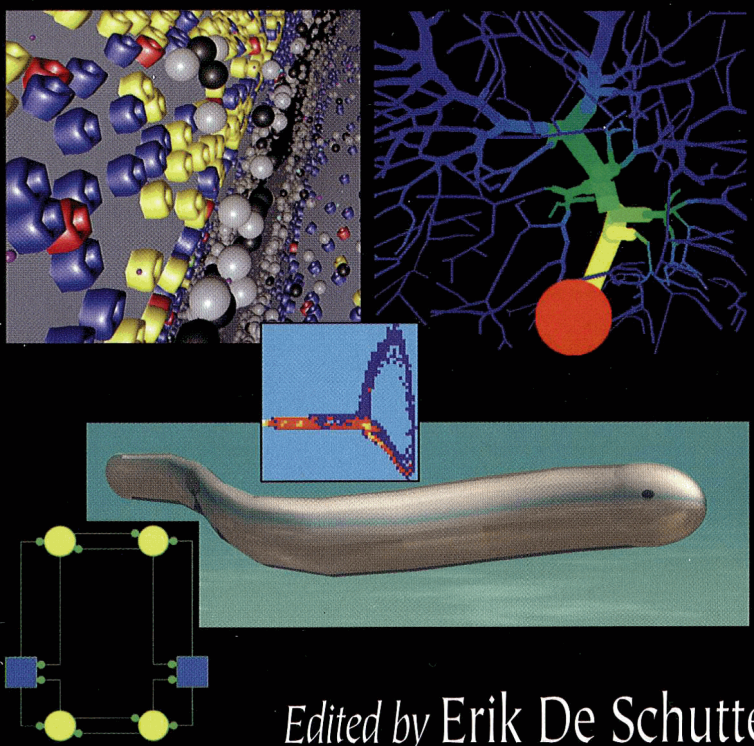
COMPUTATIONAL NEUROSCIENCE

REALISTIC MODELING FOR EXPERIMENTALISTS



Edited by Erik De Schutter

METHODS & NEW FRONTIERS IN NEUROSCIENCE

4 Monte Carlo Methods for Simulating Realistic Synaptic Microphysiology Using MCell

Joel R. Stiles and Thomas M. Bartol

CONTENTS

0-8493-2068-2/01/$0.00+$.50

4.1 INTRODUCTION

The function of the nervous system can be investigated at widely different scales, and computational approaches can range from space-filled atomic resolution (e.g., molecular dynamics simulations of protein structure) to space-independent methods (e.g., information theory applied to spike trains). At the level of cellular and subcellular signaling, models can be tailored to different questions that require different levels of structural realism. For example, neuronal excitation can be addressed using isopotential compartmental models (Chapters 5–9), biochemical networks of signaling pathways might be simulated using well-mixed assumptions (Chapter 2), and certain types of reaction-diffusion problems can be handled using spatial simplifications and boundary conditions that allow effectively one-dimensional (1-D) analytic and/or finite difference approaches (Chapter 3).

At some level of reaction-diffusion problems, however, realistic 3-D cell structures become critical components of quantitative modeling. For example, if identification of pre- and postsynaptic factors that contribute to synaptic function, variability, and crosstalk are of interest,[3,8,11,15,22,23,33] then incorporation of actual ultrastructure into the model cannot be avoided. Similarly, if quantitative simulations of realistic Ca^{+2} dynamics in and around dendritic spines are the goal, then the shapes, sizes, and other biophysical properties of real intra- and extracellular spaces will have to be included in models. Thus, a realistic, fully 3-D approach is required for quantitative modeling of *microphysiology*, i.e., cellular physiology that includes *3-D ultrastructural organization* (intra- and extracellular diffusion spaces, site densities and distributions of macromolecules), *movements of mobile molecules* (signaling ligands, ions, gases, free water), and *biochemical reactions* (transition paths, reaction rates).

In this chapter we illustrate *Monte Carlo methods* for realistic 3-D modeling of synaptic microphysiology. We begin with a discussion of the steps required to create a microphysiological model, and then briefly compare the two alternative computational paradigms that can be used to simulate the model, *finite element* versus *Monte Carlo* (Section 4.2). Although the general power and increased realism of Monte Carlo methods have been recognized for many years,[9,13,21] inadequate computer resources precluded many applications until the recent explosion in speed, memory, and storage capacity. We will describe in this chapter MCell, Monte Carlo simulator of cellular microphysiology. MCell is freely available and can be obtained for a wide variety of UNIX, Windows, and MacIntosh environments, see *http://www.mcell.cnl.salk.edu* or *http://www.mcell.psc.edu*. With MCell highly realistic synaptic simulations can be run on present-day workstations given suitably optimized algorithms and run-time conditions. In addition, very large-scale projects can be run on increasingly accessible parallel processing resources if necessary.

The definition and control of model input and output parameters, simulation operations, and critical run-time optimizations are covered in Section 4.3. The theoretical foundation of simulation operations and numerical accuracy is discussed in Section 4.4. Finally, Sections 4.5 and 4.6 illustrate the use of MCell's diffusion, reaction, and optimization features to simulate quantal current generation[1,5] together with neurotransmitter exocytosis[30–32] at a realistic neuromuscular junction[33] and neuronal cell body.

4.2 GENERAL BACKGROUND ON DESIGN AND SIMULATION OF MICROPHYSIOLOGICAL MODELS

4.2.1 Model Design

Quantitative microphysiological modeling is in its infancy, largely because realistic 3-D models necessarily entail a "scaled-up" computational approach compared to simpler models, and can rapidly escalate into supercomputing territory. In essence, such modeling encompasses four steps, each of which can require considerable computing resources and expertise: reconstruction, model visualization and design, simulation, and visualization and analysis of results. The third step is the primary focus of this chapter; the third and fourth together can involve large-scale parameter fitting and sensitivity analyses (i.e., the sensitivity of a model output quantity to the value of one or more input parameters, Chapter 1, and see also Reference 32).

To simulate a realistic microphysiological system, a representation of the relevant cellular and subcellular structures must first be generated at very high resolution. It is not necessary (nor presently possible) to include the 3-D structure of each molecule, but their positions in space with respect to diffusion boundaries (i.e., cell and organelle membranes) must either be known and included in the model, or predicted by comparison of simulation results to experimental data. The requisite model of cellular structures thus requires a highly accurate 3-D reconstruction, most likely at the EM rather than light level. While light level methods for single cells are fairly well established (Chapter 6), EM level reconstruction of intra- and extracellular spaces for use with simulations is new ground, and is heavily dependent on large-scale computer graphics algorithms. Some of the issues introduced by the need for high resolution and accuracy are outlined in Reference 33, and a general treatment of surface representation and visualization can be found in Reference 28, which also includes the *Visualization Toolkit* software. Another extremely useful software package for model visualization and multidimensional data analysis is IBM DataExplorer (OpenDX, which includes extensive documentation and tutorials; http://www.research.ibm.com/dx).

4.2.2 Simulations Based on Microphysiological Models

Regardless of the methods used to create a microphysiological model, it must be designed so that it subsequently can be imported into a simulation program. At this level of detail and realism, an interactive link between model design and simulation

is not a trivial problem, nor is a general solution presently available. For the simulation methods covered in this chapter, the present state of the art is a combination of the above graphics software and a Model Description Language (MDL) covered in Section 4.3.

The simulation program itself can be based on one of two general numerical paradigms, finite element (FE) or Monte Carlo (MC). Both methods have been utilized heavily in computational chemistry and physics but have not yet become commonplace in computational biology, again largely due to the effort involved in first creating the model to be simulated. As outlined below, the FE approach is a 3-D extension of familiar methods based on differential equations (Section 1.1), while the MC approach is altogether different. With a set of equations one predicts the idealized average behavior of a system, while with MC methods one uses random numbers and probabilities to simulate individual cases of the system's behavior. This use of random numbers to "throw dice" and make decisions led Ulam and Von Neumann to coin the "Monte Carlo" appellation in the days of the Manhattan Project.[21] Specifics of MC methods vary according to the problems being investigated, but the results invariably include some form of quantitative stochastic noise that reflects the underlying probabilistic algorithms. In some situations this noise is an undesired drawback and must be reduced by averaging across multiple simulations, but in other cases (especially biology) the variability itself may be of great interest (e.g., synaptic current variability) and may contain useful information.

FE Methods — With FE simulations, a 3-D space is subdivided into contiguous volume elements, or *voxels*. Well-mixed conditions are assumed within each voxel, and differential equations that describe mass action kinetics are used to compute fluxes between and reaction rates within each voxel.[29,34] The methods for solution of the equations are essentially no different than has been described in preceding chapters for problems of reduced dimensionality. High numerical accuracy can be achieved by finely subdividing both space and time, i.e., by using fine spatial and temporal granularity.

As long as the voxels of a FE model are arranged in a regular 3-D grid, their implementation in a simulation can be relatively straightforward, but with realistic cellular structures the grid must be extremely fine and/or must be irregular in shape. With an extremely fine grid the voxels are small and numerous, and the computational expense can grow to be very large (e.g., hydrodynamics and turbulence simulations). With irregular voxels the design of the grid itself becomes an additional large-scale problem, and the dependence of numerical accuracy on grid properties becomes more difficult to assess.

As voxel size decreases, the product of voxel volume and reactant concentration is likely to yield only fractional amounts of molecules. Mass action equations still predict the average behavior of the system, but the stochastic nature and sometimes non-intuitive variability of interactions on the molecular scale are ignored. In principle, some degree of stochastic behavior can be incorporated into equation-based methods,[2,10,18] but at this stage the alternative MC modeling approach becomes highly appealing.

MC Methods — For 3-D reaction-diffusion problems MC methods replace voxels and sets of differential equations with stochastic molecular events simulated

directly within the reconstructed volume of tissue. Individual ligand molecules diffuse by means of *random walk movements*, which reproduce net displacements that in reality would arise from Brownian motion. Movement trajectories can be reflected from arbitrary surfaces that represent cell and organelle membranes, and thus *a quantitative simulation of diffusion in complex spatial locales is obtained without the use of voxels*. In addition, reaction transitions such as ligand binding, unbinding, and protein conformational changes can be simulated probabilistically on a molecule-by-molecule basis, using random numbers to throw the dice and test each different possibility against *a corresponding MC probability*. Tests for binding are performed each time a diffusing ligand molecule "hits" an available binding site, and successful tests for unbinding are followed by newly generated random walk movements. *Hence, combined reaction and diffusion can be simulated in any arbitrary 3-D space*. The MC modeling approach thus is very general, and in a sense is easier to implement for complex structures than the FE approach. In addition, MC simulations reproduce the stochastic variability and non-intuitive behavior of discontinuous, realistic 3-D microenvironments that contain finite numbers of molecules.

4.3 GENERAL MONTE CARLO SIMULATIONS: SPECIFIC ISSUES OF DESIGN AND EXECUTION

4.3.1 A Matter of Scale

While sets of differential equations can be evaluated numerically in a variety of ways (Andrew Huxley first computed the Hodgkin–Huxley equations for action potential generation using a mechanical hand-cranked calculator), and models based simply on coupled differential equations can be designed and run within equation-solving software environments (e.g., Matlab or Mathematica), the scope and characteristics of general MC simulations dictate a very different approach. Until recently, MC simulation programs in neuroscience were tailored specifically to a single problem and a simplified structure.[1,5–7,11,15,35] For *fully arbitrary* and *realistic* reaction-diffusion simulations, on the other hand, one must first generate a high-resolution 3-D reconstruction and populate it with molecular constituents (as outlined briefly in Section 4.2.1), and then a representation of the complete model (e.g., 5–50 Mbytes for a synaptic reconstruction) must be read and simulated. If the simulation program itself is based on computationally "naïve" MC algorithms, the time required to run large-scale simulations will be overwhelming (say, months to years) on even the fastest existing computers. With highly optimized algorithms, however, the same simulation might literally run in minutes on a workstation (Section 4.5).

Here we illustrate the design and simulation of realistic MC models using MCell and its Model Description Language (MDL). Both are presently unique tools — MCell as a highly optimized simulation program, and the MDL as a high-level user interface and link between the steps of reconstruction, model design, simulation, and output of results. We outline the underlying concepts and operations in detail so that: (1) the non-programmer can begin to run complex simulations with a minimum of effort; (2) an understanding of MC methods precludes major mistakes in the choice of input parameter values which may lead to profoundly degraded numerical accuracy; and

(3) one could, in principle at least, write a general MC simulation program from scratch (although we expect the 20+ years of aggregate theoretical, programming, and development work will dissuade even the most adventuresome).

4.3.2 Input and Output Parameters

Input parameters for a MC simulation include those that define the microphysiological environment (*simulation objects*, e.g., surfaces, positions of molecules, etc.), together with those that specify how the simulation is actually run (e.g., the length of the MC time-step Δt, the number of time-step iterations, and run-time optimizations). For MCell in particular, simulations are designed and controlled using its MDL user interface, in essence a simple programming language designed for generality, readability, and easy use by programming novices. The MDL and MCell's program flow are summarized in Figure 4.1 and Boxes 4.1 and 4.2. Specific examples follow in Sections 4.5 and 4.6.

Random Number Seed — In a sense, the "first" input parameter for a MC simulation is a *seed value* for generation of random numbers. This seed value determines the values of the random numbers used in the simulation (Section 4.3.4), and hence determines the outcome of decisions made during the simulation. When an MCell simulation is started, the seed value is specified together with the name of an MDL *input file*.* As shown schematically in Figure 4.1, the input files are parsed (read and interpreted), the specified simulation conditions are initialized, and then time-step iterations begin.

Arbitrary Surfaces — To be used with MC random walk and binding algorithms, *arbitrary surfaces* are created from *polygon meshes*. Meshes that represent reconstructed pre- and postsynaptic membranes can easily contain of order 10^6 individual polygons (*mesh elements*, MEs; Figures 4.1, 4.3, and 4.4), and each ME may be subdivided to obtain discrete *effector tiles* (ETs) that can be used to model stationary molecules (see below).** It is important to note that meshes generated for use with MC simulations must be exact, i.e., must "*hold water*." If vertices shared by adjacent MEs do not agree exactly, then random walk movements will pass through the gaps no matter how small. Reconstructions created just for visualization purposes or for use with compartmental equation-based models do not share this requirement, and typically are inadequate for MC simulations.

Surface Properties and Ligand Molecules — In order to simulate an impermeable membrane, individual MEs of a surface must be *reflective* to diffusing ligand molecules. On the other hand, it is often necessary to place stationary molecules (i.e., ETs) on surfaces that do not constitute diffusion boundaries. For example, the acetylcholinesterase (AChE) molecules in the example of Section 4.5 are located on a *transparent* surface that represents the synaptic basal lamina within the synaptic

* The specified input file itself often makes reference to other additional MDL files that contain objects and parameter definitions to be included in the simulation (see INCLUDE_FILE keyword statement in Box 4.2).

** With MCell, spatial dimensions are given in μm, and polygon meshes are created from a POLYGON_LIST object (Box 4.2). Each ME can have any number of vertices but must be convex (all internal angles less than 180°) and exactly planar. Triangular MEs are used most often (guaranteed convex and planar), and a ME *must* be triangular if it contains ETs.

