Annals of Biomedical Engineering, Vol. 30, pp. 578–587, 2002
Printed in the USA. All rights reserved.

0090-6964/2002/30(4)/578/10/$15.00

Comparison of Algorithms for the Simulation of Action Potentials with Stochastic Sodium Channels

HIROYUKI MINO,[1] JAY T. RUBINSTEIN,[1,2] and JOHN A. WHITE[3]

[1]Department of Otolaryngology—Head and Neck Surgery, [2]Department of Physiology and Biophysics and Department of Biomedical Engineering, The University of Iowa College of Medicine, 200 Hawkins Drive, Iowa City, IA, and [3]Department of Biomedical Engineering, Center for BioDynamics, Boston University, Boston, MA

(Received 14 June 2001; accepted 19 February 2002)

Abstract—This paper presents a comparison of computational algorithms to simulate action potentials using stochastic sodium channels. Four algorithms are compared in single-node models: Strassberg and DeFelice (1993) (SD), Rubinstein (1995) (R), Chow and White (1996) (CW), and Fox (1997) (F). Neural responses are simulated to a simple and a preconditioned monophasic current pulse. Three exact algorithms implementing Markov jumping processes (SD, R, CW) resulted in similar responses, while the approximation algorithm using Langevin's equation (F) showed quite different responses from those in the exact algorithms. The computational time was measured as well: 1(F), 7(CW), 32(SD), 39(R) relative to that of the F algorithm. Furthermore, it is shown that as the sampling step for integration of the transmembrane potential increases, neural responses in all algorithms tended to be different from those in dense sampling steps, however, the CW algorithm was robust even at a sparse sampling step. It is concluded that the most computationally efficient algorithm having appropriate properties of neural excitability is the CW algorithm. © *2002 Biomedical Engineering Society.* [DOI: 10.1114/1.1475343]

Keywords—Computer simulations, Markov jumping process, Action potentials, Hodgkin–Huxley equations, Neural excitability.

INTRODUCTION

Nonlinear deterministic models such as the Hodgkin–Huxley model[7] have played a key role in numerical simulations of action potentials. However, it has been well known that stochastic phenomena influence action potential initiation since Verveen's extensive descriptions of spike timing and threshold fluctuations.[18] Lecar and Nossal[8] predicted that fluctuations of sodium currents account for the stochastic nature of spike timing and threshold in the node of Ranvier; this prediction was verified experimentally by Sigworth.[14] The kinetics of stochastic ion channels are well-fit as continuous-time, discrete-state Markov jumping processes.[3,10] These microscopic models have been incorporated into macroscopic models of space-clamped membrane.[1,2,5,13,15,16] The biological importance of ionic channel noise has been recently reviewed.[19]

Computational implementation of these stochastic membrane models is hampered by their biophysical complexity; large demands are made for memory and CPU time, particularly when large numbers of stochastic sodium channels are incorporated into nodes of Ranvier in cable models of myelinated nerve fibers. This problem is compounded in applications that require simultaneous simulation of populations of fibers, like the approximately 30,000 fibers of the VIIIth cranial nerve. Although supercomputer technologies can solve these problems under some restricted conditions (i.e., reducing the number of nodes and/or the number of fibers), it is desirable to simulate neuronal responses in a realistic distributed cable model using the most computationally efficient algorithm.

The algorithms for computer simulation can be categorized into two groups: approximation algorithms of the differential equation using Langevin's equation (F algorithm),[5] and exact algorithms of conductance fluctuations using Markov jumping process.[1,2,13,15,16] The exact algorithms may be further subdivided into channel-state-tracking (CST) algorithms, and channel-number-tracking (CNT) algorithms. The CST algorithm tracks the states of each channel and superimposes individual channel currents corresponding to the states in order to generate sodium channel current fluctuations. This algorithm is simple in principle, however, it is computationally intensive. It has been utilized by Clay and DeFelice[2] in 1983, Strassberg and DeFelice (SD algorithm)[16] in 1993, and Rubinstein (R algorithm)[13] in 1995. An alternative algorithm proposed by Gillespie[6] in 1977 and since implemented for nonlinear membrane conductances,[15] by Chow and White (CW algorithm)[1] in 1996 tracks the number of channels in each state, assuming that multichannel systems are independent and memoryless. The CNT algorithm promises much greater efficiency in

Address correspondence to Hiroyuki Mino, PhD, Department of Otolaryngology, The University of Iowa College of Medicine, 200 Hawkins Drive, Iowa City IA 52242. Electronic mail: hiromino@earpower.oto.uiowa.edu

cases where many channels are considered because the algorithm implements only the single-channel system possessing an ''effective transition rate'' associated with the transition rates in the multichannel system and the number of channels in each state. The approximation algorithm[5] takes advantage of classical results from physics (reviewed by van Kampen, 1992)[17] to approximate stochastic fluctuations by adding a white noise term to the traditional Hodgkin–Huxley gating equations.

This paper evaluates and compares the performance of F, CW, SD, and R algorithms, and the approximation algorithms in the simulation of neural excitability in mammalian nodes of Ranvier and examines the effects of the sampling interval for numerical integration. Numerical simulations are performed of the poststimulus time histogram (PSTH) and input-ouput function (I/O) in which the number of sodium channels is assumed to be 1000, and in which both a simple and a preconditioned monophasic pulse are applied as electric stimuli.

MATERIALS AND METHODS

Preliminaries

Stochastic ion channels can be modeled well[3,4,10] as continuous-time discrete-state Markov processes (Markov jumping processes).[11] Since the sodium channel has been assumed to possess three activating m gates with four distinct states and one inactivating h gate with two distinct states according to the Hodgkin–Huxley model, the kinetic scheme based on Markov process theory has eight states, m_0h_1, m_1h_1, m_2h_1, m_3h_1, m_0h_0, m_1h_0, m_2h_0, and m_3h_0 with 20 transition rates designated by $\alpha_{h,m}$'s and $\beta_{h,m}$'s. The kinetic scheme is expressed as follows:

$$\begin{array}{ccccccc} & 3\alpha_m & & 2\alpha_m & & \alpha_m & \\ m_0h_0 & \rightleftharpoons & m_1h_0 & \rightleftharpoons & m_2h_0 & \rightleftharpoons & m_3h_0 \\ & \beta_m & & 2\beta_m & & 3\beta_m & \\ \alpha_h\downarrow\uparrow\beta_h & & \alpha_h\downarrow\uparrow\beta_h & & \alpha_h\downarrow\uparrow\beta_h & & \alpha_h\downarrow\uparrow\beta_h \\ & 3\alpha_m & & 2\alpha_m & & \alpha_m & \\ m_0h_1 & \rightleftharpoons & m_1h_1 & \rightleftharpoons & m_2h_1 & \rightleftharpoons & m_3h_1 \\ & \beta_m & & 2\beta_m & & 3\beta_m & \end{array} \tag{1}$$

