developed in the region A, 2.4% is observed at the top and bottom surfaces and 3.4% is observed at the left and right surfaces. The maximum Mises stress, of 27.5 MPa, is developed in the matrix material at the A-B interface (Figure 4.5b).

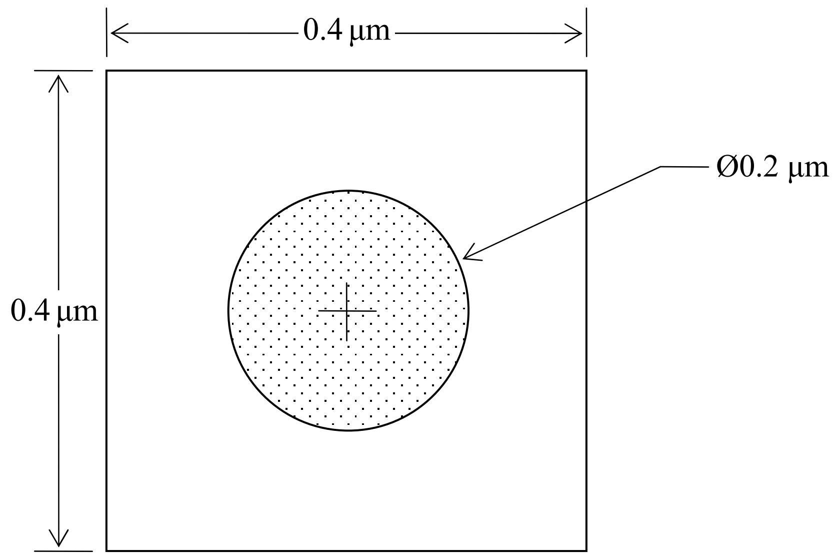
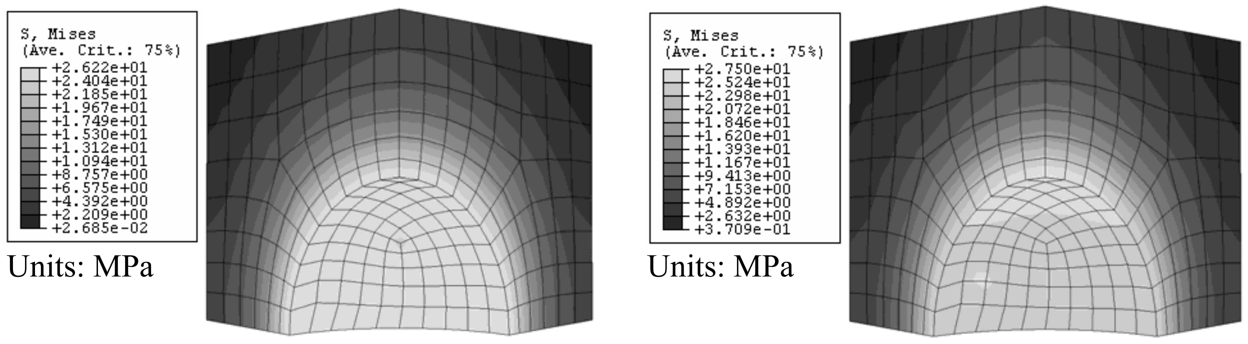


Figure 4.4: Spherical Actuator Element



(a) Free Displacement (b) Blocked Force

Figure 4.5: Spherical Actuator Mises Stress: Free Displacement and Blocked Force

4.3.2 Cylindrical Actuator

While the spherical actuator represents the ultimate goal of the program, current experimental work is focused on creating a simpler cylindrical actuator (Figure 4.6). In this actuator, secondary active transport mechanisms for H^+/Sucrose cotransport (SUT4) are embedded into a membrane that sits on a porous membrane frame. A barrel plate is positioned on top of the membrane frame. This plate has a series of holes in it that will hold the working fluid for the actuator. A thin cover plate is placed on top of the assembly. The vertical displacement the cover plate at the center of one of the cylinders will be measured.

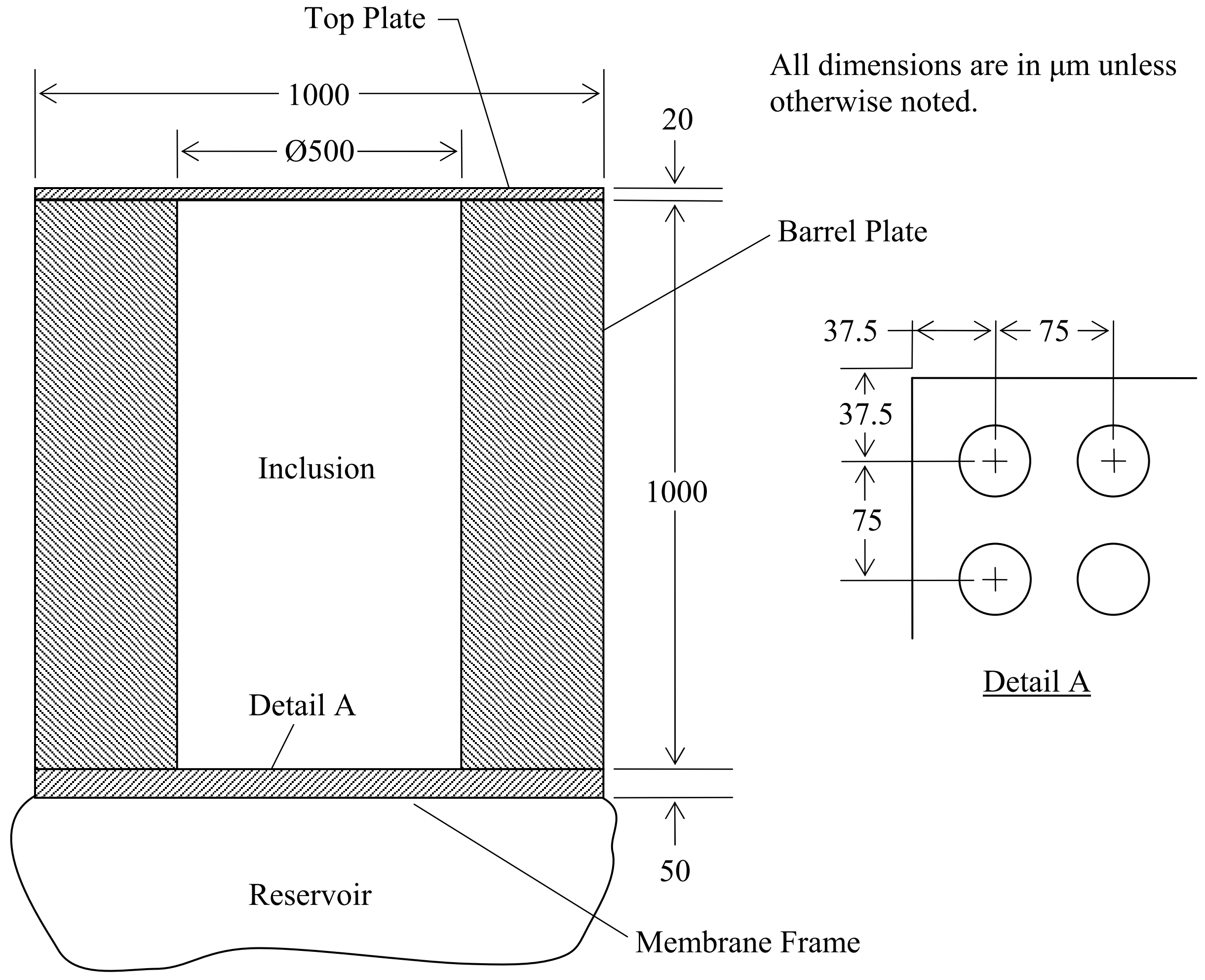


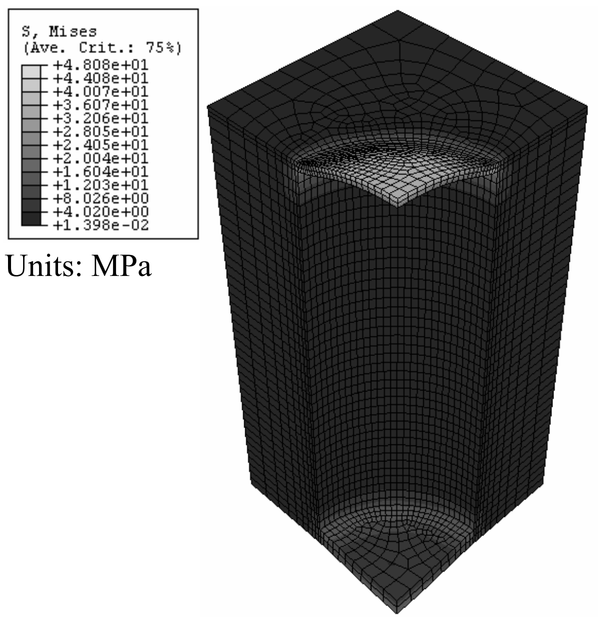
Figure 4.6: Cylindrical Actuator Unit

Validation of the coupled model requires the simulation of this cylindrical actuator. Therefore, a finite element analysis of a single actuator unit is carried out. The model is created so that the components of the actuator can be made from different materials. This model is tested using ideal gas expansion in a similar manner as the spherical case with the gas having the properties of air. The initial pressure is 0.1 MPa at a temperature of 293 K and the initial volume fraction of the inclusion A is 18.3%. The temperature of the ideal gas inside the inclusion is increased linearly to 6,000 K over a period of 20 s.

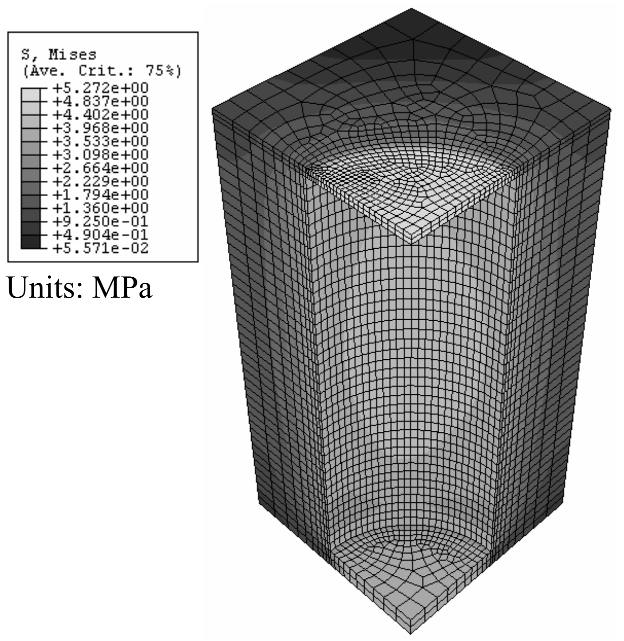
For the free displacement case, the gas expands by 4.55% resulting in a new volume fraction of 19.2%, with a final internal pressure of 1.96 MPa. A total actuator displacement of

5.28% is developed in the top plate at the actuator axis. A maximum stress of 48.1 MPa is developed in the top plate at its intersection with the barrel plate (Figure 4.7a).

For the blocked force case, the gas expands by 1.39% resulting in a new volume fraction of 18.6%, with a final internal pressure of 2.02 MPa. A maximum stress of 5.27 MPa is developed in the top plate at its intersection with the barrel plate (Figure 4.7b).



(a) Free Displacement



(b) Blocked Force

Figure 4.7: Cylindrical Actuator Mises Stress: Free Displacement and Blocked Force

4.4 IMPLEMENTATION OF THE COUPLED MODEL

The coupled model is implemented using ABAQUS/Standard 6.5. The ABAQUS package is chosen because of its ability to incorporate user-defined subroutines to define material behavior as well as to increase the functionality of several ABAQUS options for which data line usage alone may be too restrictive. The user-defined subroutine sets are written in FORTRAN 77 and compiled using Compaq Visual FORTRAN 6.6 (See Appendix C and Appendix D for source codes).

ABAQUS already includes the ability to model hyperelastic behavior using the Mooney-Rivlin form of the strain energy potential. The polymer matrix is meshed using 8-node hybrid linear 3D hexahedral continuum elements with the hyperelastic option (C3D8RH). This element type is chosen due to constraints imposed by the coupling of the hyperelastic model with the hydrostatic pressure generated by the transport model as well as recommendations from the ABAQUS documentation. These will be discussed further below.

4.4.1 Element Coupling of Inclusion and Matrix Regions

The challenge of coupling the transport model with the hyperelastic model comes from the fact that the response of the matrix depends upon not only the external loads but also the pressure exerted by the fluid, which in turn is affected by the deformation of the matrix. The actual coupling of the deformation of the matrix to the hydrostatic pressure of the inclusion fluid is accomplished through the use of hydrostatic fluid elements [70]. In ABAQUS, hydrostatic fluid elements are linear surface elements that share the nodes with the material elements surrounding an enclosed cavity. The fluid properties including hydrostatic pressure are assigned to a cavity reference node that lies on the cavity surface or at the intersection of the planes of symmetry. The temperature and pressure of the fluid inside the cavity are assumed to be uniform. The cavity volume is determined by summing the volume of the pyramidal subvolumes formed by the hydrostatic fluid elements and the cavity reference node [87]. The expected cavity volume during each increment is calculated from the fluid mass and density. Because the hydrostatic fluid elements are linear, linear continuum elements must be used to maintain continuity. Since the continuum elements are hexahedral and thus have 4-node faces, 4-node 3D hydrostatic fluid elements (F3D4) are used.

ABAQUS does not have a specific function for creating the hydrostatic fluid elements on the faces of the continuum elements. Therefore a program has been written (Appendix F) using Microsoft Visual Basic 6.0 that reads the ABAQUS input file and uses the node definitions to determine which nodes lay on the boundary of the inclusion. The program then reads the element definitions for the continuum elements and determines which faces of the continuum elements lay on the inclusion surface. A hydrostatic fluid element is created for each element face on the inclusion boundary.

4.4.2 Incorporation into an ABAQUS User-Subroutine Set

The transport model is incorporated into an ABAQUS analysis by creating a user-defined fluid (`UFLUID`) subroutine that defines the fluid density, ρ, pressure compliance, C_p, and temperature compliance, C_T, of the fluid within each fluid cavity at each time increment based upon pressure and temperature information passed into the subroutine by ABAQUS. The fluid pressure and temperature compliances are defined as [70],

$$C_p = \frac{d\rho^{-1}}{dp} = -\rho^{-2}\frac{d\rho}{dp} \tag{4.4.1}$$

$$C_T = \frac{d\rho^{-1}}{dT} = -\rho^{-2}\frac{d\rho}{dT} \tag{4.4.2}$$

where p is the fluid cavity pressure and T is the fluid temperature. At the beginning of an analysis ABAQUS calculates the mass of fluid within each cavity based upon the initial cavity volume and density. The expected cavity volume during each increment is calculated from the fluid mass and density. The transport model is incorporated into the `UFLUID` and the density is changed to reflect solvent flow into or out of the inclusion. This is done because there is no way for a user-defined subroutine to change the mass inside a fluid-filled cavity. Because of how ABAQUS calculates the volume, changing the fluid density will have the same effect as changing the mass as long as the change in density is compensated for in the fluid pressure compliance or the fluid is incompressible, i.e. the fluid pressure compliance is zero.

The coupling of species and solvent transport to the matrix response is accomplished by integrating the system of ODEs that comprise the transport model in parallel with the ABAQUS simulation. This is achieved by using the `UFLUID` subroutine as a driver routine for a numerical ODE solver (Figure 4.8). With this configuration, a current value of the hydrostatic pressure due to matrix response is fed into the transport model, which in turn feeds back the corresponding volume change into ABAQUS. Since this feedback is through a constitutive model subroutine, ABAQUS will iterate the matrix and fluid response calculations until a quasi-equilibrium hydrostatic pressure and volume change are established before commencing to the next time increment.

Model input parameters and initial conditions are read from a text file using the `UEXTERNALDB` utility subroutine within ABAQUS [70]. It is called at the start of an analysis, at the beginning and end of each increment, and at the end of an analysis. Data is passed between the `UEXTERNALDB` subroutine and the other subroutines in the user-subroutine set using `COMMON` blocks (Figure 4.8). Output data from the transport model is written to a space delimited ASCII text file at the end of each increment.

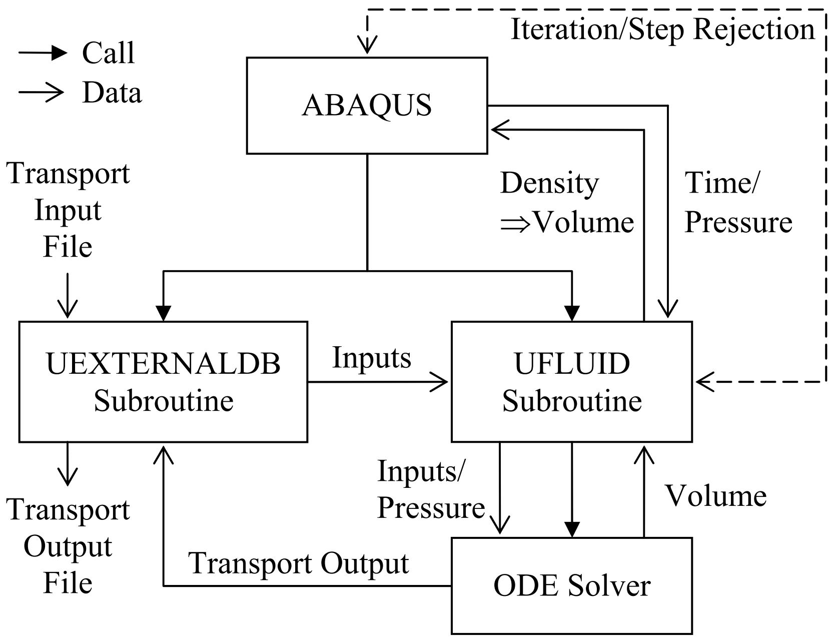


Figure 4.8: Organization of Transport Model Implementation into an ABAQUS user-subroutine set

4.5 NUMERICAL INTEGRATION OF THE MODEL EQUATIONS

The numerical integration of the transport model equations presents many challenges. The first is that the system of ODEs is highly nonlinear and stiff. In the transport model, the gating variables, transporter rate parameters, and membrane potential all have relatively short time constants with respect to the overall response time of the nastic material. Another numerical integration challenge comes from the coupling with ABAQUS. The `UFLUID` subroutine is called by ABAQUS for each hydrostatic fluid element in the model during each equilibrium iteration in a time step, the size of which is determined by ABAQUS. In addition, ABAQUS can reject a time step at any point before or after calling the `UFLUID` subroutine. The driver for the ODE solver coded into the `ULFUID` subroutine must be able to accommodate these constraints.

Several ODE integration algorithms have been tested on the transport model at varying material stimulation power levels. To minimize the effects of roundoff error, especially in the stiff portions of the ODE system, double precision variables are used throughout the source codes for all algorithms. In addition, all algorithms use a variable time step determined by the root mean square (RMS) error estimation (Section 2.4.3). The algorithms tested are chosen based upon information given in Endresen [68] and Victorri *et al.* [64] as well as knowledge that the transport model ODEs are stiff. The following ODE integration algorithms have been tested,

- 5th Order Dorman and Prince formulation of the Explicit Runge-Kutta Method (DOPRI5)
- 4th Order Rosenbrock Method
- Semi-Implicit Extrapolation Method
- Lawrence Livermore LSODE package – multistep Backward Differentiation Formula (BDF) Methods (Gear Methods)
- Lawrence Livermore VODE package – variable coefficient multistep BDF Methods

It is found that the Lawrence Livermore VODE package, which uses BDF methods (Section 2.4.2), is able to integrate the transport model with the least number of time steps. However, since it uses its own internally generated grid points in time and does not have the ability to "back up" integration to the beginning of the previous increment and continue from that point, there are compatibility issues with its use within an ABAQUS simulation. Therefore, a separate driver package to call the ABAQUS `UFLUID` subroutine is created (Appendix E) which has subroutines that mimic the utility routines available within ABAQUS. This package uses a polynomial fit to pressure-volume data taken from an ABAQUS simulation of the actuator to feed back hydrostatic pressure to the `UFLUID` subroutine. This eliminates the need for the driver routine, which mimics ABAQUS, to reject a step after calling `UFLUID` causing the ODE solver to "back up". Although this driver package has the ability to dramatically reduce run times, it is limited to simulation of matrix materials that are not history dependent. For low power simulations, it is found that the DOPRI5 algorithm (Section 2.4.1) is relatively efficient despite not being stiffly stable. Also, since the code uses the ABAQUS grid points in time, it has no compatibility issues with use in an ABAQUS simulation.

5.0 PREDICTIONS AND PARAMERTIC STUDIES

The ultimate goal of this work is to facilitate the development and implementation of high-power nastic materials. This work, in concert with parametric studies will form the basis for a feedback loop in material synthesis and implementation efforts. By perturbing the properties of the validated model, synthesis guidance for optimizing material properties and response may be offered. The model may subsequently be used in a predictive manner. In this instance, the objective is to generate loading curves on which a structural length scale model may be based. *Target responses of greatest concern are free strain, blocked force, and rate of response.* To this end a series of parametric studies has been performed. The goal of these studies is to determine the relative importance of the various parameters associated with both the polymer matrix and the biological transport components.

5.1 CYLINDRICAL ACTUATOR

5.1.1 Matrix Properties of the Cylindrical Actuator

For simplicity, the ideal gas expansion approach detailed in Seciton 4.3 is employed in the parametric studies of the Mooney-Rivlin coefficients. These studies implement the parametric study capability within ABAQUS. Studies are performed for a free displacement case and a blocked force case where expansion in the vertical direction is constrained. A parametric study is carried out for each of the three Mooney-Rivlin coefficients: C_{10}, C_{01}, and D_1. The baseline properties chosen are those for polytetrafluoroethylene (PTFE), which has an average shear modulus of 178 MPa and an average bulk modulus of 2167 MPa. The baseline values for the

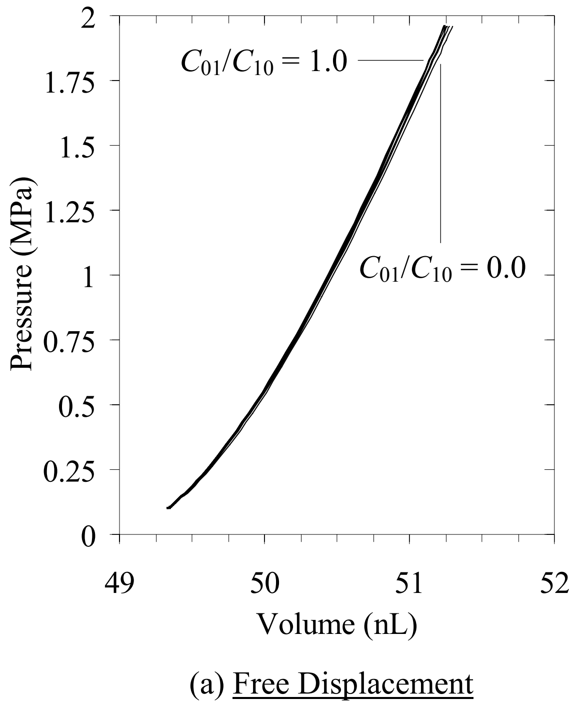
Mooney-Rivlin coefficients are determined using the relations and assumptions outlined in Section 3.3, which yields the values $C_{10} = 89\,\text{MPa}$, $C_{01} = 0.0\,\text{MPa}$, and $D_1 = 9.23\times10^{-4}\ \text{MPa}^{-1}$.

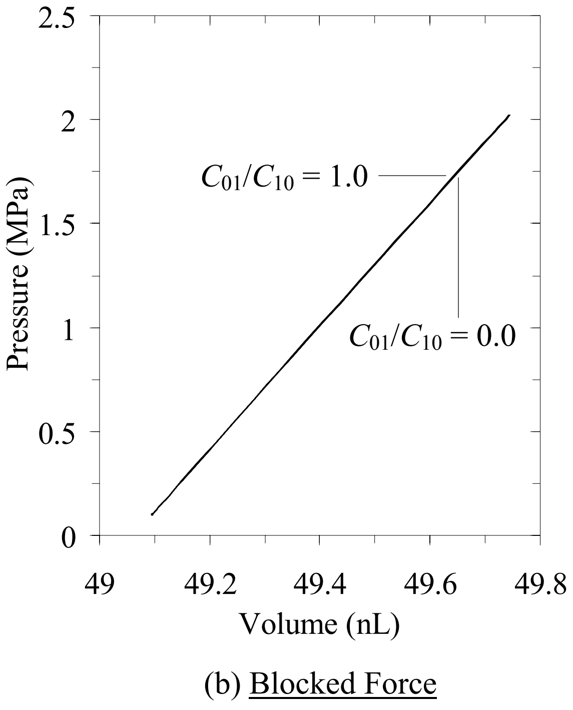
Varying the ratio C_{01}/C_{10} has little effect on the material response in either the free displacement or blocked force cases. The pressure-volume response in the free displacement case is nearly linear with the largest variation with C_{01}/C_{10} occurring at large volumes (Figure 5.1a). Blocking vertical expansion causes the pressure-volume response to be linear and shifted to the left on the pressure axis (Figure 5.1b). The free displacement is modestly affected, ranging from 55.9 µm at $C_{01}/C_{10} = 1.0$ to 56.5 µm at $C_{01}/C_{10} = 0.0$ (Figure 5.2).

The coefficient C_{10} has the most influence on the material response, especially with regard to the pressure-volume response. The pressure-volume response in the free displacement case is nonlinear with slope increasing as C_{10} is increased (Figure 5.3a). Blocking vertical expansion causes the pressure-volume response to be approximately linear and shifted to the left on the pressure axis (Figure 5.3b). The free displacement varies significantly with C_{10}, ranging from 72.4 µm at $C_{10} = 44.5\,\text{MPa}$ to 44.4 µm at $C_{10} = 178\,\text{MPa}$ (Figure 5.4).

Varying the coefficient D_1 has a modest effect on the pressure-volume response and a significant effect on the free displacement. The pressure-volume response in the free displacement case is nonlinear with the largest variation with D_1 occurring at large volumes (Figure 5.5a). Blocking vertical expansion causes the pressure-volume response to be more linear and shifted to the left on the pressure axis (Figure 5.5b). The free displacement is significantly affected, ranging from 55.1 µm at $D_1 = 9.23\times10^{-5}\ \text{MPa}^{-1}$ to 67.7 µm at $D_1 = 9.23\times10^{-3}\ \text{MPa}^{-1}$ (Figure 5.6).

In summary, the coefficient C_{10} has the greatest impact on the pressure-volume response of the cylindrical actuator in both the free displacement and blocked force cases with the slope of the pressure-volume curves increasing as C_{10} is decreased. Varying the coefficient D_1 has a moderate impact on the pressure-volume response in both cases with the slope of the pressure-volume curves increasing as D_1 is increased. The ratio C_{01}/C_{10} has negligible impact on the material response. Treatment of C_{01} as zero is validated as an appropriate simplifying assumption.





(a) Free Displacement (b) Blocked Force

Figure 5.1: Cylindrical Actuator Pressure-Volume Response for Varied C_{01}/C_{10}

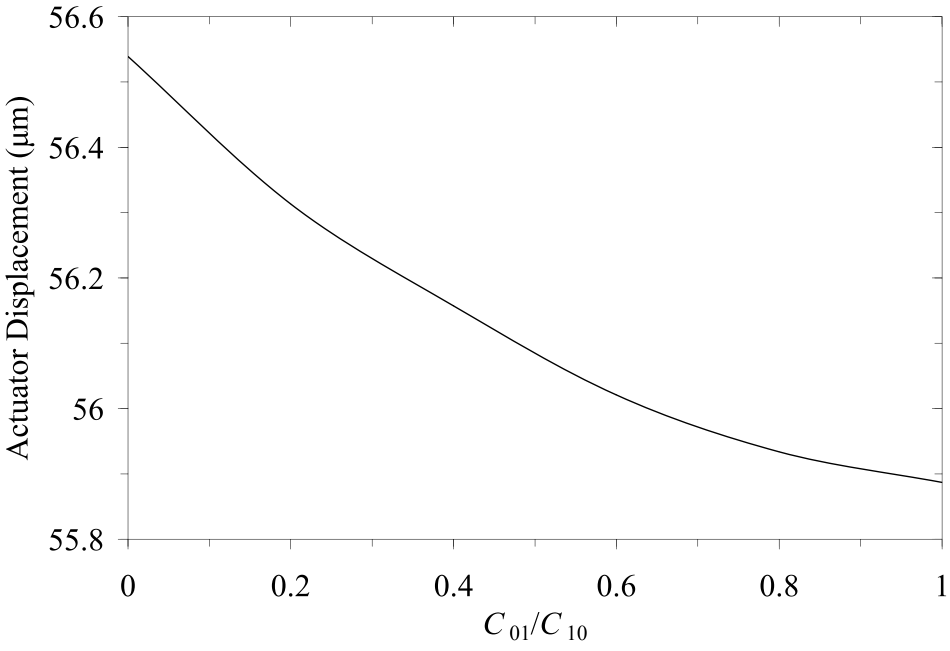


Figure 5.2: Cylindrical Actuator Free Displacement for Varied C_{10}/C_{10}

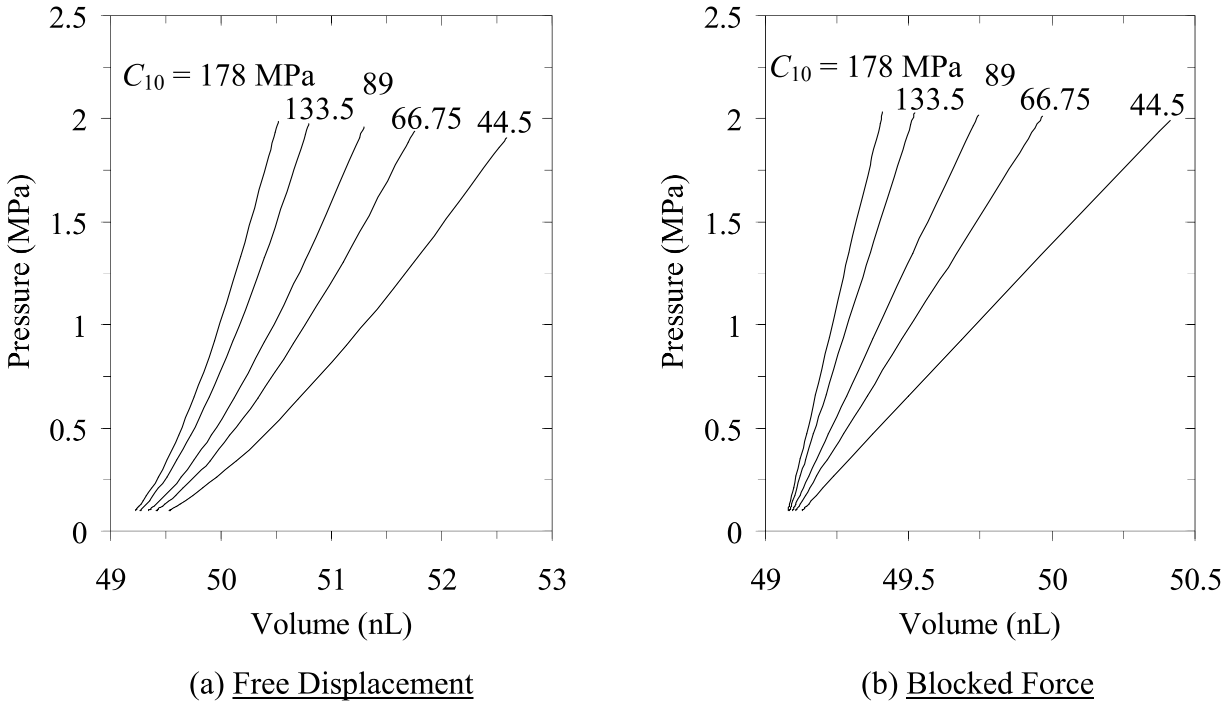


Figure 5.3: Cylindrical Actuator Pressure-Volume Response for Varied C_{10}

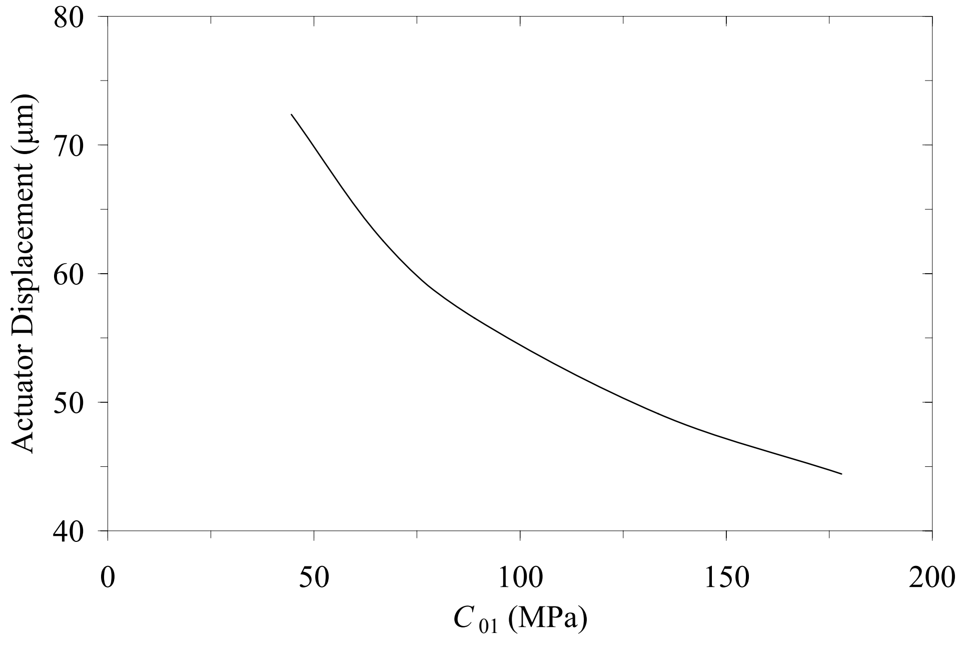
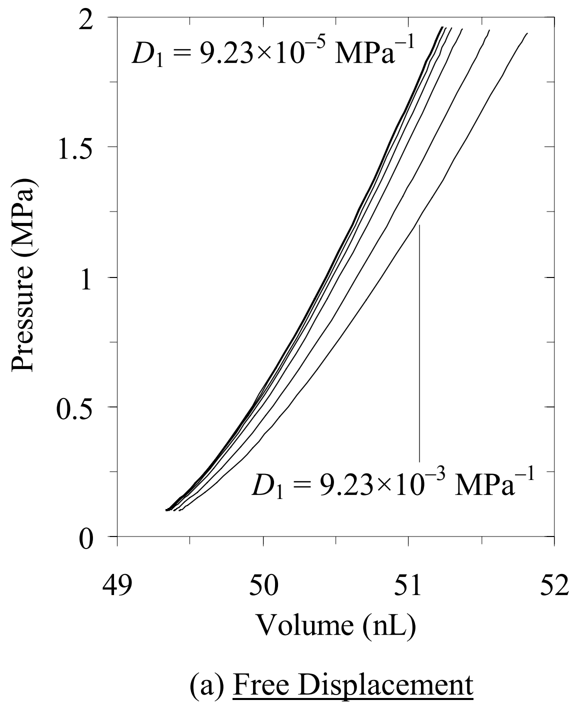


Figure 5.4: Cylindrical Actuator Free Displacement for Varied C_{10}



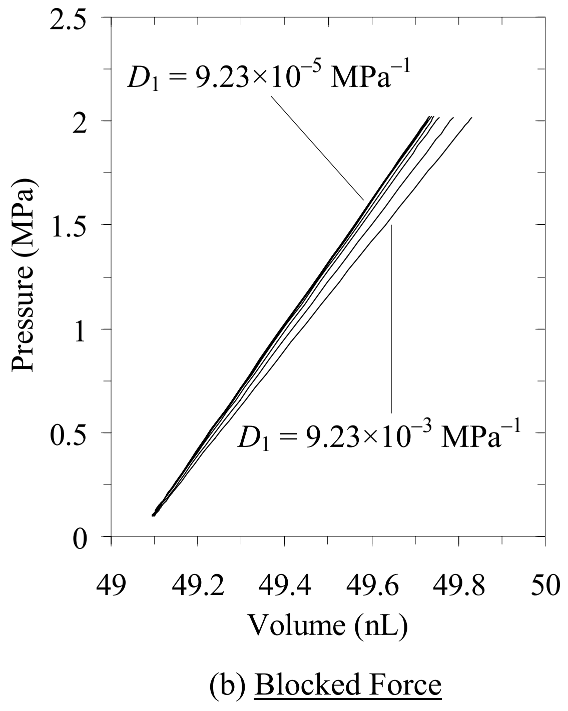


Figure 5.5: Cylindrical Actuator Pressure-Volume Response for Varied D_1

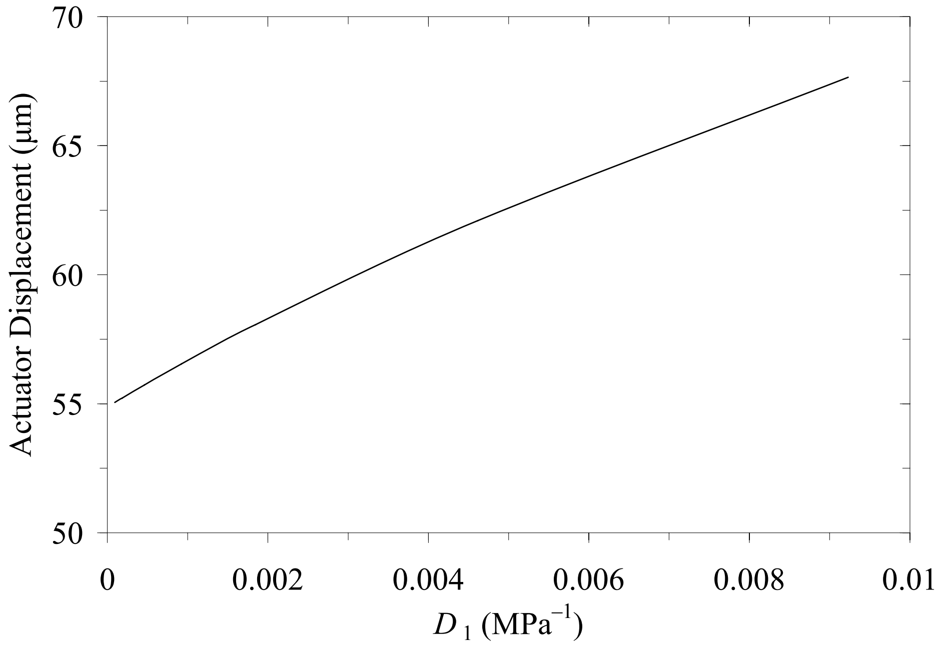


Figure 5.6: Cylindrical Actuator Free Displacement for Varied D_1

5.1.2 Coupled Model Parametric Studies of the Cylindrical Actuator

The cylindrical actuator transport model parametric studies are carried out a 'medium-power' level as compared to biological systems. This means that the ATP free energy release, ΔG_{ATP}, is equal to that produced by typical physiological concentrations of ATP, ADP, and P_{io}. The goal of these studies is to determine how each input parameter affects the peak displacement, equilibrium displacement, and the actuator response time. The following transport parameters are varied:

- ATP free energy release
- External and initial internal pH
- External and initial internal sucrose concentrations
- SUT4 cotransporter density expressed through varying $k_{\mathrm{cotransporter}}$
- Effective water permeability of the membrane
- H^+ pump density expressed through varying k_{pump}
- SUT4 water stiochiometry
- Membrane Capacitance
- Matrix stiffness expressed through varying C_{10}

A summary of the results from these parametric studies is given in Table 5.2. The inputs for the baseline case are based upon those used in the assay cup experiment simulation presented in Section 4.2. Because nastic material development in the foreseeable future will remain focused on engineered membranes containing proton pumps and proton/sucrose cotransporters, these studies do not include ion channels or exchangers. Based on symmetry, the input parameters are for one quarter of the total membrane area.

The membrane of the baseline case includes ion pumps that transport one H^+ out of the inclusion per cycle [88], SUT4 H^+/Sucrose cotransporters [82], and permeability to both water and protons. It is assumed that H^+ and sucrose are the only solutes in the system. The system is initially in equilibrium upon the introduction of ATP. The concentration of ATP is maintained such that the free-energy available to the H^+ pumps, ΔG_{ATP}, is constant. The cylindrical actuator unit is 1000 μm wide, has a 500 μm diameter by 1000 μm tall inclusion with a 20 μm thick top plate, and 50 μm thick bottom plate. The polymer matrix is modeled using typical values for

PTFE. The Mooney-Rivlin coefficients applied assume compression loading [73]. The input parameters are given in Table 5.1.

Table 5.1: Input Parameters for the Cylindrical Actuator Baseline Case

Parameter	Value
$\mathrm{pH}_e = \mathrm{pH}_{i0}$	5.0
$[\mathrm{Suc}]_e = [\mathrm{Suc}]_{i0}$	1.0 mM
$N_{\mathrm{SUT4}} e \lambda_{\mathrm{SUT4}} \sqrt{[\mathrm{W}]_e^{n_\mathrm{W}} [\mathrm{W}]_i^{n_\mathrm{W}}}$	70 μA/(mM H^+)·(mM Suc)
n_W	350
C	0.115 F
K	1.8×10^{-12} m³/N·s
A	1.72×10^{4} (μm)²
σ	1.0
k_{pump}	70 μA
ΔG_{ATP}	-1.0×10^{-20} J
v_0	0.0 V
V_0	4.91×10^{7} (μm)³
C_{10}	89 MPa
C_{01}	0.0 MPa
D_1	9.23×10^{-4} MPa

The time dependent response for the baseline case is illustrated in Figures 5.7 through 5.11. The proton pumps quickly establish a pH gradient of 1.07 upon the introduction of ATP which in turn activates the SUT4 cotransporters (Figure 5.7). The cotransporters are able to hold the pH gradient relatively constant while pumping water and sucrose into the inclusion (Figure 5.9). Because the opposing sucrose gradient (Figure 5.8) rises slowly, the SUT4 cotransporters are able to move water into the inclusion faster than it can diffuse out. This causes the hydrostatic pressure inside the inclusion to rise considerably higher than the osmotic pressure (Figure 5.10) to a *peak* in the transport response. Eventually the hydrostatic pressure combined with the slowing of the SUT4 cotransporters, due to the establishment of a sucrose gradient, allows the outward water flux due to diffusion to overtake the inward flux generated by the SUT4. The outward flux of water further increases the sucrose gradient which causes further

SUT4 slowing. The system reaches *equilibrium* when the hydrostatic pressure inside the inclusion is balanced by the osmotic pressure (Figure 5.10). The expansion of the inclusion induces a maximum Mises stress in the surrounding matrix of 21.3 MPa at peak displacement and 1.48 MPa at equilibrium (Figure 5.11). The actuator displacement rises to 36.2 μm in approximately 1 min then falls to an equilibrium value of 4.56 μm in approximately 17 min.

Blocking vertical expansion affects both the peak and equilibrium inclusion volumes as well as the actuator response time. The peak and equilibrium inclusion volumes of 49.4 and 49.1 nL are lower than the 50.1 and 49.2 nL generated in the free displacement case (Figure 5.12). Also, the material response time to both peak volume and equilibrium volume of 0.46 and 9.7 min are faster than the 1.2 and 15 min of the free displacement case. The peak hydrostatic pressure of 1 MPa is higher than the 0.65 MPa generated in the free displacement case (Figure 5.13). The maximum Mises stress generated in the blocked force case is also lower since the load on the top plate is supported by the blocking rather than the barrel plate (Figure 5.14).

Table 5.2: Summary of Parametric Study Results for the Cylindrical Actuator[†]

Parameter		Peak			Equilibrium		
Symbol	Value	Time [min]	Pressure[‡] [MPa]	Vol./Disp. [nL/μm]	Time [min]	Pressure[‡] [MPa]	Vol./Disp. [nL/μm]
Baseline		1.1	0.651	50.1/36.2	17	0.0291	49.2/4.6
ΔG_{ATP}	-0.5×10^{-20} J	0.64	0.193	49.5/19.6	14	0.00602	49.1/0.99
	-1.5×10^{-20} J	1.5	1.08	50.6/44.8	76	0.140	49.4/16
$\mathrm{pH}_{\mathrm{e,i0}}$	3	0.11	2.79	51.9/64.8	11	0.0236	49.1/4.5
	8	-	-	-	5800	0.0291	49.2/4.5
$[\mathrm{Suc}]_{\mathrm{e,i0}}$	0.2 mM	1.6	0.0970	49.3/12.4	12	0.00543	49.1/0.89
	5.0 mM	0.87	4.08	52.8/73.0	97	0.0316	49.7/26
SUT4	0.1×	4.1	0.109	49.4/13.5	85	0.0291	49.2/4.6
	10×	0.31	2.12	51.4/58.3	10	0.0291	49.2/4.6
K	1.8×10^{-13} m³/N·s	3.1	2.21	51.5/59.3	98	0.0291	49.2/4.6
	1.8×10^{-11} m³/N·s	0.35	0.108	49.4/13.5	20	0.0291	49.2/4.6
n_{W}	0	-	-	-	17	0.0268	49.2/4.2
	700	0.91	1.46	50.9/50.4	21	0.0321	49.2/5.0
C_{10}	44.5 MPa	1.4	0.517	50.5/43.6	20	0.0291	49.3/8.9
	178 MPa	0.85	0.774	49.8/29.5	14	0.0291	49.1/2.8

[†] Free displacement.
[‡] Hydrostatic pressure generated inside the inclusion.

System power is a function of ATP free energy (Figure 5.15). The ATP concentration is varied so that the resulting free energy release from the hydrolysis of a single ATP molecule, ΔG_{ATP}, varies between −0.5 and -1.5×10^{-20} J. The ATP free energy affects both the displacement at peak pressure and at equilibrium. The peak pressure and equilibrium displacements range from 19.6 and 0.99 μm for the -0.5×10^{-20} J case to 44.8 and 16.0 μm for the -1.5×10^{-20} J case respectively. Also, the material response time to equilibrium is increased from 14 min for the -0.5×10^{-20} J case to 76 min for the -1.5×10^{-20} J case.

Varying the external and initial internal pH values impacts both the pump and cotransporter reaction rates (Figure 5.16). The displacement at equilibrium varies modestly across the pH range with a value of around 4.5 μm. The displacement at peak pressure varies considerably from 64.8 μm at a pH of 3 to nonexistent at pH 7 and 8. The pH has a dramatic impact on the time needed to reach equilibrium which ranges from 11 min for a pH of 3 to 5800 min for a pH of 8.

The external and initial internal sucrose concentrations affect both the cotransporter reaction rate and the osmotic pressure (Figure 5.17). The result is variation in displacement at peak pressure and equilibrium from 12.4 and 0.89 μm for the 0.2 mM case to 73.0 and 25.9 μm for the 5.0 mM case respectively. Also, the actuator response time to equilibrium is increased from 12 min for the 0.2 mM case to 97 min for the 5.0 mM case.

Varying the density of SUT4 cotransporters in the membrane increases the transporter current which affects both the peak pressure displacement and actuator response time (Figure 5.18). The peak pressure displacement and actuator response time to equilibrium range from 13.5 μm and 85 min for the 0.1×Baseline case to 58.3 μm and 10 min for the 10×Baseline case respectively.

The membrane permeability to water, *K*, impacts the rate of water flux in the presence of a hydrostatic or osmotic pressure gradient (Figure 5.19). Because the membrane water permeability has no effect on the energy available for transport, the equilibrium displacement of the top of the cylindrical actuator is constant across range of permeabilities with a value of 4.6 μm. However, peak pressure displacement, which is determined by the equilibrium point between active water transport and osmotic diffusion, varies considerably. The maximum peak pressure displacement of 59.3 μm occurs at a decreased permeability of 1.8×10^{-13} m³/N·s; a minimum of 13.5 μm occurs at an increased permeability of 1.8×10^{-11} m³/N·s. Also, the actuator response times to both peak pressure and equilibrium are increased from 0.35 and 20 min respectively for the 1.8×10^{-11} m³/N·s case to 3.1 and 98 min respectively for the 1.8×10^{-13} m³/N·s case.

Contrary to expectation, varying the density of H^+ pumps in the membrane has little effect on either the displacement or response time (Figure 5.20). This is because the establishment of a membrane potential limits the pH gradient which in turn limits the SUT4 reaction rate. Because the pH gradient cannot exceed the limit imposed by the membrane potential, the H^+ pumps can only pump protons out of the inclusion at the rate with which they are being introduced to the inclusion by the SUT4 and proton diffusion. Hence, the pumps operate at a rate much slower than their kinetics allow and so even a significant change in the pump density has no effect on the material response since it adds or removes excess pump capacity. In the extreme, however, a severe reduction in pump density that reduces the pump capacity below the operating point *will* have an effect on the material response.

Varying the SUT4 water stiochiometry increases the active water flux which affects the peak pressure displacement (Figure 5.21). This displacement ranges from 50.4 μm with a stiochiometry of 700 to nonexistent with no active water transport. Unlike the other parameters considered thus far, there is no known experimental control over this parameter. This study is carried out in an attempt assess the effect of the assumed value of 350.

The membrane capacitance, *C*, which has a considerable impact on the predicted assay cup transport response as presented in Section 4.2, has no effect on the actuator response (Figure 5.22). Since the H^+ pumps and the SUT4 establish a proton loop, the magnitude of the membrane potential has little effect on the rate of sucrose and water transport. This is because an increase in the membrane potential tends to decrease the rate of proton removal by the H^+ pumps and increase the rate of proton addition by the SUT4. This would have the effect of reducing the pH gradient which would tend to increase the H^+ pump rate and decrease the SUT4 rate. Changing the membrane capacitance when H^+ pumps and SUT4 are present simply alters the membrane potential and pH gradient but has no other effect on the transport response. Thus application of a capacitance value which is known to be artificially high will not adversely affect system predictions.