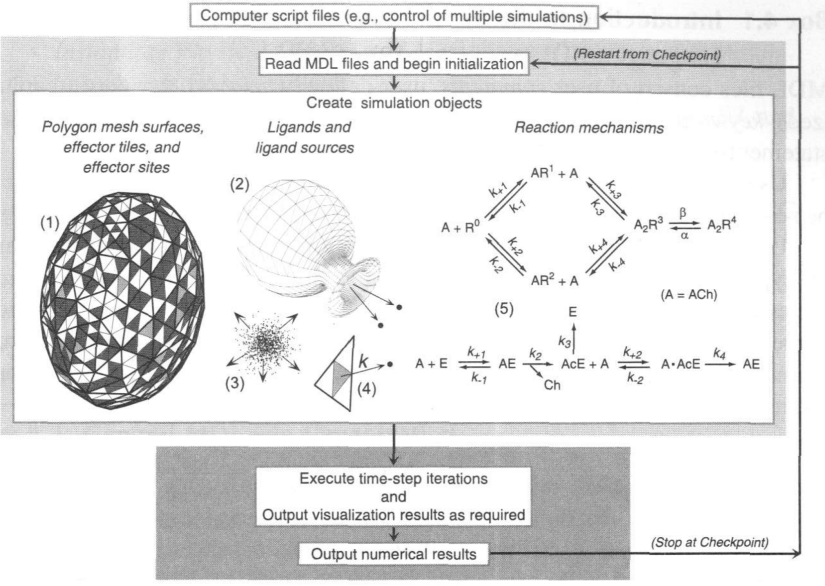


FIGURE 4.1 General Overview of MCell Simulations. Several examples of simulation objects include: (1) ovoid polygon mesh (similar to the simplified nerve cell body used in Section 4.6); (2) complex polygon mesh (synaptic vesicle and exocytotic fusion pore, see Section 4.5); (3) spherical source of ligand molecules; (4) surface that creates ligand molecules at rate k; and (5) reaction mechanisms for acetylcholine (A), acetylcholine receptors (R), and acetylcholinesterase (E, Section 4.5). Objects (1) and (4) illustrate barycentric subdivision of mesh elements (triangles with heavy black outlines) to create effector tiles (smaller triangles) that cover the surface completely and may contain effector sites (shaded triangles) in different chemical states (light vs. dark shading). Program flow and operations during initialization (light gray box) and time-step iterations (dark gray box) are detailed in Figures 4.2A and 4.2B, respectively. MDL, MCell Model Description Language.

cleft. Ligand molecules (acetylcholine, ACh) thus are able to diffuse through the basal lamina unless they encounter and bind to an AChE site. Section 4.5 also shows how transparent surfaces may be used to sample the concentration of diffusing molecules (ACh and choline, Ch, obtained after hydrolysis of ACh by AChE) in restricted regions of space. Each type of ligand molecule used in a simulation is minimally defined by a diffusion coefficient value (D_L),* which, together with the value of Δt, determines the distribution of step lengths used for random walk movements (Sections 4.4.1 and 4.4.2).

* Each type of ligand molecule is defined by two user-specified input parameters, an arbitrary name and a diffusion coefficient value ($cm^2 \cdot s^{-1}$; see DEFINE_LIGAND object in Box 4.2). Multiple predefined (see SPHERICAL_RELEASE_SITE object in Box 4.2) or arbitrary initial distributions of diffusing molecules can be used. An arbitrary initial distribution can be created within a closed polygon mesh (to shape the distribution), and then the bounding mesh can be removed (or changed from reflective to transparent) in order to initiate outward diffusion (see Section 4.3.3). In addition, different temporal patterns can be designed to trigger the release of different amounts and types of ligand molecules (see DEFINE_RELEASE_PATTERN object in Box 4.2, and Sections 4.5 and 4.6).

Box 4.1 Introduction to the MCell Model Description Language (MDL)

MDL files consist of user *comments*, user-defined *variables*, *keywords* (capitalized), *keyword statements*, and *keyword statement blocks* that have *subordinate* statements enclosed in braces (see Box 4.2 for general syntax).

User *comments* are begun with "/*" and ended with "*/". Comments can be nested (comments within comments), and are ignored when the file is parsed.

User *variables* are defined by equating an arbitrary name to a value. Names are typically given in lower case to distinguish them from upper case keywords. Types of variables include text, text expressions, numerical values, numerical expressions, and numerical arrays (elements may be values or expressions). Numerical expressions may include a variety of standard math functions. Examples:

```
user_id = "qwerty"                  /* definition of a text variable named user_id */
output_file = "run_001." & user_id  /* text expression named output_file with elements
                                       joined by the & operator; result would be
                                       "run_001.qwerty" */
k_plus = 1.5e8                      /* definition of a numerical variable named k_plus */
new_k_plus = k_plus*SQRT(3)         /* numerical expression named new_k_plus, uses the
                                       square root operator */
location = [0, 1, 0]                /* definition of numerical array named location */
```

Keyword statements and *blocks* are used to:

- *Define Values for Simulation Parameters.* As illustrated in Box 4.2, two statements are required for every simulation, to define the time-step value and the number of time-step iterations. Many optional statements can be used to define other parameters and run-time optimizations.
- *Define Logical Objects.* Logical objects have no physical location, and specify sets of input parameters for different types of ligand molecules, ligand release patterns, and chemical reaction mechanisms. Logical object definitions begin with a keyword that contains the word DEFINE.
- *Design Templates for Physical Objects.* Physical objects have a location in space, and include various types of surfaces and ligand release sites. The user specifies an object name, and the first keyword describes the object. Physical objects are initially invoked as templates that can be modified in various ways, and only exist in a simulation if instantiated (see below).
- *Design Metaobject Templates.* Physical object templates can be grouped into metaobjects, which in turn can be grouped into unlimited levels of higher order metaobjects.

(continued)

Box 4.1 (continued)

- *Instantiate Physical Objects and Metaobjects.* Creates actual simulation objects from templates.
- *Output Data for Visualizations and Animations.* See page 97 and Box 4.2.
- *Output Reaction Data Statistics.* See Box 4.2.
- *Output Other Data.* Uses syntax and formatting similar to the C programming language. Allows arbitrary file creation and write operations, printing of messages to the command line window, and conversion of numerical values to text variables.

Box 4.2 General MDL File Organization

The left column shows an abbreviated version of an MDL file, as explained briefly in the right column. MDL keywords are capitalized, and italics indicate names, values, or expressions that would be supplied by the user. Subordinate statements within statement blocks have been omitted, and their positions are indicated by ellipsis marks. When the simulation is started, the file is read (parsed) from top to bottom. Some calculations for initialization are performed while parsing, so there is some order-dependence to the file layout.

/* comment to describe purpose of MDL file */	Comments can appear anywhere to document the file.
variable_name_1 = *text_expression* *variable_name_2* = *numerical_expression* *variable_name_3* = *numerical_array*	User-defined variables can appear anywhere between statement blocks.
INCLUDE_FILE = *text_expression*	Include files can appear anywhere between statement blocks and should be documented consistently for submission to an online repository (see MCell web sites).
/* Required keyword statements */	
TIME_STEP = *numerical_expression*	Value given in seconds.
ITERATIONS = *numerical_expression*	Total number of time-step iterations.
/* Optional keyword statements */	
EFFECTOR_GRID_DENSITY = *numerical_expression*	Global value for barycentric tiling (tiles · μm^{-2}).
PARTITION_X = [*numerical array*]	Positions along x-axis to insert spatial partitions.
PARTITION_Y = [*numerical array*]	Positions along y-axis to insert spatial partitions.
PARTITION_Z = [*numerical array*]	Positions along z-axis to insert spatial partitions.
CHECKPOINT_INFILE = *text_expression*	Name of checkpoint file to read during initialization.
CHECKPOINT_OUTFILE = *text_expression*	Name of checkpoint file to write before stopping.
CHECKPOINT_ITERATIONS = *numerical_expression*	Number of checkpoint iterations to run before stopping

(continued)

Box 4.2 (continued)

/* Optional logical object definitions */	
DEFINE_LIGAND {...}	Requires user-specified name and diffusion coefficient.
DEFINE_RELEASE_PATTERN {...}	Timing and amount of ligand release.
DEFINE_REACTION {...}	Chemical reaction mechanism associated with ESs.
/* Optional physical object templates */	
name_1 BOX {...}	For simple structures and/or ligand sampling. May include ESs.
name_2 POLYGON_LIST {...}	For complex polygon mesh structures. May include ESs.
name_3 SPHERICAL_RELEASE_SITE {...}	Spherical distribution of ligand molecules (arbitrary diameter). May be associated with a release pattern.
/* Optional metaobject templates */	
name_4 OBJECT {...}	Hierarchical groups of physical objects and/or other metaobjects.
/* Instantiation statement(s) */	
INSTANTIATE *name_5* OBJECT {...}	Create an instance of a physical object. May include geometric transformations.
/* Optional visualization and reaction statistics output */	
VIZ_DATA_OUTPUT {...}	See example models (Sections 4.5 and 4.6).
REACTION_DATA_OUTPUT {...}	See example models (Sections 4.5 and 4.6).

Effector Sites and Reaction Mechanisms — Individual ligand-binding or other stationary molecules (e.g., receptors, transporters, or enzymes) are simulated by using *effector sites* (ESs) on MEs of surfaces. *Each ES is an ET that has a chemical reaction mechanism (see below) associated with it.* Effector sites used in conjunction with arbitrary polygon mesh surfaces and reaction mechanisms constitute one of the most powerful and unique features of MCell simulations. When the user specifies that a ME contains a particular type of ES at some density, σ_{ES} (sites $\cdot$ μm^{-2}), MCell:

1. Covers the ME with a triangular grid of ETs using a method called barycentric subdivision (Figure 4.1). The number of ETs (N_{ET}) depends on the area of the ME (A_{ME}) and a global input parameter, the effector grid density σ_{EG} (tiles $\cdot$ μm^{-2}).* With barycentric tiling, the area of an ET (A_{ET}) is given exactly by A_{ME}/N_{ET}, and approximately by $A_{EG} = 1/\sigma_{EG}$. A_{ME}, on the other hand, depends only on how the surface was constructed.
2. Uses random numbers to decide which of the available ETs are ESs (the ratio $(\sigma_{ES} \cdot A_{ME})/(N_{ET})$ gives the probability that an ET is an ES). Note that *different types of ESs* (e.g., different subclasses of glutamate receptors) *can be intermixed on the same ME.*

Since ESs occupy positions on surfaces in a real 3-D space, reaction mechanisms** used in MC simulations can encompass the *polarity* (directionality) of

* See EFFECTOR_GRID_DENSITY keyword statement in Box 4.2.

** See DEFINE_REACTION object in Box 4.2, and Sections 4.5 and 4.6.

ligand binding and unbinding, as well as the typical *state diagram* and associated *rate constants* that define transitions between different chemical states (Figure 4.1). For example:

1. If an ES on a reflective polygon mesh is used to model a receptor protein on the plasmalemma, binding and unbinding could be defined to occur on the extracellular side of the surface.
2. An ES used as a transporter protein could bind molecules on one side of the surface and unbind on the other side.
3. An ES that represents an enzyme localized on an intra- or extracellular scaffold (transparent surface) could bind and unbind from either side.

With MCell's MDL, rate constants are input as conventional bulk solution values, in units of $(M^{-1} \cdot s^{-1})$ for *bimolecular associations* (i.e., ligand *binding*, k_{+n} values in Figure 4.1), or (s^{-1}) for *unimolecular transitions* (i.e., ligand *unbinding* [k_- values], *transformation* [k_2], *destruction* [k_3 and k_4], or *de novo production* [k], as well as ES conformational changes [α and β]). When the simulation is initialized (Section 4.3.4), bimolecular and unimolecular rate constants are converted into MC probabilities (p_b and p_k values, respectively; Sections 4.4.3–4.4.5).

Output of Visualization and Reaction Data — Since diffusion and all reaction transitions (events) occur on a molecule-by-molecule basis in a MC simulation, it is possible to track the statistics of each and every kind of event in a space- and/or time-dependent manner (e.g., the number of ESs in a particular state, the number of ligand molecules in a sampled volume, fluxes, transitions). In addition, it is often necessary to visualize snap-shots of a simulation, or create an animation from successive snap-shots. Thus, the amount of output information can be enormous. As illustrated in Sections 4.5 and 4.6, MCell's MDL includes unique facilities for selective filtering, formatting, and timing of output.*

4.3.3 Maximizing Flexibility and Efficiency Through Checkpointing

Given the nature of realistic MC simulations, computer time can be lengthy even with highly optimized algorithms. Premature termination of a simulation, with concomitant loss of results (e.g., because one's present allotment of time on a multi-user computer is used up), can be avoided through the use of *checkpointing*. As indicated by the feedback loop in Figure 4.1, checkpointing is a general technique that allows a running program to be stopped and restarted at specified checkpoints (hence the name).**

---

* See VIZ_DATA_OUTPUT and REACTION_DATA_OUTPUT in Box 4.2. Supported formats for visualization output include: IBM DataExplorer (open source, http://www.research.ibm.com/dx), Pixar RenderMan (proprietary, http://www.pixar.com), Blue Moon Rendering Tools (shareware, adheres to the RenderMan interface standard, http://www.bmrt.org), rayshade (open source, ftp://graphics.stanford.edu/pub/rayshade), povray (open source, http://www.povray.org), irit (open source, http://www.cs.technion.ac.il/~irit).

** At each checkpoint, all the information required to restart is saved in a *checkpoint file*. With MCell simulations, each checkpoint (given as a number of time-step iterations) and checkpoint file name is specified with a keyword statement in the MDL input file (Box 4.2).

One simple use of checkpointing is to subdivide a lengthy simulation into a sequence of shorter runs, and thereby avoid premature termination. A more powerful adaptation is to introduce one or more parameter changes when the simulation restarts. It is also possible to reuse intermediate results saved at different checkpoints. In this way computation time can be reduced dramatically, and simulation conditions can branch from a common point into parallel tracks in which one or more input parameters are varied. Examples of changes that can be incorporated into an MCell checkpoint sequence* include: existing surfaces can be removed and new surfaces can be added; existing surfaces can be moved (e.g., to simulate an expanding exocytotic fusion pore that joins a synaptic vesicle to a presynaptic membrane[30]); surfaces can be changed from reflective to transparent or absorptive, etc.; new ligand molecules and effector sites can be added; new reaction mechanisms can be added; existing reaction mechanisms can be modified; the simulation time-step can be changed; the type and amount of information to be output as results can be changed.

4.3.4 Initialization, Optimized Random Number Generation, and Time-Step Events

Initialization — Figure 4.2A summarizes how an MCell simulation is initialized before time-step execution begins. In essence, initialization includes: (1) instantiation (creation) of pre-existing objects from preceding checkpoint data (if any); (2) instantiation of new simulation objects; and (3) set-up of runtime optimizations. As outlined below and demonstrated in Section 4.5.2, the user must specify runtime optimizations efficiently to run large-scale realistic simulations, because this can decrease the required computer time by *orders of magnitude*.

Random Numbers — Since random numbers are used to make decisions even during initialization (e.g., to place ESs on MEs), a stream of available values is set up when initialization begins (Figure 4.2A). A computer's "random" numbers in reality are "pseudorandom" because they are computed as a deterministic stream using a mathematical algorithm and an initial seed value (for background information on various algorithms, see References 14 and 19). Each number in the stream is actually a sequence of binary bits, and the best algorithms return not just statistically uncorrelated numbers, but also uncorrelated bits within each number. Although most programming languages include built-in functions that compute pseudorandom numbers, the underlying algorithms generally are not adequate to produce uncorrelated bits. MCell therefore uses a self-contained, 64-bit cryptographic-quality algorithm that has been tested at the bit level, and also includes 3000 pre-defined seed values. The advantages are: speed, because the bits in each "single" random number can be subdivided to obtain multiple values for the computational price of one (a random number can be split and used to pick both a distance and direction for a random walk movement, Section 4.4.2); and reproducibility, because

* Modifications to input parameters in successive MDL files used for a checkpoint sequence can be set up by the user in advance, and/or the MDL can be used to introduce incremental parameter changes. In addition, users familiar with operating system script files (e.g., UNIX shell scripts or DOS batch files) can use them for an additional level of flexibility and automation (Figure 4.1).

a given seed produces the same stream of values, and therefore identical simulation results, regardless of the computer platform or operating system.

