SANDIA REPORT
SAND2001-3780
Unlimited Release
Printed November 2001

Applications of Transport/Reaction Codes to Problems in Cell Modeling

Shawn A. Means, Mark D. Rintoul and John N. Shadid

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia is a multiprogram laboratory operated by Sandia Corporation,
a Lockheed Martin Company, for the United States Department of
Energy under Contract DE-AC04-94AL85000.

Sandia National Laboratories

SAND2001-3780
Unlimited Release
Printed November 2001

Applications of Transport/Reaction Codes to Problems in Cell Modeling

Shawn A. Means
Computational Materials and Molecular Biology

Mark D. Rintoul
New Initiatives

John N. Shadid
Computational Sciences

Sandia National Laboratories
P.O. Box 5800
Albuqeruque, NM 87185-0316

Abstract

We demonstrate two specific examples that show how our exiting capabilities in solving large systems of partial differential equations associated with transport / reaction systems can be easily applied to outstanding problems in computational biology. First, we examine a three-dimensional model for calcium wave propagation in a *Xenopus Laevis* frog egg and verify that a proposed model for the distribution of calcium release sites agrees with experimental results as a function of both space and time. Next, we create a model of the neuron's terminus based on experimental observations and show that the sodium-calcium exchanger is not the route of sodium's modulation of neurotransmitter release. These state-of-the-art simulations were performed on massively parallel platforms and required almost no modification of existing Sandia codes.

Acknowledgements

We would like to acknowledge a large number of colleagues who helped make this work possible. First, we would like to thank Jason Shepherd and the CUBIT meshing team for providing us with tools and assistance for generating the meshes used in these computations. This includes Rob Leland who encouraged his already busy staff to help us with this project. The authors would also like to acknowledge Gary Hennigan, Andrew Salinger and Roger Pawlowski of the MPSalsa transport / reaction system modeling team for contributions in developing the MPSalsa code and applications to cell modeling. We also would like to acknowledge the contributions of our external collaborations with John Wagner and Boris Slepchenko (University of Connecticut Health Center) regarding the Xenopus egg simulations, and Pedro Embid and Eric Toolson (University of New Mexico Mathematics and Biology Departments, respectively) regarding the neuron terminus model. Finally, we are grateful for the funding for this project through Gerry Yonas and the LDRD office, and for the help of Mark Derzon in securing this funding. Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under Contract DE-ALO4-94AL8500.

Contents

Figures

1. Introduction

The cell is the fundamental unit of all life. Cells are simple structures bound by membranes and generally filled with an aqueous solution that contains various chemicals. The most remarkable (and in a sense, defining) property of cells is their ability to grow and reproduce. As the basic unit of living things, almost all biological processes related to the health of an organism must be studied at the cellular level. For this reason, much of the scientific community is interested in having a comprehensive computational model of various types of cells in order to better understand how cells behave with respect to different environments.

Unfortunately, most typical cells are impossible to model at the most fundamental level using today's computational power. Each cell can contain billions of proteins (representing thousands of different species of protein), billions of base pairs of nucleic acid, millions of RNA molecules, millions of ribosomes, and dozens of mitochondria that are themselves almost as complex as cells. Besides these major important cellular constituents, there are countless other organic molecules and ions that are used as part of the building and signaling processes within the cell. Even if the individual cellular components are considered at a whole and not at the molecular level, a truly comprehensive cell model is still probably decades away from being computationally feasible.

Despite the fact that a comprehensive cell model is presently not possible, there are still many ways in which slightly more limited cell models can provide a useful tool for performing simulations. One way of doing this is by just simulating the individual molecules of interest in a specific reaction. This is useful when considering the spatial and temporal effects in a reaction that involves a small number of reacting components in a specified geometry. However, with this type of simulation, each individual reactant must be followed (limiting the total number of reacting molecules to perhaps a million at most) and the effects of all other molecules of the systems must be ignored. Another type of cell simulation involves treating all of the species in the cells as nodes on a graph and all the reactions as edges. This model is intended to use experimental data to build initial networks and test the effect of changing the concentrations of various components. This model has the problem that it is often difficult to experimentally determine the given concentrations and reaction rates, and it does not take into account the geometry of the cell.

There is another type of cell model that has gained in popularity due to the relative strengths of its assumptions and practicality of its applications. This model is often referred to as a *continuum* model of a cell. In this model, all of the species of interest are modeled not as individual objects but as a concentration that varies as a function of space and time. Their interactions are handled by means of partial differential equations (generally known as *diffusion/reaction* equations) that specify the result of having certain concentrations of various interacting species together in a given place at a given time.

One of the primary advantages to this method is that now the large number of individual molecules actually benefits the method since there is an assumption that there will be a relatively continuous distributions of reactants throughout the volume of interest. Another advantage to this type of model is that it reflects many problems of interest in real biological applications, such as activity in neuronal cells and cardiac cells. It is important to note that while the initial concentrations are necessary as input parameters to the model, details of the individual reactions do not necessarily have to be experimentally determined, but can be constructed given knowledge about the biochemistry of the reactants.

The disadvantages associated with continuum cell modeling are primarily related to the computational challenges. There will generally be many coupled partial differential equations that describe the behavior of the system as a function of time and space. Also, because this model is a spatial model, one must incorporate the effects of the cell geometry into the solution. As a general rule, solving the resulting equations is a very challenging computational problem that requires specialized algorithms and large amounts of computing power. However, Sandia has already invested many man-years of effort developing both the algorithms and the actual computer programs to solve these problems on massively parallel computers. This is due to the need to models the flow of interacting fluids in stockpile-related simulations. With these capabilities already in hand, we were able to apply these methods quickly and easily to various unsolved problems in biology in order to explain observations that had previously been not clearly understood.

We will first present here an explanation of the algorithms used to solve the equations that arise in our models, and then apply them to questions related to calcium waves in egg fertilization and the question of the role of sodium on intracellular calcium and its role in neurotransmitter release.

2. Overview of MPSalsa Solution for Diffusion/Reaction Systems

2.1. Introduction

This section briefly overviews the numerical solution methodology that is currently used in MPSalsa to approximate the solution of the multi-species diffusion/reaction equations that are used in the continuum biological cell simualtions. MPSalsa [12] is a general parallel transport/reaction solver that is used to solve the governing transport/reaction PDEs describing fluid flow, thermal energy transfer, mass transfer and non-equilibrium chemical reactions in complex engineering domains. In the current study we take advantage of the general framework and limit the transport mechanisms that are included to only a multi-species diffusion transport by mass fraction gradients as described by Fick's law.

2.1.1. Brief Overview of the Galerkin Finite Element Formulation

The governing PDEs for multi-componet diffusion mass transfer and non-equilibrium chemical reactions are presented in Table 1 in residual form. This residual definition is used in the subsequent brief discussion of the Galerkin FE formulation. The continuous problem, defined by the transport / reaction equations, is approximated by a Galerkin FE formulation. The resulting weak form of the equations are shown in Table 2.

Table 1. Governing Diffusion / Reaction PDE s

Species Mass Fraction for Species k	$R_{Y_k} = [\frac{\partial Y_k}{\partial t} - \nabla \bullet D_k \nabla Y_k - W_k \dot{\omega}_k] \qquad k = 1,2,..,N$

$$F_{Y_k} = \int_{\Omega} \Phi[\frac{\partial Y_k}{\partial t} - \nabla \bullet D_k \nabla Y_k - W_k \dot{\omega}_k] d\Omega$$

Table 2. Galerkin Formulation of Diffusion / Reaction PDEs

Species Mass Fraction for Species k	

2.1.2. Discrete Equations; Interpolation Functions and Quadrature Rules

Within each element the species mass fractions are approximated by the expansion

$$Y_k(\mathbf{x},t) = \sum_{J=1}^{N_{nodes}} (\hat{Y}_k)_J(t)\Phi_J(\mathbf{x})$$

where $\Phi_J(\mathbf{x})$ is the standard polynomial finite element basis function associated with the Jth global node and N_{nodes} is the total number of global nodes in the domain.

Thermodynamic and transport properties, as well as volumetric source terms, are interpolated from their nodal values using the finite element shape functions. Evaluation of volumetric integrals is performed by standard Gaussian quadrature. For quadrilateral and hexahedral elements, two-point quadrature (in each dimension) is used with linear basis functions, while three-point quadrature is used for quadratic interpolated elements. For example, for tri-linear hexahedral elements, eight Gaussian quadrature points within an element are used to evaluate its volumetric integrals.

Evaluation of surface integrals is performed by standard Gaussian quadrature on the side of the element. As with the volumetric integrals, two-point quadrature (in each direction) is used with linear shape functions, while three-point quadrature is used with quadratic shape functions. For example, for a three-dimensional problem with linear shape functions, four Gaussian quadrature points located on the side of an element are used to evaluate its surface integrals.

2.2. Solution Procedures

In this section, we present the general procedures used in MPSalsa for the steady state and the time dependent solution of equations that describe the discrete problem. The choice of numerical methods in MPSalsa has been made from the standpoint of robustness, efficiency of implementation on parallel architectures, and the ease of including new solution kernels. The major solution kernels used in MPSalsa are the first- and second-order implicit time integration routines, an inexact Newton procedure and the linear system solvers of the **Aztec** [6] parallel Krylov solver library, developed in conjunction with MPSalsa. Next we give a brief overview of the implementation of the unstructured finite element method on multiple processors, since this aspect underlies much of the discussion and implementation of the solution algorithms for the linear system.

2.2.1. Implementation on Multiple Processors

MPSalsa is designed to solve problems on massively parallel (MP) multiple instruction multiple data (MIMD) computers with distributed memory. For this reason the basic parallelization of the finite element problem is accomplished by a domain partitioning approach. The initial task on an MP computer is to partition the domain among the available processors, where each processor is assigned a subdomain of the original domain. It communicates with its neighboring processors along the boundaries of each subdomain. There are two fundamental ways to partition the FE domain among processors: either element or node assignment. Each method has its own advantages and fundamentally affects the solution strategies and interprocessor communications. Dividing the mesh according to elements quite naturally can lead to an element-by-element (EBE) solution scheme, whereas dividing the mesh according to nodes leads most naturally to a fully-summed distributed matrix solutions. In the EBE case, each element's matrix is stored separately and is not summed with its contributions from neighboring elements. All matrix-vector operations are performed with these dense elemental block matrices and the vector result is obtained only after summing over all elements. This scheme substantially increases the matrix storage requirements and the amount of computation needed relative to fully-summed distributed matrix solution strategies. For example, for 3-D linear hexahedral elements, this method requires approximately 60% more storage and greater than three times as many floating point computations are required for the EBE approach. Although the larger block sizes associated with the EBE approach may yield an increase in the number of operations performed per second, this improved performance is unlikely to compensate for the increased operation count. Because of this, nodal decomposition was chosen in MPSalsa to allow the implementation of computationally efficient, minimum flop algorithms for the matrix-vector multiply kernel. Also, storing the fully summed equations allows the use of robust general preconditioners, such as domain decomposition incomplete factorizations and direct sparse subdomain solvers.

The parallel solution of a particular FE problem proceeds as follows. At the start of the problem, each processor is "assigned" a set of finite element nodes that it "owns." A processor is responsible for forming the residual and the corresponding row in the fully summed distributed matrix for the unknowns at each of its assigned FE nodes. To calculate the residual for unknowns at each assigned node, the processor must perform element integrations over all elements for which it owns at least one element node. To do this the processor requires 1) the local geometry of the element and 2) the value of all unknowns at each of the FE nodes in each element for which it owns at least one node. The required elemental geometry is made available to the processor through the initial partitioning and database distribution part of the algorithm. Here, a broadcast of all information in a serial EXODUS database to all processors is used in MPSalsa. Then, each processor extracts its geometry information form the FE database. In addition to the broadcast algorithm, MPSalsa has the capability to use a parallel FE database for geometry input as well as all parallel I/O. The unstructured interprocessor communication of FE unknowns is handled by an Aztec routine that exchanges the necessary interprocessor information.

Figure 1, which depicts a partitioning scheme of an unstructured mesh, graphically represents the above concepts. An unstructured mesh is divided into four regions by assigning ownership of the nodes. Nodes in each processor are classified as "border" and "internal" nodes, at which border and internal unknowns, respectively, are defined. Border unknowns are those unknowns whose values must be communicated to neighboring processors so they may complete their element integrations; the remaining "owned" unknowns on a processor are designated as internal unknowns. Those unknowns required for a processor's element integrations but assigned to a neighboring processor are stored in the local solution vector and designated as "external" unknowns. Interprocessor communication occurs when an owning processor communicates the values of its border unknowns to a neighboring processor to update the value of the neighboring processor's corresponding external unknowns. Figure 1 demonstrates how Processor 0 would classify the nodes in the internal, border, and external categories. Processor 0 has three neighboring processors. During the interprocessor communication phase, it sends each neighboring processor a message containing the values of each border unknown that the neighboring processor needs. The value of each border unknown may be needed by more than one processor, as it may appear in the external node lists of more than one of the neighboring processors. Processor 0 also receives a message from each of its surrounding processors containing the values of its external unknowns. Processor 0 doesn't have to know about unknowns defined at elemental nodes, which are not internal, boarder or external nodes for this processor.