where the state m_3h_1 denotes the open channel, in which all four gates are open. The transition rates in mammalian neuron at 37 °C are expressed as[9]

$$\alpha_m = \frac{1.872(V_m - 25.41)}{1 - e^{(25.41 - V_m)/6.06}}, \tag{2}$$

$$\beta_m = \frac{3.973(21.001 - V_m)}{1 - e^{(V_m - 21.001)/9.41}}, \tag{3}$$

$$\alpha_h = \frac{-0.549(27.74 + V_m)}{1 - e^{(V_m + 27.74)/9.06}}, \tag{4}$$

$$\beta_h = \frac{22.57}{1 + e^{(56.0 - V_m)/12.5}}, \tag{5}$$

in which the transition rates have units of ms^{-1} and the transmembrane potential V_m has units of mV. Note that these transition rates are dependent on the transmembrane potential. The transmembrane potential $V_m(t)$ is described as a function of time by

$$C_m \frac{dV_m(t)}{dt} + \frac{V_m(t)}{R_m} + \gamma_{\mathrm{Na}} N_{\mathrm{Na}}(t)[V_m(t) - E_{\mathrm{Na}}] = I_{\mathrm{app}}(t), \tag{6}$$

where $I_{\mathrm{app}}(t)$ denotes the stimulus current, C_m and R_m, respectively, denote capacitance and resistance of membrane, and γ_{Na}, $N_{\mathrm{Na}}(t)$ stand for conductance of single sodium ion channels and the number of sodium channels activated at time t in the single node under consideration. In exact algorithms, the number of sodium channels activated $N_{\mathrm{Na}}(t)$ plays a key role in introducing its stochasticity into the transmembrane potentials $V_m(t)$.

Single Node Model

A mammalian single node model was represented as the first-order ordinary differential equation described in Eq. (6) with the following parameters: $C_m = 1.5$ pF, $R_m = 23.3\ \mathrm{M}\Omega$, $\gamma_{\mathrm{Na}} = 25.69$ pS, $E_{\mathrm{Na}} = 66$ mV,[9] and the maximum number of sodium channels $N_{\mathrm{Na}}^{\max} = 1000$. The differential equation was solved using forward Euler integration at 1, 2, 5, and 10 (μs) steps implemented in C on a Pentium PC running Linux. The random numbers were generated according to Press *et al.*[12]

We implemented and compared the approximation algorithm by Fox (F), the exact CST algorithm by Rubinstein (R) based on Clay and DeFelice, Strassberg and DeFelice (SD), the exact CNT algorithm by Chow and White (CW), and the deterministic model by Hodgkin and Huxley (H). The critical output of each algorithm is the calculation of the number of sodium channels activated, N_{Na}.

H algorithm: The number of sodium channels activated is represented by

$$N_{\mathrm{Na}}(t) = N_{\mathrm{Na}}^{\max} m^3(t) h(t), \tag{7}$$

where

$$\frac{dm(t)}{dt} = \alpha_m[1 - m(t)] - \beta_m m(t), \tag{8}$$

$$\frac{dh(t)}{dt}=\alpha_h[1-h(t)]-\beta_h h(t).$$

F algorithm: The number of sodium channels activated is expressed as

$$N_{\mathrm{Na}}(t)=N_{\mathrm{Na}}^{\max}m^3\ (t)h(t). \tag{9}$$

Introducing the perturbations $g_m(t)$ and $g_h(t)$ to $m(t)$ and $h(t)$ kinetics yields

$$\frac{dm(t)}{dt}=\alpha_m[1-m(t)]-\beta_m m(t)+g_m(t), \tag{10}$$

$$\frac{dh(t)}{dt}=\alpha_h[1-h(t)]-\beta_h h(t)+g_h(t), \tag{11}$$

where the perturbations are sampled from Gaussian probability density functions, $g_m(t)\approx N[0,\sigma_m^2(t)]$, $g_h(t)\approx N[0,\sigma_h^2(t)]$ with variances

$$\sigma_m^2(t)=\frac{2}{N_{\mathrm{Na}}^{\max}}\frac{\alpha_m(t)\beta_m(t)}{\alpha_m(t)+\beta_m(t)}, \tag{12}$$

$$\sigma_h^2(t)=\frac{2}{N_{\mathrm{Na}}^{\max}}\frac{\alpha_h(t)\beta_h(t)}{\alpha_h(t)+\beta_h(t)}. \tag{13}$$

Variances are chosen to match equilibrium fluctuation levels.[5,18] Note that $m(t)$'s and $h(t)$'s are truncated so as to take a value between 0 and 1.

R algorithm: Introducing a two state Markov process to each particle, $m_1(t)$, $m_2(t)$, $m_3(t)$, and $h(t)$, i.e.,

$$\begin{array}{cccc} C_{m_1} & C_{m_2} & C_{m_3} & C_h \\ \alpha_m\downarrow\uparrow\beta_m & \alpha\downarrow\uparrow\beta m & \alpha_m\downarrow\uparrow\beta_m & \alpha_h\downarrow\uparrow b_h \\ O_{m_1} & O_{m_2} & O_{m_3} & O_h \end{array} \tag{14}$$

yields the number of sodium channels activated

$$N_{\mathrm{Na}}(t)=\sum_{k=1}^{N_{\mathrm{Na}}^{\max}}\tilde{m}_{1,k}(t)\tilde{m}_{2,k}(t)\tilde{m}_{3,k}(t)\tilde{h}_k(t), \tag{15}$$

where $\tilde{m}_{1,k}(t)$, $\tilde{m}_{2,k}(t)$, $\tilde{m}_{3,k}(t)$, $\tilde{h}_k(t)$ designate the Markov jumping processes taking zero (closed) or one (open) values.

SD algorithm: The number of channels activated is identical to the number of sodium ion channels activated at the state m_3h_1 shown in Eq. (1):

$$N_{\mathrm{Na}}(t)=N_{m_3h_1}(t). \tag{16}$$

Note that the exact SD and R algorithms are classified as CST algorithms. The SD algorithm is based explicitly upon the eight state kinetic scheme described in Eq. (1), while the R algorithm utilizes four particles equivalent implicitly to the eight state kinetic scheme.