Time-Step Events — During each simulation time-step, many decisions must be made between different possible events that can occur to *reactant* molecules, where *reactant* includes all existing ligand molecules, bound ligand-ES complexes, and any unbound ESs currently in a chemical state that can undergo a unimolecular reaction transition. A list of all reactants is created when a simulation is initialized (Figure 4.2A), and subsequent time-step events and program flow are summarized in Figure 4.2B. Most computation time for a simulation is consumed by diffusing molecules, because their movement trajectories, or *rays* (a radial distance l_r and direction ξ), must be traced through space to determine whether the ray intersects a ME of a surface. If not, the ligand molecule is simply placed at the endpoint of the ray for the duration of the time-step. If so, the final result depends on the presence or absence of a reactive ES at the point of intersection (P_I), and/or the properties of the ME, as outlined in Figure 4.2B. If the ligand molecule is neither absorbed by the surface nor retained at P_I because of a bimolecular reaction, then its movement must be continued. After passing through or reflecting from the ME (transparent or reflective ME, respectively), the search for an intersection begins again and this process of *ray marching* continues until the molecule is either absorbed, reacts with an ES, or travels a total distance l_r and remains at the resulting endpoint of motion for this time-step.

4.3.5 Additional Run-Time Optimizations: Spatial Partitions and Fast Realistic Random Walk Methods

Spatial Partitions — As illustrated in Figures 4.2 and 4.3, MCell's ray marching algorithms also include a run-time optimization called *spatial partitions*. In the *absence* of partitions, every ME must be checked for a potential intersection every time a diffusing molecule moves. Thus, the computer time for a simulation is roughly proportional to the total number of MEs, i.e., a simulation with 100,000 MEs would require ~1000-fold more computer time than one with 100, and a simulation of one synaptic current at a reconstructed synapse might run for weeks or months. With spatial partitions, however, the number of MEs has very little impact on computer time, and simulations of currents in realistic synaptic structures can run in minutes (Section 4.5.2).

Spatial partitions are transparent planes (Figure 4.3) that the user places along the x, y, and/or z axes* to create spatial subvolumes (SSVs). When the simulation is initialized, the partitions are inserted and the MEs included within each resulting SSV are determined (Figures 4.2A and 4.3). The user's goal is to arrange a sufficient number of partitions so that each SSV includes no more than a few MEs. During time-step iterations, the SSV in which each ligand molecule currently resides is always known (Figure 4.2), so the search for intersections during ray marching can be limited to those MEs included in the ligand molecule's current SSV, as well as

* See PARTITION_X, PARTITION_Y, and PARTITION_Z keyword statements in Box 4.2.

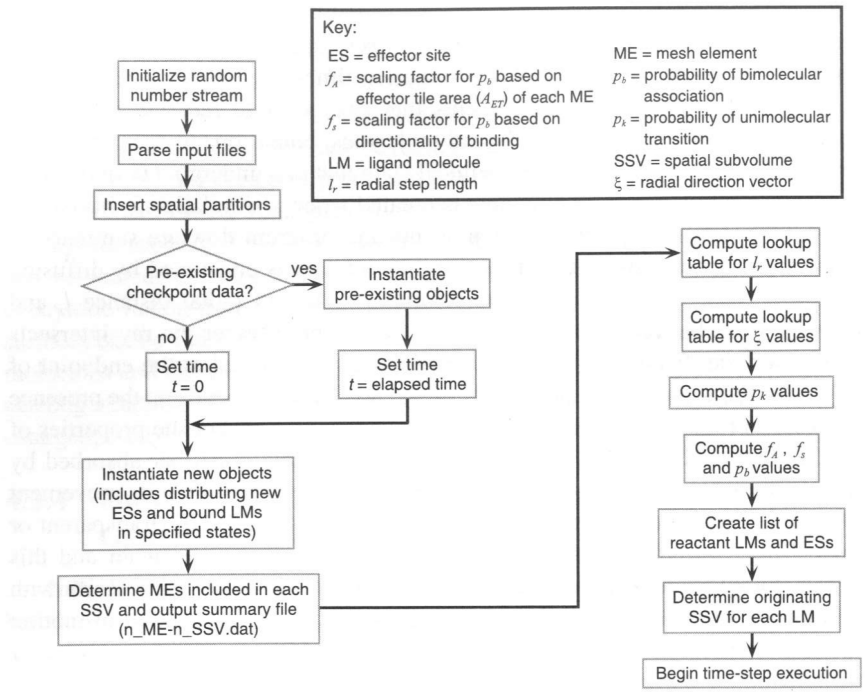


FIGURE 4.2A Flowchart of Initialization Operations. In general, initialization includes set-up of: (1) random numbers; (2) pre-existing and new objects; (3) run-time optimizations such as spatial partitions and random walk look-up tables; (4) calculation of probabilities and scaling factors for reaction transitions; and (5) a reactant list that is traversed during the first time-step (Figure 4.2B). To help the user optimize the placement and number of spatial partitions (Sections 4.3.5 and 4.5.2), MCell writes an output file (n_ME-n_SSV.dat) that can be used to create a frequency histogram for the number of mesh elements included in spatial subvolumes (see Figure 4.5B).

the boundaries of the SSV itself. If a ligand molecule passes through an SSV boundary, ray marching simply continues in the new SSV using an updated list of MEs and boundaries (Figure 4.2B).

As simulations grow in structural complexity, and therefore include an increasing number of MEs, the number of spatial partitions can be increased to keep the average number of MEs per SSV about constant. Under these conditions, *MCell's execution speed will also remain nearly constant, i.e., independent of the number of MEs.* Since partitions influence only the simulation's execution speed, and not the random number stream, net ligand displacements, nor reaction decisions, *simulation results are identical regardless of partitioning.*

Random Walk — While most of the computational expense incurred for diffusing molecules arises from ray marching, another significant component is generation of the random walk rays themselves. The simplest (and least realistic) random walk algorithm uses a constant step length and chooses a positive or negative x, y,

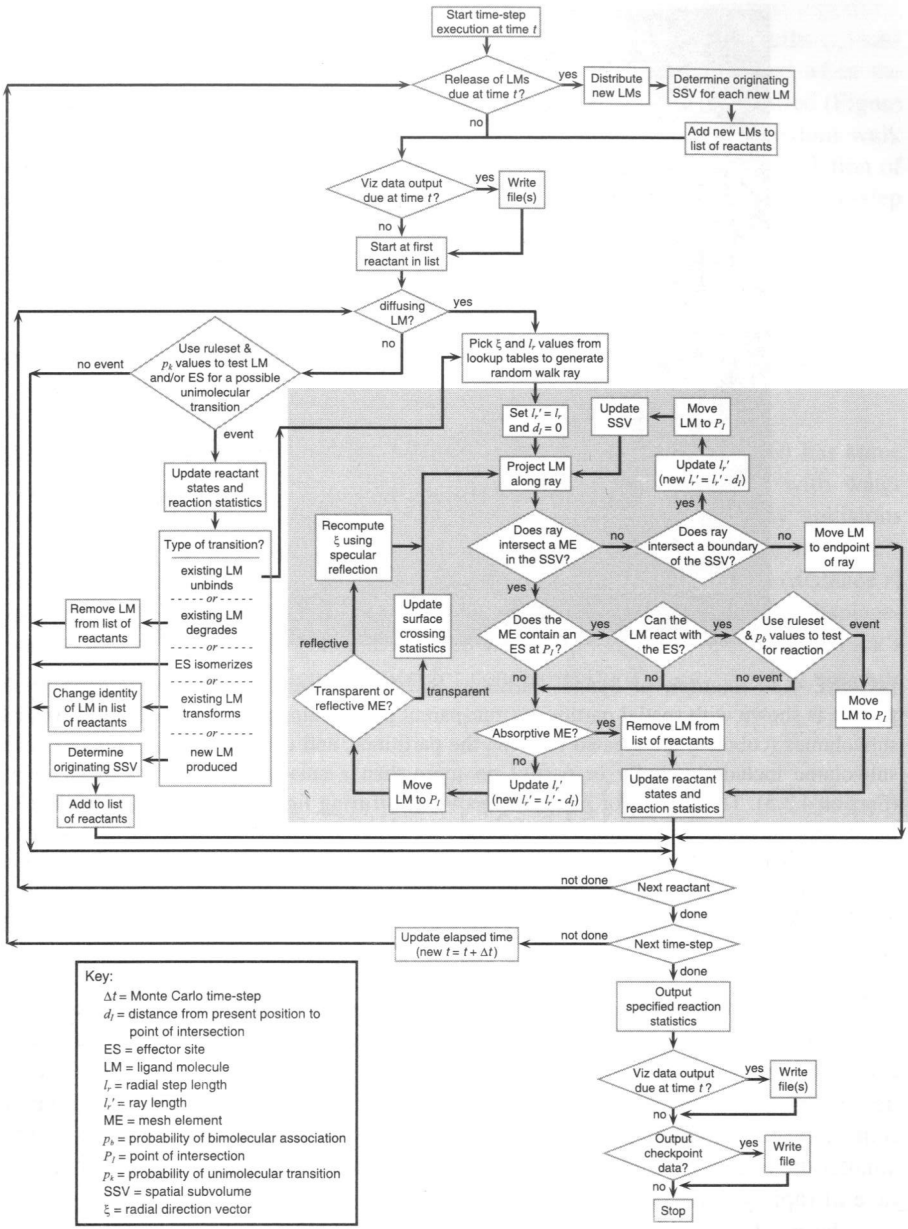


FIGURE 4.2B Flowchart of Time-Step Operations. During each time-step, a list of valid reactants is traversed. Depending on each reactant's identity, random numbers are used either to generate a random walk movement and test for possible bimolecular associations (gray box), or to test for possible unimolecular transitions. Additional operations may include release of new ligand molecules and output of visualization data. Specified reaction statistics and checkpoint data are written to output files after the last iteration.

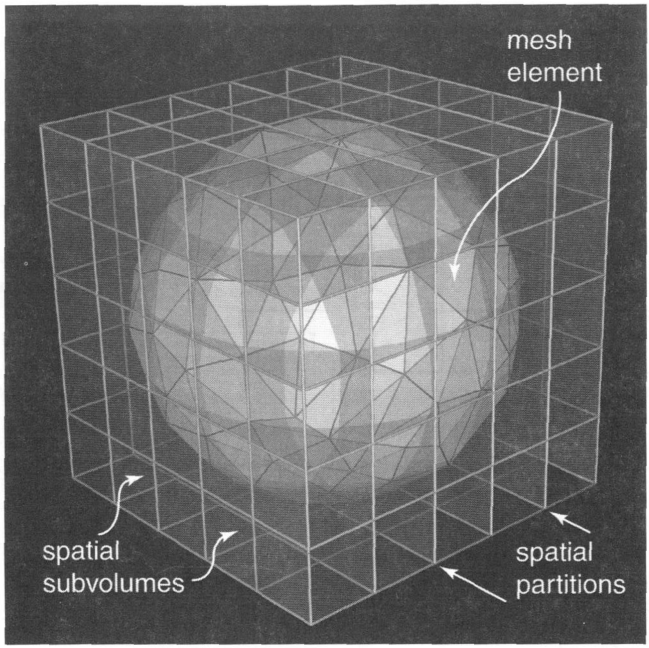


FIGURE 4.3 Example of Spatial Partitions and Subvolumes. A spherical polygon mesh surface is shown with spatial partitions (transparent planes) along the *x*, *y*, and *z* axes. Spatial subvolumes (cuboids) are created between the partitions, and under optimal conditions each subvolume includes (wholly or partly) no more than a small number of mesh elements (Section 4.3.5). The search for collisions between diffusing ligand molecules and mesh elements (ray tracing and marching, Figure 4.2B) then can be restricted to individual subvolumes, dramatically increasing execution speed (Section 4.5.2). As shown for clarity, each partition extends just beyond the dimensions of the sphere. In an actual simulation, however, partitions extend to the bounds of available space ($\sim 10^{151}$ μm $\cong \infty$), and may be placed anywhere along each axis.

and *z* displacement for each movement.[9] Movements are generated rapidly, but molecules occupy positions on a 3-D Cartesian lattice. For accurate simulation of diffusion, the granularity of the lattice must be very fine compared to the size of any restrictive structures through which the molecules move. Fine granularity requires a very small step length, and hence a very small simulation time-step. The smaller the time-step, the more time-step iterations are required, and thus the advantage of rapidly generated movements is negated.

The most realistic random walk algorithm computes a random radial direction and step length for each movement. Because the step lengths are obtained from a theoretical diffusion distribution (Section 4.4.1) and the molecules do not move along a lattice, accurate simulation of fluxes through restrictive structures can be obtained with a longer time-step than is required with simpler algorithms. However, the trigonometric and numerical integration operations required for each movement are very time-consuming, so the overall execution speed is very slow. MCell achieves

the same maximal realism and accuracy with the speed of the simplest algorithm by using extensive look-up tables for equally probable radial step lengths (l_r) and direction vectors (ξ). The values in each table are computed only once when the simulation is initialized (Figure 4.2A), and thereafter are chosen as required (Figure 4.2B) using a subdivided random number (Section 4.3.4). MCell's random walk algorithms are outlined more fully in the following section, as is the calculation of p_b and p_k values used in testing for possible reaction transitions during time-step iterations (Figure 4.2B).