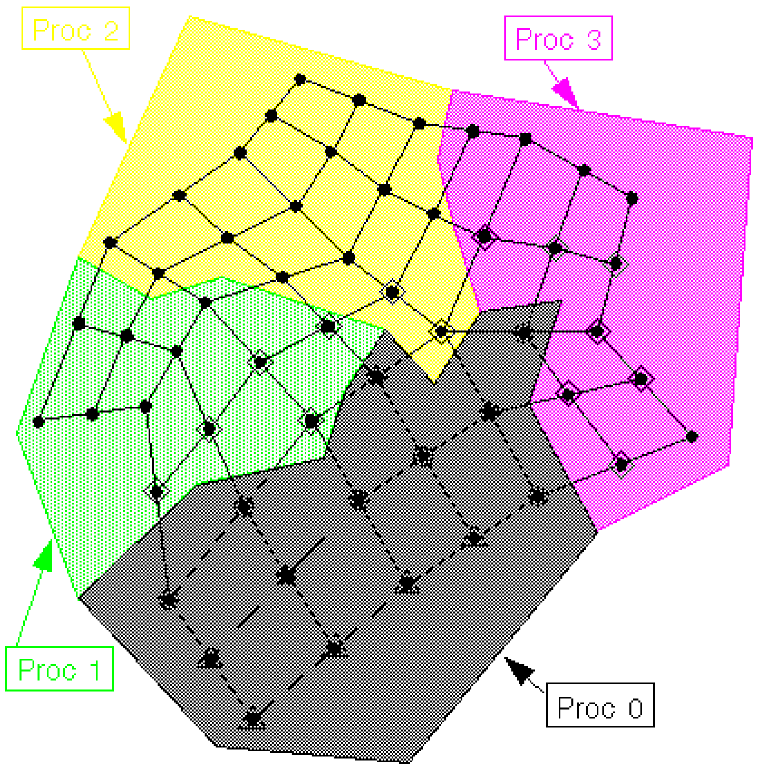


Figure 1. Division of the modes of an element amongst the processors, and the further differentiation of the nodes into interior (△), border (○), and external (◇) categories on Processor 0

2.2.2. Distributed Vectors

On each processor, a solution vector is stored which corresponds to the internal, border, and external unknowns defined on that processor. The solution vector is reordered locally so that local internal unknowns appear first, border unknowns appear second, and external unknowns, grouped by the owning neighboring processor, appear last. A local-to-global mapping vector is maintained, so that the global solution vector may be regenerated using "fan-in" operations. This local reordering scheme minimizes the gather/scatter operations involved in the interprocessor communication step. Only a gather operation at the originating processor to gather all of the border unknowns needed by a single neighboring processor into a contiguous space in memory is required. This message can then be directly sent to the contiguous space in the destination processor's solution vector corresponding to the external unknowns owned by the originating processor. No scatter operations are needed on the destination processor. Moreover, the communications stencil required for this operation may be calculated once and used over and over again for a static mesh discretization. The communications stencil refers to the content of the message that each processor needs to send to each of its neighboring processors and the length of the return message containing the external unknown values from each neighboring processor.

2.2.3. Distributed Matrices

MPSalsa stores the Jacobian matrix in a distributed version of the Variable Block Row (VBR) sparse matrix format. Each processor is responsible for storing rows of the Jacobian corresponding to its unknowns. Once a specific partition and assignment of the unknowns to internal, border, and external sets has been defined and the local solution vector has been reordered a distributed VBR sparse matrix is constructed. Each row of the Jacobian may include column entries corresponding to internal, border, and external unknowns defined on that processor. During the matrix-vector multiply kernel of the Krylov subspace iterative methods, each processor is responsible for carrying this out for its rows. This necessitates an interprocessor communication step wherein all external entries in the vector are updated with values from the neighboring unknowns, before the start of the operation. Calculation of matrix-vector products on rows corresponding to the internal unknowns requires no external node values and can therefore proceed simultaneously with the communication step.

Much of MPSalsa's parallel implementation is designed with the goal of maximizing the speed of this matrix-vector multiplication, which essentially requires minimizing the time needed to perform the communications. This subsection has described several strategies employed by MPSalsa to achieve rapid interprocessor communications: reordering of the solution vector to minimize work involved with the communications step, the pre-setup of the communications stencil, and the ability to do calculations during the communications step. The other basic algorithmic aspect of highly efficient unstructured communication is the partitioning of the FE mesh in a way that reduces the total communication volume and message start-ups while achieving load balance over all of the processors. To do this, MPSalsa currently uses a static partitioning generated by **Chaco** [4], a general graph partitioning code that was developed in conjunction with MPSalsa. **Chaco** supports a variety of new and established graph partitioning heuristics, such as spectral techniques, geometric methods, multilevel algorithms and the Kernighan-Lin method. All of these approaches may be applied in bisection, quadrisection, or octasection mode to recursively partition general graphs for mapping onto hypercube and mesh architectures of arbitrary size. Using these techniques, a problem mapping with low communications volume, good load balancing, minimum message start-ups and small amounts of congestion can be generated.

2.3. Numerical methods

The system of diffusion/reaction equations is a system of nonlinear self-adjoint PDEs. The stiffness and the strongly coupled nature of the reaction operators, combined with the elliptic behavior of the pressure for incompressible flows, lead to a natural choice of fully implicit time integration techniques to provide stable time integration.

2.3.1. Transient Solution Methods

The transient time integration methods used in MPSalsa follow closely the development of Gartling [2] in the NACHOS II code and the work of Gresho [3]. Two types of implicit predictor/corrector integrators are used in MPSalsa: Forward/Backward Euler and Adams-Bashforth/Trapezoidal Rule. As discussed above, implicit solution methods are preferred for diffusion-reaction equations. Explicit methods suffer from a number of difficulties, including a) the strong elliptic nature of the diffusion operator, b) severe time step limitation for the stiff terms to maintain stability, c) consistent mass matrices require inversion –thereby defeating the efficiency of the explicit method, d) the reduction of accuracy due to mass lumping to avoid (c). Though computationally expensive, implicit methods are desirable because of their stability and ability to integrate efficiently to steady state solutions for problems where the diffusion operator is important. The implicit time integrators in MPSalsa are based on predictor/corrector methods to improve their accuracy and efficiency. Both integrators may be used with either a constant or dynamic time step selection algorithm. A solution of the resulting nonlinear, algebraic system for each time plane is obtained by the inexact Newton method described in Section 3.4

The time integration procedures above can be used with either a user-defined constant time step or a dynamically controlled time step that is initialized with the user-defined time step size. In general, the *a priori* selection of a time step size can be a very difficult task, especially for stiff reacting flow equations with complex fluid flows. One of the benefits of using the predictor/corrector algorithms is that they provide a rational basis for dynamically selecting the time step size.

The details of time step control algorithm can be found in Gresho et al. [3]. The general formulation of the time step selection process comes from well established ODE and DAE literature (e.g. [5]). By using a comparison of the time truncation errors for two time integration methods of comparable order, a formula can be developed for selecting the next time step, based on a specified user error tolerance

2.3.2. Nonlinear Solvers

In this section, we briefly discuss an implementation of Newton's method that uses approximate iterative solution techniques to solve the sequence of linear problems produce by the Newton linearization scheme. The particular implementation we use follows the work of Eisenstat and Walker [1]. This method differs from standard Newton implementations as follows. First the inexact Newton scheme uses iterative solution techniques rather than direct matrix inversion methods. Second, at each stage of the Newton iteration, the algorithm selects an appropriate level of convergence required for the iterative linear solver. This strategy is used to increase robustness of the nonlinear algorithm and to ensure that the linear equations are not over-solved at early stages of the Newton iteration when the Jacobian matrix is not very accurate. Third, this algorithm requires that at each step of the Newton iteration, the nonlinear residual must decrease. If this condition is not satisfied, a backtracking algorithm decreases the Newton step size and re-evaluates the residual at this new proposed solution. The backtracking algorithm is called recursively until the residual reduction criteria is satisfied and a new approximate solution is obtained. More details on the perfomance of this algorithm can be found in [7,10].

Two separate convergence requirements are enforced for the Newton scheme. The first requires that the ratio of the norm of the current nonlinear residual to the norm of the initial residual be reduced by a preset factor (default: 10^{-2}). The second criterion requires that the Newton correction for any variable be suitably "small" compared to the magnitude of the variable. This criterion is very similar to the ratio used to dynamically control the time step size and is standard in general purpose ODE packages.

This convergence criterion is given by

$$\sqrt{\frac{1}{N_{unk}}\sum_{i=1}^{N_{unk}}\left(\frac{|\Delta V_i|}{\varepsilon_r |V_i| + \varepsilon_a}\right)^2} \leq 1.0$$

This criterion requires the ratio of the Newton correction $|\Delta V_i|$ to be small relative to the variable $|V_i|$ with constant ε_r, and to be small in absolute terms compared to ε_a. This assures that all variables, even variables with small magnitude (e.g., trace species), are considered in determining when to halt the Newton iteration.

2.3.3. Linear Solvers

The linear subproblems generated from the inexact Newton method are solved by preconditioned Krylov methods as implemented in our **Aztec** solver library [6]. The parallel Krylov algorithms implemented in **Aztec** include techniques such as the conjugate gradient methods (CG), the restarted generalized minimal residual [GMRES(k)] and transpose-free quasi-minimal residual techniques for nonsymmetric systems. All Krylov methods rely on a small, well defined set of basic kernel routines. These kernel routines consist of parallel matrix-vector, vector-vector, vector inner-product, and preconditioning operations [9,11,13].

It is well known that the overall performance of Krylov methods can be substantially improved when one uses preconditioning. The basic idea is that instead of solving the system $\mathbf{Ax} = \mathbf{b}$, the system $\mathbf{AM^{-1}y} = \mathbf{b}$ is solved where M^{-1} is an approximation to $\mathbf{A^{-1}}$ and is easily computed, since only matrix-vector products are needed, it is not necessary to explicitly form $\mathbf{AM^{-1}}$ (only software to solve $\mathbf{Mv} = \mathbf{y}$ is needed). We note that the preconditioning described here corresponds to "right" preconditioning and that is that it is also possible to precondition on the "left" (i.e. $\mathbf{M^{-1}A}$). In this paper only right preconditioning is considered as the comparisons are more straightforward. Specifically, when left preconditioning is used the computed residual corresponds to a preconditioned residual. Thus, if convergence is based on the size of the residual, changing the preconditioner effectively changes the convergence criteria.

In our numerical experiments, we use **Aztec** [6] to implement the 1-level Schwarz preconditioning techniques. **Aztec** automatically constructs the overlapping submatrices. While a direct factorization could be used on the subdomains, our experience indicates that this is rarely practical as the storage and time associated with this directed factorization is too high. Instead of solving the submatrix systems exactly we use incomplete factorization technique on each subdomain (processors). In this study we use two specific ILU factorizations. The standard ILU(0) method with no fill-in. As well as the ILUT (fill-in, drop) incomplete factorization, which allows specification of a user-specified fill-in parameter ($fill - in \geq 1.0$) and a drop tolerance. In this nomenclature a fill-in of 1.5 denotes an ILU factor with up to 1.5 times as many nonzeroes as the original matrix. In future studies we will use the **ML** multi-level solver library [14] to implement the coarse grid solve for a 2-level domain decomposition methods that can exhibit optimal convergence characteristics.

3. Calcium Wave Propagation in a *Xenopus Laevis* Frog Egg

One of the classic experimental models in biology is the *Xenopus Laevis* frog egg. The egg cell is approximately 1mm in diameter, making it especially easy to study. At fertilization, a Ca2+ wave travels across the egg at a speed of 5-10 μm/s. This wave can be observed visually under the right experimental conditions and its behavior as a function of time has been well documented. There are a number of features about this Ca^{2+} wave that are worth noting. The wave front is a very sharp and well-defined concave wave (and not convex as expected with simple diffusion), with the wake of the wave containing approximately an order of magnitude more Ca^{2+} than the part of the cell in front of the wave. The wave speed also varies along the wave front, with higher speeds near the edges. This causes the wave to initiate as a crescent shape and propagate in a concave shape until it reaches the edge of the wave. Further, the concentrations of calcium in the egg are maintained at this new, elevated level illustrating the so-called bistability of calcium levels in the *Xenopus* egg. These observations indicate that there is a somewhat complicated mechanism that causes the release of Ca^{2+} into the cell cytoplasm, facilitating not only the characteristic shape of the wave front, but the bistable concentration levels for intracellular calcium.

A possible mechanism modulating the release of Ca^{2+} from internal stores of the egg (i.e., Endoplasmic Reticulum), and hence resulting in the concave wave front, was explored. Based upon the two-dimensional simulations performed by J. Wagner, et. al. [15], we implemented a full three-dimensional simulation of a Calcium wave initiated after a fertilization event. The primary mechanism simulated which releases calcium from said internal stores is the inositol triphosphate receptor calcium channel (IP3R) located on the endoplasmic reticulum.

Evidently, IP3R's are densely distributed in the cortical region of the egg while sparsely distributed in the interior of the egg, and this distribution is reflected in the three-dimensional simulation. We employed a diffusive representation of calcium and a non-diffusive representation of IP3R in a diffusion/reaction system of equations, as well as a non-homogeneous radial distribution of the IP3R, whose depiction is given below.

$$\frac{1}{\beta}\frac{\partial C}{\partial t} = \lambda\left[\left(v_L + \left(\frac{I}{I+d_1}\right)^3\left(\frac{C}{C+d_{act}}\right)^3 h^3\right)(C_{er}-C) - v_P\frac{C^2}{C^2+k_P^2}\right] + D\nabla^2 C$$

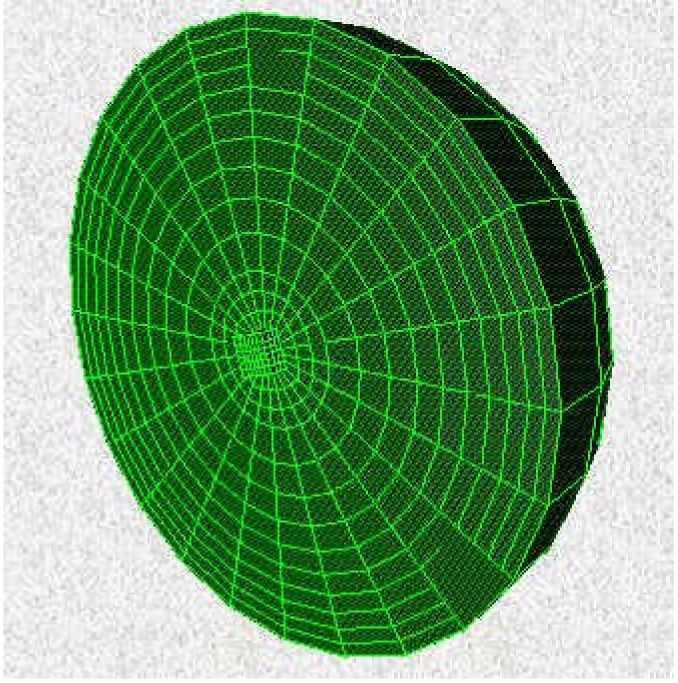
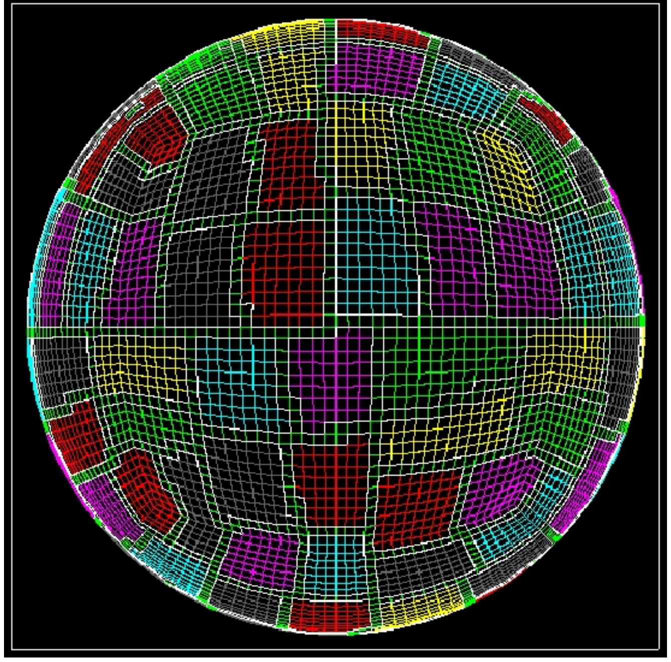
$$\frac{dh}{dt} = \left(d_{inh} - (C+d_{inh})h\right)/\tau_0$$

$$[IP_3]_r = I_s\left(1 + I_h \exp\left(\left(\frac{r}{r_c}-1\right)\Big/ I_w\right)\right)$$

Here C represents Ca^{2+}, h represents the fraction of inactivated IP3R, and $I = [\mathrm{IP}_3]_r$. This distribution of IP3R strongly modulates the shape and progression of the Ca^{2+} wave front, which is a natural result since these are the channels responsible for release of calcium from the internal stores. One can see in the initial results for the simulations this preferential, radial distribution in Figure 3 at time $t = 1$ immediately after the fertilization event.

