LETTER Communicated by Lloyd Watts

Efficient Event-Driven Simulation of Large Networks of Spiking Neurons and Dynamical Synapses

Maurizio Mattia
Paolo Del Giudice
Physics Laboratory, Istituto Superiore di Sanità, 00161 Roma, Italy

A simulation procedure is described for making feasible large-scale simulations of recurrent neural networks of spiking neurons and plastic synapses. The procedure is applicable if the dynamic variables of both neurons and synapses evolve deterministically between any two successive spikes. Spikes introduce jumps in these variables, and since spike trains are typically noisy, spikes introduce stochasticity into both dynamics. Since all events in the simulation are guided by the arrival of spikes, at neurons or synapses, we name this procedure event-driven.

The procedure is described in detail, and its logic and performance are compared with conventional (synchronous) simulations. The main impact of the new approach is a drastic reduction of the computational load incurred upon introduction of dynamic synaptic efficacies, which vary organically as a function of the activities of the pre- and postsynaptic neurons. In fact, the computational load per neuron in the presence of the synaptic dynamics grows linearly with the number of neurons and is only about 6% more than the load with fixed synapses. Even the latter is handled quite efficiently by the algorithm.

We illustrate the operation of the algorithm in a specific case with integrate-and-fire neurons and specific spike-driven synaptic dynamics. Both dynamical elements have been found to be naturally implementable in VLSI. This network is simulated to show the effects on the synaptic structure of the presentation of stimuli, as well as the stability of the generated matrix to the neural activity it induces.

1 Introduction

Computer simulations of model neural networks with feedback play an increasingly prominent role in the effort to model brain functions. Besides complementing theory when it cannot fully describe the richness of the phenomenology embedded in the mathematical model, simulations aspire to be predictive and realistic enough to serve as a partial guide in planning the experiments and helping in the interpretation of real data.

Although there is wide agreement on these statements, network simulations lack a crucial ingredient: synaptic dynamics. Learning has been

incorporated in the modeling of (recurrent) feedback neural networks thus far as a static prescription for synaptic efficacies, according to some chosen "learning rule." In this way, a long and important series of results have been obtained, which have made these models candidates for an account of the retrieval and maintenance of "learned" internal representations of a stimulus in tasks implying a working memory (see, e.g., Amit, 1995; Amit & Brunel, 1997a, 1997b.).

But learning is guided by neural activities, which in turn are governed by the acquired synapses. Hence, for these models to make a leap toward a realistic description of learning, as it results from the dynamical interaction with the environment, a mechanism has to be incorporated in the model to describe the evolution of synaptic variables. Such an evolution will be driven by neural activities, which are driven in turn by afferent stimuli. With dynamical synapses, an incoming stimulus will not only make learned information pop up from the memory store and be put in an active state, but will also contribute to expanding the store.

Moreover, if synaptic dynamics is a function of neural activities, then the question arises of the stability of any acquired synaptic stock and any activity distribution. This is true for both underlying levels of spontaneous activity and computational expressions by selective activity distributions. In other words, any neural activity distribution may produce synaptic changes and consequently start drifting away from what was considered spontaneous rates or a computationally relevant distribution. The question of the stability, for any computational paradigm, must be considered central, in the sense that both the selectivity of the computational outcome and its separability from noise (spontaneous activity) are put into question by synaptic dynamics. Thus, simulating joint dynamics is doubly necessary.

There are serious problems in carrying out such a program. Semirealistic networks must contain a large number of cells to reproduce realistic spike dynamics. A typical cortical module contains on the order of 10^5 cells. Even with fixed synapses and simple integrate-and-fire (IF) neurons, a standard workstation will run a typical simulation of a network of this size with a very high ratio between the CPU time and the network's "biological" time.

One might hope that a suitably rescaled network of $\mathcal{O}(10^4)$ neurons would be feasible and still capture the most relevant aspects of collective behavior. But the number of synapses is a factor of 10^3–10^4 greater than that of the neurons. If synaptic dynamics is to be an organic part of the simulation, then the number of dynamical variables threatens to increase by a corresponding factor, making the simulation impracticable.

To this one should add the expectation that synaptic dynamics is intrinsically slower than neural dynamics, which implies that in order to observe the effects of synaptic dynamics on the neural activities, longer simulations (in neural time) are required, just when everything is about to slow down because of the number of dynamical variables. It is therefore important to devise and explore simulation strategies that could make fea-

sible large-scale simulations with double, coupled dynamics of neurons and synapses.

We have found some attempts in this direction. An event-driven approach, similar in concept to the approach presented here, was first proposed by Watts (1994). It is rather general and flexible but not very suitable for large networks with extensive randomness because a specific "event" is generated and handled for every modification of every element in the network. For example, a spike that affects cN neurons will entail the management of cN events, with the associated computational load of the corresponding time ordering. These features of the simulation are consistent with its intended purpose of simulating small networks to be implemented in silicon and envisage the possibility of simulating complex neural elements.