CW algorithm: The number of channels activated is identical to the number of sodium channels activated at the state m_3h_1 shown in Eq. (1):

$$N_{\mathrm{Na}}(t)=N_{m_3h_1}(t). \tag{17}$$

Note that although the CW algorithm looks similar to the SD algorithm, calculation is quite different, since the CW algorithm is a CNT algorithm whereas the SD algorithm is a CST algorithm. See the Appendix for a more detailed description of the CW algorithm. The algorithm implemented here is different from that of Chow and White[1] in one small regard, in that the present algorithm integrates Eq. (6) with a relatively large, fixed time step instead of a very small, randomly varying time step (see the Appendix).

The CST algorithm does renew the states of sodium channels only at each sampled time without tracking the state transitions between sampled times. In the CST algorithm, even if sodium channels are less active, e.g., $V_m(t)\approx$ the resting potential, it is necessary to generate many random numbers at each sampled time, implying inefficient implementation. The CNT algorithm renews the number of sodium channels even between the sampled times, generating many random numbers, if sodium channels are very active. Nevertheless, if sodium channels are less active, the CNT algorithm generates only few random numbers because of a long time length generated with a large effective transition rate. This may imply a computational efficiency of the CNT algorithm.

Simulation Procedure

Two kinds of stimuli, simple and preconditioned monophasic pulses, were applied intracellularly to the single node model in order to compare the neural responses of the algorithms. The monophasic stimulus was used for investigating the fundamental statistical parameters, while the preconditioned stimulus was utilized for differentiating temporal properties of the algorithms since the small number of sodium channels are expected to generate current fluctuations even at subthreshold levels and thereby stochastically modify temporal response properties.

The monophasic stimulus pulse, as shown in Fig. 1(a), was applied intracellularly as $I_{\mathrm{app}}(t)$ to the single node in Eq. (6). The application of stimuli was repeated one

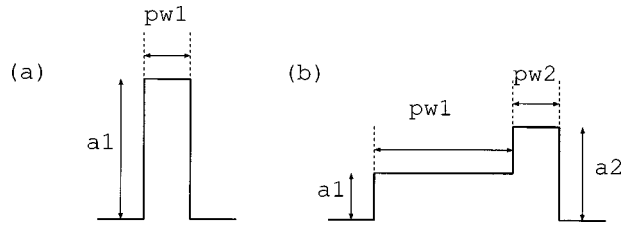

FIGURE 1. Stimulus wave form: monophasic in (a) where a_1 and pw_1, respectively, denote the stimulus current intensity and duration, and preconditioned monophasic in (b), where a_1, pw_1, a_2, and pw_2 stand for the preconditioned current intensity and duration, and the suprathreshold current intensity and duration.

thousand times in order to simulate the PSTH and allow calculation of firing efficiency (FE) (the number of spikes occurred, divided by the total number of stimuli), latency (LT) (the mean value of spike occurrence times), and jitter (JT) (standard deviation of spike occurrences times) using 1, 2, 5, and 10 (μs) time steps.

The preconditioned stimulus pulse, as shown in Fig. 1(b), was applied to the single node in Eq. (6) in order to investigate temporal dynamics of action potentials. The stimulus parameters in Fig. 1(b) were set as follows: for the subthreshold stimulus the current intensity $a_1 = 2.5$ pA with the duration of $pw_1 = 500$ μs, and for the suprathreshold stimuli the current intensity $a_2 = 3.5$ pA with the duration of $pw_2 = 100$ μs. The application of stimuli was again repeated 1000 times at 1, 2, 5, and 10 (μs) time steps.

RESULTS

Monophasic Stimulus

Figure 2 shows ten typical sample paths of the transmembrane potential (left column) at a sampling step of 1 μs and the PSTH (right column) generated from one thousand Monte Carlo runs for five different algorithms in which the stimulus parameters in Fig. 1(a) were set as follows: the stimulus current intensity $a_1 = 6.2$ pA and the duration of monophasic pulse $pw_1 = 100$ μs. With the exception of the outputs of the F and H algorithms, these transmembrane potentials, PSTHs, and firing statistics look identical. Note that the H algorithm cannot simulate spike timing fluctuations because it does not include a stochastic component. As a result, its JT value is 0.0, as shown in the inset of the right-bottom PSTH. The results at sampling steps of 2 and 5 μs were similar to those shown in Fig. 2, whereas at the sampling step of

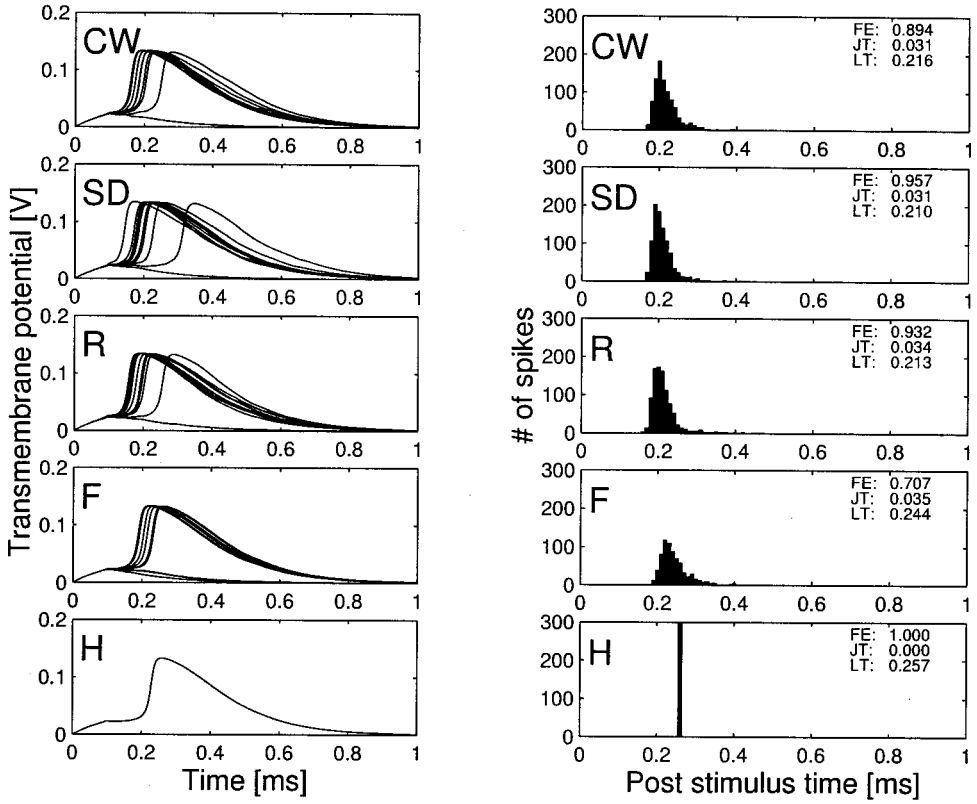

FIGURE 2. Transmembrane potentials in response to ten identical monophasic stimulus pulses with an amplitude of 6.2 pA and a duration of 100 μs (left) and poststimulus time histograms generated from 1000 Monte Carlo runs (right), where FE, JT, and LT are shown in each inset. The sampling step was set at 1 μs.