3.1. Meshing and partitioning of the spherical egg domain

In order to perform the finite element calculation, the egg must be broken down into a mesh. Hexahedral (6 sided) elements were used due to the increased quality of the solution over tetrahedral (4 sided) elements. The egg is represented as a 500 μm radius sphere. Illustrated on the left of Figure 2 is the meshed form of the spherical region. Radially symmetric partitions extend from the outer region toward the surface, with a cube of rectangular elements at the center. This resulted in 371,200 elements and 376,185 of nodes. Additionally, the mesh was partitioned into 512 subregions for solution on 512 processors. The partitioning is illustrated in Figure 2 on the right (surface view).

3.2. Results

We show here the results of the simulation as a function of time. Figures 3-20 illustrate the calcium wave propagating across the surface, and the concave wavefront through the interior. Note the change in scales as time progresses for the color indicators, where Ca^{2+} concentrations are in μM, and the rotation of surface views following the wavefront. At time t = 1s the slide shows the initial distribution of ions as localized on the `left' face of the egg; this is the result of utilizing a distribution function for the protein channels, IP3R, localized similarly as a sub-spherical shell. The data here agrees very well with the experimental data, and strongly suggests that the hypothesis regarding the distribution of IP3 receptors is correct. This was the first fully three-dimensional verification of this hypothesis.

Figure 2. The left shows the meshed spherical domain representing the egg, while the domain partitioning across the 512 processors is shown on the right.

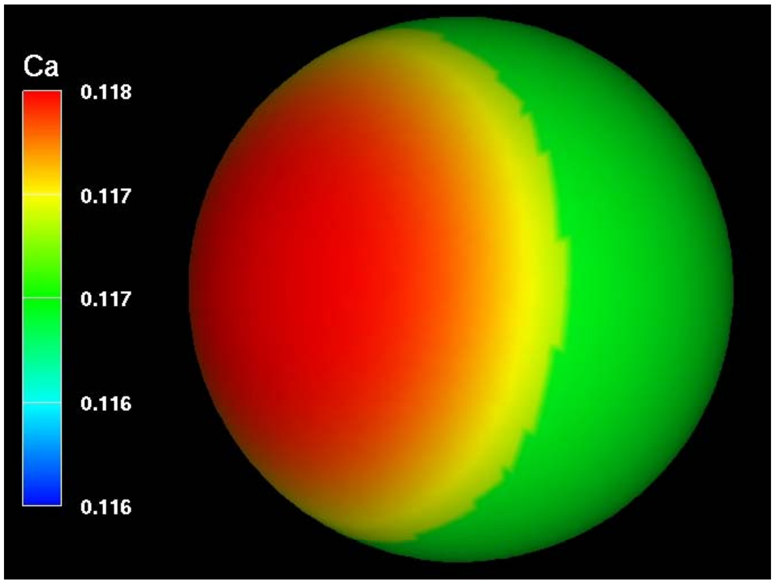


Figure 3. Egg surface, *t* = 1 s.

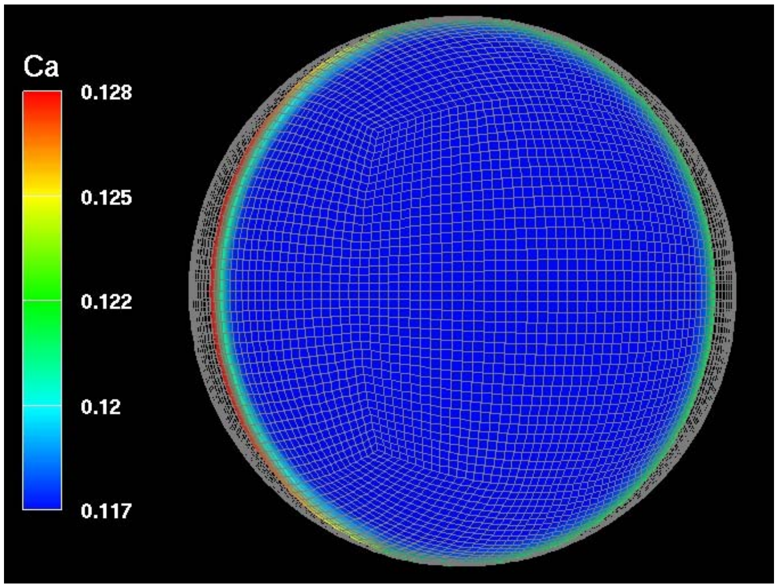


Figure 4. Slice plane along axis of symmetry, *t* = 1 s.

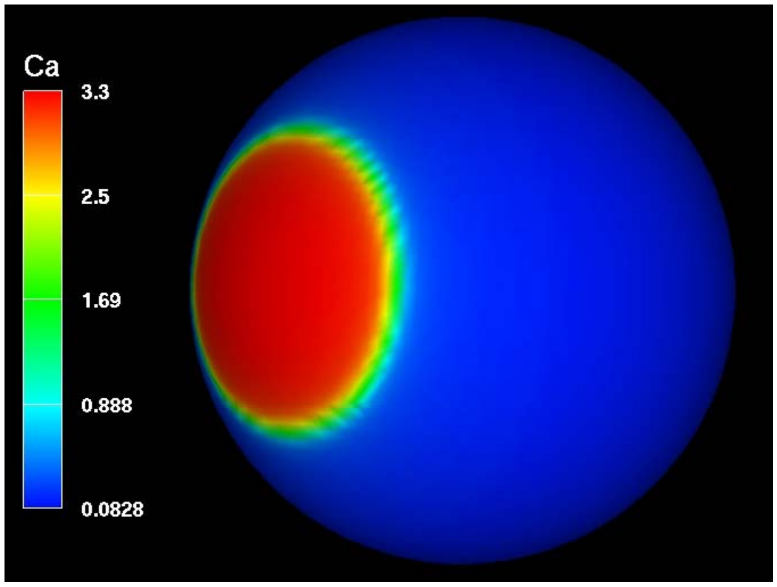


Figure 5. Egg surface, *t* = 20 s.

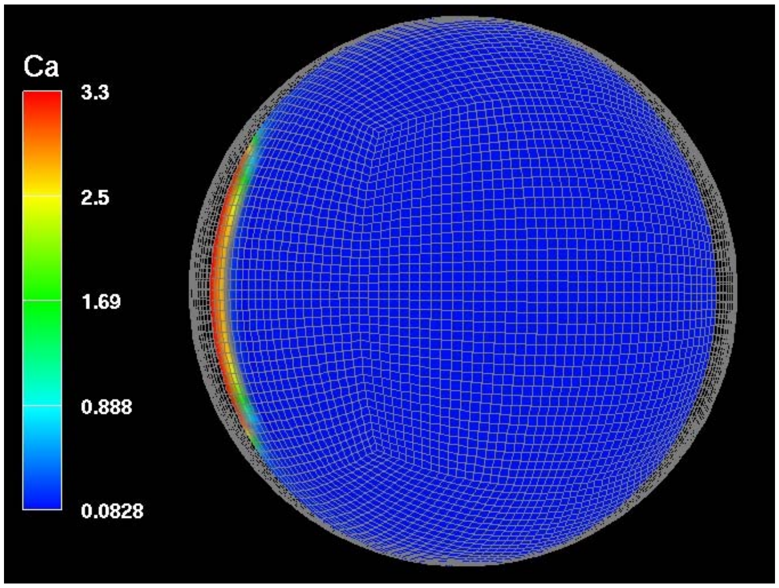


Figure 6. Slice plane along axis of symmetry, *t* = 20 s.

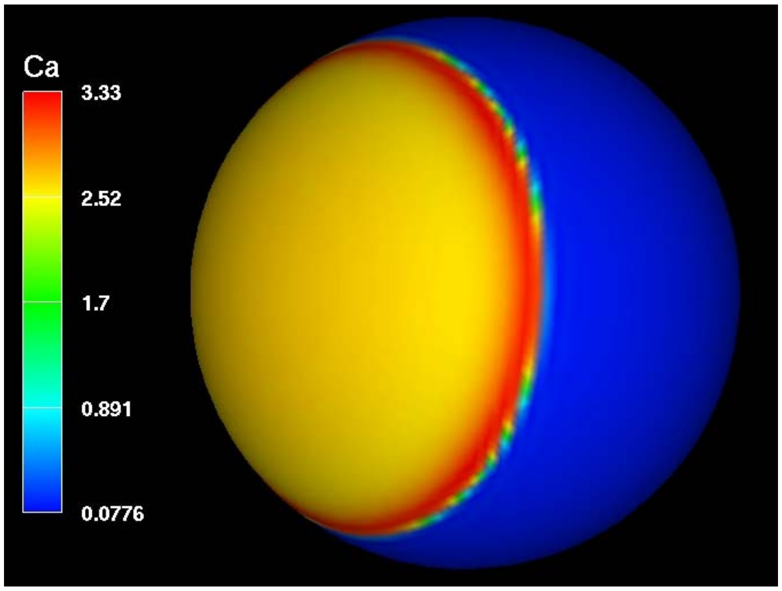


Figure 7. Egg surface, *t* = 40 s.

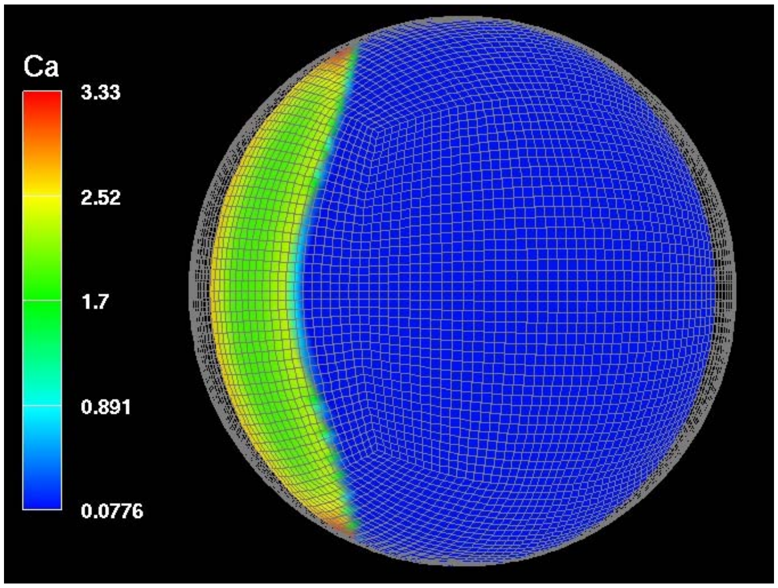


Figure 8. Slice plane along axis of symmetry, *t* = 40 s.

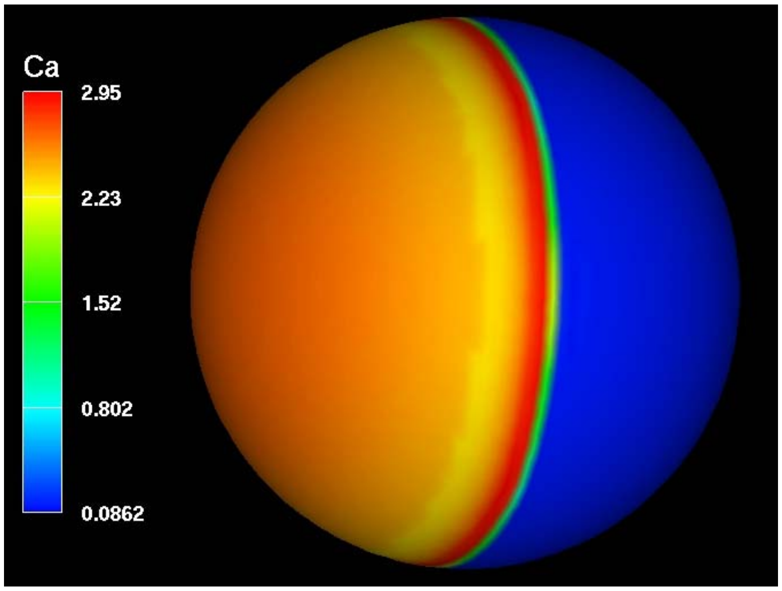


Figure 9. Egg surface, *t* = 60 s.

