LETTER Communicated by William Lytton

Exact Simulation of Integrate-and-Fire Models with Synaptic Conductances

Romain Brette
brette@di.ens.fr
Département d'Informatique, Equipe Odyssée, Ecole Normale Supérieure, 75230 Paris Cedex 05, France

Computational neuroscience relies heavily on the simulation of large networks of neuron models. There are essentially two simulation strategies: (1) using an approximation method (e.g., Runge-Kutta) with spike times binned to the time step and (2) calculating spike times exactly in an event-driven fashion. In large networks, the computation time of the best algorithm for either strategy scales linearly with the number of synapses, but each strategy has its own assets and constraints: approximation methods can be applied to any model but are inexact; exact simulation avoids numerical artifacts but is limited to simple models. Previous work has focused on improving the accuracy of approximation methods. In this article, we extend the range of models that can be simulated exactly to a more realistic model: an integrate-and-fire model with exponential synaptic conductances.

1 Introduction

There is an increasingly large body of evidence that in neurons, spike timing matters. Neurons have been shown to produce spike trains with submillisecond accuracy in vitro (Mainen & Sejnowski, 1995), a property that is shared by a large class of spiking neuron models (Brette & Guigon, 2003). Functional properties of in vivo neural networks rely on precise synchronization of neural discharges (e.g., in olfaction; Stopfer, Bhagavan, Smith, & Laurent, 1997). Synaptic plasticity depends on the relative timing of presynaptic and postsynaptic spikes (Abbott & Nelson, 2000), which has important functional implications (Song & Abbott, 2001). Therefore, spiking neuron models, in particular, the integrate-and-fire model (Lapicque, 1907; Knight, 1972) and variants, have become increasingly popular in computational neuroscience (Gerstner & Kistler, 2002), which has raised the problem of simulating them in an efficient and accurate way.

Assuming that only chemical synapses are considered (we will not consider electrical gap junctions in this article), neurons communicate with each other by spikes, which are discrete events. A spiking neuron model describes a transformation from a set of input spike trains into an output spike train. It typically comprises a set of state variables (e.g., the membrane potential) whose evolution is governed by a set of differential equations. Incoming spikes induce discrete changes in the state variables, and outgoing spikes are triggered by a threshold condition. One of the simplest spiking models is the leaky integrate-and-fire model with instantaneous synaptic interactions, which is described by one state variable, the membrane potential V, governed by the following equation,

$$\tau \frac{dV}{dt} = -(V - V_0), \tag{1.1}$$

where τ is the membrane time constant and V_0 is the resting potential. A spike coming from synapse i at time t induces an instantaneous change in the potential: $V \to V + w_i$, where w_i is the weight of synapse i. A spike is triggered when V reaches the threshold V_t, when it is instantaneously reset to V_r. More complex models include conductances, in particular, synaptic conductances, for example,

$$\tau \frac{dV}{dt} = -(V - V_0) - \sum_{ij} w_i g(t - t_{ij})(V - E_s), \tag{1.2}$$

where t_{ij} is the time of the jth spike coming from synapse i, E_s is the synaptic reversal potential, and $g(\cdot)$ is the unitary conductance triggered by a single spike. Although it would seem that simulating such a model requires storing spike times, biophysical models can usually be reformulated in the form of a spiking model as described above—a set of differential equations with spikes triggering discrete changes in the state variables (Destexhe, Mainen, & Sejnowski, 1994). For example, equation 1.2 with exponential conductances $g(s) = \exp(-s/\tau_s)$ can be rewritten as

$$\begin{aligned}\tau \frac{dV}{dt} &= -(V - V_0) - g \times (V - E_s)\\ \tau_s \frac{dg}{dt} &= -g,\end{aligned}$$

and each spike coming from synapse i at time t_{ij} triggers an instantaneous change in the total synaptic conductance $g \to g + w_i$. With this formulation, storing spike times is not necessary (unless transmission delays are considered). Transformations of this kind can be applied to all reasonable synaptic models (e.g., $\alpha-$functions), and in many cases, the number of state variables grows with the number of synapse types, not with the number of synapses (Lytton, 1996).

There are essentially two strategies to simulate networks of spiking models: (1) integrating the differential equations with an approximation method (e.g., Euler, Runge-Kutta), advancing the simulation by fixed time steps and communicating the spikes only at the boundaries of the steps; and (2) simulating the network exactly (up to machine precision) in an event-driven way, that is, advancing the simulation from spike to spike and calculating spike times exactly. By "exact," we mean that results are derived from analytical formulas but are stored with the precision of the floating-point representation of the machine (this is not symbolic computation). There exist hybrid strategies such as independent adaptive time-stepping methods (Lytton & Hines, 2005), which we will not discuss here. Both strategies perform similarly in terms of simulation time, as we will show in the next section, but they are not equivalent. The greatest asset of the approximation strategy is that it can be applied to any model. However, by definition, it is imprecise. It may not seem very relevant at first sight because real neurons are noisy, but numerical error is not equivalent to random noise. For example, Hansel, Mato, Meunier, and Neltner (1998) showed that networks of spiking neurons could fail to display synchronization properties when implemented with a naive Euler or Runge-Kutta algorithm, which led them to improve it with a better handling of the reset (see also Shelley & Tao, 2001). It is generally hard to predict how small errors in spike times are reverberated in recurrent networks, whether they are amplified to create numerical artefacts or remain irrelevant, and this is certainly an important issue for research purposes. Exact simulation obviously avoids this problem and makes results perfectly reproducible, but it applies to a limited range of models—essentially, to models with instantaneous interactions (Mattia & Del Giudice, 2000) or, at best, to linear models with synaptic currents (Makino, 2003). The purpose of this article is to extend exact simulation to more realistic models with noninstantaneous synaptic conductances.

We will describe a method to simulate exactly a common spiking model: the leaky integrate-and-fire model with exponential synaptic conductances, in which the membrane potential V is governed by the following differential equation,

$$\tau \frac{dV}{dt} = -(V - V_0) - g^+(t)(V - E^+) - g^-(t)(V - E^-), \tag{1.3}$$

where $g^+(\cdot)$ (resp. $g^-(\cdot)$) is the total excitatory (resp. inhibitory) conductance (relative to the leak conductance) and E^+ is the excitatory (resp. inhibitory) reversal potential. Both conductances are described as sums of spike-triggered exponential conductances,

$$g(t) = \sum_{ij} w_i \exp(-(t - t_{ij})/\tau_s)\Theta(t - t_{ij}), \tag{1.4}$$

where $\Theta(s) = 1$ if $s \geq 0$ and $\Theta(s) = 0$ otherwise, and τ_s is the synaptic time constant. The constraint of the method we present is that both excitatory and inhibitory conductances must have the same synaptic time constant τ_s. This is an important limitation, but it still significantly advances the boundaries of exact simulation in terms of biophysical realism.

In the next section, we analyze the computational costs of the two simulation strategies and show that for large networks, the simulation time grows linearly with the number of synapses. Then we will describe our method to simulate exactly integrate-and-fire models with exponential conductances. Finally, we will test it in two settings: (1) a single neuron with spike-time-dependent plasticity and excitatory inputs (as in Song, Miller, & Abbott, 2000) and (2) random networks of excitatory and inhibitory neurons (as in Brunel, 2000, but with conductances instead of currents).

