Fast Calculation of Synaptic Conductances

Rajagopal Srinivasan
Department of Electrical Engineering,
Case Western Reserve University, Cleveland, OH 44106 USA

Hillel J. Chiel
Departments of Biology and Neuroscience,
Case Western Reserve University, Cleveland, OH 44106 USA

Synaptic conductances are often modeled as sums of α functions

$$g(t) = \sum_{i=1}^{k} \frac{(t-t_i)}{\tau} e^{-(t-t_i)/\tau} \tag{1}$$

where t is the current time, t_i is the time of the ith spike in the presynaptic neuron, and τ is the time constant of the synapse. If the time of decay of the synapse, τ_{D}, is not equal to its time of onset, τ_{O}, the conductance at time t after k spikes have occurred is

$$g(t) = \left[\frac{\tau_{\mathrm{D}}\tau_{\mathrm{O}}}{\tau_{\mathrm{D}} - \tau_{\mathrm{O}}}\right] \sum_{i=1}^{k} \left(e^{-(t-t_i)/\tau_{\mathrm{D}}} - e^{-(t-t_i)/\tau_{\mathrm{O}}}\right) \tag{2}$$

The drawback of these solutions is that one must keep track of the times of occurrences of each spike that initiated the synaptic potentials, and recalculate each exponential in the summation at each time step. This creates a large storage and computational overhead. Since both these equations represent the impulse response of a second-order differential equation, another approach is to numerically integrate additional differential equations for each synapse in the network (Wilson and Bower 1989).

We have developed an improved method for computing synaptic conductances that separates equations 1 and 2 into two components: one that is a function of the current time of the simulation and one that accumulates the contributions of previous spike events to the synaptic conductance. We demonstrate that this method requires only the storage of two running sums and the time constants for each synapse, and that it is mathematically equivalent to equations 1 and 2. We will then demonstrate that it is also faster for a given level of precision than numerically integrating differential equations for each synapse. We will first describe our algorithm for equation 1, and then for equation 2.

Equation 1 can be rewritten as follows:

$$\begin{aligned}\sum_{i=1}^{k}\frac{(t-t_i)}{\tau}e^{-(t-t_i)/\tau} &= \frac{e^{-t/\tau}}{\tau}\sum_{i=1}^{k}(t-t_i)e^{t_i/\tau}\\ &= \frac{e^{-t/\tau}}{\tau}\left[t\sum_{i=1}^{k}e^{t_i/\tau}-\sum_{i=1}^{k}t_ie^{t_i/\tau}\right]\end{aligned} \tag{3}$$

When the $k+1$st spike occurs at time t_{k+1}, single terms can be added to each of the two summations in brackets to update them, eliminating the need to store spike times. To keep the exponentials inside the summation and outside the brackets from growing too large or small (respectively) as t increases over time, the exponentials can be rescaled. The left-hand term can be rewritten as

$$\begin{aligned}\frac{e^{-t/\tau}}{\tau}t\sum_{i=1}^{k}e^{t_i/\tau} &= \frac{e^{-(t-t_k)/\tau}}{\tau}e^{-t_k/\tau}t\sum_{i=1}^{k}e^{t_i/\tau}\\ &= \frac{e^{-(t-t_k)/\tau}}{\tau}t\sum_{i=1}^{k}e^{-(t_k-t_i)/\tau}\end{aligned} \tag{4}$$

We will refer to the terms within the summation as $\mathrm{Sum1}(t_k)$. It can be updated once the $k+1$st spike occurs as follows:

$$\mathrm{Sum1}(t_{k+1}) = e^{-(t_{k+1}-t_k)/\tau}\,\mathrm{Sum1}(t_k)+1 \tag{5}$$

because after the $k+1$st spike occurs,

$$\frac{e^{-(t-t_{k+1})/\tau}}{\tau}t\sum_{i=1}^{k+1}e^{-(t_{k+1}-t_i)/\tau} = \frac{e^{-(t-t_{k+1})/\tau}}{\tau}t\left[e^{-(t_{k+1}-t_k)/\tau}\sum_{i=1}^{k}e^{-(t_k-t_i)/\tau}+1\right] \tag{6}$$