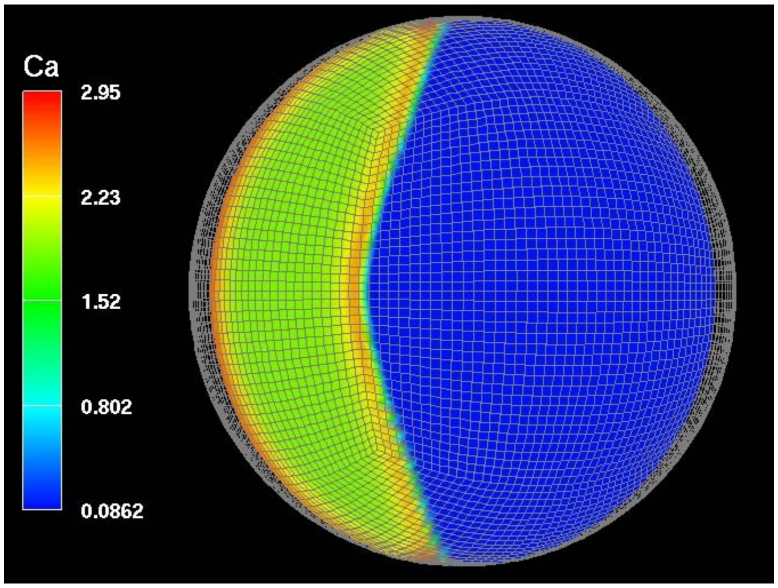


Figure 10. Slice plane along axis of symmetry, *t* = 60 s.

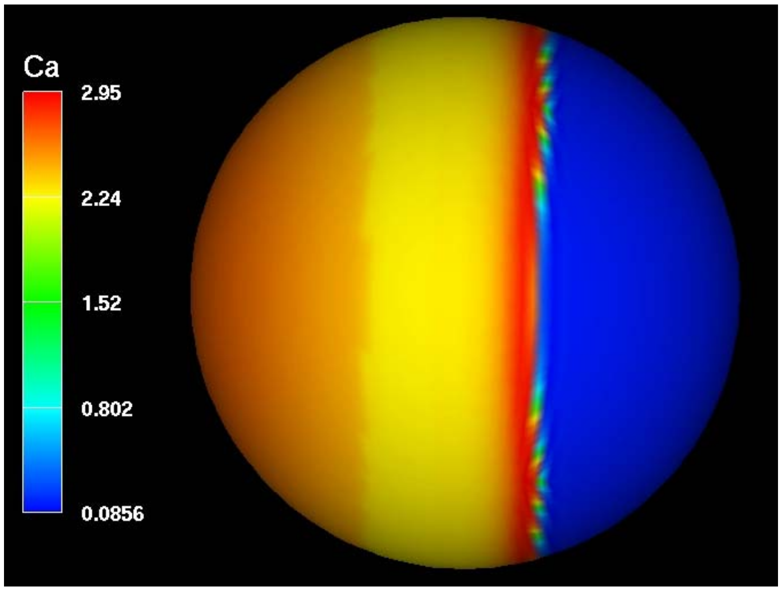


Figure 11. Egg surface, *t* = 80 s.

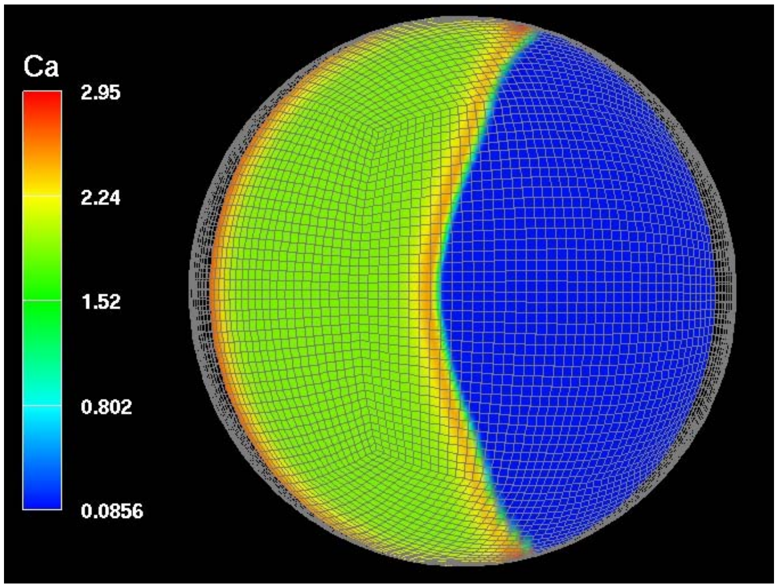


Figure 12. Slice plane along axis of symmetry, *t* = 80 s.

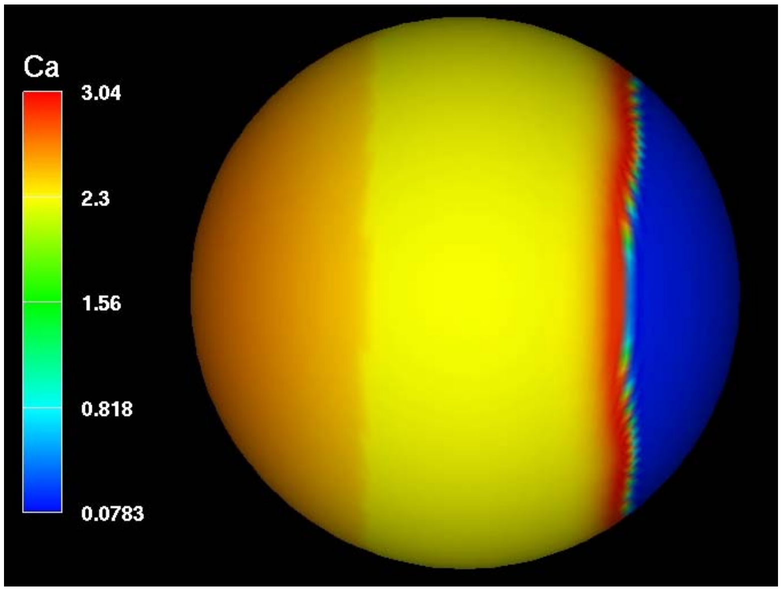


Figure 13. Egg surface, *t* = 100 s.

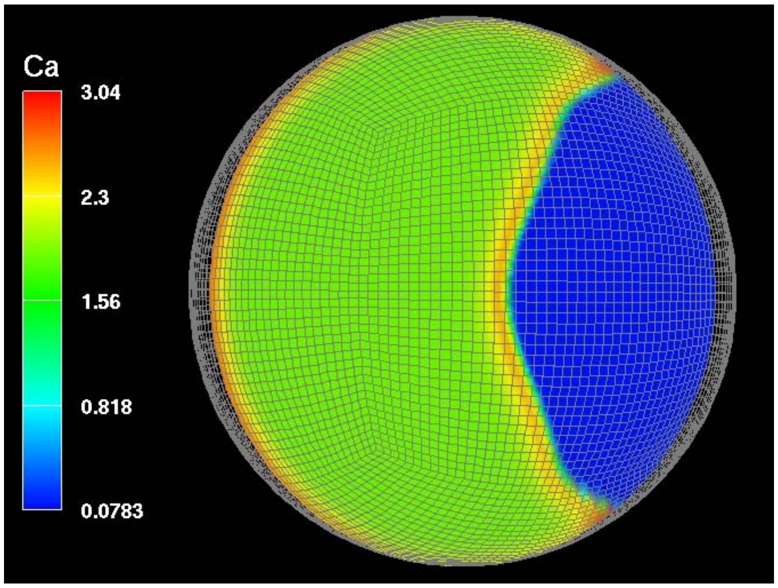


Figure 14. Slice plane along axis of symmetry, *t* = 100 s.

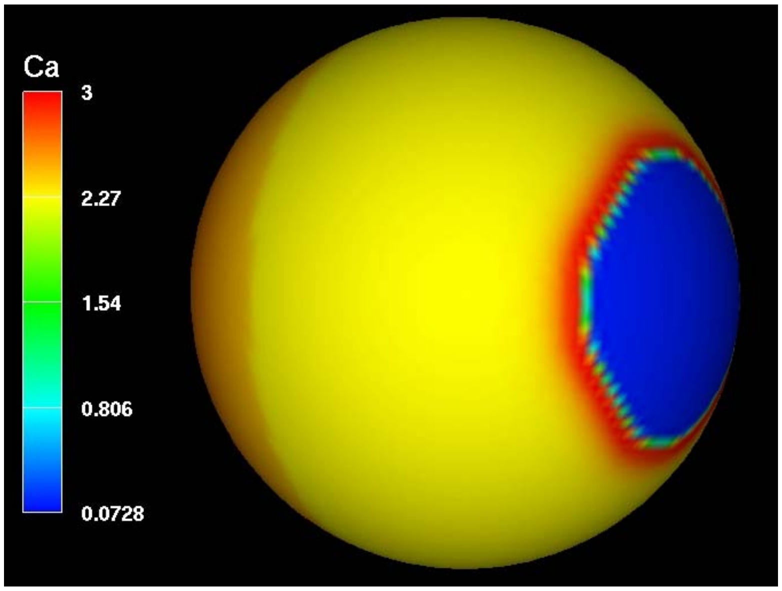


Figure 15. Egg surface, *t* = 120 s.

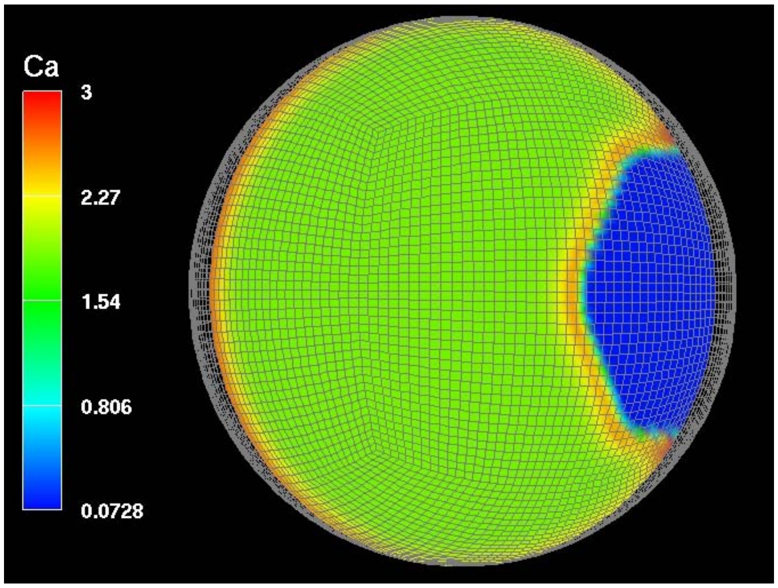


Figure 16. Slice plane along axis of symmetry, *t* = 120 s.

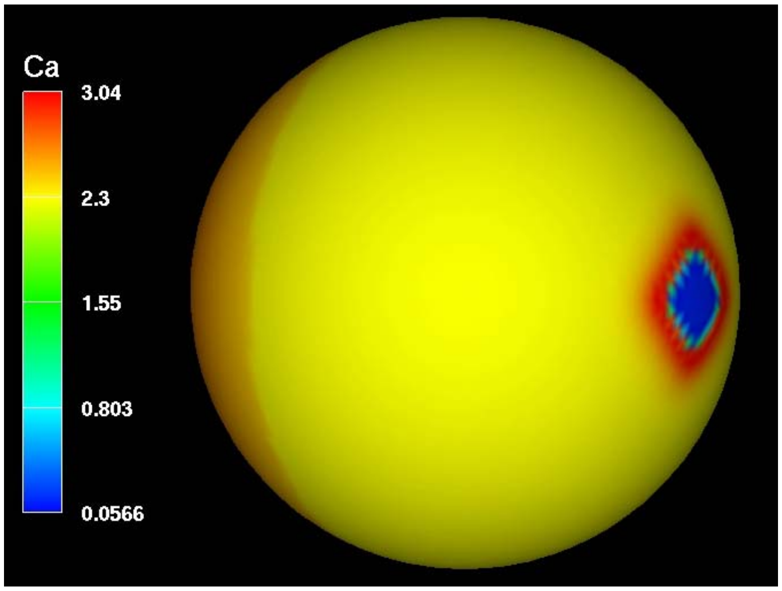


Figure 17. Egg surface, *t* = 140 s.

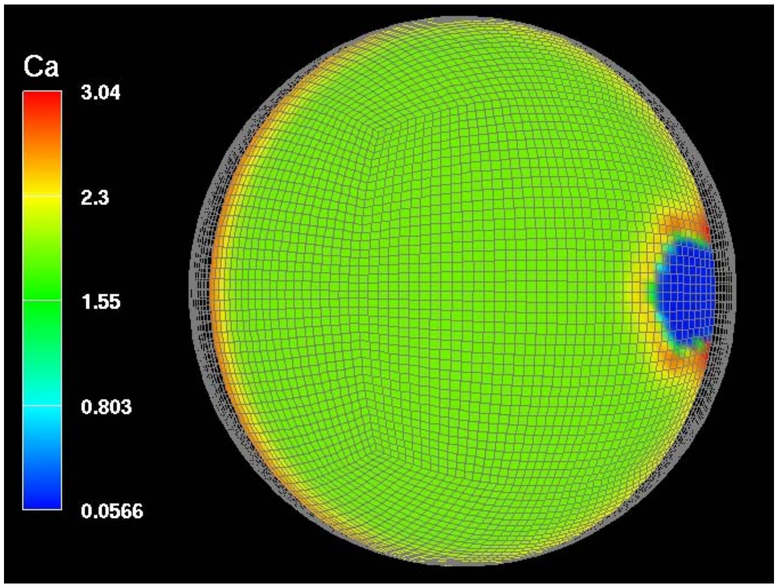


Figure 18. Slice plane along axis of symmetry, *t* = 140 s.

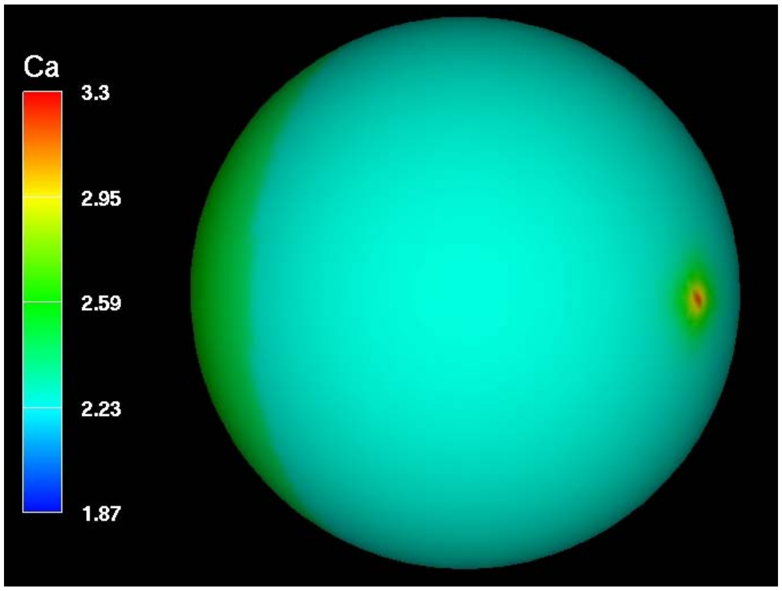


Figure 19. Egg surface, *t* = 150 s.