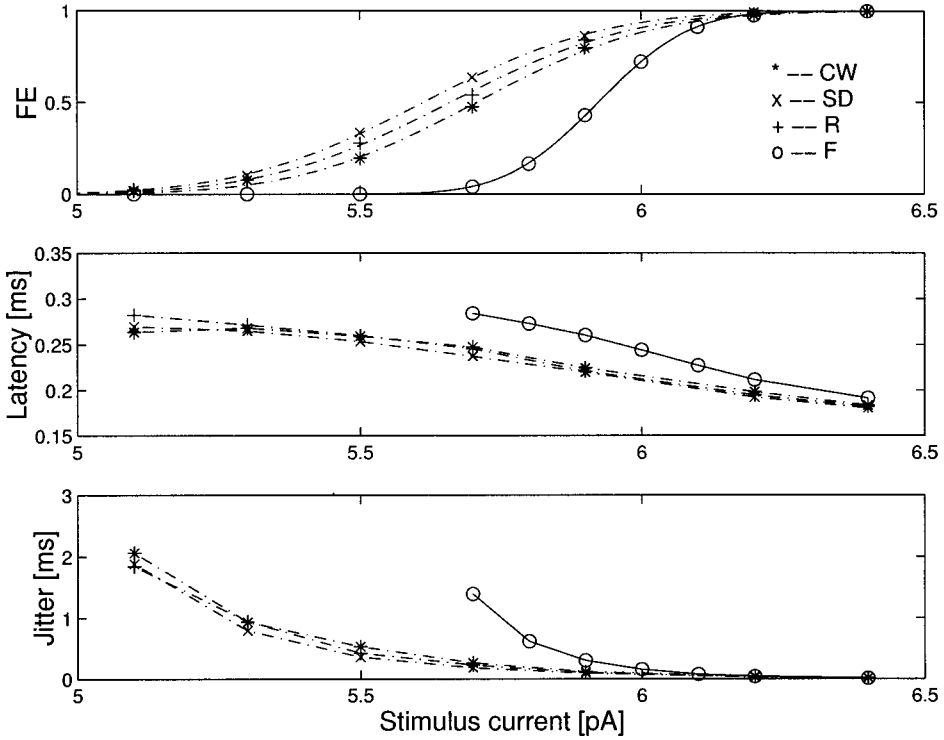

FIGURE 3. FE (top), latency (middle), and jitter (bottom) as a function of stimulus current intensity for four algorithms at $\Delta t = 1$ μs. Stimulus duration was 100 μs. From the data shown in the top panel, I_{th} and RS were estimated for four algorithms: $I_{th} = 5.658$ pA and RS = 0.0350 (CW), $I_{th} = 5.610$ pA and RS = 0.0441 (SD), $I_{th} = 5.657$ pA and RS = 0.0436 (R), and $I_{th} = 5.931$ pA and RS = 0.0215 (F).

10 μs the transmembrane potentials demonstrated numerical error, particularly for the F and H algorithms (data not shown).

Figure 3 shows the firing efficiency (top), latency (middle), and jitter (bottom) as a function of stimulus current intensity for the four algorithms based on stochastic models at the sampling step of 1 μs in which data of CW, SD, R, and F algorithms are, respectively, plotted by the marks $*$, $\times$, $+$, and $\bigcirc$. Each point was generated from one thousand Monte Carlo runs driven by a monophasic current pulse. The stimulus duration was set at 100 μs as in Fig. 2, whereas the stimulus current intensity was varied between 5.2 and 6.4 pA in order to obtain the statistical parameters. For the I/O function, the data were fit by the simplex method with appropriate starting values to an integrated Gaussian function. The curves are comparable except for that of the F algorithm. The threshold current intensity I_{th} and the relative spread (RS; defined as coefficient of variation) of the F algorithm are also quite different from those of the CW, SD, and R algorithms, as is shown in figure caption. The latency and jitter curves were drawn by straight lines between data points, and the curves of the F algorithm are not consistent with those of the CW, SD, and R algorithms. The data at the sampling steps of 2, 5, and 10 μs were similar to those shown in Fig. 3, with the exception of the I/O function showing substantial numerical error at the sampling step of 10 μs (data not shown).

Preconditioned Stimulus

Figure 4 shows ten sample paths of the transmembrane potentials (left column) and the PSTH (right column) generated from 1000 Monte Carlo runs for five different algorithms at the sampling step of 1 μs. The subthreshold stimulus current of 2.5 pA was applied initially for a duration of 500 μs, followed by a stimulus with an amplitude of 3.5 pA and a duration of 100 μs. The transmembrane potentials and PSTHs from the CW, SD, and R algorithms are very similar, as are the firing efficiencies, jitters, and latencies. The data produced by the F algorithm are quite different from those of the CW, SD, and R algorithms, especially for the latency and jitter data. The latency of the F algorithm is more comparable to that of the deterministic H algorithm. The responses at the sampling step of 2 μs were similar to those at the sampling step of 1 μs (data not shown).