Varying the stiffness of the polymer matrix, C_{10}, affects peak pressure displacement and equilibrium displacement as well as the actuator response time (Figure 5.23). The peak pressure and equilibrium displacements range from 29.5 and 2.78 μm for the 178 MPa case to 43.6 and 8.94 μm for the 44.5 MPa case respectively. Also, the material response time is increased from 14 min for the 178 MPa case to 20 min for the 44.5 MPa case.

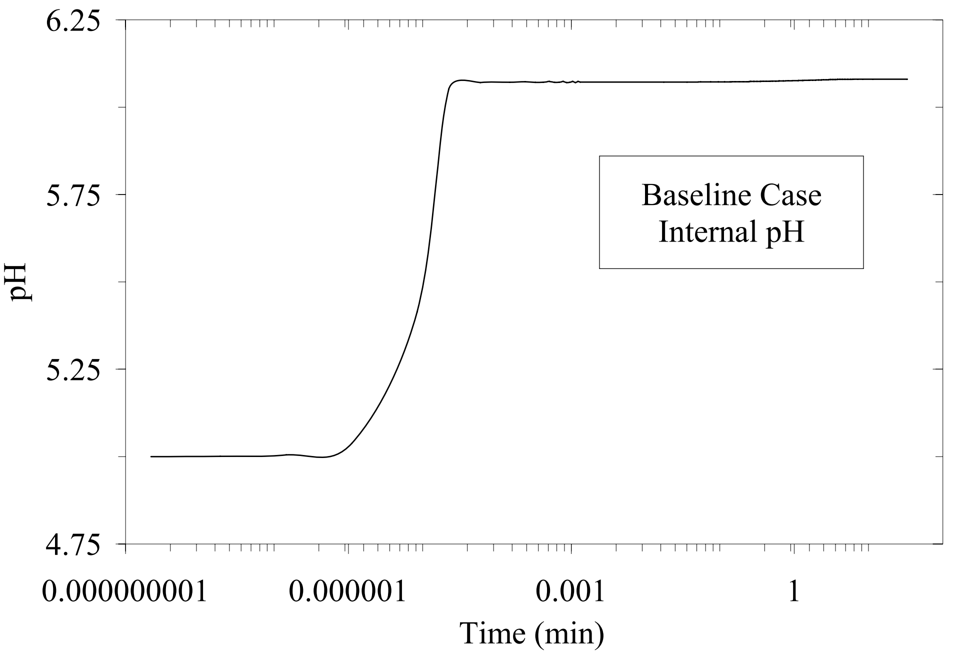


Figure 5.7: Cylindrical Actuator Baseline pH – The pH rises rapidly in the first 100 μs after the introduction of ATP but levels off with the establishment of a membrane potential and the activation of the SUT4.

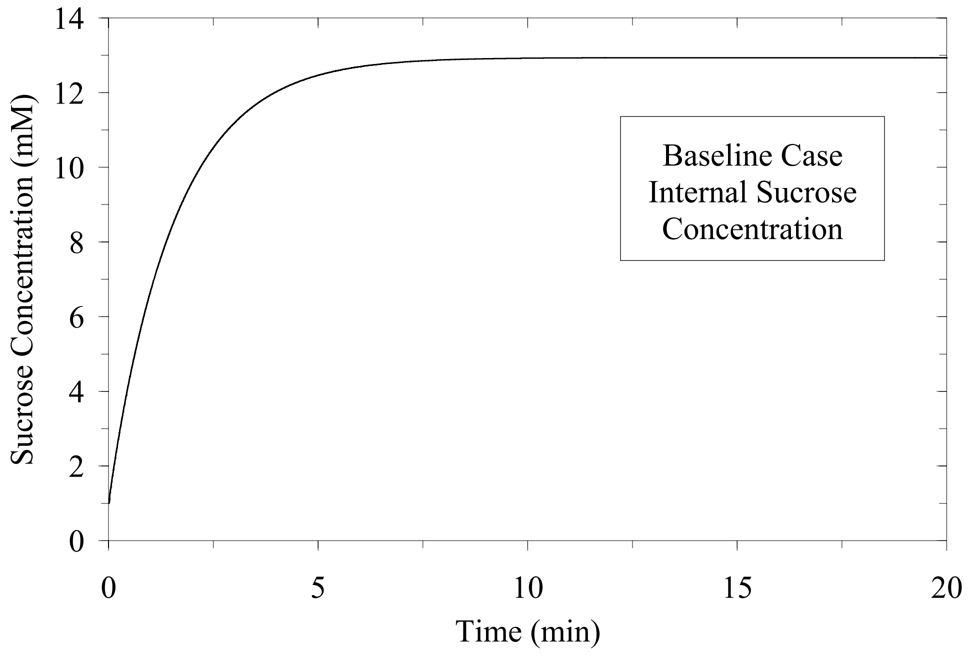


Figure 5.8: Cylindrical Actuator Baseline Sucrose Concentration

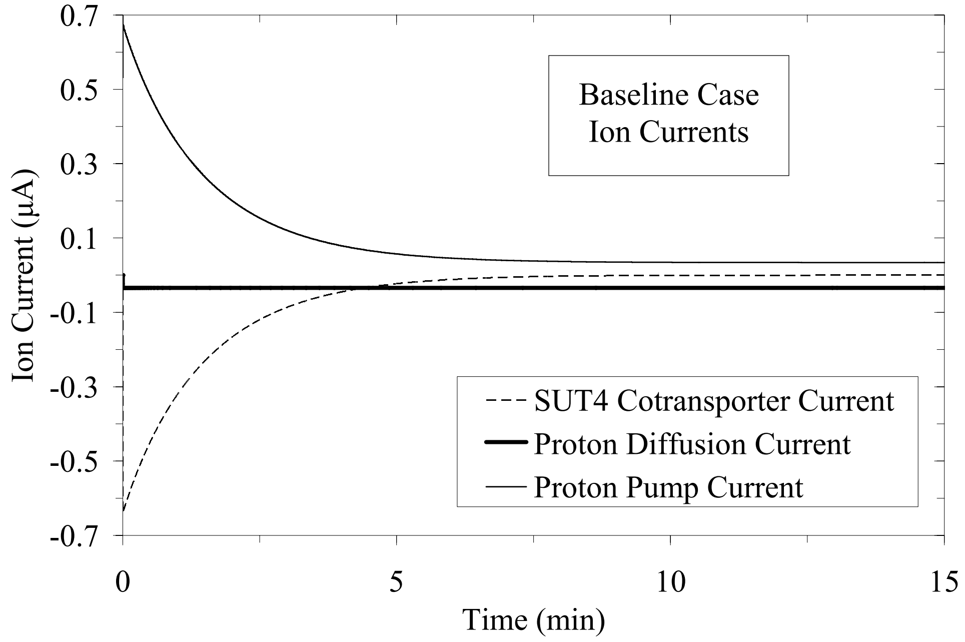


Figure 5.9: Cylindrical Actuator Baseline Transporter Currents – Electric currents generated by the biological transporters and volumetric flux of water across the membrane for the baseline case. Because of the rapid establishment of a membrane potential, the pH gradient remains relatively constant and the H^+ pump, SUT4, and proton diffusion currents cancel each other.

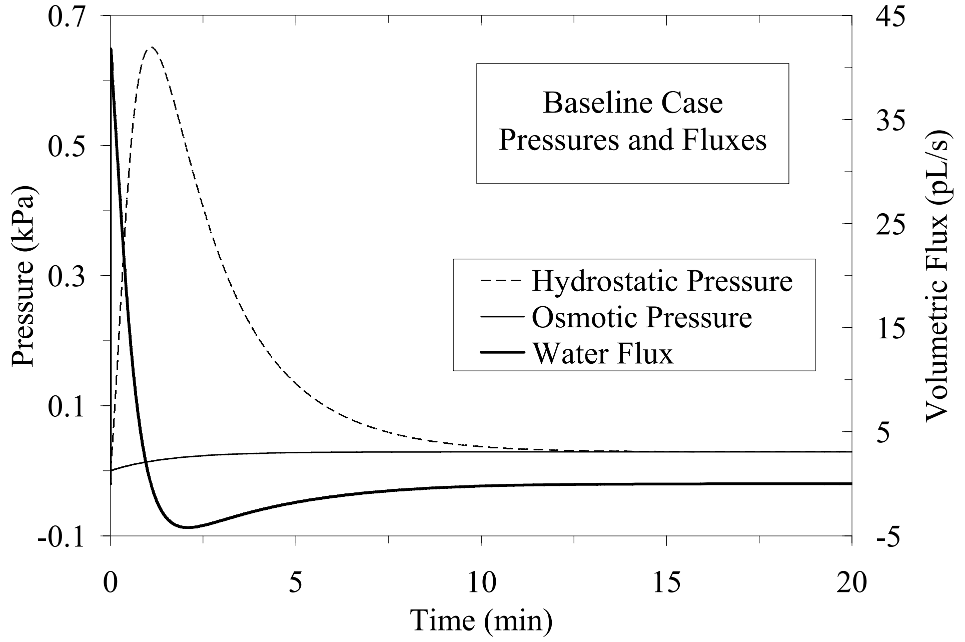
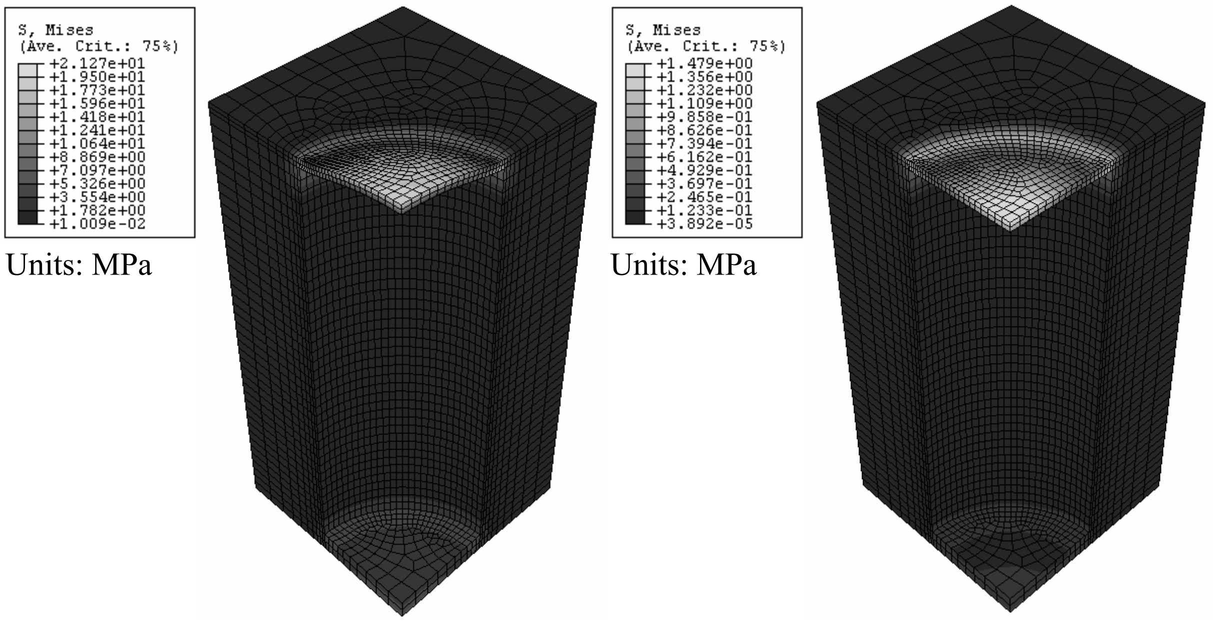


Figure 5.10: Cylindrical Actuator Baseline Pressures and Fluxes – Inclusion hydrostatic and osmotic pressures and volumetric flux of water for the baseline case. The volumetric water flux is proportional to both the difference between the hydrostatic and osmotic pressures and the SUT4 current.



(a) Peak Pressure (b) Equilibrium

Figure 5.11: Cylindrical Actuator Free Displacement Mises Stress: Peak Pressure and Equilibrium – The maximum stress occurs where the top plate meets the cylinder wall.

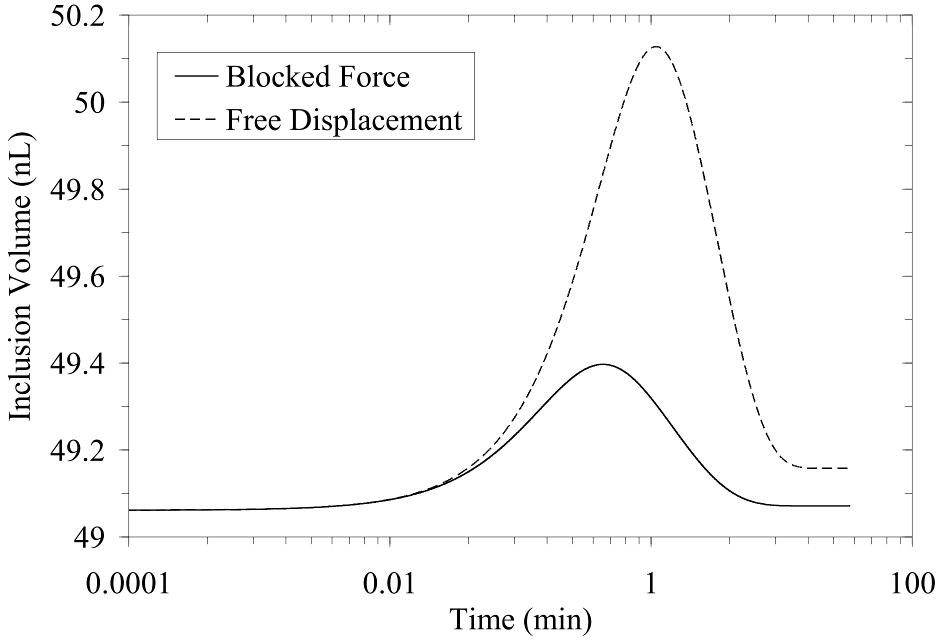


Figure 5.12: Cylindrical Actuator Inclusion Volume: Blocked Force and Free Displacement

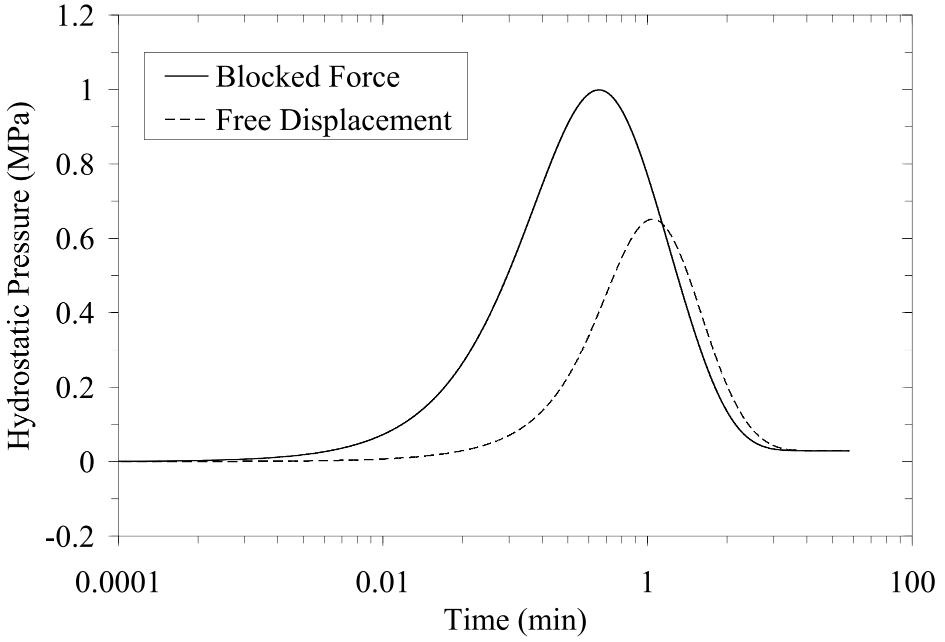
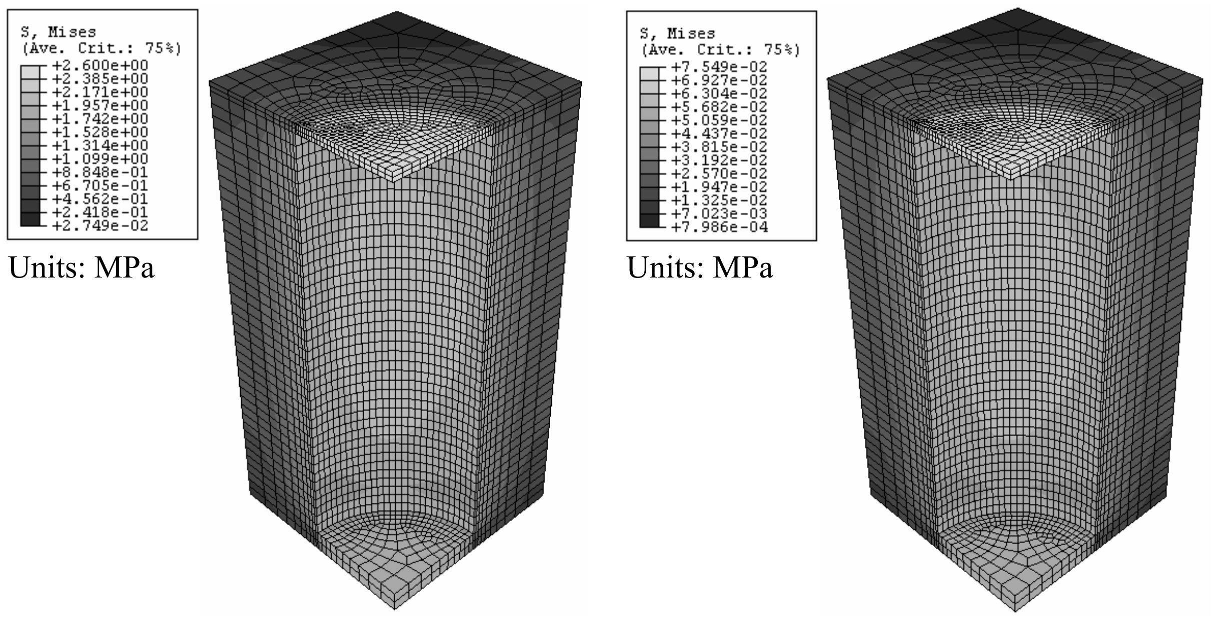


Figure 5.13: Cylindrical Actuator Hydrostatic Pressure: Blocked Force and Free Displacement



(a) Peak Pressure (b) Equilibrium

Figure 5.14: Cylindrical Actuator Blocked Force Mises Stress: Peak Pressure and Equilibrium

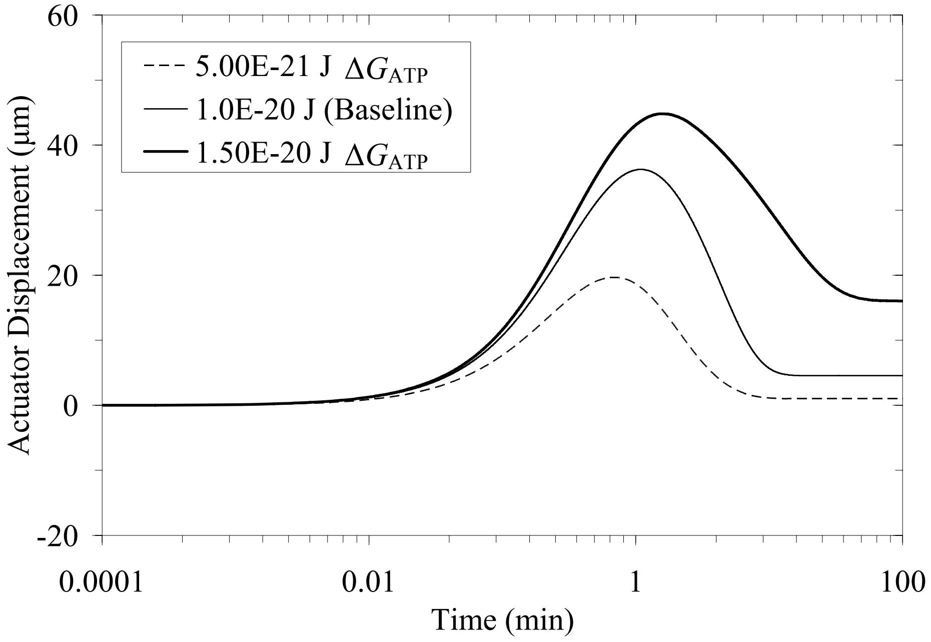


Figure 5.15: Cylindrical Actuator Displacement for Varied ΔG_{ATP} – Displacement of the center of the top plate.

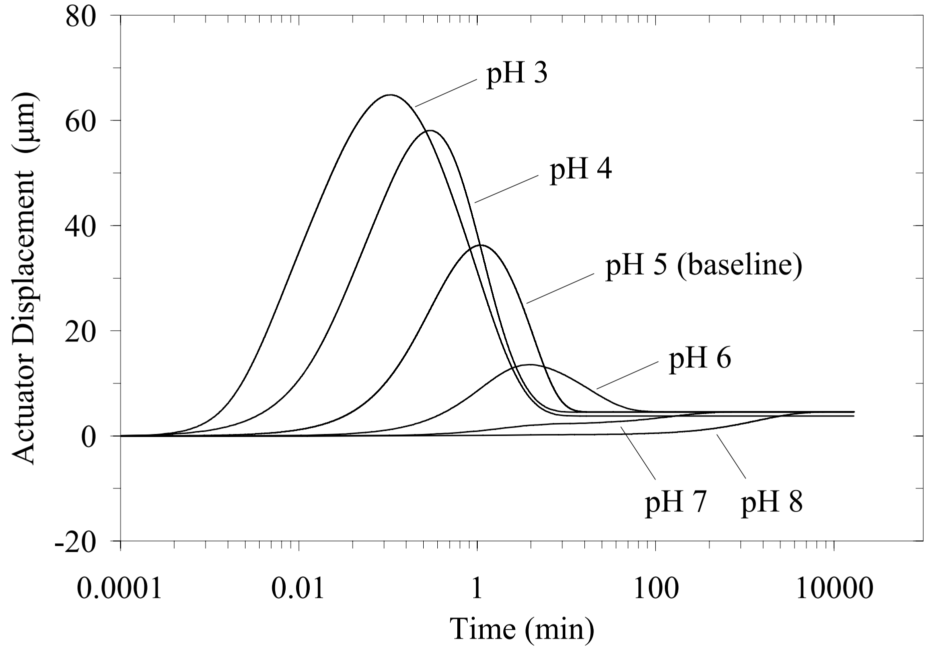


Figure 5.16: Cylindrical Actuator Displacement for Varied pH – Values of external and initial internal pH varied together to avoid an initial pH gradient.

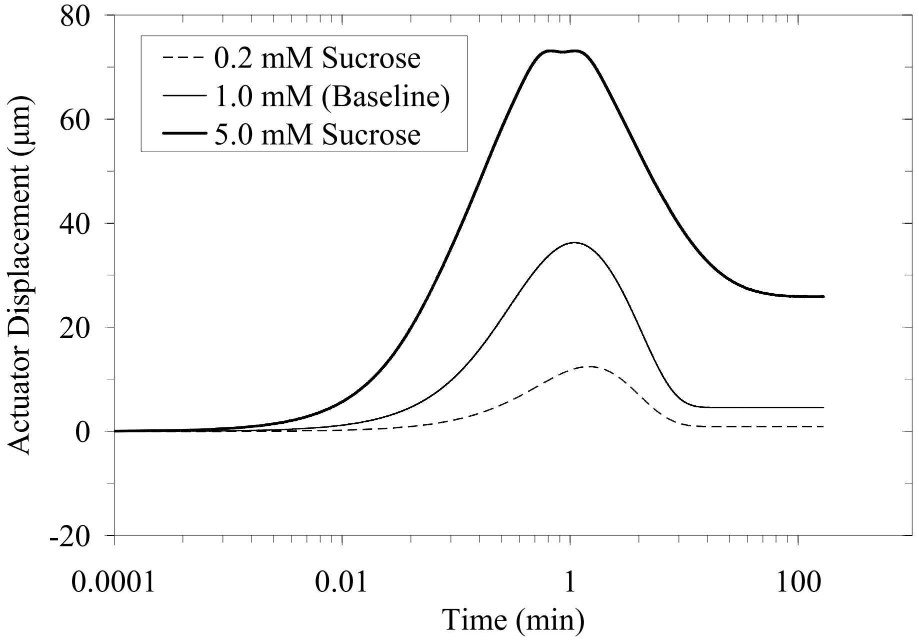


Figure 5.17: Cylindrical Actuator Displacement for Varied Sucrose Concentration – External and initial internal sucrose concentrations varied together to avoid an initial sucrose gradient.

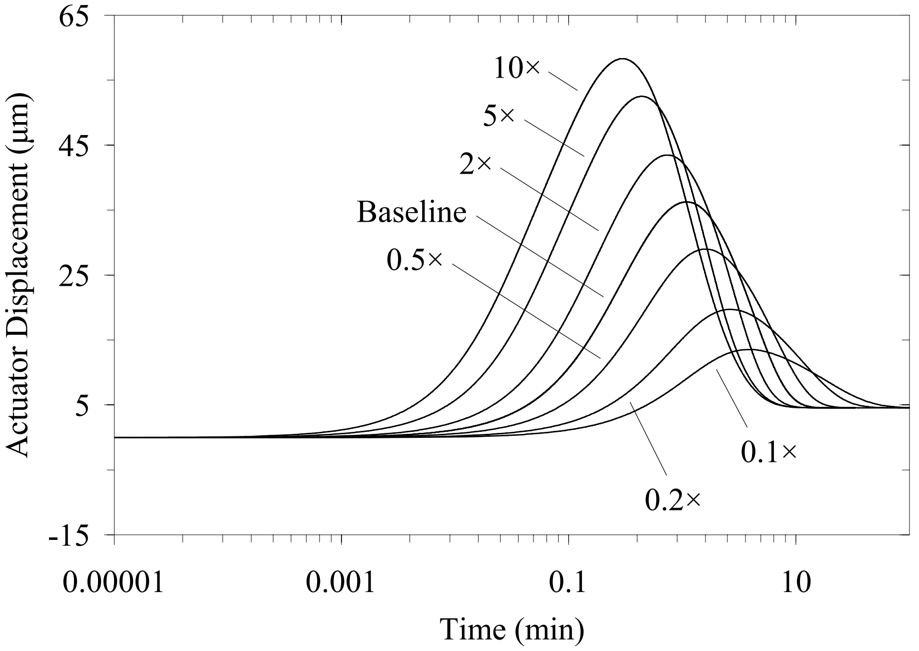


Figure 5.18: Cylindrical Actuator Displacement for Varied SUT4 Cotransporter Density

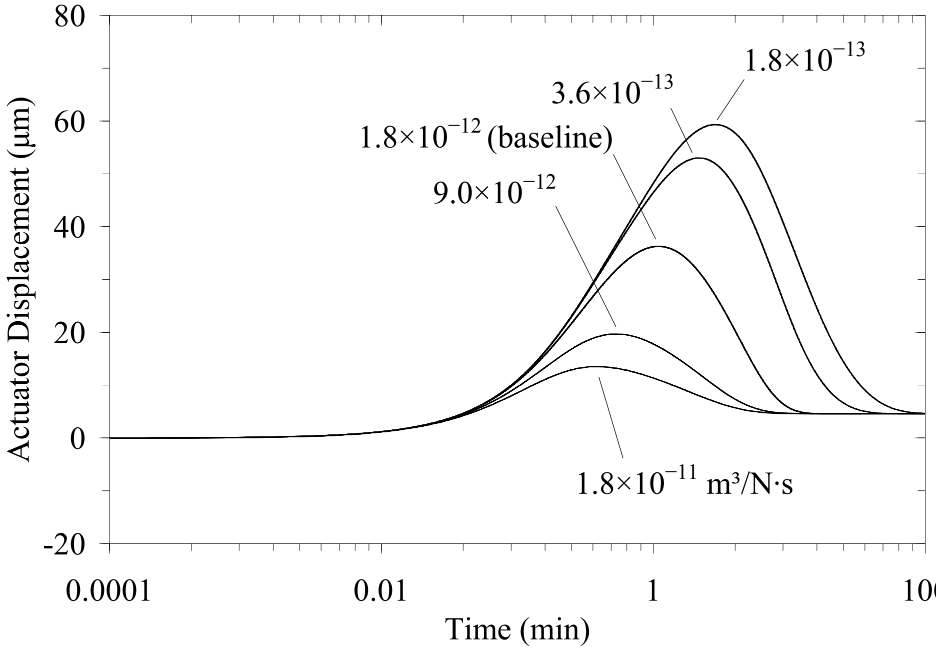


Figure 5.19: Cylindrical Actuator Displacement for Varied Membrane Water Permeability

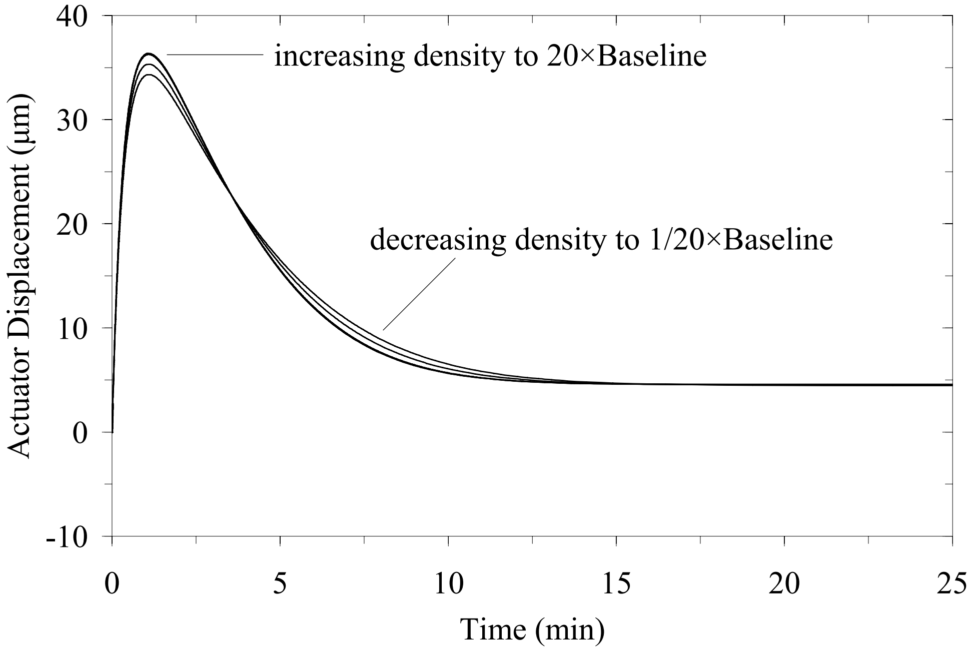


Figure 5.20: Cylindrical Actuator Displacement for Varied H^+ Pump Density – The H^+ pump density has negligible effect on the displacement and response time.

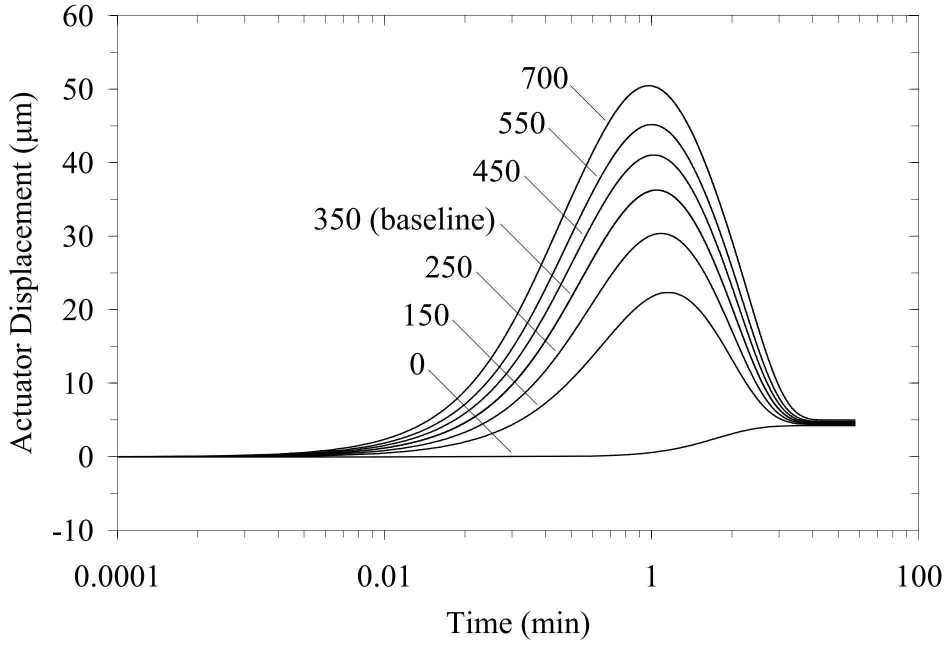


Figure 5.21: Cylindrical Actuator Displacement for Varied SUT4 Water Stoichiometry

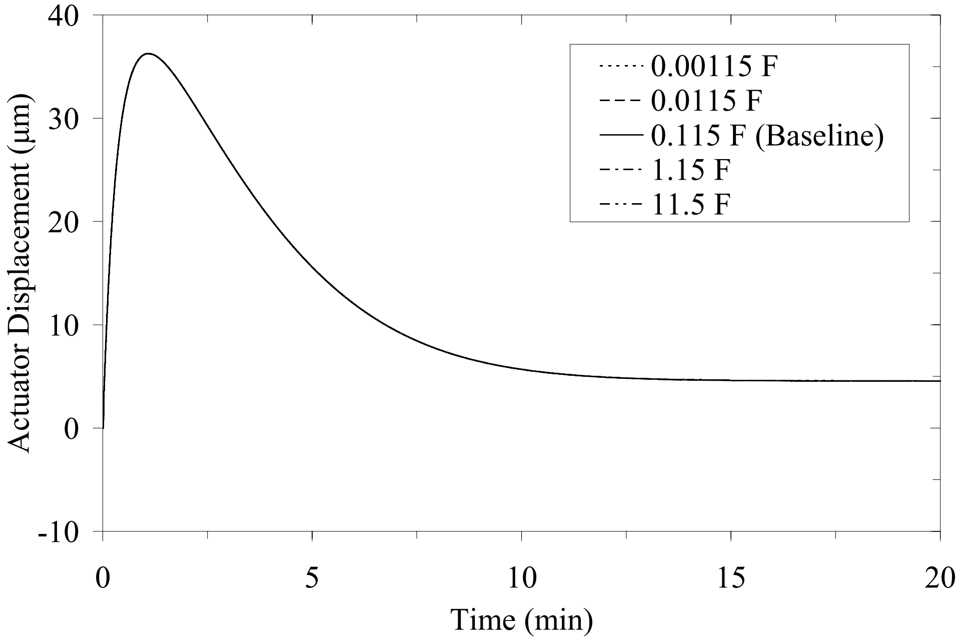


Figure 5.22: Cylindrical Actuator Displacement for Varied Membrane Capacitance – The membrane capacitance has no effect on the displacement or response time.

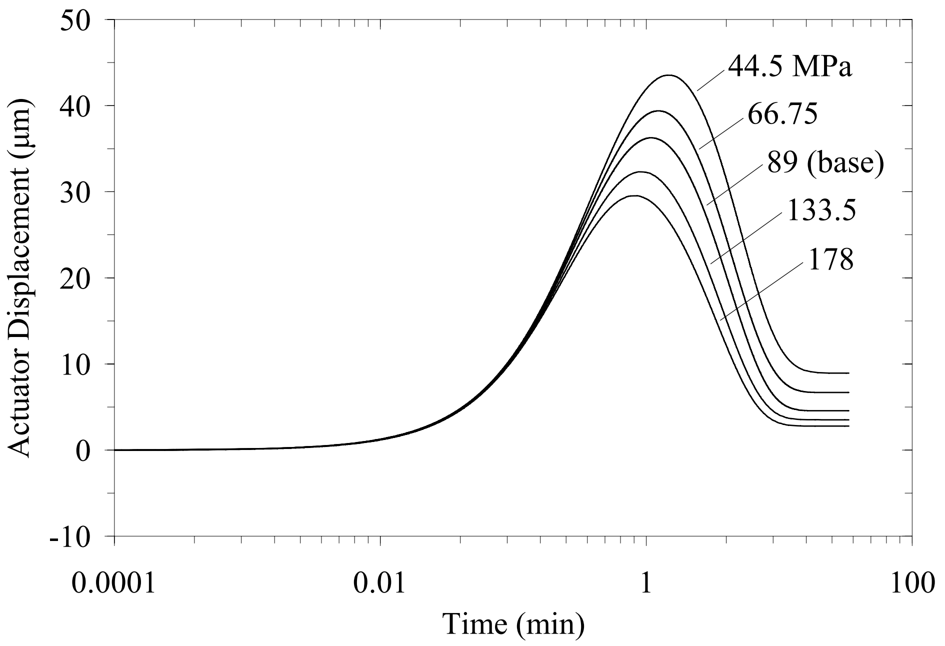


Figure 5.23: Cylindrical Actuator Displacement for Varied Matrix Stiffness

5.1.3 Discussion of Cylindrical Actuator Parametric Studies

The above parametric studies demonstrate that the material response is influenced by several factors and that these factors often influence more than one aspect of that response. Results suggest that decreasing the membrane water permeability can significantly increase the peak pressure displacement but at the expense of increasing the actuator response time. It is also found that decreasing the external and initial internal pH dramatically increases the peak pressure displacement while decreasing the material response time. For a pH above 6, the actuator proceeds directly to the equilibrium displacement rather than overshooting and returning to equilibrium. Increasing the ATP free energy and the external and initial internal sucrose concentrations have the greatest impact on equilibrium displacement. Also, it is found that increasing the SUT4 cotransporter density can increase the peak pressure displacement and modestly decrease the actuator response time. Increasing the stiffness of the matrix decreases both the peak pressure and equilibrium displacements as well as the material response time. It is also found that the H^+ pump density and membrane capacitance have no effect on the displacement or response time since the membrane potential and SUT4 limit the pump reaction rate. The conclusions from the parametric studies are summarized in Table 5.3.

Table 5.3: Summary of Parametric Study Conclusions for the Cylindrical Actuator

Optimization	Parameters to Increase[†]	Parameters to Decrease[†]
Maximize Peak Displacement	• Sucrose Concentration • SUT4 Water Stiochiometry • SUT4 Density • ATP Free Energy	• pH • Membrane Water Permeability • Matrix Stiffness
Maximize Equilibrium Displacement	• Sucrose Concentration • ATP Free Energy	• Matrix Stiffness
Minimize Time to Peak Displacement	• Membrane Water Permeability • SUT4 Density • Sucrose Concentration • Matrix Stiffness	• pH • ATP Free Energy • SUT4 Water Stoichiometry
Minimize Time to Equilibrium	• Membrane Water Permeability • SUT4 Density • Matrix Stiffness	• pH • ATP Free Energy • Sucrose Concentration • SUT4 Water Stoichiometry

[†] Parameters listed in order from greatest to least impact on response.

The ultimate goal of the nastic material development effort is to engineer a new class of active materials for application as a skin for a morphing aircraft. To accomplish this, the nastic material must maximize both the blocked force and free displacement while minimizing the time over which this occurs. In other words, the goal is to maximize the mechanical power response to a chemical energy stimulus. Optimizing response will focus on minimizing the time to peak pressure, maximizing the peak itself, and maximizing the time at peak. Selecting the corresponding highest performance cases from the parametric studies yields the input parameters shown in Table 5.4.

Table 5.4: Optimum Input Parameters for the Cylindrical Actuator

Parameter	Value
$\mathrm{pH}_e = \mathrm{pH}_{i0}$	3.0
$[\mathrm{Suc}]_e = [\mathrm{Suc}]_{i0}$	5.0 mM
$N_{\mathrm{SUT4}} e \lambda_{\mathrm{SUT4}} \sqrt{[\mathrm{W}]_e^{n_\mathrm{W}} [\mathrm{W}]_i^{n_\mathrm{W}}}$	700 μA/(mM H^+)·(mM Suc)
n_W	700
C	0.15 F
K	1.8×10^{-13} m³/N·s
A	1.72×10^{4} (μm)²
σ	1.0
k_{pump}	70 μA
ΔG_{ATP}	-1.5×10^{-20} J
v_0	0.0 V
V_0	4.91×10^{7} (μm)³
C_{10}	44.5 MPa
C_{01}	0.0 MPa
D_1	9.23×10^{-4} MPa

The above inputs increase the actuator free displacement to such an extent that the finite element analysis cannot accommodate the element distortion (Figure 5.24). The stress induced in the top plate is over an order of magnitude higher than the yield stress of PTFE. Because the baseline case predictions are within the expectations of performance in the existing experimental cylindrical actuator, this result is extraordinarily encouraging for the future of nastic materials.

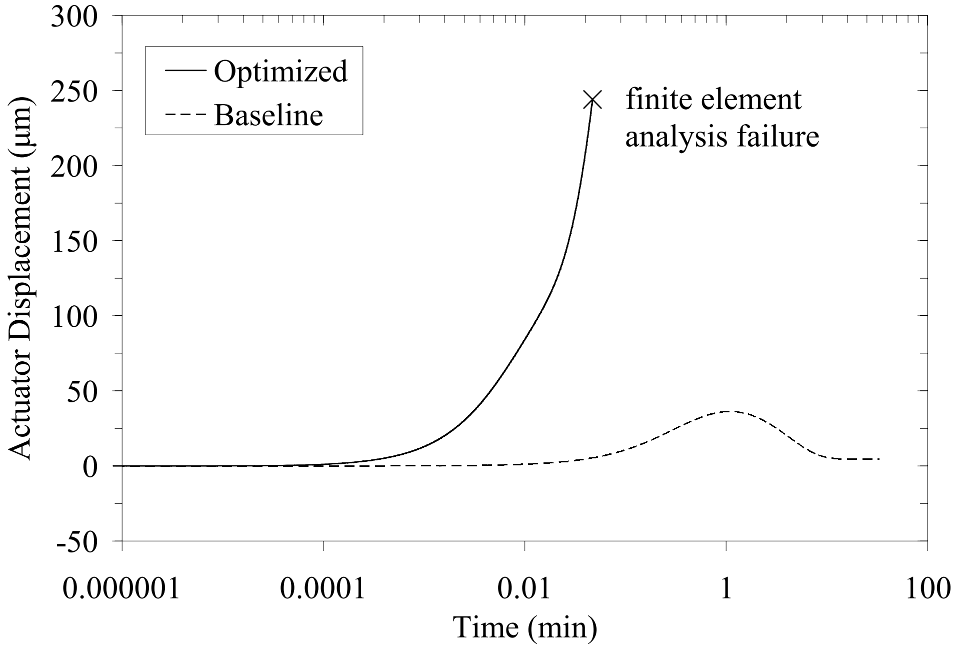


Figure 5.24: Cylindrical Actuator Free Displacement: Baseline and Optimized

5.2 SPHERICAL ACTUATOR CASE

The long-term goal of this program is to develop a material system based on spherical inclusions. Increasing the ratio of membrane surface area to inclusion volume is expected to significantly improve system power. The following is a set of parametric studies of this system based on inputs used in the cylindrical case.

5.2.1 Matrix Properties of the Spherical Actuator Element

For simplicity, the ideal gas expansion approach detailed in Section 4.3 is again employed in the parametric studies of the Mooney-Rivlin coefficients. These studies implement the parametric study capability within ABAQUS. Studies are performed for a free displacement case and a blocked force case where expansion in the vertical direction is constrained. A parametric study is carried out for each of the three Mooney-Rivlin coefficients: C_{10}, C_{01}, and D_1. As before, the initial pressure and temperature inside the inclusion are 0.1 MPa and 293 K with the temperature increasing linearly to 60,000 K over a time of 20 s.

Varying the ratio C_{01}/C_{10} has little effect on the material response in either the free displacement or blocked force cases. The pressure-volume response in the free displacement case is nearly linear with the only variation with C_{01}/C_{10} occurring at large volumes (Figure 5.25a). Blocking the spherical actuator element causes the pressure-volume response to be shifted slightly to the left on the pressure axis (Figure 5.25b). The average radial free displacement is also modestly affected ranging from 0.00309 μm at $C_{01}/C_{10} = 1.0$ to 0.00316 μm at $C_{01}/C_{10} = 0.0$ (Figure 5.26).

The coefficient C_{10} has the most influence on the material response, especially with regard to the pressure-volume response. The pressure-volume response in the free displacement case is nearly linear with the slope increasing as C_{10} is increased (Figure 5.27a). Blocking the spherical actuator element has little effect on the pressure-volume response causing the pressure-volume curves to be shifted slightly to the left on the pressure axis (Figure 5.27b). The average radial free displacement varies significantly with C_{10} ranging from 0.00611 μm at $C_{10} = 44.5\,\text{MPa}$ to 0.00162 μm at $C_{10} = 178\,\text{MPa}$ (Figure 5.28).

Varying the coefficient D_1 has little effect on the material response in either the free displacement or blocked force cases. The pressure-volume response in the free displacement case is nearly linear with the only variation with D_1 occurring at large volumes (Figure 5.29a). Blocking the spherical actuator element causes the pressure-volume curves to be shifted slightly to the left (Figure 5.29b). The average radial free displacement is also modestly affected ranging from 0.00313 μm at $D_1 = 9.23\times10^{-5}\,\text{MPa}^{-1}$ to 0.00344 μm at $D_1 = 9.23\times10^{-3}\,\text{MPa}^{-1}$ (Figure 5.30).

The effects of varying the Mooney-Rivlin coefficients in the spherical actuator are similar to those in the cylindrical actuator case. The coefficient C_{10} has the greatest impact on the pressure-volume response of the spherical actuator element in both the free displacement and blocked force cases with the slope of the pressure-volume curves increasing as C_{10} is decreased. Varying the coefficient D_1 has a modest impact on the pressure-volume response in both cases with the slope of the pressure-volume curves increasing as D_1 is increased. The value of D_1 has less of an impact on material response than in the cylindrical actuator case. The ratio C_{01}/C_{10}, as before, has negligible impact on the material response. Although useful in determining general trends, these studies should not be used for general consideration as the relative volume fraction is not yet optimized.