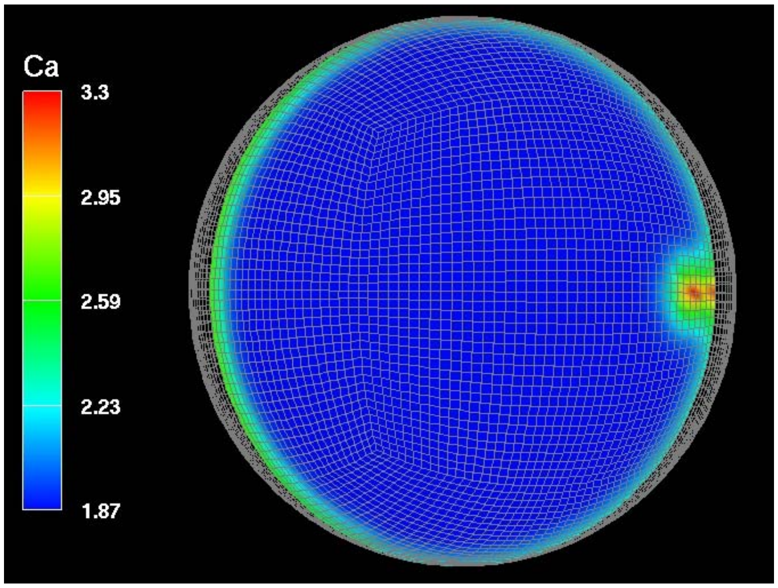


Figure 20. Slice plane along axis of symmetry, *t* = 150 s.

4. The Impact of Sodium on Intracellular Calcium and its Implications for Neurotransmitter Release

4.1. Introduction

Neurotransmitter release propagates signals between neurons, is instrumental in receiving information from sensory organs, and maintains control of muscles. A general description of the process involved in release of the chemicals and proteins known as neurotransmitters is well established. Calcium ions flood the terminus of a neuron and initiate machinery in the cell to release these neurotransmitters.

Sodium is the primary ion responsible for signal propagation down a neuron. After receiving a signal from a previous neuron, a sequence of sodium influxes triggered in a domino effect down its body and axon propagates to the neuronal terminus, where the calcium mediated release occurs. Evidence suggests that sodium indeed has a modulatory control over neurotransmitter release, either directly or indirectly via calcium.

In order to explore that hypothesis, we constructed a model of the neuron's terminus utilizing a continuum, diffusion/reaction representation, including two ionic species (sodium and calcium) and two protein species (mobile and immobile calcium buffer). Included in the model are depictions of voltage-gated ion channels, ionic pumping mechanisms (calcium ATP-ase and the sodium/potassium ATP-ase), and of paramount interest here, the sodium-calcium exchanger. The geometric representation used is given in Figure 21. The details of the differential equations and concentrations used in the model are given in Shawn Means' master's thesis [16], and we provide a brief summary of the equations below.

$$\frac{\partial N}{\partial t} = D_N \nabla^2 N$$

$$\frac{\partial C}{\partial t} = D_C \nabla^2 C + \Phi_C(C, B, S1, S2, S3, S4, I, IC)$$

$$\frac{\partial B}{\partial t} = D_B \nabla^2 B + \Phi_B(C, B, S1)$$

$$\frac{\partial S1}{\partial t} = D_{S1} \nabla^2 S1 + \Phi_{S1}(C, B, S1, S2)$$

$$\frac{\partial S2}{\partial t} = D_{S2} \nabla^2 S2 + \Phi_{S2}(C, S1, S2, S3)$$

$$\frac{\partial S3}{\partial t} = D_{S3} \nabla^2 S3 + \Phi_{S3}(C, S2, S3, S4)$$

$$\frac{\partial S4}{\partial t} = D_{S4} \nabla^2 S4 + \Phi_{S4}(C, S3, S4)$$

$$\frac{\partial I}{\partial t} = \Phi_I(C, I, IC)$$

$$\frac{\partial IC}{\partial t} = -\Phi_I(C, I, IC)$$

In the equations above, *N* represents sodium, *C* calcium, *B* unbound mobile protein calcium buffer, *S1-S4* the site-bound mobile buffer, *I* and *IC* the unbound and bound immobile calcium buffers respectively. Each source term is unique for the individual species, yet coupling between the mobile buffers occurs only with the 'neighboring' site bound species (i.e., *S1* with *B* and *S2*, not *S3* or *S4*).

The primary mechanism of interest, however, is not protein calcium buffering. Of potential impact on intracellular levels of calcium is the sodium-calcium exchanger, as noted earlier. This term is represented as a boundary flux source/sink for calcium. Typically, the exchanger ejects intracellular calcium at the expense of sodium uptake; but reverses operation under certain conditions during the activation of a neuron. A mathematical depiction of this exchanger is presented below, which was provided by considerable prior work performed - see Hilgemann (1988) [17] for a summary.

$$V_m(t) = 120e^{-0.7(t-t_{ap})^2} - 70e^{-(t-(t_{ap}+0.75))^2} - 60$$

$$q_e = \sum_{\mathbf{x}_e \in \varepsilon} \phi_e(N,C,t)\delta^{\varepsilon}(\mathbf{x}-\mathbf{x}_e)$$

$$\phi_e(N,C,t) = K_e\left\lfloor N^3 C_0 \kappa(t) - N_0^3 C \kappa(t)^{-1}\right\rfloor \times H(C-400nM)$$

$$\kappa(t) = e^{V_m(t)\frac{F}{2RT}}$$

Again, *N* indicates sodium, *C* calcium – but internal species. The ‘o’ subscript indicates external species that are held constant for the simulation. A difference of gaussian exponentials is utilized to simulate the variation in membrane voltage during neuronal excitation (function ‘V’), modulating the response of the hyperbolic sine function ϕ as shown above.

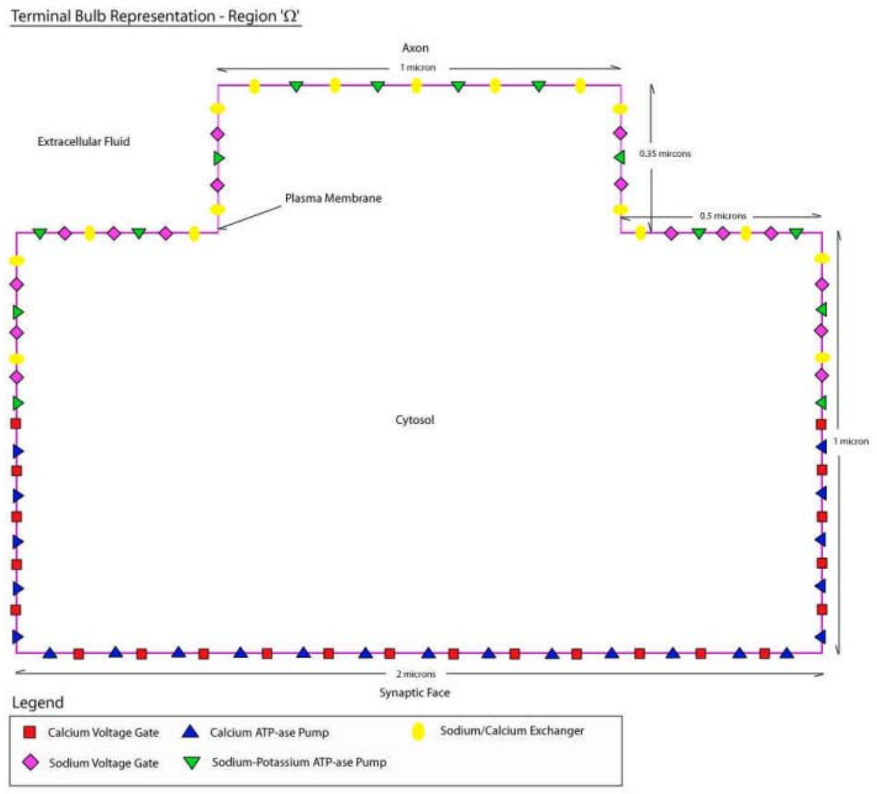


Figure 21. Neuronal Terminal Bulb –model's geometric representation of the neuron's axon terminus using a simple rectangular geometry with dimensions of 2 μm by about 1.35 μm. Distribution of ion channels, pumps and the exchanger noted as in the legend.

4.2. Spatial Discretization (Meshing) of 2D Terminal Bulb Representation and Partition into Subdomains

Meshing the terminal bulb representation (shown in Figure 22) requires many more elements near the boundary then the interior, since dramatic concentration changes occur when the ion channel mouths open and the pumping mechanisms respond. Only buffering protein-Calcium ion reactions occur in the interior. This mesh has 6292 elements and 6962 nodes. The partition of the mesh (above) is for 128 subregions and hence 128 processors.

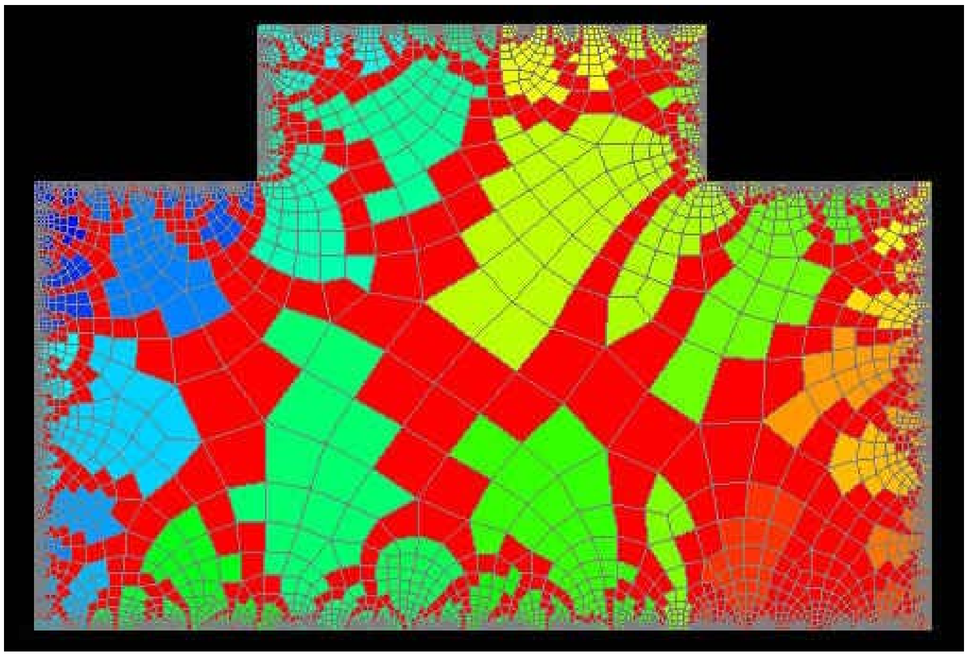


Figure 22. The mesh (small regions) and partitions (different colors) for the terminal bulb.

4.3. Results

Figures 23-26 show respective concentrations for sodium, calcium, mobile and immobile calcium-buffer complexes at time $t = 1.0$ ms (maximal activation event) for a solution run done over $t = 1–55$ ms. Surface plots are initially shown for the ionic species to illustrate the dramatic change in calcium concentration (from baseline of 0.1 μM to peak of ~26 μM, and the relatively nominal change for sodium (from baseline of 10 mM to about 10.1 mM) at maximal neuronal activation. Contour plots for the calcium-buffer complexes show their response to the rapid increase in calcium levels at the synaptic face. Notably, the calcium-bound mobile buffers increase an order of magnitude, whereas the immobile hardly increases at all.

