SPATIALLY REALISTIC COMPUTATIONAL PHYSIOLOGY: PAST, PRESENT AND FUTURE

J.R. Stiles[a*], W.C. Ford[a†], J.M. Pattillo[a], T.E. Deerinck[b], M.H. Ellisman[b], T.M. Bartol[c], and T.J. Sejnowski[c]

[a]Pittsburgh Supercomputing Center, Carnegie Mellon University, 4400 Fifth Ave., Pittsburgh, PA 15213
[b]Department of Neurosciences, University of California at San Diego, La Jolla, CA 92093
[c]Computational Neurobiology Laboratory, The Salk Institute, La Jolla, CA 92037

1. COMPUTATIONAL PHYSIOLOGY, SPATIAL REALISM, AND PARALLEL COMPUTING

Recent technological advances have changed biology from a largely data-limited, qualitative science to one of increasing quantitation that spans wide ranges of space and time. This transition creates two important downstream needs, the first aimed at large-scale data storage and analysis (Bioinformatics), and the second at simulation and prediction (Computational Physiology and Biophysics).

Physiological function depends on the spatial and temporal dynamics of specific genes, proteins, signaling molecules, and metabolites within and between cells. Realistic physiological simulations present a grand challenge because of the wide range of underlying space and time scales, as well as the widely disparate organization and properties of different cells. Real cells, as opposed to typical textbook cartoons, are highly organized and structured. The relevant spatial dimensions within and around cytoskeletal components, membranous organelles, and macromolecular complexes are generally on the scale of tens of nm, and thus are on the same approximate scale as macromolecular complexes themselves. Many such complexes function as molecular machines, illustrating the discrete, discontinuous nature of cellular physiology and the need for spatially realistic simulations.

Spatially realistic cell models are presently the exception rather than the rule. In the past this was a direct consequence of inadequate computing power, and even on today's massively parallel machines, atomic-to-molecular simulation methods with fs time-steps cannot be applied to problems on the cellular scale. Therefore, the overriding difficulty lies in deciding how currently available computing power can be applied effectively to complex, quantitative physiological simulations. In short, a critical challenge is to develop modeling and simulation methods that allow integration of mechanisms, kinetics, and stochastic behaviors at the molecular level with structural organization and function at the cellular level. In this paper we describe unique methods that we have developed for spatially realistic physiological simulations (Section 2), illustrate their use with simulations of synaptic transmission (Section 3), and discuss future directions and needs for algorithm design and efficient large-scale parallel computing (Section 4).

2. MCell and DReAMM – A MICROPHYSIOLOGICAL MODELING ENVIRONMENT

2.1. Building Spatially Realistic Models of Physiological Systems

We are presently developing a microphysiological modeling environment that consists of MCell, a Monte Carlo cellular simulation engine, (www.mcell.psc.edu or www.mcell.cnl.salk.edu), and DReAMM, a program for the Design, Rendering, and Animation of MCell Models (www.mcell.psc.edu/DReAMM).

MCell is a general tool for spatially realistic reaction-diffusion simulations [1,2,3]. It is written in C, reads input files composed of a high-level, modular Model Description Language (MDL), and is based on unique Stochastic Space Leaping (SSL) computational algorithms. These SSL methods include the use of surface meshes to represent cellular and subcellular structures, grid-free Brownian Dynamics (BD) and ray tracing/ray marching (RTRM) algorithms for molecular diffusion, and Monte Carlo probabilities that couple chemical reactions to BD and RTRM operations.

DReAMM is a model design, visualization, and analysis program designed to read general mesh data as well as MCell-specific output. DReAMM runs within the OpenDX visual programming and rendering environment (www.opendx.org), but is designed to look and feel like a typical GUI-based application program. Even small MCell models can easily lead to large rendering tasks (>>10^7 polygons), and so the user chooses what to render or animate, and at what level of detail. Unless indicated otherwise, all 3-D

[*] Supported by NIH R01 GM068630, P20 GM65805, P41 RR06009 (JRS), P41 RR004050 (MHE), and HHMI (TJS)
[†] Present Address: Computation and Neural Systems, California Institute of Technology

images and animations included in this paper were produced with DReAMM, but further details are not included here.