4.4 THEORETICAL FOUNDATION

4.4.1 Analytic Diffusion Theory for Net Radial Displacements

An actual diffusing ligand molecule initially at point P_1 at time ($t = 0$) has some thermal velocity and undergoes Brownian movements (i.e., collides with water molecules) on the sub-ps timescale at room temperature.[4] After many collisions accumulate over a longer time interval ($t = \Delta t$), the difference between the molecule's ending position P_2 and initial position P_1 yields a net radial displacement (distance r) in a random direction. To calculate the theoretical distribution of net radial displacements (that will be used to generate random walk step lengths), we start from Fick's second law, i.e., the diffusion equation for concentration C at any point in space as a function of time:

$$\left(\frac{\partial C}{\partial t}\right)_{x,y,z} = D_L\left[\left(\frac{\partial^2 C}{\partial x^2}\right)_t + \left(\frac{\partial^2 C}{\partial y^2}\right)_t + \left(\frac{\partial^2 C}{\partial z^2}\right)_t\right] \tag{4.1}$$

If a concentration gradient exists in only one dimension (ϑ), whether linear (x, y, or z) or radial (r), Equation 4.1 reduces to:

$$\left(\frac{\partial C}{\partial t}\right)_{\vartheta} = D_L\left(\frac{\partial^2 C}{\partial \vartheta^2}\right)_t, \quad \vartheta = x, y, z \text{ or } r \tag{4.2}$$

which can be solved analytically for certain sets of boundary conditions. For a point source of M molecules and a time interval Δt, the solution is[9]

$$C(r, \Delta t) = \frac{M}{\left(4\pi D_L \Delta t\right)^{3/2}} e^{-r^2/4D_L\Delta t} \tag{4.3}$$

Equation 4.3 gives the theoretical ligand concentration (molecules per unit volume) as a continuous function of time and radial distance (and appears in similar form in Chapter 3). Multiplying Equation 4.3 by the volume of a spherical shell

($4\pi r^2\, dr$) gives the amount of ligand (N_r, number of molecules) contained in the shell at time Δt:

$$N_r = \frac{M}{\left(4\pi D_L \Delta t\right)^{3/2}} e^{-r^2/4D_L\Delta t}\left(4\pi r^2\, dr\right) \tag{4.4}$$

Dividing both sides of Equation 4.4 by the total amount of ligand (M) gives the fractional amount (N_r/M) in the shell, or the *fractional probability* (p_r) of a net radial displacement between r and (r + dr) *for a single diffusing molecule*. Hence:

$$p_r = \frac{1}{\left(4\pi D_L \Delta t\right)^{3/2}} e^{-r^2/4D_L\Delta t}\left(4\pi r^2\, dr\right) \tag{4.5}$$

The mean radial displacement ($\bar{l}_r$) is obtained from the expectation value of r, i.e., by integrating across all probability-weighted values of r, and is:

$$\bar{l}_r = 2\sqrt{\frac{4D_L\Delta t}{\pi}} \tag{4.6}$$

If P_1 is defined as the origin of a Cartesian coordinate system and a set of radial displacements in random directions is evaluated, the average (±) x, y, and z components ($\Delta\bar{x}$, $\Delta\bar{y}$, and $\Delta\bar{z}$) for the set are all equal to ($\bar{l}_\perp$), where:

$$\bar{l}_\perp = \sqrt{\frac{4D_L\Delta t}{\pi}} \tag{4.7}$$

Because of radial symmetry, the same result is obtained for (±) $\Delta\bar{x}$, $\Delta\bar{y}$, and $\Delta\bar{z}$ no matter how the Cartesian axes are rotated around P_1. In general, then, $\bar{l}_\perp$ is the average value of (±) $\Delta\psi$, where ψ is any arbitrary axis that passes through P_1. For a plane at any arbitrary location in space, there exists an axis ψ that is perpendicular (normal) to the plane. Thus, the average increase or decrease in distance between a diffusing molecule and the plane (measured with respect to ψ) is given by $\bar{l}_\perp$. Comparison of Equations 4.6 and 4.7 shows that $\bar{l}_\perp$ is equal to $\bar{l}_r/2$.

4.4.2 Optimized Brownian Dynamics Random Walk

The probability distribution function of Equation 4.5 can be rewritten for a dimensionless parameter s, where s is defined as $r/\sqrt{4D_L\Delta t}$:

$$p_s = \frac{4}{\sqrt{\pi}} s^2 e^{-s^2}\, ds \tag{4.8}$$

When an MCell simulation begins, Equation 4.8 is integrated to obtain a large set of equally probable values of s (1024 under default conditions), and these values

are stored in a look-up table. Thereafter, each time a radial step length l_r is needed for a random walk movement, a random number is used to pick a value of s from the table. To convert from s to l_r, the chosen value of s is multiplied by the scaling factor $\sqrt{4D_L\Delta t}$ (which has units of distance).

A second look-up table is used for an extensive set of equally probable radial directions (>130,000 by default). These directions are stored as the x, y, and z components of unit vectors (ξ) that radiate out from a point and are spaced evenly from each other. An initial subset is computed to fill one octant of a unit sphere, and then is reflected to fill the remaining seven octants. This symmetrical replication of the initial subset guarantees the complete absence of directional bias, which is critical because even a tiny net bias can accumulate over thousands of time-steps to produce substantial drift. Once random numbers have been used to pick l_r and ξ for a random walk ray, the x, y, and z components of ξ are multiplied by l_r. The results then are added to the (x, y, z) values of the molecule's initial position (P_1) to find a possible endpoint of motion (P_2). Finally, the ray is traced and marched as required (Figure 4.2B).

As introduced briefly in Section 4.3.5, simpler random walk algorithms can be designed in many ways, but the MCell method has the following advantages (which also distinguish it from our own earlier algorithm used with MC simulations):[5]

1. Realism — movements are radially symmetrical during each individual time-step, and under default conditions are essentially free of discernible spatial granularity.
2. Speed — no penalty is incurred for realism, because numerous optimizations make it as fast or faster than simpler, less realistic methods.
3. Numerical accuracy — extensive sets of radial distances and directions are available for each movement, so concentration gradients are reproduced accurately within distances as small as twice the mean radial step length $\bar{l}_r$ (Equation 4.6). Thus, to simulate diffusion through a constriction like an exocytotic fusion pore, the time-step Δt need only be chosen so that $\bar{l}_r$ is about twofold smaller than the radius of the pore.[30,33]
4. Adjustability — although the default numbers of radial step lengths (1024) and directions (~130,000) are sufficient for almost any conceivable circumstance, the user can increase or decrease either value, or even replace the look-up table for direction vectors with directions chosen at random for each movement (full machine precision, slower execution).

4.4.3 General Overview of Monte Carlo Probabilities

Monte Carlo binding and unbinding probabilities were first introduced in simulations of a simplified vertebrate neuromuscular junction that contained regular rectangular junctional folds, and therefore simple rectangular grids for binding sites.[5] The simulation program was soon merged with another developed independently at about the same time,[33] and after extensive generalization the earliest version of MCell was created;[30,31] (see References 26 and 33 for a more complete historical overview). Simulation of highly realistic rather than simplified models is now possible after

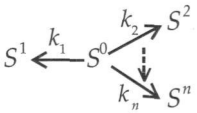
much additional optimization and development. Here we present detailed general derivations of the MC probabilities for bimolecular (p_b) and unimolecular (p_k) reaction transitions (see Chapter 2, Box 2.1 for an overview of reaction mechanisms), as required for the new methods and capabilities. This is important but difficult ground for newcomers, and luckily the practical "take-home" message can be summarized simply: *For a given set of input conditions, simulation results will converge to the correct answer as a single input parameter, the time-step Δt, decreases.* Unlike FE simulations, there is no separate parameter (i.e., voxel size and shape) that determines the simulation's spatial granularity. Instead, *spatial granularity decreases hand-in-hand with decreasing temporal granularity, because random walk distances become smaller as Δt becomes smaller.*

Each time that a reactant molecule is tested for possible chemical transitions (Figure 4.2B), the value of a single random number (κ) is compared to the relevant MC probability value computed during initialization (Figure 4.2A). Each probability value is subdivided into fractional amounts if the reactant has more than one available transition path. For example, in the reaction mechanisms of Figure 4.1:

1. A random walk ray for a diffusing A molecule might intersect ("hit") an ES in the R^0 state, and A then would be tested for binding to either of two independent binding sites defined by parallel transition paths and the bulk solution rate constants k_{+1} and k_{+2}. Each of the two rate constants would be used to compute a fraction of p_b, the MC probability for bimolecular association (see below). In general, each ES can represent a molecule with an *arbitrary number of binding sites.*
2. A ray might hit an ES in the E state, which has a single available binding site, and hence a single rate constant would be used to compute p_b.
3. An ES in the A_2R^3 state might be tested simultaneously for unbinding from either binding site, or isomerization to the (A_2R^4) state. The three rate constants k_{-3}, k_{-4}, and β would each be used to compute a fraction of p_k, the MC probability for a unimolecular transition.

4.4.4 The MC Probability of Unimolecular Transitions

Derivation — As illustrated in Figures 4.1 and 4.2B, MCell simulations presently can include five different types of unimolecular transition. In each case (e.g., the $A_2R^3 \rightarrow A_2R^4$ isomerization in Figure 4.1), the transition is governed by a first-order rate constant k that has units of inverse time (s^{-1}). In general, if initial state S^0 can undergo one of n different possible transitions:



(4.9)

and the reaction proceeds for some time Δt, the total probability (p_{kt}) that a *single molecule in the S^0 state undergoes any transition* is given by (Box 4.3):

$$p_{kt} = 1 - \exp\left[-\left(\sum_{1}^{n} k_i\right)\Delta t\right] \tag{4.10}$$

and the fractional probabilities of each individual transition are:

$$p_{k1} = p_{kt} \cdot \frac{k_1}{\sum_{1}^{n} k_i}, \ldots p_{kn} = p_{kt} \cdot \frac{k_n}{\sum_{1}^{n} k_i}; \quad \sum_{1}^{n} p_{ki} = p_{kt} \tag{4.11}$$

Implementation, Validation, and Accuracy — Since p_{kt} is the probability of any transition during Δt, the probability of no transition is $(1 - p_{kt})$. The decision between all possible events (including no transition) is made with maximal run-time efficiency by comparing the value of a single random number ($0 \leq \kappa \leq 1$) to the cumulative set of probabilities (p_{k1}, $p_{k1} + p_{k2}$, ..., p_{kt}, 1).*

MCell results for unimolecular transitions can be verified with a simple simulation of Equation 4.9, i.e., one that starts with a set of ESs in state S^0, and then tallies:** (1) the number of all transitions from S^0 per time-step Δt, to verify the expected exponential distribution of lifetimes for the S^0 state (with a mean value of τ, Box 4.3); and (2) the number of ESs in each state after each time-step, to verify the expected proportions (Equation 4.11). For high numerical accuracy, the value of Δt used for the simulation must be small compared to τ. Of course, the MCell output will also include stochastic noise, the magnitude of which will depend on the absolute number of ESs. Such noise can be reduced by averaging across multiple simulations run with different random number seeds, and will decrease by a factor $1/\sqrt{n_s}$, where n_s is the number of simulations.

4.4.5 The Momte Carlo Probability of Bimolecular Associations

Derivation — For a simulation of bimolecular association between ligand A and receptor R with n possible binding sites:

$$\begin{array}{l} \xrightarrow{k_{+1}} \mathrm{AR}^1 \\ \mathrm{A} + \mathrm{R} \quad \vdots \\ \xrightarrow{k_{+n}} \mathrm{AR}^n \end{array} \tag{4.12}$$

fractional values of p_b are used to test for binding to any one of the available sites each time a random walk ray hits an ES in state R (Figure 4.2B). The total number

* If $\kappa \leq p_{k1}$, then the first possible transition occurs. If $p_{k1} < \kappa \leq (p_{k1} + p_{k2})$, then the second possible transition occurs, and so on. If $\kappa > p_{kt}$, then no transition occurs.

** See REACTION_DATA_OUTPUT in Box 4.2 and COUNT statements in Sections 4.5 and 4.6.

of times that a *particular* ES is hit during a time-step Δt can range from zero to any (+) integer value, depends on the local *concentration-dependent* flux of A molecules into the ES, and shows stochastic variability across successive trials. The *average* number of hits per time-step (N_H) thus is a (+) *real number that approaches zero as either* Δt *or the concentration of A approaches zero.*

Box 4.3 Derivation of the MC Probability for Unimolecular Transitions

If the reaction of Equation 4.9 (Section 4.4.4) proceeds for any time t from an initial concentration $(S^0)_o$, the total probability (p_{kt}) that a *single molecule in the* S^0 *state undergoes a transition* is given by the *fraction of* $(S^0)_o$ *that undergoes any transition during time t*:

$$p_{kt} = \frac{(S^1)_t + (S^2)_t + \ldots (S^n)_t}{(S^0)_o} = 1 - \frac{(S^0)_t}{(S^0)_o} \tag{1}$$

The general rate equation depends only on time and (S^0):

$$\begin{aligned} -d(S^0) &= d(S^1) + d(S^2) + \ldots d(S^n) \\ &= (k_1 + k_2 + \ldots k_n)(S^0)dt = \left(\sum_1^n k_i\right)(S^0)dt \end{aligned} \tag{2}$$

and hence can be integrated directly to obtain p_{kt}. From Equation 4.2:

$$\int_{(S^0)_o}^{(S^0)_t} \frac{d(S^0)}{(S^0)} = -\left(\sum_1^n k_i\right)\int_0^t dt \tag{3}$$

and the solution is

$$\frac{(S^0)_t}{(S^0)_o} = \exp\left[-\left(\sum_1^n k_i\right)t\right] \tag{4}$$

From Equation 4 the lifetime of S^0 is exponentially distributed with a mean value

$$\tau = 1/\sum_1^n k_i \tag{5}$$

From Equations 4 and 1:

$$p_{kt} = 1 - \exp\left[-\left(\sum_1^n k_i\right)t\right] \tag{6}$$

as given in Equation 4.10 (Section 4.4.4).

Since p_b is the probability that binding *occurs* at any one of the available sites after a *single* hit, the probability that binding *does not occur* after a *single* hit is $(1 - p_b)$. The probability that binding has not occurred *after a total of* N_H *hits* is $(1 - p_b)^{N_H}$, and thus the total probability (p_{bt}) that binding *has occurred* during Δt, i.e., *after any one of the* N_H *hits*,* is given by:

$$p_{bt} = 1 - (1 - p_b)^{N_H} \tag{4.13}$$

In order for the binding kinetics of a MC simulation to be quantitatively correct, *the average instantaneous binding rate must equal the idealized binding rate predicted by mass action kinetics*. Therefore, the value of p_{bt} must equal a corresponding probability (p_t) calculated from mass action rate equations. For a short interval of time Δt:

$$p_t \cong \zeta = \left(\sum_1^n k_{+i} \right) (A)_o \, \Delta t \tag{4.14}$$

where $(A)_o$ is the local concentration of ligand molecules around a single R molecule at the beginning of Δt (Box 4.4). To calculate p_b for use in MC simulations, Equation 4.13 thus is set equal to Equation 4.14 (see Box 4.5):

$$1 - \left(1 - p_b\right)^{N_H} = p_t \cong \zeta = \left(\sum_1^n k_{+i} \right) (A)_o \, \Delta t \tag{4.15}$$

and then Equation 4.15 must be solved for p_b. Doing so directly yields a ligand concentration-dependent expression for p_b, which would make p_b a space- and time-dependent parameter specific to each ES during a running MC simulation. The resulting computational cost would be staggering. Fortunately, however, as Δt approaches 0, p_b, and N_H must also approach 0, and under such conditions the term $(1 - p_b)^{N_H}$ in Equation 4.15 approaches $(1 - N_H \cdot p_b)$.** After substitution and rearrangement:

$$p_b = \left(\sum_1^n k_{+i} \right) \frac{(A)_o \, \Delta t}{N_H} \quad ; \text{for small } \Delta t \tag{4.16}$$

---

* A concrete analogy to this probability problem goes as follows: Roll a six-sided die 3 (i.e., N_H) times. What is the probability that a 1 *is not* obtained on *any* of the three trials? The probability of rolling a 1 (i.e., p_b) is 1/6, and so the probability of *not* rolling a 1 is (1 – 1/6 = 5/6) for the first trial *and* the second trial *and* the third trial. Thus, the probability that a 1 is not obtained within three rolls is $(5/6) \cdot (5/6) \cdot (5/6)$, or $(5/6)^3$, i.e., $(1 - p_b)^{N_H}$. The probability that a 1 *will be* rolled (i.e., binding will occur) within three trials therefore must be $1 - (5/6)^3$, or $1 - (1 - p_b)^{N_H}$ as given in Equation 4.13.

** At the limit of $\Delta t = 0$, ligand molecules make no movements at all, and hence no hits can occur. With $N_H = 0$, both $(1 - p_b)^{N_H}$ and $(1 - N_H \cdot p_b)$ evaluate to unity. For small non-zero values of N_H, the agreement between the two terms is especially close if p_b is also small (and p_b, like N_H, decreases as Δt decreases). For example, if both N_H and p_b are 0.1, the two terms agree to within 0.05%.