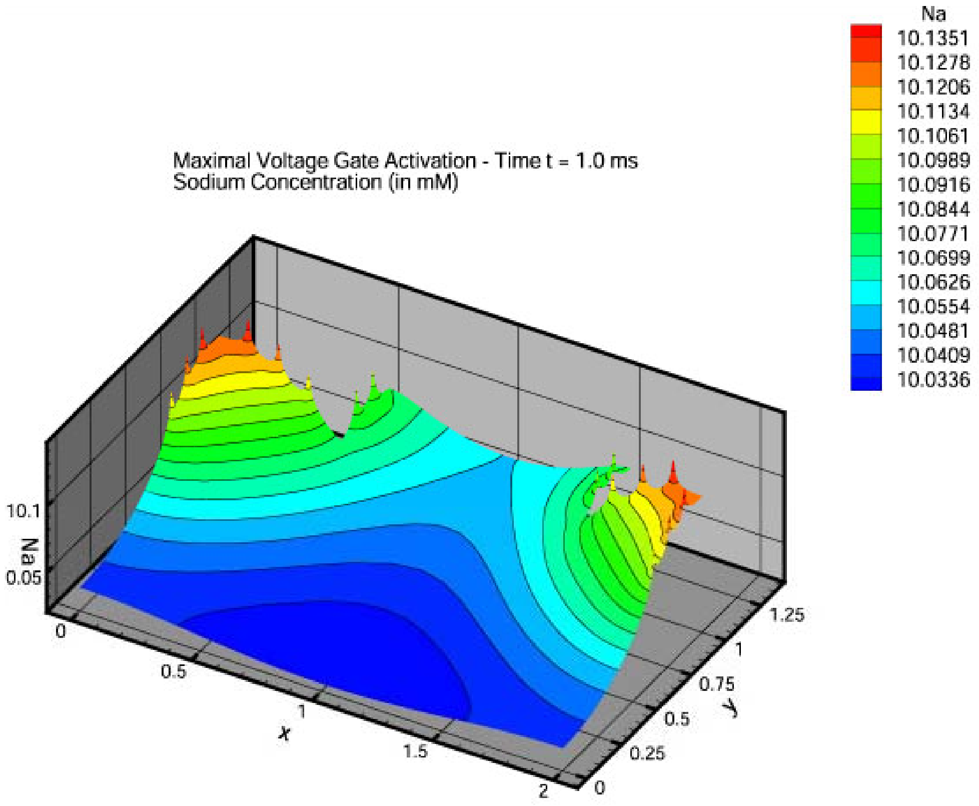


Figure 23

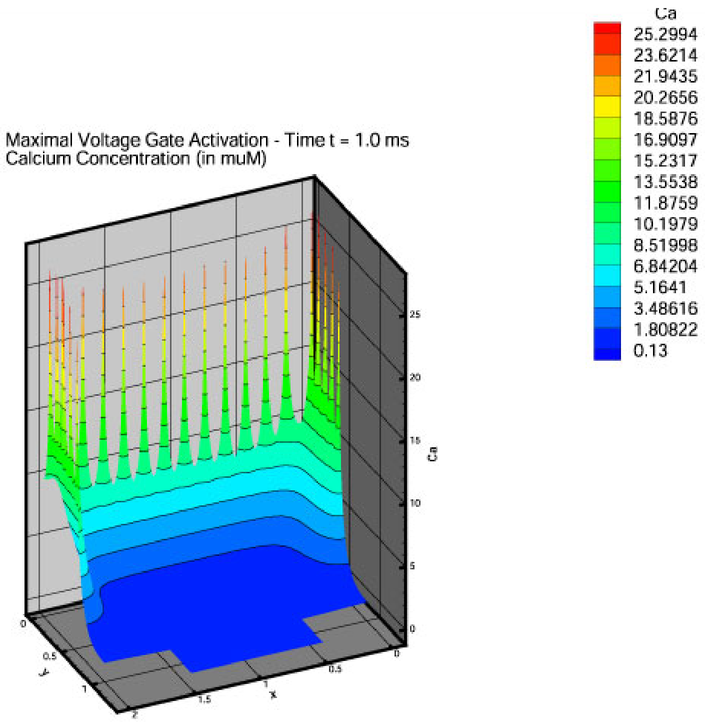


Figure 24

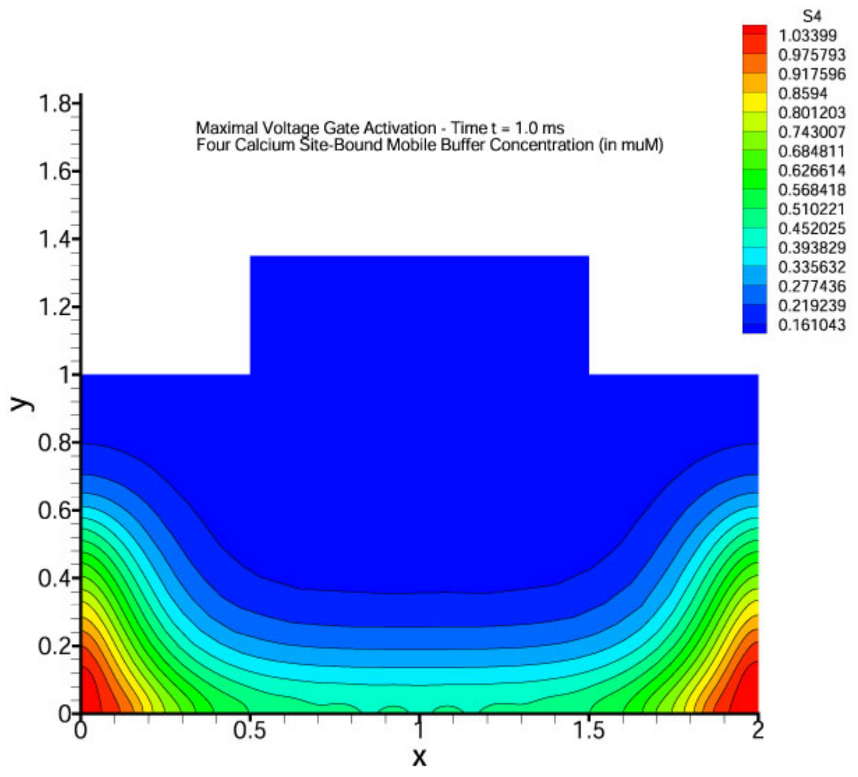


Figure 25

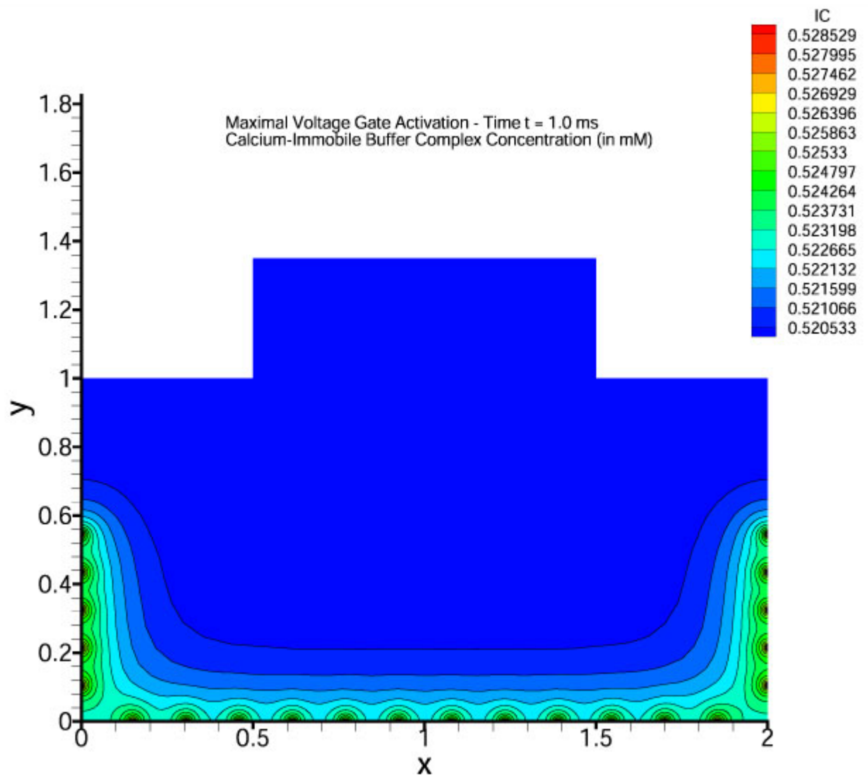


Figure 26

Slides 27-30 show respective concentrations for sodium, calcium and calcium-buffer complexes at time *t* = 13.0 ms where action potential are at maximum effect during a third stimulation event, and activation of ion-channels and depolarization occurs at this time step. Sodium ion channels dominate the distributions toward the axonal side, yet the maximum levels are still are mere 0.3 mM above baseline. Maximal calcium levels are elevated a bit more than the single event (compare 25.3 µM to 25.6) due to the activity of protein buffers, yet the minimal levels are fully an order of magnitude greater than baseline. Consequently, we also see an increase in the levels of calcium-buffer complexes as well, with however, still a minor impact on the immobile species.