2 Computational Complexity of Network Simulations

In order to simplify the analysis, we will ignore the problem of managing transmission delays and consider only simulations on a single CPU. We analyze the time required to simulate 1 second of biological time for a network comprising N neurons, p synapses per neuron, with average firing rate F.

Before analyzing separately the simulation time of both strategies, we first note that any algorithm must make at least $F \times N \times p$ operations per second of biological time: indeed, the network produces on average $F \times N$ spikes per second, and each of these needs to be sent to p target neurons. Therefore, the computational cost of any algorithm is at least linear in the number of synapses.

2.1 Approximation Methods. A step $t \to t + dt$ of a typical approximation algorithm consists of two phases:

Update: The state variables of all neurons are updated with an integration method (e.g., Euler, Runge-Kutta), and the neurons that are going to spike are identified by a threshold condition ($V > V_t$).

Propagation: Spikes are propagated to the targets of these neurons.

Assuming that the number of state variables does not grow with the number of neurons or synapses, which is usually the case (see Lytton, 1996), the cost of the update phase is of order N for each step, so it is $O(N/dt)$ per second of biological time. This component grows with the complexity of the neuron models and the precision of the simulation.

Every second (biological time), an average of $F \times N$ spikes is produced by the neurons. Each of these needs to be propagated to p target neurons (*propagation* meaning changing the state variables of the target neurons).

Thus, the propagation phase consists of $F \times N \times p$ spike propagations per second. These are essentially additions of weights w_i to state variables, and thus are simple operations whose cost does not grow with the complexity of the models.

Summing up, the total computational cost per second of biological time is of the order

$$\begin{array}{l} \textit{Update} + \textit{Propagation} \\ c_U \times \frac{N}{dt} + c_P \times F \times N \times p, \end{array}$$

where c_U is the cost of one update and c_P is the cost of one spike propagation; typically, c_U is much higher than c_P, but this is implementation dependent. Therefore, for very dense networks, the total is dominated by the propagation phase and is linear in the number of synapses, which is optimal. In practice, however, the first phase is negligible when the following condition is met:

$$\frac{c_P}{c_U} \times F \times p \times dt \gg 1 \qquad (*).$$

For example, the average firing rate in the cortex might be as low as $F = 1$ Hz (Olshausen & Field, 2005), and assuming $p = 10,000$ synapses per neuron and $dt = 0.1$ ms, we get $F \times p \times dt = 1$. In this case, considering that each operation in the update phase is heavier than in the propagation phase (especially for complex models), that is, $c_P < c_U$, the former is likely to dominate the total computational cost. Thus, it appears that even in networks with realistic connectivity, increases in precision (smaller dt) can be detrimental to the efficiency of the simulation. One way to get around this problem is to use higher-order integration methods, allowing larger time steps for comparable levels of precision. Several authors (Hansel et al., 1998; Shelley & Tao, 2001) have noticed that for integrate-and-fire models, the discontinuity in the state variables at spike time annihilates the increase in precision associated with higher-order integration methods (e.g., second-order Runge-Kutta), but they have proposed a correction of the reset that solves this problem. Considering that the additional cost is negligible (of order $F \times N$), this correction should always be used in simulations. We note, however, that it imposes smoothness constraints on the models, which implies higher values of c_U.

2.2 Exact Simulation. Exact simulation of networks of spiking models fits in the more general setting of the simulation of discrete event systems. Implementations are typically more complicated than their approximate analog, but they are well established because they are not specific to neural networks (see, e.g., Ziegler, Praehofer, & Kim, 2000). We can describe a step of a typical exact event-driven algorithm as follows:

1. Determine the next event, that is, which neuron is going to spike next.
2. Update the state variables of this neuron.
3. Propagate the spike, that is, update the state variables of the target neurons.

In order to determine the next event, one needs to maintain an ordered list of future events. This list contains the time of the next spike for every neuron. Spike times are conditional to the fact that no spike is received in the meantime, but causality implies that the earliest spike time in the list is always valid. In terms of data structure, this list is a priority queue. For every event, we need to extract the highest-priority item and insert or modify p items in this queue. There exist data structures and algorithms that can implement these two operations in constant ($O(1)$) time, for example, calendar queues (Brown, 1988). These are similar to the ring buffers used in Morrison, Mehring, Geisel, Aertsen, and Diesmann (2005), except there is no fixed time step (schematically, events are stored in a calendar queue as in an agenda, but the duration of days can change). We can subdivide the computational cost of handling one event as follows:

- Updating the neuron and its targets: $p+1$ updates (the cost is modulated by model complexity)
- Updating the spike times of $p+1$ neurons (again, the cost is modulated by model complexity)
- Extracting the highest-priority event and inserting p events in the queue: $p+1$ operations (depends on the implementation of the priority queue)

Since there are $F \times N$ spikes per second of biological time, the computational cost is approximately proportional to $F \times N \times p$. The total computational cost per second of biological time can be written concisely as follows:

$$\begin{array}{l} \mathit{Update} + \mathit{Spike} + \mathit{Queue} \\ \quad (c_U + c_S + c_Q) \times F \times N \times p, \end{array}$$

where c_U is the cost of one update of the state variables, c_S is the cost of calculating the time of the next spike, and c_Q is the cost of inserting an event in the priority queue. Thus, the simulation time is linear in the number of synapses, which is optimal. Nevertheless, we note that the operations involved are heavier than in the propagation phase of approximation methods (see the previous section); therefore, the multiplicative factor is likely to be larger. However, in the worst case, exact simulation is slower than approximate simulation only by a constant multiplicative factor, and it can outperform it in cases when condition ($*$) is not met. Therefore, it seems advisable to use exact simulation when possible.

Several exact simulation engines have been developed specifically for spiking neural networks that can handle propagation delays and complex network structures (Mattia & Del Giudice, 2000; Rochel & Martinez, 2003). Considerable effort has been devoted to the efficient management of event queues (Lee & Farhat, 2001; Mattia & Del Giudice, 2000; Connolly, Marian, & Reilly, 2003), and event-handling algorithms can be made parallel (see, e.g., Grassmann & Anlauf, 1998). In principle, general clock-driven simulation engines such as NEST (Morrison et al., 2005) can also handle exact simulation, provided there is a minimum transmission delay (larger than the time step), although they are probably less efficient than dedicated event-driven simulation engines (but one may appreciate the greater expressivity).

To simulate a particular neuron model exactly, one needs to provide the simulator with three functions:

- A function that updates the neuron following an incoming spike
- A function that updates the neuron following an outgoing spike (reset)
- A function that gives the time of the next spike (possibly $+\infty$ if there is none), provided the values of the state variables

So far, algorithms have been developed for simple pulse-coupled integrate-and-fire models (Claverol, Brown, & Chad, 2002; Delorme & Thorpe, 2003) and more complex ones such as some instances of the spike response model (Makino, 2003; Marian, Reilly, & Mackey, 2002; Gerstner & Kistler, 2002), but there is none for more realistic models with synaptic conductances, which are often used in computational neuroscience.