Figure 5 shows ten sample paths of the transmembrane potentials (left column) and the PSTH (right column) generated from 1000 Monte Carlo runs for the five algorithms with a sampling step of 5 μs. Note that the

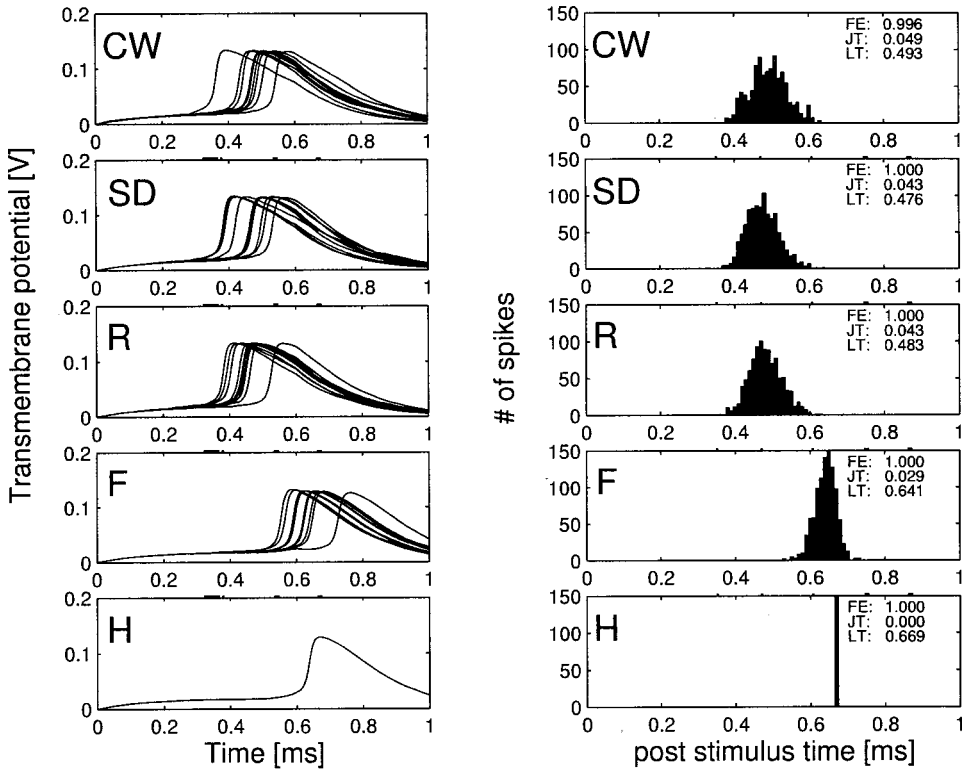

FIGURE 4. Transmembrane potentials in response to ten identical stimulus pulses conditioned (left) at $\Delta t = 1$ μs. Poststimulus time histograms given from 1000 Monte Carlo runs (right). The subthreshold stimulus current of 2.5 pA was applied initially for a duration of 500 μs, followed by a stimulus with an amplitude of 3.5 pA and a duration of 100 μs.

distribution of the PSTHs in the R algorithm sharpen and shifted to the left, although the transmembrane potentials in the CW, SD, and R algorithms look identical.

Figure 6 shows ten sample paths of the transmembrane potentials (left column) and the PSTH (right column) generated from one thousand Monte Carlo runs for five different algorithms at the sampling step of 10 μs. Note that the PSTHs produced by not only the R algorithm but also the SD algorithm shifted to the left. On the other hand, the PSTHs of the CW algorithm at the sampling step of 10 μs were comparable to those of the CW algorithm at the sampling steps of 1, 2, and 5 μs as shown in Figs. 4 and 5. However, the firing efficiency decreased at the sampling step of 10 μs (see the inset of PSTHs), implying a higher threshold value, while the latency and jitter in the CW algorithm at the sampling step of 10 μs are similar to those at the sampling steps of 1, 2, and 5 μs.

Figure 7 shows the firing efficiency (top), latency (middle), and jitter (bottom) as a function of the sampling step for the four algorithms based on stochastic models in which data of CW, SD, R, and F algorithms are, respectively, plotted by the marks $*$, $\times$, $+$, and $\bigcirc$. The CW algorithm (solid lines) is the most robust for calculation of latency and jitter at the sparse sampling step of 10 μs.

Computation Time

The computation time consumed was compared under the case where 20 transmembrane potentials were generated by the CW, SD, R, and F algorithms at the sampling step of 1 μs. Table 1 shows the computation time relative to that of the F algorithm. As expected, the approximation algorithm was the fastest, while among the exact algorithms the CNT algorithm was faster than the CST algorithms. The most computationally efficient, exact algorithm is the CW algorithm.

DISCUSSION

In this paper, we have measured the performance of the computational algorithms simulating action potentials in the presence of sodium current fluctuations. PST histograms produced by the four algorithms were compared at the sampling steps of 1, 2, 5, and 10 μs and the computation times were calculated at a sampling step of 1 μs.

The statistical parameters characterizing neural excitability were in good agreement among the exact algorithms (the CW, SD, and R algorithms). In particular, the firing efficiency, latency, and jitter as a function of stimulus current in the exact algorithms were shown to be

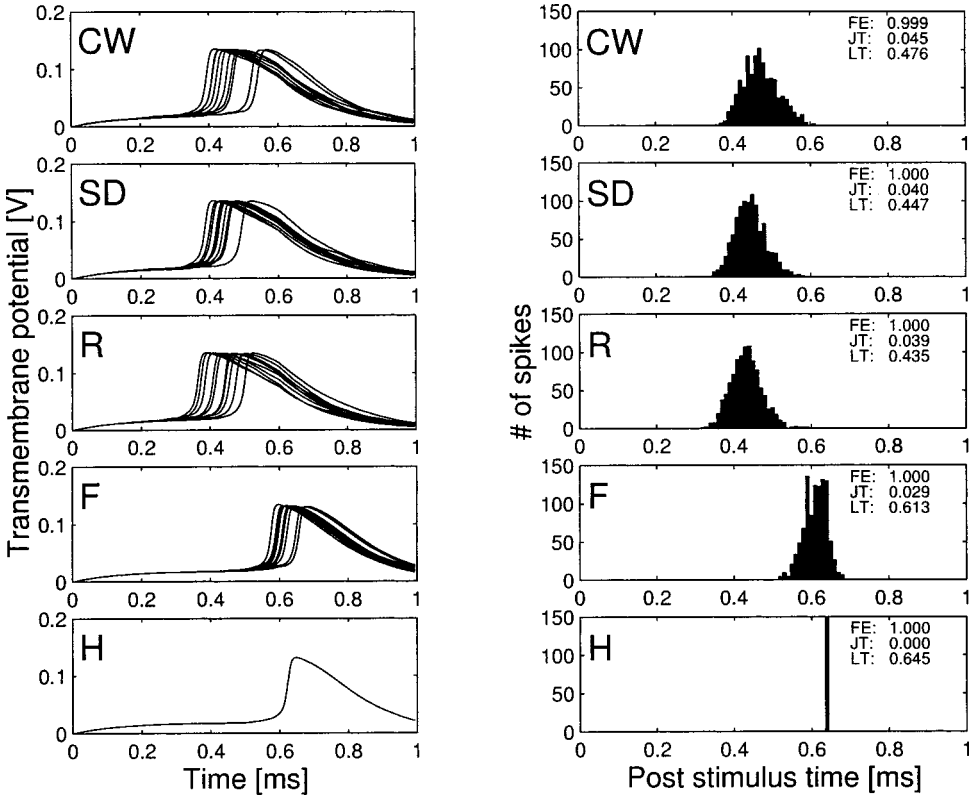

FIGURE 5. Transmembrane potentials in response to ten identical stimulus pulses conditioned (left) at $\Delta t = 5$ μs. Poststimulus time histograms given from 1000 Monte Carlo runs (right). The subthreshold stimulus current of 2.5 pA was applied initially for a duration of 500 μs, followed by a stimulus with an amplitude of 3.5 pA and a duration of 100 μs.

similar. Also, the temporal dynamics of the responses looked identical among the exact algorithms, even if preconditioned stimuli were presented, as is shown in Fig. 4. These facts imply that although the implementation of each algorithm is different, the biophysical result is identical for the CW, SD, and R algorithms.