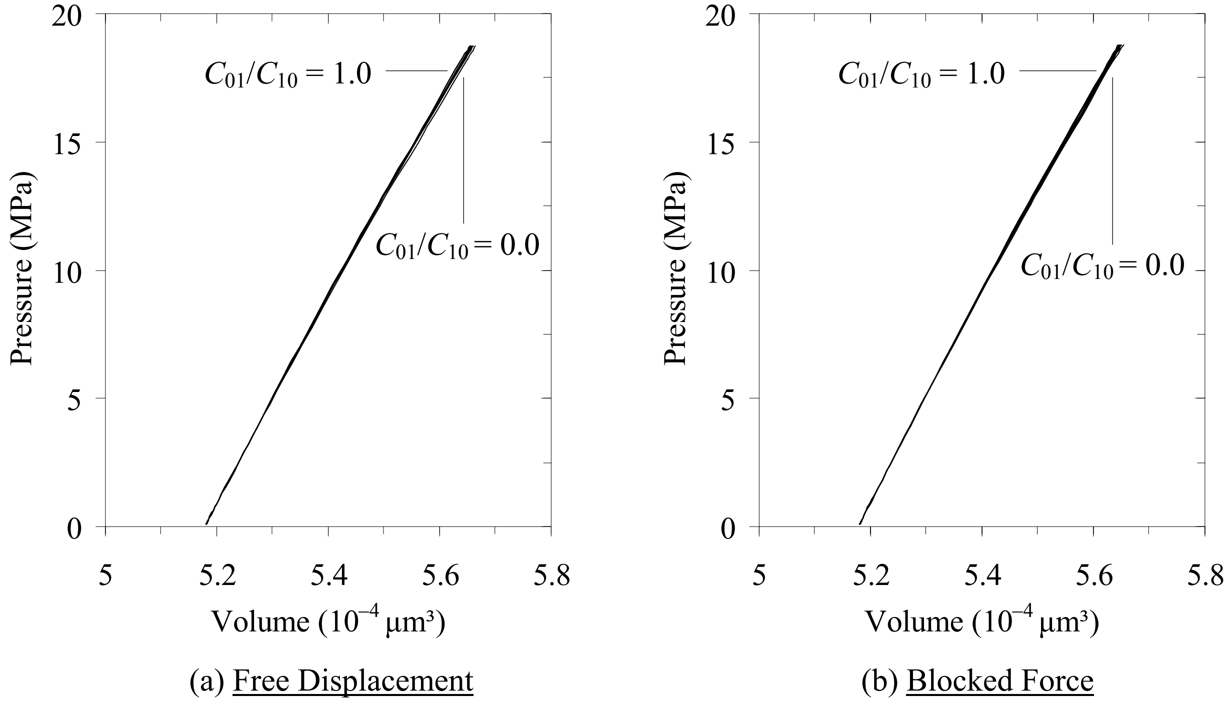


Figure 5.25: Spherical Actuator Pressure-Volume Response for Varied C_{01}/C_{10}

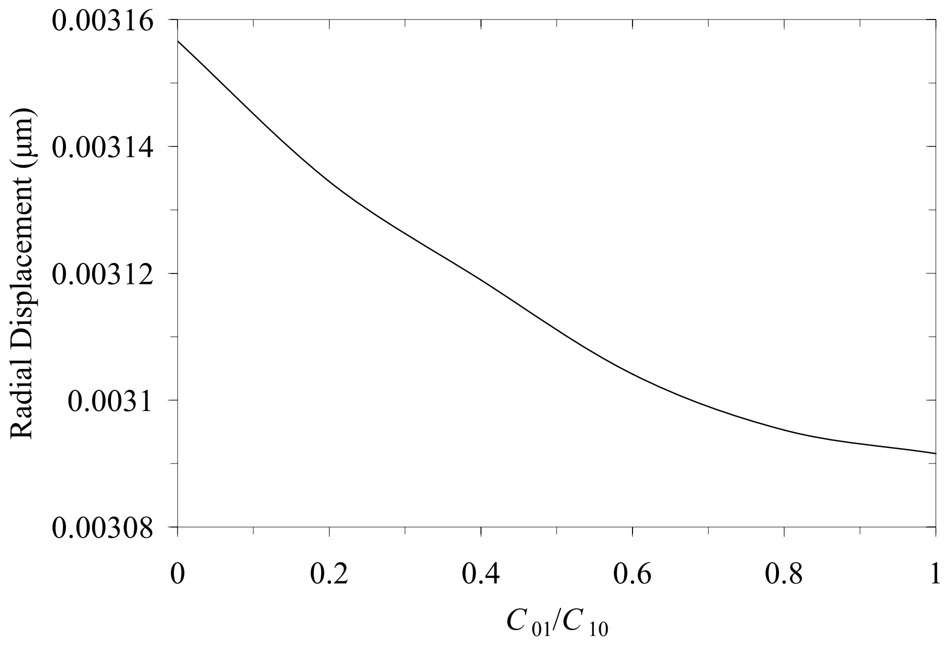


Figure 5.26: Spherical Actuator Average Radial Free Displacement for Varied C_{01}/C_{10}

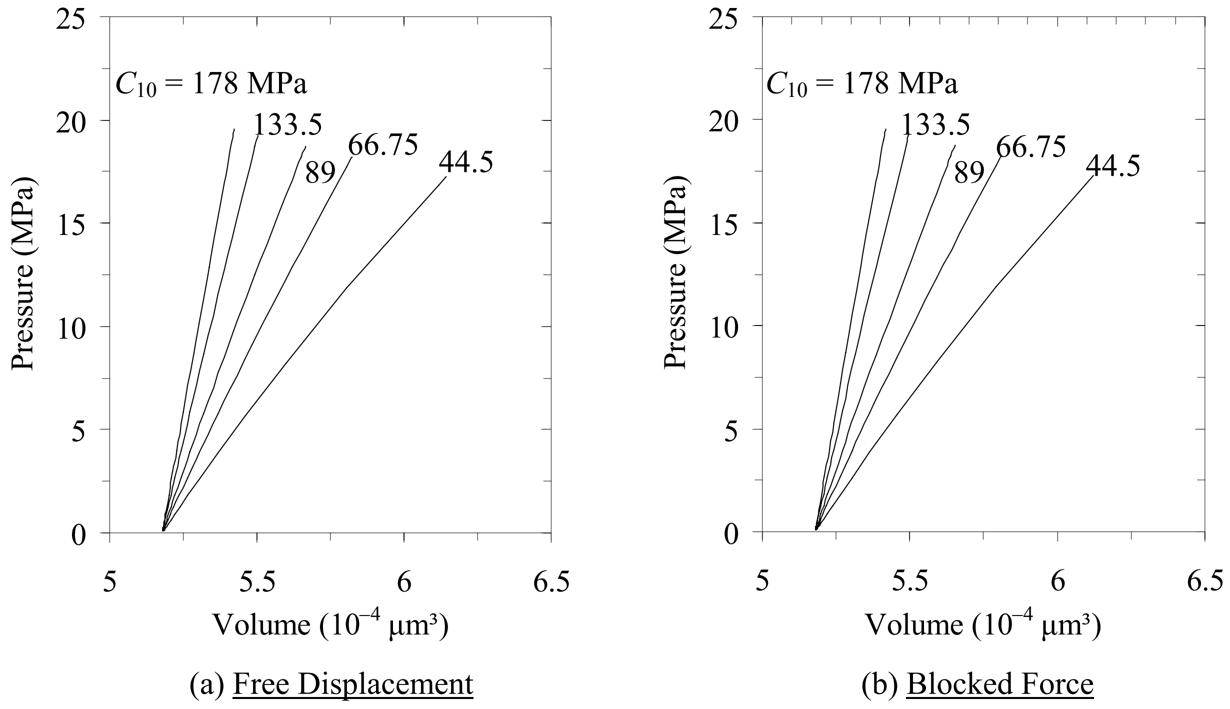


Figure 5.27: Spherical Actuator Pressure-Volume Response for Varied C_{10}

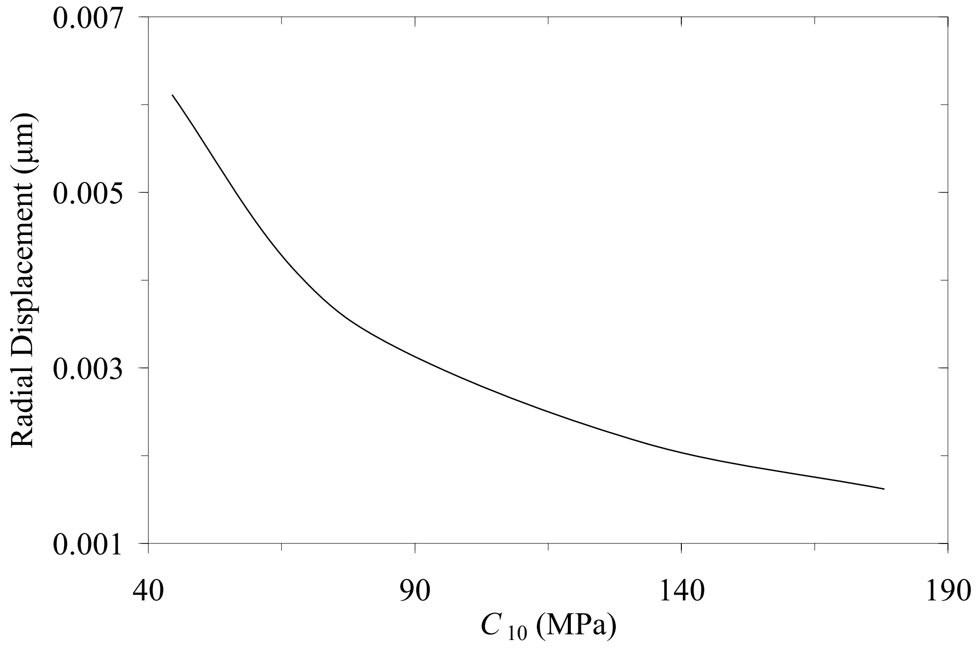


Figure 5.28: Spherical Actuator Average Radial Free Displacement for Varied C_{10}

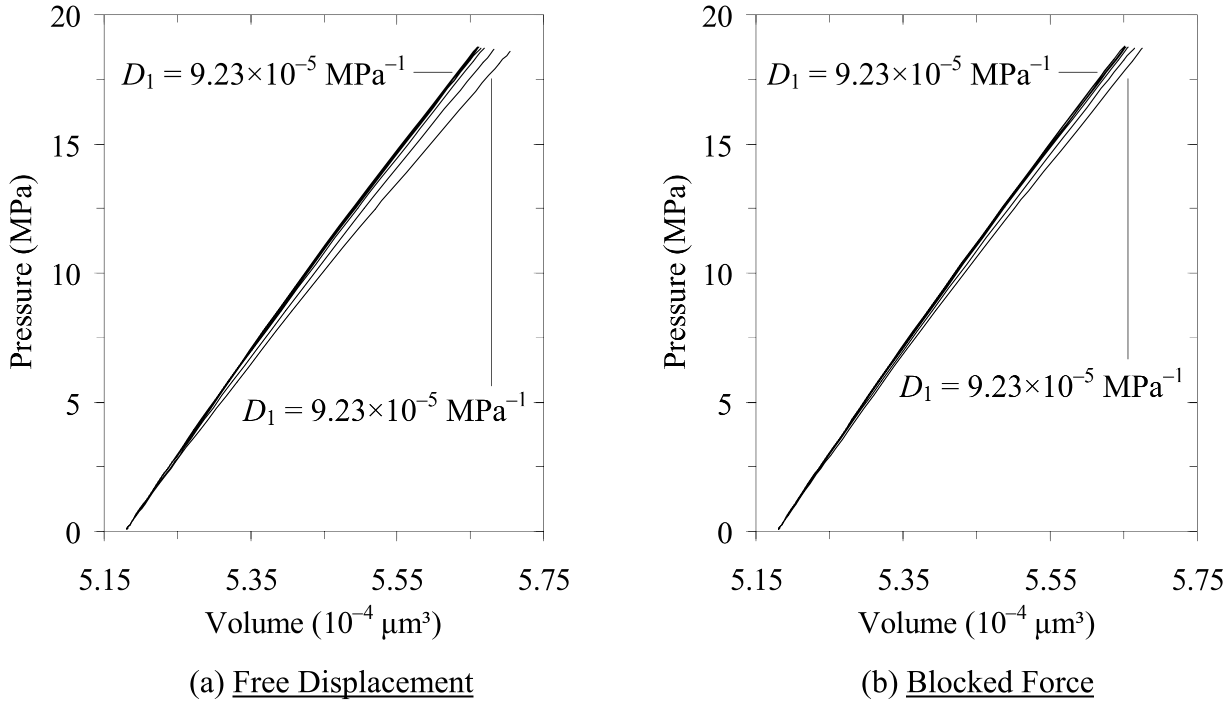


Figure 5.29: Spherical Actuator Pressure-Volume Response for Varied D_1

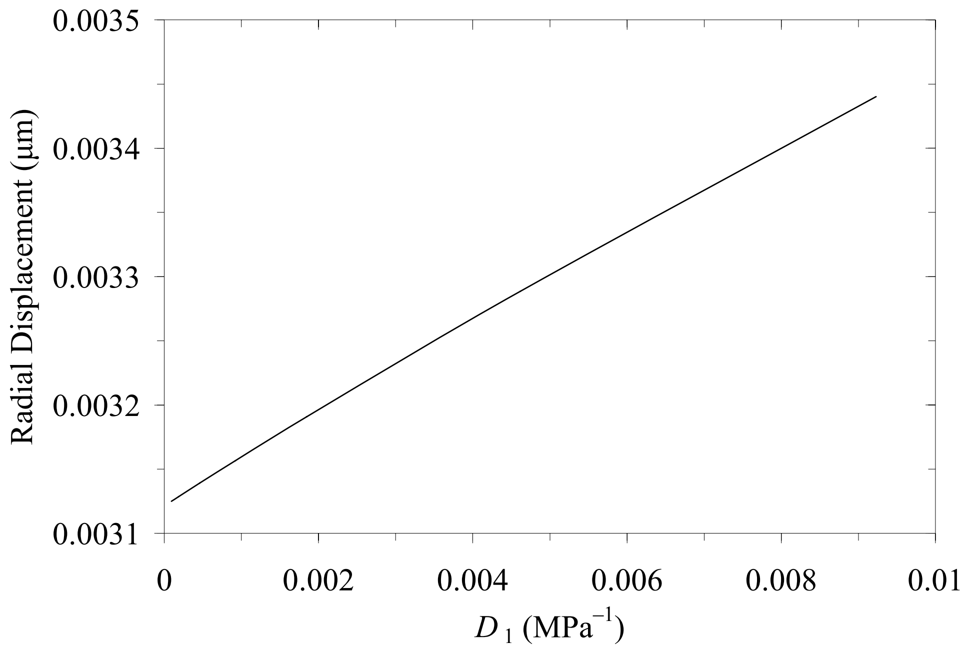


Figure 5.30: Spherical Actuator Average Radial Free Displacement for Varied D_1

5.2.2 Coupled Model Parametric Studies of the Spherical Actuator

A series of parametric studies has been performed using the spherical actuator element (Figure 4.4). The inputs for the baseline case are based upon those used in the assay cup experiment simulation presented in Section 4.2 in the same manner as for the cylindrical actuator. Based on symmetry, the input parameters are for one eighth of the total membrane area. The input parameters are given in Table 5.5. These parameters are scaled according to membrane area from the baseline inputs given in Table 5.1. A summary of the results from the parametric studies is given in Table 5.6.

Table 5.5: Input Parameters for the Spherical Actuator Baseline Case

Parameter	Value
$\mathrm{pH}_\mathrm{e} = \mathrm{pH}_\mathrm{i0}$	5.0
$[\mathrm{Suc}]_\mathrm{e} = [\mathrm{Suc}]_\mathrm{i0}$	1.0 mM
$N_\mathrm{SUT4} e \lambda_\mathrm{SUT4} \sqrt{[\mathrm{W}]_\mathrm{e}^{n_\mathrm{W}} [\mathrm{W}]_\mathrm{i}^{n_\mathrm{W}}}$	65 pA/(mM H^+)·(mM Suc)
n_W	350
C	0.105 μF
K	1.65×10^{-18} m^3/N·s
A	1.57×10^{-2} $(\mu\mathrm{m})^2$
σ	1.0
k_pump	65 pA
ΔG_ATP	-1.0×10^{-20} J
v_0	0.0 V
V_0	5.18×10^{-4} $(\mu\mathrm{m})^3$
C_{10}	89 MPa
C_{01}	0.0 MPa
D_1	9.23×10^{-4} MPa

The time dependent response for the baseline case is illustrated in Figures 5.31 through 5.47. As illustrated in Figures 5.31 through 5.34, the response is analogous to the cylindrical actuator case with the following significant differences: (i) the material is response is much faster, (ii) the relative change in volume at both peak volume and equilibrium are significantly lower, and (iii) the peak volume is sustained for a longer period of time. The expansion of the inclusion induces a maximum Mises stress in the surrounding matrix of 22.8 MPa at peak volume and 0.0392 MPa at equilibrium (Figure 5.35). The inclusion pressure is 16.3 MPa at peak volume and 0.0287 MPa at equilibrium (Figure 5.34). The actuator free displacement is 1.63 nm at peak volume and 2.67 pm at equilibrium.

Blocking the spherical actuator element has little effect on the transport response. The inclusion volume is nearly identical to that generated in the free displacement case (Figure 5.36). This is because the pressure-volume curves are virtually the same for the free displacement and blocked force cases (Figure 5.27). The peak hydrostatic pressure of 16.7 MPa generated is slightly higher than the 16.3 MPa generated in the free displacement case (Figure 5.37). The maximum Mises stress of 24.4 MPa at peak volume and 0.0411 MPa at equilibrium are higher than the 22.8 MPa at peak volume and 0.0392 MPa at equilibrium in the free displacement case due to the triaxial stress state in the top and bottom surfaces of the inclusion boundary (Figure 5.38).

Table 5.6: Summary of Parametric Study Results for the Spherical Actuator[†]

Parameter		Peak			Equilibrium		
Symbol	Value	Time [10^{-3} min]	Pressure[‡] [MPa]	Volume [10^{-4} μm³]	Time [min]	Pressure[‡] [MPa]	Volume [10^{-4} μm³]
Baseline		0.12	16.3	5.5968	8.3	0.0287	5.1795
ΔG_{ATP}	-0.5×10^{-20} J	0.067	3.45	5.2620	11	0.00605	5.1790
	-1.5×10^{-20} J	1.6	997	8.0643	7.4	0.0139	5.1821
$\mathrm{pH}_{e,i0}$	3	0.0035	14.5	5.5469	11	0.0233	5.1794
	8	110	15.5	5.5734	11	0.0288	5.1795
$[\mathrm{Suc}]_{e,i0}$	0.2 mM	0.31	3.05	5.2521	17	0.00536	5.1790
	5.0 mM	0.71	226000	13.125	2	0.297	5.1859
SUT4	0.1×	2.0	16.3	5.5964	15	0.0287	5.1795
	10×	0.20	16.3	5.5970	15	0.0287	5.1795
K	1.65×10^{-19} m³/N·s	0.20	16.3	5.5970	75	0.0287	5.1795
	1.65×10^{-17} m³/N·s	0.20	16.3	5.5964	1.4	0.0287	5.1795
n_{W}	0	-	-	-	10	0.0265	5.1795
	700	0.24	39.3	6.2060	10	0.0316	5.1796
C_{10}	44.5 MPa	10	8.20	5.5969	25	0.0287	5.1802
	178 MPa	2.0	32.3	5.5968	6	0.0287	5.1792

[†] Free displacement.
[‡] Hydrostatic pressure generated inside the inclusion.

System power is a function of ATP free energy (Figure 5.39). The ATP concentration is regulated such that ΔG_{ATP} varies between −0.5 and -1.5×10^{-20} J. The peak and equilibrium volumes range from 0.000526 and 0.0005179 μm³ for the -0.5×10^{-20} J case to 0.000806 and

0.0005182 μm³ for the -1.5×10^{-20} J case respectively. Decreasing ΔG_{ATP} decreases the time required to reach peak volume and has little effect on the response time to equilibrium.

Varying the external and initial internal pH values impacts both the pump and cotransporter reaction rates (Figure 5.40). The equilibrium volume varies only slightly across the pH range with a value of around 0.00051794 μm³. The peak volume varies modestly from 0.000560 μm³ except in the pH 3 case where it is 0.000555 μm³. The pH has no impact on the time needed to reach equilibrium, which is around 11 min. However, the time to peak volume varies considerably, ranging from 3.5×10^{-6} min at pH 3 to 0.11 min at pH 8. The time at which volume relaxation begins varies from 5.8×10^{-3} min at pH 3, decreasing to 2.2×10^{-3} min at pH 4 then increasing to 0.11 min at pH 8.

The external and initial internal sucrose concentrations affect both the cotransporter reaction rate and the osmotic pressure (Figure 5.41). The result is variation in the peak and equilibrium volumes from 0.000525 and 0.0005179 μm³ for the 0.2 mM case to 0.00131 and 0.0005186 μm for the 5.0 mM case respectively. Also, the response time to equilibrium is increased from 2 min for the 5.0 mM case to 17 min for the 0.2 mM case. An interesting feature of the volume curve for the 5.0 mM case is the knee that occurs during relaxation. This occurs because the high sucrose concentration generated by the SUT4 and the outward flux of water causes the SUT4 to reverse and pump sucrose out of the inclusion. This reduces the sucrose concentration and hence the osmotic pressure which slows the rate at which water flows out of the inclusion. As the sucrose concentration drops, the SUT4 current goes to zero, allowing the relaxation process to continue to equilibrium.

Increasing the density of SUT4 cotransporters in the membrane increases the transporter current, decreasing the time required to reach the peak volume and increasing the total time at peak volume (Figure 5.42). The time required to reach peak volume varies from 0.0002 min for the 10×Baseline case to 0.002 min for the 0.1×Baseline case respectively. The response time to equilibrium is the same in all cases at 15 min.

The membrane permeability to water, K, impacts the rate of water flux in the presence of a hydrostatic or osmotic pressure gradient (Figure 5.43). The equilibrium volume is constant across range of permeabilities with a value of 0.0005195 μm³. The peak volume is achieved at 0.0002 min in all cases. However, the time *at* peak volume varies considerably. The time at which the inclusion volume begins to relax toward equilibrium varies from 0.0003 min (equilibrium at 1.2 min) in the 1.65×10^{-17} m³/N·s case to 0.03 min (equilibrium at 75 min) in the 1.65×10^{-19} m³/N·s case.

Varying the density of H^+ pumps in the membrane has little effect on either the volume or response time (Figure 5.44). This is directly analogous for the parallel study for the cylindrical actuator given in Section 5.1.2.

Varying the SUT4 water stiochiometry increases the active water flux which affects the peak volume (Figure 5.45). The peak volume ranges from 0.00034 μm³ with a stiochiometry of 700 to nonexistent with no active water transport.

The membrane capacitance, C, which has a considerable impact on the predicted assay cup transport response as presented in Section 4.2, has no effect on the spherical actuator response (Figure 5.46). This is directly analogous for the parallel study for the cylindrical actuator given in Section 5.1.2.

The predominant effect of varying the stiffness of the polymer matrix, C_{10}, is variation in time over which the peak volume is maintained (Figure 5.47). The time required to reach the peak volume is unaffected. The time at which the inclusion volume begins to relax and the response time to equilibrium range from 0.002 and 6 min for the 178 MPa case to 0.01 and 25 min for the 44.5 MPa case respectively.

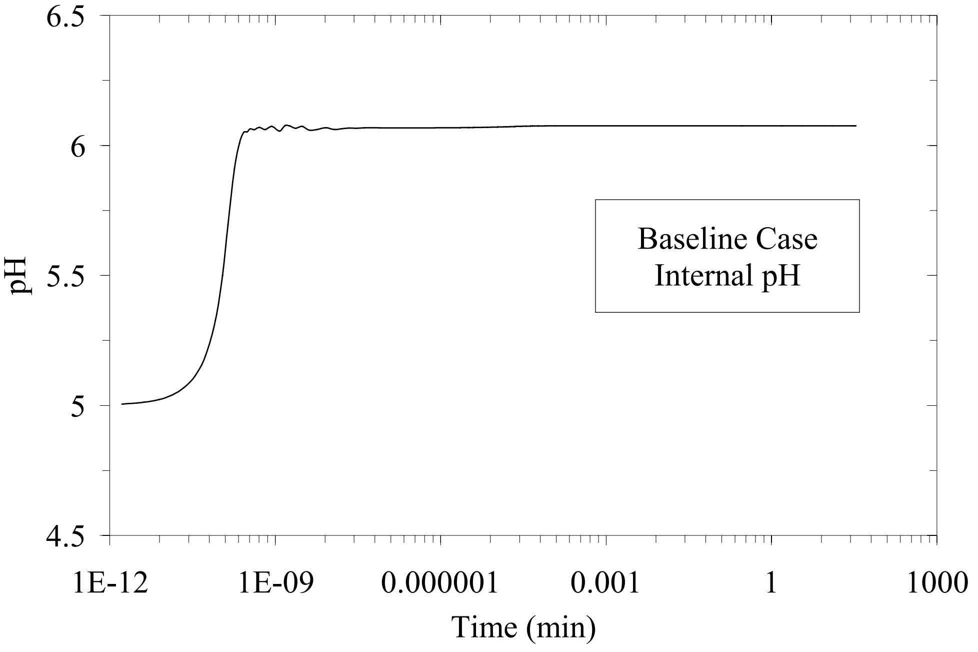


Figure 5.31: Spherical Actuator Baseline pH – The pH rises rapidly in the first 100 μs after the introduction of ATP but levels off with the establishment of a membrane potential and the activation of the SUT4.

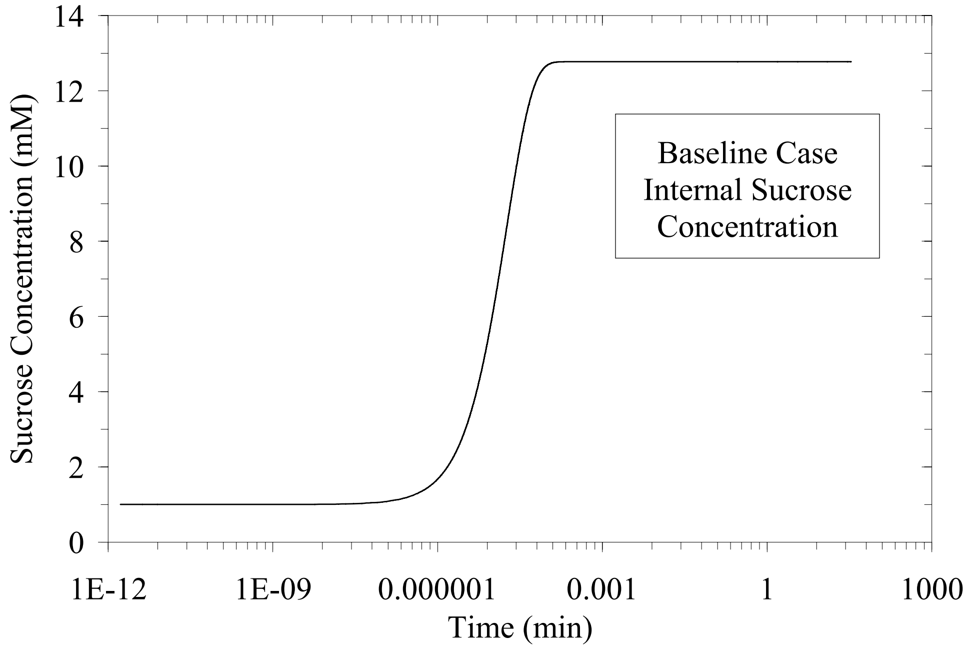


Figure 5.32: Spherical Actuator Baseline Sucrose Concentration

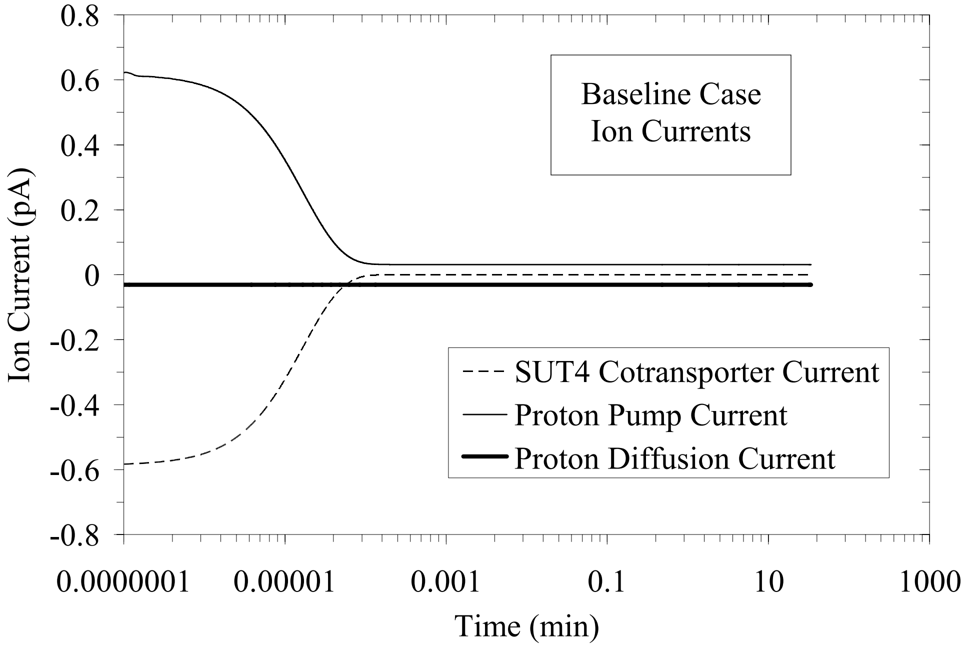


Figure 5.33: Spherical Actuator Baseline Transporter Currents – Electric currents generated by the biological transporters and volumetric flux of water across the membrane for the baseline case. Because of the rapid establishment of a membrane potential, the pH gradient remains relatively constant and the H^+ pump, SUT4, and proton diffusion currents cancel each other.

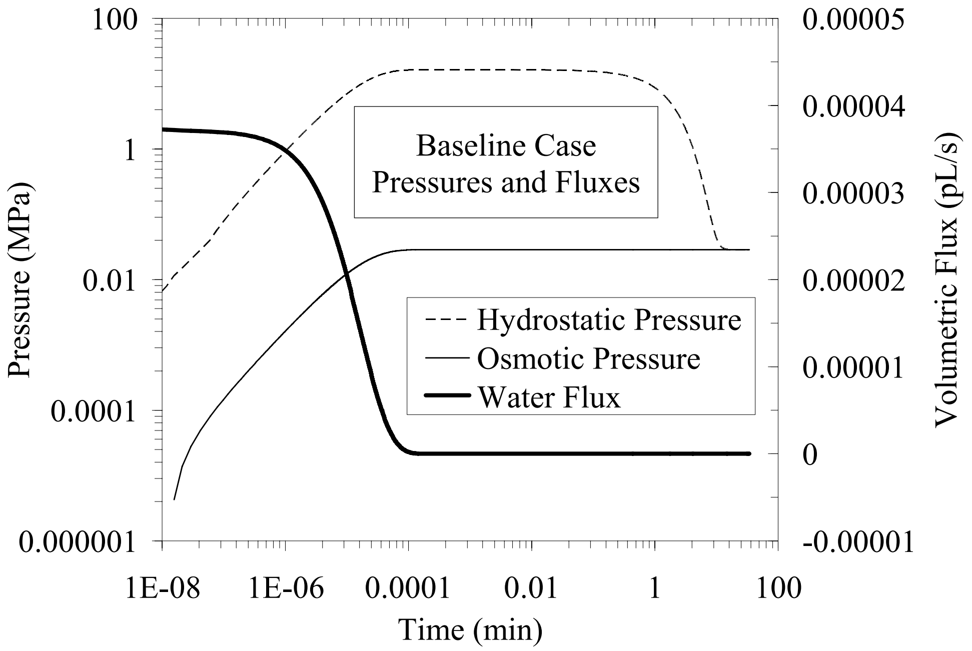
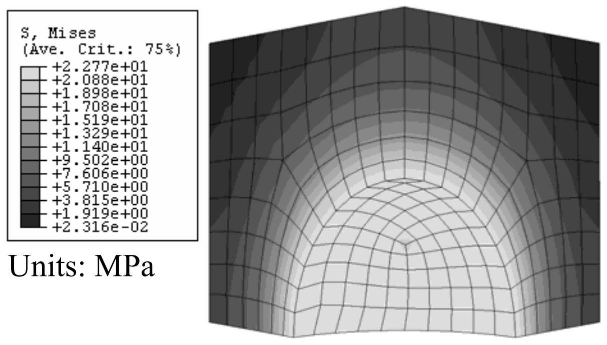
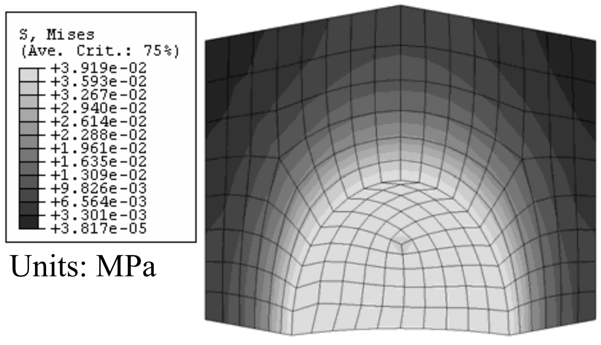


Figure 5.34: Spherical Actuator Baseline Pressures and Fluxes – Inclusion hydrostatic and osmotic pressures, and volumetric flux of water for the baseline case. The volumetric water flux is proportional to both the difference between the hydrostatic and osmotic pressures and the SUT4 current.



(a) Peak Pressure



(b) Equilibrium

Figure 5.35: Spherical Actuator Free Displacement Mises Stress: Peak Pressure and Equilibrium

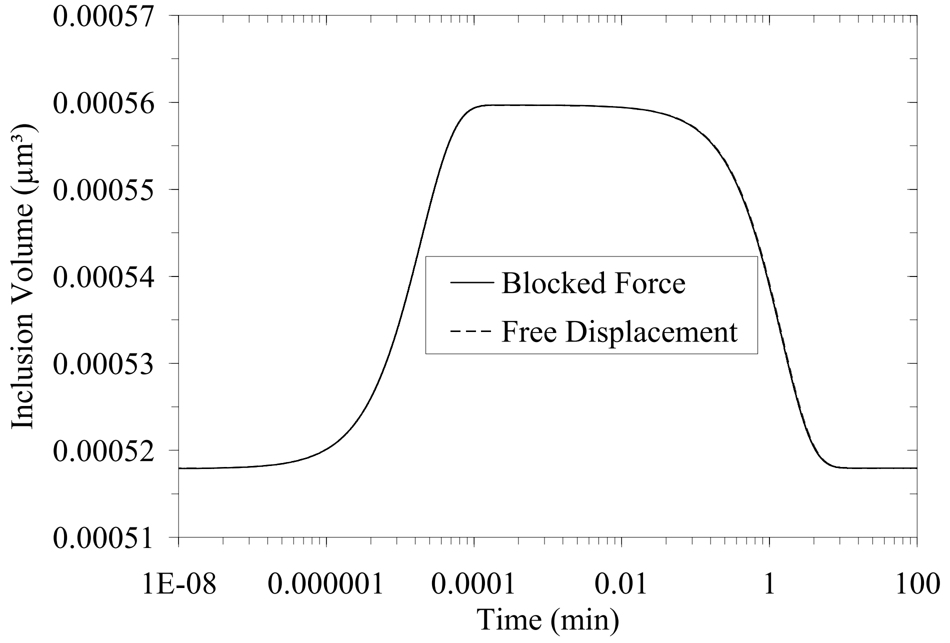


Figure 5.36: Spherical Actuator Inclusion Volume: Blocked Force and Free Displacement

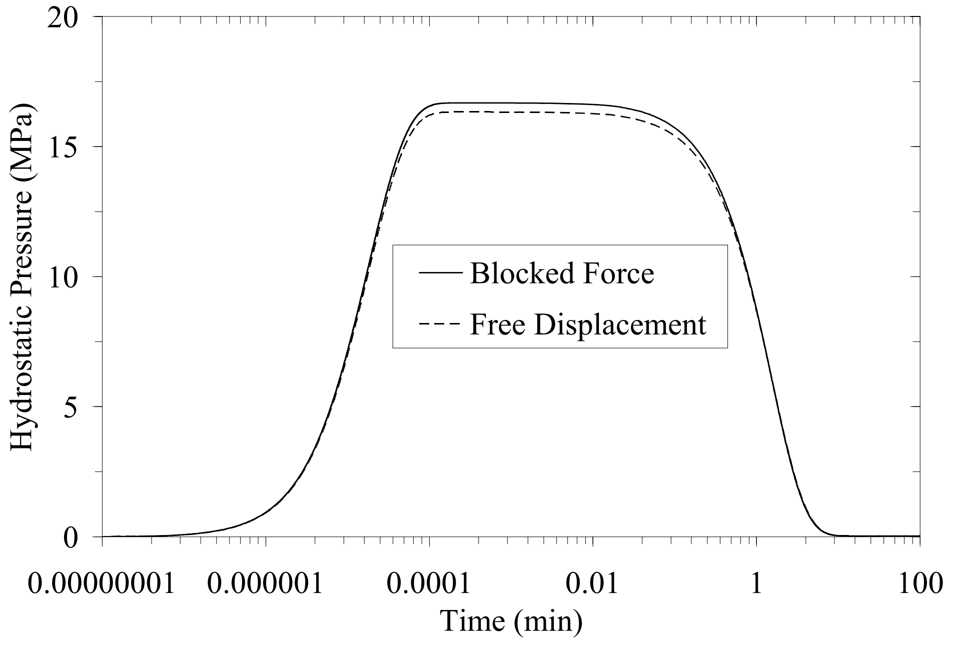
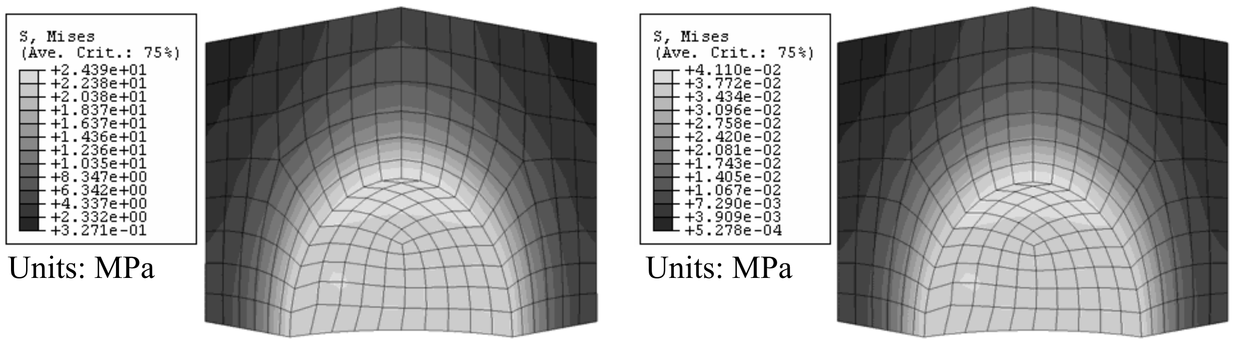


Figure 5.37: Spherical Actuator Hydrostatic Pressure: Blocked Force and Free Displacement



(a) Peak Pressure (b) Equilibrium

Figure 5.38: Spherical Actuator Blocked Force Mises Stress: Peak Pressure and Equilibrium

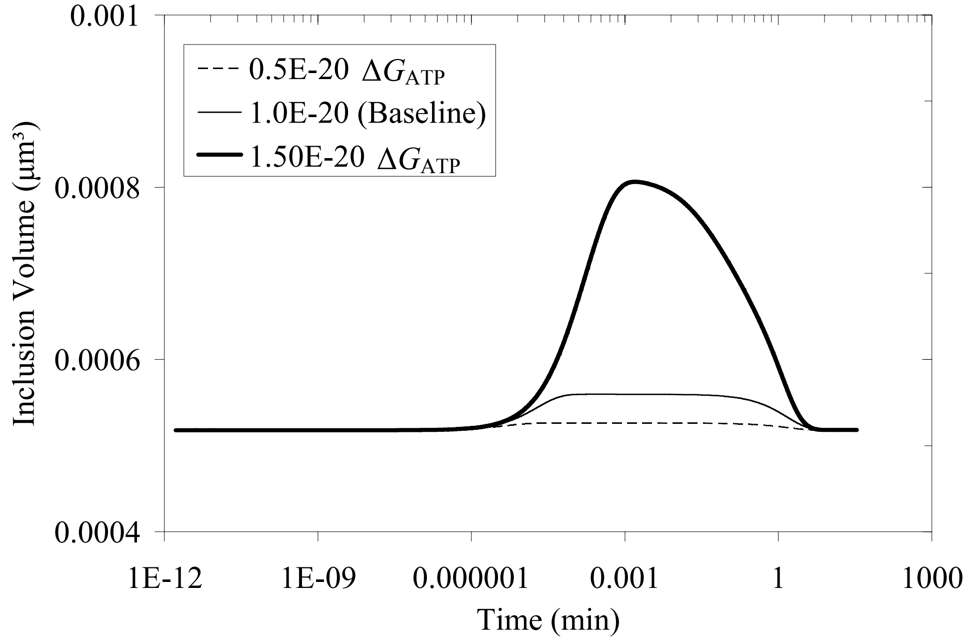


Figure 5.39: Spherical Actuator Volume for Varied ΔG_{ATP}

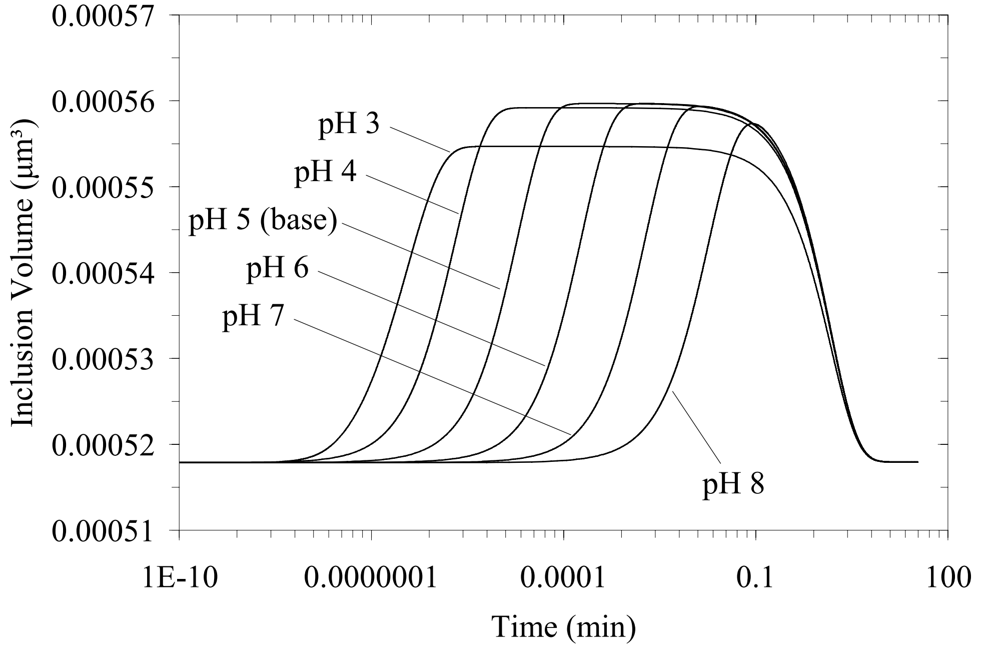


Figure 5.40: Spherical Actuator Volume for Varied pH – External and initial internal pH values varied together to avoid an initial pH gradient.

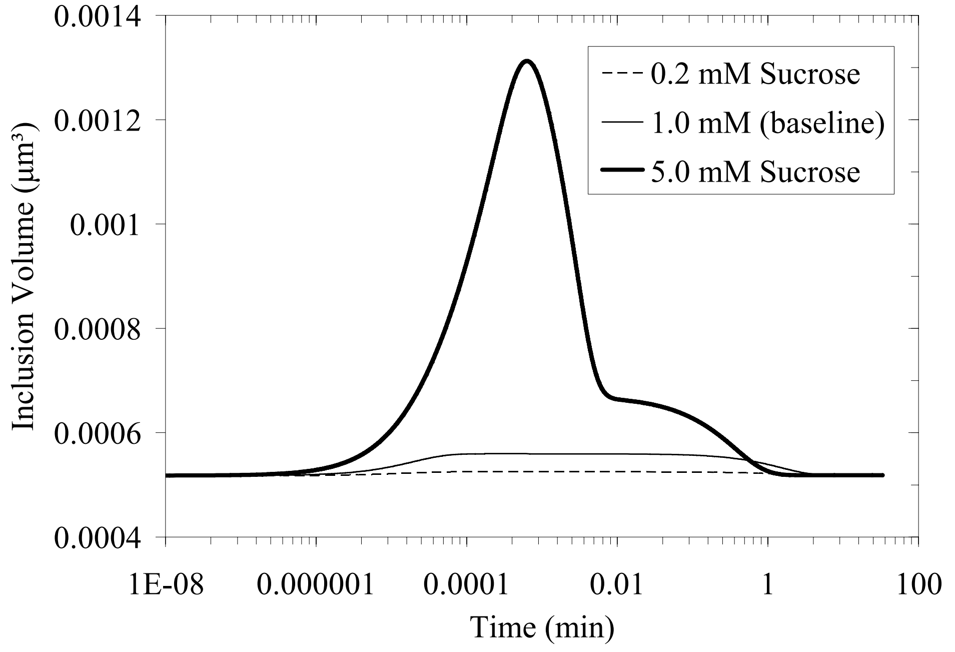


Figure 5.41: Spherical Actuator Volume for Varied Sucrose Concentration – External and initial internal sucrose concentrations varied together to avoid an initial sucrose gradient.

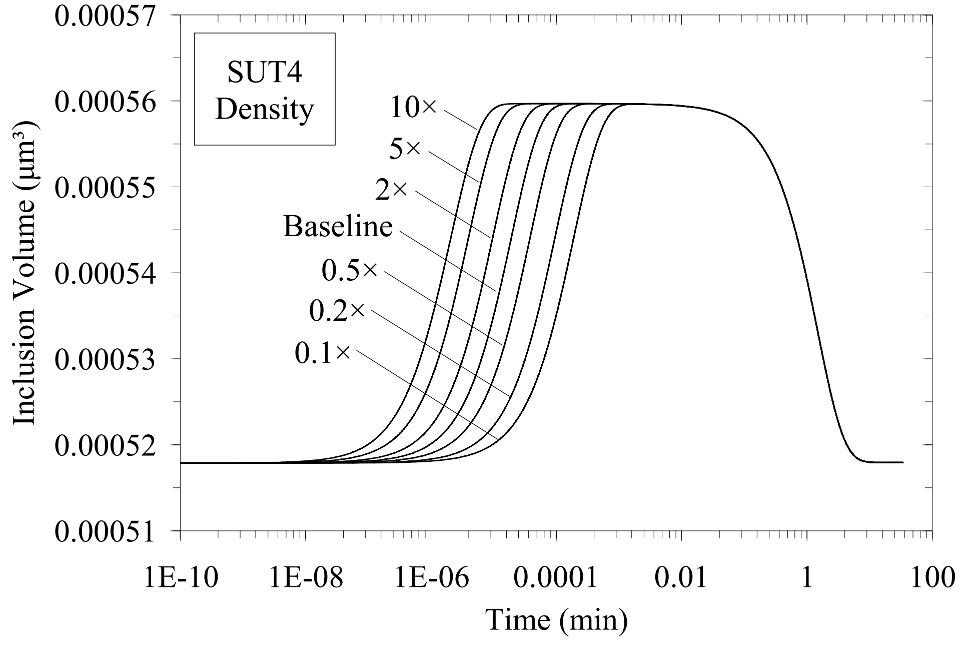


Figure 5.42: Spherical Actuator Volume for Varied SUT4 Cotransporter Density

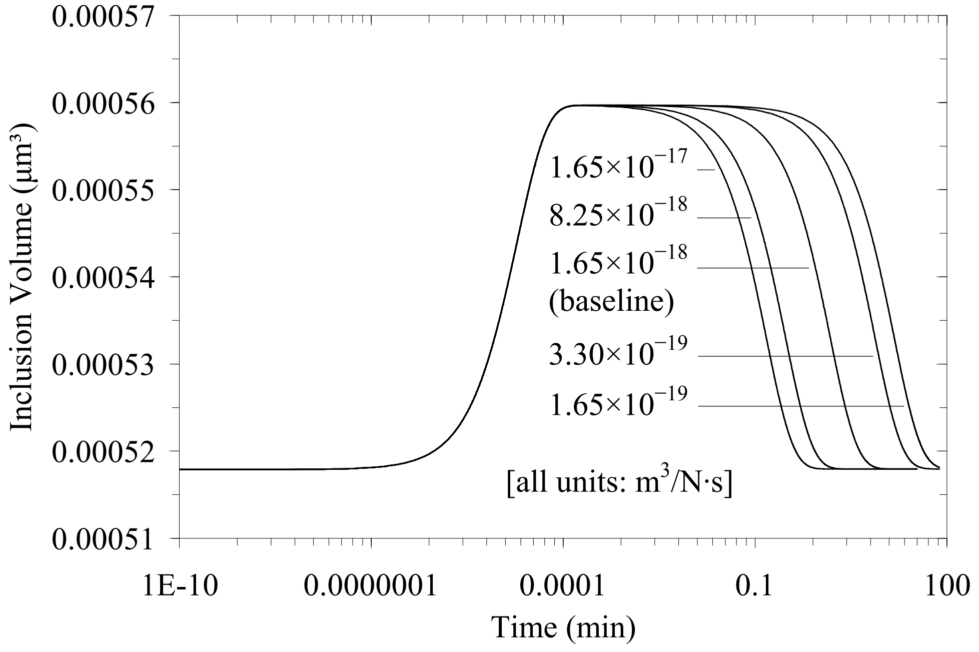


Figure 5.43: Spherical Actuator Volume for Varied Membrane Water Permeability

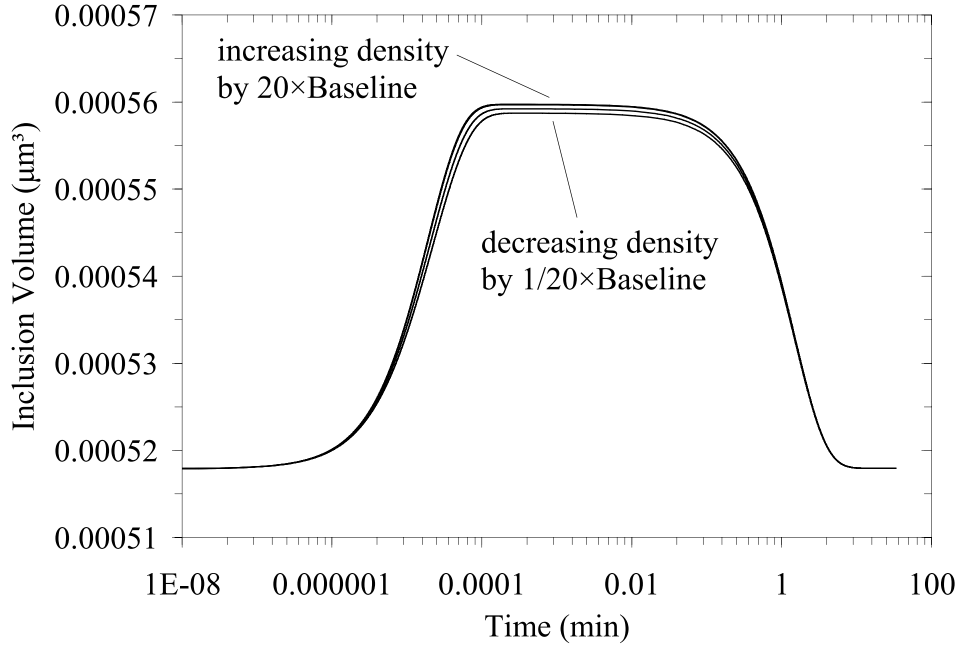


Figure 5.44: Spherical Actuator Volume for Varied H^+ Pump Density

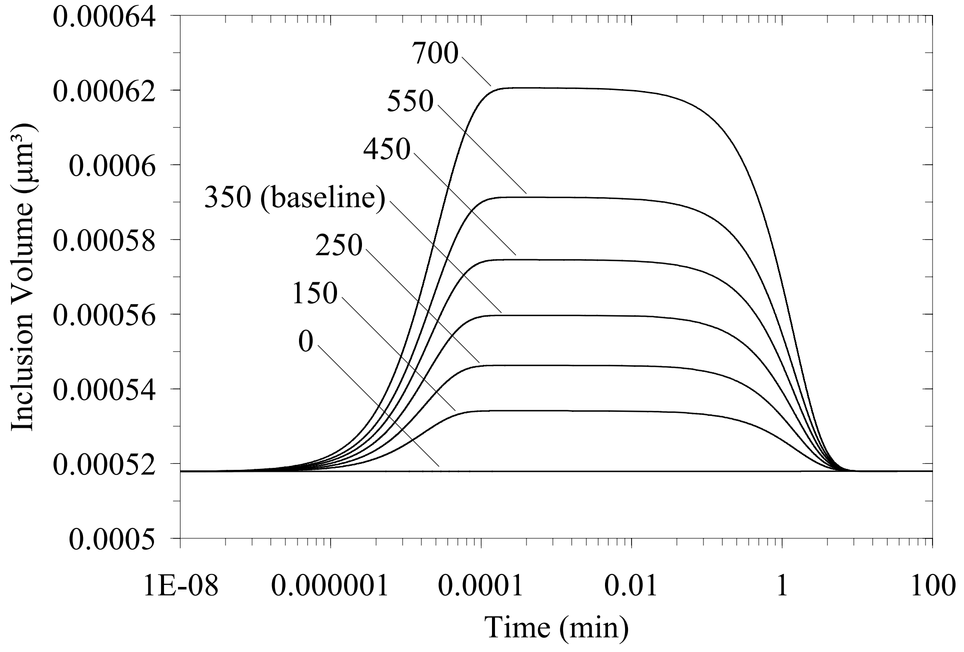


Figure 5.45: Spherical Actuator Volume for Varied SUT4 Water Stoichiometry

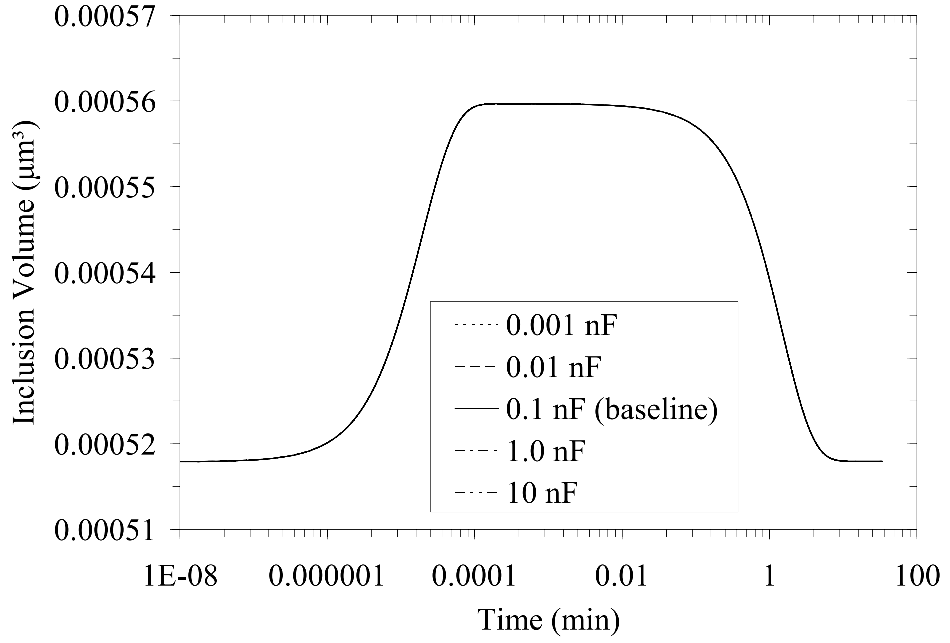


Figure 5.46: Spherical Actuator Volume for Varied Membrane Capacitance – The membrane capacitance has no effect on the displacement or response time.