3 Exact Calculations for the IF Model with Conductances

In this section we describe the functions required to simulate an integrate-and-fire model with exponential synaptic conductances, as defined by equations 1.3 and 1.4. First, we reformulate the model into a standard spiking neuron model with two dynamic state variables. Then we calculate the solution of this couple of differential equations given the initial state. Finally, we describe a method to calculate the time of the next spike, with a quick test to check whether it is $+\infty$ (i.e., no spike).

3.1 Rewriting the Model. We start by expressing equations 1.3 and 1.4 as a spiking model as described in section 1—a set of autonomous differential equations with incoming spikes triggering instantaneous changes in the state variables. First, we express time in units of the membrane time constant τ, with $t = 0$ as the origin of time, and voltage relatively to the resting potential V_0, that is, we assume $V_0 = 0$.

We can rewrite equation 1.3 as follows:

$$\frac{dV}{dt} = -V + (g^+(t) + g^-(t))(E_s(t) - V), \tag{3.1}$$

where $E_s(t)$ is the effective synaptic reversal potential defined as

$$E_s(t) = \frac{g^+(t)E^+ + g^-(t)E^-}{g^+(t) + g^-(t)}.$$

In intervals with no spike, $E_s(t)$ is constant. Indeed, $g^+(\cdot)$ and $g^-(\cdot)$ both satisfy

$$\tau_s \frac{dg^+}{dt} = -g^+$$
$$\tau_s \frac{dg^-}{dt} = -g^-,$$

and it follows, assuming no spike in $[0, t]$,

$$\begin{aligned} E_s(t) &= \frac{g^+(0)e^{-t/\tau_s}E^+ + g^-(0)e^{-t/\tau_s}E^-}{g^+(0)e^{-t/\tau_s} + g^-(0)e^{-t/\tau_s}} \\ &= \frac{g^+(0)E^+ + g^-(0)E^-}{g^+(0) + g^-(0)}. \end{aligned}$$

We define the total synaptic conductance as $g(t) = g^+(t) + g^-(t)$. Then we can rewrite the model as a system of two differential equations,

$$\frac{dV}{dt} = -V + g(E_s - V) \tag{3.2a}$$

$$\tau_s \frac{dg}{dt} = -g, \tag{3.2b}$$

and both g and E_s are instantaneously modified when a spike is received. The reset conditions follow from simple calculations:

- When a spike is received from an inhibitory synapse with weight $w > 0$:

  $$E_s \to \frac{gE_s + wE^-}{g + w}$$
  $$g \to g + w$$

  (note that the order is important)

- When a spike is received from an excitatory synapse with weight $w > 0$:

$$E_s \to \frac{gE_s + wE^+}{g + w}$$
$$g \to g + w$$

- When V reaches the threshold V_t:

$$V \to V_r$$

Thus, we have reformulated the integrate-and-fire model with exponential synaptic conductances as a two-variable spiking model (i.e., an autonomous system of two differential equations with incoming spikes triggering instantaneous resets). We can sum up the first two reset conditions in a single condition by using signed weights (i.e., $w < 0$ for spikes coming from inhibitory synapses) as follows:

$$E_s \to \frac{gE_s + \alpha w + \beta |w|}{g + |w|}$$
$$g \to g + |w|$$

with $\alpha = (E^+ - E^-)/2$ and $\beta = (E^+ + E^-)/2$.

3.2 Solution of the Differential Equation. In equation 3.2a, let us express V as a function of g:

$$\frac{dV}{dg} = \tau_s \left(1 + \frac{1}{g}\right) V - \tau_s E_s.$$

It follows that

$$\frac{d}{dg}(V \exp(-\tau_s(g + \log g))) = -\tau_s E_s \exp(-\tau_s(g + \log g)).$$

Integrating between $g(0)$ and $g(t)$, we get:

$$\begin{aligned}
&V(t) \exp(-\tau_s g(t)) g(t)^{-\tau_s} - V(0) \exp(-\tau_s g(0)) g(0)^{-\tau_s} \\
&\quad = -\tau_s E_s \int_{g(0)}^{g(t)} g^{-\tau_s} \exp(-\tau_s g) dg \\
&\quad = -\tau_s^{\tau_s} E_s \int_{\tau_s g(0)}^{\tau_s g(t)} h^{-\tau_s} e^{-h} dh \\
&\quad = -\tau_s^{\tau_s} E_s (\gamma(1 - \tau_s, \tau_s g(t)) - \gamma(1 - \tau_s, \tau_s g(0))),
\end{aligned}$$

where

$$\gamma(a, x) = \int_0^x e^{-t} t^{a-1} dt$$

is the nonnormalized incomplete gamma integral. The incomplete gamma integral has fast computation algorithms, which are implemented in most numerical libraries. In the following, we will use the algorithm from Press, Flannery, Teukolsky, and Vetterling (1993). Recalling that $g(t) = g(0)e^{-t/\tau_s}$, we get:

$$\begin{aligned}&g(0)^{-\tau_s} \exp\left(t - \tau_s g(0)e^{-t/\tau_s}\right) V(t)\\&= V(0)e^{-\tau_s g(0)} g(0)^{-\tau_s} - \tau_s^{\tau_s} E_s\left(\gamma\left(1 - \tau_s, \tau_s g(t)\right) - \gamma\left(1 - \tau_s, \tau_s g(0)\right)\right).\end{aligned}$$

The incomplete gamma integral can be developed as either a power series,

$$\gamma(a, x) = e^{-x} x^a \sum_{n=0}^{\infty} \frac{\Gamma(a)}{\Gamma(a+1+n)} x^n, \tag{3.3}$$

where

$$\Gamma(a) = \int_0^\infty e^{-t} t^{a-1} dt$$

is the gamma integral, or a continued fraction,

$$\gamma(a, x) = e^{-x} x^a \Gamma(a) \left(\frac{1}{x+1-a-} \frac{1 \cdot (1-a)}{x+3-a-} \frac{2 \cdot (2-a)}{x+5-a-} \cdots \right). \tag{3.4}$$

It appears that in both expressions, we can factorize by $e^{-x}x^a$. Let us define $\rho(a, x) = e^x x^{-a} \gamma(a, x)$. Then we can rewrite:

$$\begin{aligned}\tau_s^{\tau_s} E_s \gamma(1 - \tau_s, \tau_s g(0)) &= \tau_s E_s g(0) \rho(1 - \tau_s, \tau_s g(0)) e^{-\tau_s g(0)} g(0)^{-\tau_s}\\ \tau_s^{\tau_s} E_s \gamma(1 - \tau_s, \tau_s g(t)) &= \tau_s E_s g(t) \rho(1 - \tau_s, \tau_s g(t)) g(0)^{-\tau_s} \exp(t - \tau_s g(0) e^{-t/\tau_s}).\end{aligned}$$

Thus we get:

$$\begin{aligned}&e^{t-\tau_s g(t)}(V(t) + \tau_s E_s g(t) \rho(1 - \tau_s, \tau_s g(t)))\\&= e^{-\tau_s g(0)}(V(0) + \tau_s E_s g(0) \rho(1 - \tau_s, \tau_s g(0))).\end{aligned}$$

Finally, we get the following expression for $V(t)$:

$$V(t) = -\tau_s E_s g(t)\rho(1-\tau_s, \tau_s g(t)) + \exp(-t + \tau_s(g(t) - g(0)))(V(0) + \tau_s E_s g(0)\rho(1-\tau_s, \tau_s g(0))). \quad (3.5)$$

Thus, we can calculate the value of the solution at any time very efficiently. It turns out that the power series, equation 3.3, converges rapidly for $x < a + 1$, while the continued fraction, equation 3.4, is more efficient for $x > a + 1$. Although at first sight it may seem that using an infinite series or continuous fraction makes the calculation very heavy, it is in fact not fundamentally different from calculating an exponential function. Only the first terms are indeed necessary to achieve an accuracy close to the precision of the floating-point representation.