However, the neural responses of the approximation algorithm were not in agreement with those of the exact algorithms. In particular, the I/O function, latency, and jitter as a function of stimulus intensity were substantially different from those in the exact algorithms. Furthermore, when the preconditioned stimuli were presented, the shape of PSTHs, i.e., the latency and jitter parameters, was quite different from those of the exact algorithms, as shown in Fig. 4. The latency resulting from the approximation algorithm was more like that of the deterministic H algorithm which does not have any jitter.

At first glance, it is surprising that the classically used Langevin method should be so inadequate for simulation of the stochastic Hodgkin–Huxley equations. Two factors could contribute to the inaccuracy of the approximation algorithm. First, it is conceivable that the first-order integration method used to solve the Langevin equation is insufficiently accurate. However, this explanation seems unlikely, because the inaccuracy of the approximation algorithm persists with time steps as small as 1 μs (e.g., Fig. 2). The second contributing factor is that the Langevin approach is often simply not suitable for modeling systems with nonlinear, internally generated noise.[18] The poor performance of the approximation algorithm is likely to reflect the strong degree of internally generated nonlinearity in the Hodgkin–Huxley equations.

In the introduction, we have subdivided the exact algorithms based on Markov jumping process into CST algorithms and the CNT algorithm. The CST algorithms (SD and R algorithms) tracks the states of each channel only at the sampling time t_i, so that any state transitions between the sampling times, i.e., event occurrences at (t_i, t_{i+1}) cannot be updated by this sort of algorithm. The CST algorithm retains accuracy only as long as the sampling step is sufficiently dense enough relative to the channel state transition rates. The inappropriate shape of PSTHs in SD and R algorithms at the sampling step of 10 μs, as shown in Fig. 6, is due to this inherent property of the CST algorithm. On the other hand, the CNT algorithm (CW) renews the number of channels in each state whenever any state transition occurs, so that this algorithm is expected to possess much greater accuracy in updating state transitions between the sampling times.

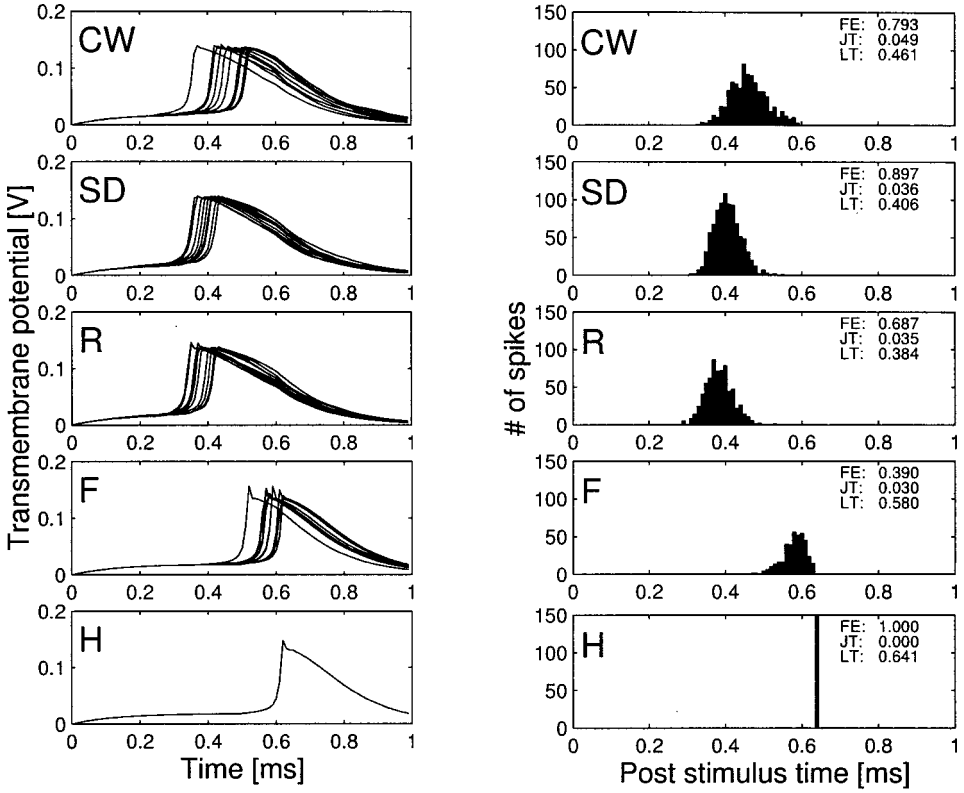

FIGURE 6. Transmembrane potentials in response to ten identical stimulus pulses conditioned (left) at $\Delta t = 10$ μs. Poststimulus time histograms given from 1000 Monte Carlo runs (right). The subthreshold stimulus current of 2.5 pA was applied initially for a duration of 500 μs, followed by a stimulus with an amplitude of 3.5 pA and a duration of 100 μs.

As a result, the shape of PSTHs in the CW algorithm at the sampling step of 10 μs shown in Fig. 6 is comparable to that of the CW algorithm at the sampling step of 1 μs shown in Fig. 4. Also, the latency and jitter in the CW algorithm were robust over the sampling steps of 1–10 μs, as shown in Fig. 7. The firing efficiency was decreased, however, due to the relation between the membrane time constant and the numerical integration.