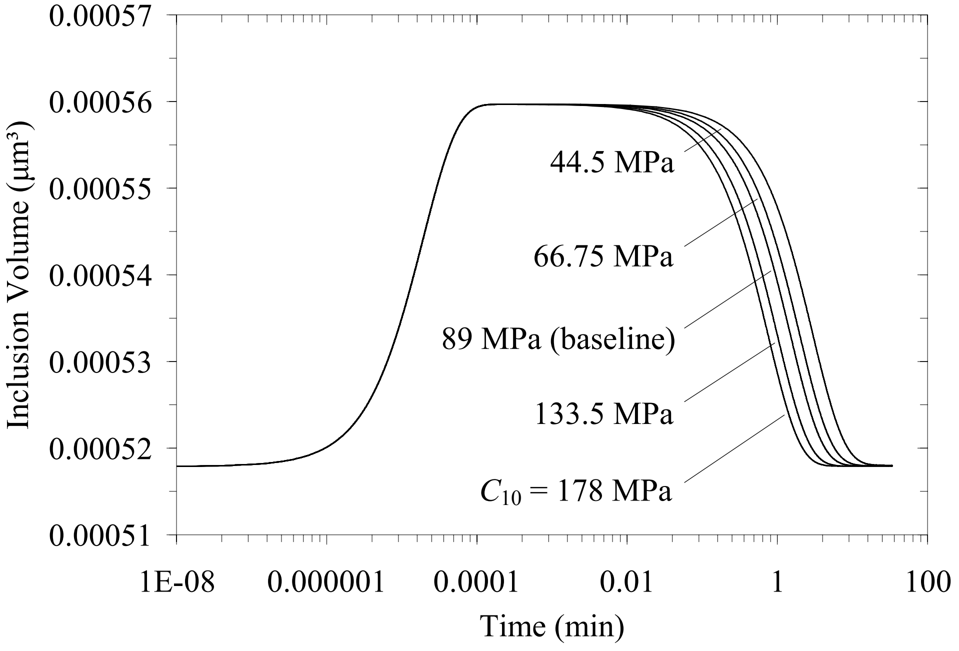


Figure 5.47: Spherical Actuator Volume for Varied Matrix Stiffness

5.2.3 Discussion of Spherical Actuator Parametric Studies

The spherical actuator parametric studies again demonstrate the significant impact of the input parameters, but with the added observation that these effects are also a function of system geometry. Many of the same trends of the cylindrical actuator case are again observed, but with some notable differences; the most significant are (1) the decreased time to peak pressure and (2) the increased time at peak pressure. A summary of the spherical actuator parametric studies is as follows: Reducing the membrane water permeability decreases the time at peak volume, which has the effect of decreasing the response time to equilibrium. Decreasing the external and initial internal pH increases the time at peak volume. Increasing the ATP free energy and the external and initial internal sucrose concentrations have the greatest impact on equilibrium volume. Increasing the SUT4 cotransporter density can decrease the time needed to reach peak volume. Increasing the stiffness of the matrix decreases both the peak and equilibrium volumes as well as the material response time. It is also again found that the H^+ pump density and membrane capacitance have no significant effect on the displacement or response time. The conclusions from the parametric studies are summarized in Table 5.7.

Table 5.7: Summary of Parametric Study Conclusions for the Spherical Actuator

Optimization	Parameters to Increase[†]	Parameters to Decrease[†]
Maximize Peak Volume	• Sucrose Concentration • ATP Free Energy • SUT4 Water Stiochiometry	
Maximize Equilibrium Volume	• Sucrose Concentration • ATP Free Energy	• Matrix Stiffness
Minimize Time to Peak Volume	• SUT4 Density • Sucrose Concentration	• pH • ATP Free Energy
Maximize Time at Peak Volume	• SUT4 Density	• pH • ATP Free Energy • Membrane Water Permeability • Matrix Stiffness
Minimize Time to Equilibrium	• Membrane Water Permeability • Matrix Stiffness	

[†] Parameters listed in order from greatest to least impact on response.

In line with the ultimate nastic program goal of engineering a new, optimized class of active materials for application as a skin for a morphing aircraft, an optimized set of input parameters is considered. Again, the goal is to minimize the time to peak response, maximize the peak response itself, and maximize the time at peak. Selecting the corresponding 'best' responses from the parametric studies yields the input parameters shown in Table 5.8. It is worthy of mention that the 'best' parameters for this case are not identical to those for the cylindrical actuator case.

Table 5.8: Optimum Input Parameters for the Spherical Actuator

Parameter	Value
$\text{pH}_\text{e} = \text{pH}_\text{i0}$	4.0
$[\text{Suc}]_\text{e} = [\text{Suc}]_\text{i0}$	5.0 mM
$N_\text{SUT4} e \lambda_\text{SUT4} \sqrt{[\text{W}]_\text{e}^{n_\text{W}} [\text{W}]_\text{i}^{n_\text{W}}}$	650 pA/(mM H^+)·(mM Suc)
n_W	700
C	0.105 μF
K	1.65×10^{-19} m³/N·s
A	1.57×10^{-2} (μm)²
σ	1.0
k_pump	65 pA
ΔG_ATP	-1.5×10^{-20} J
v_0	0.0 V
V_0	5.18×10^{-4} (μm)³
C_{10}	44.5 MPa
C_{01}	0.0 MPa
D_1	9.23×10^{-4} MPa

The above inputs increase the peak volume in both the free displacement and blocked force case by 30.4 % from 0.00056 to 0.00073 μm³ (Figure 5.48). The inclusion pressure is increased by 95.7% to 31.9 MPa at peak volume and by 143% to 0.0697 MPa at equilibrium. The resulting free displacement is increased by 378% to 7.8 nm at peak volume and by 365% to 12 pm at equilibrium. The maximum Mises stress in the free displacement case is increased by 112% from 22.8 to 48.3 MPa at peak volume and by 146% from 0.039 to 0.097 MPa at equilibrium (Figure 5.49). The maximum Mises stress in the blocked force case is increased by 113% from 24.4 to 51.9 MPa at peak volume and by 132% from 0.041 to 0.095 MPa at equilibrium (Figure 5.50). From the standpoint of nastic material development, and because this system has not been geometrically optimized, the response of greatest significance is response time. In this system the time to peak is measured in milliseconds while the time at peak is measured in hours. Because a concerted effort was made to employ baseline input parameters that are consistent with the current experimental configuration, while parametric studies have focused on ranges that are experimentally achievable (albeit challenging), this optimized prediction is extraordinarily encouraging for the future of nastic materials.

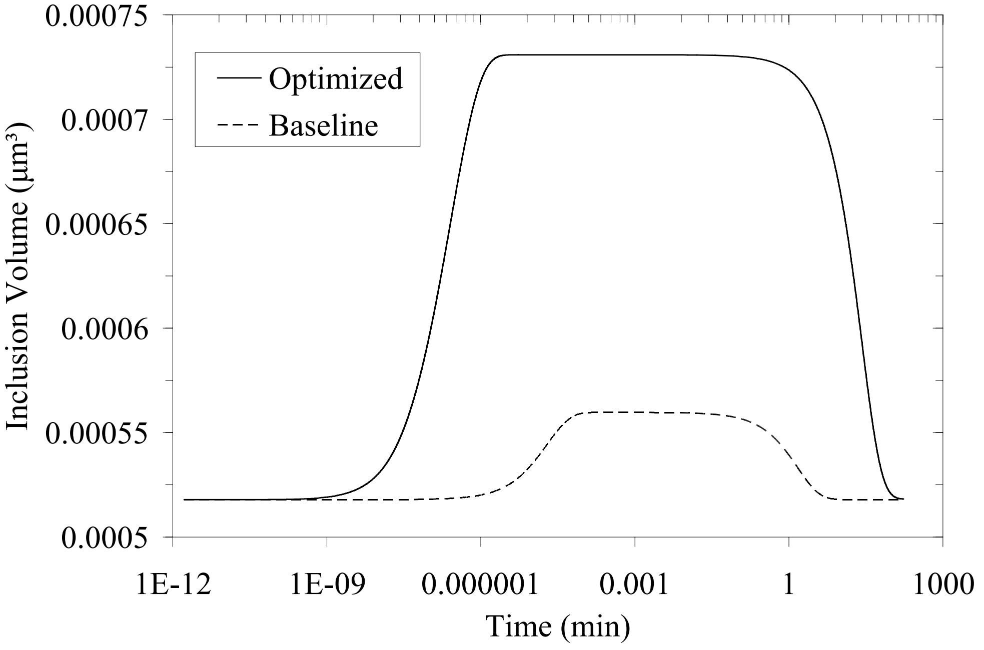
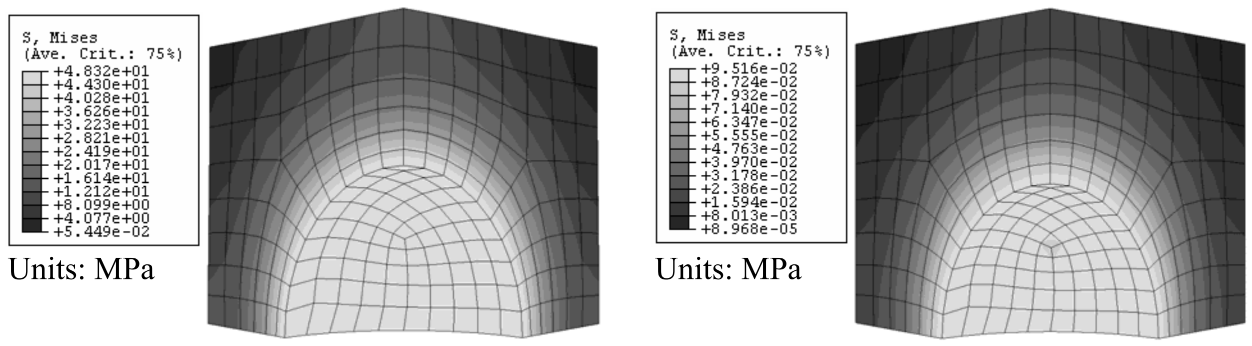
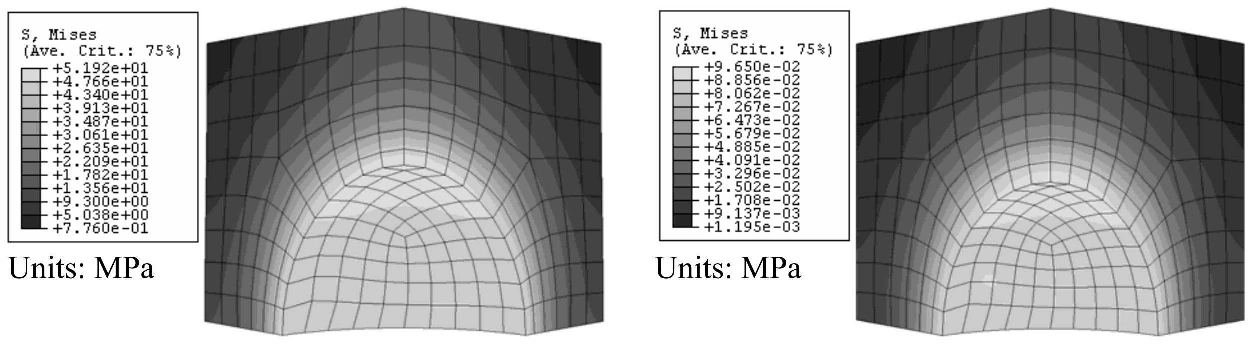


Figure 5.48: Spherical Actuator Inclusion Volume: Baseline and Optimized – The material response is the same in both the free displacement and blocked force cases.



(a) Peak Pressure (b) Equilibrium

Figure 5.49: Spherical Actuator Mises Stress for Optimized Free Displacement Case – Mises Stress at peak pressure and equilibrium for the baseline case.



(a) Peak Pressure (b) Equilibrium

Figure 5.50: Spherical Actuator Mises Stress for Optimized Blocked Force Case – Mises Stress at peak pressure and equilibrium for the baseline case.

6.0 CONCLUSIONS

In order to achieve high energy density in the development of nastic materials, informed application of biological mechanisms within a synthetic matrix is required. The proposed materials utilize controlled transport of charge and fluid across a selectively-permeable membrane to achieve bulk deformation. The nastic material considered in this work consists of synthetic membranes containing biological ion pumps surrounding fluid-filled cavities embedded within a polymer matrix. A physics-based computational model for a nastic material utilizing biological transport mechanisms is created that is able to couple the effects of biological transport with the mechanical constraint of the surrounding polymer matrix. The transport model is able to take into account the contributions of ion pumps, ion channels, ion exchangers/cotransporter, ion diffusion, and solvent diffusion. The finite element analysis of the polymer matrix is carried out using the ABAQUS/Standard 6.5 package with the fluid/matrix interaction implemented via the use of hydrostatic fluid elements. The transport model is coupled to the finite element analysis by solving the system of nonlinear ODEs formed by the transport model in parallel with the finite element analysis through the use of an ABAQUS user-subroutine set. The resulting model is capable of predicting material response for a wide range of transport/inclusion configurations. Although results comparable to experimental data can be achieved, the current membrane potential model, which assumes that all transported ions contribute to charging the membrane capacitance, does not accurately represent the physics of the membrane/solution interface. An artificially high membrane capacitance must be employed to slow the rate at which the membrane potential is established in order to account for the electrodiffusion process that takes place at the membrane/solution interface.

Results from the parametric studies suggest that maximizing the free displacement and blocked force while minimizing the response time can be achieved by maximizing the concentration of transported species as well as the density of active and secondary active transporters in the membrane. The membrane capacitance has little effect on the material response when a proton loop is established. Reducing the stiffness of the polymer matrix increases both the free displacement and blocked force.

The model development and input parameters focused on mimicking the existing nastic experiments. The model predictions of this scenario are within the range of experimental error. Parametric studies employed ranges that are experimentally achievable. Predictions of system response were subsequently made based on 'best' response from each of the parametric studies. In the cylindrical actuator case, the predicted power generation is sufficient to destroy the actuator itself. In the spherical actuator case, which represents to future of nastic material development, time to peak response is predicted to be a few milliseconds, while the time at peak response is predicted to be several hours. Both of these optimized predictions bode well for the future of nastic material development.

7.0 FUTURE WORK

7.1 TRANSPORT EQUATIONS FOR NON-IDEAL COTRANSPORT

The ideal cotransporter model presented in Section 3.1 uses a simple first order kinetics model for cotransporters and exchangers that has a basis in enzyme kinetics, but incorporates many simplifying assumptions to reduce and lump together several of the rate constants [80, 81]. A better approach to modeling a cotransporter is to use a pure enzyme kinetics model such as that proposed by Sanders *et al.* [13] (Appendix G). The Sanders model assumes that the transport sequence of the cotransporter can be modeled as a series of enzyme reactions. Using enzyme kinetics theory, Sanders is able to relate the rate constants for the various cotransporter reactions to experimentally determined kinetics parameters through a set of nonlinear algebraic equations.

The solution to the resulting system of equations, however, is nontrivial. The equations are highly nonlinear and thus can only be solved by an iterative scheme, such as the Newton-Raphson method. Such methods, however, require rather accurate initial guess values in order to converge to the desired solution. In the case of the kinetic coefficients for the Sanders cotransporter model, their magnitude is not known. The values of one or two of the kinetic constants can be determined directly from experimental data in certain cases. The relative magnitudes of many of the other constants can be determined from experimental data as outlined in Sanders *et al.* [13], however, Sanders gives no information on how to determine the magnitudes of all of the coefficients.

The Sanders cotransporter model has the potential to be more accurate than the ideal cotransporter model presented in Section 3.1.1.4 because it is based upon the actual physics of the cotransporter rather than lumped energetics. Also, the Sanders model can provide insights into the operation of the cotransporter that could be used to determine the rate limiting steps in the cotransport process. In order to make the Sanders model viable, a method must be determined to find the approximate values for all twelve of the kinetic coefficients accurately enough to solve for them using the Michaelis-Menten coefficient equations described in Appendix G. A compromise between the lumped and generalized kinetics models has been developed by Parent *et al.* [89] which accounts for the various stages of transport but with inputs that can be determined from experimental data.

Another potential approach to simulating the cotransport process is presented in Johnson *et al.* [90]. This approach involves deriving a state diagram and using statistical optimization software to determine the values of the rate constants. This approach has the potential to be readily adapted SUT4 and provide a means to determine the rate constants for a pure enzyme kinetics model.

7.2 SIMULATION OF THE DIFFUSE DOUBLE LAYER

The Hodgkin-Huxley membrane potential model presented in Section 3.1.2 does not accurately represent the physics of the diffuse double layer at the membrane/solution interfaces. This resulted in the application of an artificially high membrane capacitance in the presented model. While this did a satisfactory job of predicting response trends, a more accurate approach is desirable. A more accurate approach, from a physics standpoint, is to model ion transport as a set of electrodiffusion processes. This is accomplished by solving the coupled Nernst-Planck, continuity, and Poisson equations with an appropriate set of boundary conditions. As presented in Section 2.2, several authors have proposed methods of modeling the ion transport process in this manner with varying boundary conditions and integration approaches. One of the simplest and possibly most readily adaptable to the nastic material of these models is that of Manzanares *et al.* [51] for bulk solutions with equal concentrations of a 1:1 binary electrolyte separated by a membrane with an imposed current across it. A more generalized version of their model is presented in Appendix H using information found in [44].

The electrodiffusion model coupled with the transport model will yield a more accurate membrane potential by taking into account the physics of the membrane/solution interface. In addition, the above model will yield the concentration profiles of ions near the membrane making it possible to use the local ion concentrations at the membrane faces in the transporter equations instead of the bulk solution concentrations. Also, if the concentration of fixed charge within the lipid bilayer is incorporated into the above model, the potential profile within the membrane can be determined and used in place of the assumed linear profile employed in the ion channel and ion diffusion equations presented in Sections 2.2.1 and 2.1.7 respectively. The net result of these enhancements is a transport model that is more accurate and more complete.

7.3 INCLUSION MOSAICS

Since the ultimate goal of this modeling effort is to optimize the nastic material with respect to free displacement and blocked force, upcoming work will focus on the optimization of a single inclusion element. Once optimized, inclusion elements will be combined into three-dimensional arrays, or mosaics, in order to simulate bulk material response. Consequently, the assumption of the inclusion element being immersed in an infinite fluid reservoir will no longer be valid since it must compete with neighboring inclusion elements for available fluid. Therefore, a fluid diffusion model must be developed that takes into account the effects of matrix deformation on the permeability of the polymer. A possible solution is to adapt the fluid diffusion model for soil analysis within ABAQUS.

In order to accomplish this, the user-subroutine set described in Section 4.4.2 must be modified to handle inclusion mosaics. The user-subroutine set can be extended to handle multiple inclusions by indexing the transport model variables according to the cavity reference node number passed into the `UFLUID` subroutine by ABAQUS and adapting the ODE solver to switch between the sets transport model ODEs as directed by the `UFLUID` subroutine. The Save/Restore feature within DVODE can be used to solve the transport ODEs for each inclusion with a single solver.

The current method of performing nastic simulations uncoupled from ABAQUS would no longer be valid since the pressure-volume behavior of the inclusion element is influenced by its neighboring elements. Therefore, the compatibility issues between ABAQUS and the DVODE package must be resolved. This can be accomplished by using the Save/Restore feature within DVODE to store the DVODE variables after a successful time step within ABAQUS and restore them if the next increment is rejected. This is equivalent to solving two sets of transport ODEs with the integration of one set lagging one increment behind the other. If the time increment attempted using the first problem fails, the second problem can proceed with integration from the beginning of the time increment. This would eliminate the need for DVODE to "back up" integration, allowing it to be used within an ABAQUS simulation.

7.4 EFFECTS OF HYDROSTATIC PRESSURE ON MEMBRANE TRANSPORT

Even in the absence of a pressure gradient across the membrane, high hydrostatic pressures affect membrane transport processes. These effects are outlined by Macdonald [91] and include changes in both the lipid bilayer and the kinetics of membrane transport proteins. The phase transition temperature, T_{m}, of the lipid bilayer is increased with pressure according to the Clausius-Clapeyron relation [91],

$$\frac{dT_{\mathrm{m}}}{dp}=\frac{T_{\mathrm{m}}\Delta V}{\Delta H} \tag{7.4.1}$$

where T_{m} is the phase transition temperature, and ΔV and ΔH are the changes in volume and enthalpy respectively of the lipids during a phase change. If the pressure is raised high enough, this can result in an isothermal transition to the gel state, which can reduce the permeability of the membrane and decrease the reaction rate of transport proteins. High pressure also causes a reduction in the atomic spacing between the lipid monolayers. This has the effect of altering the passive permeability of the membrane to both ions and solvent. Usually, the permeability of the membrane decreases linearly as the pressure is increased unless a phase change in the membrane occurs, in which case the permeability will typically increase [91]. Voltage gated ion channels are relatively unaffected by an increase in hydrostatic pressure, except for a slowing of the gating kinetics [91, 92]. Ion pumps, exchangers, and cotransporters are also affected by high hydrostatic pressures. The extent of the change varies depending on the particular transporter; however, in most instances, both the maximum reaction velocity and saturation concentration are reduced as hydrostatic pressure is increased. The dependence of the various kinetics parameters on hydrostatic pressure will need to be determined experimentally and the tabulated data will need to be used instead of assuming that the parameters are constants.

7.5 EFFECTS OF TEMPERATURE ON MEMBRANE TRANSPORT

A change in temperature can have a significant impact on the transport processes. As discussed in the previous section, a change in temperature can cause a change in the phase of the lipid bilayer, which affects both the ion and water permeabilities of the membrane. Also, the temperature affects the kinetics of pumps and exchangers. The energetics aspect of these effects is taken into account by the Nernst equilibrium potential; however, the effects on the actual enzyme kinetics are not. Channel gating is also influenced by temperature. In general, an increase in temperature will cause an increase in permeability and reaction rates. The temperature can also affect the stiffness of the polymer matrix. In order to take into account the effects of temperature, the permeability, rate, and material constants in the model must be tabulated according to temperature.

APPENDIX A

DERIVATION OF TRANSPORT MODEL EQUATIONS

A.1 DERIVATION OF ION PUMP CURRENT EQUATION

For a generic ion pump with the reaction,

$$\mathrm{ATP} + m\mathrm{S}_{\mathrm{i}}^{z_{\mathrm{S}}} + n\mathrm{X}_{\mathrm{e}}^{z_{\mathrm{X}}} \underset{\beta}{\overset{\alpha}{\rightleftharpoons}} \mathrm{ADP} + \mathrm{P}_{\mathrm{io}} + m\mathrm{S}_{\mathrm{e}}^{z_{\mathrm{S}}} + n\mathrm{X}_{\mathrm{i}}^{z_{\mathrm{X}}} \tag{A.1.1}$$

that moves *m*-S ions out of the cell and *n*-X ions into the cell in a single cycle, the energy required to move *m*-S ions from the intracellular space to the extracellular space against their electrochemical gradient is given by,

$$\Delta G_{\mathrm{S}} = -mz_{\mathrm{S}}e\left(v - v_{\mathrm{S}}\right) \tag{A.1.2}$$

The energy required to move *n*-X ions from the extracellular space to the intracellular space against their electrochemical gradient is given by,

$$\Delta G_{\mathrm{X}} = nz_{\mathrm{X}}e\left(v - v_{\mathrm{X}}\right) \tag{A.1.3}$$

The energy balance is thus,

$$\Delta G = \Delta G_{\mathrm{X}} + \Delta G_{\mathrm{S}} + \Delta G_{\mathrm{ATP}} = e\left[mz_{\mathrm{S}}v_{\mathrm{S}} - nz_{\mathrm{X}}v_{\mathrm{X}} + \left(nz_{\mathrm{X}} - mz_{\mathrm{S}}\right)v\right] + \Delta G_{\mathrm{ATP}} \tag{A.1.4}$$

where ΔG_{ATP} is the free energy released by the hydrolysis of ATP. According to Endresen *et al.* [67], ion pumps quickly reach saturation and therefore the sum of the forward and backward rates of the reaction is chosen to be a constant to represent the maximum possible reaction rates in the forward and backward directions. Thus,

$$\alpha + \beta = \lambda = \text{const.} \tag{A.1.5}$$

Endresen also indicates that since the forward and backward reaction rates must be equal at equilibrium, they are related by,

$$\frac{\alpha}{\beta} = \exp\left(-\frac{\Delta G}{kT}\right) \tag{A.1.6}$$

Solving Eqs. (A.1.5) and (A.1.6) simultaneously for α and β yields the following expression for the net reaction rate $\alpha - \beta$ [67],

$$\alpha - \beta = \lambda \tanh\left(-\frac{\Delta G}{2kT}\right) \tag{A.1.7}$$

Thus, the net pump current is given by,

$$\begin{aligned} i_{\text{pump}} &= (mz_{\text{S}} - nz_{\text{X}})Me(\alpha - \beta) \\ &= (mz_{\text{S}} - nz_{\text{X}})Me\lambda \tanh\left(-\frac{\Delta G}{2kT}\right) \\ &= (mz_{\text{S}} - nz_{\text{X}})Me\lambda \tanh\left(\frac{e[(mz_{\text{S}} - nz_{\text{X}})v + nz_{\text{X}}v_{\text{X}} - mz_{\text{S}}v_{\text{S}}] - \Delta G_{\text{ATP}}}{2kT}\right) \\ &= k_{\text{pump}} \tanh\left(\frac{e[(mz_{\text{S}} - nz_{\text{X}})v + nz_{\text{X}}v_{\text{X}} - mz_{\text{S}}v_{\text{S}}] - \Delta G_{\text{ATP}}}{2kT}\right) \end{aligned} \tag{A.1.8}$$

where,

$$k_{\text{pump}} = (mz_{\text{S}} - nz_{\text{X}})Me\lambda \tag{A.1.9}$$

and *M* is the number of pumps.

A.2 DERIVATION OF ION EXCHANGER CURRENT EQUATION

For a generic ion exchanger with the reaction,

$$a\text{S}_{\text{e}}^{z_{\text{S}}} + b\text{X}_{\text{i}}^{z_{\text{X}}} \underset{\beta}{\overset{\alpha}{\rightleftharpoons}} a\text{S}_{\text{i}}^{z_{\text{S}}} + b\text{X}_{\text{e}}^{z_{\text{X}}} \tag{A.2.1}$$

that moves *a*-S ions into the cell and *b*-X ions out of the cell in a single cycle, the energy required to move *a*-S ions from the extracellular space to the intracellular space against their electrochemical gradient is given by,

$$\Delta G_{\text{S}} = az_{\text{S}}e(v - v_{\text{S}}) \tag{A.2.2}$$

The energy required to move *n*-X ions from the intracellular space to the extracellular space against their electrochemical gradient is given by,

$$\Delta G_{\mathrm{X}} = -bz_{\mathrm{X}}e(v - v_{\mathrm{X}}) \tag{A.2.3}$$

The energy balance is thus,

$$\begin{aligned}
\Delta G &= \Delta G_{\mathrm{S}} + \Delta G_{\mathrm{X}} \\
&= az_{\mathrm{S}}e(v - v_{\mathrm{S}}) - bz_{\mathrm{X}}e(v - v_{\mathrm{X}}) \\
&= az_{\mathrm{S}}e\left(v - \frac{kT}{z_{\mathrm{S}}e}\ln\frac{[\mathrm{S}]_{\mathrm{e}}}{[\mathrm{S}]_{\mathrm{i}}}\right) - bz_{\mathrm{X}}e\left(v - \frac{kT}{z_{\mathrm{X}}e}\ln\frac{[\mathrm{X}]_{\mathrm{e}}}{[\mathrm{X}]_{\mathrm{i}}}\right) \\
&= (az_{\mathrm{S}} - bz_{\mathrm{X}})ev - akT\ln\frac{[\mathrm{S}]_{\mathrm{e}}}{[\mathrm{S}]_{\mathrm{i}}} + bkT\ln\frac{[\mathrm{X}]_{\mathrm{e}}}{[\mathrm{X}]_{\mathrm{i}}} = 0 \qquad \text{(A.2.4)}\\
&\Rightarrow a\ln\frac{[\mathrm{S}]_{\mathrm{e}}}{[\mathrm{S}]_{\mathrm{i}}} - b\ln\frac{[\mathrm{X}]_{\mathrm{e}}}{[\mathrm{X}]_{\mathrm{i}}} = (az_{\mathrm{S}} - bz_{\mathrm{X}})\frac{ev}{kT} \\
&\Rightarrow \left(\frac{[\mathrm{S}]_{\mathrm{e}}}{[\mathrm{S}]_{\mathrm{i}}}\right)^{a}\left(\frac{[\mathrm{X}]_{\mathrm{i}}}{[\mathrm{X}]_{\mathrm{e}}}\right)^{b} = \exp\left[(az_{\mathrm{S}} - bz_{\mathrm{X}})\frac{ev}{kT}\right] \\
&\Rightarrow [\mathrm{S}]_{\mathrm{e}}^{a}[\mathrm{X}]_{\mathrm{i}}^{b}\exp\left[-(az_{\mathrm{S}} - bz_{\mathrm{X}})\frac{ev}{2kT}\right] = [\mathrm{S}]_{\mathrm{i}}^{a}[\mathrm{X}]_{\mathrm{e}}^{b}\exp\left[(az_{\mathrm{S}} - bz_{\mathrm{X}})\frac{ev}{2kT}\right]
\end{aligned}$$

The left-hand side of the above expression can be related to the S influx (hence the forward reaction rate) and the right-hand side can be related to the S efflux (hence the rate of the reverse reaction) according to Mullins [81]. Therefore, the rates of the forward and backward reactions are,

$$\alpha = \lambda[\mathrm{S}]_{\mathrm{e}}^{a}[\mathrm{X}]_{\mathrm{i}}^{b}\exp\left[-(az_{\mathrm{S}} - bz_{\mathrm{X}})\frac{ev}{2kT}\right] \tag{A.2.5}$$

$$\beta = \lambda[\mathrm{S}]_{\mathrm{i}}^{a}[\mathrm{X}]_{\mathrm{e}}^{b}\exp\left[(az_{\mathrm{S}} - bz_{\mathrm{X}})\frac{ev}{2kT}\right] \tag{A.2.6}$$

where λ is a rate constant determined from experimental data. Therefore, the net current is,

$$\begin{aligned}
i_{\mathrm{exchanger}} &= (bz_{\mathrm{X}} - az_{\mathrm{S}})Ne(\alpha - \beta) \qquad \text{(A.2.7)}\\
&= (bz_{\mathrm{X}} - az_{\mathrm{S}})Ne\lambda \\
&\quad\times\left\{\begin{aligned}&[\mathrm{S}]_{\mathrm{e}}^{a}[\mathrm{X}]_{\mathrm{i}}^{b}\exp\left[-(az_{\mathrm{S}} - bz_{\mathrm{X}})\frac{ev}{2kT}\right] \\ &-[\mathrm{S}]_{\mathrm{i}}^{a}[\mathrm{X}]_{\mathrm{e}}^{b}\exp\left[(az_{\mathrm{S}} - bz_{\mathrm{X}})\frac{ev}{2kT}\right]\end{aligned}\right\}
\end{aligned}$$

$$= (bz_X - az_S)Ne\lambda\sqrt{[S]_e^a[S]_i^a[X]_e^b[X]_i^b}$$

$$\times\left\{\begin{array}{l}\sqrt{\dfrac{[S]_e^a}{[S]_i^a}}\sqrt{\dfrac{[X]_i^b}{[X]_e^b}}\exp\left[-(az_S - bz_X)\dfrac{ev}{2kT}\right] \\ -\sqrt{\dfrac{[S]_i^a}{[S]_e^a}}\sqrt{\dfrac{[X]_e^b}{[X]_i^b}}\exp\left[(az_S - bz_X)\dfrac{ev}{2kT}\right]\end{array}\right\}$$

$$= (bz_X - az_S)Ne\lambda\sqrt{[S]_e^a[S]_i^a[X]_e^b[X]_i^b}$$

$$\times\left\{\begin{array}{l}\exp\left[\dfrac{v_S az_S e}{2kT}\right]\exp\left[-\dfrac{v_X bz_X e}{2kT}\right]\exp\left[-(az_S - bz_X)\dfrac{ev}{2kT}\right] \\ -\exp\left[-\dfrac{v_S az_S e}{2kT}\right]\exp\left[\dfrac{v_X bz_X e}{2kT}\right]\exp\left[(az_S - bz_X)\dfrac{ev}{2kT}\right]\end{array}\right\}$$

$$= 2(bz_X - az_S)Ne\lambda\sqrt{[S]_e^a[S]_i^a[X]_e^b[X]_i^b}$$

$$\times\frac{1}{2}\left\{\begin{array}{l}\exp\left[-\dfrac{e[v(az_S - bz_X) + v_X bz_X - v_S az_S]}{2kT}\right] \\ -\exp\left[\dfrac{e[v(az_S - bz_X) + v_X bz_X - v_S az_S]}{2kT}\right]\end{array}\right\}$$

$$= 2(bz_X - az_S)Ne\lambda\sqrt{[S]_e^a[S]_i^a[X]_e^b[X]_i^b}$$

$$\times\sinh\left(-\frac{e[(az_S - bz_X)v - az_S v_S + bz_X v_X]}{2kT}\right)$$

$$= k_{SX}\sinh\left(\frac{e[(bz_X - az_S)v + az_S v_S - bz_X v_X]}{2kT}\right)$$

where,

$$k_{SX} = 2(bz_X - az_S)Ne\lambda\sqrt{[S]_e^a[S]_i^a[X]_e^b[X]_i^b} \tag{A.2.8}$$

and N is the number of exchangers.

A.3 DERIVATION OF H$^+$/SUCROSE COTRANSPORTER CURRENT EQUATION

The H$^+$/Sucrose cotransporter has the reaction,

$$H_e^+ + Suc_e + n_W H_2O_e \underset{\beta}{\overset{\alpha}{\rightleftharpoons}} H_i^+ + Suc_i + n_W H_2O_i \tag{A.3.1}$$

This device can be modeled in a similar manner as an exchanger except that it must be taken into account that sucrose has no valence charge. The energy released by the movement of a single H^+ from the extracellular space to the intracellular space down its concentration and charge gradient is given by,

$$\Delta G_{\mathrm{H}^+} = e(v - v_{\mathrm{H}}) \tag{A.3.2}$$

where,

$$v_{\mathrm{H}} = \frac{kT}{e}\ln\frac{[\mathrm{H}^+]_{\mathrm{e}}}{[\mathrm{H}^+]_{\mathrm{i}}} = \frac{kT}{e}\ln(10)(\mathrm{pH}_{\mathrm{i}} - \mathrm{pH}_{\mathrm{e}}) \tag{A.3.3}$$

The energy required to move a single Sucrose from the extracellular space to the intracellular space against its concentration gradient is given by,

$$\Delta G_{\mathrm{Suc}} = kT\ln\frac{[\mathrm{Suc}]_{\mathrm{i}}}{[\mathrm{Suc}]_{\mathrm{e}}} \tag{A.3.4}$$

And finally, energy required to move n_{W} water molecules from the extracellular space to the intracellular space against their concentration gradient is given by,

$$\Delta G_{\mathrm{H_2O}} = n_{\mathrm{W}} kT\ln\frac{[\mathrm{H_2O}]_{\mathrm{i}}}{[\mathrm{H_2O}]_{\mathrm{e}}} \tag{A.3.5}$$

Thus, the energy balance for equilibrium must be,

$$\Delta G_{\mathrm{H}^+} + \Delta G_{\mathrm{Suc}} + \Delta G_{\mathrm{H_2O}} = e(v - v_{\mathrm{H}}) + kT\ln\frac{[\mathrm{Suc}]_{\mathrm{i}}}{[\mathrm{Suc}]_{\mathrm{e}}} + n_{\mathrm{W}} kT\ln\frac{[\mathrm{H_2O}]_{\mathrm{i}}}{[\mathrm{H_2O}]_{\mathrm{e}}} = 0 \tag{A.3.6}$$

Substituting in the definition for v_{H} and rearranging the above expression yields,

$$\frac{[\mathrm{H}^+]_{\mathrm{e}}}{[\mathrm{H}^+]_{\mathrm{i}}} = \frac{[\mathrm{Suc}]_{\mathrm{i}}}{[\mathrm{Suc}]_{\mathrm{e}}}\left(\frac{[\mathrm{H_2O}]_{\mathrm{i}}}{[\mathrm{H_2O}]_{\mathrm{e}}}\right)^{n_{\mathrm{W}}} \exp\left(\frac{ev}{kT}\right) \tag{A.3.7}$$

A symmetrical rearrangement of this expression gives,

$$[\mathrm{H}^+]_{\mathrm{e}}[\mathrm{Suc}]_{\mathrm{e}}[\mathrm{H_2O}]_{\mathrm{e}}^{n_{\mathrm{W}}} \exp\left(-\frac{ev}{2kT}\right) = [\mathrm{H}^+]_{\mathrm{i}}[\mathrm{Suc}]_{\mathrm{i}}[\mathrm{H_2O}]_{\mathrm{i}}^{n_{\mathrm{W}}} \exp\left(\frac{ev}{2kT}\right) \tag{A.3.8}$$

Note that the above rearrangement yields extracellular concentrations on the left-hand side and intracellular concentrations on the right-hand side which matches the subscripts in the reaction equation whereas the symmetrical rearrangement for an exchanger yields a cross product of intracellular and extracellular concentrations which also matches the subscripts on the right- and left-hand sides of the reaction. The left-hand side of the above expression can be related to the

H^+ influx (hence the forward reaction rate) and the right-hand side can be related to the H^+ efflux (hence the rate of the reverse reaction) according to Mullins [81]. Therefore, the rates for the forward and backward reactions are,

$$\alpha = \lambda [H^+]_e [\mathrm{Suc}]_e [H_2O]_e^{n_W} \exp\left(-\frac{ev}{2kT}\right) \tag{A.3.9}$$

$$\beta = \lambda [H^+]_i [\mathrm{Suc}]_i [H_2O]_i^{n_W} \exp\left(\frac{ev}{2kT}\right) \tag{A.3.10}$$

where λ is a rate constant determined from experimental data. Thus for a membrane with N cotransporters, the net cotransporter current is given by,

$$\begin{aligned}
i_{\mathrm{cotransporter}} &= -Ne(\alpha - \beta) \qquad \text{(A.3.11)}\\
&= Ne\lambda \begin{bmatrix} [H^+]_i [\mathrm{Suc}]_i [H_2O]_i^{n_W} \exp\left(\frac{ev}{2kT}\right) \\ -[H^+]_e [\mathrm{Suc}]_e [H_2O]_e^{n_W} \exp\left(-\frac{ev}{2kT}\right) \end{bmatrix}\\
&= Ne\lambda \sqrt{[H^+]_e [H^+]_i [\mathrm{Suc}]_e [\mathrm{Suc}]_i [H_2O]_e^{n_W} [H_2O]_i^{n_W}}\\
&\times \begin{bmatrix} \sqrt{\frac{[H^+]_i}{[H^+]_e}} \sqrt{\frac{[\mathrm{Suc}]_i}{[\mathrm{Suc}]_e}} \sqrt{\frac{[H_2O]_i^{n_W}}{[H_2O]_e^{n_W}}} \exp\left(\frac{ev}{2kT}\right) \\ -\sqrt{\frac{[H^+]_e}{[H^+]_i}} \sqrt{\frac{[\mathrm{Suc}]_e}{[\mathrm{Suc}]_i}} \sqrt{\frac{[H_2O]_e^{n_W}}{[H_2O]_i^{n_W}}} \exp\left(-\frac{ev}{2kT}\right) \end{bmatrix}\\
&= Ne\lambda \sqrt{[H^+]_e [H^+]_i [\mathrm{Suc}]_e [\mathrm{Suc}]_i [H_2O]_e^{n_W} [H_2O]_i^{n_W}}\\
&\times \begin{bmatrix} \exp\left(-\frac{v_H e}{2kT}\right) \exp\left(-\frac{v_{\mathrm{Suc}} e}{2kT}\right) \exp\left(-\frac{n_W v_{H_2O} e}{2kT}\right) \exp\left(\frac{ev}{2kT}\right) \\ -\exp\left(\frac{v_H e}{2kT}\right) \exp\left(\frac{v_{\mathrm{Suc}} e}{2kT}\right) \exp\left(\frac{n_W v_{H_2O} e}{2kT}\right) \exp\left(-\frac{ev}{2kT}\right) \end{bmatrix}\\
&= 2Ne\lambda \sqrt{[H^+]_e [H^+]_i [\mathrm{Suc}]_e [\mathrm{Suc}]_i [H_2O]_e^{n_W} [H_2O]_i^{n_W}}\\
&\times \left\{ \frac{1}{2} \begin{bmatrix} \exp\left(\frac{e(v - v_H - v_{\mathrm{Suc}} - n_W v_{H_2O})}{2kT}\right) \\ -\exp\left(-\frac{e(v - v_H - v_{\mathrm{Suc}} - n_W v_{H_2O})}{2kT}\right) \end{bmatrix} \right\}
\end{aligned}$$

$$= k_{\text{cotransporter}} \sinh\left(\frac{e\left(v - v_{\text{H}} - v_{\text{Suc}} - n_{\text{W}} v_{\text{H}_2\text{O}}\right)}{2kT} \right)$$

where,

$$k_{\text{cotransporter}} = 2Ne\lambda\sqrt{\left[\text{H}^+\right]_{\text{e}}\left[\text{H}^+\right]_{\text{i}}\left[\text{Suc}\right]_{\text{e}}\left[\text{Suc}\right]_{\text{i}}\left[\text{H}_2\text{O}\right]_{\text{e}}^{n_{\text{W}}}\left[\text{H}_2\text{O}\right]_{\text{i}}^{n_{\text{W}}}} \qquad \text{(A.3.12)}$$

and,

$$v_{\text{Suc}} = \frac{kT}{e}\ln\frac{\left[\text{Suc}\right]_{\text{e}}}{\left[\text{Suc}\right]_{\text{i}}} \qquad \text{(A.3.13)}$$

$$v_{\text{H}_2\text{O}} = \frac{kT}{e}\ln\frac{\left[\text{H}_2\text{O}\right]_{\text{e}}}{\left[\text{H}_2\text{O}\right]_{\text{i}}} = \frac{\overline{V}_{\text{H}_2\text{O}}}{F}\sum_{\text{S}}\pi_{\text{S}} \qquad \text{(A.3.14)}$$

where $\overline{V}_{\text{H}_2\text{O}}$ is the partial molal volume of water and $\Sigma\pi_{\text{S}}$ is the total osmotic pressure difference across the membrane.

A.4 DERIVATION OF H$^+$/SUCROSE COTRANSPORTER CURRENT EQUATION WITH SATURATION

The H$^+$/Sucrose cotransporter current equation derived in Appendix Section A.3 ignores saturation effects. A current equation can be derived that does include the effects of saturation by using a more generalized version of the Mullins [80, 81] formulation. The derivation uses the enzyme kinetics approach with the reaction scheme shown in Figure A.1 in which X is the H$^+$ binding site and Y is the induced sucrose binding site.

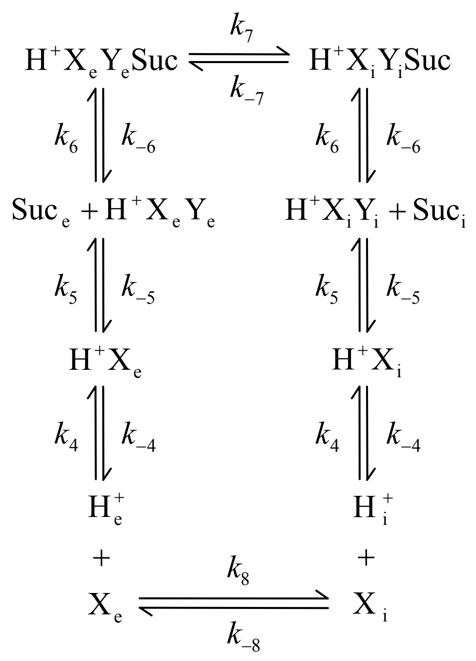


Figure A.1: Reaction Scheme for H$^+$/Sucrose Cotransporter – The simplified enzyme kinetics reaction scheme for a cotransporter with saturation effects. The reaction scheme shows the binding, translocation, and release steps of the cotransporter.