Figure 1 illustrates the two basic approaches that can be used to build spatially realistic models. The first step is to generate the necessary meshes. For some projects, representative model geometry is built *in silico* from average anatomical measurements and computer aided design software (Fig. 1B&C). With this

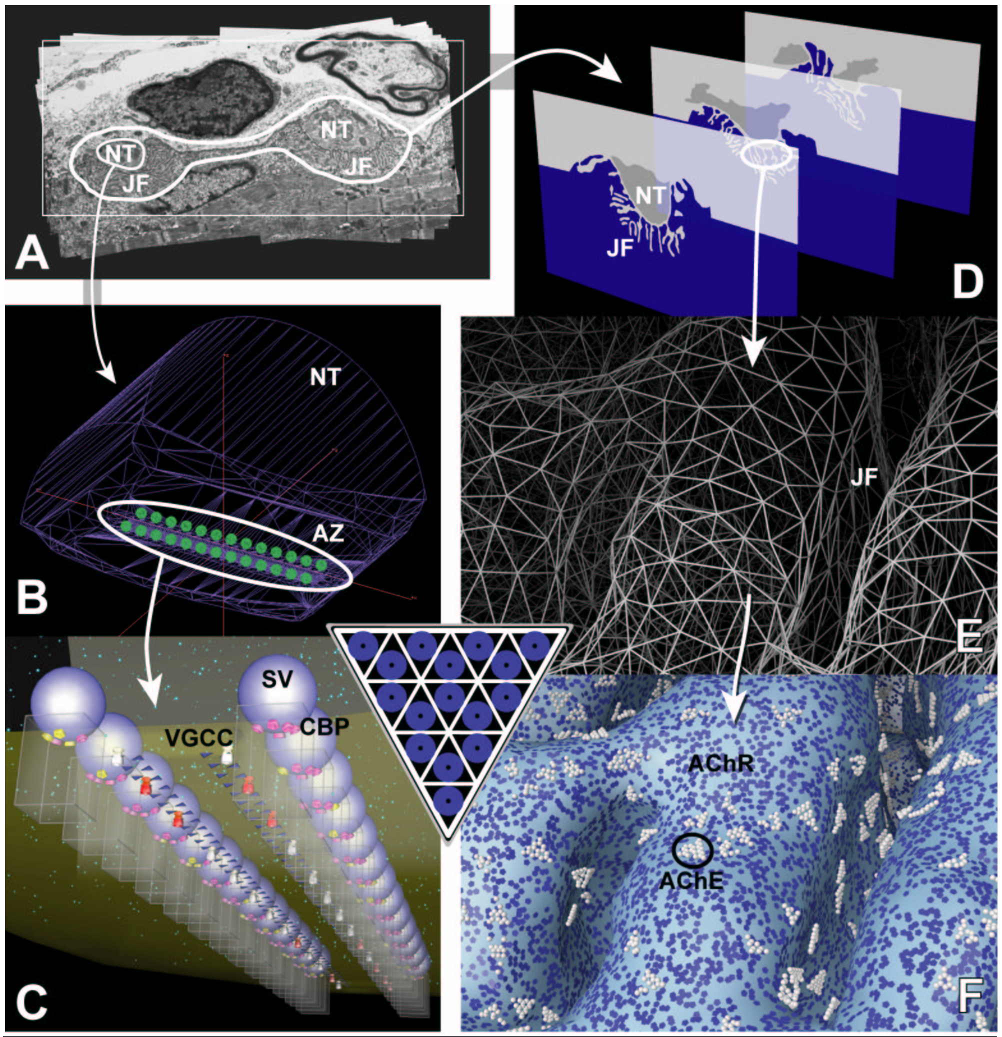

Figure 1. Building spatially realistic models of the neuromuscular junction. Electron microscopic data (A) is used to create meshes *in silico* (B, using FormZ, www.formz.com) or by contouring and reconstructing surfaces (D, E). In either case, mesh elements are tiled (inset, see text) to create positions for molecules (C, F) such as voltage-gated calcium channels (VGCC), calcium binding proteins (CBP), acetylcholine receptors (AChR), and acetylcholinesterase (AChE). NT, nerve terminal; JF, junctional folds; AZ, active zone; SV, synaptic vesicle.