GENESIS (Bower & Beeman, 1998), a widely used, general-purpose simulation software, can serve as a basis for a wide class of simulation strategies. It has been used mostly in simulations of relatively small networks of complex, multicompartment model neurons. In computational terms, it is essentially a synchronous simulation, with a wide range of choices for the numerical integration method; a recently proposed evolution of the original package (PGENESIS, parallel GENESIS) implements a distributed version of clocked, synchronous simulations over several processors. It may be considered partially asynchronous as far as the management of the communication between modules is concerned.[1]

SpikeNET (Delorme, Gautrais, van Rullen, & Thorpe, 1999) is a simulator with the stated purpose of simulating large networks of IF neurons. In it, the single neuron dynamics is fully synchronous, with a surprisingly high value for the time step ($dt = 1$ ms for neurons spiking at 1 Hz); the network operates in a feedforward fashion and is constrained to be sparsely connected: each layer propagates to the next one a list of neurons that spiked in the last dt (each spike affecting about 50 neurons in a network of 400,000 neurons); finally, there is no ongoing synaptic dynamics (learning is effected in a supervised, noniterative fashion).

Here we describe a way of exploiting specific features of the dynamics of IF neurons and a class of dynamical synapses driven by spikes in order to achieve a gentle scaling of the computational complexity of the full simulation with the network size. In section 2 we summarize the general idea. In section 3 we set the context of the simulation, listing the main features of the networks we consider. In section 4 we provide a detailed description of the algorithm. In section 5 we define a specific context for a simulation whose results are sketched in section 6. In section 7 we describe the performances of the algorithm. In the appendix we illustrate possible extensions and improvements of the algorithm, some of which are the subject of work in

[1] See http://www.psc.edu/Packages/PGENESIS/.

progress. A summary and outlook are provided in section 8. A preliminary account of this work is given in Mattia, Del Giudice, and Amit (1998).

2 An Event-Driven Approach

The usual approach to numerical simulations of networks of IF neurons is based on the finite difference integration of the associated differential equations. Whatever method is used (such as Euler or Runge-Kutta), a time step Δt is chosen that serves as a clock, providing the timing for the synchronous update of the dynamical variables. Δt also sets a cutoff on the temporal resolution of the simulation and, therefore, on the ability to capture short-time dynamical transients (Hansel, Mato, Meurier, & Neltner, 1998). Because of the role played by Δt, we call these simulations *synchronous*.

To get a quantitative estimate of the scaling of the computational complexity of a synchronous simulation when synaptic dynamics is introduced, suppose there are N neurons in the network and that (1) the connectivity is c (a spike emitted by a neuron is communicated on average to cN neurons) and (2) neurons emit on average ν spikes per second. With a time step Δt seconds, one has on average $cN/\Delta t$ synaptic updates per second per neuron. On the other hand, Δt must be limited to prevent conflicts between arriving spikes: the average number of spikes per second received by a neuron is νcN, so that the average interval between two successive spikes' hitting a neuron is $1/\nu cN$; but in order not to miss dynamical effects possibly related to single spikes, the probability of having more than one spike received by a neuron in Δt should be negligible. This implies the bound $\Delta t \ll 1/\nu cN$. Therefore, the number of synaptic updates per second per neuron is $\gg \nu c^2 N^2$, which leads to an N^2 scaling of the computational complexity per neuron.

In the context of simulations with fixed synapses (and with several other simplifications), a few suggestions have been put forward (Hansel et al., 1998; Tsodyks, Mit'kov, & Sompolinsky, 1993), pointing to the development of more efficient synchronous algorithms. Margins for improvement, already at the purely neural level, are provided by the following observation. In realistic conditions, the neurons emit at low rates, with a high variability in the time intervals between successive spikes. Because of this, a synchronous algorithm will spend most of the time effecting the analog update of the depolarization of the IF neuron, whose evolution is deterministic between successive afferent spikes and could be exactly interpolated.

Assume also that the synaptic dynamics is spike driven and is deterministic between spikes. This is a rather generic assumption, leaving open the particular type of dynamics. Specific choices will be discussed. In that case, the events (spikes) whose occurrence determines changes in the synaptic state will be much sparser in time, because a synapse sees only two neurons, while a neuron contacts cN synapses. The finely grained deterministic update, at the root of the N^2 scaling, could be eliminated if most of the computation is concentrated in the asynchronous determination of the effects of

the single spikes on neural as well as on synaptic variables, letting the deterministic interspike evolution of both be exactly interpolated at the same instances and become computationally negligible.

Such an event-driven approach is proposed in this article. Its computational complexity per neuron is linear in N. In fact, each spike emitted by a generic neuron affects on average cN synapses, so an average number of νcN synapses are affected per second by that neuron, and this gives a linear dependence on N. In this approach, there is no intrinsic temporal cutoff, and the simulation is an exact solution of the evolution equations of the system, under fairly general conditions to be specified below. The simulation therefore reproduces transients on arbitrarily short timescales.

This strategy is very effective, but its implementation is complicated by the various sources of inhomogeneities and randomness in the network, as we explain.