It is therefore concluded that the most computationally efficient and robust algorithm having appropriate neural response properties is the CNT algorithm (CW). Because the model is rather simplistic, consisting solely of voltage-dependent sodium channels, leakage conductance, and membrane capacitance, it is not expected that any of the algorithms will simulate in detail the temporal properties of real neurons. However, in order to develop more sophisticated models to meet this demand, it is critical that the simplistic model be simulated correctly and in the most efficient manner. Our calculations demonstrate that the approximate solution fails the former requirement. This failure involves firing properties that are readily resolvable in physiological experiments and therefore limits any potential application of the approximate solution. We have incorporated the CW algorithm into a distributed axon moel for simulating stochastic properties of auditory neurons activated by cochlear implants. The greater efficiency of this algorithm has dramatically improved the performance of these very large calculations. Becaused these calculations are much more complex and require many more assumptions than the single-node simulation, a detailed analysis of the properties of the distributed solution will be presented in a future publication.

ACKNOWLEDGMENTS

This work was supported by Neural Prosthesis Program Contract Nos. NO1-DC-9-2106 and NO1-DC-9-2107 from the National Institutes of Health, as well as Grant No. BES-0085177 from the National Science Foundation.

APPENDIX

The CNT algorithm was originally developed by Gillespie,[6] and later used in the context of modeling stochastic membrane mechanisms in neurons.[1,15] In this appendix the CNT algorithm is described in depth as it is the most efficient and robust of the algorithms studied.

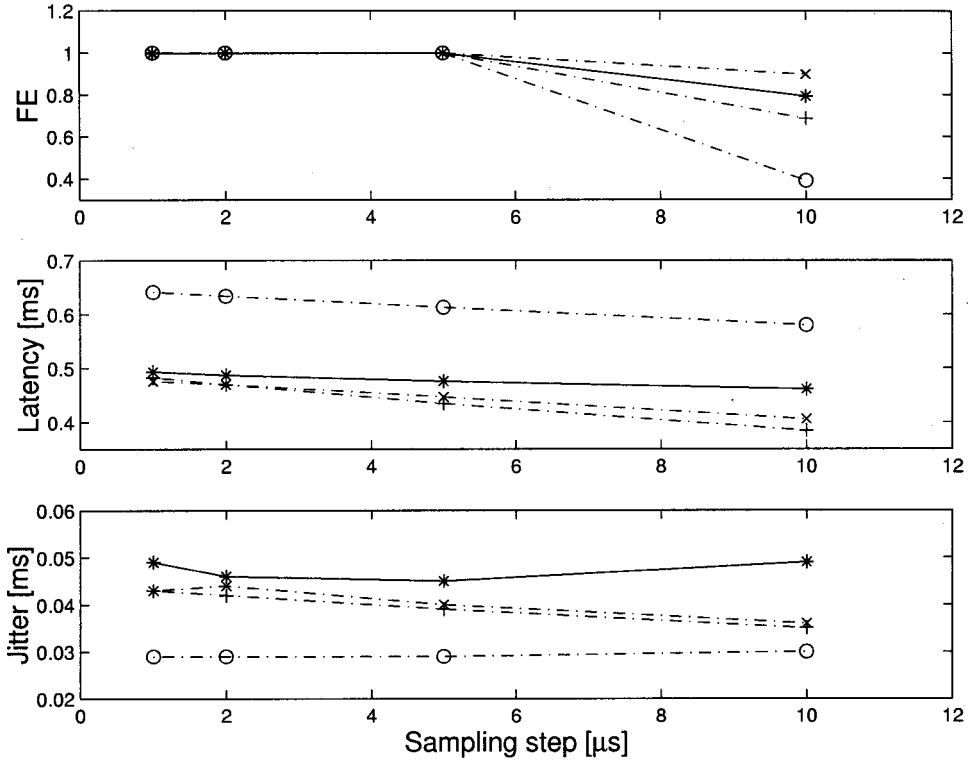

FIGURE 7. FE, latency, jitter vs. the sampling step [1, 2, 5, and 10 (μs)]. Those statistical parameters at each sampling step were estimated from 1000 Monte Carlo runs in which the preconditioned stimuli were presented. The data of CW, SD, R, and F algorithms are, respectively, plotted by the marks ∗, ×, +, and ○.

Algorithm for numerical simulations is to initialize the number of channels in each channel at time t_0, and then to repeat (i) generating lifetime, and (ii) updating the number of channels in the current and new states by determining which one of the state transitions occurs.

The probability density function of the lifetime staying in a specific state at $t=[t_0, t_{\max}]$ in multichannel systems is represented by

$$p(\tau,t)=\lambda(t)e^{-\lambda(t)\tau}, \tag{A1}$$

where the effective transition rate $\lambda(t)$ at t is expressed as

$$\lambda(t)=\sum_{i=0}^{3}\sum_{j=0}^{1} N_{m_ih_j}(t)\zeta_{ij}(t). \tag{A2}$$

In Eq. (A2), $N_{m_ih_j}(t)$ stands for the number of channels in state m_ih_j $(i=0,1,2,3;\ j=0,1)$, and $\zeta_{ij}(t)$ denotes the sum of transition rates associated with escapes from state m_ih_j $(i=0,1,2,3;\ j=0,1)$, where

$$\begin{cases} \zeta_{00}(t)=3\alpha_m+\alpha_h \\ \zeta_{10}(t)=\beta_m+\alpha_h+2\alpha_m \\ \zeta_{20}(t)=2\beta_m+\alpha_h+\alpha_m \\ \zeta_{30}(t)=3\beta_m+\alpha_h \\ \zeta_{01}(t)=3\alpha_m+\beta_h \\ \zeta_{11}(t)=\beta_m+\beta_h+2\alpha_m \\ \zeta_{21}(t)=2\beta_m+\beta_h+\alpha_m \\ \zeta_{31}(t)=3\beta_m+\beta_h \end{cases}. \tag{A3}$$

The lifetime t_l from the current state to the next state is a random variable sampled from a space specified by the exponential distribution of Eq. (A1), and is generated by a random number

$$t_l=-\frac{ln(u_1)}{\lambda(t)}, \tag{A4}$$

where u_1 denotes a random number uniformly distributed within $[0,1]$.

The number of channels in each state is tracked on the basis of the probability of state transition from the current state at $t=t_0$ to the next state at $t=t_0+t_l$. The

TABLE 1. Relative computation time of simulation algorithms.