It is possible to use precalculated tables for the exponential function and incomplete gamma integral to calculate expression 3.5. If linear interpolation is used, one can obtain almost exact results. We tested the accuracy of formula 3.5 calculated with tables with linear interpolation for the exponential function and the ρ function (more precisely, the function $g \mapsto \tau_s g \times \rho(1-\tau_s, \tau_s g)$). The results are shown in Table 1: with conductances tabulated with precision 10^{-4} (in units of the leak conductance), the relative accuracy for the membrane potential is of order 10^{-9}, 1000 times better than with a second-order Runge-Kutta (RK) method with $dt = 0.01$ ms (10 million times better than RK with $dt = 1$ ms; note that time steps lower than 0.1 ms are likely to slow the simulation; see the previous section). Thus, it is certainly acceptable to use precalculated tables to speed up the simulation while keeping the calculations almost exact. The speed-up factor for computing $V(t)$ was about 10 (compared to using the series expansion 3.3 with relative precision 10^{-15}).

3.3 Spike Test. In many situations, the time of the next spike for a given neuron (ignoring future incoming spikes) will be $+\infty$ most of the time, because the distribution of the membrane potential in cortical networks usually displays an approximately gaussian distribution centered far from the threshold (see, e.g., Destexhe, Rudolph, Fellous, & Sejnowski, 2001). This phenomenon appears clearly in the simulations (see the next section). Here we will describe an algorithm that can quickly test whether the neuron is going to spike.

We assume that $E_s > V_t$; otherwise, there can be no spike. The dynamics of the differential system, equations 3.2a and 3.2b, and therefore whether the membrane potential reaches the threshold V_t after some time, is completely determined by the initial condition $(V(0), g(0))$. If the model spikes for an initial condition (V_0, g_0), then it does so for any initial condition (V_1, g_0) such that $V_0 < V_1$. Thus, there is a minimal potential $V^*(g_0)$ above which

Table 1: Accuracy of the Calculation of $V(t)$ with Precalculated Tables (Using Equation 3.5 and with the Second-Order Runge-Kutta Method, for $t = \tau_s$ and $t = 5\tau_s$ ($\tau_s = 3$, $\tau = 20$ ms)).

		Table 10^{-3}	Table 10^{-4}	RK 1 ms	RK 0.01 ms
$t = \tau_s$	Accuracy	$1.9 \pm 1.2 \times 10^{-7}$	$1.9 \pm 1.2 \times 10^{-9}$	$1.9 \pm 1.2 \times 10^{-2}$	$1.6 \pm 1 \times 10^{-6}$
	Speed (Hz)	8.7×10^{6}	8.8×10^{6}	5.9×10^{6}	8.2×10^{4}
$t = 5\tau_s$	Accuracy	$8.5 \pm 5.2 \times 10^{-8}$	$8.5 \pm 5.2 \times 10^{-10}$	$1.2 \pm 0.6 \times 10^{-2}$	$1 \pm 0.5 \times 10^{-6}$
	Speed	8.8×10^{6}	8.8×10^{6}	1.5×10^{6}	1.6×10^{4}

Notes: The initial potential $V(0)$ is picked at random from a uniform distribution between 0 and 1 (normalized potential); the initial conductance is also picked from a uniform distribution, between 0.5 and 5.0. The mean and standard deviation are calculated from 10^5 iterations. We chose $\tau_s = 3$ ms and $\tau = 20$ ms. We used two levels of precision for the tables: for the first column, 10^{-3} for conductance (in units of the conductance leak) and 10^{-3} for time (in units of the membrane time constant); for the second column, 10^{-4} for conductance and 10^{-4} for time. The true value for $V(t)$ was calculated using the series formula for the incomplete gamma integral, with relative precision 10^{-15}. For the Runge-Kutta method, we used time step 1 ms and 0.01 ms. The speed is indicated as the number of operations per second. Individual update operations were about two to three times faster for the Runge-Kutta method than for the quasi-exact method with tables.

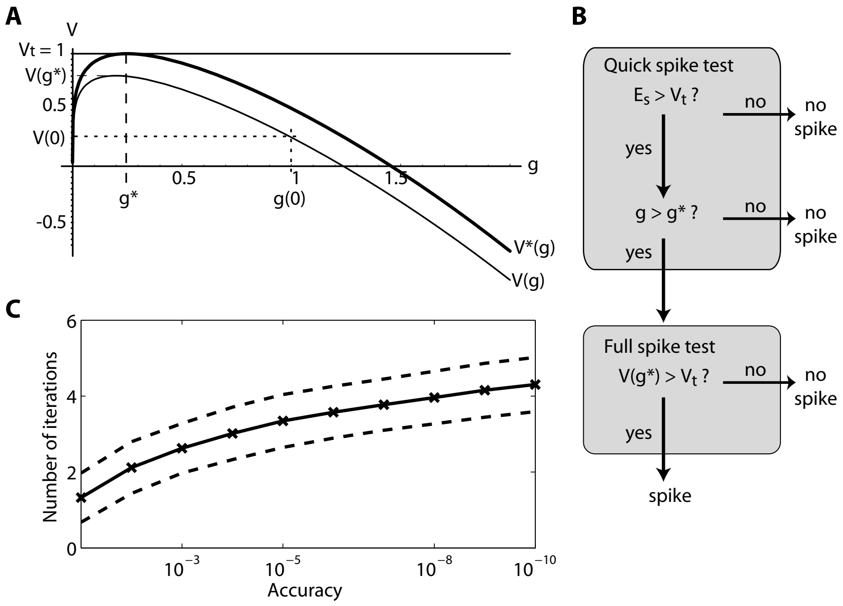

Figure 1: Computation of spike timings. (A) Minimal potential for spiking V^* as a function of the synaptic conductance g (see text). The solution $V(g)$ is below V^*; hence, it never hits the threshold (no spike). We used $V_t = 1$, $E_s = 5$, $\tau_s = 1/4$, $g(0) = 1$, $V(0) = 0.2$. (The time goes from right to left, since g decreases with time.) (B) Summary of the spike test algorithm. (C) Number of iterations in spike timing computation as a function of accuracy. The initial potential $V(0)$ is picked at random from a uniform distribution between 0 and 1 (normalized potential); the initial conductance is also picked from a uniform distribution, between 2.5 and 5 (we chose values high enough to guarantee spiking). The mean and variance are calculated from 10^5 iterations (the dashed lines represent mean $\pm$ standard deviation). Accuracy is calculated on the potential axis; if t is the computed spike time and ε is the accuracy, $V(t)$ is within ε of the threshold.

the neuron spikes. The set of points

$$\mathcal{C}^* = \{(V^*(g), g)\}, g \geq 0\}$$

defines the minimum spiking curve (see Figure 1A); that is, the neuron spikes if and only if its state (V, g) is above this curve.