3 General Features of the Simulated Network

The simulated network to be considered has the following typical ingredients:

- Neurons are of the IF type. The membrane depolarization undergoes a deterministic evolution between two subsequent incoming (excitatory or inhibitory) spikes; spikes are instantaneous events (zero length in time) generated when the neuron's depolarization surpasses a threshold. The interpolating dynamics can be generic.
- In addition to recurrent spikes, all neurons in the network receive spikes from outside. Low-frequency, nonselective, external spike trains implement the background activity in the area surrounding the local module; higher-rate, selective external spike trains code for afferent stimuli. The resulting external current is assumed to be random in both cases. Typically the stochastic current will be a Poissonian train of spikes since external neurons emit consecutive spikes at random, independent instants with exponentially distributed interspike intervals (ISI), and spike emission by different external neurons is uncorrelated. Current from outside is then a superposition of uncorrelated stochastic Poisson processes. At the end of section 4.2 we describe how its generation is implemented in the simulation.
- In the absence of a paradigm for a model of spike-driven synaptic dynamics, such as the IF model for the neuron, the synaptic dynamics will be assumed to be governed by the (instantaneous) presynaptic spikes and the depolarization of the postsynaptic neuron, following a recently proposed model (Annunziato, Badoni, Fusi, & Salamon, 1998; Annunziato & Fusi, 1998; Fusi, Annunziato, Badoni, Salamon, & Amit, 2000). The presynaptic spikes serve as triggers for synaptic changes,

and the postsynaptic potential determines whether the change is potentiating or depressing. Long-term memory is maintained by two discrete values of the synaptic efficacy. In this model, synaptic changes have a Hebbian character and are stochastic because of the random nature of the spikes driving them. There is some empirical support (Petersen, Malenka, Nicoll, & Hopfield, 1998) and strong logical motivations (Amit & Fusi, 1994) for considering synapses with a discrete, small set of stable values for their efficacies. Theory suggests (Amit & Fusi, 1994) that stochastic dynamics provide an effective mechanism for implementing learning with discrete synapses.

- Both architectural and dynamical sources of randomness are present in the network. The first, static (or "quenched") noise is associated with the random pattern of connections among the neurons in the network and the random distribution of delays in the transmission of recurrent spikes. The latter is due to external spikes' hitting the neurons in the network at random instants of time. Moreover, the recurrent spikes traveling in such a network exhibit a high degree of randomness in their times of emission. The network draws from this distributed reservoir of randomness in order to implement stochastic learning.
- The recurrent network is assumed to include several interacting populations of neurons (excitatory and inhibitory neurons in the simplest case).

Both the neural and the synaptic dynamics merge stochastic and deterministic components, the first being associated with the random nature of the sequences of spikes (which act in both cases as trigger events for the dynamics) and the second expressing the evolution of the synaptic or neural state variables in the time interval between two successive afferent spikes. The algorithm to be described exploits the fact that the state variables of neurons and synapses evolve most of the time deterministically. They deviate only upon the stochastic arrival of spikes.

The procedure will be described for both a monotonic decay of the state variables between two successive spikes (for neurons, this is the case for the usual leaky IF dynamics and also for the "linear" IF neuron; see Fusi & Mattia, 1999), as well as for a generic deterministic evolution between two spikes.

4 The Algorithm

The core of the algorithm is the management of the temporal hierarchy of the spikes that are generated in the network and those that arrive from outside. We first describe the coding of the elementary events and the data structures, found convenient in reducing the computational load due to the

necessary ordering of events in time. Then we go into detail on the working of the algorithm.

4.1 Main Elements and Data Structures. Each recurrent, instantaneous spike (*event*) emitted by an IF neuron of the simulated network is represented by the pair (i, t) where i is the emitting neuron and t is the emission time of the spike.

We assume a discrete set of D ordered delays $d_0 < d_1 \cdots < d_{D-1}$ for spike transmission. Synapses are organized in matrix-structured layers, each layer corresponding to one value of delay. The generic column i of the synaptic matrix in layer l represents all those synapses on the axon of neuron i that transmit a spike from neuron i to target neurons, with delay d_l. Target neurons are identified by the row index in each layer of the synaptic matrix. Each element in the column contains the synaptic (dynamic) state variables: an analog (internal) variable that evolves continuously, determining in turn the evolution of a discrete variable that represents long-term memory and acts as the synaptic efficacy, mediating the effects of presynaptic spikes. The union of all layers represents the complete synaptic matrix.

The synaptic structure is compressed, taking advantage of the sparse nature of the synaptic connections. Many vacancies would be left in a straightforward representation of the synaptic matrix; the layers are even sparser. The compressed version of the synaptic matrix in which only nonzero elements are stored is such that the generic element in one column of a given layer contains the address of the next neuron on the same portion of the axon (in terms of the difference of their indices). This coding does not imply additional complexity in scanning the axon.