It is assumed that the rate constants are the same for both loading and unloading of the carrier and that the system is in equilibrium. Under these assumptions, the above reaction scheme can be described by the following sets of equations:

For H$^+$ binding,

$$k_4\left[\mathrm{H}^+\right]_e\left[\mathrm{X}\right]_e = k_{-4}\left[\mathrm{H}^+\mathrm{X}\right]_e \tag{A.4.1}$$

$$k_4\left[\mathrm{H}^+\right]_i\left[\mathrm{X}\right]_i = k_{-4}\left[\mathrm{H}^+\mathrm{X}\right]_i \tag{A.4.2}$$

For induction and sucrose binding,

$$k_5\left[\mathrm{H}^+\mathrm{X}\right]_e = k_{-5}\left[\mathrm{H}^+\mathrm{XY}\right]_e \tag{A.4.3}$$

$$k_5\left[\mathrm{H}^+\mathrm{X}\right]_i = k_{-5}\left[\mathrm{H}^+\mathrm{XY}\right]_i \tag{A.4.4}$$

$$k_6\left[\mathrm{H}^+\mathrm{XY}\right]_e\left[\mathrm{Suc}\right]_e = k_{-6}\left[\mathrm{H}^+\mathrm{XYSuc}\right]_e \tag{A.4.5}$$

$$k_6\left[\mathrm{H}^+\mathrm{XY}\right]_i\left[\mathrm{Suc}\right]_i = k_{-6}\left[\mathrm{H}^+\mathrm{XYSuc}\right]_i \tag{A.4.6}$$

And for translocation,

$$k_7\left[\mathrm{H}^+\mathrm{XYSuc}\right]_e = k_{-7}\left[\mathrm{H}^+\mathrm{XYSuc}\right]_i \tag{A.4.7}$$

$$k_8\left[\mathrm{X}\right]_e = k_{-8}\left[\mathrm{X}\right]_i \tag{A.4.8}$$

The energy balance for equilibrium must be,

$$\Delta G = e\left(v - \frac{kT}{e}\ln\frac{[\mathrm{H^+}]_\mathrm{e}}{[\mathrm{H^+}]_\mathrm{i}}\right) - kT\ln\frac{[\mathrm{Suc}]_\mathrm{e}}{[\mathrm{Suc}]_\mathrm{i}} = 0$$
$$\Rightarrow [\mathrm{H^+}]_\mathrm{e}[\mathrm{Suc}]_\mathrm{e} = [\mathrm{H^+}]_\mathrm{i}[\mathrm{Suc}]_\mathrm{i}\exp\left(\frac{ev}{kT}\right) \tag{A.4.9}$$

From the reaction scheme shown in Figure A.1, the influx and efflux of H^+ is given by,

$$m_\mathrm{i} = k_7[\mathrm{H^+XYSuc}]_\mathrm{e} \tag{A.4.10}$$

$$m_\mathrm{e} = k_{-7}[\mathrm{H^+XYSuc}]_\mathrm{i} \tag{A.4.11}$$

Combining the H^+ binding and the induction and sucrose binding equations yields,

$$[\mathrm{H^+XYSuc}]_\mathrm{e} = \frac{k_6}{k_{-6}}\frac{k_5}{k_{-5}}\frac{k_4}{k_{-4}}[\mathrm{H^+}]_\mathrm{e}[\mathrm{Suc}]_\mathrm{e}[\mathrm{X}]_\mathrm{e} \tag{A.4.12}$$

$$[\mathrm{H^+XYSuc}]_\mathrm{i} = \frac{k_6}{k_{-6}}\frac{k_5}{k_{-5}}\frac{k_4}{k_{-4}}[\mathrm{H^+}]_\mathrm{i}[\mathrm{Suc}]_\mathrm{i}[\mathrm{X}]_\mathrm{i} \tag{A.4.13}$$

Substituting the above relations and Eq. (A.4.9) into Eq. (A.4.7) yields,

$$\frac{k_7}{k_{-7}} = \exp\left(-\frac{ev}{kT}\right) \tag{A.4.14}$$

To determine the intracellular and extracellular carrier concentrations, $[\mathrm{X}]_\mathrm{i}$ and $[\mathrm{X}]_\mathrm{e}$ respectively, the total carrier concentration, $([\mathrm{X}])_\mathrm{T}$, is employed,

$$([\mathrm{X}])_\mathrm{T} = ([\mathrm{X}]_\mathrm{e})_\mathrm{T} + ([\mathrm{X}]_\mathrm{i})_\mathrm{T} \tag{A.4.15}$$

From the assumption of equilibrium, it can be demonstrated [80] that,

$$[\mathrm{X}]_\mathrm{i} = [\mathrm{X}]_\mathrm{e} \tag{A.4.16}$$

and that,

$$([\mathrm{X}]_\mathrm{e})_\mathrm{T} = [\mathrm{X}]_\mathrm{e} + [\mathrm{H^+X}]_\mathrm{e} + [\mathrm{H^+XY}]_\mathrm{e} + [\mathrm{H^+XYSuc}]_\mathrm{e} \tag{A.4.17}$$

Combining Eqs. (A.4.15), (A.4.16) and (A.4.17) and substituting Eqs. (A.4.12), and (A.4.13), and the equations for H^+ binding and induction and sucrose binding yields,

$$[\mathrm{X}]_\mathrm{e} = [\mathrm{X}]_\mathrm{i} \tag{A.4.18}$$
$$= \frac{([\mathrm{X}])_\mathrm{T}}{2 + \frac{k_4}{k_{-4}}\left([\mathrm{H^+}]_\mathrm{e} + [\mathrm{H^+}]_\mathrm{i}\right) + \frac{k_5}{k_{-5}}\frac{k_4}{k_{-4}}\left[[\mathrm{H^+}]_\mathrm{e} - [\mathrm{H^+}]_\mathrm{i} + \frac{k_6}{k_{-6}}\left([\mathrm{H^+}]_\mathrm{e}[\mathrm{Suc}]_\mathrm{e} - [\mathrm{H^+}]_\mathrm{i}[\mathrm{Suc}]_\mathrm{i}\right)\right]}$$

Subtracting Eqs. (A.4.10) and (A.4.11) and substituting in Eqs. (A.4.12), (A.4.13), and (A.4.18) yields the net H$^+$ flux,

$$m_i - m_e$$
$$= \frac{\frac{k_6}{k_{-6}}\frac{k_5}{k_{-5}}\frac{k_4}{k_{-4}}\left(k_7[\mathrm{H}^+]_e[\mathrm{Suc}]_e - k_{-7}[\mathrm{H}^+]_i[\mathrm{Suc}]_i\right)([\mathrm{X}])_T}{2+\frac{k_4}{k_{-4}}\left([\mathrm{H}^+]_e+[\mathrm{H}^+]_i\right)+\frac{k_5}{k_{-5}}\frac{k_4}{k_{-4}}\left[[\mathrm{H}^+]_e-[\mathrm{H}^+]_i+\frac{k_6}{k_{-6}}\left([\mathrm{H}^+]_e[\mathrm{Suc}]_e-[\mathrm{H}^+]_i[\mathrm{Suc}]_i\right)\right]} \tag{A.4.19}$$

Let,

$$\lambda = \frac{k_6}{k_{-6}}, \quad \gamma = \frac{k_5}{k_{-5}}, \quad K = \frac{k_4}{k_{-4}} \tag{A.4.20}$$

and noting that a symmetrical rearrangement of Eq. (A.4.14),

$$k_7 = \exp\left(\frac{ev}{2kT}\right) \tag{A.4.21}$$

$$k_{-7} = \exp\left(-\frac{ev}{2kT}\right) \tag{A.4.22}$$

where λ is an overall rate constant and K and γ determine the saturation of the cotransporter for increasing H$^+$ and sucrose concentrations respectively. Substituting the above relations for the rate constants into Eq. (A.4.19) and converting to a current yields,

$$i_{\mathrm{cotransporter}} = \frac{NF\lambda\gamma K\left(\exp\left(\frac{ev}{2kT}\right)[\mathrm{H}^+]_e[\mathrm{Suc}]_e - \exp\left(-\frac{ev}{2kT}\right)[\mathrm{H}^+]_i[\mathrm{Suc}]_i\right)}{2+K\left([\mathrm{H}^+]_e+[\mathrm{H}^+]_i\right)+\gamma K\left[[\mathrm{H}^+]_e-[\mathrm{H}^+]_i+\lambda\left([\mathrm{H}^+]_e[\mathrm{Suc}]_e-[\mathrm{H}^+]_i[\mathrm{Suc}]_i\right)\right]} \tag{A.4.23}$$

APPENDIX B

TRANSPORT VALIDATION SOURCE CODE

The transport validation program is written in Microsoft Visual Basic 6.0 and simulates the rabbit sinoatrial node using the transport model presented in Chapter 3.0 and the DOPRI5 algorithm (Section 2.4.1). It features a Windows user interface where the user can select the input file containing the model parameters and name an output file. The user can also change the simulation time, initial time step, relative error tolerance, maximum step size, maximum number of steps and the frequency with which output is written. When the user clicks the **Run** button, the program performs the simulation and writes the output data to the designated text file.

```
VERSION 5.00
Object = "{F9043C88-F6F2-101A-A3C9-08002B2F49FB}#1.2#0"; "COMDLG32.OCX"
Object = "{831FDD16-0C5C-11D2-A9FC-0000F8754DA1}#2.0#0"; "MSCOMCTL.OCX"
Begin VB.Form frmMain
   Caption         =   "Nastic Model"
   ClientHeight    =   3645
   ClientLeft      =   60
   ClientTop       =   510
   ClientWidth     =   9270
   LinkTopic       =   "Form1"
   ScaleHeight     =   3645
   ScaleWidth      =   9270
   StartUpPosition =   3  'Windows Default
   Begin MSComctlLib.ProgressBar pgbProgress
      Height          =   375
      Left            =   4320
      TabIndex        =   18
      Top             =   2880
      Width           =   4695
      _ExtentX        =   8281
      _ExtentY        =   661
      _Version        =   393216
      Appearance      =   1
      Scrolling       =   1
   End
   Begin VB.TextBox txtMaxTimeSteps
      Height          =   325
      Left            =   4320
      TabIndex        =   17
      Text            =   "txtMaxTimeSteps"
      Top             =   2280
      Width           =   1215
   End
   Begin VB.TextBox txtMaxTimeStep
      Height          =   325
      Left            =   1440
      TabIndex        =   15
      Text            =   "txtMaxTimeStep"
      Top             =   2280
      Width           =   1335
   End
   Begin VB.TextBox txtMaxRelError
      Height          =   325
      Left            =   6480
      TabIndex        =   13
      Text            =   "txtMaxRelError"
      Top             =   1800
      Width           =   1455
   End
   Begin VB.TextBox txtInitTimeStep
      Height          =   325
      Left            =   3720
      TabIndex        =   11
      Text            =   "txtInitTimeStep"
      Top             =   1800
      Width           =   1215
   End
   Begin VB.TextBox txtMaxTime
      Height          =   325
      Left            =   1080
      TabIndex        =   9
      Text            =   "txtMaxTime"
      Top             =   1800
      Width           =   1215
   End
   Begin VB.CommandButton cmdExit
      Caption         =   "Exit"
      Height          =   495
      Left            =   1800
      TabIndex        =   7
      Top             =   2880
      Width           =   1335
   End
   Begin VB.CommandButton cmdRun
      Caption         =   "Run"
      Height          =   495
      Left            =   240
      TabIndex        =   6
      Top             =   2880
      Width           =   1455
```

```
End
Begin VB.TextBox txtInputFile
   Height          =   325
   Left            =   960
   TabIndex        =   3
   Text            =   "txtInputFile"
   Top             =   720
   Width           =   6135
End
Begin VB.CommandButton cmdInputBrowse
   Caption         =   "Browse..."
   Height          =   375
   Left            =   7200
   TabIndex        =   2
   Top             =   720
   Width           =   1335
End
Begin VB.TextBox txtOutputFile
   Height          =   325
   Left            =   1080
   TabIndex        =   1
   Text            =   "txtOutputFile"
   Top             =   1320
   Width           =   6015
End
Begin VB.CommandButton cmdOutputBrowse
   Caption         =   "Browse..."
   Height          =   375
   Left            =   7200
   TabIndex        =   0
   Top             =   1320
   Width           =   1335
End
Begin MSComDlg.CommonDialog CommonDialog
   Left            =   3480
   Top             =   2880
   _ExtentX        =   847
   _ExtentY        =   847
   _Version        =   393216
   CancelError     =   -1  'True
End
Begin VB.Label lblMaxTimeSteps
   Caption         =   "Max. Time Steps"
   Height          =   255
   Left            =   3000
   TabIndex        =   16
   Top             =   2320
   Width           =   1215
End
Begin VB.Label lblMaxTimeStep
   Caption         =   "Max. Time Step"
   Height          =   255
   Left            =   240
   TabIndex        =   14
   Top             =   2320
   Width           =   1215
End
```

```
   Begin VB.Label lblMaxRelError
      Caption         =   "Max. Rel. Error"
      Height          =   255
      Left            =   5280
      TabIndex        =   12
      Top             =   1840
      Width           =   1215
   End
   Begin VB.Label lblInitTimeStep
      Caption         =   "Init. Time Step"
      Height          =   255
      Left            =   2640
      TabIndex        =   10
      Top             =   1840
      Width           =   1095
   End
   Begin VB.Label lblMaxTime
      Caption         =   "Max. Time"
      Height          =   255
      Left            =   240
      TabIndex        =   8
      Top             =   1840
      Width           =   735
   End
   Begin VB.Label lblInputFile
      Caption         =   "Input File"
      Height          =   255
      Left            =   240
      TabIndex        =   5
      Top             =   765
      Width           =   735
   End
   Begin VB.Label lblOutputFile
      Caption         =   "Output File"
      Height          =   255
      Left            =   240
      TabIndex        =   4
      Top             =   1365
      Width           =   975
   End
End
Attribute VB_Name = "frmMain"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
'----------------------------------
'|      Nastic Model Program      |
'|       BY: Chris Homison        |
'----------------------------------

Option Explicit

' Physical Constants
Const k As Double = 1.38065812E-20          ' Boltzmann's constant
Const e As Double = 1.6021773349E-19        ' Elementary charge
Const F As Double = 96485.30929             ' Faraday's constant
Const R As Double = 8314.51193511388         ' Universal gas constant
Const v_T As Double = 26.7268239657583

' Variable Declarations-----------------------------------------------------
' File System Variables
Dim FSO As New FileSystemObject              ' File system object for _
                                               reading text files
Dim EIFile As File                           ' File that is being read/written
Dim EIStream As TextStream                   ' Text stream that is being _
                                               read/written to/from the file

' Model Inputs (Observed Parameters)
Dim K_e As Double                            ' Extracellular potassium _
                                               concentration
Dim Na_e As Double                           ' Extracellular sodium _
                                               concentration
Dim Ca_e As Double                           ' Extracellular calcium _
                                               concentration
Dim v_x As Double
Dim v_d As Double
Dim v_f As Double
Dim v_m As Double
Dim v_h As Double
Dim v_ATP As Double                          ' ATP potential
Dim V As Double                              ' Cell volume
Dim C As Double                              ' Cell membrane capacitance
Dim tau As Double
Dim T As Double                              ' Temperature

' Model Inputs (Adjusted Parameters)
Dim k_Ca As Double
Dim k_Na As Double
Dim k_K As Double
Dim k_NaCa As Double
Dim k_NaK As Double
Dim k_bCa As Double

Dim v_mem As Double                          ' Membrane potential

Private Function Sinh(x As Double) As Double
    Sinh = (Exp(x) - Exp(-x)) / 2
End Function

Private Function Cosh(x As Double) As Double
    Cosh = (Exp(x) + Exp(-x)) / 2
End Function

Private Function Tanh(x As Double) As Double
    Tanh = (Exp(x) - Exp(-x)) / (Exp(x) + Exp(-x))
End Function

Private Function Fcn(x As Double, y() As Double) As Double()
    Dim Fc(6) As Double
    Dim v_K As Double
    Dim v_Ca As Double
    Dim v_Na As Double
    Dim i_K As Double
    Dim d_inf As Double
    Dim i_Ca As Double
    Dim m_inf As Double
    Dim i_Na As Double
    Dim i_NaK As Double
    Dim i_NaCa As Double

    v_mem = y(5)
    v_K = k * T / e * Log(K_e / y(2))
    v_Ca = k * T / (2 * e) * Log(Ca_e / y(3))
    v_Na = k * T / e * Log(Na_e / y(4))
    i_K = k_K * y(0) * Sinh(e * (v_mem - v_K) / (2 * k * T))
    d_inf = 0.5 * (1 + Tanh(2 * e * (v_mem - v_d) / (k * T)))
    i_Ca = (k_Ca * (1 - y(0)) * d_inf + k_bCa) * Sinh(e * (v_mem - v_Ca) / _
        (k * T))
    m_inf = 0.5 * (1 + Tanh(2 * e * (v_mem - v_m) / (k * T)))
    i_Na = k_Na * y(1) * m_inf * Sinh(e * (v_mem - v_Na) / (2 * k * T))
    i_NaK = k_NaK * Tanh(e * (v_mem + 2 * v_K - 3 * v_Na - v_ATP) / (2 * k _
        * T))
    i_NaCa = k_NaCa * Sinh(e * (v_mem - 3 * v_Na + 2 * v_Ca) / (2 * k * T))
    Fc(0) = 1 / tau * Cosh(2 * e * (v_mem - v_x) / (k * T)) * (0.5 * (1 + _
        Tanh(2 * e * (v_mem - v_x) / (k * T))) - y(0))
    Fc(1) = 1 / tau * Cosh(2 * e * (v_mem - v_h) / (k * T)) * (0.5 * (1 - _
        Tanh(2 * e * (v_mem - v_h) / (k * T))) - y(1))
    Fc(2) = (2 * i_NaK - i_K) / (F * V)
    Fc(3) = (2 * i_NaCa - i_Ca) / (2 * F * V)
    Fc(4) = (-i_Na - 3 * i_NaK - 3 * i_NaCa) / (F * V)
    Fc(5) = -1 / C * (i_K + i_Ca + i_Na + i_NaCa + i_NaK)
    Fcn = Fc
End Function

Private Sub Form_Load()
    pgbProgress.Value = 0
    txtInitTimeStep.Text = "0.01"
    txtInputFile.Text = ""
    txtMaxRelError.Text = "0.00000000001"
    txtMaxTime.Text = "2000"
    txtMaxTimeStep.Text = "10"
    txtMaxTimeSteps.Text = "25000"
    txtOutputFile.Text = ""
End Sub

Private Sub cmdExit_Click()
    End
End Sub

Private Sub cmdInputBrowse_Click()
    On Error Resume Next
    CommonDialog.Filter = "Input File (*.inp)|*.inp"
    CommonDialog.ShowOpen
    If CommonDialog.FileName = "" Or Err.Number = 32755 Then
        Exit Sub
    End If
    txtInputFile.Text = CommonDialog.FileName
End Sub

Private Sub cmdOutputBrowse_Click()
```

```
    Dim MsgResult As VbMsgBoxResult

    On Error Resume Next
    CommonDialog.Filter = "Output File (*.out)|*.out"
    CommonDialog.FileName = ""
RetrySave:
    CommonDialog.ShowSave
    If CommonDialog.FileName = "" Or Err.Number = 32755 Then
        Exit Sub
    End If
    txtOutputFile.Text = CommonDialog.FileName
    If FSO.FileExists(txtOutputFile.Text) = True Then
        MsgResult = MsgBox("File already exists. Overwrite?", vbYesNoCancel _
            + vbExclamation, Me.Caption)
        If MsgResult = vbNo Then
            GoTo RetrySave
            Exit Sub
        End If
        If MsgResult = vbCancel Then
            Exit Sub
        End If
    End If
End Sub

Private Sub cmdRun_Click()
    Const N As Integer = 6
    Dim intI As Integer
    Dim NStep As Integer
    Dim NAccpt As Integer
    Dim NRejct As Integer
    Dim Reject As Boolean
    Dim H As Double
    Dim HMAX As Double
    Dim HNEW As Double
    Dim x As Double
    Dim XEnd As Double
    Dim XPH As Double
    Dim Eps As Double
    Dim Denom As Double
    Dim Err As Double
    Dim Fac As Double
    Dim K1() As Double
    Dim K2() As Double
    Dim K3() As Double
    Dim K4() As Double
    Dim K5() As Double
    Dim y() As Double
    Dim y1(6) As Double

    ' Gather the input data
    If FSO.FileExists(txtInputFile.Text) = False Then
        MsgBox "Input file does not exist.", vbCritical, frmMain.Caption
        Exit Sub
    End If
    Set EIFile = FSO.GetFile(txtInputFile.Text)
    Set EIStream = EIFile.OpenAsTextStream(1)
    K_e = EIStream.ReadLine
    Na_e = EIStream.ReadLine
    Ca_e = EIStream.ReadLine
    v_x = EIStream.ReadLine
    v_d = EIStream.ReadLine
    v_f = EIStream.ReadLine
    v_m = EIStream.ReadLine
    v_h = EIStream.ReadLine
    v_ATP = EIStream.ReadLine
    V = EIStream.ReadLine
    C = EIStream.ReadLine
    tau = EIStream.ReadLine
    T = EIStream.ReadLine
    k_Ca = EIStream.ReadLine
    k_Na = EIStream.ReadLine
    k_K = EIStream.ReadLine
    k_NaCa = EIStream.ReadLine
    k_NaK = EIStream.ReadLine
    k_bCa = EIStream.ReadLine
    ReDim y(5)
    y(0) = EIStream.ReadLine
    y(1) = EIStream.ReadLine
    y(2) = EIStream.ReadLine
    y(3) = EIStream.ReadLine
    y(4) = EIStream.ReadLine
    y(5) = F * V / C * (y(2) - K_e + 2 * (y(3) - Ca_e) + y(4) - Na_e)
    EIStream.Close
    ' Create the Output File
    Set EIStream = FSO.CreateTextFile(txtOutputFile.Text)
    ' DOPRI5 Algorithm
    H = CDbl(txtInitTimeStep.Text)
    HMAX = CDbl(txtMaxTimeStep.Text)
    XEnd = CDbl(txtMaxTime.Text)
    Eps = CDbl(txtMaxRelError.Text)
    Reject = False
    NAccpt = 0
    NRejct = 0
    NStep = 0
    x = 0
    K1 = Fcn(x, y)
    EIStream.WriteLine "Step" & vbTab & "Accepted" & vbTab & "Rejected" & _
        vbTab & "Time" & vbTab & "v" & vbTab & "[K]" & vbTab & "[Ca]" & _
        vbTab & "[Na]" & vbTab & "Error"
    EIStream.WriteLine CStr(NStep) & vbTab & CStr(NAccpt) & vbTab & _
        CStr(NRejct) & vbTab & CStr(x) & vbTab & CStr(v_mem) & vbTab & _
        CStr(y(2)) & vbTab & CStr(y(3)) & vbTab & CStr(y(4)) & vbTab & "0"
    ' Basic Integration Step
NextStep:
    pgbProgress.Value = x / XEnd * 100
    If NStep > CInt(txtMaxTimeSteps.Text) Or x + 0.1 * H = x Then
        GoTo FailExit
    End If
    ' Check to see if the max. time has been reached
    If XEnd - x < 0.000000000001 Then
        Beep
        EIStream.Close
        Exit Sub
    End If
```

```
If x + H > XEnd Then
    H = XEnd - x
End If
NStep = NStep + 1
' The First 6 Stages
For intI = 0 To N - 1
    y1(intI) = y(intI) + H * 0.2 * K1(intI)
Next
K2 = Fcn(x + 0.2 * H, y1)
For intI = 0 To N - 1
    y1(intI) = y(intI) + H * (3 / 40 * K1(intI) + 9 / 40 * K2(intI))
Next
K3 = Fcn(x + 0.3 * H, y1)
For intI = 0 To N - 1
    y1(intI) = y(intI) + H * (44 / 45 * K1(intI) - 56 / 15 * K2(intI) + _
        32 / 9 * K3(intI))
Next
K4 = Fcn(x + 0.8 * H, y1)
For intI = 0 To N - 1
    y1(intI) = y(intI) + H * (19372 / 6561 * K1(intI) - 25360 / 2187 * _
        K2(intI) + 64448 / 6561 * K3(intI) - 212 / 729 * K4(intI))
Next
K5 = Fcn(x + 8 / 9 * H, y1)
For intI = 0 To N - 1
    y1(intI) = y(intI) + H * (9017 / 3168 * K1(intI) - 355 / 33 * _
        K2(intI) + 46732 / 5247 * K3(intI) + 49 / 176 * K4(intI) - 5103 _
        / 18656 * K5(intI))
Next
XPH = x + H
K2 = Fcn(XPH, y1)
For intI = 0 To N - 1
    y1(intI) = y(intI) + H * (35 / 384 * K1(intI) + 500 / 1113 * _
        K3(intI) + 125 / 192 * K4(intI) - 2187 / 6784 * K5(intI) + 11 _
        / 84 * K2(intI))
Next
' Compute the intermediate sum
For intI = 0 To N - 1
    K2(intI) = 71 / 57600 * K1(intI) - 71 / 16695 * K3(intI) + 71 / _
        1920 * K4(intI) - 17253 / 339200 * K5(intI) + 22 / 525 * K2(intI)
Next
' The Last Stage
K3 = Fcn(XPH, y1)
For intI = 0 To N - 1
    K4(intI) = (K2(intI) - 1 / 40 * K3(intI)) * H
Next
' Error Estimation
Err = 0
For intI = 0 To N - 1
    Denom = 0.00001
    If Abs(y1(intI)) > Denom Then
        Denom = Abs(y1(intI))
    End If
    If Abs(y(intI)) > Denom Then
        Denom = Abs(y(intI))
    End If
    Err = Err + (K4(intI) / Denom) ^ 2
Next
```

```
    Err = Sqr(Err / N)
    ' Computation of HNEW
    ' Require that .2 <= HNEW <= 10
    Fac = 5
    If Fac > (Err / Eps) ^ (1 / 5) / 0.9 Then
        Fac = (Err / Eps) ^ (1 / 5) / 0.9
    End If
    If Fac < 0.1 Then
        Fac = 0.1
    End If
    HNEW = H / Fac
    If Err <= Eps Then
        ' Step is accepted
        NAccpt = NAccpt + 1
        K1 = K3
        y = y1
        x = XPH
        EIStream.WriteLine CStr(NStep) & vbTab & CStr(NAccpt) & vbTab & _
            CStr(NRejct) & vbTab & CStr(x) & vbTab & CStr(v_mem) & vbTab & _
            CStr(y(2)) & vbTab & CStr(y(3)) & vbTab & CStr(y(4)) & vbTab & _
            CStr(Err)
        If HNEW > HMAX Then
            HNEW = HMAX
        End If
        If Reject = True And HNEW > H Then
            HNEW = H
        End If
        Reject = False
      Else
        ' Step is rejected
        Reject = True
        If NAccpt >= 1 Then
            NRejct = NRejct + 1
        End If
    End If
    H = HNEW
    GoTo NextStep
FailExit:
    MsgBox "The DOPRI5 method has failed at time " & x & ".  Exiting...", _
        vbCritical, Me.Caption
End Sub
```

APPENDIX C

USER-SUBROUTINE SET FOR IMPLEMENTING THE TRANSPORT MODEL USING THE DOPRI5 ALGORITHM

Below is an ABAQUS user-subroutine set for implementing the transport model using the DOPRI5 algorithm [75] (Section 4.5). It is written using Compaq Visual FORTRAN 6.6 and is FORTRAN 77 compatible. The DOPRI5 algorithm is coded into the ABAQUS user-defined fluid (`UFLUID`) subroutine which is responsible for coupling the transport model to the ABAQUS analysis. Step size control is implemented within the `UFLUID` subroutine by modifying the value of the variable `PNEWDT`, which is the ratio of the suggested new time increment to the time increment being used (`DTIME`). The ODE solver will attempt to use the ABAQUS time increment to perform an integration step and will set the ABAQUS variable `PNEWDT` appropriately based on its own time step algorithm. ABAQUS will repeat the current increment if `PNEWDT` is set to a value less than 1. Input and output data are read/written using the ABAQUS utility routine `UEXTERNALDB`. All coding conventions set out in the ABAQUS user-subroutine set documentation [70] are followed.

```
C === ==============================================================
C     DESCRIPTION: User-defined subroutine set to incorporate the
C                  transport model for coupling the ion/solvent flux to
C                  the hyperelastic model in ABAQUS/Standard
C
C     WRITTEN BY:  Chris Homison, University of Pittsburgh
C
C     REVISIONS:   08/04/2005 Initial release
C                  11/22/2005 Cosmetic changes, added messsage printing
C                  12/06/2005 Added support for proton diffusion, SUT4
C                             water transport, and buffered internal pH
C                  01/14/2006 Revised ATP free energy release
```

```
C                             calculation, added currents/fluxes to the
C                             output
C === ==================================================================
C
C --- ------------------------------------------------------------------
C     User subroutine for reading/writing input/output data
C
C     NOTE: No restart information is written for the user-subroutines
C --- ------------------------------------------------------------------
*DECK UEXTERALDB
      SUBROUTINE UEXTERNALDB(LOP,LRESTART,TIME,DTIME,KSTEP,KINC)
C
      INCLUDE 'ABA_PARAM.INC'
C
      REAL*8 TIME,DTIME
      REAL*8, ALLOCATABLE :: SucData(:,:),SucDataTemp(:,:)
      INTEGER LOP,LRESTART,KSTEP,KINC
      DIMENSION TIME(2)
      REAL*8 pH_i0,Suc_i0,v_0,Vol_0,Rho_0,n0_ATP,n0_ADP,n0_P
      REAL*8 pH_i,Suc_i,v,Vol,Mass_0,n_ATP,n_ADP,n_P,T,P,i_SUT4,i_HPump,
     1      i_HDiff,dVol,dPi
      REAL*8 pH_e,Suc_e,m_Suc_e,slv_act_e,slu_act_e,osm_e,N_SUT4,
     1      lamda_SUT4,n_W,Cap,alpha,A,sigma,M_HPump,lamda_HPump,n_H,
     2      dG0_ATP,Beta_H,D_H,delta
      REAL*8 ERR
      LOGICAL iBUFF,C_ATP,C_ADP,C_P
      INTEGER NDAT
      INTEGER NACCPT,NREJCT,LKINC,REFNODE
      INTEGER LENOUTDIR,N_INC,N_write
      CHARACTER*256 OUTDIR
C --- NOTE: COMMON blocks must begin with a 'K'
C --- Transport Model Input Parameters --------------------------------
      COMMON/KTINP/pH_e,Suc_e,m_Suc_e,slv_act_e,slu_act_e,osm_e,N_SUT4,
     1      lamda_SUT4,n_W,Cap,alpha,A,sigma,M_HPump,lamda_HPump,n_H,
     2      dG0_ATP,Beta_H,D_H,delta,iBUFF,C_ATP,C_ADP,C_P
C     pH_e        External pH
C     Suc_e       External sucrose concentration
C     m_Suc_e     External molal concentration of sucrose
C     N_SUT4      Number of SUT4 cotransporters in the membrane
C     lamda_SUT4  Rate constant for SUT4 cotransporters
C     n_W         Water stoichiometry for SUT4
C     Cap         Membrane capacitance
C     alpha       Membrane hydraulic conductivity
C     A           Area over which solvent flux occurs
C     sigma       Osmotic reflection coefficient
C     M_HPump     Number of proton pumps in the membrane
C     lamda_HPump Rate constant for the proton pumps
C     n_H         Number of protons transported per pump cycle
C     dG0_ATP     Standard free energy release for ATP hydrolysis
C     Beta_H      Partition coefficient for protons
C     D_H         Diffusion coefficient "         "
C     delta       Membrane thickness
C     iBUFF       Flag for pH buffer in the internal solution
C     C_ATP       Flag for constant ATP concentration
C     C_ADP       Flag for constant ADP concentration
C     C_P         Flag for constant phosphate concentration
C --- ------------------------------------------------------------------
```

```
C --- Initial Conditions ------------------------------------------
      COMMON/KINIT/pH_i0,Suc_i0,v_0,Vol_0,Rho_0,n0_ATP,n0_ADP,n0_P
C     pH_i0       Initial internal pH
C     Suc_i0      Initial internal sucrose concentration
C     v_0         Initial membrane potential
C     Vol_0       Initial volume (ABAQUS DOES NOT PROVIDE)
C     Rho_0       Initial density of internal fluid
C     n0_ATP      Initial amount of ATP available (moles)
C     n0_ADP      Initial amount of ADP available (moles)
C     n0_P        Initial amount of P_io available (moles)
C --- -------------------------------------------------------------
C --- Transport Model Variables -----------------------------------
      COMMON/KVARS/pH_i,Suc_i,v,Vol,Mass_0,n_ATP,n_ADP,n_P,T,P,i_SUT4,
     1      i_HPump,i_HDiff,dVol,dPi
C     pH_i        Internal pH
C     Suc_i       Internal sucrose concentration
C     v           Membrane potential
C     Vol         Volume
C     Mass_0      Initial mass of internal fluid
C     n_ATP       Amount of ATP available (moles)
C     n_ADP       Amount of ADP available (moles)
C     n_P         Amount of P_io available (moles)
C     T           Temperature of the vesicle fluid
C     P           Hydrostatic pressure inside the vesicle
C     i_SUT4      Electric current generated by SUT4
C     i_HPump     "                            " proton pumps
C     i_HDiff     "                            " proton diffusion
C     dVol        Rate of change of inclusion volume
C     dPi         Omsotic pressure difference across the membrane
C --- -------------------------------------------------------------
C --- Step Rejection Stats ----------------------------------------
      COMMON/KSTAT/NACCPT,NREJCT,LKINC,REFNODE,ERR
C     NACCPT      Number of time steps accepted by DOPRI5
C     NREJCT      "                  " rejected by DOPRI5
C     LKINC       Index of the previous increment
C     REFNODE     Current cavity reference node
C     ERR         Root Mean Square (RMS) error for the current increment
C --- -------------------------------------------------------------
C --- Sucrose Data Vars -------------------------------------------
      COMMON/KSUCDAT/NDAT
C     NDAT        Number of data lines in the sucrose data file
C --- -------------------------------------------------------------
C
      SELECT CASE (LOP)
C ---       If this is the start of an analysis
C ---       NOTE: Unit numbers must be between 15 and 18 or greater than
C                 100 because ABAQUS uses all the others
            CASE (0)
                  CALL GETOUTDIR(OUTDIR,LENOUTDIR)
C ---             Gather and pass the sucrose data
                  OPEN(15,FILE=OUTDIR(1:LENOUTDIR)//'\SUC_DATA.TXT')
                  NDAT=0
                  DO WHILE (.NOT. EOF(15))
                        NDAT=NDAT+1
                        IF(ALLOCATED(SucData))DEALLOCATE(SucData)
                        ALLOCATE(SucData(4,NDAT))
                        SucData(1:4,1:NDAT-1)=SucDataTemp(1:4,1:NDAT-1)
                        READ(15,*) SucData(1:4,NDAT)
                        IF(ALLOCATED(SucDataTemp))DEALLOCATE(SucDataTemp)
                        ALLOCATE(SucDataTemp(4,NDAT))
                        SucDataTemp(1:4,1:NDAT)=SucData(1:4,1:NDAT)
                  END DO
                  DEALLOCATE(SucDataTemp)
                  CALL KPASSSUCDATA(1,SucData)
                  CLOSE(15,STATUS='KEEP')
C ---             Gather the input data and initial conditions
                  OPEN(15,FILE=OUTDIR(1:LENOUTDIR)//'\TRANS_INP.INP')
                  READ(15,*) pH_e
                  READ(15,*) Suc_e
                  READ(15,*) N_SUT4
                  READ(15,*) lamda_SUT4
                  READ(15,*) n_W
                  READ(15,*) Cap
                  READ(15,*) alpha
                  READ(15,*) A
                  READ(15,*) sigma
                  READ(15,*) M_HPump
                  READ(15,*) lamda_HPump
                  READ(15,*) n_H
                  READ(15,*) dG0_ATP
                  READ(15,*) Beta_H
                  READ(15,*) D_H
                  READ(15,*) delta
                  READ(15,*) pH_i0
                  READ(15,*) Suc_i0
                  READ(15,*) v_0
                  READ(15,*) Vol_0
                  READ(15,*) Rho_0
                  READ(15,*) n0_ATP
                  READ(15,*) n0_ADP
                  READ(15,*) n0_P
                  READ(15,*) N_write
                  CLOSE(15,STATUS='KEEP')
                  A=1.D-12*A
                  Mass_0=1.D-18*Rho_0*Vol_0
                  m_Suc_e=Suc_e/Rho_0
                  CALL KSOLINFO(m_Suc_e,slv_act_e,slu_act_e,osm_e)
                  IF(pH_i0 .LT. 0.D0)THEN
                        pH_i0=DABS(pH_i0)
                        iBUFF=.TRUE.
                  ELSE
                        iBUFF=.FALSE.
                  END IF
                  IF(n0_ATP .LT. 0.D0)THEN
                        n0_ATP=DABS(n0_ATP)
                        C_ATP=.TRUE.
                  ELSE
                        C_ATP=.FALSE.
                  END IF
                  IF(n0_ADP .LT. 0.D0)THEN
                        n0_ADP=DABS(n0_ADP)
                        C_ADP=.TRUE.
                  ELSE
                        C_ADP=.FALSE.
                  END IF
                  IF(n0_P .LT. 0.D0)THEN
                        n0_P=DABS(n0_P)
                        C_P=.TRUE.
                  ELSE
                        C_P=.FALSE.
                  END IF
C ---             Set up the output file
                  OPEN(16,FILE=OUTDIR(1:LENOUTDIR)//'\TRANS_OUT.OUT'
     1                ,RECL=400)
                  WRITE(16,*) 'INC TIME pH_i Suc_i v VOL nATP nADP nP P '//
     1                'dPi i_SUT4 i_HPump i_Diff dVol NACCPT NREJCT ERR'
                  WRITE(16,*) KINC,TIME(1),pH_i0,Suc_i0,v_0,Vol_0,n0_ATP
     1                ,n0_ADP,n0_P,P,dPi,i_SUT4,i_HPump,i_HDiff,dVol,NACCPT,
     2                  NREJCT,ERR
                  LKINC=0
                  N_INC=0
C ---       If this is the end of an increment, write the output data from
C           the previous increment
            CASE (2)
                  N_INC=N_INC+1
                  IF(N_INC .EQ. N_write .OR. KINC .LT. N_write)THEN
                      WRITE(16,*) LKINC,TIME(1),pH_i,Suc_i,v,Vol,n_ATP,n_ADP
     1                    ,n_P,P,dPi,i_SUT4,i_HPump,i_HDiff,dVol,NACCPT,
     2                    NREJCT,ERR
                      N_INC=0
                  END IF
C ---       Close the output files when the analysis is finished
            CASE (3)
                  IF(N_INC .NE. 0)THEN
                      WRITE(16,*) LKINC,TIME(1),pH_i,Suc_i,v,Vol,n_ATP,n_ADP
     1                    ,n_P,P,dPi,i_SUT4,i_HPump,i_HDiff,dVol,NACCPT,
     2                    NREJCT,ERR
                      N_INC=0
                  END IF
                  CLOSE(16,STATUS='KEEP')
      END SELECT
      RETURN
      END
C
C
C --- ------------------------------------------------------------------
C     Stores (IOP=1) and returns (IOP<>1) the sucrose data
C --- ------------------------------------------------------------------
*DECK KPASSSUCDATA
      SUBROUTINE KPASSSUCDATA(IOP,SucData)
C
      INCLUDE 'ABA_PARAM.INC'
C
      REAL*8 SucData
      REAL*8, ALLOCATABLE, SAVE :: SucDataStor(:,:)
      INTEGER IOP,NDAT
      DIMENSION SucData(4,NDAT)
      COMMON/KSUCDAT/NDAT
C
      IF(IOP.EQ.1)THEN
C ---       Store the sucrose data
```

```
            IF(ALLOCATED(SucDataStor))DEALLOCATE(SucDataStor)
            ALLOCATE(SucDataStor(4,NDAT))
            SucDataStor(1:4,1:NDAT)=SucData(1:4,1:NDAT)
      ELSE
C ---       Return the sucrose data
            SucData(1:4,1:NDAT)=SucDataStor(1:4,1:NDAT)
      END IF
      RETURN
      END
C
C
C --- ------------------------------------------------------------------
C     Returns the solvent activity, solute activity, and osmotic
C     coefficients for the given molal concentration of sucrose
C --- ------------------------------------------------------------------
      SUBROUTINE KSOLINFO(m_Suc,slv_act,slu_act,osm)
C
      INCLUDE 'ABA_PARAM.INC'
C
      REAL*8 m_Suc,slv_act,slu_act,osm
      REAL*8 SucData
      INTEGER NDAT,I
      DIMENSION SucData(4,NDAT)
      COMMON/KSUCDAT/NDAT
C
      CALL KPASSSUCDATA(2,SucData)
      I=1
      DO WHILE (SucData(1,I) .LT. m_Suc .AND. I .LT. NDAT)
            I=I+1
      END DO
      IF(SucData(1,I) .EQ. m_Suc .OR. I .EQ. 1 .OR. I .EQ. NDAT)THEN
            slv_act=SucData(2,I)
            slu_act=SucData(3,I)
            osm=SucData(4,I)
      ELSE
C ---       Perform linear interpolation
            slv_act=(m_Suc-SucData(1,I-1))/(SucData(1,I)-SucData(1,I-1))
     1            *(SucData(2,I)-SucData(2,I-1))+SucData(2,I-1)
            slu_act=(m_Suc-SucData(1,I-1))/(SucData(1,I)-SucData(1,I-1))
     1            *(SucData(3,I)-SucData(3,I-1))+SucData(3,I-1)
            osm=(m_Suc-SucData(1,I-1))/(SucData(1,I)-SucData(1,I-1))
     1            *(SucData(4,I)-SucData(4,I-1))+SucData(4,I-1)
      END IF
      RETURN
      END
C
C
C --- ------------------------------------------------------------------
C     Transport Model Equations
C --- ------------------------------------------------------------------
*DECK KFCN
      SUBROUTINE KFCN(N,P,T,TIME,Y,D)
C
      INCLUDE 'ABA_PARAM.INC'
C
      REAL*8 P,T,TIME,Y(N),D(N),Suc_i,m_Suc_i,v_H,v_Suc,v_W,n_ADP,n_P,
     1      dG_ATP,slv_act_i,slu_act_i,osm_i
```

```
      REAL*8 pH_e,Suc_e,m_Suc_e,slv_act_e,slu_act_e,osm_e,N_SUT4,
     1      lamda_SUT4,n_W,Cap,alpha,A,sigma,M_HPump,lamda_HPump,n_H,
     2      dG0_ATP,Beta_H,D_H,delta
      LOGICAL iBUFF,C_ATP,C_ADP,C_P
      REAL*8 pH_i0,Suc_i0,v_0,Vol_0,Rho_0,n0_ATP,n0_ADP,n0_P
      REAL*8 pH_i,Suc_iv,v,Vol,Mass_0,n_ATPv,n_ADPv,n_Pv,Tv,Pv,i_SUT4,
     1      i_HPump,i_HDiff,dVol,dPi
      REAL*8 k,e,F,R,V_bar_H2O
      INTEGER N
      COMMON/KTINP/pH_e,Suc_e,m_Suc_e,slv_act_e,slu_act_e,osm_e,N_SUT4,
     1      lamda_SUT4,n_W,Cap,alpha,A,sigma,M_HPump,lamda_HPump,n_H,
     2      dG0_ATP,Beta_H,D_H,delta,iBUFF,C_ATP,C_ADP,C_P
      COMMON/KINIT/pH_i0,Suc_i0,v_0,Vol_0,Rho_0,n0_ATP,n0_ADP,n0_P
      COMMON/KVARS/pH_i,Suc_iv,v,Vol,Mass_0,n_ATPv,n_ADPv,n_Pv,Tv,Pv,
     1      i_SUT4,i_HPump,i_HDiff,dVol,dPi
C --- Physical constants -------------------------------------------
      DATA k/1.38065812D-23/,e/1.60217733D-19/,F/96485.30929D0/,
      1     R/8.314511935D0/,V_bar_H2O/0.00001801/,W_H2O/18.015D0/
C     k           Boltzmann's constant
C     e           Elementary charge
C     F           Faraday's constant
C     R           Ideal gas constant
C     V_bar_H2O   Partial molal volume of water
C     W_H2O       Molecular weight of water
C --- -------------------------------------------------------------
C
      Suc_i=1.D-12*Y(2)/Y(4)
      IF(Suc_i .EQ. 0.D0)Suc_i=1.D-3
      m_Suc_i=Suc_i/Rho_0
      CALL KSOLINFO(m_Suc_i,slv_act_i,slu_act_i,osm_i)
C --- Osmotic pressure
      dPi=R*T*(0.1D0**(Y(1)-3.D0)-0.1D0**(pH_e-3.D0)+
     1      W_H2O/(1.D3*V_bar_H2O)*(m_Suc_i*osm_i-m_Suc_e*osm_e))
C --- Modifed Nernst equilibrium potentials
      v_H=DLOG(1.D1)*(Y(1)-pH_e)
      v_Suc=DLOG(m_Suc_e*slu_act_e/(m_Suc_i*slu_act_i))
      v_W=V_bar_H2O/F*dPi
C --- SUT4 cotransporter current
      i_SUT4=2.D0*N_SUT4*e*lamda_SUT4*rho_0
     1      *DSQRT(0.1D0**(pH_e-3.D0)*0.1D0**(Y(1)-3.D0)*m_Suc_e*slu_act_e
     2      *m_Suc_i*slu_act_i*(slv_act_e*slv_act_i)**n_W)
     3      *DSINH(5.D-1*(e*(Y(3)-n_W*v_W)/(k*T)-v_H-v_Suc))
C --- Moles of ADP and P_io
      IF(C_ATP)THEN
            V_ATP=1.D-18*Vol_0
      ELSE
            V_ATP=Y(4)
      END IF
      IF(C_ADP)THEN
            n_ADP=n0_ADP
            V_ADP=1.D-18*Vol_0
      ELSE
            n_ADP=n0_ADP+n0_ATP-Y(5)
            V_ADP=Y(4)
      END IF
      IF(C_P)THEN
            n_P=n0_P
            V_P=1.D-18*Vol_0
      ELSE
            n_P=n0_P+n0_ATP-Y(5)
            V_P=Y(4)
      END IF
C --- Free energy release from ATP hydrolysis
      dG_ATP=1.D0/F*(dG0_ATP+R*T*DLOG(n_ADP*n_P*V_ATP/(1.D3*V_ADP*V_P*
     1      Y(5))))
C --- Proton pump current
      i_HPump=M_HPump*e*lamda_HPump
     1      *DTANH(5.D-1*(e*(n_H*Y(3)-dG_ATP)/(k*T)-n_H*v_H))
C --- Proton Diffusion
      IF(1-DEXP(-1.D0*F*Y(3)/(R*T)) .NE. 0.D0)THEN
            i_HDiff=Beta_H*D_H/delta*A*F**2.D0*Y(3)/(R*T)
     1            *(0.1D0**(Y(1)-3.D0)-0.1D0**(pH_e-3.D0)
     2            *DEXP(-1.D0*F*Y(3)/(R*T)))/(1-DEXP(-1.D0*F*Y(3)/(R*T)))
      ELSE
            i_HDiff=Beta_H*D_H/delta*A*F*
     1            (0.1D0**(pH_e-3.D0)-0.1D0**(Y(1)-3.D0))
      END IF
      dVol=alpha*A*(sigma*dPi-P)
     1      -n_W*V_bar_H2O*i_SUT4/F                           ! Solvent Flux
C --- Differential Equations ---------------------------------------------
      IF(iBUFF)THEN                                           ! pH
            D(1)=0.D0
      ELSE
            D(1)=(i_SUT4+n_H*i_HPump+i_HDiff)/(F*Y(4))
     1            /(DLOG(10.D0)*0.1D0**(Y(1)-3.D0))
      END IF
      D(2)=-1.D12*i_SUT4/F                                    ! Moles sucrose
      D(3)=(-1.D0/Cap)*(i_SUT4+n_H*i_HPump+i_HDiff) ! Membrane Potential
      D(4)=dVol                                               ! Solvent Flux
      IF(C_ATP)THEN
            D(5)=0.D0
      ELSE
            D(5)=-1.D0*i_HPump/F                              ! Moles ATP
      END IF
      RETURN
      END
C
C
C --- ------------------------------------------------------------------
C     UFLUID for implementing the transport model using DOPRI5
C --- ------------------------------------------------------------------
*DECK UFLUID
      SUBROUTINE UFLUID(RHO,CP,CT,PNEWDT,ENER,PRESS,DPRESS,
      1PRESSI,TEMP,DTEMP,TEMPI,TIME,DTIME,KSTEP,KINC,NONUM,
      2FLNAME,FLAG)
C
      INCLUDE 'ABA_PARAM.INC'
C
      REAL*8 RHO,CP,CT,PNEWDT,ENER,PRESS,DPRESS,PRESSI,TEMP,DTEMP,TEMPI
      REAL*8 TIME,DTIME
      INTEGER KSTEP,KINC,NONUM,LFLAG
      CHARACTER*80 FLNAME
      DIMENSION TIME(2)
C --- ABAQUS Message Vars -----------------------------------------------
```

```
      REAL*8 REALV
      INTEGER INTV
      DIMENSION REALV(3)
      CHARACTER*8 CHARV
C     REALV Real values to be inserted in place of the %R's in the
C           message
C     INTV  Integer values to be inserted in place of the %I's in the
C           message
C     CHARV String to be inserted in place of the %S's in the message
C --- -----------------------------------------------------------------
      REAL*8 K1(5),K2(5),K3(5),K4(5),K5(5),Y1(5),Y(5),P,T,ERR,EPS,LDTIME
     1      ,LP
      REAL*8 pH_i0,Suc_i0,v_0,Vol_0,Rho_0,n0_ATP,n0_ADP,n0_P
      REAL*8 pH_i,Suc_i,v,Vol,Mass_0,n_ATP,n_ADP,n_P,i_SUT4,i_HPump,
     1      i_HDiff,dVol,dPi
      INTEGER LKINC,NACCPT,NREJCT,I,REFNODE,N
      LOGICAL REJECT
      COMMON/KINIT/pH_i0,Suc_i0,v_0,Vol_0,Rho_0,n0_ATP,n0_ADP,n0_P
      COMMON/KVARS/pH_i,Suc_i,v,Vol,Mass_0,n_ATP,n_ADP,n_P,T,P,i_SUT4,
     1      i_HPump,i_HDiff,dVol,dPi
      COMMON/KSTAT/NACCPT,NREJCT,LKINC,REFNODE,ERR
C --- DOPRI5 Parameters -----------------------------------------------
      DATA N/5/
C     EPS         Relative error tolerance
C     N           Number of equations
C --- -----------------------------------------------------------------
      CP=0.D0                 ! Fluid pressure compliance
      CT=0.D0                 ! Fluid temperature compliance
C --- -----------------------------------------------------------------
C     Fluid density at the end of a time increment
C
C     Fifth Order Explicit Runge-Kutta method of Dorman and Prince with
C     step size control (DOPRI5)
C --- -----------------------------------------------------------------
C
      EPS=1.D-4
      P=PRESS*1.D6            ! Convert cavity pressure from MPa to Pa
      T=TEMPI                 ! Pass cavity temperature to the other subs
      REFNODE=NONUM
C --- -----------------------------------------------------------------
C     If this is the start of an analysis, return the initial density
C --- -----------------------------------------------------------------
      IF(LKINC .EQ. 0)THEN
            REJECT=.FALSE.
            LKINC=1
            NACCPT=0
            NREJCT=0
            RHO=Mass_0/Vol_0
            RETURN
      END IF
C --- -----------------------------------------------------------------
C     If ABAQUS has already called UFLUID for this equilibrium
C     iteration, then return the current results
C --- -----------------------------------------------------------------
      IF(KINC .EQ. LKINC .AND. DTIME .EQ. LDTIME .AND. P .EQ. LP)THEN
            GOTO 10
      END IF
      LDTIME=DTIME
      LP=P
C --- ----------------------------------------------------------------
C     If this is the first increment, intialize
C --- ----------------------------------------------------------------
      IF(LKINC .EQ. 1)THEN
C ---       Assign initial conditions
            Y(1)=pH_i0
            Y(2)=1.D-6*Suc_i0*Vol_0
            Y(3)=v_0
            Y(4)=1.D-18*Vol_0
            Y(5)=n0_ATP
            Vol=Vol_0
            CALL KFCN(N,P,T,TIME(1),Y,K1)
      END IF
C --- If the last increment was not rejected by ABAQUS, proceed to the
C     next increment in DOPRI5
      IF(KINC .GT. LKINC)THEN
            LKINC=KINC
            NACCPT=NACCPT+1
            DO I=1,N
                  K1(I)=K3(I)
                  Y(I)=Y1(I)
            END DO
      ELSE
C ---       ABAQUS is repeating the increment, assign the variable vals
C           from the original increment
            pH_i=Y(1)
            Suc_i=1.D-12*Y(2)/Y(4)
            v=Y(3)
            Vol=1.D18*Y(4)
            n_ATP=Y(5)
            n_ADP=n0_ADP+n0_ATP-Y(5)
            n_P=n0_P+n0_ATP-Y(5)
      END IF
C --- ----------------------------------------------------------------
C     Integration Stages
C --- ----------------------------------------------------------------
C --- Stage 1
      DO I=1,N
            Y1(I)=Y(I)+DTIME*.2D0*K1(I)
      END DO
      IF(Y1(2) .LT. 0.D0 .OR. Y1(4) .LT. 0.D0 .OR. Y1(5) .LT. 0.D0)THEN
            GO TO 20
      END IF
      CALL KFCN(N,P,T,TIME(1)+.2D0*DTIME,Y1,K2)
C --- Stage 2
      DO I=1,N
            Y1(I)=Y(I)+DTIME*((3.D0/40.D0)*K1(I)+(9.D0/40.D0)*K2(I))
      END DO
      IF(Y1(2) .LT. 0.D0 .OR. Y1(4) .LT. 0.D0 .OR. Y1(5) .LT. 0.D0)THEN
            GO TO 20
      END IF
      CALL KFCN(N,P,T,TIME(1)+.3D0*DTIME,Y1,K3)
C --- Stage 3
      DO I=1,N
            Y1(I)=Y(I)+DTIME*((44.D0/45.D0)*K1(I)-(56.D0/15.D0)*K2(I)
     1            +(32.D0/9.D0)*K3(I))
      END DO
      IF(Y1(2) .LT. 0.D0 .OR. Y1(4) .LT. 0.D0 .OR. Y1(5) .LT. 0.D0)THEN
            GO TO 20
      END IF
      CALL KFCN(N,P,T,TIME(1)+.8D0*DTIME,Y1,K4)
C --- Stage 4
      DO I=1,N
            Y1(I)=Y(I)+DTIME*((19372.D0/6561.D0)*K1(I)
     1            -(25360.D0/2187.D0)*K2(I)+(64448.D0/6561.D0)*K3(I)
     2            -(212.D0/729.D0)*K4(I))
      END DO
      IF(Y1(2) .LT. 0.D0 .OR. Y1(4) .LT. 0.D0 .OR. Y1(5) .LT. 0.D0)THEN
            GO TO 20
      END IF
      CALL KFCN(N,P,T,TIME(1)+(8.D0/9.D0)*DTIME,Y1,K5)
C --- Stage 5
      DO I=1,N
            Y1(I)=Y(I)+DTIME*((9017.D0/3168.D0)*K1(I)-(355.D0/33.D0)*K2(I)
     1            +(46732.D0/5247.D0)*K3(I)+(49.D0/176.D0)*K4(I)
     2            -(5103.D0/18656.D0)*K5(I))
      END DO
      IF(Y1(2) .LT. 0.D0 .OR. Y1(4) .LT. 0.D0 .OR. Y1(5) .LT. 0.D0)THEN
            GO TO 20
      END IF
      XPH=TIME(1)+DTIME
      CALL KFCN(N,P,T,XPH,Y1,K2)
C --- Stage 6
      DO I=1,N
            Y1(I)=Y(I)+DTIME*((35.D0/384.D0)*K1(I)+(500.D0/1113.D0)*K3(I)
     1            +(125.D0/192.D0)*K4(I)-(2187.D0/6784.D0)*K5(I)
     2            +(11.D0/84.D0)*K2(I))
      END DO
C --- Intermediate sum
      DO I=1,N
            K2(I)=(71.D0/57600.D0)*K1(I)-(71.D0/16695.D0)*K3(I)
     1            +(71.D0/1920.D0)*K4(I)-(17253.D0/339200.D0)*K5(I)
     2            +(22.D0/525.D0)*K2(I)
      END DO
C --- The last stage
      CALL KFCN(N,P,T,XPH,Y1,K3)
      DO I=1,N
            K4(I)=(K2(I)-(1.D0/40.D0)*K3(I))*DTIME
      END DO
C --- -----------------------------------------------------------------
C     Error estimation
C --- -----------------------------------------------------------------
      ERR=0.D0
      DO I=1,N
            DENOM=DMAX1(1.D-5,DABS(Y1(I)),DABS(Y(I)))
            ERR=ERR+(K4(I)/DENOM)**2.D0
      END DO
      ERR=DSQRT(ERR/REAL(N,8))
C --- -----------------------------------------------------------------
C     Computation of the suggested time step ratio (PNEWDT)
C --- -----------------------------------------------------------------
C --- Require that .2 .GE. PNEWDT .LE. 5.
```

```
      PNEWDT=DMIN1(5.D0,DMAX1(.2D0,(EPS/ERR)**(1.D0/5.D0)*.85D0))
      IF(ERR .LE. EPS)THEN
C --- Step is accepted
C ---       If ABAQUS proceeds to the next increment, the output data can
C           be written to the output file, otherwise it will be reset to
C           the values at the beginning of the increment
            pH_i=Y1(1)
            Suc_i=1.D-12*Y1(2)/Y1(4)
            v=Y1(3)
            Vol=1.D18*Y1(4)
            n_ATP=Y1(5)
            n_ADP=n0_ADP+n0_ATP-Y1(5)
            n_P=n0_P+n0_ATP-Y1(5)
            IF(REJECT)PNEWDT=DMIN1(PNEWDT,1.D0)
            REJECT=.FALSE.
            IF(ERR.LE.EPS*0.1D0)PNEWDT=DMAX1(PNEWDT,1.D0)
      ELSE
C --- Step is rejected
            REALV(1)=ERR
            CALL STDB_ABQERR(1,'UFLUID-- The current step was rejected '//
     1            'by DOPRI5 with an RMS relative error of %R.',INTV,REALV,
     2            CHARV)
            REJECT=.TRUE.
            NREJCT=NREJCT+1
      END IF
10    RHO=Mass_0/Vol
      RETURN
C --- ---------------------------------------------------------------
C     If there is a negative concentration or volume, then the system is
C     becoming unstable. Reject the increment and reduce the time step.
C --- ---------------------------------------------------------------
20    CALL STDB_ABQERR(-1,'UFLUID-- An illegal value for one of the '//
     1      'model variables was calculated. Rejecting step and '//
     2      'reducing the time step.',INTV,REALV,CHARV)
      REJECT=.TRUE.
      NREJCT=NREJCT+1
      PNEWDT=0.5D0
      RHO=Mass_0/Vol
      RETURN
      END
```