We need to calculate this curve. Consider the trajectory in the phase space of a solution $V(g)$ starting on $\mathcal{C}^*$ from $(V^*(g_0), g_0)$. By definition, all solutions starting from (V_0, g_0) with $V_0 > V^*(g_0)$ hit the threshold, and all solutions starting from (V_0, g_0) with $V_0 < V^*(g_0)$ do not. Because the phase space is two-dimensional, trajectories cannot cross. It follows that any trajectory that hits the threshold must be above the trajectory of $V(g)$ at all times, and conversely. Therefore, the trajectory of $V(g)$ is precisely the minimum spiking curve $\mathcal{C}^*$. Besides, this trajectory must be tangential to the threshold $V = V_t$; otherwise, there would be a trajectory below it that hits the threshold. Therefore, the minimal potential $V^*(g)$ is the solution of the following differential equation,

$$\frac{dV^*}{dg} = \tau_s\left(1 + \frac{1}{g}\right)V^* - \tau_s E_s,$$

such that $dV/dg = 0$ at threshold, that is, with g^* being the conductance at threshold:

$$\begin{aligned} 0 &= \left(1 + \frac{1}{g^*}\right)V_t - E_s \\ g^* &= \frac{1}{(E_s/V_t) - 1}. \end{aligned}$$

Note that in Figure 1A, solutions travel from right to left in the phase space. Therefore, for $g(0) < g^*$, there can be no spike in the future (the solution hits threshold at a negative time), so that g^* is the minimal conductance for spiking. To test whether the trajectory of the membrane potential is above or below the trajectory of the minimal potential, we only need to compare $V(t)$ at the time when $g(t) = g^*$ with the threshold V_t. We can calculate this value using equation 3.5:

$$\begin{aligned} V(g^*) &= -\tau_s E_s g^* \rho(1 - \tau_s, \tau_s g^*) \\ &\quad + \left(\frac{g^*}{g(0)}\right)^{\tau_s} \exp(\tau_s(g^* - g(0)))(V(0) + \tau_s E_s g(0)\rho(1 - \tau_s, \tau_s g(0)) \end{aligned} \tag{3.6}$$

The neuron spikes if and only if $V(g^*) > V_t$. In the worst case, the algorithm takes about the same time as updating the membrane potential. In many cases, the simple checks $E_s > V_t$ and $g > g^*$ are sufficient most of the time (for example, 86% of the time for the random networks simulated in the next section). The algorithm is summed up in Figure 1B.

3.4 Computation of Spike Timing. When the spike test is positive, we need to compute the timing of the spike. This can be done with very high accuracy using the Newton-Raphson method.

One can easily show that the potential $V(\cdot)$ is first increasing, then decreasing (the second derivative is negative when the first derivative cancels out), which implies that there are two threshold crossings. The spike corresponds to the first crossing, on the increasing part of the trajectory. On this part, the trajectory is concave (derive equation 3.2a); therefore, the Newton-Raphson method converges to the first crossing, which is the correct one.

At every iteration of the Newton-Raphson method, we need to evaluate the membrane potential at the current approximation of the spike timing. Note that only one evaluation of the incomplete gamma integral is then necessary, since the one for time 0 is the same for all iterations. Numerical results show that this method converges very quickly in the present case. Figure 1C shows that extremely high accuracy is reached after just four iterations. Considering that in many situations (see the next section with simulation results) spike timings are computed for only a few percentages of all events, computing spike times is expected to have a small impact on the performance of the simulation.

3.5 Summary: Three Functions for Exact Simulation. In section 2.2, we indicated that a simulator needs three functions to simulate a model exactly (recall that since we are not doing symbolic calculus, "exact" should be understood as several orders of magnitude more accurate than approximation methods—we discuss this matter further in section 5). Here we sum up our results and describe these functions.

The state of a neuron is determined by three variables: the membrane potential V, the total synaptic conductance g, and the effective synaptic reversal potential E_s. The first two variables evolve continuously between spikes according to a system of differential equations (3.2a and 3.2b). Variable V is reset when a spike is produced (threshold condition $V > V_t$), and variables g and E_s are updated when a spike is received.

3.5.1 Update After an Incoming Spike. When a spike is received with signed weight w (i.e., $w < 0$ when the synapse is inhibitory) at time t, the following operations are executed, given that t_l is the time of the last update:

1. $V \to V(t - t_l)$ using formula 3.5 with $V(0) = V$ and $g(0) = g$.
2. $g \to g \times \exp(-(t - t_l)/\tau_s)$ (note that the calculation has actually been done when updating V).
3. $E_s \to \frac{gE_s+\alpha w+\beta|w|}{g+|w|}$ with $\alpha = (E^+ - E^-)/2$ and $\beta = (E^+ + E^-)/2$.
4. $g \to g + |w|$.

3.5.2 Update After an Outgoing Spike. When a spike is produced at time t, the following operations are executed, given that t_l is the time of the last update:

1. $V \to V_r$.
2. $g \to g \times \exp(-(t - t_l)/\tau_s)$.

3.5.3 Spike Timing. To calculate the time of the next spike, we first check whether this is $+\infty$ according to the algorithm summed up in Figure 1B. If it is finite (i.e., if the neuron is going to spike), then spike timing is computed by the Newton-Raphson method (about four iterations of formula 3.5), which is guaranteed to converge to the correct value.

4 Simulations

Models were simulated on a standard desktop PC with a general event-driven simulator called MVASpike (developed by Olivier Rochel and freely available online at http://www.comp.leeds.ac.uk/olivierr/mvaspike/). The C code for the functions discussed in this article is available on the author's Web page (http://www.di.ens.fr/~brette/papers/Brette2005NC.htm).

To give an idea of typical simulation times, we provide some figures for the coefficients in the formulas of section 2 (complexity). In our simulations, the cost c_U of one update of the membrane potential was approximately:

$$c_U\,(\text{exact}) \approx 1\,\mu s \quad \text{series expansion (precision } 10^{-15})$$
$$c_U\,(\text{table}) \approx 0.1\,\mu s \quad \text{precalculated tables}$$
$$c_U\,(\text{RK2}) \approx 0.04\,\mu s \quad \text{second-order Runge-Kutta}$$

Calculating the timing of the next spike is at most $c_S \approx c_U \approx 0.1\,\mu$s if the spike test is negative (which occurred most of the time in our simulations; see below), and about $c_S \approx 3c_U - 4c_U \approx 0.3\,\mu\text{s} - 0.4\,\mu$s (with tables) if the spike test is positive. The cost of one queue insertion c_Q depends on the structure of the network (and obviously on the implementation of priority queues) and can be fairly high when the network has heterogeneous transmission delays. For example, for the random network described in section 4.2, and with the software we used for event management, $c_Q \approx 4\,\mu\text{s} - 5\,\mu$s, so that the program spends most of its time managing events and delays. In this case, the problem of dealing with transmission delays is the same with an approximation method—$c_P \approx c_Q$. When delays are homogeneous, there can be a drastic reduction in computation time because for a given outgoing spike, the p corresponding events are inserted at the same place in the queue.