The events (i, t) representing recurrent spikes are directed to the synaptic layers for neural and synaptic updates through a set of queues, one for each layer (i.e., for each delay). Each element in the lth queue is the pair $(i, t + d_l)$ (d_l is the value of the lth delay), which is the spike generated by neuron i at time t. This event will trigger neural updates at time $t + d_l$ using efficacies in layer l and synaptic updates in the same layer. Each queue is of the first-in-first-out (FIFO) type: events are ordered in time, with the oldest element at the top of the list.

External spikes from outside the network are modeled by stochastic trains of instantaneous events impinging on the cells in the network. Each external spike is coded by a pair (j, t), where j is the target neuron in the network and t is the time when the external spike will be received. The external stochastic processes viewed by different neurons are assumed to be uncorrelated and such that the linear superposition of several external streams of spikes retains the same statistical distribution of the single train; this is the case for trains of spikes with Poissonian or gaussian statistics.

These choices allow significant simplification in the management of the external spikes. One can generate a single train of spikes, for example, with a Poisson distribution, store successive an external spikes in a single reg-

ister serving as a buffer, and choose at random the target neuron for the current spike in the register, such that each neuron in the network receives an external Poissonian spike train of prescribed frequency (see the end of section 4.2 for details).

The major advantage brought about by this simplification will be clear in the next section.

4.2 Life Cycle of a Spike. The algorithm is explained with reference to Figure 1.

The simulation is started by igniting activity in the network. This is done by either prefilling the queues of events or starting with empty queues having the network driven by incoming external spikes. The time unit is provided by one of the parameters with dimensions of time that enter the dynamics: the absolute refractory period, the frequency of external (Poisson) events and the decay constant of the neuron's depolarization.

Suppose a spike is emitted by neuron k at time T (A in Figure 1). The event (k, T) enters a first queue Q_0 associated with the first synaptic layer (with minimal delay d_0), and its time label is updated to $T + d_0$, which is when the spike will affect its target neurons (B). The event waits to be handled until its temporal label becomes the oldest in the queue (it is in the first position in the queue) (B). When all previously emitted spikes have affected all their postsynaptic targets with delay d_0, the event reaches the first position in Q_0, and at this moment it becomes the first in line to act, among spikes that will act with delay d_0.

If, upon sorting with the top elements of the other queues (F), it turns out to be the oldest event around, it starts affecting the synapses in column k of the first layer (C), and updating the state of each postsynaptic neuron, identified by the row index of the synapse.

Target neurons for the spike are sequentially addressed, and their depolarization is updated:

1. The time at which the last spike reached the target neuron is known. Using the time difference to the arrival of the present spike, the value of the neuron's depolarization is given by its deterministic evolution.
2. The postsynaptic contribution of the spike to the depolarization is the corresponding synaptic efficacy. It is added as a discrete value.
3. The new value of the depolarization is compared with the threshold for emission of a spike.
4. If spike emission occurs, the new event enters the queue Q_0 with time label $T + d_0 + d_0$.
5. The synaptic efficacies are updated by the combination of their deterministic evolution in the interval since the arrival of the last spike and the spike-driven contribution.