As mentioned above, the term N_H in Equation 4.16 is determined by the local concentration-dependent flux (J) of A molecules into one ES. For small Δt, J (and therefore N_H) is directly proportional to $(A)_o$, and after substitution of a final expression for N_H (Box 4.6) into Equation 4.16:

$$p_b = \left(\sum_1^n k_{+i}\right)\frac{1}{2(N_a)(A_{ET})}\left(\frac{\pi\Delta t}{D_L}\right)^{1/2} \tag{4.17}$$

where N_a is Avogadro's number, A_{ET} is the area of an ET, and D_L is the diffusion coefficient of ligand A. The expression for p_b in Equation 4.17 is independent of free ligand concentration, and therefore can be implemented very efficiently in simulations.

Box 4.4 Bimolecular Associations: Derivation of ζ

If the reaction of Equation 4.12 (Section 4.4.5) proceeds for some time t from initial concentrations $(A)_o$, $(R)_o$, and $(AR)_o$, the mass action probability (p_t) that a *single R molecule becomes bound* is given by the fraction of $(R)_o$ that becomes bound, i.e.,

$$p_t = \frac{\sum_1^n\left[\left(AR^i\right)_t - \left(AR^i\right)_o\right]}{(R)_o}, \tag{1}$$

or simply

$$\frac{\sum\left(AR^i\right)_t}{(R)_o} \tag{2}$$

if

$$\sum = \sum_1^n \text{ and } \sum\left(AR^i\right)_o = 0 \tag{3}$$

(as used below). The general rate equation for production of the n bound states is

$$\partial\left[\sum\left(AR^i\right)\right] = -\partial(A) = -\partial(R) = \left(\sum k_{+i}\right)(A)(R)\partial t \tag{4}$$

(continued)

Box 4.4 (continued)

but Equation 4 cannot be integrated directly to obtain (AR)t because (A), (R), and (AR) are functions of space as well as time. In Equation 4, the quantity ($D_A + D_R$), i.e., the sum of the diffusion coefficients for A and R, is implicitly included in the values of k_+, together with the sizes and shapes of the molecules, and the activation energy for each binding reaction. If at least one of the diffusion coefficients is large but the k_+ values are small (e.g., because the activation energy is large), then the rate of reaction is not "diffusion-limited," i.e., the solution is always "well-mixed" because the rate of binding is slow compared to the rate of diffusion. Under such conditions, appreciable spatial concentration gradients do not form as binding proceeds, so the partial differentials of Equation 4 can be replaced with ordinary differentials:

$$d\left[\sum\left(AR^i\right)\right] = -d(A) = -d(R) = \left(\sum k_{+i}\right)(A)(R)dt \tag{5}$$

Since the concentration terms in Equation 5 are independent of space, by definition they are *equally valid* for the bulk solution and at the *local level in the vicinity of single molecules*. Equation 5 can be integrated to determine $(AR)_t$:

$$\int_{\sum(AR^i)_o}^{\sum(AR^i)_t} d\left[\sum\left(AR^i\right)\right] = \left(\sum k_{+i}\right)\int_o^t (A)(R)dt = \left(\sum k_{+1}\right)\int_o^t \left(A_o - \sum AR^i\right)\left(R_o - \sum AR^i\right)dt \tag{6}$$

After integration, final analytic expressions for p_t are

$$p_t = \frac{\sum\left(AR^i\right)_t}{(R)_o} = \frac{1}{(R)_o}\left(\frac{\left(b^2+q\right)\left(\exp\left[\left(\sum k_{+1}\right)\cdot t\sqrt{-q}\right]-1\right)}{2\left(b+\sqrt{-q}-\left(b-\sqrt{-q}\right)\exp\left[\left(\sum k_{+i}\right)t\sqrt{-q}\right]\right)}\right) \tag{7}$$

$$\left(\text{if } (A)_o \neq (R)_o \text{ ; where } b = -\left(A_o + R_o\right) \text{ and } q = 4(A_o)(R_o) - b^2\right)$$

or

$$p_t = \frac{\sum\left(AR^i\right)_t}{(R)_o} = \frac{1}{(R)_o}\left(\frac{\left(\sum k_{+i}\right)(A)_o^2 t}{1+\left(\sum k_{+i}\right)(A)_o t}\right) = \frac{\left(\sum k_{+i}\right)(A)_o t}{1+\left(\sum k_{+i}\right)(A)_o t} \tag{8}$$

$$\left(if (A)_o = (R)_o\right)$$

(continued)

Box 4.4 (continued)

From Equations 7 and 8, the probability (p_t) that a single R molecule becomes bound during an arbitrarily long interval of time (t) depends on all the k_+ values, $(A)_o$, and $(R)_o$; $p_t = 0$ for $t = 0$, and for $t = \infty$, $p_t = 1$ if $(A)_o \geq (R)_o$, or $p_t = (A)_o/(R)_o$ if $(A)_o < (R)_o$.

If the interval of time is very short, so that $t = \Delta t$ and $(AR^i)_{\Delta t}$ is much less than both $(A)_o$ and $(R)_o$, then

$$\left(A_o - \sum\left(AR^i\right)_{\Delta t}\right) \cong (A)_o \text{ and } \left(R_o - \sum\left(AR^i\right)_{\Delta t}\right) \cong (R)_o. \tag{9}$$

Equation 4.3 then becomes:

$$\int_{\sum(AR^i)_o}^{\sum(AR^i)_{\Delta t}} d\left[\sum\left(AR^i\right)\right] \cong \left(\sum k_{+i}\right)(A)_o(R)_o \int_o^{\Delta t} dt \tag{10}$$

and after integration:

$$\sum\left(AR^i\right)_{\Delta t} \cong \left(\sum k_{+i}\right)(A)_o(R)_o \Delta t \tag{11}$$

Thus, for a short interval of time Δt:

$$p_t = \frac{\sum\left(AR^i\right)_{\Delta t}}{(R)_o} \cong \zeta = \left(\sum k_{+i}\right)(A)_o \Delta t \tag{12}$$

where $(A)_o$ is both the instantaneous local concentration of ligand molecules in the immediate vicinity of a single R molecule, and the average bulk solution ligand concentration.

If the MEs of a surface have different shapes and sizes (which is generally the case for reconstructions), the value of A_{ET}, and hence p_b, is different for each ME. The exact values of A_{ET} are determined at the time of barycentric subdivision, and are slightly smaller than the approximate value, A_{EG} (which itself is the inverse of σ_{EG}, the global effector grid density; Section 4.3.2 and Box 4.2). A factor

$$f_A = \frac{A_{EG}}{A_{ET}} = \frac{1}{\sigma_{EG} \cdot A_{ET}} \tag{4.18}$$

is calculated for each ME during initialization (Figure 4.2A), and a final expression for p_b is:

$$\begin{aligned} p_b &= \sum_1^n p_{bi} = \sum_1^n \left(f_{si} \cdot k_{+i}\right) \cdot X = f_{s1} \cdot k_{+1} \cdot X + \ldots + f_{sn} \cdot k_{+n} \cdot X; \\ X &= \left(\frac{f_A \cdot \sigma_{EG}}{2 \cdot N_a}\right)\left(\frac{\pi \Delta t}{D_L}\right)^{1/2} \end{aligned} \tag{4.19}$$

where the term f_{si} is used to account for the polarity of binding to each of the n possible binding sites (as specified in the reaction mechanism for the ligand molecule and ES; see Sections 4.5 and 4.6). If binding can occur after a hit to either side of the ET, then both "poles" of the ES are valid surface area for binding and the value of f_{si} is unity. If binding can only occur after a hit to one particular side of the ET, then only the "positive" or "negative" pole of the ES is valid surface area, and f_{si} is 0 if the invalid pole is hit, or 2 if the valid pole is hit. The value of 2 in the latter case compensates for the apparent twofold reduction in N_H.

Box 4.5 Bimolecular Associations: Equating p_{bt} to p_t

Equation 12 in Box 4.4 was derived under conditions in which concentration gradients are absent at all times. If this is not true, the expression for p_t still holds in small local regions where the concentration change is negligible, but on the macroscopic scale Equation 4 in Box 4.4 must be evaluated numerically using either finite element or MC methods. With a finite element approach, space is subdivided into small volume elements (voxels), and the contents of each voxel are assumed to be well mixed. The rate of reaction within each voxel is computed using an ordinary differential equation like Equation 5 in Box 4.4, and the flux between voxels is computed from concentration differences and the values of D_A and D_R. With MCell's MC algorithms, ligand fluxes and binding depend only on random walk movements and intersections with individual ESs, and hence are *always local* on the spatial scale of the random walk and the temporal scale of Δt (see Box 4.6). The binding probability p_b is calculated by equating p_{bt} to $p_t \cong \zeta$ (Equation 4.15, Section 4.4.5), so the MC binding rate satisfies the requirements of mass action kinetics at all points in space. If, for a particular geometric arrangement of ESs, the effective rate of ligand diffusion happens to be fast compared to the rate of binding, the formation of concentration gradients during binding will be negligible and the MCell simulation will match the analytic prediction of Equations 7 or 8 in Box 4.4. If the relative rate of diffusion is not fast, however, the rate of binding in the simulation will depend on both space and time, as idealized in Equation 4 in Box 4.4 (rather than Equations 7 or 8) for an infinite number of molecules.

Box 4.6 Bimolecular Associations: Derivation and Use of N_H to Compute p_b

In principle, the instantaneous flux of ligand molecules into a surface ("hits" per unit time) depends on the net velocity (v, distance per unit time) of the molecules *toward* the surface, the surface's area (A_s), and the instantaneous ligand concentration (A) adjacent to the surface:

$$\frac{\text{hits}}{\partial t} = J = \frac{1}{2}(N_a)(v)(A_s)(A) \qquad (1)$$

(continued)

Box 4.6 (continued)

where N_a is Avogadro's number, and the factor (1/2) accounts for the fraction of molecules that have a net velocity *away* from the surface. The number of hits during an interval of time t is obtained by integrating Equation 4.1:

$$(\text{number of hits})_t = \int_o^t J\,\partial t = \frac{1}{2}(N_a)(v)(A_s)\int_o^t (A)\partial t \tag{2}$$

In an MCell simulation, ESs occupy ETs on MEs, and each side (front and back) of an ET has some area, A_{ET}. Thus, A in Equation 2 is $2 \cdot A_{ET}$. The average radial distance traveled by each ligand molecule during a time-step Δt is $\bar{l}_r$ (Equation 4.6, Section 4.4.1), but the average net displacement ($\Delta\bar{\psi}$) toward or away from any ME (a portion of a plane) is $\bar{l}_\perp$ (Equation 4.7, Section 4.4.1). Therefore, half of all ligand molecules move toward the ME with an apparent velocity of $v = \bar{l}_\perp/\Delta t$. Substituting into Equation 2:

$$N_H = \frac{1}{2}(N_a)(\bar{l}_\perp/\Delta t)(2\cdot A_{ET})\int_o^t (A)\partial t = (N_a)(\bar{l}_\perp/\Delta t)(A_{ET})\int_o^t (A)\partial t \tag{3}$$

As Δt decreases, the distance $\bar{l}_\perp$ decreases, and hence the population of ligand molecules that can reach the ET becomes increasingly restricted to the local region of space adjacent to the ET. This reduces the sampling of any static or changing concentration gradients in the region. Therefore, as Δt decreases, the flux of A into the ET is determined by a concentration (A) that approaches the instantaneous concentration $(A)_o$ adjacent to the surface, and Equation 4.3 becomes:

$$\begin{aligned} N_H &= (N_a)(\bar{l}_\perp/\Delta t)(A_{ET})(A)_o\int_o^{\Delta t} dt \\ &= (N_a)(\bar{l}_\perp/\Delta t)(A_{ET})(A)_o\Delta t; \quad \text{for small } \Delta t \end{aligned} \tag{4}$$

The $(A)_o$ term in Equation 4 has the same meaning as $(A)_o$ in Equation 4.16 (Section 4.4.5), and thus the two terms cancel when Equation 4 is substituted into Equation 4.16 to obtain p_b:

$$p_b = \left(\sum_1^n k_{+i}\right)\frac{1}{(N_a)(\bar{l}_\perp/\Delta t)(A_{ET})} \tag{5}$$

After Equation 4.7 (Section 4.4.1) is used to substitute for $\bar{l}_\perp$ in Equation 5:

$$p_b = \left(\sum_1^n k_{+i}\right)\frac{1}{2(N_a)(A_{ET})}\left(\frac{\pi\Delta t}{D_L}\right)^{1/2} \tag{6}$$

as given in Equation 4.17 (Section 4.4.5). Thus, p_b depends on $\sqrt{\Delta t}$ (see Box 4.7).

Box 4.7 Bimolecular Associations: Dependence of p_b on $\sqrt{\Delta t}$

The dependence of p_b on $\sqrt{\Delta t}$ arises because ζ (Equation 4.15, and the numerator of Equation 4.16, Section 4.4.5) is directly proportional to Δt, but N_H (denominator of Equation 4.16, Section 4.4.5) is proportional to $\sqrt{\Delta t}$. If both ζ and N_H were linearly dependent on Δt, p_b would be independent of the simulation time-step. The dependence of N_H on $\sqrt{\Delta t}$ arises because the apparent velocity of ligand molecules toward ESs is inversely proportional to $\sqrt{\Delta t}$. Over time Δt, where Δt is longer than the time between actual Brownian collisions (sub-ps scale at room temperature), a real diffusing molecule follows some tortuous path between a starting position P_1 and ending position P_2. The *total* path length is determined by the molecule's real thermal velocity and scales directly with Δt, but the *average radial distance* ($\bar{l}_r$) between P_1 and P_2 scales with $\sqrt{\Delta t}$, as does the *average axial displacement* ($\bar{l}_\perp$) measured with respect to any plane's normal axis ψ (see discussion of Equations 4.6 and 4.7, Section 4.4.1). Since a random walk approximation of Brownian motion replaces the molecule's actual tortuous path with straight-line radial movement between P_1 and P_2, the apparent velocity of motion toward any plane (distance traveled per unit time;

$$v = \bar{l}_\perp / \Delta t = \sqrt{4D_L / (\pi \Delta t)}\ )$$

is less than the real thermal velocity and changes as $1/\sqrt{\Delta t}$.

Implementation, Validation, and Accuracy — Since p_b gives the probability that binding occurs to any one of n binding sites, $(1 - p_b)$ gives the probability that no binding occurs. As discussed previously for unimolecular transitions, the decision between all possible events is made by comparing the value of a single random number ($0 \leq \kappa \leq 1$) to the cumulative set of probabilities (p_b, $p_{b1} + p_{b2}$, ..., p_b, 1).

Equation 4.19 shows how p_b depends on the value(s) of k_+ and three additional user-specified input parameters; p_b is directly proportional to k_+ and σ_{EG}, inversely proportional to $\sqrt{D_L}$, and directly proportional to $\sqrt{\Delta t}$, (Box 4.7). The parameter σ_{EG} is usually set to a fixed value such as the maximum packing density of receptor proteins in plasma membrane (e.g., $10–15 \times 10^3\ \mu m^{-2}$), and k_+ and D_L ordinarily are constrained by experimental or theoretical estimates. The user thus can vary Δt to adjust the value of p_b, which is one determinant of a simulation's numerical accuracy.

In general, Δt should be chosen so that $p_b << 1$, or, as a rule of thumb, not greater than ~0.5 to obtain errors less than 1–2% (relative amounts of bound and unbound reactant states). The user must be aware that an inappropriately long time-step can cause p_b to exceed 1, in which case numerical errors can be quite large in favor of unbound reactants. If k_+ is increased and/or D_L is decreased appreciably (e.g., during a search of parameter space to fit experimental data), the resulting increase in p_b may require a concomitant decrease in Δt to maintain accuracy. Also, if the free ligand concentration, and therefore N_H, varies in time and space (e.g., during ligand release from a synaptic vesicle), then (in principle) the numerical

accuracy varies in time and space for a given value of p_b. This follows from the approximation used between Equations 4.15 and 4.16, i.e., that $(1 - p_b)^{N_H}$ approaches $(1 - N_H \cdot P_b)$ as N_H and p_b decrease with decreasing Δt. For reasonable values of Δt, the overall effect of free ligand concentration on accuracy is likely to be very small. If necessary, however, checkpointing can be used to change Δt during the course of the simulation. Such an adaptive time-step would be shorter during those times when free ligand concentration is high, thus maintaining accuracy.