	F	CW	SD	R
Computation time	1	7	32	39

state transition will occur after the lifetime t_l passes and which one of 20 state transitions occurs at $t=t_0+t_l$ is determined by generating a random number uniformly distributed within [0,1] and calculating the cumulative state transition probability

$$P_i(t_0)=\sum_{j=0}^{i}\eta_j/\lambda(t_0) \quad (i=0,1,\ldots,20), \qquad \text{(A5)}$$

where

$$\begin{cases}
\eta_0=0 & \\
\eta_1=3\alpha_m N_{m_0h_0} & (m_0h_0\rightarrow m_1h_0)\\
\eta_2=\beta_m N_{m_1h_0} & (m_1h_0\rightarrow m_0h_0)\\
\eta_3=2\alpha_m N_{m_1h_0} & (m_1h_0\rightarrow m_2h_0)\\
\eta_4=2\beta_m N_{m_2h_0} & (m_2h_0\rightarrow m_1h_0)\\
\eta_5=\alpha_m N_{m_2h_0} & (m_2h_0\rightarrow m_3h_0)\\
\eta_6=3\beta_m N_{m_3h_0} & (m_3h_0\rightarrow m_2h_0)\\
\eta_7=\alpha_h N_{m_0h_0} & (m_0h_0\rightarrow m_0h_1)\\
\eta_8=\beta_h N_{m_0h_1} & (m_0h_1\rightarrow m_0h_0)\\
\eta_9=\alpha_h N_{m_1h_0} & (m_1h_0\rightarrow m_1h_1)\\
\eta_{10}=\beta_h N_{m_1h_1} & (m_1h_1\rightarrow m_1h_0)\\
\eta_{11}=\alpha_h N_{m_2h_0} & (m_2h_0\rightarrow m_2h_1)\\
\eta_{12}=\beta_h N_{m_2h_1} & (m_2h_1\rightarrow m_2h_0)\\
\eta_{13}=\alpha_h N_{m_3h_0} & (m_3h_0\rightarrow m_3h_1)\\
\eta_{14}=\beta_h N_{m_3h_1} & (m_3h_1\rightarrow m_3h_0)\\
\eta_{15}=3\alpha_m N_{m_0h_1} & (m_0h_1\rightarrow m_1h_1)\\
\eta_{16}=\beta_m N_{m_1h_1} & (m_1h_1\rightarrow m_0h_1)\\
\eta_{17}=2\alpha_m N_{m_1h_1} & (m_1h_1\rightarrow m_2h_1)\\
\eta_{18}=2\beta_m N_{m_2h_1} & (m_2h_1\rightarrow m_1h_1)\\
\eta_{19}=\alpha_m N_{m_2h_1} & (m_2h_1\rightarrow m_3h_1)\\
\eta_{20}=3\beta_m N_{m_3h_1} & (m_3h_1\rightarrow m_2h_1)
\end{cases}. \qquad \text{(A6)}$$

If the random number generated is within $[P_{i-1}(t_0),P_i(t_0)]$, $(i=1,2,\ldots,20)$, then the ith state transition is regarded to occur at $t=t_0+t_l$ in which the ith state transition corresponds to that designated in parentheses in Eq. (A5). Once the state transition is determined, the number of channels at $t=t_0+t_l$ is updated such that one is subtracted from the number of channels in the current state as well as one is added to that in the next state. For instance, if the random number is within $[P_3(t_0),P_4(t_0)]$, then the state transition from the current state m_2h_0 to the next state m_1h_0 [see Eq. (A6)] occurs at $t=t_0+t_l$, and the number of channels in states m_2h_0 and m_1h_0 is updated as $N_{m_2h_0}:=N_{m_2h_0}-1$ and $N_{m_1h_0}:=N_{m_1h_0}+1$. This algorithm continues by coming back to lifetime generation, until time reaches at $t_{\max}$. Note that if the lifetime generated would be less than the next sampling time for action potential generation, the number of channels in each state should be updated.

REFERENCES

[1] Chow, C. C., and J. A. White. Spontaneous action potentials due to channel fluctuations. *Biophys. J.* 71:3013–3021, 1996.

[2] Clay, J. R., and L. J. DeFelice. Relationship between membrane excitability and single channel open-close kinetics. *Biophys. J.* 42:151–157, 1983.

[3] Colquhoun, D., and A. G. Hawkes. Relaxation and fluctuations of membrane currents that flow through drug-operated channels. *Proc. R. Soc. London, Ser. B* 199:231–262, 1977.

[4] Colquhoun, D., and A. G. Hawkes. On the stochastic properties of single ion channels. *Proc. R. Soc. London, Ser. B* 211:205–235, 1981.

[5] Fox, R. F. Stochastic versions of the Hodgkin–Huxley equations. *Biophys. J.* 72:2069–2074, 1997.

[6] Gillespie, D. T. Exact stochastic simulation of coupled chemical reactions. *J. Phys. Chem.* 81:2340–2361, 1977.

[7] Hodgkin, A. L., and A. F. Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *J. Physiol. (London)* 117:500–544, 1952.

[8] Lecar, H., and R. Nossal. Theory of threshold fluctuations in nerves. *Biophys. J.* 11:1048–1067, 1971.

[9] Matsuoka, A. J., J. T. Rubinstein, P. J. Abbas, and C. A. Miller. The effects of interpulse intervals on stochastic properties of electrical stimulation: Models and measurements. *IEEE Trans. Biomed. Eng.* 48:416–424, 2001.

[10] Neher, E., and C. F. Stevens. Conductance fluctuations and ionic pores in membranes. *Annu. Rev. Biophys. Bioeng.* 6:249–273, 1977.

[11] Papoulis, A. Probability, Random Variables, and Stochastic Processes, 2nd ed. Singapore: McGraw-Hill, 1985.

[12] Press, W. H., S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. Numerical Recipes in C: The Art of Scientific Computing. New York: Cambridge University Press, 1993.

[13] Rubinstein, J. T. Threshold fluctuations in an *N* sodium channel model of the node of ranvier. *Biophys. J.* 68:779–785, 1995.

[14] Sigworth, F. J. The variance of sodium current fluctuations at the node of ranvier. *J. Physiol. (London)* 307:97–129, 1980.

[15] Skaugen, E., and L. Walloe. Firing behaviour in a stochastic nerve membrane model based upon the Hodgkin–Huxley equations. *Acta. Physiol. Scand.* 49:343–363, 1979.

[16] Strassberg, A. F., and L. J. DeFelice. Limitation of the Hodgkin–Huxley formalism: Effects of single channel kinetics on transmembrane voltage dynamics. *Neural Comput.* 5:843–855, 1993.

[17] van Kampen, N. G. Stochastic Processes in Physics and Chemistry, 2nd ed. Amsterdam: Elsevier, 1992.

[18] Verveen, A. A. Fluctuation in Excitability. Amsterdam: Drukkerij, Holland N.V., 1961.

[19] White, J. A., J. T. Rubinstein, and A. R. Kay. Channel noise in neurons. *Trends Neurosci.* 23:131–137, 2000.