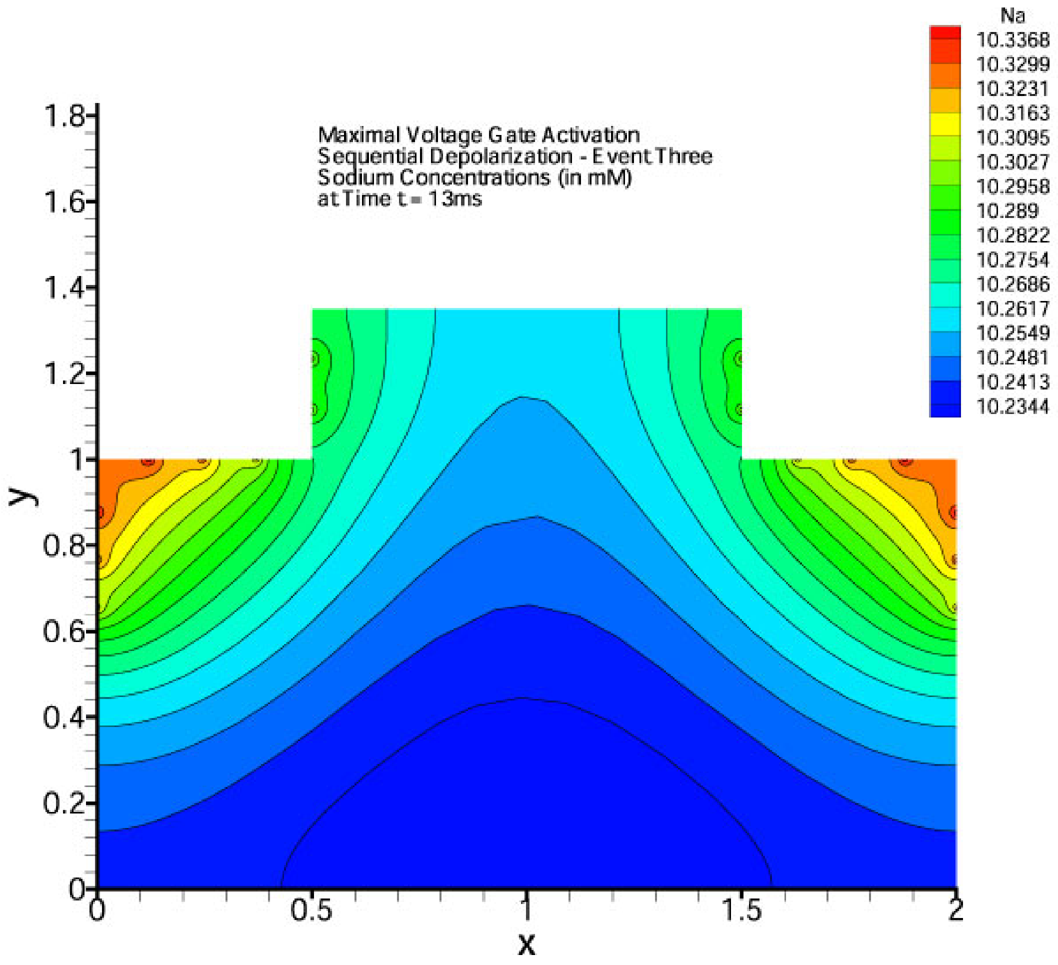


Figure 27

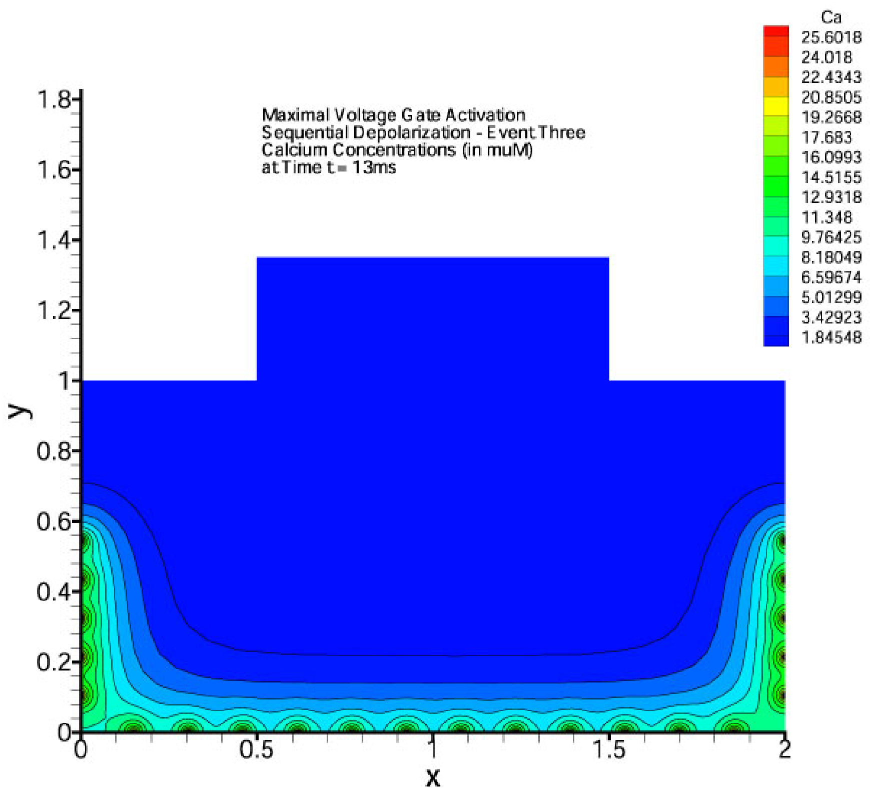


Figure 28

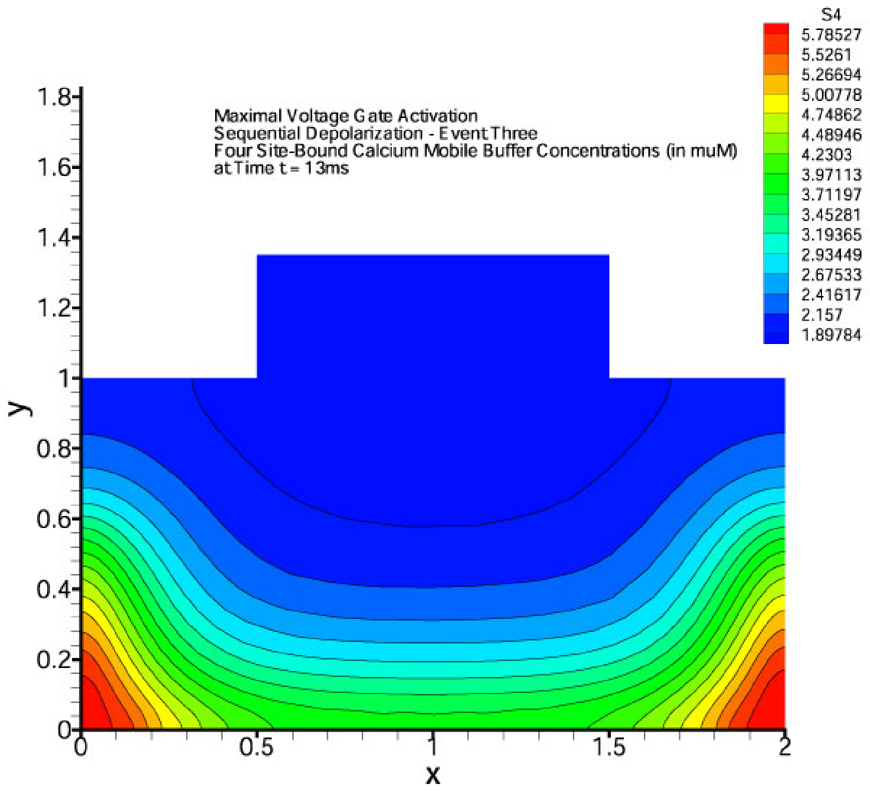


Figure 29

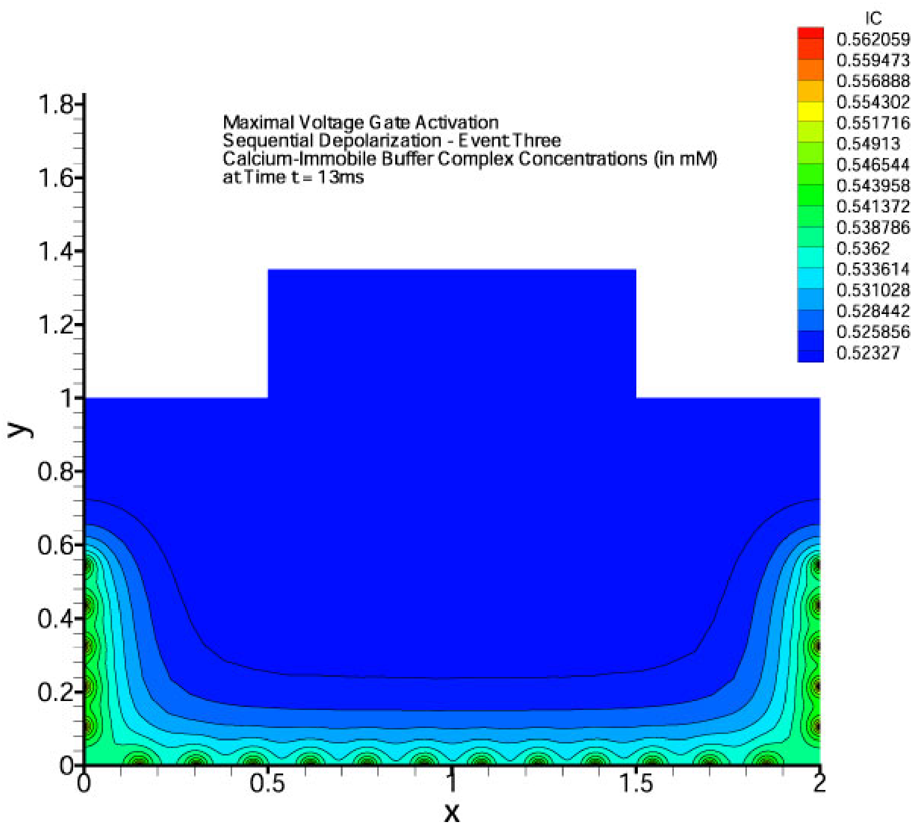


Figure 30

Slides 31-34 show respective concentrations for sodium, calcium and calcium-buffer complexes at time *t* = 55.0 ms, during the tenth activation event. Sodium levels elevate a mere 1 mM or so over baseline, whereas maximal calcium rises to a little more than 26 μM – still about the same as during a single event. Minimal calcium levels are still quite a bit higher than at baseline, around 2 μM instead of the initial 0.13 μM. Mobile buffer-calcium complex continues to rise, although saturation is not quite evident at this point (initial unbound buffers set at 360 μM). The immobile buffer-calcium complex remains relatively unaffected – an increase of only 1 mM overall after ten stimulations.

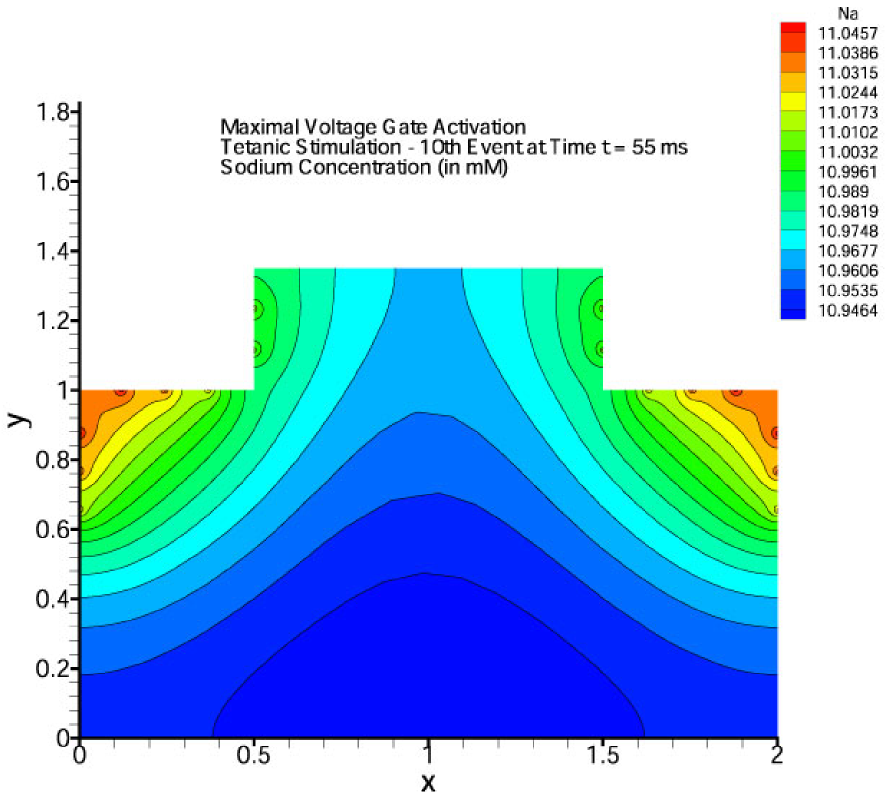


Figure 31

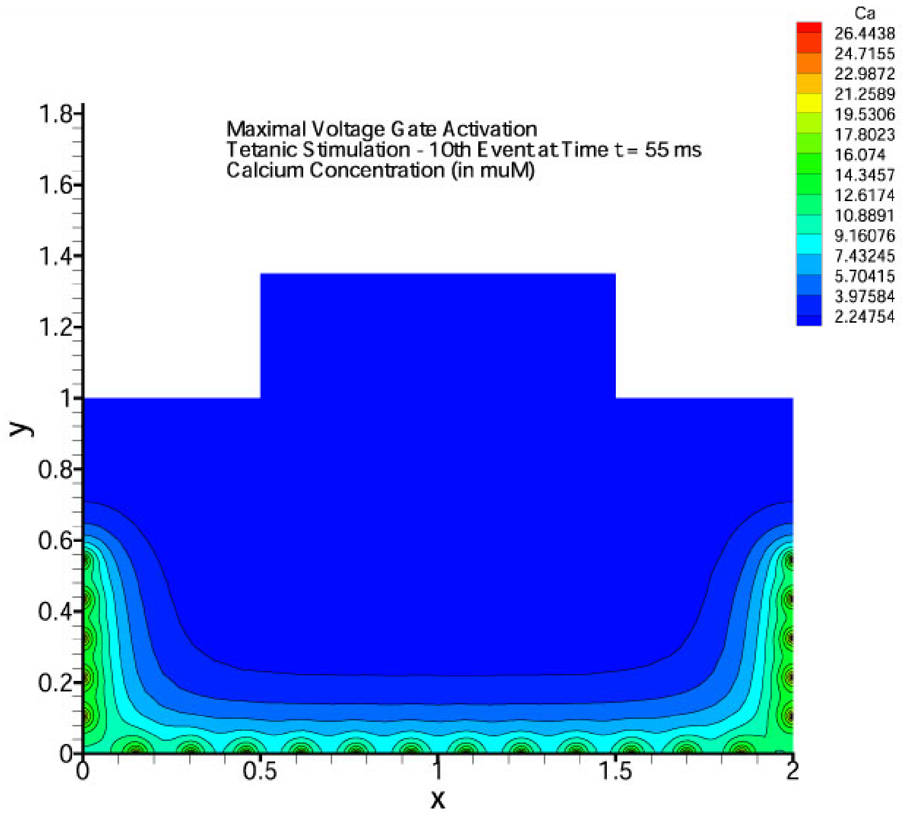


Figure 32

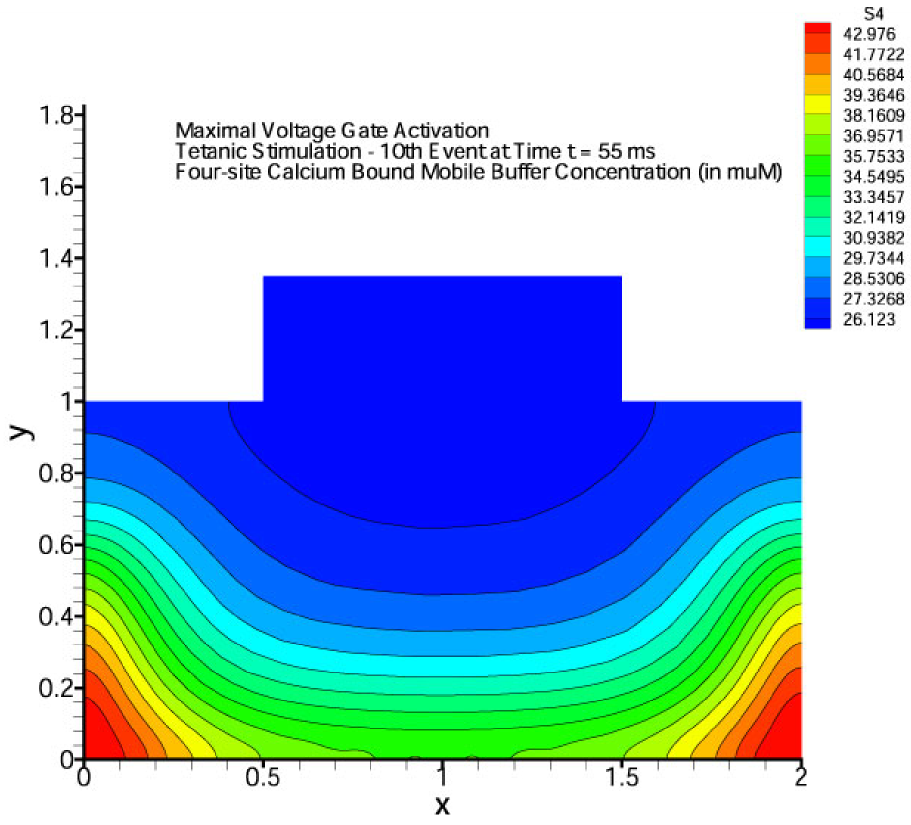


Figure 33

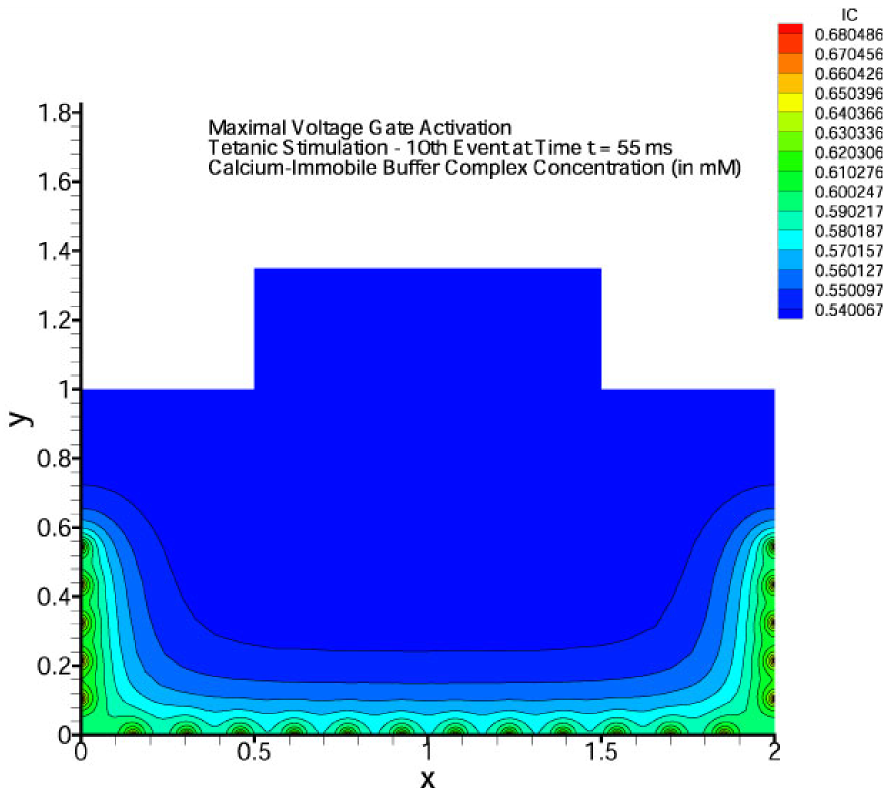


Figure 34

Figures 35 and 36 show the initial recovery of the neuron after the ten stimulation sequence for both sodium and calcium. The activity of the sodium-calcium exchanger is evident in both the plots – slight increases for sodium at the upper end of the region and drops in concentrations for calcium as expected.