The validity of Equation 4.19 and MCell's binding algorithms can be tested by comparing a simple simulation of Equation 4.12 to the time course of binding predicted by mass action rate theory (Equations 7 or 8 in Box 4.4). For example, if the ligand molecules and ESs are distributed uniformly to simulate well-mixed conditions, the MCell results will converge to the analytic expectation as Δt is decreased. On the other hand, a simulation run with arbitrary non-uniform ligand molecule and/or ES distributions will converge to results that accurately reproduce the influence of diffusion on the time course of binding.[33] In general, this type of result cannot be obtained analytically, and is the typical aim of MCell simulations (see Box 4.5).

4.4.6 The Heuristics of Reversible Reactions and Multiple Binding Sites

If a reaction mechanism includes only irreversible transitions, and ESs can bind only a single ligand molecule, each reactant can undergo no more than one transition per Δt. Thus, the MCell reaction algorithms (Figure 4.2B) are strictly first-order under such conditions. With reversible reactions and/or multiple binding sites per ES, however, the algorithms are higher-order, i.e., individual molecules may undergo multiple "sub-Δt" transitions during each time-step. Thus, some degree of hidden reversibility is introduced. For example, an ES initially in the AR^1 state (Figure 4.1) might unbind to reach the R^0 state at some point during a time-step, and then later during the same time-step might bind 1 or 2 ligand molecules to begin the next iteration in the AR^1, AR^2, or A_2R^3 state.

In principle, first-order and higher-order algorithms converge to the same result as Δt decreases, but if the higher-order approach can be suitably balanced for complex cyclic reactions, its advantage is higher numerical accuracy for a given value of Δt. Different higher-order approaches can be tested by simulating simple and complex reactions at equilibrium, and then comparing the fractional amounts of each reactant to analytic predictions or to a finite difference simulation of the corresponding rate equations. As indicated in Figure 4.2B, MCell's algorithms use an optimized set of rules to make decisions regarding sub-Δt transitions, and additional details can be found in Reference 33.

4.5 EXAMPLE MODEL: ACETYLCHOLINE EXOCYTOSIS AND MINIATURE ENDPLATE CURRENT GENERATION AT A REALISTIC NEUROMUSCULAR JUNCTION

4.5.1 Logistics of Design, Input, and Output

Medium- to large-scale reconstructions and simulations of single synapses or neuropil can be used to investigate pre- and postsynaptic factors that contribute to synaptic variability. As one example, Figure 4.4A shows a realistic medium-scale model of rat diaphragm neuromuscular junction. The model consists of pre- and postsynaptic membrane surfaces, a transparent basal lamina layer that follows the postsynaptic contour inside the cleft space, and 30 ACh vesicles arrayed above the openings into six junctional folds that have highly variable topology. The size and time course of simulated miniature endplate currents (mEPCs) varies significantly from one release site to another.[33]

As illustrated for the postsynaptic membrane in Figure 4.4B, each of the surfaces in the model is a polygon mesh that has been optimized for MCell simulations and visualization, i.e., each ME is a nearly equilateral triangle.[33] It is important that the front and back faces of each ME are oriented consistently so that the polarity of ESs on the surface is also consistent. Figure 4.4C shows how the right-hand-rule is applied to MEs as they are imported from the MDL file, to distinguish the front from the back face. In this model, the total number of MEs for nerve, muscle, and basal lamina surfaces exceeds 54,000. For large-scale synaptic reconstructions presently in development, this number can easily increase by more than an order of magnitude.

Figure 4.4D shows the MEs that comprise the muscle membrane, together with glyphs that indicate ESs for acetylcholine receptors (AChRs) on the surface (~91,000). As discussed in Section 4.3.2, each ME has been covered with ETs (not shown) using barycentric subdivision and a global effector grid density (σ_{EG}) that in this case is 10,000 μm^{-2}. The position and area of each ET ($A_{ET} \cong 1/\sigma_{EG} = 100$ nm^2) is indicated approximately by the position and size of the AChR glyphs (8.5 nm diameter). The membrane surface is actually composed of three different meshes that fit together *exactly* (Section 4.3.2), i.e., a top, middle and bottom portion with respect to the depth of the folds (muscle-membrane.mdl), and AChR ESs are present on the top two parts at different site densities. On the tops of the folds, the site density is 7250 μm^{-2} (~66,000 AChRs), and in the middle region is about 70% less (~25,000 AChRs). Thus, about 70% or 20% of all ETs are occupied by ESs in the top and middle regions, respectively, as can be seen in Figure 4.4D by the relative spacing of glyphs located either above or down inside the fold. Aside from AChR ESs, the model also includes ~59,000 AChE sites distributed throughout the basal lamina (1800 μm^{-2}), and ~5800 choline (Ch) reuptake sites (ChRs) in the nerve membrane (1000 μm^{-2}). The ChRs bind Ch molecules inside the cleft (where they

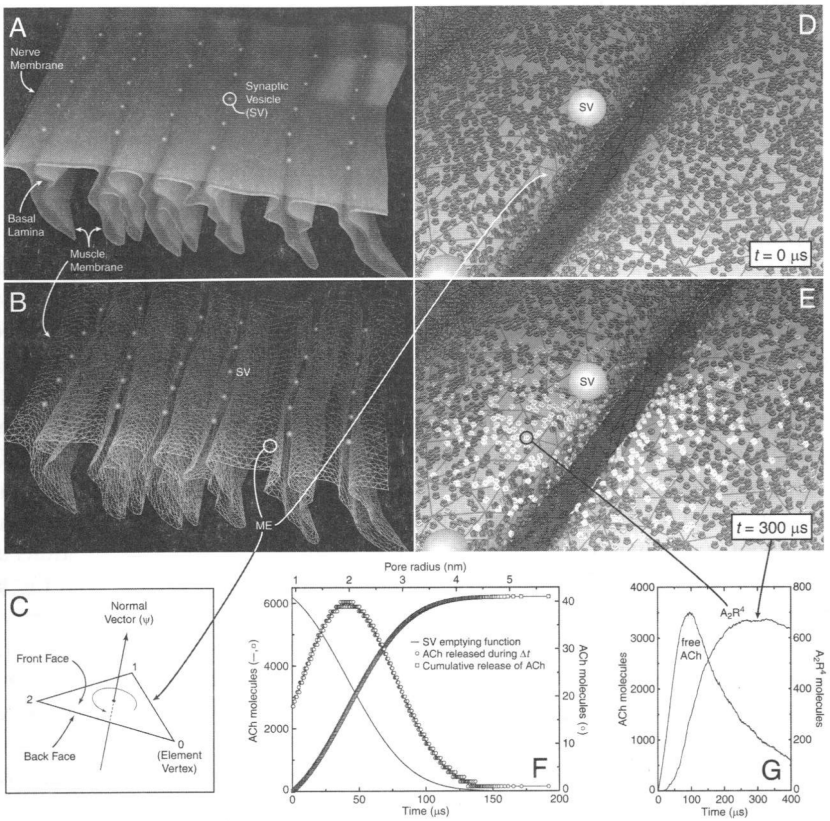


FIGURE 4.4 Realistic Neuromuscular Junction: Model Design and Acetylcholine (ACh) Exocytosis. (A) Model surfaces, and possible locations of ACh release indicated by synaptic vesicles. (B) Optimized polygon mesh for the muscle membrane surface. The entire mesh is composed of individual mesh elements (MEs) that are nearly equilateral triangles. (C) Each ME is defined by its vertex positions and an ordered list of connections between the vertices. This diagram shows how the right-hand-rule is applied to the list of connections to determine the front and back faces of the ME. The first connection extends from vertex 0 to vertex 1 and so on (vertex numbering is arbitrary), and the normal vector (ψ) defined by the right-hand-rule passes through the ME from back to front. (D) Closer view of muscle membrane with nerve and basal lamina surfaces removed. Individual MEs are subdivided into effector tiles, and each tile may contain an effector site as indicated by glyphs that represent acetylcholine receptors (AChRs, see text). (E) Snap-shot of muscle membrane during the simulation, showing bound AChRs (lighter gray) close to the point of ACh release (for more detailed views and explanation, see Color Figures 4.1–4.3). (F) The time course of SV emptying during ACh exocytosis is shown by the solid curve (see text). The corresponding time course of ACh appearance in the cleft can be simulated using incremental release during time-steps (circles), to obtain the desired cumulative release (squares). (G) Example of output from one simulation, showing the amount of free ACh in the entire synaptic cleft, and the number of AChRs in the double-bound open conformation (A_2R^4 state). Panels A, B, D, and E were rendered with IBM DataExplorer.

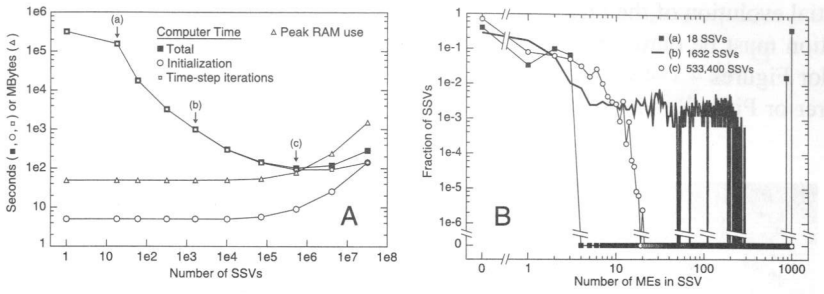


FIGURE 4.5 Relationship Between Computer Time, Memory (RAM) Use, and Spatial Partitioning. (A) Ten simulations were run using increasing numbers of spatial partitions to create increasing numbers of spatial subvolumes (SSVs; abcissa). For each simulation, the log-log plot shows the: (1) time required for initialization of SSVs and all other simulation objects (circles); (2) time required to execute the time-step iterations (open squares); (3) total run time (closed squares); and (4) peak memory use during the simulation. (B) Frequency distributions (log-log scales) for mesh elements (MEs) contained in SSVs, shown for three of the partitioning conditions (labeled a, b, and c) used in A. With few partitions and SSVs (a), many of the SSVs contain 900 or more MEs, making ray tracing and ray marching (Figure 4.2B) extremely inefficient and execution speed extremely slow. With an optimal number and size (see text) of SSVs (c), most of the SSVs contain fewer than 10 MEs, and no SSV contains more than 20. Thus, execution speed is faster by orders of magnitude.

are produced by ACh hydrolysis) and release them on the opposite (intracellular) side of the nerve membrane. The site densities and distributions used for AChR and AChE ESs are from EM autoradiographic measurements,[24,25,27] while the density for ChRs is an illustrative guess.

Exocytosis of ACh from one synaptic vesicle (SV) was simulated using a time course of vesicle emptying (Figure 4.4F) predicted by rapid fusion pore expansion.[30] An explicit model of the vesicle and expanding fusion pore can be simulated with MCell using checkpointing (Section 4.3.3), but requires a sub-ns time-step as the ACh diffuses out through the constrictive pore. As detailed elsewhere[31,33] and shown in Figure 4.4F, the same time course of release can be obtained with a time-step on the μs scale if the actual vesicle and pore are replaced by incremental releases of ACh directly within the cleft under the (missing) pore (release_sites.mdl). Aside from allowing a much longer time-step and thus far fewer iterations, this method also side-steps the need for checkpointing, and hence the simulation runs as fast as if all the ACh were released instantaneously.

The reaction mechanisms for AChR and AChE ESs are as shown in Figure 4.1. The illustrative reaction for Ch reuptake is simply reversible binding followed by rate-limiting translocation and release across the nerve membrane (reaction_mechanisms.mdl). To obtain the predicted size and time course of an mEPC, the number of AChR ESs in the A_2R^4 state is output for each time-step. This is shown for the rising phase and peak in Figure 4.4G, together with the amount of free ACh in the entire cleft. To visualize a snap-shot of the simulation (e.g., Figure 4.4E), the positions of each ME, ES, and ligand molecule must be output along with current state information for the ESs and ligand molecules. For an animation that shows the

spatial evolution of the mEPC (i.e., "saturated disk" formation),[5,16,17,26,33] state information must be output repeatedly for multiple time-steps. Figures 4.4 and 4.6 and Color Figures 4.1–4.6* show examples of 3-D renderings done with IBM DataExplorer or Pixar RenderMan.

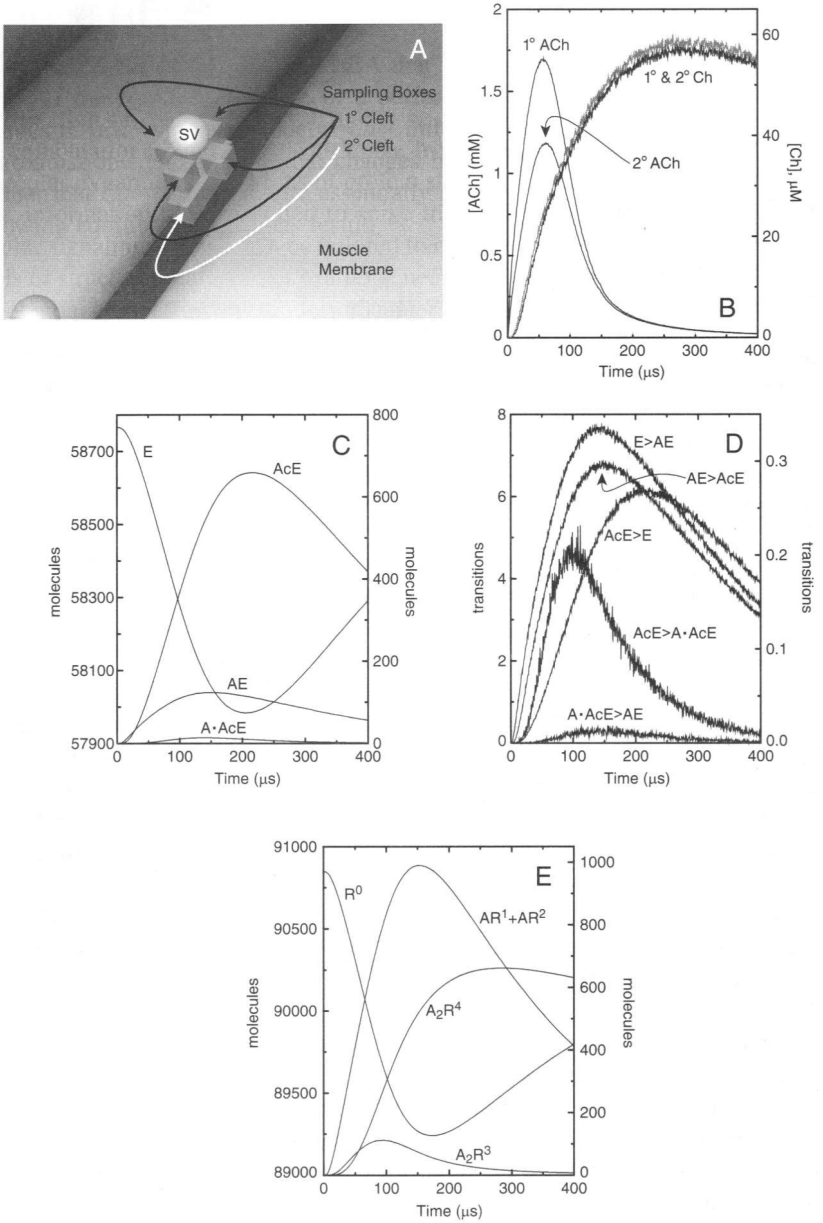


FIGURE 4.6

(continued)

* Color Figures 4.1–4.6 follow page 140.