The right-hand term in equation 3 can also be rewritten as

$$\frac{e^{-(t-t_k)/\tau}}{\tau}\sum_{i=1}^{k}t_ie^{(t_i-t_k)/\tau} \tag{7}$$

Once the $k+1$st spike occurs, the new form of the terms within the summation [which we refer to as $\mathrm{Sum2}(t_{k+1})$] would be

$$\sum_{i=1}^{k+1}t_ie^{(t_i-t_{k+1})/\tau} = e^{-(t_{k+1}-t_k)/\tau}\sum_{i=1}^{k}t_ie^{-(t_k-t_i)/\tau}+t_{k+1} \tag{8}$$

so that $\mathrm{Sum2}(t_k)$ is updated using the following rule:

$$\mathrm{Sum2}(t_{k+1}) = e^{-(t_{k+1}-t_k)/\tau}\,\mathrm{Sum2}(t_k)+t_{k+1} \tag{9}$$

Thus, at time $t > t_{k+1}$, from equations 3–9, the synaptic conductance is equal to

$$\frac{e^{-(t-t_{k+1})/\tau}}{\tau}\left[t(\mathrm{Sum1}(t_{k+1}))-(\mathrm{Sum2}(t_{k+1}))\right] \tag{10}$$

The conductance needs to be evaluated at each time step of the simulation, that is, from time t to time $t + \Delta t$. This can be accomplished by multiplying the term at time t, $e^{-(t-t_{k+1})/\tau}$, by $e^{-\Delta t/\tau}$, which yields the term at time $t + \Delta t$, $e^{-[(t+\Delta t)-t_{k+1}]/\tau}$. Thus, the conductances can be updated as follows:

$$\mathrm{Sum1}(t + \Delta t) = e^{-\Delta t/\tau}\, \mathrm{Sum1}(t) + S \tag{11}$$

$$\mathrm{Sum2}(t + \Delta t) = e^{-\Delta t/\tau}\, \mathrm{Sum2}(t) + St \tag{12}$$

where $S = 1$ if a spike occurred at time t and $S = 0$ otherwise. The synaptic conductance can then be calculated at time $t + \Delta t$ from

$$g = \frac{1}{\tau}[(t + \Delta t)\ \mathrm{Sum1}(t + \Delta t) - \ \mathrm{Sum2}(t + \Delta t)] \tag{13}$$

Equations 11, 12, and 13 summarize our algorithm for updating equation 1, which requires storage of only Sum1, Sum2, and τ. Since these equations are mathematically identical to equation 1, the accuracy of this method does not depend on the step size Δt, unless the step size becomes so large that spikes are missed. Of course, this constraint on step size is true for equation 1 as well.

By the same logic, equation 2 can be updated as follows:

$$\mathrm{Sum3}(t + \Delta t) = e^{-\Delta t/\tau_D}\mathrm{Sum3}(t) + S \tag{14}$$

$$\mathrm{Sum4}(t + \Delta t) = e^{-\Delta t/\tau_O}\mathrm{Sum4}(t) + S \tag{15}$$

where $S = 1$ if a spike occurred at time t and $S = 0$ otherwise, $\mathrm{Sum3}(t) = \sum_{i=1}^{k} e^{-(t-t_i)/\tau_D}$, and $\mathrm{Sum4}(t) = \sum_{i=1}^{k} e^{-(t-t_i)/\tau_O}$. The synaptic conductance can then be calculated at time $t + \Delta t$ from

$$g = \left[\frac{\tau_D \tau_O}{\tau_D - \tau_O}\right] [\mathrm{Sum1}(t + \Delta t) - \mathrm{Sum2}(t + \Delta t)] \tag{16}$$

Determining the value of the conductance requires only that Sum1, Sum2, τ_O, and τ_D be saved for each synapse. In addition, $\tau_D\tau_O/(\tau_D - \tau_O)$ is a constant for a given synapse, and therefore can be precalculated for each synapse.

This method requires far fewer exponentiations and additions than does the original closed-form solution (compare equations 1 and 2 to equations 11–16). Furthermore, the accuracy of our method is limited only by the machine precision. It also requires far less memory storage to maintain this accuracy. The number of spikes that must be stored to maintain the precision of equations 1 or 2 depends on (1) the spike frequency, (2) the synaptic time constant, and (3) the required precision level, ε. To ensure that a spike that has occurred at some time t_0 in the past will add less than ε to equation 1, it must be true that $((t - t_0/t)e < \varepsilon$. It can be shown that, for $\varepsilon < 10^{-3}$, setting $(t - t_0)/\tau$ equal to

$$P(\varepsilon) = \ln(1/\varepsilon) + [1 + 1/\ln(1/\varepsilon)][\ln(\ln(1/\varepsilon))] \tag{17}$$