4.1 Single Neuron with STDP

4.1.1 The Model. We simulated the scenario depicted in Song and Abbott, 2001, (Fig. 1.B), consisting of a single integrate-and-fire model receiving excitatory inputs from 1000 synapses (with exponential conductances) with spike-timing-dependent plasticity (see Figure 2A). Each

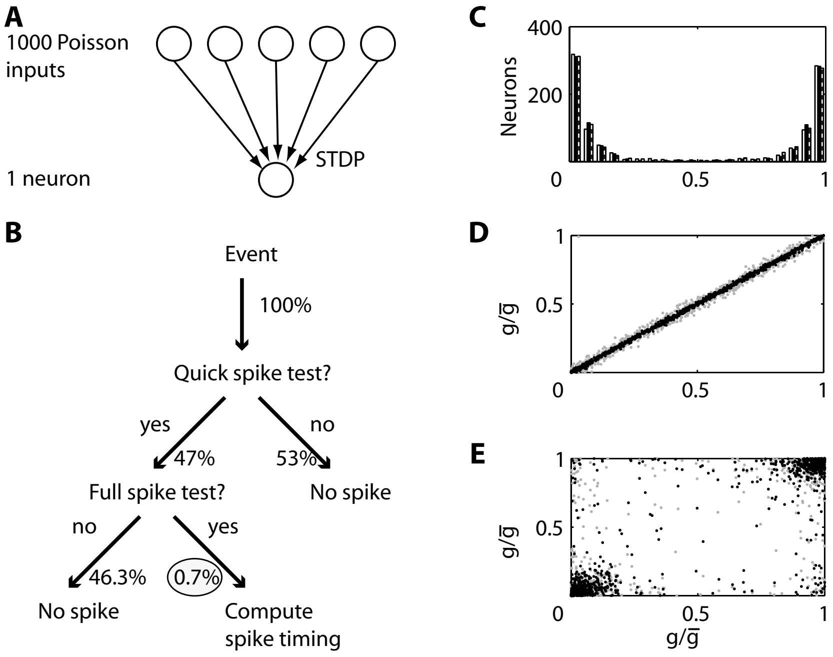

Figure 2: Simulation of an integrate-and-fire model with spike-time-dependent plasticity. (A) Architecture of the model (see details in the text). (B) Statistics of the outcome of the spike test (cf. Figure 1B). (C) Distribution of the weights after 30 minutes of simulation. White: without precalculated tables; black: with tables (precision 10^{-4}); stripes: with the original algorithm but one initial weight is flipped (see the text). (D) Black: weights obtained after 30 seconds with precalculated tables versus weights without tables. Gray: weights obtained after 30 seconds without tables and one initial weight flipped versus weights without tables and weights. (E) As in *D* but after 30 minutes.

synaptic input consists of spike trains modeled as Poisson processes with constant rate 10 Hz. The parameters of the integrate-and-fire model are $V_0 = -74$ mV, $V_t = -54$ mV, $V_r = -60$ mV, and $\tau = 20$ ms. Synaptic conductances are initially distributed uniformly in $[0, \overline{g}]$, with $\overline{g} = 0.015$ (in units of the leak conductance). The synaptic time constant is $\tau_s = 5$ ms, and the reversal potential is $E_s = E^+ = 0$ mV. Conductances evolve slowly with spike-time-dependent plasticity as described in Song and Abbott (2001): a pair of presynaptic and postsynaptic spikes occurring at times t_{pre} and t_{post} triggers a synaptic change equal to

$$\begin{aligned} &A^+ \exp(-(t_{\text{post}} - t_{\text{pre}})/\tau^+) \quad \text{if} \quad t_{\text{pre}} < t_{\text{post}} \\ -&A^- \exp(-(t_{\text{pre}} - t_{\text{post}}/\tau^-) \quad \text{if} \quad t_{\text{post}} < t_{\text{pre}} \end{aligned}$$

with $\tau^+ = \tau^- = 20$ ms, $A^+ = 0.005 \times \overline{g}$, and $A^- = A^+ \times 1.05$. All synaptic changes combine linearly, but synaptic conductances must lie in the interval $[0, \overline{g}]$. This learning rule has been shown to produce synaptic competition and stabilization of firing rates (Song & Abbott, 2001; Kempter, Gerstner, & van Hemmen, 2001; Cateau & Fukai, 2003). We ran the model for 30 minutes of biological time (which took about 10 minutes of CPU time).

4.1.2 Profiling. The heaviest operation is the computation of spike timing, which is about three to four times slower than updating the membrane potential. We suggested earlier that this operation should be executed only rarely, because most of the time, the neuron is in a state in which it cannot spike without receiving other excitatory events. Every time a spike is received, the time of the next (outgoing) spike is recalculated, but only if the spike test is positive (otherwise there is no spike; the algorithm is described in Figure 1B). We analyzed the outcome of the spike test (see Figure 2B) for the model described in this section. The neuron received about 18 million spikes, each inducing one spike test. It failed the spike test in 99.3% of the cases, meaning that spike timing was computed only 126,000 times. Thus, computing the spike times was a negligible component of the overall simulation time. Besides, in half of the cases, only the quick spike test (see Figure 1B, top) was executed.

4.1.3 Accuracy. After 30 minutes (biological time), the distribution of the weights was bimodal and concentrated near the bounds of the limiting interval $[0, \overline{g}]$, as expected (see Figure 2C). The distribution was almost unchanged if precalculated tables (precision 10^{-4}) were used instead of the full expression in formula 3.5. Individually, however, the weights were slightly different after 30 seconds (Pearson correlation $c = 0.9996$; see Figure 2D), and significantly different after 30 minutes ($c = 0.88$; see Figure 2E). About 8% of the weights were flipped from one side to the other side of the bimodal distribution. However, 30 minutes is a very long time from the point of view of the neuron: it corresponds to 90,000 membrane time constants and 18 million received spikes, so the observed discrepancy might be due only to the fact that the model, seen as a dynamical system, is chaotic or at least unstable.

In order to test this hypothesis, we tested the effect of changing the initial conditions very slightly in the original algorithm (without tables). We flipped one of the initial weights from 0 to $\overline{g}$ and ran the simulation again with the same seed for the random number generator, so that the timings of all incoming spikes are unchanged. It turns out that after 30 seconds, the weights differ from the original ones even more than the weights computed previously using tables (Pearson correlation $c = 0.9912$ versus $c = 0.9996$; see Figure 2D). After 30 minutes, the weights differed significantly from the original ones, in the same way as with tables (Pearson correlation $c = 0.84$ versus $c = 0.88$; see Figure 2E), but the distribution of the weights remained almost unchanged (see Figure 2C). Interestingly, after 30 minutes, the correlation between the weights obtained with the original algorithm and with precalculated tables was unaffected by the precision of the tables ($c = 0.87 - 0.88$ for precisions 10^{-3}, 10^{-4}, and 10^{-5}). Moreover, the correlation between weights obtained with different precisions for the tables was equivalent to the correlation with the weights obtained with the original algorithm ($c = 0.85$). We conclude that the long-term outcome of the simulation is very sensitive to initial conditions, and increasing the precision of the tables does not seem to make the weights converge. This indicates that the system is chaotic, which implies that only statistical measures are meaningful. We note that the distribution of weights is very well reproduced when using precalculated tables, and in the initial phase of the simulation (at least the first 30 seconds), the individual weights follow the original ones more faithfully than if just one initial weight (out of 1000) is modified.