APPENDIX D

USER-SUBROUTINE SET FOR IMPLEMENTING THE TRANSPORT MODEL USING THE DVODE PACKAGE

Below is an ABAQUS user-subroutine set for implementing the transport model using the Lawrence Livermore DVODE package [79] (Section 4.5). It is written using Compaq Visual FORTRAN 6.6 and is FORTRAN 77 compatible. The ABAQUS user-defined fluid (`UFLUID`) subroutine acts as a driver routine for DVODE and is responsible for coupling the transport model to the ABAQUS analysis. Step size control is implemented within the `UFLUID` subroutine by modifying the value of the variable `PNEWDT`, which is the ratio of the suggested new time increment to the time increment being used (`DTIME`). The ODE solver will attempt to use the ABAQUS time increment to perform an integration step and will set the ABAQUS variable `PNEWDT` appropriately based on its own time step algorithm. ABAQUS will repeat the current increment if `PNEWDT` is set to a value less than 1. Input and output data are read/written using the ABAQUS utility routine `UEXTERNALDB`. All coding conventions set out in the ABAQUS user-subroutine set documentation [70] are followed.

```
C === ============================================================
C     DESCRIPTION: User-defined subroutine set to incorporate the
C                  transport model for coupling the ion/solvent flux to
C                  the hyperelastic model in ABAQUS/Standard
C
C     WRITTEN BY:  Chris Homison, University of Pittsburgh
C
C     REVISIONS:   10/13/2005 Initial release
C                  12/07/2005 Added support for proton diffusion, SUT4
C                             water transport, and buffered internal pH
C                  01/13/2006 Revised ATP free energy release
C                             calculation, added currents/fluxes to the
```

```
C                            output
C === ==============================================================
C
C --- --------------------------------------------------------------
C     User subroutine for reading/writing input/output data
C
C     NOTE: No restart information is written for the user-subroutines
C --- --------------------------------------------------------------
*DECK UEXTERALDB
      SUBROUTINE UEXTERNALDB(LOP,LRESTART,TIME,DTIME,KSTEP,KINC)
C
      INCLUDE 'ABA_PARAM.INC'
C
      REAL*8 TIME,DTIME
      REAL*8, ALLOCATABLE :: SucData(:,:),SucDataTemp(:,:)
      INTEGER LOP,LRESTART,KSTEP,KINC
      DIMENSION TIME(2)
      REAL*8 pH_i0,Suc_i0,v_0,Vol_0,Rho_0,n0_ATP,n0_ADP,n0_P
      REAL*8 pH_i,Suc_i,v,Vol,Mass_0,n_ATP,n_ADP,n_P,T,P,i_SUT4,i_HPump,
     1      i_HDiff,dVol,dPi
      REAL*8 pH_e,Suc_e,m_Suc_e,slv_act_e,slu_act_e,osm_e,N_SUT4,
     1      lamda_SUT4,n_W,Cap,alpha,A,sigma,M_HPump,lamda_HPump,n_H,
     2      dG0_ATP,Beta_H,D_H,delta
      LOGICAL iBUFF,C_ATP,C_ADP,C_P
      INTEGER NDAT
      INTEGER NACCPT,NREJCT,LKINC,REFNODE,ERR
      INTEGER LENOUTDIR,N_INC,N_write
      CHARACTER*256 OUTDIR
C --- NOTE: COMMON blocks must begin with a 'K'
C --- Transport Model Input Parameters --------------------------------
      COMMON/KTINP/pH_e,Suc_e,m_Suc_e,slv_act_e,slu_act_e,osm_e,N_SUT4,
     1      lamda_SUT4,n_W,Cap,alpha,A,sigma,M_HPump,lamda_HPump,n_H,
     2      dG0_ATP,Beta_H,D_H,delta,iBUFF,C_ATP,C_ADP,C_P
C     pH_e        External pH
C     Suc_e       External sucrose concentration
C     m_Suc_e     External molal concentration of sucrose
C     N_SUT4      Number of SUT4 cotransporters in the membrane
C     lamda_SUT4  Rate constant for SUT4 cotransporters
C     n_W         Water stoichiometry for SUT4
C     Cap         Membrane capacitance
C     alpha       Membrane hydraulic conductivity
C     A           Area over which solvent flux occurs
C     sigma       Osmotic reflection coefficient
C     M_HPump     Number of proton pumps in the membrane
C     lamda_HPump Rate constant for the proton pumps
C     n_H         Number of protons transported per pump cycle
C     dG0_ATP     Standard free energy release for ATP hydrolysis
C     Beta_H      Partition coefficient for protons
C     D_H         Diffusion coefficient "         "
C     delta       Membrane thickness
C     iBUFF       Flag for pH buffer in the internal solution
C     C_ATP       Flag for constant ATP concentration
C     C_ADP       Flag for constant ADP concentration
C     C_P         Flag for constant phosphate concentration
C --- --------------------------------------------------------------
C --- Initial Conditions --------------------------------------------
      COMMON/KINIT/pH_i0,Suc_i0,v_0,Vol_0,Rho_0,n0_ATP,n0_ADP,n0_P
```

```
C     pH_i0       Initial internal pH
C     Suc_i0      Initial internal sucrose concentration
C     v_0         Initial membrane potential
C     Vol_0       Initial volume (ABAQUS DOES NOT PROVIDE)
C     Rho_0       Initial density of internal fluid
C     n0_ATP      Initial amount of ATP available (moles)
C     n0_ADP      Initial amount of ADP available (moles)
C     n0_P        Initial amount of P_io available (moles)
C --- ------------------------------------------------------------------
C --- Transport Model Variables ----------------------------------------
      COMMON/KVARS/pH_i,Suc_i,v,Vol,Mass_0,n_ATP,n_ADP,n_P,T,P,i_SUT4,
     1      i_HPump,i_HDiff,dVol,dPi
C     pH_i        Internal pH
C     Suc_i       Internal sucrose concentration
C     v           Membrane potential
C     Vol         Volume
C     Mass_0      Initial mass of internal fluid
C     n_ATP       Amount of ATP available (moles)
C     n_ADP       Amount of ADP available (moles)
C     n_P         Amount of P_io available (moles)
C     T           Temperature of the vesicle fluid
C     P           Hydrostatic pressure inside the vesicle
C     i_SUT4      Electric current generated by SUT4
C     i_HPump     "                           " proton pumps
C     i_HDiff     "                           " proton diffusion
C     dVol        Rate of change of inclusion volume
C     dPi         Omsotic pressure difference across the membrane
C --- ------------------------------------------------------------------
C --- Step Rejection Stats ---------------------------------------------
      COMMON/KSTAT/NACCPT,NREJCT,LKINC,REFNODE
C     NACCPT      Number of time steps accepted by DOPRI5
C     NREJCT      "                  " rejected by DOPRI5
C     LKINC       Index of the previous increment
C     REFNODE     Current cavity reference node
C --- ------------------------------------------------------------------
C --- Sucrose Data Vars ------------------------------------------------
      COMMON/KSUCDAT/NDAT
C     NDAT        Number of data lines in the sucrose data file
C --- ------------------------------------------------------------------
C
      SELECT CASE (LOP)
C ---       If this is the start of an analysis
C ---       NOTE: Unit numbers must be between 15 and 18 or greater than
C                 100 because ABAQUS uses all the others
            CASE (0)
                  CALL GETOUTDIR(OUTDIR,LENOUTDIR)
C ---             Gather and pass the sucrose data
                  OPEN(15,FILE=OUTDIR(1:LENOUTDIR)//'\SUC_DATA.TXT')
                  NDAT=0
                  DO WHILE (.NOT. EOF(15))
                        NDAT=NDAT+1
                        IF(ALLOCATED(SucData))DEALLOCATE(SucData)
                        ALLOCATE(SucData(4,NDAT))
                        SucData(1:4,1:NDAT-1)=SucDataTemp(1:4,1:NDAT-1)
                        READ(15,*) SucData(1:4,NDAT)
                        IF(ALLOCATED(SucDataTemp))DEALLOCATE(SucDataTemp)
                        ALLOCATE(SucDataTemp(4,NDAT))
                        SucDataTemp(1:4,1:NDAT)=SucData(1:4,1:NDAT)
                  END DO
                  DEALLOCATE(SucDataTemp)
                  CALL KPASSSUCDATA(1,SucData)
                  CLOSE(15,STATUS='KEEP')
C ---             Gather the input data and initial conditions
                  OPEN(15,FILE=OUTDIR(1:LENOUTDIR)//'\TRANS_INP.INP')
                  READ(15,*) pH_e
                  READ(15,*) Suc_e
                  READ(15,*) N_SUT4
                  READ(15,*) lamda_SUT4
                  READ(15,*) n_W
                  READ(15,*) Cap
                  READ(15,*) alpha
                  READ(15,*) A
                  READ(15,*) sigma
                  READ(15,*) M_HPump
                  READ(15,*) lamda_HPump
                  READ(15,*) n_H
                  READ(15,*) dG0_ATP
                  READ(15,*) Beta_H
                  READ(15,*) D_H
                  READ(15,*) delta
                  READ(15,*) pH_i0
                  READ(15,*) Suc_i0
                  READ(15,*) v_0
                  READ(15,*) Vol_0
                  READ(15,*) Rho_0
                  READ(15,*) n0_ATP
                  READ(15,*) n0_ADP
                  READ(15,*) n0_P
                  READ(15,*) N_write
                  CLOSE(15,STATUS='KEEP')
                  A=1.D-12*A
                  Mass_0=1.D-18*Rho_0*Vol_0
                  m_Suc_e=Suc_e/Rho_0
                  CALL KSOLINFO(m_Suc_e,slv_act_e,slu_act_e,osm_e)
                  IF(pH_i0 .LT. 0.D0)THEN
                        pH_i0=DABS(pH_i0)
                        iBUFF=.TRUE.
                  ELSE
                        iBUFF=.FALSE.
                  END IF
                  IF(n0_ATP .LT. 0.D0)THEN
                        n0_ATP=DABS(n0_ATP)
                        C_ATP=.TRUE.
                  ELSE
                        C_ATP=.FALSE.
                  END IF
                  IF(n0_ADP .LT. 0.D0)THEN
                        n0_ADP=DABS(n0_ADP)
                        C_ADP=.TRUE.
                  ELSE
                        C_ADP=.FALSE.
                  END IF
                  IF(n0_P .LT. 0.D0)THEN
                        n0_P=DABS(n0_P)
                        C_P=.TRUE.
                  ELSE
                        C_P=.FALSE.
                  END IF
C ---       Set up the output file
                  OPEN(16,FILE=OUTDIR(1:LENOUTDIR)//'\TRANS_OUT.OUT'
     1            ,RECL=400)
                  WRITE(16,*) 'INC TIME pH_i Suc_i v VOL nATP nADP nP P '//
     1            'dPi i_SUT4 i_HPump i_Diff dVol NACCPT NREJCT'
                  WRITE(16,*) KINC,TIME(1),pH_i0,Suc_i0,v_0,Vol_0,n0_ATP,
     1            n0_ADP,n0_P,P,dPi,i_SUT4,i_HPump,i_HDiff,dVol,NACCPT,
     2            NREJCT
                  LKINC=0
                  N_INC=0
C ---    If this is the end of an increment, write the output data from
C        the previous increment
            CASE (2)
                  N_INC=N_INC+1
                  IF(N_INC .EQ. N_write .OR. KINC .LT. N_write)THEN
                      WRITE(16,*) LKINC,TIME(1),pH_i,Suc_i,v,Vol,n_ATP,n_ADP
     1              ,n_P,P,dPi,i_SUT4,i_HPump,i_HDiff,dVol,NACCPT,
     2              NREJCT
                      N_INC=0
                  END IF
C ---    Close the output files when the analysis is finished
            CASE (3)
                  IF(N_INC .NE. 0)THEN
                      WRITE(16,*) LKINC,TIME(1),pH_i,Suc_i,v,Vol,n_ATP,n_ADP
     1              ,n_P,P,dPi,i_SUT4,i_HPump,i_HDiff,dVol,NACCPT,
     2              NREJCT
                      N_INC=0
                  END IF
                  CLOSE(16,STATUS='KEEP')
      END SELECT
      RETURN
      END
C
C
C --- -----------------------------------------------------------------
C     Stores (IOP=1) and returns (IOP<>1) the sucrose data
C --- -----------------------------------------------------------------
*DECK KPASSSUCDATA
      SUBROUTINE KPASSSUCDATA(IOP,SucData)
C
      INCLUDE 'ABA_PARAM.INC'
C
      REAL*8 SucData
      REAL*8, ALLOCATABLE, SAVE :: SucDataStor(:,:)
      INTEGER IOP,NDAT
      DIMENSION SucData(4,NDAT)
      COMMON/KSUCDAT/NDAT
C
      IF(IOP.EQ.1)THEN
C ---    Store the sucrose data
            IF(ALLOCATED(SucDataStor))DEALLOCATE(SucDataStor)
            ALLOCATE(SucDataStor(4,NDAT))
            SucDataStor(1:4,1:NDAT)=SucData(1:4,1:NDAT)
      ELSE
C ---       Return the sucrose data
            SucData(1:4,1:NDAT)=SucDataStor(1:4,1:NDAT)
      END IF
      RETURN
      END
C
C
C --- ------------------------------------------------------------
C     Returns the solvent activity, solute activity, and osmotic
C     coefficients for the given molal concentration of sucrose
C --- ------------------------------------------------------------
      SUBROUTINE KSOLINFO(m_Suc,slv_act,slu_act,osm)
C
      INCLUDE 'ABA_PARAM.INC'
C
C --- ABAQUS Message Vars ----------------------------------------
      REAL*8 REALV
      INTEGER INTV
      CHARACTER*8 CHARV
C     REALV Real values to be inserted in place of the %R's in the
C           message
C     INTV  Integer values to be inserted in place of the %I's in the
C           message
C     CHARV String to be inserted in place of the %S's in the message
C --- ------------------------------------------------------------
      REAL*8 m_Suc,slv_act,slu_act,osm
      REAL*8 SucData
      INTEGER NDAT,I
      DIMENSION SucData(4,NDAT)
      COMMON/KSUCDAT/NDAT
C
      CALL KPASSSUCDATA(2,SucData)
      I=1
      DO WHILE (SucData(1,I) .LT. m_Suc .AND. I .LT. NDAT)
            I=I+1
      END DO
      IF(SucData(1,I) .EQ. m_Suc .OR. I .EQ. 1 .OR. (I .EQ. NDAT .AND.
     1  SucData(1,I) .LT. m_Suc))THEN
            IF((I .EQ. 1 .OR. I .EQ. NDAT) .AND. SucData(1,I) .NE. m_Suc)
     1        THEN
                  CALL STDB_ABQERR(-1,'KSOLINFO-- The molal concentration'//
     1                ' of sucrose is outside the limits of the data.',INTV,
     2                REALV,CHARV)
            END IF
            slv_act=SucData(2,I)
            slu_act=SucData(3,I)
            osm=SucData(4,I)
      ELSE
C ---       Perform linear interpolation
            slv_act=(m_Suc-SucData(1,I-1))/(SucData(1,I)-SucData(1,I-1))
     1            *(SucData(2,I)-SucData(2,I-1))+SucData(2,I-1)
            slu_act=(m_Suc-SucData(1,I-1))/(SucData(1,I)-SucData(1,I-1))
     1            *(SucData(3,I)-SucData(3,I-1))+SucData(3,I-1)
            osm=(m_Suc-SucData(1,I-1))/(SucData(1,I)-SucData(1,I-1))
     1            *(SucData(4,I)-SucData(4,I-1))+SucData(4,I-1)
      END IF
      RETURN
      END
C
C
C --- ----------------------------------------------------------------
C     Transport Model Equations
C --- ----------------------------------------------------------------
*DECK KFCN
      SUBROUTINE KFCN(N,TIME,Y,D,RPAR,IPAR)
C
      INCLUDE 'ABA_PARAM.INC'
C
      REAL*8 TIME,Y,D,RPAR,Suc_iv,m_Suc_i,v_H,v_Suc,v_W,dG_ATP,slv_act_i
     1      ,slu_act_i,osm_i,V_ATP,V_ADP,V_P
      INTEGER IPAR,N
      DIMENSION Y(N),D(N),RPAR(*),IPAR(*)
      REAL*8 pH_e,Suc_e,m_Suc_e,slv_act_e,slu_act_e,osm_e,N_SUT4,
     1      lamda_SUT4,n_W,Cap,alpha,A,sigma,M_HPump,lamda_HPump,n_H,
     2      dG0_ATP,Beta_H,D_H,delta
      LOGICAL iBUFF,C_ATP,C_ADP,C_P
      REAL*8 pH_i0,Suc_i0,v_0,Vol_0,Rho_0,n0_ATP,n0_ADP,n0_P
      REAL*8 pH_i,Suc_i,v,Vol,Mass_0,n_ATP,n_ADP,n_P,T,P,i_SUT4,i_HPump,
     1      i_HDiff,dVol,dPi
      REAL*8 k,e,F,R,V_bar_H2O
      COMMON/KTINP/pH_e,Suc_e,m_Suc_e,slv_act_e,slu_act_e,osm_e,N_SUT4,
     1      lamda_SUT4,n_W,Cap,alpha,A,sigma,M_HPump,lamda_HPump,n_H,
     2      dG0_ATP,Beta_H,D_H,delta,iBUFF,C_ATP,C_ADP,C_P
      COMMON/KINIT/pH_i0,Suc_i0,v_0,Vol_0,Rho_0,n0_ATP,n0_ADP,n0_P
      COMMON/KVARS/pH_i,Suc_i,v,Vol,Mass_0,n_ATP,n_ADP,n_P,T,P,i_SUT4,
     1      i_HPump,i_HDiff,dVol,dPi
C --- Physical constants ---------------------------------------------
      DATA k/1.38065812D-23/,e/1.60217733D-19/,F/96485.30929D0/,
      1     R/8.314511935D0/,V_bar_H2O/0.00001801/,W_H2O/18.015D0/
C     k           Boltzmann's constant
C     e           Elementary charge
C     F           Faraday's constant
C     R           Ideal gas constant
C     V_bar_H2O   Partial molal volume of water
C     W_H2O       Molecular weight of water
C --- ----------------------------------------------------------------
C
      Suc_iv=1.D-12*Y(2)/Y(4)
      IF(Suc_iv .EQ. 0.D0)Suc_iv=1.D-3
      m_Suc_i=Suc_iv/Rho_0
      CALL KSOLINFO(m_Suc_i,slv_act_i,slu_act_i,osm_i)
C --- Osmotic pressure
      dPi=R*T*(0.1D0**(Y(1)-3.D0)-0.1D0**(pH_e-3.D0)+
     1      W_H2O/(1.D3*V_bar_H2O)*(m_Suc_i*osm_i-m_Suc_e*osm_e))
C --- Modifed Nernst equilibrium potentials
      v_H=DLOG(1.D1)*(Y(1)-pH_e)
      v_Suc=DLOG(m_Suc_e*slu_act_e/(m_Suc_i*slu_act_i))
      v_W=V_bar_H2O/F*dPi
C --- SUT4 cotransporter current
      i_SUT4=2.D0*N_SUT4*e*lamda_SUT4*Rho_0
     1      *DSQRT(0.1D0**(pH_e-3.D0)*0.1D0**(Y(1)-3.D0)*m_Suc_e*slu_act_e
     2      *m_Suc_i*slu_act_i*(slv_act_e*slv_act_i)**n_W)
     3      *DSINH(5.D-1*(e*(Y(3)-n_W*v_W)/(k*T)-v_H-v_Suc))
```

```
C --- Moles of ADP and P_io
      IF(C_ATP)THEN
            V_ATP=1.D-18*Vol_0
      ELSE
            V_ATP=Y(4)
      END IF
      IF(C_ADP)THEN
            n_ADP=n0_ADP
            V_ADP=1.D-18*Vol_0
      ELSE
            n_ADP=n0_ADP+n0_ATP-Y(5)
            V_ADP=Y(4)
      END IF
      IF(C_P)THEN
            n_P=n0_P
            V_P=1.D-18*Vol_0
      ELSE
            n_P=n0_P+n0_ATP-Y(5)
            V_P=Y(4)
      END IF
C --- Free energy release from ATP hydrolysis
      dG_ATP=1.D0/F*(dG0_ATP+R*T*DLOG(n_ADP*n_P*V_ATP/(1.D3*V_ADP*V_P*
     1      Y(5))))
C --- Proton pump current
      i_HPump=M_HPump*e*lamda_HPump
     1      *DTANH(5.D-1*(e*(n_H*Y(3)-dG_ATP)/(k*T)-n_H*v_H))
C --- Proton Diffusion
      IF(1-DEXP(-1.D0*F*Y(3)/(R*T)) .NE. 0.D0)THEN
            i_HDiff=Beta_H*D_H/delta*A*F**2.D0*Y(3)/(R*T)
     1            *(0.1D0**(Y(1)-3.D0)-0.1D0**(pH_e-3.D0)
     2            *DEXP(-1.D0*F*Y(3)/(R*T)))/(1-DEXP(-1.D0*F*Y(3)/(R*T)))
      ELSE
            i_HDiff=Beta_H*D_H/delta*A*F*
     1            (0.1D0**(pH_e-3.D0)-0.1D0**(Y(1)-3.D0))
      END IF
      dVol=alpha*A*(sigma*dPi-P)
     1      -n_W*V_bar_H2O*i_SUT4/F                              ! Solvent Flux
C --- Differential Equations ------------------------------------------
      IF(iBUFF)THEN                                              ! pH
            D(1)=0.D0
      ELSE
            D(1)=(i_SUT4+n_H*i_HPump+i_HDiff)/(F*Y(4))
     1            /(DLOG(10.D0)*0.1D0**(Y(1)-3.D0))
      END IF
      D(2)=-1.D12*i_SUT4/F                                       ! Moles sucrose
      D(3)=(-1.D0/Cap)*(i_SUT4+n_H*i_HPump+i_HDiff) ! Membrane Potential
      D(4)=dVol                                                  ! Solvent Flux
      IF(C_ATP)THEN
            D(5)=0.D0
      ELSE
            D(5)=-1.D0*i_HPump/F                                 ! Moles ATP
      END IF
      RETURN
      END
C
C
C --- -----------------------------------------------------------------
```

```
C     User subroutine for implementing the transport model using DVODE
C --- -----------------------------------------------------------------
*DECK UFLUID
      SUBROUTINE UFLUID(RHO,CP,CT,PNEWDT,ENER,PRESS,DPRESS,
      1PRESSI,TEMP,DTEMP,TEMPI,TIME,DTIME,KSTEP,KINC,NONUM,
      2FLNAME,LFLAG)
C
      INCLUDE 'ABA_PARAM.INC'
C
      EXTERNAL KFCN
      REAL*8 RHO,CP,CT,PNEWDT,ENER,PRESS,DPRESS,PRESSI,TEMP,DTEMP,TEMPI
      REAL*8 TIME,DTIME
      INTEGER KSTEP,KINC,NONUM,LFLAG
      CHARACTER*80 FLNAME
      DIMENSION TIME(2)
C --- ABAQUS Message Vars ---------------------------------------------
      REAL*8 REALV
      INTEGER INTV
      DIMENSION REALV(3)
      CHARACTER*8 CHARV
C     REALV Real values to be inserted in place of the %R's in the
C           message
C     INTV  Integer values to be inserted in place of the %I's in the
C           message
C     CHARV String to be inserted in place of the %S's in the message
C --- -----------------------------------------------------------------
C --- DVODE Parameters ------------------------------------------------
      INTEGER N,ITOL,ITASK,ISTATE,IOPT,LRW,IWORK,LIW,MF,IPAR
      DATA N/5/
C     N           Number of equations
C     ITOL        Indicator for the type of error control
C     ITASK       Task to be performed
C     ISTATE      State of the calculation
C     IOPT        Flag for optional input
C     LRW         Length of the real working array
C     IWORK       Integer working array
C     LIW         Length of the integer working array
C     MF          Method flag
C     IPAR        Parameter for communicating with user-supplied
C                 subroutines
      REAL*8 TI,Y1,Y,TOUT,RTOL,ATOL,RWORK,RPAR
C     TI          Value of time successfully reached during integration
C     Y1          Vector of dependent variables
C     Y           Y1 from the previous increment
C     TOUT        Value of time that DVODE reached or is to reach
C     RTOL        Relative error tolerance
C     ATOL        Absolute error tolerance vector
C     RWORK       Real working array
C     RPAR        Parameter for communicating with user-supplied
C                 subroutines
      DIMENSION Y(5),Y1(5),ATOL(5),RWORK(183),IWORK(37)
C --- -----------------------------------------------------------------
C --- Stored DVODE Vars for Restarting Integration --------------------
      REAL*8 RWORK_temp,RSAV,RPAR_temp
      INTEGER IWORK_temp,ISAV,IPAR_temp
      DIMENSION RWORK_temp(183),IWORK_temp(37),RSAV(49),ISAV(41)
C --- -----------------------------------------------------------------
```

```
C --- Locally Used Vars ------------------------------------------
      REAL*8 LDTIME,LP,NEWDT,MSGDT,dV,dV_max,D
      DIMENSION D(5)
      INTEGER I,NUMSE,NUMSE_max,MSGINC
      DATA NUMSE_max/15/,dV_max/5.D1/
C     LDTIME      Last value of DTIME for which integration proceeded
C     LP          Last value of P for which integration proceeded
C     NEWDT       The value of PNEWDT applied after each iteration
C     MSGDT       Value of DTIME for which the last message was written
C     dV          Change in volume for the current increment/attempt
C     dV_max      Maximum allowed change in volume per time increment
C     I           Generic index variable
C     NUMSE       Number of times the number of steps has been exceeded
C     NUMSE_max   Maximum ""
C     MSGINC      Increment for which the last message was written
C     D           Values of the derivatives
C --- ------------------------------------------------------------
C --- COMMON Block Vars ------------------------------------------
      REAL*8 pH_i0,Suc_i0,v_0,Vol_0,Rho_0,n0_ATP,n0_ADP,n0_P
      REAL*8 pH_i,Suc_i,v,Vol,Mass_0,n_ATP,n_ADP,n_P,T,P,i_SUT4,i_HPump,
     1      i_HDiff,dVol,dPi
      INTEGER NACCPT,NREJCT,LKINC,REFNODE
      COMMON/KINIT/pH_i0,Suc_i0,v_0,Vol_0,Rho_0,n0_ATP,n0_ADP,n0_P
      COMMON/KVARS/pH_i,Suc_i,v,Vol,Mass_0,n_ATP,n_ADP,n_P,T,P,i_SUT4,
     1      i_HPump,i_HDiff,dVol,dPi
      COMMON/KSTAT/NACCPT,NREJCT,LKINC,REFNODE
C --- ------------------------------------------------------------
      CP=0.D0               ! Fluid pressure compliance
      CT=0.D0               ! Fluid temperature compliance
C --- ------------------------------------------------------------
C     Fluid density at the end of a time increment
C --- ------------------------------------------------------------
      P=PRESS*1.D6          ! Convert cavity pressure from MPa to Pa
      T=TEMP                ! Pass cavity temperature to the other subs
      REFNODE=NONUM
C --- If this is the start of an analysis, return the initial density
      IF(KINC.EQ.0)THEN
            LKINC=0
            NACCPT=0
            NREJCT=0
            NUMSE=0
            Vol=Vol_0
            GOTO 100
      END IF
C --- If ABAQUS has already called UFLUID for this equilibrium
C     iteration, then return the current results
      IF(KINC.EQ.LKINC .AND. DTIME.EQ.LDTIME .AND. P.EQ.LP)THEN
            GOTO 100
      END IF
      LDTIME=DTIME
      LP=P
      TOUT=TIME(1)+DTIME
      LRW=183
      LIW=37
      MF=22                 ! Internally generated full Jacobian
      ITASK=1               ! Have DVODE proceed to TOUT
C --- Initial Preparations
```

```
      IF(KINC.LE.1)THEN
            TI=0.D0
C ---       Set up the error control to have a scalar relative error
C           tolerance and a vector absolute tolerance
            ITOL=2
            RTOL=1.D-7
            ATOL(1)=1.D-1
            ATOL(2)=1.D0
            ATOL(3)=1.D-5
            ATOL(4)=1.D-12
            ATOL(5)=1.D-7
            ISTATE=1          ! Have DVODE initialize
            IOPT=1            ! Optional input will be used
C ---       Assign initial conditions
            Y1(1)=pH_i0
            Y1(2)=1.D-6*Suc_i0*Vol_0
            Y1(3)=v_0
            Y1(4)=1.D-18*Vol_0
            Y1(5)=n0_ATP
            DO I=1,N
                  Y(I)=Y1(I)
            END DO
            Vol=Vol_0
            RWORK(5)=DTIME    ! Attempt DTIME as the initial step size
      ELSE
            ISTATE=3          ! Have DVODE proceed with changed HMAX
      END IF
C --- If the last increment was NOT rejected by ABAQUS, proceed to
C     the next time step in DVODE
      IF(KINC.GT.LKINC)THEN
            NUMSE=0
            LKINC=KINC
C ---       Store the DVODE vars so that integration can be restarted
C           from the beginning of the increment if the current step
C           is rejected
            DO I=1,N
                  Y(I)=Y1(I)
            END DO
            DO I=1,LRW
                  RWORK_temp(I)=RWORK(I)
            END DO
            DO I=1,LIW
                  IWORK_temp(I)=IWORK(I)
            END DO
            CALL KDVSRCO(RSAV,ISAV,1)
            RPAR_temp=RPAR
            IPAR_temp=IPAR
      ELSE
C ---       ABAQUS is repeating the increment.
C ---       Assign the variable vals from the original increment
            pH_i=Y(1)
            Suc_i=1.D-12*Y(2)/Y(4)
            v=Y(3)
            Vol=1.D18*Y(4)
            n_ATP=Y(5)
            TI=TIME(1)
            DO I=1,N
```

```
                  Y1(I)=Y(I)
            END DO
            DO I=1,LRW
                  RWORK(I)=RWORK_temp(I)
            END DO
            DO I=1,LIW
                  IWORK(I)=IWORK_temp(I)
            END DO
            CALL KDVSRCO(RSAV,ISAV,2)
            RPAR=RPAR_temp
            IPAR=IPAR_temp
      END IF
      RWORK(6)=1.5D0*DTIME    ! Limit HMAX
      IWORK(6)=1              ! Allow 1 step to reach TOUT
C --- Call DVODE
      CALL KDVODE(KFCN,N,Y1,TI,TOUT,ITOL,RTOL,ATOL,ITASK,ISTATE,IOPT,
     1      RWORK,LRW,IWORK,LIW,KFCN,MF,RPAR,IPAR)
C --- Error Handler
      REALV(1)=TI
      SELECT CASE (ISTATE)
            CASE (1)
                  REALV(1)=TOUT
                  CALL STDB_ABQERR(-2,'UFLUID-- Integration did '//
     1                'not proceed because TOUT = T = %R s'
     2                ,INTV,REALV,CHARV)
                  GOTO 100
C           CASE (-1)
            CASE (-10)
                  PNEWDT=9.99999999999D-1*TI/TOUT
                  NUMSE=NUMSE+1
                  REALV(2)=PNEWDT*DTIME
                  CALL STDB_ABQERR(-1,'UFLUID-- The maximum '//
     1                'number of steps was exceeded. Integration '//
     2                'succesfully reached time %R s. A new time step '//
     3                'of %R will be suggested.',INTV,REALV,CHARV)
                  IF(NUMSE.GE.NUMSE_max)THEN
                      INTV=NUMSE_max
                      CALL STDB_ABQERR(-3,'UFLUID-- The above '//
     1                    'error message was displayed %I times. '//
     2                    'Integration succesfully reached time %R s'
     3                    ,INTV,REALV,CHARV)
                  END IF
                  GOTO 100
            CASE (-2)
                  REALV(1)=RWORK(14)
                  REALV(2)=TI
                  CALL STDB_ABQERR(-3,'UFLUID-- Accuracy requested '//
     1                'exceeds machine precision. A suggested scale '//
     2                'factor of %R should be applied to the absolute '//
     3                'and/or relative tolerances. Integration '//
     4                'successfully reached time %R s',INTV,REALV,CHARV)
                  GOTO 100
            CASE (-3)
                  CALL STDB_ABQERR(-3,'UFLUID-- An illegal input was '//
     1                'provided to DVODE',INTV,REALV,CHARV)
                  GOTO 100
            CASE (-4)
```

```
                INTV=IWORK(16)
                CALL STDB_ABQERR(-3,'UFLUID-- Repeated error test '//
     1              'failures on this step with greatest error in '//
     2              'variable %I. Integration successfully reached '//
     3              'time %R s',INTV,REALV,CHARV)
                GOTO 100
            CASE (-5)
                INTV=IWORK(16)
                CALL STDB_ABQERR(-3,'UFLUID-- Repeated convergence '//
     1              'test failures on this step with greatest error in '//
     2              'variable %I. Integration successfully reached '//
     3              'time %R s',INTV,REALV,CHARV)
                GOTO 100
            CASE (-6)
                CALL STDB_ABQERR(-3,'UFLUID-- One of the error '//
     1              'weighting factors became zero. Integration '//
     2              'successfully reached time %R s',INTV,REALV,CHARV)
                GOTO 100
        END SELECT
        NACCPT=IWORK(11)
        IF(TI.GE.TOUT)THEN
C --- Step is accepted
C ---       If ABAQUS proceeds to the next increment, the output data
C           can be written to the output file, otherwise it will be
C           reset to the original values at the start of the increment.
            pH_i=Y1(1)
            Suc_i=1.D-12*Y1(2)/Y1(4)
            v=Y1(3)
            Vol=1.D18*Y1(4)
            n_ATP=Y1(5)
            NEWDT=DMAX1(1.D0,9.99999999999D-1*RWORK(12)/DTIME)
        ELSE
C --- Step is rejected
            NEWDT=9.99999999999999D-1*RWORK(11)/DTIME
            IF(MSGDT.NE.DTIME .OR. MSGINC.NE.KINC)THEN
                MSGDT=DTIME
                MSGINC=KINC
                NREJCT=NREJCT+1
                REALV(1)=DTIME
                REALV(2)=RWORK(11)
                REALV(3)=NEWDT*DTIME
                CALL STDB_ABQERR(1,'UFLUID-- The current step was '//
     1              'rejected because the ABAQUS time step %R s '//
     2              'exceeds the DVODE time step %R s. The suggested '//
     3              'time step will be %R s.',INTV,REALV,CHARV)
            END IF
            GOTO 100
        END IF
C --- To maintain stability, the volume change must be limited
        dV=1.D18*(Y1(4)-Y(4))
        IF(DABS(dV).GT.dV_max)THEN
            NEWDT=9.D-1*dV_max/DABS(dV)
            NREJCT=NREJCT+1
            REALV(1)=dV
            REALV(2)=dV_max
            REALV(3)=NEWDT*DTIME
            CALL STDB_ABQERR(1,'UFLUID-- The current step was '//
     1          'rejected because the volume change of %R exceeds '//
     2          'the maximum allowed volume change %R. The suggested '//
     3          'time step will be %R s.',INTV,REALV,CHARV)
            GOTO 100
      END IF
C --- This is to ensure that the values of the currents are calculated
C     using the current variable values and not some trial values set
C     by DVODE
      CALL KFCN(N,TOUT,Y1,D,RPAR,IPAR)
C --- Check for illegal values in the results
100   IF((Vol.LE.0.D0 .OR. Y1(1).LE.0.D0 .OR. Y1(2).LT.0.D0) .AND.
     1      KINC.GT.0)THEN
            PNEWDT=DMIN1(NEWDT,7.5D-1)
            IF(MSGDT.NE.DTIME .OR. MSGINC.NE.KINC)THEN
                MSGDT=DTIME
                MSGINC=KINC
                REALV(1)=PNEWDT*DTIME
                CALL STDB_ABQERR(1,'UFLUID-- The current step was '//
     1              'rejected because a negative volume or '//
     2              'concentration was encountered. The suggested '//
     3              'time step will be %R s.',INTV,REALV,CHARV)
            END IF
            RHO=Mass_0/Y(4)
      ELSE
            PNEWDT=NEWDT
            RHO=Mass_0/Vol
      END IF
      RETURN
      END
C
C
C --- -----------------------------------------------------------------
C                                    DVODE ODE Solver Package
C
C     SOURCE: Lawrence Livermore National Laboratory
C
C     NOTE: This package has been adapted for use in an ABAQUS
C           user-defined subroutine set:
C           (i)   The names of all subroutines and functions were
C                 changed to start with a 'K'
C           (ii)  The names of all COMMON blocks were changed to
C                 begin with a 'K'
C           (iii) An INCLUDE statement for the ABAQUS Parameter
C                 file was added to all subroutines and functions.
C           (iv)  Replaced calls to error message writer subroutine
C                 with calls to STDB_ABQERR
C --- -----------------------------------------------------------------
```

Modified source code for DVODE package not included.