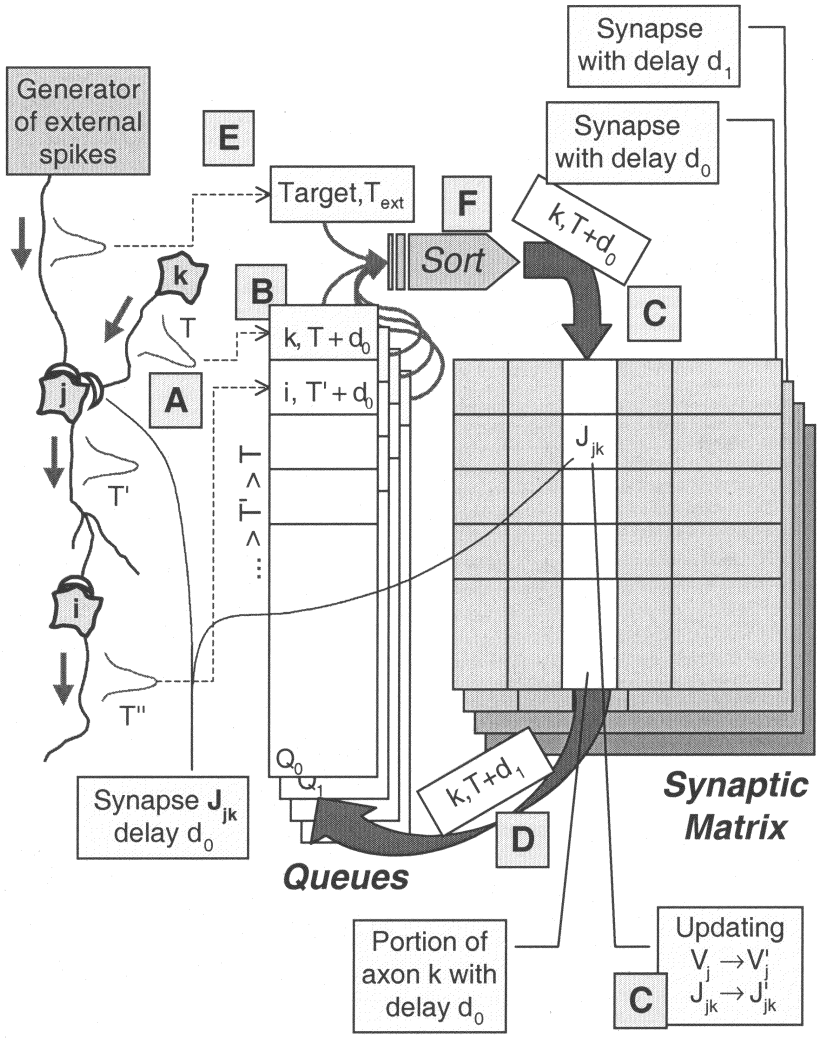

Figure 1: Schematic illustration of the algorithm. (From the left): A portion of the network, three neurons, two of which emit spikes at different times; the main data structures used in the simulation: the queues of events associated with each layer, represented as a stack of arrays; then the synaptic matrix—a superposition of D grids, one for each synaptic delay (the layers), each element containing the synaptic variables, including the discrete efficacy. The arrows illustrate the main successive steps taken in the management of an event, and the letters mark the main phases in the lifetime of a spike. See the text for details.

Here, for simplicity, we have taken the time at which a spike updates the depolarization of a neuron to be the same as that at which it updates the synaptic variable (see section 5 for details of the chosen synaptic dynamics.) This limitation can be relaxed (see the appendix).

Once all the postsynaptic updates associated with delay d_0 are completed, the event is appended with time label $T + d_1$ ($d_1 > d_0$) to the bottom of the next queue Q_1, attached to the layer with delay d_1, and it is handled as above (D).

Queues are filled in such a way that the time labels are automatically ordered inside each queue (the first element is guaranteed to be the oldest of its queue), without need of further sorting. However, in order to choose the oldest among all events, it is necessary to sort the first events in all the queues to be processed. With a small number of allowed discrete values for the delays, this is a negligible additional computational load.

We have assumed that the neuron's depolarization undergoes a monotonic decay between successive incoming spikes. This ensures that a spike can be generated only at the time a spike arrives (see the appendix for possible extensions).

The life cycle of an external spike is simpler, because it carries no delay and has a unique target, and it is very much simplified by the statistical assumptions we have made. In the simulation, a single random generator produces trains of external spikes with assigned frequencies. If the last external spike is received at time t_n by the network, the next one will arrive at time $t_{n+1} = t_n + \Delta t$, where Δt is randomly chosen from the exponential density function $p(\Delta t)$,

$$p(\Delta t) = \nu_{ext} c_{ext} N^2 e^{-\nu_{ext} c_{ext} N^2 \Delta t}, \tag{4.1}$$

where we assume that external neurons emit spikes at a mean rate ν_{ext} and the average number of external connections received by a neuron in the network $c_{ext}N$. The generation of the new event is completed by choosing at random from the recurrent network, using a uniform distribution, the target neuron for the external spike. In this way, each neuron sees incoming spikes with Poissonian statistics of mean rate $\nu_{ext} c_{ext} N$. Random selection of the receiver ensures that trains of external spikes afferent on different neurons are statistically independent. Each new external event is stored in a register (E), and its time label participates in the sorting process together with the top elements of the queues. As soon as the event in the register has been processed, it is replaced by a new one.

The corresponding computational load per second of simulation (in "biological time") is the generation of an average of $\nu_{ext} c_{ext} N$ external spikes per neuron, which is of $\mathcal{O}(N)$. The choice of independent Poisson trains of external spikes, which allows us to use a single random generator and a single register for all external spikes, is crucial in keeping low the load related to spike sorting (see section 4.3).

For a network composed of several populations of neurons (e.g., a population of inhibitory neurons and several populations of excitatory neurons, activated by different external stimuli), one spike generator with appropriate frequency and one register are required for each population. So sorting involves only $D + P$ elements, where P is the number of populations in the simulated network (different populations may receive in general external currents with different statistical properties), with an additional complexity of $\mathcal{O}(\log_2(D+P))$, which is negligible in the usual situation $D+P \ll N$ (see section 7 and Figure 6).

4.3 Remarks on Specific Features of the Algorithm.

4.3.1 Layered Structure. Coding the synaptic matrix in a single layer would imply, for each spike to be processed, scanning the whole axon as many times as there are delays in order to choose the appropriate neurons for update. Also, the spike just processed for one value of delay would have to be reinserted in the global pool of events to be processed (the single queue of length, say, m), implying an additional $\log_2(m)$ computational effort (insertion of one element in an ordered list), to be multiplied by the number of delays.