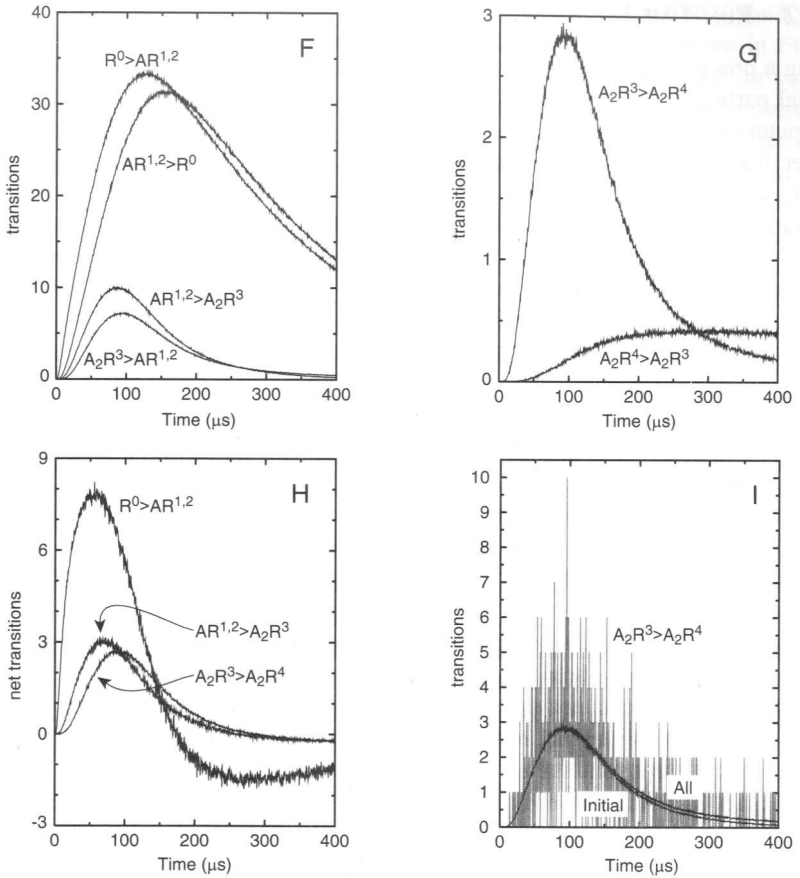


FIGURE 4.6 (Continued) Detailed simulation results (average of 2000 simulations). (A) Positions of transparent boxes used to sample acetylcholine (ACh) and choline (Ch) concentrations during the simulations. (B) Time course of ACh and Ch concentration sampled in primary and secondary cleft. (C and E) Amounts of unbound and bound AChE and AChR intermediates (states are labeled as in Figure 4.1). (D, F, and G) Selected examples of absolute transition rates (number of transitions from starting to ending state per time-step; $\Delta t = 0.5$ μs). Results are summed for the two single-bound AChR states, as indicated by the $AR^{1,2}$ label. (H) Examples of net transition rates (forward transitions minus reverse transitions per time-step) for AChR states. Note the change from positive to negative values during the mEPC rising phase or at time of peak amplitude (cf. A_2R^4 curve in E). (I) Detail of AChR opening transitions. Noisy results from one simulation are shown for comparison with the averaged curves labeled "All" and "Initial." "All" indicates *all transitions*, and is the same curve as shown previously in G on an expanded scale. "Initial" indicates a *subset of all transitions*, i.e., $A_2R^3 > A_2R^4$ transitions that *were not preceded by occupation of the A_2R^4 state during the previous time-step*. In terms of single channel kinetics, "Initial" transitions are those that initiate a burst of openings, versus those that occur within a burst after a "flicker" closure. Here the difference between All and Initial is small because the channel opening probability ($\beta/(\beta + k_{-1} + k_{-2})$, Figure 4.1) is small and flickering is rare, but for other channels and/or rate constant values the difference can be large. The time course of the mEPC is determined by the time course of Initial openings, and direct prediction of such transition subtypes can only be obtained from Monte Carlo simulations.

4.5.2 Run-Time Logistics

Using a present-day workstation to simulate one complete mEPC in the absence of spatial partitions (i.e., where the entire model space constitutes a single SSV), the computer memory and time requirements are approximately 50 MBytes and 1 week, respectively (Figure 4.5A).* As the number of spatial partitions (which require memory) is increased in all three dimensions, the resulting SSVs contain fewer MEs (Figure 4.5B), and therefore the efficiency of ray marching (Section 4.3.5) is increased substantially. As a result, computer time drops precipitously and memory use increases gradually, until optimal conditions are obtained with ~500,000 SSVs (condition (c) in Figure 4.5A and B). At this point the required computer time is only ~100 seconds, and memory use has increased to ~80 MBytes. Thus, for only a 60% increase in memory, computer time drops by about *3.5 orders of magnitude*. If even more spatial partitions are added, the distance between adjacent partitions becomes smaller than the mean random walk step length $\bar{l}_r$ (Equation 4.6), and diffusing ligand molecules pass through a partition with virtually every movement. This introduces additional operations per time-step (Figure 4.2B), and hence computer time increases again slightly (by ~3-fold for the right-most point in Figure 4.5A). With very numerous partitions and SSVs, however, memory use can increase dramatically (e.g., to over 1 GByte as shown for ~30 million SSVs). Nevertheless, Figure 4.5A shows that most of the benefit obtained with spatial partitions is achieved with far fewer than are required for the shortest possible computer time. Hence, memory use and computer time can be "titrated" as needed for any model and computer system, with little reduction in overall throughput.

4.5.3 Detailed Output

In this particular model, diffusing ACh molecules cannot escape from the synaptic cleft space, and so a simple count of all free ACh molecules gives a spatial summation within the entire cleft volume (Figure 4.4G). To quantify the concentration of diffusing molecules in any particular region of space, a transparent box (or any other closed transparent mesh) can be placed in that region, and then the free ligand molecules can be counted within the box.** Figure 4.6A shows four such sampling boxes located in the primary cleft under the central synaptic vesicle indicated in Figure 4.4. In addition, another box is located just beneath the first four, i.e., at the

---

* This computer time figure reflects ACh release from a corner vesicle if the entire structure shown in Figure 4.4A is enclosed in an absorptive bounding box. The presence of the bounding box has essentially no effect on the amplitude of the mEPC regardless of vesicle position, but with release from a corner a significant fraction of diffusing ACh and Ch molecules are removed by absorption rather than hydrolysis and reuptake. With release from a central vesicle (e.g., as labeled in Figure 4.4) about *fourfold* more computer time is required because far fewer molecules are absorbed, and therefore far more ray marching is required before the simulation completes.

** As detailed in Figure 4.2B, each time a diffusing ligand molecule crosses a transparent ME, the event is detected. The direction of crossing is also known, and for a fully closed mesh would be either from outside to inside or vice versa. Thus, the number of molecules that enter and exit an enclosed space can be counted during each time-step, and the amount remaining within the space at the end of each time-step can be specified as output.

entrance to the secondary cleft created by the underlying junctional fold (ligand_sampling_boxes.mdl). To improve the signal-to-noise ratio of results shown in Figure 4.6, the simulation was run 2000 times with different seed values and the results were averaged (~45-fold reduction of noise, see Figure 4.6I). With one present-day single-processor workstation this requires 1–2 days of computer time, but since each simulation is an independent computation, many processors can be used simultaneously.

The time course of sampled ACh concentration is shown in Figure 4.6B, and such simulation results can be compared to increasingly sophisticated experimental estimates.[3,8,22,23] The difference between the primary and secondary cleft concentrations illustrates a steep ACh gradient across the height of the primary cleft (~50 nm), but the gradient persists only during exocytosis and the early rising phase of the mEPC (Figures 4.4F and G and 4.6E). The concentration of Ch sampled in the same volumes is far lower at all times, reaches its peak much later, and is virtually identical in both regions, reflecting relatively slow Ch production within the cleft spaces by AChE sites (despite AChE's fast catalytic turnover number, 16,000 s^{-1} in this example).[20] Figure 4.6C and E show the temporal evolution of each AChE and AChR state, and the remaining panels illustrate detailed output of reaction transition rates. Such data is presently being used to investigate pre- and postsynaptic factors that influence synaptic noise and variability, and to generate functions that can be incorporated into compartmental models of neuronal function. Snap-shots of an individual simulation are shown in Color Figures 4.1–4.3 for selected times at which different reactants are present at peak concentrations or amounts.

4.6 EXAMPLE MODEL: POTENTIAL SPATIAL AND TEMPORAL INTERACTION OF NEURONAL GLUTAMATERGIC CURRENTS

The preceding example focuses on the detailed time course and variability of quantal currents at a peripheral synapse. Here the focus is potential spatial and temporal interaction between central glutamatergic synapses on an interneuron. The model is strictly qualitative and is used as a conceptually interesting and considerably smaller scale example than the preceding.

The model surfaces consist of an inner and outer ovoid (Color Figure 4.4), with the inner representing the nerve cell body, and the outer defining a closed bounding layer of intercellular diffusion space (total of 470 MEs for the two surfaces). In reality of course, the nerve cell would have processes and the diffusion space would extend out between many different surrounding neural and glial elements. Effector sites representing glutamate receptors with long-lived desensitization states are present on the nerve cell (AMPA GluR; reaction mechanism shown in Figure 4.7A obtained from Reference 12), and reuptake sites (transporters) are present on the surrounding surface (as for preceding example, simple reversible binding followed by rate-limiting translocation and release).

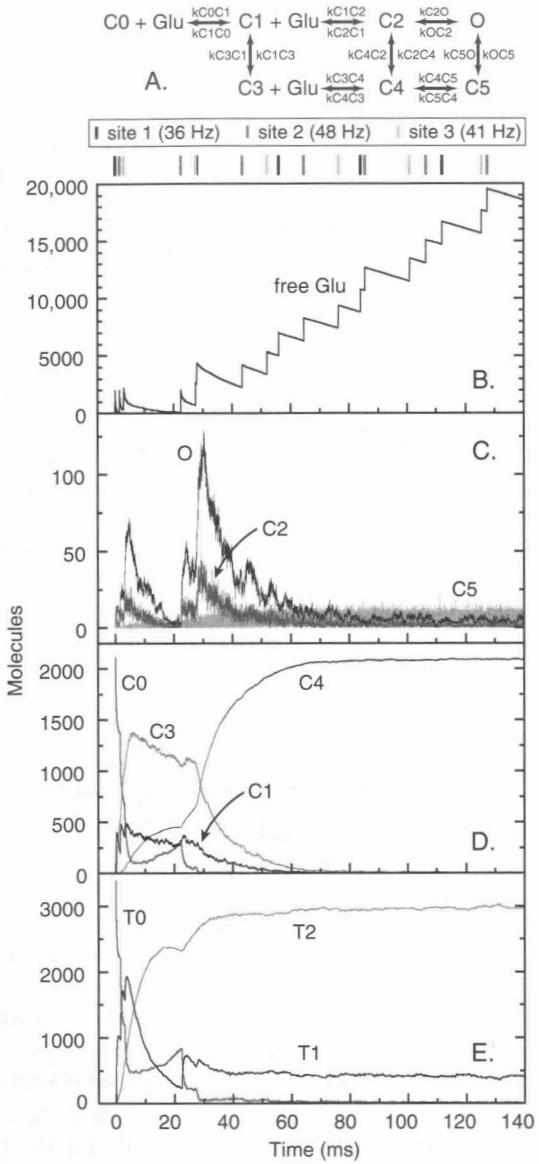


FIGURE 4.7 Glutamate Receptor Reaction Mechanism and Simulation Results. (A) Reaction mechanism with three primary closed states (C0, C1, and C2), one open state (O), and three desensitized states (C3, C4, and C5). Timing of Glu release from three different sites is indicated (2000 molecules per release). (B) Amount of free Glu in the intercellular space that surrounds the model cell. (C and D) Evolution of bound GluR states. Early release events potentiate to open channels, but at late times almost all GluRs are in the C4 desensitized state (see Figure 4.12). (E) Evolution of bound Glu reuptake (transporter) sites. T0 is unbound, T1 is reversibly bound, and T2 is the rate-limiting state that transports and releases Glu molecules. At late times, transporter sites are saturated due to the slow rate of release.

Three different sites are defined for quantal Glu release to represent different presynaptic boutons, and the MDL is also used to define a train of release events at each site (frequencies of 36, 41, or 48 Hz; for simplicity, instantaneous release is used rather than a release function). Figure 4.7A shows the composite timing of release from the three sites during the simulation, which on a present-day workstation requires about 4 hours of computer time for the conditions illustrated here (140,000 iterations for a total elapsed simulation time of 140 ms). Snap-shots are shown for two selected early times and one late time when desensitization predominates (Color Figures 4.4–4.6).

As shown in Figure 4.7B, free Glu builds up in the intercellular space over the course of subsequent release events. For the initial ~30 ms, release events potentiate to produce an increased number of open GluR channels, but thereafter the Glu reuptake sites become saturated and the GluR's are driven almost exclusively into desensitized states for the remainder of the simulation. This model is not included here to make quantitative realistic predictions, but rather to point out the important need for accurately determined 3-D input parameters, e.g., reconstructed diffusion spaces, release positions, receptor and reuptake site densities and distributions. All of these factors, together with all of the rates for transmitter release, diffusion, reuptake, and receptor activation and desensitization, will determine the physiological behavior of the system. As increasingly accurate experimental determinations of these input parameters become available, realistic 3-D MC simulations will be increasingly important to a quantitative understanding of nervous system function.