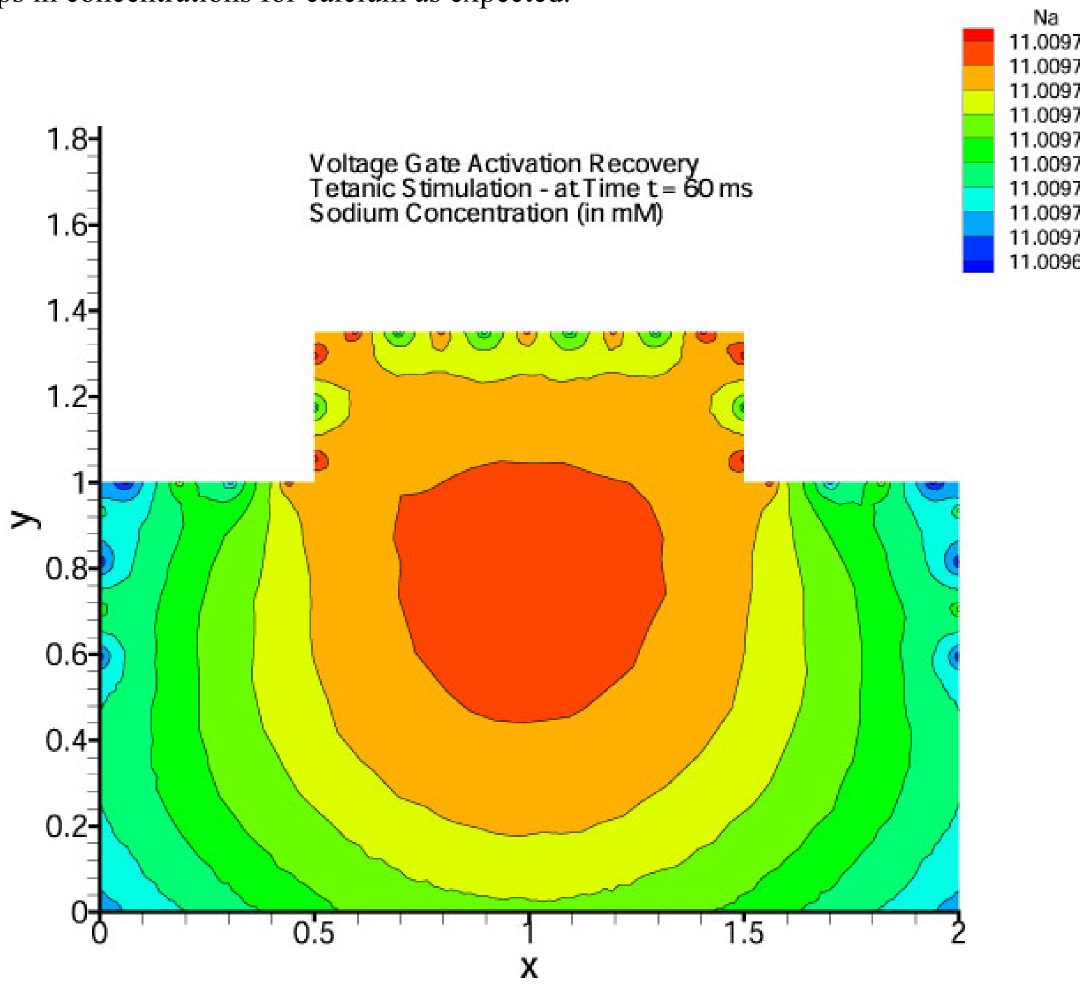


Figure 35

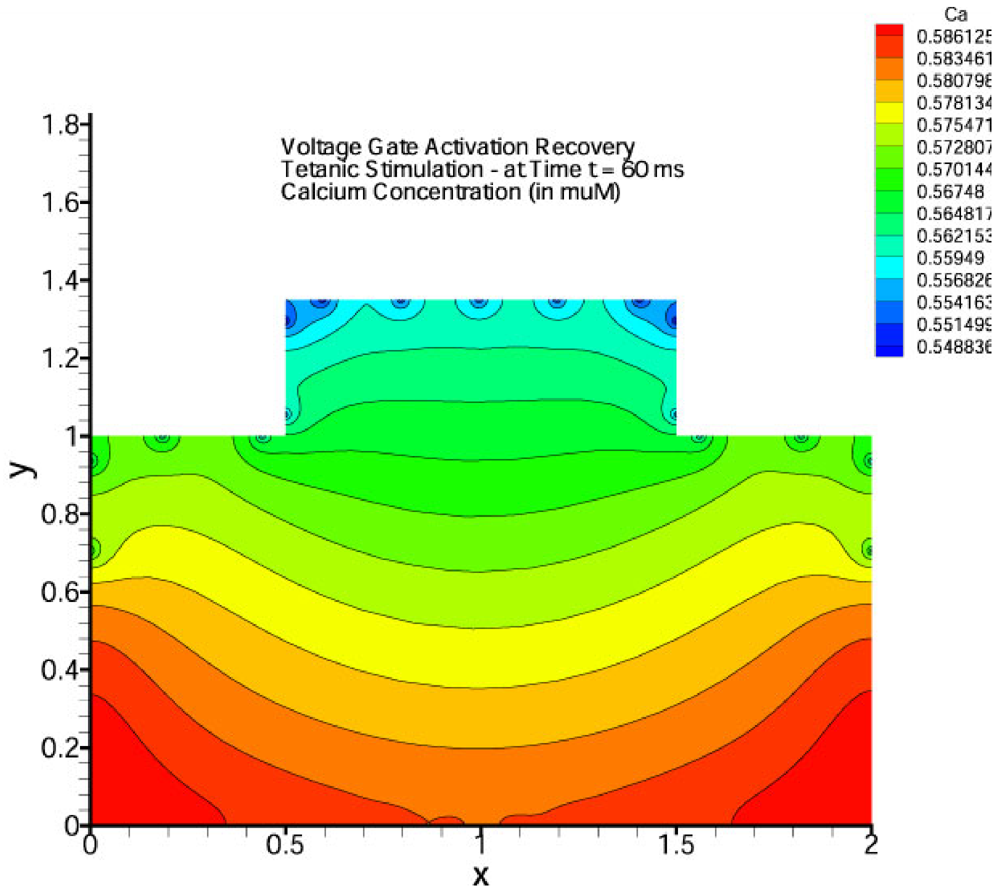


Figure 36

It is the impact of the sodium-calcium exchanger on calcium levels at the neurotransmitter release sites that are of interest here, so we present the difference plot for the simulations both with the exchanger enabled and disabled. Clearly, the only impact of the exchanger is to decrease calcium levels – see Figure 37.

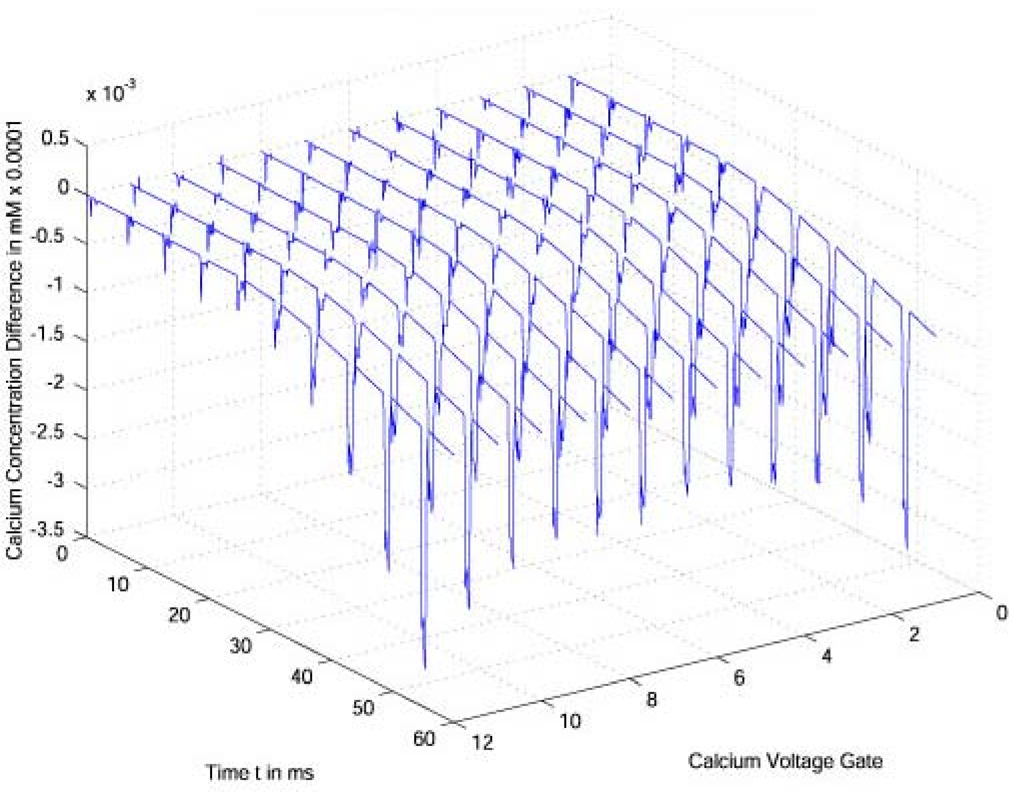


Figure 37. Difference in Calcium Concentrations at Ion-Channel Mouths

4.4. Summary of the Neuronal Terminal Bulb Simulation

The sodium-calcium exchanger usually ejects calcium ions at the expense of sodium uptake. Under certain conditions, this exchanger reverses operation, and hence may increase intracellular calcium levels. Simulations of the neuronal terminus indicate that given the increases in calcium levels during stimulation, the intracellular concentrations of sodium are insufficient to induce reversal of the exchanger. Only reduction in calcium concentrations were evident due to the exchanger activity and no increases were observed. We thus conclude that the sodium-calcium exchanger is not the route of sodium's modulation of neurotransmitter release.

5. Conclusions

The two simulations described here are an important demonstration of how massively parallel computing can be applied to solve problems in biology related to reaction and transport on at a continuum scale. More importantly, they show that the existing Sandia software and hardware that was developed for issues related to stockpile stewardship can also be easily applied to fundamental problems in medicine and biology. As such problems become more critical to the nation, we believe that there are many more applications (such as cardiac cell simulation and more extensive neurological cell simulation) that can be addressed using the technologies described here.

6. References

1. Eisenstat, S. C., and Walker, H. F., “Globally convergent inexact Newton methods*”, SIAM J. Optimization*, 4 (1994) 393-422
2. Gartling, D. K., “NACHOS II: A Finite Element Computer Program for Incompressible Flow Problems. Part 1 – Theoretical Background,” *Sandia National Laboratories Report,* SAND86– 1816, Albuquerque, NM (1986).
3. Gresho, P. M., Lee, R. L., Sani, R. L., “On the time dependent solution of the incompressible Navier-Stokes equations in two and three dimensions,” Recent Advances in Numerical Methods in Fluids, Vol. 1, Pineridge Press, Swansea, U. K., 27–81 (1980).
4. B. Hendrickson and R. Leland. “A user’s guide to Chaco, Version 1.0.” Sandia National Laboratories Technical Report, SAND93-2339, Albuquerque, NM, (1993).
5. A. C. Hindmarsh, “LSODE and LSODEI: Two new Initial Value Ordinary Differential Equation Solvers”, ACM Signum Newsletter, 15, No. 4, pp 10-11, 1980
6. S. A. Hutchinson, L., Prevost, J. N. Shadid, C. Tong, and R. S. Tuminaro, “Aztec User’s Guide Version 2.0", Sandia National Laboratories Technical Report, Sand99-8801J 1999
7. J. N. Shadid, “A Fully-coupled Newton-Krylov Solution Method for Parallel Unstructured Finite Element Fluid Flow, Heat and Mass Transfer Simulations”, Int J. CFD, Vol 12, pp. 199-211, 1999
8. J. N. Shadid, “A Comparison of Parallel Preconditioners for Solution of Unstructured Finite Element Fluid Flow, Heat and Mass Transfer Simulations”, Proceedings of the Fourth japan-US Symposium on Finite Element Methods in Large-Scale Computational Fluid Dynamics, Nihon University, Tokyo Japan, April 2- 4, 1998
9. Shadid, J.N., Hutchinson, S.A., Hennigan, G.L., Moffat, H.K., Devine, K.D., Salinger, A. G., “Efficient Parallel Computation of Unstructured Finite Element Reacting Flow Solutions”, *Parallel Computing* 23, 1307-1325, 1997
10. Shadid, J.N., Tuminaro, R.S., and Walker, H.F., “An Inexact Newton Method for Fully-Coupled Solution of the Navier-Stokes Equations with Heat and Mass Transport.” *J. Comput. Phys*., 137, 155-185 (1997)
11. Shadid, J.N., and Tuminaro, R.S., “A Comparison of Preconditioned Nonsymmetric Krylov Methods on a Large-Scale MIMD Machine,” *SIAM J. Sci. Comput.*, Vol 15, No. 2, pp 440-459, March 1994
12. Shadid, J., Salinger, A., Schmidt, R., Smith, T., Hutchinson, S., Hennigan, G., Devine, K., Moffat, H., “MPSalsa Version 1.5: A “MPSalsa: A Finite Element Computer Program for Reacting Flow Problems; Part 1 - Theoretical Development”, Sandia National Laboratories Technical Report, SAND98-2864
13. R. S. Tuminaro, J. N. Shadid, and S. A. Hutchinson, “Parallel Sparse Matrix-Vector Multiply Software for Matrices with Data Locality”, Concurrency: Practice and Experience, Vol 10(3), 229--247, March 1998
14. R. S. Tuminaro, C. H. Tong, J.N. Shadid, K.D. Devine, D.M. Day, “On a Multilevel Preconditioning Module for Unstructured Mesh Krylov Solvers: Two -level Schwarz”, Submitted to Comm. Numer. Meth. Eng,
15. J.Wagner, Y.Li, J.Pearson, J.Keizer, “Simulation of the Fertilization Ca2+ Wave in Xenopus laevis Eggs”, Biophysical J. Vol. 75, 2088-2097, October 1988
16. S. A. Means, Master’s thesis, The University of New Mexico, July 2001
17. D.W. Hilgemann, “Numerical Approximation of Sodium-Calcium Exchange”, Progress in Biophysics and Molecular Biology, Vol. 51, 1-45, 1988.

Distribution

MS 0151	T. Hunter, 9000
MS 0321	W. Camp, 9200
MS 1110	D. Womble, 9211
MS 0847	R. Leland, 9220
MS 0310	P. Yarrington, 9230
MS 0310	G. Heffelfinger, 9209
MS 0316	S. Dosanjh, 9233
MS 0316	J. Aidun, 9235
MS 0318	G. Davidson, 9200
MS 0316	M. Rintoul, 9209 **(10)**
MS 0847	J. Shepherd, 9226
MS 1111	J. Shadid, 9233 **(5)**
MS 1111	A. Salinger, 9233
MS 0316	R. Pawlowski, 9233
MS 0316	G. Hennigan, 9233
MS 0196	S. Means, 9235 **(5)**
MS 0513	A. Romig, 1000
MS 1427	J. Phillips, 1100
MS 1425	M. Derzon, 1740
MS 0885	M. Cieslak, 1801
MS 9004	J. Vitko, 8100
MS 9951	L. Napolitano, 8130
MS 0839	G. Yonas, 16000
MS 9016	Central Technical Files, 8945-1
MS 0899	Technical Library, 9616
MS 0612	Review & Approval Desk, 9612 For DOE/OSTI