approach, changes to the model's spatial parameters and their potential impact can be investigated systematically. For other projects, mesh generation requires surface reconstruction from serial electron microscopic data (Fig. 1A, D-F). Complex reconstructed models are not as easy to manipulate as those generated *in silico*, but they can be critically important to studies investigating the biophysical factors underlying biological variability.

Meshes can have different properties and functions. For example, they may be reflective or transparent to diffusing molecules, and may form cellular superstructures, scaffolding for molecular positions, or sampling boundaries. Meshes to be populated with molecules (e.g., membrane-bound proteins) are first triangulated and then each element is tiled using barycentric subdivision (Fig. 1, inset). Each tile serves as a potential location for a molecule, and the average tile size usually matches the area occupied by a typical protein. Thus, MCell's algorithms represent single molecule locations explicitly, but give up details of molecular structure for the sake of computational efficiency.

Depending on the mesh topology and need for precisely positioned molecules, some mesh regions may be highly refined while others remain fairly coarse (Fig. 1B&C). If a mesh must form a continuous diffusion boundary its elements must share edges exactly, but otherwise this is not required. In some cases disjoint polygons are used to produce sites for aggregated molecule locations (Fig. 1F). Unlike finite element simulations, mesh topology and refinement *per se* have virtually no impact on MCell's numerical accuracy, but can have a major impact on compute time and memory use (Section 4; [2,3]).

2.2. SSL Algorithms: Grid-free Brownian Dynamics

Movements of cellular constituents occur by a combination of diffusion, bulk flow, and protein-mediated (active) transport mechanisms. Diffusion is critically important over small subcellular-to-cellular distances, and a common numerical approach in 3-D is to discretize the relevant flux equations using a volume mesh. This approach is generally considered satisfactory as long as the product of voxel size and molecule concentration yields many molecules per voxel. With a realistic subcellular mesh, however, this requirement is likely to be violated. For example, embedding the surface meshes of Fig. 1 into a volume would generate many small voxels with edge lengths of 50 nm or less. With correspondingly small volumes, physiological concentrations then could easily yield less than one molecule per voxel.

At the level of single molecules, diffusion is driven by thermal velocity and collisions on the scale of Brownian motion. Numerical simulation can be implemented with many different random walk algorithms, but for realism and computational efficiency the algorithm must allow each moving molecule to sample every point in space (i.e., be grid-free). In addition, it must use a time-step that is very long with respect to true Brownian motion (~10^{-13} sec; [4]). To satisfy these constraints, individual molecular movements can be of variable lengths in random radial directions.

A net displacement of length l in a random radial direction during time Δt in reality results from a sequence of many actual (unknown) Brownian steps. The probability distribution for l is given by [3]:

$$p_r = \frac{1}{(4\pi D\Delta t)^{3/2}} e^{-r^2/4D\Delta t} (4\pi r^2\, dr) \qquad (1)$$

where r is radial distance, and p_r is the probability that l is between r and $(r + dr)$ from the original location after time Δt. D is the molecule's diffusion coefficient and reflects an effective mobility in solution. Its value depends on the nature of the solution (e.g., physiological saline) and temperature (e.g., [5]).