To estimate the typical value of m, we note the following. Each spike emitted in the network will travel along its axon for a time d_{D-1} (provided at least one target synapse exists for that neuron with maximal delay d_{D-1}). That spike will therefore be "active," waiting for all its effects to be computed by the simulation, for a time window d_{D-1} after its emission. In other words, at time t, only spikes emitted in a time span $(t - d_{D-1}, t)$ will coexist on the same axon. For reasonably low frequencies and a maximal delay of a few milliseconds, the average number of spikes traveling simultaneously on the same axon (νd_{D-1}) in the time window $(t - d_{D-1}, t)$ is very small. But when we consider the whole network, all neurons contribute a factor νd_{D-1} to the average number of active events in the supposed single queue at time t, and so $m = N\nu d_{D-1}$. For example, in a regime of spontaneous activity and in the absence of external spikes, a large network of $N = 10^5$ neurons emitting $\nu = 2$ spikes per second, with maximal delay $d_{D-1} = 3$ ms has an average of $N\nu d_{D-1} = 600$ active events at any given time t. So a "brute force" (single-layer) event-driven approach would have to manage a single queue with an additional computational load of $\mathcal{O}(D \log_2(d_{D-1}\nu N))$.

4.3.2 Generation and Handling of External Spikes. Having a separate generator of external spikes for each neuron in the network would imply a relatively small increase in memory allocation but a huge increase in the computational load due to the additional sorting required. Instead of having to sort for each spike D top elements of the queues, plus elements in the registers for the external spikes, $D + N$ elements would have to be sorted.

4.3.3 Simultaneous Event-Driven Management of "Fast" and "Slow" Dynamics. One might think of a synchronous algorithm based on the simultaneous use of two very different Δt for the neural and the synaptic evolution. But we are typically interested in slow synaptic dynamics, in which the (stochastic) changes in the synaptic state occur with small probabilities. This implies that the synaptic dynamics is governed by rare, fast bursts of spikes, and a big Δt would fail to reproduce correctly the most important regime. A big Δt for synapses would also conflict with the resolution with which the time of emission of the spikes is determined, constrained in turn by the neural Δt. The input signal to the synapse would therefore be affected by a quantization error, which could in principle propagate back to the neural dynamics and distort it (for example, in regimes of fast global oscillations).

5 A Specific Context: Linear IF Neuron and Spike-Driven Synapses

We adopt a linear IF neuron as an example. Such a neuron is suitable for VLSI implementation, and networks of such neurons retain most of the collective features of the leaky IF neuron (Fusi & Mattia, 1999). The choice of the neuronal model is not critical for the algorithm. The recurrent network includes N_E excitatory neurons (possibly including subpopulations activated by different external stimuli) and N_I inhibitory neurons. Neuronal depolarization integrates linearly the afferent stream of spikes, with a constant decay β between two subsequent incoming spikes. Each spike from neuron j contributes a jump in the depolarization V_i of receiving neuron i, equal to the synaptic efficacy J_{ij}. If V_i crosses a threshold θ, a spike is emitted, and V_i is reset to $H \in (0, \theta)$, where it is forced to remain for a time τ_{arp}, the absolute refractory period. In what follows, $H = 0$. A reflecting barrier at 0 forces V_i to stay at 0 whenever it is driven toward negative values, by either the constant decay or inhibitory spikes. This dynamics is illustrated in Figure 2.

The synapses connecting pairs of excitatory neurons are plastic; all the others are fixed. The dynamics of the plastic synapses is defined as in Annunziato et al. (1998), where an analytic treatment of the model is described, together with results from a VLSI implementation. (For another model of a spike-driven synaptic dynamics, see Häfliger, Mahowald, & Watts, 1997; Senn, Tsodyks, & Markram, 1997). It is a specific implementation of a stochastic, Hebbian learning dynamics. The synaptic efficacy J takes one of two values—J_0 (depressed) and J_1 (potentiated)—and the learning dynamics evolves as a sequence of random transitions between J_0 and J_1, driven by the pre- and postsynaptic activities. An internal synaptic (dimensionless) variable ("synaptic potential," V_J) describes the short-time analog response of the synapse to pre- and postsynaptic events and determines the transitions between the states J_0 and J_1, as follows (see Figure 3). V_J is confined to vary in the interval $[0, 1]$. Upon arrival of a presynaptic spike, V_J undergoes a positive (negative) jump of size a (b) if the postsynaptic