4.2 Random network

4.2.1 The Model. We simulated a network of excitatory and inhibitory neurons receiving external excitatory inputs and connected randomly (see Figure 3A), as described in Brunel (2000), but with synaptic conductances instead of currents. Neuron models had the same parameter values as for the spike-timing-dependent plasticity (STDP) model, but there were two types of synapses: excitatory (reversal potential $E^+ = 0$ mV and weight $w^+ = 0.12$ in units of the leak conductance) and inhibitory ($E^- = -80$ mV, $w^- = 1.2$). Transmission delays were included in the model; they were picked at random from a uniform distribution in the interval 0.1–1 ms. Each neuron received excitatory spike trains, modeled as Poisson processes with rate 500 Hz.

We ran simulations lasting 1 second with eight different configurations for the numbers of excitatory neurons (4000–10,000) and inhibitory neurons (1000–3000) and the connection probability (0.02–0.1). The number of synapses per neuron ranged from 150 to 520 (not including external synapses); the average firing rates ranged from 2 Hz to 30 Hz. A sample of the spike trains produced is shown in Figure 3C.

4.2.2 Profiling. As for the STDP model, we analyzed the outcome of the spike test (see Figure 3B). On average, spike timing had to be calculated for only 4% of the events. Thus, again, the calculation of spike timing (the heaviest operation) has a small influence on the computational complexity of the simulation. Besides, for 86% of the cases, only the quick spike test was executed. Thus, most of the simulation time was spent in updating the state variables and handling event queues.

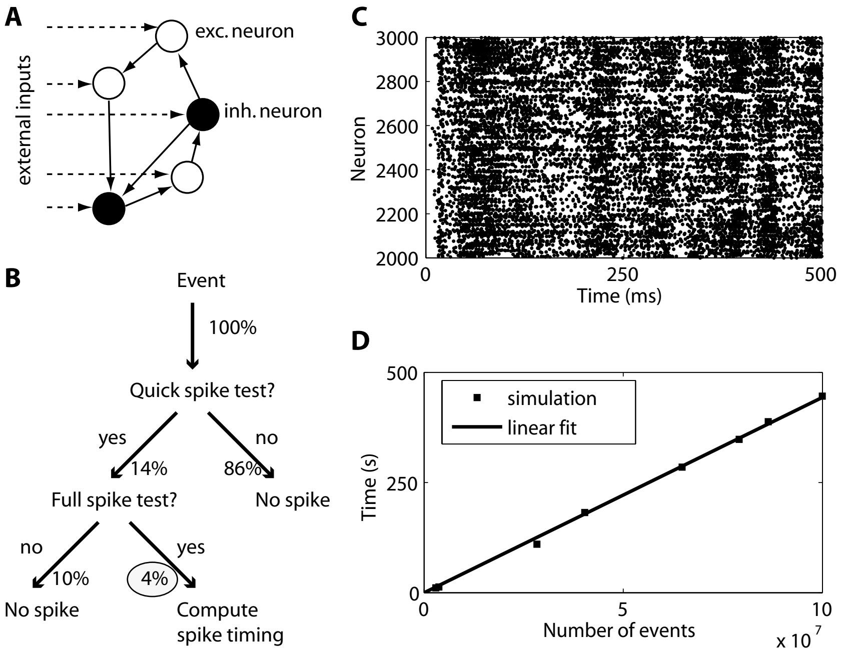

Figure 3: Simulation of random networks of excitatory and inhibitory neurons. (A) Architecture of the model (see details in the text). (B) Statistics of the outcome of the spike test (cf. Figure 1B). (C) Sample of the spike trains produced by the network in 500 ms. (D) Simulation time as a function of the total number of events (spike transmissions) for eight different configurations of the network.

4.2.3 Scaling of the Simulation Time. We wanted to check that the simulation time scales with the number of events handled, that is, is proportional to $F \times N \times p \times T$ (cf. section 2.2), where F is the average firing rate, N is the number of neurons, p is the number of synapses per neuron, and T is the time of the simulation. Figure 3D shows that for the eight configurations we simulated, the relationship between the simulation time and the total number of events is very well fitted by a line, as expected.

5 Discussion

We have shown that exact simulation of spiking neural networks, in an event-driven way, is in fact not restricted to simple pulse-coupled integrate-and-fire models. We have presented a method to simulate integrate-and-fire models with exponential synaptic conductances, which is a popular model in computational neuroscience. The simulation time scales linearly with the total number of events (number of spike transmissions), as for algorithms based on integration methods (Euler, Runge-Kutta) or, in fact, for optimal algorithms. For medium-sized networks, exact simulation may even be faster than approximation methods (this is, however, implementation dependent).

It may be argued that what we call "exact" simulation means only "more precise than integration methods," because the calculations involve series for which we consider only a finite number of terms and spike timings are calculated approximately by the Newton-Raphson method. In principle, it is true that calculations are not exact: after all, in any case, they cannot be more precise than the floating-point representation in the machine. However, it is relevant to distinguish these so-called exact methods from approximation methods with time steps for two essential reasons:

i. The error done in neglecting the tail of a series is not of the same order as the error done with Euler or Runge-Kutta methods. The former decreases exponentially with the number of iterations, while the latter decreases only polynomially. Therefore, the level of precision that can be reached with the former method is extremely high—incomparably higher than with integration methods. Such high-precision approximations are done routinely when using exponential or logarithmic functions in computer programs. The Newton-Raphson method that we use to calculate spike times also converges exponentially.
ii. In approximation methods, other types of errors are induced by the fact that spike times are constrained to the boundaries of time steps. For example, the order of spikes within one time step is lost; these spikes are glued together. This might cause problems when investigating synchronization properties or spike-timing-dependent plasticity. These errors do not arise when using exact, event-driven methods.

One important difficulty with event-driven simulations is to introduce noise in the models. It has been shown in vitro that the response of cortical neurons to realistic currents injected at the soma is reproducible (Mainen & Sejnowski, 1995); therefore, a large part of the noise would come from synapses—from either the input spike trains or transmission failures. Therefore, one reasonable way to introduce noise in an event-driven simulation is to add random input spikes to the neuron models (as in Song & Abbott, 2001) or to introduce random failures when inserting a spike in an event queue.

Finally, an important limitation of the algorithm we have presented is that the time constants of excitatory and inhibitory synapses must be identical. We acknowledge that this is restrictive, but it still makes a significant progress in the biophysical realism of exactly simulable models. We were able to find analytical expressions for the membrane potential and for the spike test because the model could be reduced to a differential system with only two variables. If the time constants of excitatory and inhibitory synapses are different, this is no longer possible. We do not know if this is definitely untractable or if other tricks can be found. One possible track of investigation might be to use series expansion of the solutions of the differential equations. Finally, we mention two other directions in which efforts should be made to extend the range of models that can be simulated exactly: (1) integrate-and-fire models with soft threshold (e.g., quadratic, Ermentrout & Kopell, 1986; and exponential, Fourcaud-Trocme, Hansel, van Vreeswijk, & Brunel, 2003), which are more realistic than the leaky integrate-and-fire model, and (2) two-dimensional integrate-and-fire models, which can account for adaptation (Liu & Wang, 2001) and resonance (Richardson, Brunel, & Hakim, 2003).