Equation 1 could be sampled to the limit of machine precision for each random walk movement, but this is computationally expensive and unnecessary. Instead, MCell begins a simulation by subdividing the distribution into many unequal lengths of equal probability, and builds a look-up table of step lengths. A large look-up table of radial directions is also constructed, using methods that assure lack of bias [3]. Thereafter, each movement is generated by subdividing a single random number into higher and

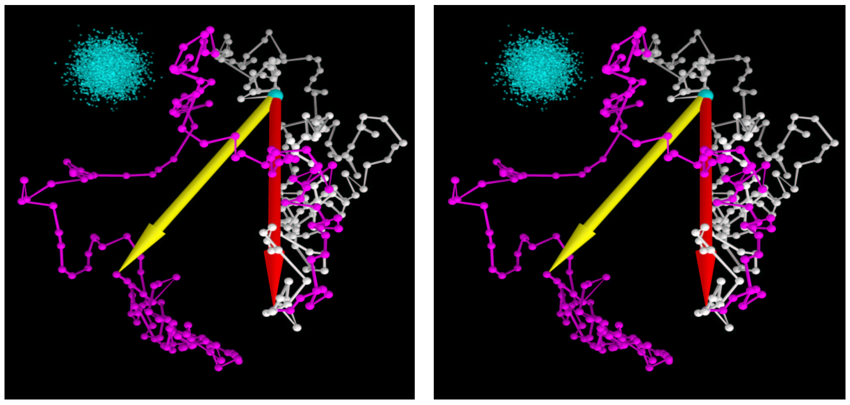
Figure 2. Grid-free Brownian Dynamics random walk. Stereo pair (cross-eyed viewing) shows cloud of molecules (upper left) obtained from a point source after a single time step (1 μs). The distribution of locations is given by Eq. 1. A path taken by a single molecule is shown for a sequence of 150 steps (white), and for an additional 150 steps (magenta). The net displacement increases by only ~10% in this example (yellow vs. red arrow), and on average would increase by $\sqrt{2}$ (Eq. 3).

lower order bits that are used to sample each table. The random numbers themselves are uniformly distributed and are read from a buffer that is refilled periodically. Performance scales linearly with the number of diffusing molecules, is very fast, and molecular movements are highly realistic within each time-step. The algorithm's realism is illustrated in Fig. 2, which shows a population of diffusing molecules that originated at a point source and have undergone one random walk step. The resulting cloud of molecule locations is radially symmetrical and essentially grid-free. The range of possible individual step lengths is also illustrated by a succession of movements for an individual molecule.

2.3. SSL Algorithms: Ray Tracing/Ray Marching and Monte Carlo Probabilities

During each simulation time-step, decisions are made between different possible unimolecular (first order) and bimolecular (second order) reaction transitions that can occur to molecules. Bimolecular events depend on collisions between two molecules in space, while unimolecular transitions are simple time-dependent Poisson processes that occur with probability (p_k) per time-step Δt [3]. A Monte Carlo implementation of unimolecular transitions is given by the Gillespie method and its more recent Stochastic Simulation Algorithm (SSA) variants [6,7], but these algorithms do not directly couple a molecular simulation of diffusion to bimolecular transitions. MCell is unique in this respect, and by directly tracing molecular movements and interactions, directly simulates the space- and time-dependent dynamics of non-equilibrium, non-well-mixed systems (e.g., transient signals like synaptic currents).

As a molecule diffuses, its current random walk trajectory (ray) must be traced through space to determine if it intersects with another object. If so, the result depends on the properties of the diffusing molecule and the object at the point of intersection. For example, if the ray hits a surface tile (Fig. 1) that corresponds to an unoccupied receptor protein, then a probability value (p_b) is compared to a random number to determine whether or not binding occurs. In general:

$$p_b \propto \left(\lambda \cdot \sqrt{\frac{\pi\,\Delta t}{D}} \right) \tag{2}$$

where the scaling factor ($\lambda \geq 0$) is a function of the tile's area and rate constant(s) for binding (for further details see [3]). In effect, λ and D are dictated by the identities of the molecules in the simulation, and so the magnitude of p_b is determined by the choice of Δt. While p_k for unimolecular transitions shows a typical asymptotic approach to unity as Δt increases, Eq. 2 shows that p_b has an unusual, non-asymptotic dependence on $\sqrt{(\Delta t)}$. This arises because the average rate of collisions depends on the apparent average velocity of motion ($\bar{v}_{app}$), and $\bar{v}_{app}$ depends on the mean radial displacement ($\bar{l}_r$; [3]) which in turn scales with $\sqrt{(\Delta t)}$:

$$\bar{l}_r = 4\sqrt{\frac{D\Delta t}{\pi}}\ ; \quad \bar{v}_{app} = \frac{\bar{l}_r}{\Delta t} = 4\sqrt{\frac{D}{\pi\Delta t}} \tag{3}$$

From Eq. 3, a 2-fold increase in Δt will increase $\bar{l}_r$ by only $\sqrt{2}$, while $\bar{v}_{app}$ decreases by $\sqrt{2}$. This is a consequence of tracking net displacements rather than the total distance traveled at the level of true Brownian motion, and is illustrated by the arrows in Fig. 2.

While MCell's numerical accuracy is governed by p_k, p_b, $\bar{l}_r$, and other factors, all of these parameters depend simultaneously on Δt and not on a voxel-based discretization of space. Therefore, a simple reduction of Δt reduces spatial (via the random walk) and temporal granularity simultaneously, and thus it is very easy to test for convergence of results [2]. Given typical biological systems, highly accurate simulations can be obtained with Δt on the scale of 10^{-6} sec, compared to 10^{-15} or 10^{-13} sec for Molecular Dynamics simulations or true Brownian motion, respectively.

3. SIMULATION OF SYNAPTIC TRANSMISSION AT THE NERVE-MUSCLE SYNAPSE

Synaptic transmission exemplifies cellular interactions in which stochastic behaviors and spatial complexity are very important. Changes in synaptic signal size and time course (plasticity) may underlie high-level cognitive functions such as learning and memory, and can also be an integral component of many neurological disorders [8]. Several of our projects focus on the vertebrate neuromuscular junction (NMJ), the synapse between a nerve and muscle cell. This is a large archetypical synapse for which there is a wealth of normal and pathological data related to structure and function, and thus it is well suited to quantitative predictive simulations [2,9].

At the NMJ, single packets of neurotransmitter molecules (acetylcholine; ACh; ≤10000 per packet) are released spontaneously from the resting nerve, and multiple packets are released when the nerve fires. Each packet originates from a small spherical vesicle (Fig. 1A-C), and the released ACh diffuses across and within the synaptic cleft to activate neurotransmitter receptor proteins (AChRs, Fig. 1F) on the muscle cell. Removal of ACh from the cleft occurs via chemical breakdown mediated by enzyme proteins (acetylcholinesterase, AChE) localized within the cleft (Fig. 1F). The synaptic signal is a small electrical current carried by sodium ions flowing into the muscle cell through open pores (channels) in the activated AChRs. The released packets of ACh are spatially segregated and so make spatially distinct synaptic currents on sub-μm scales, each called a miniature endplate current (mEPC). Maximum mEPC amplitude corresponds to ~1000 open channels, time to peak is ~0.3 ms, and the decay phase is exponential with an *e*-fold time of 1-2 ms.

Even under normal conditions, mEPCs show significant variability arising from many possible underlying factors. Their relative importance is unknown and can be investigated by simulating mEPC generation in spatially realistic models. To do the simulations, it is necessary to reconstruct a portion of NMJ architecture, and then to simulate mEPCs while systematically varying spatial and/or chemical kinetic inputs. Some of the primary candidate factors underlying variability include differences in cleft topology (volume) from one ACh release site to another, different spatial arrangements of AChR and AChE molecules, and differences in vesicular ACh content and release kinetics.