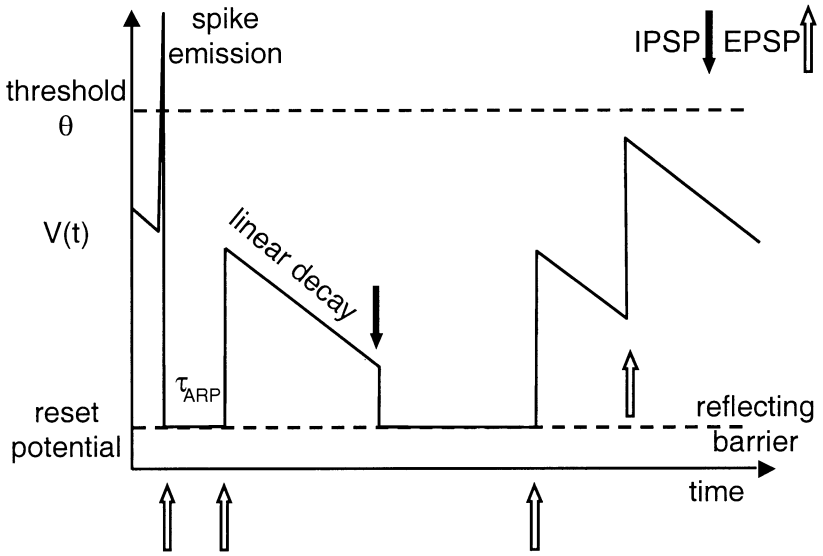

Figure 2: Illustration of the dynamics of the neural depolarization $V(t)$ (see the text). The evolution of $V(t)$ is governed by the equation $\dot{V}_i(t) = -\beta + I_i(t)$, where $I_i(t) = \sum_{j \neq i}^{1,N} J_{ij} \sum_k \delta(t - t_j^{(k)} - d_{ij})$ is the afferent current to neuron i, resulting from the sequences of spikes emitted by presynaptic neurons j at times $t_j^{(k)}$ and transmitted with delays d_{ij}, whose postsynaptic contribution to $V_i(t)$ is J_{ij}.

depolarization is found to be above (below) a threshold θ_V (not the spike emission threshold, $\theta > \theta_V$). The accessible interval for V_J is split in two parts by a synaptic threshold θ_J. The synaptic efficacy $J = J_0$ if $V_J < \theta_J$ and $J = J_1$ if $V_J > \theta_J$. Whenever a jump brings V_J above (below) θ_J, J undergoes a transition: $J_0 \rightarrow J_1$ (long-term potentiation, LTP) ($J_1 \rightarrow J_0$ (long-term depression, LTD). In the time between two successive presynaptic spikes, V_J is linearly driven toward 0 (1) if the last spike left it below (above) θ_J. The extreme values 0 and 1 act as reflecting barriers for V_J.

6 Synaptic Structuring: Expectations and Results

The synaptic dynamics we have described leads to the following qualitative expectations. First, since any change in the synaptic state is triggered by a presynaptic spike, synaptic modifications will be rare for low-frequency presynaptic neurons. Second, the synaptic potential V_J is attracted toward the barriers in 0 and 1, so in order to provoke a change in the synaptic efficacy J, the rate of trigger events has to be high enough to overcome the restoring force (that is, the time between two successive presynaptic spikes

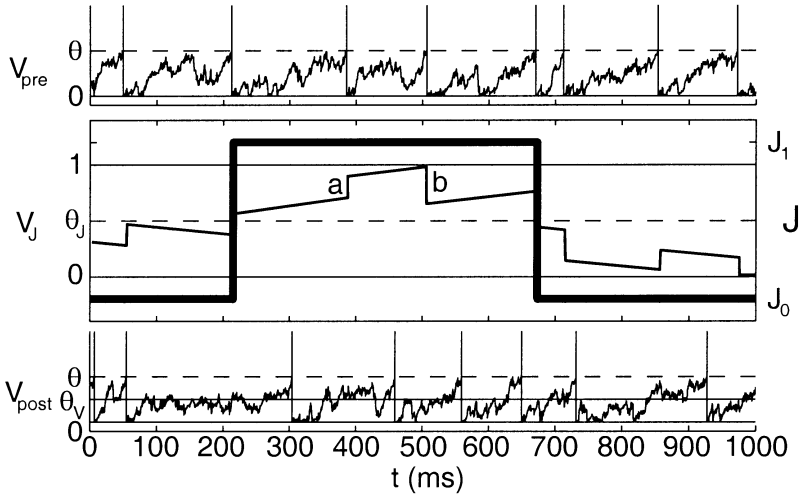

Figure 3: Illustration of the synaptic dynamics as driven by the activities of pre- and postsynaptic neurons. (Top) Depolarization and spikes of presynaptic neuron. (Bottom) Depolarization and spikes for postsynaptic neuron. (Middle) Induced time course of the two synaptic variables, the synaptic efficacy J (thick curve), and the synaptic potential V_J (thin curve). In the absence of presynaptic spikes, if $V_J(t) < \theta_J$, it drifts down; if $V_J(t) > \theta_J$, it drifts up. Presynaptic spikes trigger abrupt changes in V_J. Upon arrival of a presynaptic spike, V_J undergoes a positive jump of size a if the postsynaptic depolarization is above the threshold θ_V, and a negative jump b otherwise. As long as $V_J(t) < \theta_J$, the synaptic efficacy $J = J_0$, when $V_J(t) > \theta_J$, $J = J_1$. These represent long-term memory. The second presynaptic spike makes V_J cross the threshold θ_J (dashed line), and this provokes an instantaneous LTP transition of the synaptic efficacy $J = J_0 \to J = J_1$ (solid curve). The fifth presynaptic spike provokes a depression, LTD.

must be shorter than the time needed for the linear decay to compensate for the jumps in V_J). Third, given the statistics of presynaptic spikes, the probability distribution of the postsynaptic depolarization determines the relative frequency of up and down jumps in V_J, and therefore of potentiations (LTP) and depressions (LTD). A postsynaptic neuron emitting at a high rate will frequently have its depolarization near the threshold for spike emission θ, hence above θ_V, while a low-rate neuron spends most of its time near the reflecting barrier at 0, and V is likely to be below θ_V. In the case of interest, in which the transition probabilities for J are low, a synaptic transition can occur only when a fluctuation produces a burst of presynaptic spikes coherently pushing V_J up or down.