will always satisfy this constraint. For example, if $\epsilon < 10^{-6}$, the value of $P(\varepsilon)$ would be 16.63, which implies that $t - t_0$ must be equal to 16.63τ, that is, spike t_0 must be stored until a time of 16.63τ has elapsed, after which its contribution to equation 1 will be less than 10^{-6}. If the time constant τ of this synapse is 50 msec, this requires that a spike be stored for 831.5 msec. The worst case size of the storage queue for a synapse would then be determined by this storage time, divided by the minimum period between spikes (which determines the maximum number of spikes that may occur during this storage time). If the input cell spikes with a minimum period of 100 msec between spikes, the queue would have to have room to store $831.5/100 \approx 9$ spike times. In general,

$$\text{Max queue size} = P(\varepsilon)\tau/T_{\min} \tag{18}$$

where $T_{\min}$ is the minimum firing period of the input cell. Thus, storage requirements for equation 1 or 2 increase logarithmically with increasing precision [since, as ε decreases, the fastest growing term in equation 17 is $\ln(1/\varepsilon)$], increase linearly with the synaptic time constant τ, and increase inversely with the minimum firing period $T_{\min}$. If one chooses to implement a queue dynamically, one has the computational overhead of keeping track of which spikes have aged sufficiently to be dropped. Whether one uses a fixed size array or a dynamic array, one must use more storage as the precision, input firing frequency, or time constant of a synapse increases.

How does our method compare to numerically integrating a second-order differential equation, injecting new impulses each time an action potential occurs in the presynaptic neuron? A variety of techniques exist for numerically integrating differential equations (Press *et al.* 1988). One efficient, stable, and fairly accurate technique that is frequently employed is referred to as the exponential technique. For the second-order differential equation yielding equation 1 or 2, applying this technique yields the following finite difference equations (Wilson and Bower 1989, p. 328):

$$\begin{aligned} z_{t+\Delta t} &= z_t e^{-\Delta t/\tau_D} + \frac{x(t)}{\Delta t}\tau_D(1 - e^{-\Delta t/\tau_D}) && (19) \\ g_{t+\Delta t} &= g_t e^{-\Delta t/\tau_O} + z_t\tau_O(1 - e^{-\Delta t/\tau_O}) && (20) \end{aligned}$$

where $x(t)$ is nonzero at the time a spike occurs, and zero otherwise. A drawback of this approach is that it is not inherently as precise as our method. One must trade off speed versus precision for equations 19–20. For example, if we choose to use a 1 msec time step for our method, in order to guarantee that deviations from it are smaller than 10^{-5}, we found that this numerical integration technique must be run with a step size 4 times smaller; to obtain deviations smaller than 10^{-6} requires a step size 8 times smaller, and to obtain deviations smaller than 10^{-7} requires a step size 16 times smaller. Of course, we could choose a larger step size for our method without loss of accuracy (see above), and the step sizes for the numerical integration technique would then be proportionally smaller.

We directly compared the time taken by the three methods by writing three benchmark programs in C (code listings available from the authors on request), and timing them on a Decstation 5000/200. As a reasonable precision limit for the methods, we chose 10^{-6}. For equation 1 and equations 11–13, a 1 msec step size was used. For equations 19–20, a 1 msec step size gave relatively poor precision (deviations were on the order of 10^{-4}); a step size of 0.125 msec was necessary to limit deviations to less than 10^{-6} from the other two methods. The time constant for the synapse was 50 msec, the input spike frequency was 10 Hz, and we chose a queue size for the method of equation 1 that would maintain spike times until they had decayed to values smaller than 10^{-6} (from equation 18, we determined that the queue should hold 9 spike times). Simulation time was 200 sec. Using these parameters, our method (equations 11–13) required only 2.1 sec of real time, whereas the method of equation 1 required 15.8 sec of real time, and the method of equations 19–20 required 18.1 sec of real time. These results suggest that our method is superior both in terms of speed and accuracy to previous methods.

Acknowledgments

R. S. was supported by a Research Experience for Undergraduates Supplement to H. J. C.'s NSF Grant, BNS 88-10757. H. J. C. thanks the NSF for its support of this research. We are grateful for the comments of Dr. Randall Beer and two anonymous reviewers on an earlier draft of this manuscript.

References

Press, W. H., Flannery, B. P., Teukolsky, S. A., and Vetterling, W. T. (eds.) 1988. *Numerical Recipes in* C. Cambridge University Press, Cambridge.

Wilson, M. A., and Bower, J. M. 1989. The simulation of large-scale neural networks. In *Methods in Neuronal Modeling,* C. Koch and I. Segev, eds., pp. 291–333. MIT Press, Cambridge.

Received 11 February 1992; accepted 3 August 1992.