We have begun investigating this issue by focusing first on cleft topology. To reconstruct part of an NMJ, we selected 60 sections from ~400 serial electron micrographs (mouse sternomastoid muscle; 80 nm section thickness), and traced and interpolated nerve and muscle membrane contours. Using a marching cubes algorithm and subsequent decimation, we reconstructed pre- and postsynaptic membrane surfaces (Fig.'s 1 and 3). To sample the effect of tortuous cleft topology on mEPCs, the nerve terminal mesh was

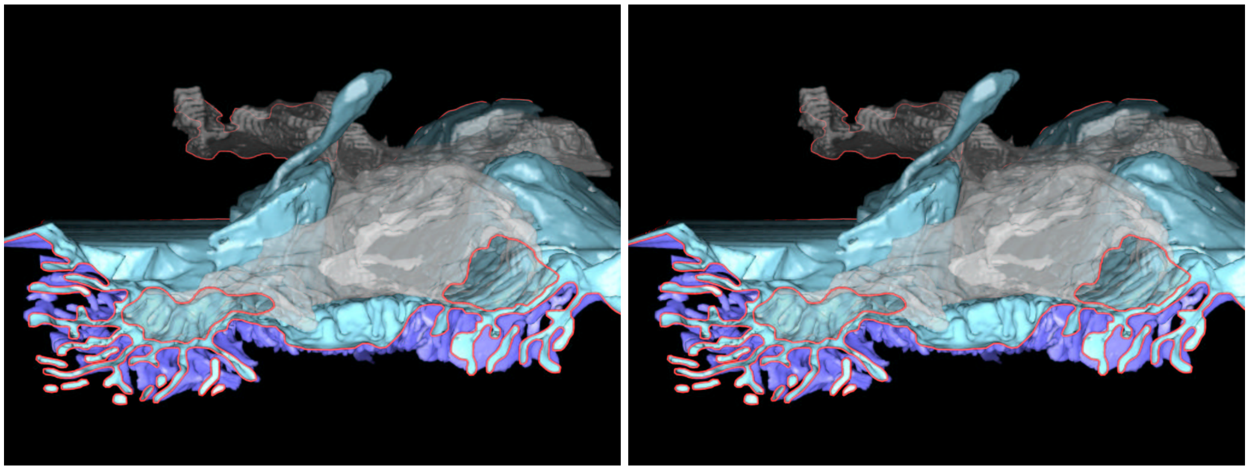

Figure 3. Mouse sternomastoid muscle NMJ reconstruction. Stereo pair (cross-eyed viewing) shows nerve terminal in translucent gray, extracellular face of muscle membrane in light blue, and cytoplasmic face of muscle membrane in dark blue. Cut edges of membrane surfaces are outlined in red. Approximate dimensions 10 x 5 x 5 μm.

sampled to obtain different potential ACh release sites at 100 nm average nearest neighbor spacing (~3800 total sites, Fig. 4). In addition, the arrangement of AChE sites within the cleft reflected different assemblies (isoforms) of AChE molecules [10], on average yielding 12-mers of the catalytic subunit (Fig. 1F).

Figure 5 shows representative samples of simulated mEPCs, obtained either with repetitive ACh release from a single site (top panels), or successive release from different sites (bottom panels). These data thus illustrate predicted Type I (arising from a single release site) versus Type II mEPC variability (arising from multiple release sites). With all other model parameters [1,2,11] held constant, it is quite evident that Type