Acknowledgments

I thank Olivier Rochel and Michelle Rudolph for fruitful discussions. This work was partially supported by the EC IP project FP6-015879, FACETS.

References

Abbott, L. F., & Nelson, S. B. (2000). Synaptic plasticity: Taming the beast. *Nat. Neurosci., 3* (Suppl), 1178.

Brette, R., & Guigon, E. (2003). Reliability of spike timing is a general property of spiking model neurons. *Neural Comput., 15*(2), 279–308.

Brown, R. (1988). Calendar queues: A fast 0(1) priority queue implementation for the simulation event set problem. *J. Commun. ACM, 31*(10), 1220–1227.

Brunel, N. (2000). Dynamics of sparsely connected networks of excitatory and inhibitory spiking neurons. *J. Comput. Neurosci., 8*(3), 183–208.

Cateau, H., & Fukai, T. (2003). A stochastic method to predict the consequence of arbitrary forms of spike-timing-dependent plasticity. *Neural Comput., 15*(3), 597–620.

Claverol, E., Brown, A., & Chad, J. (2002). Discrete simulation of large aggregates of neurons. *Neurocomputing, 47*, 277–297.

Connolly, C., Marian, I., & Reilly, R. (2003, Aug. 28–30). *Approaches to efficient simulation with spiking neural networks.* Paper presented at the Eighth Neural Computation and Psychology Workshop, University of Kent, U.K.

Delorme, A., & Thorpe, S. J. (2003). Spikenet: An event-driven simulation package for modelling large networks of spiking neurons. *Network, 14*(4), 613–627.

Destexhe, A., Mainen, Z., & Sejnowski, T. (1994). An efficient method for computing synaptic conductances based on a kinetic model of receptor binding. *Neural Comput., 6*(1), 14–18.

Destexhe, A., Rudolph, M., Fellous, J. M., & Sejnowski, T. J. (2001). Fluctuating synaptic conductances recreate in vivo–like activity in neocortical neurons. *Neuroscience, 107*(1), 13–24.

Ermentrout, B., & Kopell, N. (1986). Parabolic bursting in an excitable system coupled with a slow oscillation. *Siam J. Appl. Math., 46*(2), 233–253.

Fourcaud-Trocme, N., Hansel, D., van Vreeswijk, C., & Brunel, N. (2003). How spike generation mechanisms determine the neuronal response to fluctuating inputs. *J. Neurosci., 23*(37), 11628–11640.

Gerstner, W., & Kistler, W. M. (2002). *Spiking neuron models*. Cambridge: Cambridge University Press.

Grassmann, C., & Anlauf, J. (1998). Distributed, event driven simulation of spiking neural networks. *Proceedings of the International ICSC/IFAC Symposium on Neural Computation (NC'98)* (pp. 100–105). Canada: ICSC Academic Press.

Hansel, D., Mato, G., Meunier, C., & Neltner, L. (1998). On numerical simulations of integrate-and-fire neural networks. *Neural Comput., 10*(2), 467–483.

Kempter, R., Gerstner, W., & van Hemmen, J. L. (2001). Intrinsic stabilization of output rates by spike-based Hebbian learning. *Neural Comput., 13*(12), 2709–2741.

Knight, B. W. (1972). Dynamics of encoding in a population of neurons. *J. Gen. Physiol., 59*(6), 734–766.

Lapicque, L. (1907). Recherches quantitatives sur l'excitation électrique des nerfs traitée comme une polarisation. *J. Physiol. Pathol. Gen., 9*, 620–635.

Lee, G., & Farhat, N. H. (2001). The double queue method: A numerical method for integrate-and-fire neuron networks. *Neural Netw., 14*(6–7), 921–932.

Liu, Y. H., & Wang, X. J. (2001). Spike-frequency adaptation of a generalized leaky integrate-and-fire model neuron. *J. Comput. Neurosci., 10*(1), 25–45.

Lytton, W. W. (1996). Optimizing synaptic conductance calculation for network simulations. *Neural Comput., 8*(3), 501–509.

Lytton, W. W., & Hines, M. L. (2005). Independent variable time-step integration of individual neurons for network simulations. *Neural Comp., 17*(4), 903–921.

Mainen, Z., & Sejnowski, T. (1995). Reliability of spike timing in neocortical neurons. *Science, 268*, 1503–1506.

Makino, T. (2003). A discrete-event neural network simulator for general neuron models. *Neural Comput. and Applic., 11*, 210–223.

Marian, I., Reilly, R., & Mackey, D. (2002). Efficient event-driven simulation of spiking neural networks. In *Proceedings of the 3rd WSEAS International Conference on Neural Networks and Applications*. Interlaken, Switzerland.

Mattia, M., & Del Giudice, P. (2000). Efficient event-driven simulation of large networks of spiking neurons and dynamical synapses. *Neural Comput., 12*(10), 2305–2329.

Morrison, A., Mehring, C., Geisel, T., Aertsen, A., & Diesmann, M. (2005). Advancing the boundaries of high connectivity network simulation with distributed computing. *Neural Comput., 17*, 1776–1801.

Olshausen, B. A., & Field, D. J. (2005). How close are we to understanding V1? *Neural Comput., 17*(8), 1665–1699.

Press, W. H., Flannery, B. P., Teukolsky, S. A., & Vetterling, W. T. (1993). *Numerical recipes in C: The art of scientific computing*. Cambridge: Cambridge University Press.

Richardson, M. J., Brunel, N., & Hakim, V. (2003). From subthreshold to firing-rate resonance. *J. Neurophysiol., 89*(5), 2538–2554.

Rochel, O., & Martinez, D. (2003). An event-driven framework for the simulation of networks of spiking neurons. In *Proc. 11th European Symposium on Artificial Neural Networks* (pp. 295–300). Bruges, Belgium.

Shelley, M. J., & Tao, L. (2001). Efficient and accurate time-stepping schemes for integrate-and-fire neuronal networks. *J. Comput. Neurosci., 11*(2), 111–119.

Song, S., & Abbott, L. (2001). Cortical development and remapping through spike timing–dependent plasticity. *Neuron, 32*, 339–350.

Song, S., Miller, K. D., & Abbott, L. F. (2000). Competitive Hebbian learning through spike-timing-dependent synaptic plasticity. *Nature Neurosci., 3*(9), 919–926.

Stopfer, M., Bhagavan, S., Smith, B. H., & Laurent, G. (1997). Impaired odour discrimination on desynchronization of odour-encoding neural assemblies. *Nature, 390*(6655), 70–74.

Ziegler, B., Praehofer, H., & Kim, T. (2000). *Theory of modeling and simulation. Second edition. Integrating discrete event and continuous complex dynamic systems*. Orlando, FL: Academic Press.

Received July 15, 2005; accepted January 18, 2006.