ACKNOWLEDGMENTS

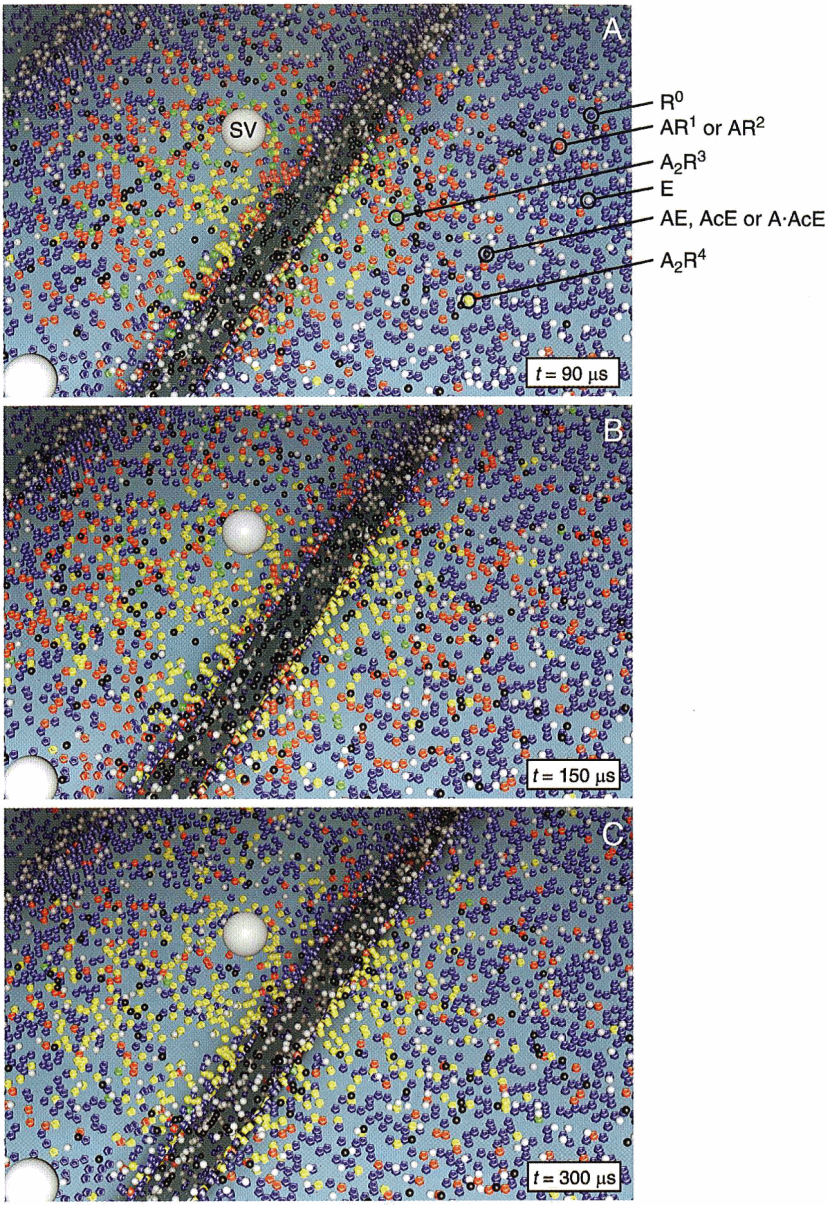
We thank Erik De Schutter for his monumental patience and Ed and Mika Salpeter and Ralph Roskies for comments on Section 4.4. Supported by NIH K08NS01776 and RR-06009 and NSF IBN-9603611.

REFERENCES

1. Anglister, L., Stiles, J. R., and Salpeter, M. M., Acetylcholinesterase density and turnover number at frog neuromuscular junctions, with modeling of their role in synaptic function, *Neuron*, 12, 783, 1992.
2. Arkin, A. P., Ross, J., and McAdams, H. H., Stochastic kinetic analysis of developmental pathway bifurcation in phage λ-infected *E. coli* cells, *Genetics*, 149, 1633, 1998.
3. Barbour, B. and Hausser, M., Intersynaptic diffusion of neurotransmitter, *Trends Neurosci.*, 20, 377, 1997.
4. Barrow, G. M., *Physical Chemistry for the Life Sciences*, McGraw-Hill, New York, 1981.
5. Bartol, T. M., Jr., Land, B. R., Salpeter, E. E., and Salpeter, M. M., Monte Carlo simulation of MEPC generation in the vertebrate neuromuscular junction, *Biophys. J.*, 59, 1290, 1991.

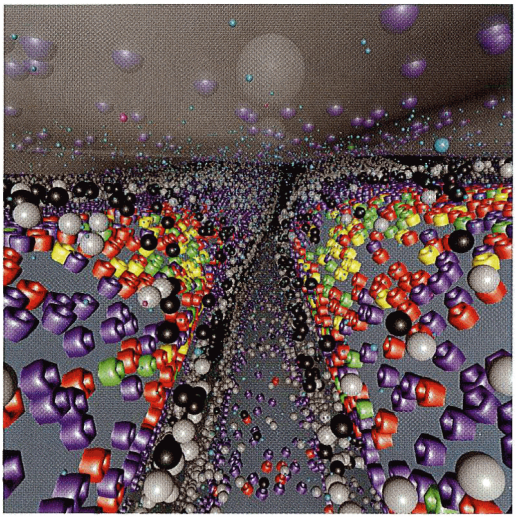
6. Bennett, M. R., Farnell, L., and Gibson, W. G., Quantal transmission at purinergic synapses: stochastic interaction between ATP and its receptors, *J. Theor. Biol.*, 175, 397, 1995.
7. Bennett, M. R., Farnell, L., Gibson, W. G., and Lavidis, N. A., Synaptic transmission at visualized sympathetic boutons: stochastic interaction between acetylcholine and its receptors, *Biophys. J.*, 72, 1595, 1997.
8. Clements, J. D., Transmitter timecourse in the synaptic cleft: its role in central synaptic function, *Trends Neurosci.*, 19, 163, 1996.
9. Crank, J., *The Mathematics of Diffusion*, Oxford University Press, London, 1956.
10. Denes, J. and Krewski, D., An exact representation for the generating function for the Moolgavkar-Venzon–Knudson two-stage model of carcinogenesis with stochastic stem cell growth, *Math. Biosci.*, 131, 185, 1996.
11. Faber, D. S., Young, W. S., Legendre, P., and Korn, H., Intrinsic quantal variability due to stochastic properties of receptor-transmitter interactions, *Science*, 258, 1494, 1992.
12. Geiger, J. R. P., Roth, A., Taskin, B., and Jonas, P., Glutamate-mediated synaptic excitation of cortical interneurons, in *Handbook of Experimental Pharmacology*, Vol. 141, *Retinoids, Ionotropic Glutamate Receptors in the CNS*, Jonas, P. and Monyer, H., Eds., Springer-Verlag, Berlin, 363, 1999.
13. Kalos, M. H. and Whitlock, P. A., *Monte Carlo Methods*, Vol. I: *Basics*, John Wiley & Sons, New York, 1986.
14. Knuth, D., *The Art of Computer Programming*, MIT Press, Cambridge, 1969.
15. Kruk, P. J., Korn, H., and Faber, D. S., The effects of geometrical parameters on synaptic transmission: a Monte Carlo simulation study, *Biophys. J.*, 73, 2874, 1997.
16. Land, B. R., Salpeter, E. E., and Salpeter, M. M., Acetylcholine receptor site density affects the rising phase of miniature endplate currents, *Proc. Natl. Acad. Sci. U.S.A.*, 77, 3736, 1980.
17. Matthews-Bellinger, J. and Salpeter, M. M., Distribution of acetylcholine receptors at frog neuromuscular junctions with a discussion of some physiological implications, *J. Physiol.*, 279, 197, 1978.
18. McQuarrie, D. A., Jachimowski, C. J., and Russell, M. E., Kinetics of small systems II, *J. Chem. Phys.*, 40, 2914, 1964.
19. Press, W. H., Flannery, B. P., Teukolsky, S. A., and Vetterling, W. T., *Numerical Recipes. The Art of Scientific Computing*, Cambridge University Press, New York, 1986.
20. Rosenberry, T., Acetylcholinesterase, *Adv. Enzymol.*, 43, 103, 1975.
21. Rubenstein, R. Y., *Simulation and the Monte Carlo Method*, John Wiley & Sons, New York, 1981.
22. Rusakov, D. A. and Kullmann, D. M., Extrasynaptic glutamate diffusion in the hippocampus: ultrastructural constraints, uptake, and receptor activation, *J. Neurosci.*, 18, 158, 1998.
23. Rusakov, D. A., Kullmann, D. M., and Stewart, M. G., Hippocampal synapses: do they talk to their neighbors? *Trends Neurosci.*, 22, 382, 1999.
24. Salpeter, M. M., Electron microscope autoradiography as a quantitative tool in enzyme cytochemistry: the distribution of acetylcholinesterase at motor endplates of a vertebrate twitch muscle, *J. Cell Biol.*, 32, 379, 1967.
25. Salpeter, M. M., Electron microscope radioautography as a quantitative tool in enzyme cytochemistry. II. The distribution of DFP-reactive sites at motor endplates of a vertebrate twitch muscle, *J. Cell Biol.*, 42, 122, 1969.

26. Salpeter, M. M., Vertebrate neuromuscular junctions: general morphology, molecular organization, and functional consequences, in *The Vertebrate Neuromuscular Junction*, Salpeter, M. M., Ed., Alan R. Liss, New York, 1987, 1.
27. Salpeter, M. M., Smith, C. D., and Matthews-Bellinger, J. A., Acetylcholine receptor at neuromuscular junctions by EM autoradiography using mask analysis and linear sources, *J. Electron Microsc. Tech.*, 1, 63, 1984.
28. Schroeder, W., Martin, K., and Lorensen, B., *The Visualization Toolkit*, 2nd Ed., Prentice-Hall, Englewood Cliffs, NJ, 1998.
29. Smart, J. L. and McCammon, J. A., Analysis of synaptic transmission in the neuromuscular junction using a continuum finite element model, *Biophys. J.*, 75, 1679, 1998.
30. Stiles, J. R., Van Helden, D., Bartol, T. M., Salpeter, E. E., and Salpeter, M. M., Miniature endplate current rise times <100 μs from improved dual recordings can be modeled with passive acetylcholine diffusion from a synaptic vesicle, *Proc. Natl. Acad. Sci. U.S.A.*, 93, 5747, 1996.
31. Stiles, J. R., Bartol, T. M., Salpeter, E. E., and Salpeter, M. M., Monte Carlo simulation of neurotransmitter release using MCell, a general simulator of cellular physiological processes, in *Computational Neuroscience*, Bower, J. M., Ed., Plenum Press, New York, 1998, 279.
32. Stiles, J. R., Kovyazina, I.V., Salpeter, E. E., and Salpeter, M. M., The temperature sensitivity of miniature endplate currents is mostly governed by channel gating: evidence from optimized recordings and Monte Carlo simulations, *Biophys. J.*, 77, 1177, 1999.
33. Stiles, J. R., Bartol, T. M., Salpeter, M. M., Salpeter, E. E., and Sejnowski, T. J., Synaptic variability: new insights from reconstructions and Monte Carlo simulations with MCell, in *Synapses*, Cowan, W. M., Stevens, C. F., and Sudhof, T. C., Eds., Johns Hopkins University Press, Baltimore, MD, in press.
34. Tomita, M., Hashimoto, K., Takahashi, K., Shimizu, T., Matsuzaki, Y., Miyoshi, F., Saito, K., Tanida, S., Yugi, K., Venter, J. C., and Hutchison, C., E-CELL: software environment for whole cell simulation, *Bioinformatics*, 15, 72, 1999.
35. Wahl, L. M., Pouzat, C., and Stratford, K. J., Monte Carlo simulation of fast excitatory synaptic transmission at a hippocampal synapse, *J. Neurophysiol.*, 75, 597, 1996.

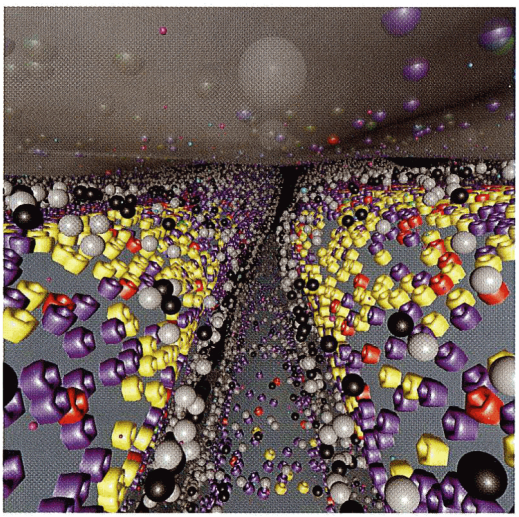


Color Figure 4.1 Spatial and temporal evolution of an mEPC. The presynaptic membrane has been removed for clarity. Postsynaptic AChR and basal lamina AChE glyphs are color-coded according to their state as indicated in A. The times for each image were chosen from Figure 4.6E to show peak numbers of A_2R^3 (A), $AR^1 + AR^2$ (B), or A_2R^4 (C). Diffusing ACh and Ch molecules are tiny (to scale) cyan and magenta spheres, respectively (see Color Figures 4.2 and 4.3). These images were rendered using Pixar RenderMan (0.5–2 hours computer time per image).

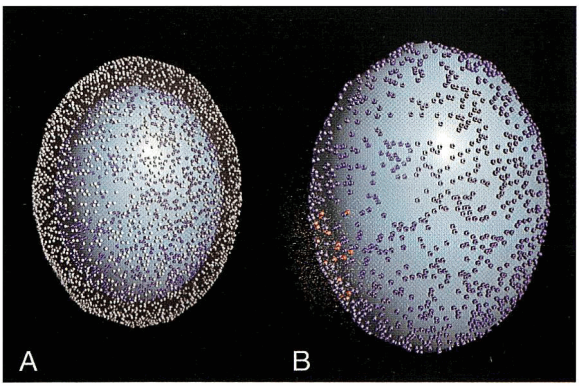
Color Figure 4.2 View of synaptic cleft at time of peak ACh concentration. The time (60 μs) was chosen from Figure 4.6B (sampling boxes not shown). ACh, Glu, AChR, and AChE molecules are color-coded as in Color Figure 4.1. The largest central synaptic vesicle (dim white) indicates the ACh release site and is seen through translucent gray nerve membrane, which contains Ch reuptake sites (blue, unbound; red, reversibly bound; green, rate-limiting intermediate that transports Ch. See text).



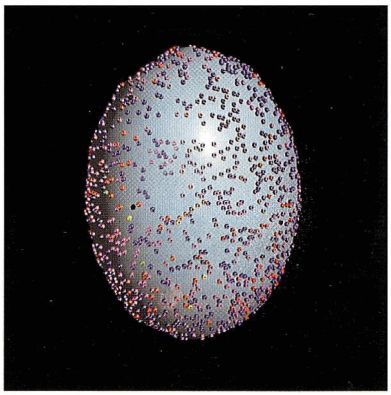
Color Figure 4.3 View of synaptic cleft at time of peak Ch concentration and mEPC amplitude. The time (300 μs) was chosen from Figures 4.6B and E. Molecules are color-coded as in Color Figure 4.1.

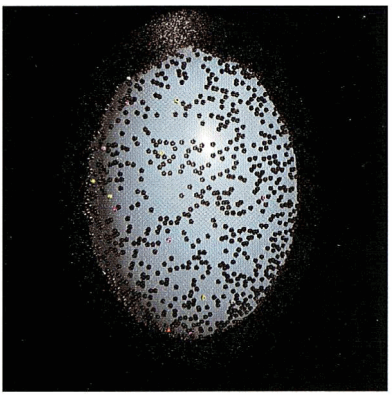


Color Figure 4.4 "Small-scale" qualitative model of neuronal glutamatergic currents. (A) Model design ($t = 0$). Inner surface (light blue) represents simplified interneuron cell body (~0.3 × ~0.5 μm), and contains ESs to simulate AMPA GluRs (blue glyphs, 4000 μm^{-2}). Translucent outer surface defines closed intercellular diffusion space, and contains Glu reuptake sites (white spheres, 4000 μm^{-2}). (B) Larger view of inner surface only, early after first release of Glu ($t = 5$ μs). Release occurred from a site at lower left. Glu molecules are small white spheres. Visible GluRs are either in the unbound state (blue, C0; see Figure 4.7) or the primary single-bound state (C1, red).



Color Figure 4.5 Second release of glutamate. Release (at $t = 1.5$ ms) preceded this image by 5 μs, and occurred from site on right side. GluRs are present in a mix of the primary closed states (C0, blue; C1, red; C2, green), the open state (O, yellow), and the initial desensitized state (C3, magenta).





Color Figure 4.6 Late release of glutamate. Release (at $t = 76.5$ ms) preceded this image by 1 μs, and occurred from site at top of view. At this late time, Glu has accumulated in the diffusion space, and nearly all GluRs are in the longest-lived desensitized state (C4, black; cf. Figures 4.7B and D). Aside from rare open and other states, several GluRs in the short-lived C5 state (white) are also visible.