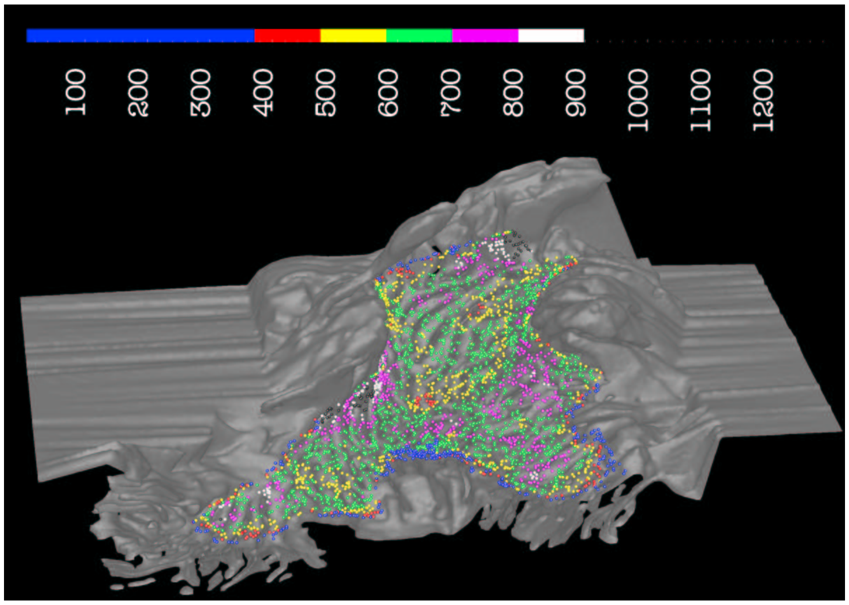

Figure 5. Predicted mEPC amplitude (open AChR channels) as a function of ACh release location. Each sphere represents a possible synaptic vesicle location (~3800 total).

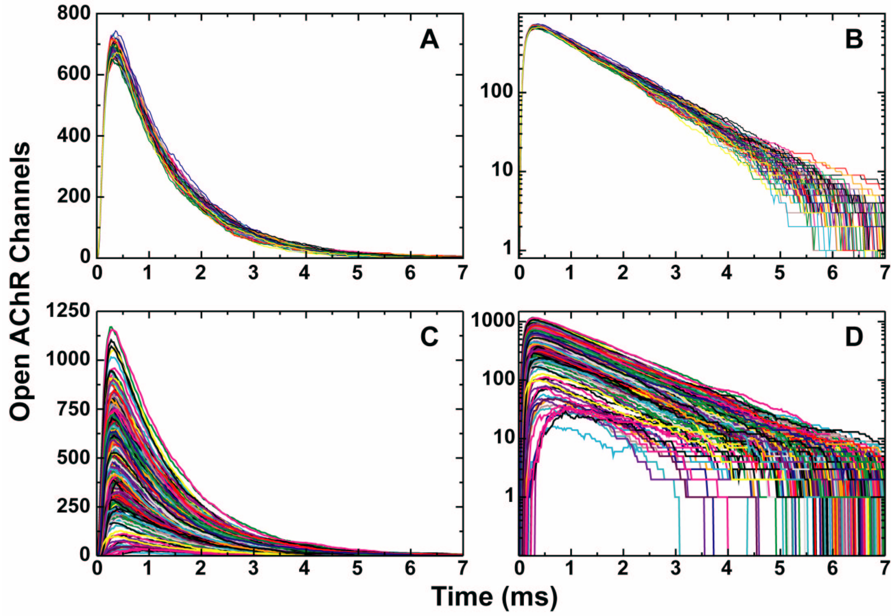

Figure 4. Predicted Type I (A and B, 100 mEPCs) and II (C and D, ~1000 mEPCs) variability. AChR and AChE distributions were similar to those shown in Fig. 1F. Similar decay times (see text) are shown by similar slopes in B and D.

II variability is predominant, and therefore synaptic topology very likely plays an important role in determining mEPC variability. Figure 4 shows how mEPC amplitude varied as a function of ACh release site location.

The predicted variability of mEPC amplitude and rise time approaches that seen experimentally (e.g., [11]), but the variability of decay time is much less than that seen experimentally. Thus, reproduction of the experimental distribution of decay times will probably require that some novel mechanism be added to the model, such as different AChR populations (based on channel opening and closing rates) in different spatial regions of postsynaptic membrane. In addition, preliminary simulations are currently being run with AChE inhibition, and these may suggest that a second AChR open channel state is required in the model to account for amplitude increases seen experimentally under these conditions (AChE inhibitor drugs are used to treat Myasthenia Gravis, an autoimmune disease that attacks AChRs and disrupts NMJ structure [9]).