APPENDIX E

SOURCE CODE FOR UFLUID DRIVER PROGRAM

The UFLUID Driver Program (Section 4.5) executes an ABAQUS `UFLUID` user-subroutine set in the same manner as ABAQUS and features a Windows user-interface (source code not included) that contains a progress bar and a **Run** button as well as an **Exit** button. Subroutines that mimic the ABAQUS utility routines `GETOUTDIR`, `STDB_ABQERR`, and `XIT` [70] are included and the `UEXTERNALDB` user-subroutine is appropriately called. When the user clicks the **Run** button, the `RUNTEST` subroutine is called which acts as the driver routine for the user-subroutine set.

```
C === ================================================================
C     DESCRIPTION:      Exectutes an ABAQUS UFLUID User-Subroutine Set
C
C     WRITTEN BY:       Chris Homison
C
C     USE: 1. Make sure that the correct and current ABAQUS parameter
C             file is inserted in the working directory of this project
C             and is named 'ABA_PARAM.INC'.
C          2. Place a copy of the user-subroutine source code file in
C             the working directory of this project OR place a copy of
C             ABA_PARAM.INC in the directory with the user-subroutine
C             source code and add it to this project.
C          3. Update the PRESS polynomial for the desired geometry and
C             material properties.
C          4. Set the adjustable parameters per their definitions below.
C === ================================================================
C
C --- ----------------------------------------------------------------
C     Driver routine for mimicing the behavior of an ABAQUS simulation
C     with a user-defined fluid constitutive model subroutine (UFLUID)
C --- ----------------------------------------------------------------
      SUBROUTINE RUNTEST(PRESS_FEED)
C
```

```
      USE dfwin
      USE dflogm
      USE UFLUID_TestGlobals
C
      INCLUDE 'ABA_PARAM.INC'
      INCLUDE 'resource.fd'
C
      LOGICAL*4 PRESS_FEED,lret
      REAL*8 RHO,CP,CT,PNEWDT,ENER,PRESS,DPRESS
      REAL*8 PRESSI,TEMP,DTEMP,TEMPI,TIME,DTIME,RHO0,DTIME0,TIME_MAX
      REAL*8 PRESS_old,RHO_old,RDPRESSTOL,RDRHOTOL,Vol_0,Vol,DTIME_MAX
      REAL*8 DTIME_MIN
C --- Adjustable Parameters ---------------------------------------------
C     TEMP        Fluid temperature
C     TEMPI       Initial fluid temperature
C     DTIME0      Initial time increment size to attempt
C     TIME_MAX    Length of time to simulate
C     Vol_0       Initial volume of inclusion
C     KINC_MAX    Maximum number of increments in a run
C     NUMCT_MAX   Maximum number of cutbacks per increment
C     DTIME_MAX   Maximum value of time increment
C     DTIME_MIN   Minimum "                      "
C     NumElem     The number of hydrostatic fluid elements to call for
C     NumIt_max   Maximum number of equilibrium iterations per increment
C     RDPRESSTOL  Maximum relative tolerance for pressure change
C     RDRHOTOL    "                            " density change
C --- -------------------------------------------------------------------
      DIMENSION TIME(2)
      INTEGER KSTEP,KINC,LRESTART,NONUM,LFLAG,KINC_MAX,NUMCT,NUMCT_MAX
      INTEGER     I,NumElem,NumIt
      CHARACTER*80 FLNAME
C --- ABAQUS Message Vars -----------------------------------------------
      REAL*8 REALV
      INTEGER INTV
      DIMENSION REALV(3)
      CHARACTER*8 CHARV
C     REALV Real values to be inserted in place of the %R's in the
C           message
C     INTV  Integer values to be inserted in place of the %I's in the
C           message
C     CHARV String to be inserted in place of the %S's in the message
C --- -------------------------------------------------------------------
      CHARACTER*256 OUTDIR
      INTEGER LENOUTDIR
C --- Adjustable Parameters ---------------------------------------------
      DATA DTEMP/0.D0/,LRESTART/0/,LFLAG/1/
C     DTEMP       Change in fluid temperature (always zero)
C     LRESTART    Flag indicating that UEXTERNALDB is not to write
C                 restart information
C     LFLAG       Flag indicating that this is not a linear perturbation
C                 step
C --- -------------------------------------------------------------------
      COMMON/INC/TIME,DTIME,KSTEP,KINC
C
C --- Initialize the progress bar
      lret = DlgSet(gdlg, IDC_PROGRESS1, 0)
      CALL DLGFLUSH (gdlg, .TRUE.)
```

```
C --- Collect the adjustable parameters for the simulation
      CALL GETOUTDIR(OUTDIR,LENOUTDIR)
      OPEN(110,FILE=OUTDIR(1:LENOUTDIR)//'\PARAM.INP',RECL=300)
      READ(110,*) TEMPI
      READ(110,*) Vol_0
      READ(110,*) DTIME0
      READ(110,*) DTIME_MAX
      READ(110,*) DTIME_MIN
      READ(110,*) TIME_MAX
      READ(110,*) KINC_MAX
      READ(110,*) NUMCT_MAX
      READ(110,*) NumIt_max
      READ(110,*) RDPRESSTOL
      READ(110,*) RDRHOTOL
      READ(110,*) NumElem
      READ(110,*) NONUM
      READ(110,*) KSTEP
      CLOSE(110,STATUS='KEEP')
C --- Create and initialize the message file
      OPEN(110,FILE=OUTDIR(1:LENOUTDIR)//'\Test.msg',RECL=300)
      WRITE(110,*) '--Begin Run--'
      TIME(1)=0.D0
      TIME(2)=TIME(1)
      KINC=0
      PRESSI=0.D0
      TEMP=TEMPI
      PRESS=PRESSI
      CALL UEXTERNALDB(0,LRESTART,TIME,DTIME,KSTEP,KINC)
C --- Determine the initial density
      CALL UFLUID(RHO,CP,CT,PNEWDT,ENER,PRESS,DPRESS,
     1            PRESSI,TEMP,DTEMP,TEMPI,TIME,DTIME,KSTEP,KINC,NONUM,
     2            FLNAME,LFLAG)
      RHO0=RHO
C --- Perform the analysis
      DTIME=DTIME0
      KINC=1
      DO WHILE (TIME(1).LT.TIME_MAX .AND. KINC.LE.KINC_MAX)
            INTV=KINC
            REALV(1)=TIME(1)
            CALL STDB_ABQERR(2,'-Time Increment %I (%R s)',INTV,REALV,
     1            CHARV)
            CALL UEXTERNALDB(1,LRESTART,TIME,DTIME,KSTEP,KINC)
            NumIt=1
10          INTV=NumIt
            CALL STDB_ABQERR(2,'-Equilibrium Iteration %I-',INTV,REALV,
     1            CHARV)
            PNEWDT=1.D0
            RHO_old=RHO
            DO I=1,NumElem
                  CALL UFLUID(RHO,CP,CT,PNEWDT,ENER,PRESS,DPRESS,
     1                  PRESSI,TEMP,DTEMP,TEMPI,TIME,DTIME,KSTEP,KINC,NONUM,
     2                  FLNAME,LFLAG)
            END DO
            IF(PNEWDT.GE.1.D0)THEN
                  PRESS_old=PRESS
                  IF(PRESS_FEED .EQ. .TRUE.)THEN
C --- ------------------------------------------------------------------
```

```
C                      This is to artificially represent the pressure
C                      feedback from the finite element analysis
C
C                      NOTE:  This function is specific to the geometry and
C                             material properties in the input file for
C                             which the P-V data was generated.
C
C                      Input File:    Cyl_Actuator_Ref2.inp
C                            C_10:    89.
C                            C_01:    0.
C                             D_1:    9.23D-4
C
                       Vol=1.D18*RHO0/RHO*Vol_0
                       IF(Vol .LT. 1.D18*Vol_0)THEN
                           PRESS=2.9049070567319D-1*Vol-1.4251938917725D7
                       ELSE IF(Vol .LT. 5.05D7)THEN
                           PRESS=-1.0755067202566D13+8.618721316413D5*Vol-
     1                         2.1524545299135D-2*Vol**2.D0-
     2                         4.6040543463644D-12*Vol**3.D0+
     3                         8.7976057468028D-18*Vol**4.D0-
     3                         1.4059445836471D-25*Vol**5.D0+
     4                         7.0394006531021D-34*Vol**6.D0
                       ELSE
                           PRESS=8.5917157307109D-8*Vol**2.D0-
     1                         7.5547628547003D0*Vol+1.6341961374420D8
                       END IF
                       PRESS=1.D-6*PRESS
C --- -----------------------------------------------------------------
C ---                  Check for equilibrium
                       IF(DABS(PRESS-PRESS_old) .GT. RDPRESSTOL*DABS(PRESS)
     1                 .OR. DABS(RHO-RHO_old) .GT. RDRHOTOL*RHO)THEN
                           IF(NumIt .GE. NumIt_max)THEN
                               INTV=NumIt_max
                               CALL STDB_ABQERR(-3,'The maximum number of '//
     1                         'equilibrium iterations (%I) for this '//
     2                         'increment has been exceeded.',INTV,REALV,
     3                         CHARV)
                           END IF
                           NumIt=NumIt+1
                           GO TO 10
                       END IF
                   ELSE
                         PRESS=PRESSI
                   END IF
C ---              Proceed to the next time increment
                   NUMCT=0
                   KINC=KINC+1
                   TIME(1)=TIME(1)+DTIME
                   IF(TIME(1).GT.TIME_MAX)THEN
                         TIME(1)=TIME_MAX
                   END IF
                   TIME(2)=TIME(1)
C ---              Update the progress bar
                   lret = DlgSet(gdlg, IDC_PROGRESS1,
     1                   IDINT(TIME(1)/TIME_MAX*1.D2))
                   CALL DLGFLUSH (gdlg, .TRUE.)
                   CALL UEXTERNALDB(2,LRESTART,TIME,DTIME,KSTEP,KINC)
```

```
            ELSE
                  NUMCT=NUMCT+1
                  IF(NUMCT.GT.NUMCT_MAX)THEN
                        INTV=NUMCT_MAX
                        CALL STDB_ABQERR(-3,'The maximum number of '//
     1                        'cutbacks in time increment (%I) has been '//
     2                        'exceeded.',INTV,REALV,CHARV)
                  END IF
            END IF
C ---       Determine the next time step
            DTIME=PNEWDT*DTIME
            IF(DTIME.GT.DTIME_MAX)DTIME=DTIME_MAX
            IF(DTIME.LT.DTIME_MIN)THEN
                  REALV(1)=DTIME_MIN
                  CALL STDB_ABQERR(-3,'The value DTIME is less than the '//
     1                  'minimum (%R s).',INTV,REALV,CHARV)
            END IF
      END DO
      IF(KINC.GT.KINC_MAX)THEN
            INTV=KINC_MAX
            CALL STDB_ABQERR(-3,'The maximum number of time increments '//
     1            '(%I) has been exceeded.',INTV,REALV,CHARV)
      END IF
      CALL UEXTERNALDB(3,LRESTART,TIME,DTIME,KSTEP,KINC)
      WRITE(110,*) '--Run Completed--'
      CLOSE(110,STATUS='KEEP')
      RETURN
      END
C
C
C --- ------------------------------------------------------------------
C     Mimics the ABAQUS GETOUTDIR subroutine and returns the directory
C     where the input/output files should be located
C --- ------------------------------------------------------------------
      SUBROUTINE GETOUTDIR(OUTDIR,LENOUTDIR)
C
      INCLUDE 'ABA_PARAM.INC'
C
      CHARACTER*256 OUTDIR
      INTEGER LENOUTDIR
      INTEGER(4) istat
C
      istat=GETCWD(OUTDIR)
      OUTDIR=OUTDIR(1:SCAN(OUTDIR,'\',BACK=.TRUE.)-1)
      LENOUTDIR=LEN_TRIM(OUTDIR)
      RETURN
      END
C
C
C --- ------------------------------------------------------------------
C     Mimics the ABAQUS STDB_ABQERR subroutine to print messages to the
C     test program message file
C --- ------------------------------------------------------------------
      SUBROUTINE STDB_ABQERR(LOP,MSG,INTV,REALV,CHARV)
C
      INCLUDE 'ABA_PARAM.INC'
C
```

```
      REAL*8 REALV
      INTEGER INTV,I,J,Q,width,IDX_INT,IDX_REAL,IDX_CHAR
      CHARACTER*8 CHARV
      CHARACTER*(*) MSG
      DIMENSION INTV(*),REALV(*),CHARV(*)
C
      SELECT CASE (LOP)
            CASE (1)
                  WRITE(110,'(A)',ADVANCE='NO') ' ***NOTE: '
            CASE (-1)
                  WRITE(110,'(A)',ADVANCE='NO') ' ***WARNING: '
            CASE (-2, -3)
                  WRITE(110,'(A)',ADVANCE='NO') ' ***ERROR: '
      END SELECT
      I=1
      IDX_INT=1
      IDX_REAL=1
      IDX_CHAR=1
      J=SCAN(MSG,'%')
      DO WHILE (J.GT.0)
            WRITE(110,'(A)',ADVANCE='NO') MSG(I:J-1)
            SELECT CASE (MSG(J+1:J+1))
                  CASE ('I')
                      width=0
                      IF(INTV(IDX_INT).NE.0)width=FLOOR(LOG10(ABS(FLOAT(
     1                    INTV(IDX_INT)))))+1
                      IF(INTV(IDX_INT).LE.0)width=width+1
                      WRITE(110,'(I<width>)',ADVANCE='NO') INTV(IDX_INT)
                      IDX_INT=IDX_INT+1
                      I=J+2
                  CASE ('R')
                      width=11
                      IF(REALV(IDX_REAL).LT.0.D0)width=width+1
                      WRITE(110,'(ES<width>.5)',ADVANCE='NO') REALV(IDX_REAL)
                      IDX_REAL=IDX_REAL+1
                      I=J+2
                  CASE ('C')
                      WRITE(110,'(A)',ADVANCE='NO') TRIM(CHARV(IDX_CHAR))
                      IDX_CHAR=IDX_CHAR+1
                      I=J+2
                  CASE DEFAULT
                      I=J
            END SELECT
            IF(I.LT.LEN(MSG))THEN
                  Q=SCAN(MSG(J+1:LEN(MSG)),'%')
                  IF(Q.GT.0)THEN
                      J=J+Q
                  ELSE
                      J=0
                  END IF
            ELSE
                  J=0
            END IF
      END DO
      WRITE(110,'(A)',ADVANCE='YES') MSG(I:LEN(MSG))
      IF(LOP.EQ.-3)CALL XIT
      RETURN
      END
C
C
C --- ---------------------------------------------------------------------
C     Mimics the ABAQUS XIT subroutine to end a run
C --- ---------------------------------------------------------------------
      SUBROUTINE XIT()
C
      INCLUDE 'ABA_PARAM.INC'
C
      INTEGER KINC,KSTEP
      REAL*8 TIME,DTIME
      DIMENSION TIME(2)
      COMMON/INC/TIME,DTIME,KSTEP,KINC
C
      CALL UEXTERNALDB(3,0,TIME(1),DTIME,KSTEP,KINC)
      WRITE(110,*) '--Run Terminated--'
      CLOSE(110,STATUS='KEEP')
      STOP
      RETURN
      END
```

APPENDIX F

HYDROSTATIC FLUID ELEMENT GENERATION PROGRAM SOURCE CODE

The hydrostatic fluid element generation program is written in Microsoft Visual Basic 6.0 and generates the definitions for the hydrostatic fluid elements representing the inclusion fluid and adds them to the ABAQUS input file (Section 4.4.1). It features a Windows user interface where the user can select the ABAQUS input file containing the element definitions for the matrix material and name the ABAQUS input file that will contain the hydrostatic fluid element definitions. The user can select between spherical and cylindrical inclusions, enter the position and size of the inclusion and specify the number and pitch of inclusions in the mosaic. The user can also change the tolerance for how close a node must be to the calculated inclusion surface to be considered on the surface. When the user clicks the **Create Output** button, the program reads the ABAQUS inputs file, locates the node definitions for each part in the assembly uses determines which nodes are on the inclusion surface. The program then locates the element definitions for each part and uses the ABAQUS face definitions (Figure F.1) to determine which element faces lie on the inclusion surface. The node information for that face is then used to generate the definition for a hydrostatic fluid element so that its positive normal points into the fluid (Figure F.2). The positive normal direction is given by the right-hand rule going around the nodes of the element in the order that they are given on the element data line.

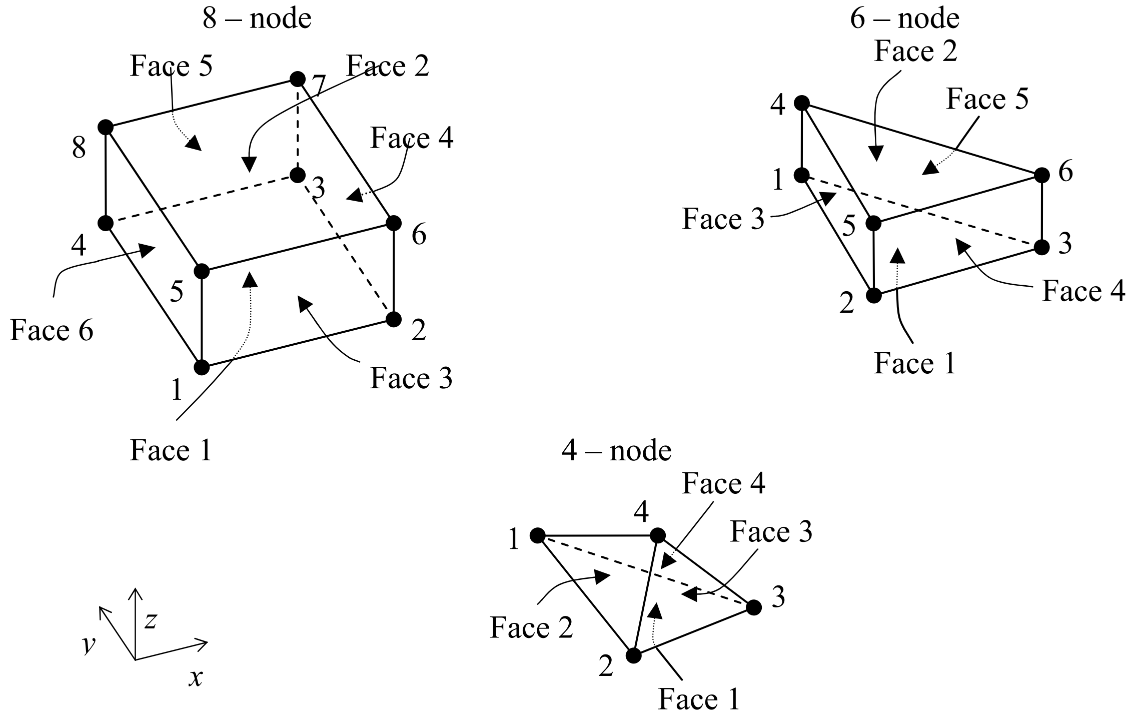


Figure F.1: ABAQUS face and node numbering for 8-, 6-, and 4-node continuum elements.

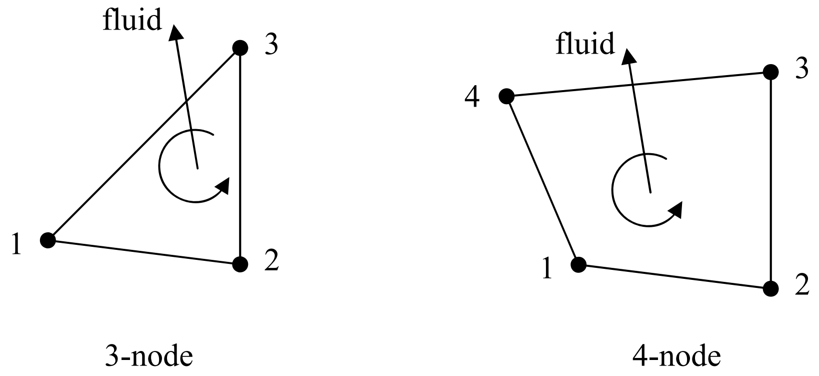


Figure F.2: Positive normal and node numbering for 3- and 4-node 3D hydrostatic fluid elements.

This process is repeated for each inclusion in the mosaic. A copy of the ABAQUS input file is created and the hydrostatic fluid element definitions are written in the appropriate location(s) within the file. The program will work for a spherical inclusion whose center coordinates and radius are entered into the program or a cylindrical inclusion whose axis is parallel to one of the axes the assembly coordinate system by specifying the coordinate on the axis at the top of the inclusion along with its radius and height. Hydrostatic fluid elements can be generated for arrays of inclusions of either shape by specifying their pitch in the *x*-, *y*-, and *z*-directions of the assembly coordinate system as well as the number in each direction. The program can handle models that have multiple parts or instances that are meshed in either the part or instance. The parts can be translated in the assembly.

```
VERSION 5.00
Object = "{F9043C88-F6F2-101A-A3C9-08002B2F49FB}#1.2#0"; "COMDLG32.OCX"
Begin VB.Form frmMain
   Caption         =   "Hydrostatic Fluid Element Generator"
   ClientHeight    =   6750
   ClientLeft      =   60
   ClientTop       =   510
   ClientWidth     =   8610
   Icon            =   "frmMain.frx":0000
   LinkTopic       =   "Form1"
   MaxButton       =   0   'False
   ScaleHeight     =   6750
   ScaleWidth      =   8610
   StartUpPosition =   3  'Windows Default
   Begin VB.Frame fraVesGeom
      Caption         =   "Vesicle Geometry"
      Height          =   3135
      Left            =   120
      TabIndex        =   9
      Top             =   1920
      Width           =   8295
      Begin VB.TextBox txtTol
         Height          =   325
         Left            =   4200
         TabIndex        =   41
         Text            =   "txtTol"
         Top             =   2640
         Width           =   1935
      End
      Begin VB.Frame fraVesicleArray
         Caption         =   "Vesicle Array"
         Height          =   1215
         Left            =   2880
         TabIndex        =   24
         Top             =   360
         Width           =   5295
         Begin VB.TextBox txtNz
            Height          =   325
            Left            =   4080
            TabIndex        =   36
            Text            =   "txtNz"
            Top             =   720
            Width           =   975
         End
         Begin VB.TextBox txtPz
            Height          =   325
            Left            =   4080
            TabIndex        =   34
            Text            =   "txtPz"
            Top             =   360
            Width           =   975
         End
         Begin VB.TextBox txtNy
            Height          =   325
            Left            =   2400
            TabIndex        =   32
            Text            =   "txtNy"
            Top             =   720
            Width           =   975
         End
         Begin VB.TextBox txtPy
            Height          =   325
            Left            =   2400
            TabIndex        =   30
            Text            =   "txtPy"
            Top             =   360
            Width           =   975
         End
         Begin VB.TextBox txtNx
            Height          =   325
            Left            =   720
            TabIndex        =   28
            Text            =   "txtNx"
            Top             =   720
            Width           =   975
         End
         Begin VB.TextBox txtPx
            Height          =   325
            Left            =   720
            TabIndex        =   26
            Text            =   "txtPx"
            Top             =   360
            Width           =   975
         End
         Begin VB.Label lblNz
            Caption         =   "No. z"
            Height          =   255
            Left            =   3480
            TabIndex        =   35
            Top             =   765
            Width           =   495
         End
         Begin VB.Label lblPz
            Caption         =   "Pitch z"
            Height          =   255
            Left            =   3480
            TabIndex        =   33
            Top             =   405
            Width           =   615
         End
         Begin VB.Label lblNy
            Caption         =   "No. y"
            Height          =   255
```

```
         Left            =   1800
         TabIndex        =   31
         Top             =   765
         Width           =   495
      End
      Begin VB.Label lblPy
         Caption         =   "Pitch y"
         Height          =   255
         Left            =   1800
         TabIndex        =   29
         Top             =   405
         Width           =   615
      End
      Begin VB.Label lblNx
         Caption         =   "No. x"
         Height          =   255
         Left            =   120
         TabIndex        =   27
         Top             =   760
         Width           =   495
      End
      Begin VB.Label lblPx
         Caption         =   "Pitch x"
         Height          =   255
         Left            =   120
         TabIndex        =   25
         Top             =   400
         Width           =   615
      End
   End
   Begin VB.Frame fraVesicleCenter
      Caption         =   "Vesicle Center"
      Height          =   1575
      Left            =   120
      TabIndex        =   15
      Top             =   1320
      Width           =   2655
      Begin VB.OptionButton optZAxisDir
         Height          =   255
         Left            =   1920
         TabIndex        =   40
         Top             =   1200
         Width           =   255
      End
      Begin VB.OptionButton optYAxisDir
         Height          =   255
         Left            =   1920
         TabIndex        =   39
         Top             =   840
         Value           =   -1  'True
         Width           =   375
      End
      Begin VB.OptionButton optXAxisDir
         Height          =   255
         Left            =   1920
         TabIndex        =   38
         Top             =   480
         Width           =   375
      End
      Begin VB.TextBox txtVz
         Height          =   325
         Left            =   435
         TabIndex        =   18
         Text            =   "txtVz"
         Top             =   1155
         Width           =   1095
      End
      Begin VB.TextBox txtVy
         Height          =   325
         Left            =   435
         TabIndex        =   17
         Text            =   "txtVy"
         Top             =   795
         Width           =   1095
      End
      Begin VB.TextBox txtVx
         Height          =   325
         Left            =   435
         TabIndex        =   16
         Text            =   "txtVx"
         Top             =   435
         Width           =   1095
      End
      Begin VB.Label lblAxisDir
         Caption         =   "Axis Dir"
         Height          =   255
         Left            =   1800
         TabIndex        =   37
         Top             =   240
         Width           =   615
      End
      Begin VB.Label lblVz
         Caption         =   "z"
         Height          =   255
         Left            =   240
         TabIndex        =   21
         Top             =   1200
         Width           =   135
      End
      Begin VB.Label lblVy
         Caption         =   "y"
         Height          =   255
         Left            =   240
         TabIndex        =   20
         Top             =   840
         Width           =   135
      End
      Begin VB.Label lblVx
         Caption         =   "x"
         Height          =   255
         Left            =   240
         TabIndex        =   19
         Top             =   480
         Width           =   135
      End
   End
   Begin VB.TextBox txtVesicleRadius
      Height          =   325
      Left            =   4200
      TabIndex        =   14
      Text            =   "txtVesicleRadius"
      Top             =   1680
      Width           =   1935
   End
   Begin VB.Frame fraVesType
      Caption         =   "Type"
      Height          =   735
      Left            =   120
      TabIndex        =   11
      Top             =   360
      Width           =   2655
      Begin VB.OptionButton optSpherical
         Caption         =   "Spherical"
         Height          =   255
         Left            =   120
         TabIndex        =   13
         Top             =   360
         Value           =   -1  'True
         Width           =   1095
      End
      Begin VB.OptionButton optCylindrical
         Caption         =   "Cylindrical"
         Height          =   255
         Left            =   1320
         TabIndex        =   12
         Top             =   360
         Width           =   1215
      End
   End
   Begin VB.TextBox txtCylDepth
      Height          =   325
      Left            =   4200
      TabIndex        =   10
      Text            =   "txtCylDepth"
      Top             =   2160
      Width           =   1935
   End
   Begin VB.Label lblTol
      Caption         =   "Tolerance"
      Height          =   255
      Left            =   3360
      TabIndex        =   42
      Top             =   2685
      Width           =   855
   End
   Begin VB.Label lblVesicleRadius
      Caption         =   "Vesicle Radius"
      Height          =   255
      Left            =   3000
      TabIndex        =   23
      Top             =   1725
```

```
         Width           =   1095
      End
      Begin VB.Label lblCylDepth
         Caption         =   "Cylinder Depth"
         Height          =   255
         Left            =   3000
         TabIndex        =   22
         Top             =   2205
         Width           =   1095
      End
   End
   Begin MSComDlg.CommonDialog CommonDialog
      Left            =   7920
      Top             =   6120
      _ExtentX        =   847
      _ExtentY        =   847
      _Version        =   393216
      CancelError     =   -1  'True
   End
   Begin VB.CommandButton cmdExit
      Caption         =   "Exit"
      Height          =   495
      Left            =   1920
      TabIndex        =   8
      Top             =   6000
      Width           =   1455
   End
   Begin VB.CommandButton cmdCreateOutput
      Caption         =   "Create Output"
      Height          =   495
      Left            =   120
      TabIndex        =   7
      Top             =   6000
      Width           =   1455
   End
   Begin VB.CommandButton cmdOutputBrowse
      Caption         =   "Browse..."
      Height          =   375
      Left            =   7080
      TabIndex        =   6
      Top             =   5280
      Width           =   1335
   End
   Begin VB.TextBox txtOutputFile
      Height          =   325
      Left            =   960
      TabIndex        =   5
      Text            =   "txtOutputFile"
      Top             =   5280
      Width           =   6015
   End
   Begin VB.CommandButton cmdInputBrowse
      Caption         =   "Browse..."
      Height          =   375
      Left            =   7200
      TabIndex        =   2
      Top             =   1320
      Width           =   1335
   End
   Begin VB.TextBox txtInputFile
      Height          =   325
      Left            =   960
      TabIndex        =   1
      Text            =   "txtInputFile"
      Top             =   1320
      Width           =   6135
   End
   Begin VB.Label lblOutputFile
      Caption         =   "Output File"
      Height          =   255
      Left            =   120
      TabIndex        =   4
      Top             =   5325
      Width           =   975
   End
   Begin VB.Label lblDescription
      Caption         =   $"frmMain.frx":0442
      Height          =   975
      Left            =   120
      TabIndex        =   3
      Top             =   120
      Width           =   8295
      WordWrap        =   -1  'True
   End
   Begin VB.Label lblInputFile
      Caption         =   "Input File"
      Height          =   255
      Left            =   240
      TabIndex        =   0
      Top             =   1365
      Width           =   735
   End
End
Attribute VB_Name = "frmMain"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
'-----------------------------------------------------------------------
'|  TITLE:      Hydrostatic Fluid Element Generator                    |
'|                                                                     |
'|  WRITTEN BY: Chris Homison                                          |
'|                                                                     |
'|  REVISIONS:  02/04/2005  1.0.0   Initial Release                    |
'|              03/12/2005  1.0.1   Fixed bug where last item in a data |
'|                                  line was not split                 |
'|              03/20/2005  1.1.0   Added multi-part/assembly support, |
'|                                  support for cylindrical vesicles,  |
'|                                  and support for mosaics            |
'|              08/25/2005  1.2.0   Added report creation              |
'|              08/26/2005  1.2.1   Minor bug and comment fixes        |
'|              11/14/2005  1.2.2   Fixed bug where report info for a  |
'|                                  vesicle was written for each part, |
'|                                  added revision history             |
'|             11/15/2005  1.3.0   Added support for tetrahedral       |
'|                                 elements                            |
'|             12/15/2005  1.4.0   Added support for triangular prism  |
'|                                 elements, shell continuum elements, |
'|                                 and multiple element types for a    |
'|                                 single part                         |
'-----------------------------------------------------------------------

Option Explicit

Const Tol As Double = 0.00001       ' Default tolerance for how close to _
                                      the calculated vesicle boundary a _
                                      node must be in order to be _
                                      considered on the boundary
Const MaxFaceNodes As Integer = 4   ' Maximum number of nodes an element _
                                      face (or hydrostatic fluid element) _
                                      can have

Dim FSO As New FileSystemObject     ' File system object for reading text _
                                      files
Dim EIFile As File                  ' File that is being read/written
Dim EIStream As TextStream          ' Text stream that is being _
                                      read/written to/from the file
Dim RepFile As File                 ' File where report info is being written
Dim RepStream As TextStream         ' Report text stream
Dim OutFile As File                 ' Output File
Dim OutStream As TextStream         ' Output file text stream
Dim strIncType As String            ' Type of inclusion
Dim strAxisDir As String            ' Direction of vesicle axis for _
                                      cylindrical inclusions

' Part information type
Private Type PartInfo
    Name As String                  ' Part name
    XOffset As Double               ' X offset in the assembly
    YOffset As Double               ' Y offset in the assembly
    ZOffset As Double               ' Z offset in the assembly
    PartMesh As Boolean             ' Flag for if the part is meshed as a _
                                      part
End Type

' Node information type
Private Type NodeInfo
    OnBoundary As Boolean           ' Records if the node is on the vesicle _
                                      boundary
    TimesUsed As Integer            ' The number of hydrostatic fluid _
                                      elements the node belongs to
End Type

' Hydrostatic Fluid Element Type
Private Type HydElemInfo
    NumNodes As Integer             ' Number of nodes in the element
    Nodes(MaxFaceNodes) As Long     ' Node numbers for the element
End Type

Dim Parts() As PartInfo             ' Part information array
Dim Nodes() As NodeInfo             ' Node information array
Dim Faces() As Integer              ' Node numbers for the continuum _
                                      element faces
Dim HydElems() As HydElemInfo       ' Hydrostatic fluid element definitions

Private Sub Form_Load()
    txtInputFile.Text = ""
    txtVesicleRadius.Text = ""
    txtOutputFile.Text = ""
    optSpherical_Click
    txtVx.Text = "0"
    txtVy.Text = "0"
    txtVz.Text = "0"
    txtCylDepth.Text = ""
    txtPx.Text = "0"
    txtNx.Text = "1"
    txtPy.Text = "0"
    txtNy.Text = "1"
    txtPz.Text = "0"
    txtNz.Text = "1"
    txtTol.Text = CStr(Tol)
End Sub

Private Sub cmdInputBrowse_Click()
    On Error Resume Next
    CommonDialog.Filter = "ABAQUS Input File (*.inp)|*.inp"
    CommonDialog.ShowOpen
    If CommonDialog.FileName = "" Or Err.Number = 32755 Then
        Exit Sub
    End If
    txtInputFile.Text = CommonDialog.FileName
End Sub

Private Sub optSpherical_Click()
    If optSpherical.Value = True Then
        strIncType = "Spherical"
        fraVesicleCenter.Caption = "Vesicle Center"
        txtCylDepth.Enabled = False
        lblCylDepth.Enabled = False
        lblPz.Enabled = True
        lblNz.Enabled = True
        txtPz.Enabled = True
        txtNz.Enabled = True
        lblAxisDir.Visible = False
        optXAxisDir.Visible = False
        optYAxisDir.Visible = False
        optZAxisDir.Visible = False
        optXAxisDir.Value = True
        optYAxisDir.Value = True
        optZAxisDir.Value = True
        optXAxisDir_Click
      Else
        strIncType = "Cylindrical"
        fraVesicleCenter.Caption = "Vesicle Axis (Top)"
        txtCylDepth.Enabled = True
        lblCylDepth.Enabled = True
        lblPz.Enabled = False
        lblNz.Enabled = False
        txtPz.Enabled = False
        txtNz.Enabled = False
        txtPz.Text = "0"
        txtNz.Text = "1"
        lblAxisDir.Visible = True
        optXAxisDir.Visible = True
        optYAxisDir.Visible = True
        optZAxisDir.Visible = True
        optXAxisDir.Value = False
        optYAxisDir.Value = True
        optZAxisDir.Value = False
        optYAxisDir_Click
    End If
End Sub

Private Sub optCylindrical_Click()
    optSpherical_Click
End Sub

Private Sub optXAxisDir_Click()
    If optXAxisDir.Value = True Then
        strAxisDir = "x"
        lblPx.Enabled = False
        lblNx.Enabled = False
        txtPx.Enabled = False
        txtNx.Enabled = False
        txtPx.Text = "0"
        txtNx.Text = "1"
        optZAxisDir_Click
        optYAxisDir_Click
      Else
        lblPx.Enabled = True
        lblNx.Enabled = True
        txtPx.Enabled = True
        txtNx.Enabled = True
    End If
End Sub

Private Sub optYAxisDir_Click()
    If optYAxisDir.Value = True Then
        strAxisDir = "y"
        lblPy.Enabled = False
        lblNy.Enabled = False
        txtPy.Enabled = False
        txtNy.Enabled = False
        txtPy.Text = "0"
        txtNy.Text = "1"
        optXAxisDir_Click
        optZAxisDir_Click
      Else
        lblPy.Enabled = True
        lblNy.Enabled = True
        txtPy.Enabled = True
        txtNy.Enabled = True
    End If
End Sub
```

```
Private Sub optZAxisDir_Click()
    If optZAxisDir.Value = True Then
        strAxisDir = "z"
        lblPz.Enabled = False
        lblNz.Enabled = False
        txtPz.Enabled = False
        txtNz.Enabled = False
        txtPz.Text = "0"
        txtNz.Text = "1"
        optXAxisDir_Click
        optYAxisDir_Click
      Else
        lblPz.Enabled = True
        lblNz.Enabled = True
        txtPz.Enabled = True
        txtNz.Enabled = True
    End If
End Sub

Private Sub cmdOutputBrowse_Click()
    Dim MsgResult As VbMsgBoxResult

    On Error Resume Next
    CommonDialog.Filter = "ABAQUS Input File (*.inp)|*.inp"
    CommonDialog.FileName = ""
RetrySave:
    CommonDialog.ShowSave
    If CommonDialog.FileName = "" Or Err.Number = 32755 Then
        Exit Sub
    End If
    txtOutputFile.Text = CommonDialog.FileName
    If FSO.FileExists(txtOutputFile.Text) = True Then
        MsgResult = MsgBox("File already exists. Overwrite?", vbYesNoCancel _
            + vbExclamation, Me.Caption)
        If MsgResult = vbNo Then
            GoTo RetrySave
            Exit Sub
        End If
        If MsgResult = vbCancel Then
            Exit Sub
        End If
    End If
End Sub

Private Sub cmdCreateOutput_Click()
    Dim lngN As Long
    Dim lngNumOB As Long
    Dim lngM As Long
    Dim lngInsLine As Long
    Dim lngNumElem(MaxFaceNodes) As Long
    Dim lngTotalNumEl As Long
    Dim intPart As Integer
    Dim intVx As Integer
    Dim intVy As Integer
    Dim intVz As Integer
    Dim intFace As Integer
    Dim intNode As Integer
    Dim intNumFaceNodes() As Integer
    Dim intHydElType As Integer
    Dim Radius As Double
    Dim VRad As Double
    Dim Vx As Double
    Dim Vy As Double
    Dim Vz As Double
    Dim Vh As Double
    Dim Tolerance As Double
    Dim ReadStr As String
    Dim DataLine() As String
    Dim PartName As String
    Dim strHydNodeDef As String
    Dim FluidFace As Boolean

    lngTotalNumEl = 0
    ' Create the report file
    FSO.CreateTextFile Left(txtOutputFile.Text, Len(txtOutputFile.Text) - _
        4) & " Report.txt"
    Set RepFile = FSO.GetFile(Left(txtOutputFile.Text, _
        Len(txtOutputFile.Text) - 4) & " Report.txt")
    Set RepStream = RepFile.OpenAsTextStream(ForWriting)
    RepStream.WriteLine "Hydrostatic Fluid Element Generator Version " & _
        App.Major & "." & App.Minor & "." & App.Revision
    RepStream.WriteLine
    RepStream.WriteLine "-------Input--------"
    RepStream.WriteLine "Input File:         " & txtInputFile.Text
    RepStream.WriteLine "Output File:        " & txtOutputFile.Text
    RepStream.WriteLine "Inclusion Type:     " & strIncType
    RepStream.WriteLine "Mosaic Size:        " & txtNx.Text & " x " & _
        txtNy.Text & " x " & txtNz.Text
    RepStream.WriteLine "Mosaic Pitch:       " & txtPx.Text & " x " & _
        txtPy.Text & " x " & txtPz.Text
    RepStream.WriteLine "Vesicle Radius:     " & txtVesicleRadius.Text
    If optSpherical.Value = True Then
        RepStream.WriteLine "Vesicle Center:     (" & txtVx.Text & ", " & _
            txtVy.Text & ", " & txtVz.Text & ")"
      Else
        RepStream.WriteLine "Vesicle Axis (Top): (" & txtVx.Text & ", " & _
            txtVy.Text & ", " & txtVz.Text & ")"
        RepStream.WriteLine "Axis Direction:     " & strAxisDir
        RepStream.WriteLine "Cylinder Depth:     " & txtCylDepth.Text
    End If
    RepStream.WriteLine "Tolerance:          " & txtTol.Text
    RepStream.WriteLine
    RepStream.WriteLine "-------Output-------"
    ' Open the Input File
    If FSO.FileExists(txtInputFile.Text) = False Then
        MsgBox "Input file does not exist.", vbCritical, frmMain.Caption
        Exit Sub
    End If
    Set EIFile = FSO.GetFile(txtInputFile.Text)
    Set EIStream = EIFile.OpenAsTextStream(1)
    ReadStr = EIStream.ReadLine
    ' ------------------------
    ' Collect part information
    ' ------------------------
    ' Find out which parts are meshed as parts and record their offset
    lngN = 0
    While LCase(ReadStr) <> "*end assembly"
        If LCase(Left(ReadStr, 9)) = "*instance" Then
            lngN = lngN + 1
            ReDim Preserve Parts(lngN)
            ' Get the part name
            DataLine = Split(ReadStr & ",", ",")
            lngM = 0
            While LCase(Left(LTrim(DataLine(lngM)), 4)) <> "part"
                lngM = lngM + 1
                If lngM = UBound(DataLine) + 1 Then
                    MsgBox "An error occurred while gathering part instance _
                        information: Part name not found.", vbCritical, _
                        Me.Caption
                    Exit Sub
                End If
            Wend
            Parts(lngN - 1).Name = Right(DataLine(lngM), _
                Len(DataLine(lngM)) - InStr(DataLine(lngM), "="))
            ReadStr = EIStream.ReadLine
            ' If there is offset data for the instance, then it is meshed _
              as a part
            If Left(ReadStr, 1) <> "*" Then
                DataLine = Split(ReadStr & ",", ",")
                Parts(lngN - 1).PartMesh = True
                Parts(lngN - 1).XOffset = CDbl(DataLine(0))
                Parts(lngN - 1).YOffset = CDbl(DataLine(1))
                Parts(lngN - 1).ZOffset = CDbl(DataLine(2))
              Else
                If LCase(ReadStr) = "*end instance" Then
                    Parts(lngN - 1).PartMesh = True
                  Else
                    Parts(lngN - 1).PartMesh = False
                End If
                Parts(lngN - 1).XOffset = 0
                Parts(lngN - 1).YOffset = 0
                Parts(lngN - 1).ZOffset = 0
            End If
        End If
        If EIStream.AtEndOfStream Then
            MsgBox "An error occurred while gathering part information: _
                Could not find end of assembly", vbCritical, Me.Caption
            Exit Sub
        End If
        ReadStr = EIStream.ReadLine
    Wend
    EIStream.Close
    ' Create the output file
    FSO.CreateTextFile txtOutputFile.Text
    Set OutFile = FSO.GetFile(txtOutputFile.Text)
    ' Loops for vesicle mosaics
    For intVx = 1 To CInt(txtNx.Text)
        For intVy = 1 To CInt(txtNy.Text)
            For intVz = 1 To CInt(txtNz.Text)
                ' ------------------------------------------------------------
                ' Create the hydrostatic fluid elements for each _
```

```
                  part/instance
                ' ------------------------------------------------------------
                For intPart = 0 To UBound(Parts) - 1
                    ' If element data has already been written, read the _
                      scratch file
                    If intPart > 0 Or intVx > 1 Or intVy > 1 Or intVz > 1 _
                      Then
                        Set EIFile = FSO.GetFile(Left(txtOutputFile.Text, _
                            Len(txtOutputFile.Text) - 4) & "~1.inp")
                    End If
                    ' Re-open the input/scratch file
                    Set EIStream = EIFile.OpenAsTextStream(1)
                    ReadStr = EIStream.ReadLine
                    ' Find the part
NextLine:
                    If Parts(intPart).PartMesh = True Then
                        If LCase(Left(ReadStr, 5)) = "*part" Then
                            DataLine = Split(ReadStr & ",", ",")
                            lngM = 0
                            While LCase(Left(LTrim(DataLine(lngM)), 4)) <> _
                              "name"
                                lngM = lngM + 1
                                If lngM = UBound(DataLine) + 1 Then
                                    MsgBox "An error occurred while _
                                        gathering part information: Part _
                                        name not found: " & _
                                        Parts(intPart).Name, vbCritical, _
                                        Me.Caption
                                    Exit Sub
                                End If
                            Wend
                            If LCase(Parts(intPart).Name) = _
                              LCase(Right(DataLine(lngM), _
                              Len(DataLine(lngM)) - InStr(DataLine(lngM), _
                              "="))) Then
                                GoTo FoundPart
                            End If
                        End If
                      Else
                        If LCase(Left(ReadStr, 9)) = "*instance" Then
                            DataLine = Split(ReadStr & ",", ",")
                            lngM = 0
                            While LCase(Left(LTrim(DataLine(lngM)), 4)) <> _
                              "part"
                                lngM = lngM + 1
                                If lngM = UBound(DataLine) + 1 Then
                                    MsgBox "An error occurred while _
                                        gathering part information: Part _
                                        name not found: " & _
                                        Parts(intPart).Name, vbCritical, _
                                        Me.Caption
                                    Exit Sub
                                End If
                            Wend
                            If LCase(Parts(intPart).Name) = _
                              LCase(Right(DataLine(lngM), _
                              Len(DataLine(lngM)) - InStr(DataLine(lngM), _
                              "="))) Then
                                GoTo FoundPart
                            End If
                        End If
                    End If
                    If EIStream.AtEndOfStream Then
                        MsgBox "An error occurred while gathering node _
                            information: Part not found: " & _
                            Parts(intPart).Name, vbCritical, Me.Caption
                        Exit Sub
                    End If
                    ReadStr = EIStream.ReadLine
                    GoTo NextLine
FoundPart:
                    ReadStr = EIStream.ReadLine
                    ' Find the *Node section for the part
                    ' NOTE: This code only reads a single *Node section per _
                            part.
                    While LCase(Left(ReadStr, 5)) <> "*node"
                        ReadStr = EIStream.ReadLine
                    Wend
                    If EIStream.AtEndOfStream Then
                        MsgBox "An error occurred while gathering node _
                            information: Node definition not found for _
                            part " & Parts(intPart).Name, vbCritical, _
                            Me.Caption
                        Exit Sub
                    End If
                    ' ---------------------------------------------------
                    ' Determine which nodes are on the vesicle surface
                    ' ---------------------------------------------------
                    lngN = 0
                    ReadStr = EIStream.ReadLine
                    VRad = CDbl(txtVesicleRadius.Text)
                    Vx = CDbl(txtVx.Text) + (intVx - 1) * CDbl(txtPx.Text)
                    Vy = CDbl(txtVy.Text) + (intVy - 1) * CDbl(txtPy.Text)
                    Vz = CDbl(txtVz.Text) + (intVz - 1) * CDbl(txtPz.Text)
                    Tolerance = CDbl(txtTol.Text)
                    lngNumOB = 0
                    While Left(ReadStr, 1) <> "*"
                        lngN = lngN + 1
                        ReDim Preserve Nodes(lngN + 1)
                        Nodes(lngN).TimesUsed = 0
                        DataLine = Split(ReadStr & ",", ",")
                        If optSpherical.Value = True Then
                            ' Spherical Vesicle
                            Radius = Sqr((CDbl(DataLine(1)) + _
                                Parts(intPart).XOffset - Vx) ^ 2 + _
                                (CDbl(DataLine(2)) + Parts(intPart).YOffset _
                                - Vy) ^ 2 + (CDbl(DataLine(3)) + _
                                Parts(intPart).ZOffset - Vz) ^ 2)
                            If Abs(Radius - VRad) <= Tolerance Then
                                lngNumOB = lngNumOB + 1
                                Nodes(lngN).OnBoundary = True
                              Else
                                Nodes(lngN).OnBoundary = False
                            End If
```

```
Else
  ' Cylindrical Vesicle
  Vh = CDbl(txtCylDepth.Text)
  If optXAxisDir.Value = True Then
      Radius = Sqr((CDbl(DataLine(3)) + _
          Parts(intPart).ZOffset - Vz) ^ 2 + _
          (CDbl(DataLine(2)) + _
          Parts(intPart).YOffset - Vy) ^ 2)
      If Radius - VRad <= Tolerance And (Sgn(Vx - _
        (DataLine(1) + Parts(intPart).XOffset)) = _
        1 Or Abs(Vx - (DataLine(1) + _
        Parts(intPart).XOffset)) <= Tolerance) _
        And (Sgn(Vx - Vh - (DataLine(1) + _
        Parts(intPart).XOffset)) = -1 Or Abs(Vx - _
        Vh - (DataLine(1) + _
        Parts(intPart).XOffset)) <= Tolerance) Then
          lngNumOB = lngNumOB + 1
          Nodes(lngN).OnBoundary = True
        Else
          Nodes(lngN).OnBoundary = False
      End If
  End If
  If optYAxisDir.Value = True Then
      Radius = Sqr((CDbl(DataLine(1)) + _
          Parts(intPart).XOffset - Vx) ^ 2 + _
          (CDbl(DataLine(3)) + _
          Parts(intPart).ZOffset - Vz) ^ 2)
      If Radius - VRad <= Tolerance And (Sgn(Vy - _
        (DataLine(2) + Parts(intPart).YOffset)) = _
        1 Or Abs(Vy - (DataLine(2) + _
        Parts(intPart).YOffset)) <= Tolerance) _
        And (Sgn(Vy - Vh - (DataLine(2) + _
        Parts(intPart).YOffset)) = -1 Or Abs(Vy - _
        Vh - (DataLine(2) + _
        Parts(intPart).YOffset)) <= Tolerance) Then
          lngNumOB = lngNumOB + 1
          Nodes(lngN).OnBoundary = True
        Else
          Nodes(lngN).OnBoundary = False
      End If
  End If
  If optZAxisDir.Value = True Then
      Radius = Sqr((CDbl(DataLine(1)) + _
          Parts(intPart).XOffset - Vx) ^ 2 + _
          (CDbl(DataLine(2)) + _
          Parts(intPart).YOffset - Vy) ^ 2)
      If Radius - VRad <= Tolerance And (Sgn(Vz - _
        (DataLine(3) + Parts(intPart).ZOffset)) = _
        1 Or Abs(Vz - (DataLine(3) + _
        Parts(intPart).ZOffset)) <= Tolerance) _
        And (Sgn(Vz - Vh - (DataLine(3) + _
        Parts(intPart).ZOffset)) = -1 Or Abs(Vz - _
        Vh - (DataLine(3) + _
        Parts(intPart).ZOffset)) <= Tolerance) Then
          lngNumOB = lngNumOB + 1
          Nodes(lngN).OnBoundary = True
        Else
```

```
                                Nodes(lngN).OnBoundary = False
                            End If
                        End If
                    End If
                    ReadStr = EIStream.ReadLine
                Wend
                If EIStream.AtEndOfStream Then
                    MsgBox "An error occurred while gathering node _
                        information: End of node definition not found _
                        for part " & Parts(intPart).Name, vbCritical, _
                        Me.Caption
                    Exit Sub
                End If
                ' Find the *Element section
                While LCase(Left(ReadStr, 8)) <> "*element"
                    ReadStr = EIStream.ReadLine
                Wend
                If EIStream.AtEndOfStream Then
                    MsgBox "An error occurred while gathering element _
                        information: Element definition not found for _
                        part " & Parts(intPart).Name, vbCritical, _
                        Me.Caption
                    Exit Sub
                End If
                '-----------------------------
                ' Determine the element type
                '-----------------------------
                ' NOTE: See 14.1.4-18,19 of ABAQUS Analysis User's _
                        Manual Vol. IV: Elements for node and face _
                        definitions for the continuum elements.
                ' NOTE: The part can have multiple *Element sections, _
                        but they must be together in the input file.
                ReDim HydElems(0)
                lngN = 0
MoreElements:   DataLine = Split(ReadStr & ",", ",")
                lngM = 0
                While LCase(Left(LTrim(DataLine(lngM)), 4)) <> "type"
                    lngM = lngM + 1
                    If lngM = UBound(DataLine) + 1 Then
                        MsgBox "An error occurred while gathering _
                            element information: Element type not _
                            found: " & Parts(intPart).Name, vbCritical, _
                            Me.Caption
                        Exit Sub
                    End If
                Wend
                Select Case LCase(Left(Right(DataLine(lngM), _
                  Len(DataLine(lngM)) - InStr(DataLine(lngM), "=")), 4))
                    ' 8 - node 3D shell continuum elements
                    Case "sc8r"
                        GoTo EightNode
                    ' 8 - node (hexahedral) 3D continuum elements
                    Case "c3d8"
EightNode:              ReDim intNumFaceNodes(6)
                        intNumFaceNodes(0) = 4
                        intNumFaceNodes(1) = 4
                        intNumFaceNodes(2) = 4
```

```
                            intNumFaceNodes(3) = 4
                            intNumFaceNodes(4) = 4
                            intNumFaceNodes(5) = 4
                            ReDim Faces(6, 4)
                            ' Face 1
                            Faces(0, 0) = 1
                            Faces(0, 1) = 2
                            Faces(0, 2) = 3
                            Faces(0, 3) = 4
                            ' Face 2
                            Faces(1, 0) = 5
                            Faces(1, 1) = 8
                            Faces(1, 2) = 7
                            Faces(1, 3) = 6
                            ' Face 3
                            Faces(2, 0) = 1
                            Faces(2, 1) = 5
                            Faces(2, 2) = 6
                            Faces(2, 3) = 2
                            ' Face 4
                            Faces(3, 0) = 2
                            Faces(3, 1) = 6
                            Faces(3, 2) = 7
                            Faces(3, 3) = 3
                            ' Face 5
                            Faces(4, 0) = 3
                            Faces(4, 1) = 7
                            Faces(4, 2) = 8
                            Faces(4, 3) = 4
                            ' Face 6
                            Faces(5, 0) = 4
                            Faces(5, 1) = 8
                            Faces(5, 2) = 5
                            Faces(5, 3) = 1
                        ' 6 - node 3D shell continuum elements
                        Case "sc6r"
                            GoTo SixNode
                        ' 6 - node (triangular prism) 3D continuum elements
                        Case "c3d6"
SixNode:                    ReDim intNumFaceNodes(5)
                            intNumFaceNodes(0) = 3
                            intNumFaceNodes(1) = 3
                            intNumFaceNodes(2) = 4
                            intNumFaceNodes(3) = 4
                            intNumFaceNodes(4) = 4
                            ReDim Faces(5, 4)
                            ' Face 1
                            Faces(0, 0) = 1
                            Faces(0, 1) = 2
                            Faces(0, 2) = 3
                            ' Face 2
                            Faces(1, 0) = 4
                            Faces(1, 1) = 6
                            Faces(1, 2) = 5
                            ' Face 3
                            Faces(2, 0) = 1
                            Faces(2, 1) = 4
```

```
            Faces(2, 2) = 5
            Faces(2, 3) = 2
            ' Face 4
            Faces(3, 0) = 2
            Faces(3, 1) = 5
            Faces(3, 2) = 6
            Faces(3, 3) = 3
            ' Face 5
            Faces(4, 0) = 3
            Faces(4, 1) = 6
            Faces(4, 2) = 4
            Faces(4, 3) = 1
        ' 4 - node (tetrahedral) 3D continuum elements
        Case "c3d4"
            ReDim intNumFaceNodes(4)
            intNumFaceNodes(0) = 3
            intNumFaceNodes(1) = 3
            intNumFaceNodes(2) = 3
            intNumFaceNodes(3) = 3
            ReDim Faces(4, 3)
            ' Face 1
            Faces(0, 0) = 1
            Faces(0, 1) = 2
            Faces(0, 2) = 3
            ' Face 2
            Faces(1, 0) = 1
            Faces(1, 1) = 4
            Faces(1, 2) = 2
            ' Face 3
            Faces(2, 0) = 2
            Faces(2, 1) = 4
            Faces(2, 2) = 3
            ' Face 4
            Faces(3, 0) = 3
            Faces(3, 1) = 4
            Faces(3, 2) = 1
        Case Else
            MsgBox "An error occurred while gathering _
                element information: Element type not _
                supported.", vbCritical, Me.Caption
            Exit Sub
    End Select
    ' ---------------------------------------------------------
    ' Determine which continuum element faces are on the _
      vesicle surface and store the node numbers
    ' ---------------------------------------------------------
    ' NOTE: The node order for the hydrostatic element is _
            opposite that of the continuum element face. _
            This is because the positive direction of the _
            normal given by the right-hand rule must point _
            into the fluid (out of the continuum element) _
            (See 18.8.1-2 of ABAQUS Analysis User's Manual _
            Vol. IV: Elements).
    ReadStr = EIStream.ReadLine
    While Left(ReadStr, 1) <> "*"
        DataLine = Split(ReadStr & ",", ",")
        For intFace = 0 To UBound(Faces, 1) - 1
            FluidFace = True
            For intNode = 0 To intNumFaceNodes(intFace) - 1
                If Nodes(CLng(DataLine(Faces(intFace, _
                  intNode)))).OnBoundary = False Then
                    FluidFace = False
                End If
            Next
            ' If this is a fluid face, create a hydrostatic _
              fluid element
            If FluidFace = True Then
                ReDim Preserve HydElems(lngN + 1)
                HydElems(lngN).NumNodes = _
                    intNumFaceNodes(intFace)
                For intNode = 0 To intNumFaceNodes(intFace) _
                  - 1
                    HydElems(lngN)_
                        .Nodes(intNumFaceNodes(intFace) - _
                        intNode - 1) = _
                        CLng(DataLine(Faces(intFace, _
                        intNode)))
                    Nodes(CInt(DataLine(Faces(intFace, _
                        intNode)))).TimesUsed = _
                        Nodes(CInt(DataLine(Faces(intFace, _
                        intNode)))).TimesUsed + 1
                Next
                lngN = lngN + 1
            End If
        Next
        lngM = CLng(DataLine(0)) + 1
        ReadStr = EIStream.ReadLine
    Wend
    If LCase(Left(ReadStr, 8)) = "*element" Then
        GoTo MoreElements
    End If
    If EIStream.AtEndOfStream Then
        MsgBox "An error occurred while gathering element _
            information: End of element definition not _
            found for part " & Parts(intPart).Name, _
            vbCritical, Me.Caption
        Exit Sub
    End If
    lngInsLine = EIStream.Line - 2
    EIStream.Close
    ' ---------------------------------------------------
    ' Write the element information to the output file
    ' ---------------------------------------------------
    ' Reopen the input file
    Set EIStream = EIFile.OpenAsTextStream(1)
    Set OutStream = OutFile.OpenAsTextStream(2)
    ' Write the input file information to the end of the _
      *Element section for the current part/instance
    For lngN = 1 To lngInsLine
        OutStream.WriteLine EIStream.ReadLine
    Next
    If UBound(HydElems) = 0 Then
        GoTo SkipElem
    End If
```

```
                    For intHydElType = 3 To MaxFaceNodes
                        lngNumElem(intHydElType - 1) = 0
                        ' Write the hydrostatic fluid element definitions
                        For lngN = 0 To UBound(HydElems) - 1
                            If HydElems(lngN).NumNodes = intHydElType Then
                                ' Hydrostatic fluid element definition line
                                If lngNumElem(intHydElType - 1) = 0 Then
                                    OutStream.WriteLine "*Element, _
                                        type=F3D" & CStr(intHydElType)
                                End If
                                lngNumElem(intHydElType - 1) = _
                                    lngNumElem(intHydElType - 1) + 1
                                strHydNodeDef = ""
                                For intNode = 0 To intHydElType - 1
                                    strHydNodeDef = strHydNodeDef & ", " & _
                                        CStr(HydElems(lngN).Nodes(intNode))
                                Next
                                OutStream.WriteLine CStr(lngM + lngN) & _
                                    strHydNodeDef
                            End If
                        Next
                    Next
                    lngTotalNumEl = lngTotalNumEl + UBound(HydElems)
                    ' Write the rest of the file
SkipElem:           While EIStream.AtEndOfStream = False
                        OutStream.WriteLine EIStream.ReadLine
                    Wend
                    EIStream.Close
                    OutStream.Close
                    ' Write the scratch file
                    FSO.CopyFile txtOutputFile.Text, _
                        Left(txtOutputFile.Text, Len(txtOutputFile.Text) - _
                        4) & "~1.inp", True
                Next
                ' Write the report information for this vesicle
                RepStream.WriteLine "Vesicle:            " & CStr(intVx) & _
                    " - " & CStr(intVy) & " - " & CStr(intVz)
                RepStream.WriteLine "Boundary Nodes:     " & CStr(lngNumOB)
                RepStream.WriteLine "Elements Created:   " & _
                    CStr(lngNumElem(2)) & " F3D3 and " & _
                    CStr(lngNumElem(3)) & " F3D4"
                RepStream.WriteLine
            Next
        Next
    Next
    ' Delete the scratch file
    FSO.DeleteFile Left(txtOutputFile.Text, Len(txtOutputFile.Text) - 4) & _
        "~1.inp"
    RepStream.Close
    ' Give the user the good news!
    MsgBox "Finished! Created " & CStr(lngTotalNumEl) & " hydrostatic fluid _
        elements.", vbInformation, Me.Caption
End Sub

Private Sub cmdExit_Click()
    Unload Me
End Sub
```