4. COMPUTATIONAL ISSUES: PRESENT AND FUTURE

The computational cost of an MCell simulation is mostly dependent on the number of tests for ray/polygon intersections. In a naïve algorithm, every mesh element would have to be checked for a potential intersection every time each diffusing molecule moves. Thus, computer time would scale roughly as *O(NM)*, where *N* is the number of diffusing molecules and *M* is the total number of mesh elements. This would be untenable as *M* increases by many orders of magnitude for large-scale models, and therefore space is partitioned into subvolumes that are used to optimize the search for collisions.

As a model's spatial complexity increases, more subvolumes can be used so that execution time scales roughly as *O(N)*. Additional subvolumes increase memory use, however, and large models can become memory-bound. For each mEPC simulation as shown in Fig. 5, optimal run-time conditions require ~2 GBytes of RAM and on the order of 10 minutes on a single 64-bit processor. With thousands of ACh release sites and thousands of replicate simulations per site, the number of simulations can easily grow to the scale of 10^5 and beyond. Thus, total computer time can easily reach the scale of CPU years. Since each simulation is generally an independent task, such projects are well suited to embarrassingly parallel execution. Under certain conditions this can be implemented effectively on the Grid, assuming adequate memory resources, efficient staging of input files, appropriately handled output, and effective scheduling strategies [12].

With the recent growth of large shared memory architectures, another possible route for efficient parallelization is to replicate a particular model many times within a single execution task. In effect, each replicate would be translated into a different region of spatial subvolumes, and the subvolume regions would map to different processors. This approach could be very effective for parameter sweeps and repeated trials run with different random numbers, and would tend to minimize load balancing issues.

Future extensions to MCell's present capabilities include simulation of arbitrary chemical interactions between diffusing molecules in solution as well as in membrane environments, and also incorporation of volume meshes with embedded surfaces. These extensions will enable hybrid Monte Carlo/finite element approaches to simulations on dramatically expanded scales of space and time, and, inevitably, will also dramatically expand the computational costs. Such algorithms will pose major challenges for efficient parallelization, since they will require asynchronous, latency-tolerant control to achieve dynamic load balancing. Day-to-day use of such a computational environment will also have to be relatively transparent to a growing community of computational physiologists who are not expert with large computer technology. Although these are difficult problems, the eventual impact on personalized health and medicine will be tremendous.

REFERENCES

[1] J.R. Stiles, D. Van Helden, T.M. Bartol, Jr., E.E. Salpeter and M.M. Salpeter, Proc. Natl. Acad. Sci. USA 93 (1996) 5747.

[2] J.R. Stiles, T.M. Bartol, M.M. Salpeter, E.E. Salpeter and T.J. Sejnowski, in *Synapses*, W.M. Cowan, C.F. Stevens, T.C. Südhof (eds.), Johns Hopkins Univ. Press, Baltimore, 2001.

[3] J.R. Stiles and T.M. Bartol, in *Computational Neuroscience: Realistic Modeling for Experimentalists*, E. De Schutter (ed.), CRC Press, Boca Raton, 2001.

[4] G.M. Barrow, *Physical Chemistry for the Life Sciences*, McGraw-Hill, Inc., New York, 1981.

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

[6] D.T. Gillespie, J. Phys. Chem. 81 (1977) 2340.

[7] D.T. Gillespie, Physica A. 188 (1992) 404.

[8] W.M. Cowan, C.F. Stevens, T.C. Südhof (eds.), *Synapses*, Johns Hopkins Univ. Press, Baltimore, 2001.

[9] M.M. Salpeter (ed.), *The Vertebrate Neuromuscular Junction*, Alan R. Liss, Inc., New York, 1987.

[10] C. Legay, Micros. Res. Tech. 49 (2000) 56.

[11] J.R. Stiles, I.V. Kovyazina, E.E. Salpeter and M.M. Salpeter, Biophys. J. 77 (1999) 1177.

[12] H. Casanova, T.M. Bartol, J. Stiles and F. Berman, Intern. J. High Perf. Comp. App. 15 (2001) 243.