APPENDIX G

DEVELOPMENT OF A GENERALIZED COTRANSPORTER MODEL

The following cotransporter model is based upon the work of Sanders *et al.* [13] and can be applied to an H^+/Sucrose cotransporter [93]. His model is based upon enzyme kinetics for Class-I cotransporters or those having a single transport loop in which the carrier transits the membrane by one path as a charged species and transits by one other path as a neutral species. In his model, there are two cotransported species H^+ and S. The activator H^+ is the species that travels down its electrochemical gradient to provide free energy to the cotransporter. Species S is the substrate, or transported species. In this case, the substrate, S, is transported against its electrochemical gradient. The activator and/or substrate bond to the free carrier C_n before crossing the membrane. The binding order of the activator and substrate can be one of four combinations: (i) First on – Last off (FL+), (ii) First on – First off (FF+), (iii) Last on – Last off (LL+), (iv) Last on – First off (LF+). The '+' designation indicates that the loaded carrier has a net positive charge. Another set of models can be created by having the unloaded carrier possess a net negative charge; these model will have the '−' designation. The cotransporter reaction occurs in six stages which are represented in Fig G.1 where side 'e' of the membrane is the extracellular space and side 'i' of the membrane is the intracellular space.

$$\begin{tikzcd}
C_e \arrow[r, shift left, "k_{65}"] \arrow[d, shift left, "k_{46}"] & C_i \arrow[l, shift left, "k_{56}"] \arrow[d, shift left, "k_{53}"] \\
SC_e \arrow[u, shift left, "k_{64}"] \arrow[d, shift left, "k_{24}"] & SC_i \arrow[u, shift left, "k_{35}"] \arrow[d, shift left, "k_{31}"] \\
SCH_e^+ \arrow[u, shift left, "k_{42}"] \arrow[r, shift left, "k_{21}"] & SCH_i^+ \arrow[u, shift left, "k_{13}"] \arrow[l, shift left, "k_{12}"]
\end{tikzcd}$$

(A): FL+

$$\begin{tikzcd}
C_e \arrow[r, shift left, "k_{65}"] \arrow[d, shift left, "k_{46}"] & C_i \arrow[l, shift left, "k_{56}"] \arrow[d, shift left, "k_{53}"] \\
SC_e \arrow[u, shift left, "k_{64}"] \arrow[d, shift left, "k_{24}"] & CH_i^+ \arrow[u, shift left, "k_{35}"] \arrow[d, shift left, "k_{31}"] \\
SCH_e^+ \arrow[u, shift left, "k_{42}"] \arrow[r, shift left, "k_{21}"] & SCH_i^+ \arrow[u, shift left, "k_{13}"] \arrow[l, shift left, "k_{21}"]
\end{tikzcd}$$

(B): FF+

$$\begin{tikzcd}
C_e \arrow[r, shift left, "k_{56}"] \arrow[d, shift left, "k_{46}"] & C_i \arrow[l, shift left, "k_{56}"] \arrow[d, shift left, "k_{53}"] \\
CH_e^+ \arrow[u, shift left, "k_{64}"] \arrow[d, shift left, "k_{24}"] & CS_i \arrow[u, shift left, "k_{35}"] \arrow[d, shift left, "k_{31}"] \\
SCH_e^+ \arrow[u, shift left, "k_{42}"] \arrow[r, shift left, "k_{21}"] & SCH_i^+ \arrow[u, shift left, "k_{13}"] \arrow[l, shift left, "k_{12}"]
\end{tikzcd}$$

(C): LL+

$$\begin{tikzcd}
C_e \arrow[r, shift left, "k_{65}"] \arrow[d, shift left, "k_{46}"] & C_i \arrow[l, shift left, "k_{56}"] \arrow[d, shift left, "k_{53}"] \\
CH_e^+ \arrow[u, shift left, "k_{64}"] \arrow[d, shift left, "k_{24}"] & CH_i^+ \arrow[u, shift left, "k_{35}"] \arrow[d, shift left, "k_{31}"] \\
SCH_e^+ \arrow[u, shift left, "k_{42}"] \arrow[r, shift left, "k_{21}"] & SCH_i^+ \arrow[u, shift left, "k_{13}"] \arrow[l, shift left, "k_{12}"]
\end{tikzcd}$$

(D): LF+

Figure G.1: Schematic Representation of Cotransporter Reactions with different binding orders of the activator and substrate: (A) First on – Last off (FL+), (B) First on – First off (FF+), (C) Last on – Last off (LL+), (D) Last on – First off (LF+). This shows the different orders with which S and H^+ can bind and release from the transporter C.

This model assumes that the total number of transporters, free and loaded, is constant, and that there is no net movement of transporter binding sites from one face of the membrane to the other (i.e. the transporters are in a steady-state). Sanders' model starts by expressing the unidirectional flux in the Michaelis-Menten form, i.e. the fluxes can be expressed in terms of the Michaelis-Menten equation [94],

$$v = \frac{V[\mathrm{S}]}{K_\mathrm{m} + [\mathrm{S}]} \tag{G.1}$$

where v is the velocity of the reaction, V is the maximum velocity of the reaction, and K_{m} is the concentration of the substrate S at half-maximal velocity. Both V and K_{m} are constants determined from experimental data using an appropriate method (such as a Lineweaver-Burk plot). The unidirectional flux of substrate, S, from side 'e' to side 'i' of the membrane is given by [13],

$$J_{\mathrm{S}}^{\mathrm{e}\to\mathrm{i}} = \frac{J_{\max}[\mathrm{S}]_{\mathrm{e}}}{K_m + [\mathrm{S}]_{\mathrm{e}}} \tag{G.2}$$

where $J_{\max}$ is the maximal velocity and K_m is the Michaelis constant,

$$J_{\max} = N\frac{P|\mathbf{M}_j|}{B_j|{}^{*}\mathbf{M}|} \tag{G.3}$$

$$K_m = \frac{A_j}{k^0 B_j} \tag{G.4}$$

where N is the number of carriers per unit of membrane area, P is the product of all forward reaction constants from the first ${}^{*}\mathrm{S}$-bound state through ${}^{*}\mathrm{S}$ release, $|\mathbf{M}|$ is the characteristic determinant for the entire carrier system with $|\mathbf{M}_j|$ as the determinant associated with the S-binding form of the carrier j = 4 or 6, and $|{}^{*}\mathbf{M}|$ is the characteristic determinant for the isotopically labeled portion of the carrier system. Relations for P and the characteristic determinants in terms of the reaction constants are given in Table G.1.

Table G.1: Expansion of terms in Eqs. (G.3 − G.4)

	FF+	FL+	LF+	LL+
j	6		4	
$\lvert \mathbf{M}_j \rvert$	$k_{13}k_{35}k_{56}[k_{46}(k_{21}+k_{24})+k_{42}k_{21}] + k_{46}k_{24}k_{12}[k_{31}(k_{53}+k_{56})+k_{35}k_{56}]$		$k_{35}k_{56}k_{64}[k_{24}(k_{12}+k_{13})+k_{21}k_{13}] + k_{24}k_{12}k_{31}[k_{53}(k_{65}+k_{64})+k_{56}k_{64}]$	
A_j	$k_{46}(k_{24}+k_{21})\cdot [k_{65}k_{53}(k_{13}+k_{31})+k_{13}k_{35}(k_{65}+k_{56})]+ (k_{42}k_{21}k_{13}+k_{46}k_{24}k_{12}) \cdot[k_{65}(k_{35}+k_{53})+k_{35}k_{56}]+ k_{53}k_{65}k_{12}k_{31}(k_{46}+k_{42}+k_{24})+k_{46}k_{24}k_{12}k_{31}\cdot (k_{53}+k_{56}+k_{65})+k_{65}k_{42}k_{21}k_{31}k_{53}$		$k_{24}(k_{12}+k_{13})\cdot [k_{46}k_{65}(k_{35}+k_{53})+k_{35}k_{56}(k_{46}+k_{64})] (k_{21}k_{13}k_{35}+k_{24}k_{12}k_{31}) \cdot[k_{46}(k_{56}+k_{65})+k_{56}k_{64}]+ k_{65}k_{46}k_{31}k_{53}(k_{24}+k_{21}+k_{12})+k_{24}k_{12}k_{31}k_{53}\cdot (k_{65}+k_{64}+k_{46})+k_{46}k_{21}k_{13}k_{53}k_{65}$	
$k^0 B_j$	$k^0_{64}\left\{\begin{matrix}(k_{56}+k_{53})\cdot \\ [k_{42}k_{21}(k_{13}+k_{31})+k_{12}k_{31}(k_{42}+k_{24})] \\ +k_{35}k_{56}[(k_{42}+k_{24})(k_{12}+k_{13})+k_{42}k_{21}] \\ +k_{21}k_{13}k_{35}(k_{42}+k_{56})\end{matrix}\right\}$		$k^0_{42}\left\{\begin{matrix}(k_{65}+k_{64})\cdot \\ [k_{21}k_{13}(k_{35}+k_{53})+k_{31}k_{53}(k_{21}+k_{12})] \\ +k_{56}k_{64}[(k_{21}+k_{12})(k_{31}+k_{35})+k_{21}k_{13}] \\ +k_{13}k_{35}k_{56}(k_{21}+k_{64})\end{matrix}\right\}$	
P	$k_{42}k_{21}k_{13}$	$k_{42}k_{21}k_{13}k_{35}$	$k_{21}k_{13}$	$k_{21}k_{13}k_{35}$
$\lvert {}^*\mathbf{M} \rvert$	$k_{46}k_{24}(k_{12}+k_{13}) + k_{21}k_{13}(k_{46}+k_{42})$	$k_{35}\cdot \begin{bmatrix} k_{46}k_{24}(k_{12}+k_{13}) \\ +k_{21}k_{13}(k_{46}+k_{42})\end{bmatrix} + k_{46}k_{24}k_{12}k_{31}$	$k_{24}(k_{12}+k_{13}) + k_{21}k_{13}$	$k_{13}k_{35}(k_{24}+k_{21}) + k_{24}k_{12}(k_{31}+k_{35})$
k_{64}	$k^0_{64}[\mathrm{S}]_\mathrm{e}$	$k^0_{64}[\mathrm{S}]_\mathrm{e}$	$k^0_{64}[\mathrm{H}^+]_\mathrm{e}$	$k^0_{64}[\mathrm{H}^+]_\mathrm{e}$
k_{42}	$k^0_{42}[\mathrm{H}^+]_\mathrm{e}$	$k^0_{42}[\mathrm{H}^+]_\mathrm{e}$	$k^0_{42}[\mathrm{S}]_\mathrm{e}$	$k^0_{42}[\mathrm{S}]_\mathrm{e}$
k_{31}	$k^0_{31}[\mathrm{H}^+]_\mathrm{i}$	$k^0_{31}[\mathrm{S}]_\mathrm{i}$	$k^0_{31}[\mathrm{H}^+]_\mathrm{i}$	$k^0_{31}[\mathrm{S}]_\mathrm{i}$
k_{53}	$k^0_{53}[\mathrm{S}]_\mathrm{i}$	$k^0_{53}[\mathrm{H}^+]_\mathrm{i}$	$k^0_{53}[\mathrm{S}]_\mathrm{i}$	$k^0_{53}[\mathrm{H}^+]_\mathrm{i}$

In the above table, the terms k^0_{nm} represent the true reaction constant without respect to ligand concentration. Also, the reaction constants for the charge transit step are related to the membrane potential by,

$$k_{12} = k^0_{12}\exp(zu/2) \tag{G.5}$$

$$k_{21} = k_{21}^{0} \exp(-zu/2) \tag{G.6}$$

for the '+' models in which the loaded carrier is the charge-transit step and by,

$$k_{56} = k_{56}^{0} \exp(zu/2) \tag{G.7}$$

$$k_{65} = k_{65}^{0} \exp(-zu/2) \tag{G.8}$$

for the '−' models in which the unloaded carrier is the charge transit step where k_{nm}^{0} are the reaction constants at zero membrane potential and *u* is the reduced membrane potential given by,

$$u = \frac{Fv}{RT} \tag{G.9}$$

where *v* is the membrane potential. In order to maintain steady-state operation models,

$$k_{65} = k_{12} \tag{G.10}$$

$$k_{56} = k_{21} \tag{G.11}$$

The net flux of substrate across the membrane is determined by,

$$J_{\mathrm{S}} = J_{\mathrm{S}}^{\mathrm{e}\to\mathrm{i}} - J_{\mathrm{S}}^{\mathrm{i}\to\mathrm{e}} \tag{G.12}$$

where $J_{\mathrm{S}}^{\mathrm{i}\to\mathrm{e}}$ is the unidirectional flux from the intracellular space to the extracellular space. In order to determine the net flux across the membrane, a total of twelve parameters must be determined (Figure G.1). Sanders demonstrates that under three types of experimental conditions, the Michaelis-Menten coefficients can be simplified,

1. Saturating membrane potential, $\left(J_{\max}^{\mathrm{sv}}, K_{m}^{\mathrm{sv}}\right)$ (i.e. the membrane potential (interior negative) is large enough to be saturating in its effect on flux)
2. Zero Trans-Ligand, $\left(J_{\max}^{\mathrm{zt}}, K_{m}^{\mathrm{zt}}\right)$ (i.e. the intracellular concentration of coupled substrate is zero. This is attained by setting both $[\mathrm{S}]_{\mathrm{i}}$ and $[\mathrm{H}^{+}]_{\mathrm{i}}$ at zero. Since the proton concentration cannot be set to zero, the pH must be set high enough that it is not rate limiting.)
3. Saturating Cis-Driver Ion, $\left(J_{\max}^{\mathrm{sd}}, K_{m}^{\mathrm{sd}}\right)$ (i.e. the extracellular concentration of the driver ion is high enough to make the reaction $\mathrm{SC}_{\mathrm{e}} + \mathrm{H}_{\mathrm{e}}^{+} \rightleftharpoons \mathrm{SCH}_{\mathrm{e}}^{+}$ at the outer surface of the membrane limited only by the availability of the carrier CS_{e})

The Michaelis-Menten coefficients for the above experimental conditions are related to the model parameters by [13],

$$J_{\max}^{\mathrm{sv}} = \frac{N[\mathrm{H}^{+}]_{\mathrm{e}} k_{42}^{0} k_{13} k_{35} k_{56}}{[\mathrm{H}^{+}]_{\mathrm{e}} k_{42}^{0} \left[(k_{13} + k_{31})(k_{53} + k_{56}) + k_{35}(k_{13} + k_{56})\right] + k_{13} k_{35} k_{56}} \tag{G.13}$$

$$K_m^{\mathrm{sv}} = \frac{\left\{[\mathrm{H}^+]_\mathrm{e} k_{42}^0 + k_{46}\right\}\left\{k_{53}k_{65}\left(k_{13}+k_{31}\right)+k_{13}k_{35}\left(k_{56}+k_{65}\right)\right\}}{k_{64}^0\left\{[\mathrm{H}^+]_\mathrm{e} k_{42}^0\left[\left(k_{13}+k_{31}\right)\left(k_{53}+k_{56}\right)+k_{35}\left(k_{13}+k_{56}\right)\right]+k_{13}k_{35}k_{56}\right\}} \tag{G.14}$$

$$J_{\max}^{\mathrm{zt}} = \frac{N[\mathrm{H}^+]_\mathrm{e} k_{42}^0 k_{21}k_{13}k_{35}k_{56}}{\begin{array}{l}[\mathrm{H}^+]_\mathrm{e} k_{42}^0\left[k_{35}k_{56}\left(k_{21}+k_{12}+k_{13}\right)+k_{21}k_{13}\left(k_{35}+k_{56}\right)\right]\\ +k_{35}k_{56}\left[k_{24}\left(k_{12}+k_{13}\right)+k_{21}k_{13}\right]\end{array}} \tag{G.15}$$

$$K_m^{\mathrm{zt}} = \frac{[\mathrm{H}^+]_\mathrm{e} k_{42}^0 k_{21}k_{13}k_{35}\left(k_{56}+k_{65}\right)+k_{46}k_{35}\left(k_{56}+k_{65}\right)\left[k_{24}\left(k_{12}+k_{13}\right)+k_{21}k_{13}\right]}{k_{64}^0\left\{\begin{array}{l}[\mathrm{H}^+]_\mathrm{e} k_{42}^0\left[k_{35}k_{56}\left(k_{21}+k_{12}+k_{13}\right)+k_{21}k_{13}\left(k_{35}+k_{56}\right)\right]\\ +k_{35}k_{56}\left[k_{24}\left(k_{12}+k_{13}\right)+k_{21}k_{13}\right]\end{array}\right\}} \tag{G.16}$$

$$J_{\max}^{\mathrm{sd}} = \frac{Nk_{21}^0\exp(-u/2)k_{13}k_{35}k_{56}}{\begin{array}{l}k_{21}^0\exp(-u/2)\left[k_{31}\left(k_{53}+k_{56}\right)+k_{35}k_{56}+k_{13}\left(k_{35}+k_{53}+k_{56}\right)\right]\\ +k_{12}^0\exp(u/2)\left[k_{31}\left(k_{53}+k_{56}\right)+k_{35}k_{56}\right]+k_{13}k_{35}k_{56}\end{array}} \tag{G.17}$$

$$K_m^{\mathrm{sd}} = \frac{\begin{array}{l}k_{21}^0\exp(-u/2)\left[k_{13}k_{35}\left(k_{56}+k_{65}\right)+k_{53}k_{65}\left(k_{13}+k_{31}\right)\right]\\ +k_{12}^0\exp(u/2)k_{31}k_{53}k_{65}\end{array}}{k_{64}^0\left\{\begin{array}{l}k_{21}^0\exp(-u/2)\left[k_{31}\left(k_{53}+k_{56}\right)+k_{35}k_{56}+k_{13}\left(k_{35}+k_{53}+k_{56}\right)\right]\\ +k_{12}^0\exp(u/2)\left[k_{31}\left(k_{53}+k_{56}\right)+k_{35}k_{56}\right]+k_{13}k_{35}k_{56}\end{array}\right\}} \tag{G.18}$$

A system of equations consisting of 6 independent equations relating the model parameters to the Michaelis-Menten coefficients can be taken from Eqs. (G.13 − G.18). If the experiments are reversed, another 6 independent equations can be created by deriving relations for the Michaelis-Menten coefficients for the unidirectional flux from the intracellular space to the extracellular space. The resulting system of equations can be solved for true kinetic parameters.

APPENDIX H

DEVELOPMENT OF ELECTRODIFFUSION MEMBRANE POTENTIAL MODEL

The electrodiffusion processes that occur near a biological membrane can be modeled using the coupled Nernst-Planck, continuity and Poisson equations, which in one-dimensional form can be expressed as,

$$J_i = -D_i\left[\frac{\partial c_i}{\partial x} + c_i \frac{z_i F}{RT}\frac{\partial \phi}{\partial x}\right] \tag{H.1}$$

$$\frac{\partial J_i}{\partial x} = -\frac{\partial c_i}{\partial t} \tag{H.2}$$

$$\frac{\partial^2 \phi}{\partial x^2} = -\frac{F}{\varepsilon_0 K}\sum_i z_i c_i \tag{H.3}$$

where J_i is the ion flux, D_i is the diffusion coefficient and c_i is concentration of ion *i*, ϕ is the potential at point *x*, ε_0 is the permittivity of free space ($\varepsilon_0 = 8.854187817\ \mathrm{C}^2/\mathrm{N}\cdot\mathrm{m}^2$), and *K* is the dielectric constant for the solution. The system consists of an exterior solution phase from $-\delta \le x < 0$, a membrane phase from $0 \le x \le d$, and an interior solution phase from $d < x \le d + \delta$. The membrane phase is assumed to have some fixed charge with a concentration *X* that can vary across the membrane. The Poisson equation is piecewise continuous across the three phases. Thus,

$$\frac{\partial^2 \phi}{\partial x^2} = -\frac{F}{\varepsilon_0 K}\begin{cases} X + \sum_i z_i c_i & 0 \le x \le d \\ \sum_i z_i c_i & -\delta \le x < 0 \text{ and } d < x \le d + \delta \end{cases} \tag{H.4}$$

The Nernst-Planck and continuity equations are combined to give,

$$\frac{\partial c_i}{\partial t} = D_i \frac{\partial}{\partial x}\left[\frac{\partial c_i}{\partial x} + c_i \frac{z_i F}{RT}\frac{\partial \phi}{\partial x}\right] \tag{H.5}$$

Note that if the membrane is not permeable to a particular ion, then the above equation will also be piecewise continuous with the right-hand side equal to zero in the membrane phase. Before the problem can be solved for nonequilibrium conditions, the initial ion and potential distributions must be determined. To accomplish this, an equilibrium problem is formulated using Eqs. (H.4) and (H.5) with the following boundary conditions,

$$\frac{\partial c_i}{\partial x}(-\delta,t) = \frac{\partial c_i}{\partial x}(d+\delta,t) = 0 \tag{H.6}$$

$$\frac{\partial \phi}{\partial x}(-\delta,t) = \phi(-\delta,t) = \frac{\partial \phi}{\partial x}(d+\delta,t) = 0 \tag{H.7}$$

Note that the boundary condition $\phi(-\delta,t)=0$ will enforce the membrane potential sign convention. The initial conditions which serve as initial guesses for the equilibrium ion and potential distributions are determined using Donnan equilibrium with the ion concentrations in the solution phases equal to those in their respective bulk solutions. The nonequilibrium problem is solved using the results from the initial equilibrium problem as initial conditions and with the following boundary conditions,

$$c_i(-\delta,t) = (c_{0i})_{\mathrm{e}} \tag{H.8}$$

$$c_i(d+\delta,t) = (c_{0i})_{\mathrm{i}} \tag{H.9}$$

$$\phi(-\delta,t) = 0 \tag{H.10}$$

$$\varepsilon_0 K \left.\frac{\partial^2 \phi}{\partial x \partial t}\right|_{(d+\delta,t)} = I(t) + F\sum_i D_i \left.\left[\frac{\partial c_i}{\partial x} + c_i \frac{z_i F}{RT}\frac{\partial \phi}{\partial x}\right]\right|_{(d+\delta,t)} \tag{H.11}$$

where $I(t)$ is the imposed membrane current such as that generated by active or secondary active transport. The last boundary condition in the set represents the total electric current density including the displacement current.

The solution of Eqs. (H.4) and (H.5) for both problems is carried out using the method lines on an adaptive grid. The equations are spatially discretized using finite differences to form a set of linearly implicit ODEs of the form,

$$\mathbf{A}\frac{d\mathbf{y}}{dt} = \mathbf{f}(\mathbf{y},t) \tag{H.12}$$

where,

$$\mathbf{y}^{\mathrm{T}} = \left(c_{11}, c_{21}, \ldots, c_{n1}, \phi_1, \ldots\ldots, c_{1\mathrm{NPT}}, c_{2\mathrm{NPT}}, \ldots, c_{n\mathrm{NPT}}, \phi_{\mathrm{NPT}}\right) \tag{H.13}$$

$$\mathbf{A} = \mathrm{diag}\left(1, 1, 1, 1, 1, 0, 1, 1, 0, \ldots, 1, 1, 0\right) \tag{H.14}$$

for the initial equilibrium problem and,

$$\mathbf{y}^{\mathrm{T}} = \left(c_{11}, c_{21}, \ldots, c_{n1}, \phi_1, \ldots\ldots, c_{1\mathrm{NPT}}, c_{2\mathrm{NPT}}, \ldots, c_{n\mathrm{NPT}}, \phi_{\mathrm{NPT}}, \left(\partial\phi/\partial x\right)_{\mathrm{NPT}}\right) \tag{H.15}$$

$$\mathbf{A} = \mathrm{diag}\left(1, 1, 1, 1, 1, 0, 1, 1, 0, \ldots, 1, 1, 0, 1\right) \tag{H.16}$$

for the general problem where *n* is the number of types of ions present in the solution and NPT is the number of spatial grid points. Note that since the matrix **A** is singular, the system is a set of differential-algebraic equations (DAEs). Note also that the Dirichlet condition $\phi(-\delta, t) = 0$ is incorporated into the problem by introducing the dummy equation $d\phi_1/dt = 0$. Since the system is highly nonlinear and stiff, the LSODI package [74] is recommended to carry out the integration of the DAEs.

REFERENCES

[1] Wickenheiser, A., Garcia, E., and Waszak, M., 2004, "Evaluation of bio-inspired morphing concepts with regard to aircraft dynamics and performance", Proc. of SPIE Smart Structures and Materials, March 15 – 18, 2004, San Diego, CA, **5390**, pp. 202 – 211.

[2] Manzo, J., Garcia, E., Wickenheiser, A., and Horner, G. C., 2004, "Adaptive structural systems and compliant skin technology of morphing aircraft structures", Proc. of SPIE Smart Structures and Materials, March 15 – 18, 2004, San Diego, CA, **5390**, pp. 225 – 234.

[3] Tan, H., Leo, D. J., Park, T., and Long, T. E., 2003, "Investigation of Performance of Active Microcapsule Actuation", Proc. of ASME – IMECE, Nov. 15 – 21, 2003, Washington, D.C., pp. 435 – 444.

[4] Sundaresan, V. B., Tan, H., Leo, D. J., and Cuppoletti, J., 2004, "Investigation on High Energy Density Materials Utilizing Biological Transport Mechanisms", Proc. of ASME – IMECE, Nov. 15 – 21, 2004, Anaheim, CA, **69**, pp. 55 – 62.

[5] Sundaresan, V. B. and Leo, D. J., 2006, "Protein-based Microhydraulic Transport for Controllable Actuation", Proc. of SPIE Smart Structures and Materials, Feb. 26 – March 2, 2006, San Diego, CA, **6168**.

[6] Tzou, H. S., Lee, H.-J., and Arnold, S. M., 2004, "Smart Materials, Precision Sensors/Actuators, Smart Structures, and Structronic Systems", Mech. Adv. Mater. Struc., **11**, pp. 367 – 393.

[7] Newnham, R. E., 1992, "Piezoelectric Sensors and Actuators: Smart Materials", Proc. of 46th IEEE Frequency Control Symposium, May 27 – 29, 1992, Hershey, PA, pp. 513 – 524.

[8] Oates, W. S., Mauck, L. D., and Lynch, C. S., 2000, "PZT piston driven hydraulic pump development", Proc. of 12th IEEE International Symposium on Applications of Ferroelectrics, Jul. 21 – Aug. 2, 2000, Honolulu, HI, **2**, pp. 733 – 736.

[9] Sundaresan, V. B. and Leo, D. J., 2005, "Chemo-mechanical Model for Actuation Based on Biological Membranes", Proc. of SPIE Smart Structures and Materials, March 1 – 8, 2005, San Diego, CA, **5761**, pp. 108 – 118.

[10] Giurgiutiu, V., Matthews, L., Leo, D. J., and Sundaresan, V. B., 2005, "Concepts for Power and Energy Analysis in Nastic Structures", Proc. of ASME – IMECE, Nov. 5 – 11, 2005, Orlando, FL.

[11] Turner, R. J., 1981, "Kinetic Analysis of a Family of Cotransport Models", Biochim. Biophys. Acta., **649**, pp. 269 – 280.

[12] Turner, R. J., 1982, "General Rate Equations and Rejection Criteria for the Rapid Equilibrium Carrier Model of Cotransport", Biochim. Biophys. Acta., **689**, pp. 444 – 450.

[13] Sanders, D., Hansen, U., Gradmann, D., and Slayman, C. L., 1984, "Generalized Kinetic Analysis of Ion-Driven Cotransport Systems: A Unified Interpretation of Selective Ionic Effects on Michaelis Parameters", J. Membrane Biol., **77**, pp. 123 – 152.

[14] Hilgemann, D. W., 1988, "Numerical Approximations of Sodium-Calcium Exchange", Prog. Biophys. Molec. Biol., **51**, pp. 1 – 45.

[15] Haynes, D. H. and Mandveno, A., 1987, "Computer Modeling of Ca^{2+} Pump Function of Ca^{2+}-Mg^{2+}-ATPase of Sarcoplasmic Reticulum", Physiological Reviews, **67**(**1**), pp. 244 – 284.

[16] Homison, C. and Weiland, L. M., 2005, "Coupled Transport/Hyperelastic Model for Nastic Materials", Proc. of ASME – IMECE, Nov. 5 – 11, 2005, Orlando, FL.

[17] Homison, C. and Weiland, L. M., 2006, "Coupled Transport/Hyperelastic Model for Nastic Materials", Proc. of SPIE Smart Structures and Materials, Feb. 26 – March 2, 2006, San Diego, CA, **6170**.

[18] Collier, J. H. and Messersmith, P. B., 2001, "Phospholipid Strategies in Biomineralization and Biomaterials Research", Annu. Rev. Mater. Res., **31**, pp. 237 – 263.

[19] Sackmann, E., 1996, "Supported Membranes: Scientific and Practical Applications", Science, **271**, pp. 43 – 48.

[20] Trojanowicz, M. and Mulchandani, A., 2004, "Analytical applications of planar bilayer lipid membranes", Anal. Bioanal. Chem., **379**, pp. 347 – 350.

[21] Nelson, D. L. and Cox, M. M., 2000, *Lehninger Principles of Biochemistry, 3rd Ed.*, Worth Publishers, New York.

[22] Mueller, P., Rudin, D. O., and Tien, H. T., 1962, "Reconstitution of Excitable Cell Membrane Structure in Vitro", Circulation, **26**, pp. 1167 – 1171.

[23] Ries, R. S., Choi, H., Blunck, R., Bezanilla, F., and Heath, J. R., 2004, "Black Lipid Membranes: Visualizing the Structure, Dynamics, and Substrate Dependence of Membranes", J. Phys. Chem. B, **108**, pp. 16040 – 16049.

[24] Sundaresan, V. B., Akle, B., and Leo, D. J., 2006, "Investigation on Micro-patterned Gold Plated Polymer Substrate for a Micro Hydraulic Actuator", Proc. of SPIE Smart Structures and Materials, Feb. 26 – March 2, 2006, San Diego, CA, **6170**.

[25] Stein, W. D., 1990, *Channels, Carriers, and Pumps. An Introduction to Membrane Transport*, Academic Press, Inc., San Diego, CA.

[26] Tien, H. T., 1985, "Planar Bilayer Lipid Membranes", Prog. Surf. Sci., **19**(**3**), pp. 169 – 274.

[27] Schultz, S. G., 1980, *Basic Principles of Membrane Transport*, Cambridge University Press, New York.

[28] Stein, W. D., 1986, *Transport and Diffusion across Cell Membranes*, Academic Press, Inc., Orlando, FL.

[29] Kedem, O., 1960, "Criteria of Active Transport", In *Membrane Transport and Metabolism: Proceedings of a Symposium held in Prague*, Kleinzeller, A., and Kotyk, A., eds., Academic Press, New York, pp. 87 – 93.

[30] Friedman, M. H., 1986, *Principles and Models of Biological Transport*, Springer-Verlag, Berlin.

[31] Post, R. L., 1999, "Active Transport and Pumps", In *Current Topics in Membranes Vol. 48: Membrane Permeability: 100 years since Ernest Overton*, Deamer, D. W., Kleinzeller, A., and Fambrough, D. M., eds., Academic Press, San Diego, CA, pp. 397 – 417.

[32] Disalvo, A., Siddiqi, F. A., and Ti Tien, H., 1989, "Membrane Transport with Emphasis on Water and Nonelectrolytes in Experimental Lipid Bilayers and Biomembranes", In *Water Transport in Biological Membranes, Vol. 1*, Benga, G., ed., CRC Press, Inc., Boca Raton, FL, pp. 41 – 75.

[33] Cooper, K., Jakobsson, E., and Wolynes, P., 1985, "The Theory of Ion Transport through Membrane Channels", Prog. Biophys. Molec. Biol., **46**, pp. 51 – 96.

[34] Katchalsky, A., 1960, "Membrane Permeability and the Thermodynamics of Irreversible Processes", In *Membrane Transport and Metabolism: Proceedings of a Symposium held in Prague*, Kleinzeller, A., and Kotyk, A., eds., Academic Press, New York, pp. 69 – 86.

[35] Hille, B., 1992, *Ionic Channels of Excitable Membranes, 2nd Ed.*, Sinauer Associates, Inc., Sunderland, MA.

[36] Bett, G. C. L. and Rasmusson, R. L., 2002, "Computer Models of Ion Channels", In *Quantitative Cardiac Electrophysiology*, Cabo, C. and Rosenbaum, D. S., eds., Marcel Dekker, Inc., New York, pp. 1 – 60.

[37] Jauch, P. and Läuger, P., 1986, "Electrogenic Properties of the Sodium-Alanine Cotransporter in Pancreatic Acinar Cells: II. Comparison with Transport Models", J. Membrane Biol., **94**, pp. 117 – 127.

[38] Tanner, W. and Caspari, T., 1996, "Membrane Transport Carriers", Annu. Rev. Plant Physiol. Plant Mol. Biol., **47**, pp. 595 – 626.

[39] Nicholls, D. G. and Ferguson, S. J., 2002, *Bioenergetics 3*, Academic Press, San Diego, CA.

[40] Wright, E. M. and Loo, D. D.F., 2000, "Coupling between Na^+, Sugar, and Water Transport across the Intestine", Ann. N. Y. Acad. Sci., **915**, pp. 54 – 66.

[41] Finkelstein, A., 1987, *Water Movement Through Lipid Bilayers, Pores, and Plasma Membranes. Theory and Reality*, John Wiley & Sons, Inc., New York.

[42] Su, Y., Lin, L., and Pisano, A., 2002, "A Water-Powered Osmotic Microactuator", J. Microelectromech. S., **11**(**6**), pp. 736 – 742.

[43] Robinson, R. A. and Stokes, R. H., 1965, *Electrolyte Solutions, 2nd Ed.*, Butterworths, London.

[44] Sten-Knudsen, O., 2002, *Biological Membranes: Theory of transport, potentials and electric impulses*, Cambridge University Press, New York.

[45] Hodgkin, A. L., Huxley, A. F., and Katz, B., 1952, "Measurement of Current-Voltage Relations in the Membrane of the Giant Axon of *Loligo*", J. Physiol. (London), **116**, pp. 424 – 448.

[46] Hodgkin, A. L. and Huxley, A. F., 1952, "A Quantitative Description of Membrane Current and its Application to Conduction and Excitation in Nerve", J. Physiol. (London), **117**, pp. 500 – 544.

[47] Cevc, G., 1990, "Membrane Electrostatics", Biochim. Biophys. Acta, **1031**(**3**), pp. 311 – 382.

[48] Grimnes, S. and Martinsen, Ø. G., 2000, *Bioimpedance and Bioelectricity Basics*, Academic Press, San Diego, CA.

[49] Walz, D., Teissié, J., and Milazzo, G., 2004, *Bioelectrochemistry of Membranes*, Birkhaüser Verlag, Boston, MA.

[50] Cevc, G. and Marsh, D., 1987, *Phospholipid Bilayers: Physical Principles and Models*, Wiley, New York.

[51] Manzanares, J. A., Murphy, W. D., Mafé, S., and Reiss, H., 1993, "Numerical Simulation of the Nonequilibrium Diffuse Double Layer in Ion-Exchange Membranes", J. Phys. Chem., **97**, pp. 8524 – 8530.

[52] Sokalski, T., Lingenfelter, P., and Lewenstam, A., 2003, "Numerical Solution of the Coupled Nernst-Planck and Poisson Equations for Liquid Junction and Ion Selective Membrane Potentials", J. Phys. Chem. B, **107**, pp. 2443 – 2452.

[53] Moya, A. A. and Horno, J., 1999, "Application of the Network Simulation Method to Ionic Transport in Ion-Exchange Membranes Including Diffuse Double-Layer Effects", J. Phys. Chem. B, **103**, pp. 10791 – 10799.

[54] Bonting, S. L. and de Pont, J. J. H. H. M., 1981, *New Comprehensive Biochemistry Volume 2 – Membrane Transport*, Elsvier, New York.

[55] Hodgkin, A. L. and Huxley, A. F., 1952, "Currents carried by sodium and potassium ions through the membrane of the giant axon of *Loligo*", J. Physiol. (London), **116**, pp. 449 – 472.

[56] Hodgkin, A. L. and Huxley, A. F., 1952, "The components of membrane conductance in the giant axon of *Loligo*", J. Physiol. (London), **116**, pp. 473 – 496.

[57] Hodgkin, A. L. and Huxley, A. F., 1952, "The dual effect of membrane potential on sodium conductance in the giant axon of *Loligo*", J. Physiol. (London), **116**, pp. 497 – 506.

[58] Cabo C., 2002, "Computation of the Action Potential of a Cardiac Cell", In *Quantitative Cardiac Electrophysiology*, Cabo, C. and Rosenbaum, D. S., eds., Marcel Dekker, Inc., New York, pp. 61 – 104.

[59] Wilders, R., Jongsma, H. J., and van Ginneken, A. C. G., 1991, "Pacemaker activity of the rabbit sinoatrial node: A comparison of mathematical models", Biophys. J., **60**, pp. 1202 – 1216.

[60] DiFrancesco, D. and Noble, D., 1985, "A model of cardiac electrical activity incorporating ionic pumps and concentration changes", Phil. Trans. R. Soc. Lond. B, **307**, pp. 353 – 398.

[61] Cronin, J., 1987, *Mathematical Aspects of Hodgkin-Huxley Neural Theory*, Cambridge University Press, New York.

[62] Rush, S. and Larsen, H., 1978, "A Practical Algorithm for Solving Dynamic Membrane Equations", IEEE Trans. Biomed. Eng., **25**(**4**), pp. 389 – 392.

[63] Moore, J. W. and Ramon, F., 1974, "On Numerical Integration of the Hodgkin and Huxley Equations for a Membrane Action Potential", J. Theor. Biol., **45**, pp. 249 – 273.

[64] Victorri, B., Vinet, A., Roberge, F. A., and Drouhard, J.-P., 1985, "Numerical Integration in the Reconstruction of Cardiac Action Potentials Using Hodgkin-Huxley-Type Models", Comp. Biomed. Res., **18**, pp. 10 – 23.

[65] Byrne, J. H. and Schultz, S. G., 1994, *An Introduction to Membrane Transport and Bioelectricity*, Raven Press, New York.

[66] Stein, W. D., 1989, "Kinetics of Transport: Analyzing, Testing, and Characterizing Models Using Kinetic Approaches", In *Methods in Enzymology Vol. 171, Part R, Transport Theory: Cells and Model Membranes*, Abelson, J. N. and Simon, M. I., eds., Academic Press, San Diego, CA, pp. 23 – 62.

[67] Endresen, L. P., Hall, K., Høye, J. S., and Myrheim, J., 2000, "A theory for the membrane potential of living cells", Eur. Biophys. J., **29**, pp. 90 – 103.

[68] Endresen, L. P., 2000, "Runge-Kutta formulas for cardiac oscillators", Institutt for fysikk, NTNU, N-7034 Trondheim, Norway.

[69] Endresen, L. P., 1997, "Chaos in weakly-coupled pacemaker cells", J. Theor. Biol., **184**, pp. 41 – 50.

[70] *ABAQUS Analysis User's Manual, Vols. II, III, IV, and VI*, 2004, ABAQUS, Inc., Providence, RI.

[71] Sussman, T. and Bathe, K.-J., 1987, "A Finite Element Formulation for Nonlinear Incompressible Elastic and Inelastic Analysis", Comput. Struct., **26**(**1-2**), pp. 357 – 409.

[72] Doll, S. and Schweizerhof, K., 2000, "On the Development of Volumetric Strain Energy Functions", J. Appl. Mech., **67**, pp. 17 – 21.

[73] Treloar, L. R. G., 1975, *The Physics of Rubber Elasticity, 3rd Ed.*, Clarendon Press, Oxford.

[74] Hindmarsh, A. C., 1983, "ODEPACK, A Systematized Collection of ODE Solvers", In *Scientific Computing (IMACS Transactions on Scientific Computation, Vol. 1)*, Stepleman, R. S., Carver, M., Peskin, R., Ames, W. F., and Vichnevetsky, R., eds., North-Holland, Amsterdam, pp. 55 – 64.

[75] Hairer, E., Nørsett, S. P., and Wanner, G., 1987, *Solving ordinary differential equations I*, Springer-Verlag, New York.

[76] Hairer, E. and Wanner, G., 1996, *Solving ordinary differential equations II*, Springer-Verlag, New York.

[77] Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P., 1992, *Numerical Recipes in FORTRAN, 2nd Ed.*, Cambridge University Press, New York.

[78] Radhakrishnan, K. and Hindmarsh, A. C., 1993, “Description and Use of LSODE, the Livermore Solver for Ordinary Differential Equations”, LLNL Report UCRL-ID-113855.

[79] Brown, P. N., Byrne, G. D., and Hindmarsh, A. C., 1989, “VODE: A Variable-Coefficient ODE Solver”, SIAM J. Sci. Stat. Comput., **10**(**5**), pp. 1038 – 1051.

[80] Mullins, L. J., 1977, “A Mechanism for Na/Ca Transport”, J. Gen. Physiol., **70**, pp. 681 – 695.

[81] Mullins, L. J., 1981, *Ion Transport in Heart*, Raven Press, New York.

[82] Weise, A., Barker, L., Kühn, C., Lalonde, S., Buschmann, H., and Frommer, W. B., 2000, “A New Subfamily of Sucrose Transporters, SUT4, with Low Affinity/High Capacity Localized in Enucleate Sieve Elements of Plants”, The Plant Cell, **12**, pp. 1345 – 1355.

[83] Kühn, C., Barker, L., Bürkle, L., and Frommer, W.-B., 1999, “Update on sucrose transport in higher plants”, J. Exp. Bot., **50**, pp. 935 – 953.

[84] Delrot, S., Atanassova, R., Gomès, E., and Coutos-Thévenot, P., 2001, “Plasma membrane transporters: a machinery for uptake of organic solutes and stress resistance”, Plant Sci., **161**, pp. 391 – 404.

[85] Sundaresan, V. B. and Leo, D. J., 2005, “Experimental Investigation for Chemo-Mechanical Actuation using Biological Transport Mechanisms”, Proc. of ASME – IMECE, Nov. 5 – 11, 2005, Orlando, FL.

[86] MacAulay, N., Hamann, S., and Zeuthen, T., 2004, “Water Transport in the Brain: Role of cotransporters”, Neuroscience, **129**, pp. 1031 – 1044.

[87] *ABAQUS Theory Manual*, 2004, ABAQUS, Inc., Providence, RI.

[88] López-Marqués, R. L., Schiøtt, M., Jakobsen, M. K., and Palmgren, M. G., 2004, “Structure, function, and regulation of primary H^+ and Ca^{2+} pumps”, In *Membrane Transport in Plants*, Blatt, M. R., ed., CRC Press, New York, pp. 72 – 104.

[89] Parent, L., Supplisson, S., Loo, D. D.F., and Wright, E. M., 1992, “Electrogenic Properties of the Cloned Na^+/Glucose Cotransporter: II. A Transport Model under Nonrapid Equilibrium Conditions”, J. Membrane Sci., **125**, pp. 63 – 79.

[90] Johnson, E. A., Lemieux, D. R., and Kootsey, J. M., 1992, “Sodium-Calcium Exchange: Derivation of a State Diagram and Rate Constants from Experimental Data”, J. Theor. Biol., **156**, pp. 443 – 483.

[91] Macdonald, A. G., 1984, “The Effects of Pressure on the Molecular Structure and Physiological Functions of Cell Membranes”, Phil. Trans. R. Soc. Lond. B, **304**, pp. 47 – 68.

[92] Heinemann, S. H., Conti, F., Stühmer, W., and Neher, E., 1987, “Effects of Hydrostatic Pressure on Membrane Processes. Sodium Channels, Calcium Channels, and Exocytosis”, J. Gen. Physiol., **90**, pp. 765 – 778.

[93] Zhou, J.-J., Theodoulou, F., Sauer, N., Sanders, D., and Miller, A. J., 1997, “A Kinetic Model with Ordered Cytoplasmic Dissociation for SUC1, an *Arabidopsis* H^+/Sucrose Cotransporter Expressed in *Xenopus* Oocytes”, J. Membrane Biol., **159**, pp. 113 – 125.

[94] West, E. S., Todd, W. R., Mason, H. S., and Van Bruggen, J. T., 1966, *Textbook of Biochemistry, 4th Ed.*, Macmillan Company, New York.