Expectations for the learning scenario due to the above mechanism are that the synaptic structure should be unaffected for low presynaptic activity,

regardless of the postsynaptic neural state, and high presynaptic activity will provoke potentiation (LTP) or depression (LTD), respectively, for high or low spike rates in the postsynaptic neuron.

An example of the time evolution of a recurrent network of linear IF neurons, subjected to the above synaptic dynamics, is a network of 1500 neurons (1200 excitatory, 300 inhibitory) connected by synapses with 10% connectivity ($\sim$ 144,000 plastic synapses). We choose this relatively "small" network to allow a faithful graphical representation of the synaptic matrix. Table 1 summarizes the values of the main parameters of the simulation.

In the initial state, a randomly chosen 10% of the excitatory synapses are potentiated; the neural parameters and the synaptic efficacies are chosen to have mean spontaneous excitatory activity of about 8 Hz.

Box B1 in Figure 4 is a snapshot of the network in its initial state. The big square sprayed with black and white dots provides a representation of the excitatory-excitatory part of the synaptic matrix; white (black) dots stand for potentiated (depressed) synapses, and their coordinates inside the square identify pre- and postsynaptic neurons. The average rate of each (of 1200) excitatory neuron labeling each axis of the synaptic matrix is shown in the (identical) horizontal and vertical rectangular boxes in B1–B4 (rates are averaged in 500 ms before the acquisition of the synapses).

The learning protocol is as follows (see boxes B2–B4 in Figure 4): there are two stimuli (S_1,S_2), which are directed to neurons 1–120 and 121–240, respectively. After 1 second of spontaneous activity, stimulus S_1 is presented for 1 second (B2). The rates of neurons 1–120 can be seen to be much higher than spontaneous activity. Then, after 1 second with no stimuli, S_2 is presented for 1 second (B3). The rates of neurons 121–240 are high; the network is then left without stimuli for 4 seconds (B4). The synaptic matrix is preserved.

The synaptic dynamics in the simulation is in qualitative agreement with expectations (see Annunziato et al., 1998). The figures show that potentiation occurs for high pre- and postsynaptic frequencies (during stimulation) depression occurs for high pre- and low postsynaptic frequencies and for low presynaptic frequencies, synapses remain essentially unaffected.

In the structured network, in the absence of stimuli (B4), neurons connected by potentiated synapses emit spikes at spontaneous rates that are, on average, about twice as high as the rates prior to the presentation of the stimuli. The scale in Figure 4 makes it difficult to notice. It is noteworthy that despite this significant change in rates, the structured synaptic matrix remains stable. The phenomenology exposed by the figure provides a first step toward answering some of the questions raised in section 1: the neurons split in two populations with different spontaneous rates as a result of the synaptic structuring induced by the incoming stimuli, yet the new higher rates turn out to be low enough and do not affect the synaptic matrix.

Table 1: Parameters of the Simulated Network.

Parameter	Value	
N_E	1200	
N_I	300	
c_E	0.2	
c_I	0.1	
β_E	1	(θ/s)
β_I	4	(θ/s)
J_{ext}	0.025	(θ)
J_{IE}	0.027	(θ)
J_{EI}	0.090	(θ)
J_{II}	0.070	(θ)
Δ_J	0.25	
D	4	
d_0	1	(ms)
d_{D-1}	3	(ms)
α	20	(s^{-1})
β	20	(s^{-1})
a	0.37	
b	0.22	
J_p	0.065	(θ)
J_d	0.02	(θ)
θ_J	0.5	
θ_V	0.8	(θ)
ν_{ext}	8	(Hz)
ν_{ext}^{stim}	48	(Hz)
x	0.5	
$c_{\mathrm{ext}} = (1-x)c_E$	0.1	

Notes: N_E, N_I = number of excitatory and inhibitory neurons. c_E, c_I = excitatory and inhibitory connectivities; the neural threshold θ for emission of a spike is set to one and serves as the unit for the depolarization. β_E, β_I = decay parameters for excitatory and inhibitory linear neurons. J_{ext} = efficacy of synapses from external neurons. $J_{\delta\gamma}$, $\delta, \gamma \in (E, I)$ = nonplastic synaptic efficacies involving inhibitory neurons (the $E-E$ synapses, which are plastic, are treated separately; see below). Δ_J = relative variance of the efficacies of the fixed synapses, all in units of θ. α (β) = refresh term driving the synaptic dynamic variable V_J toward the lower (upper) long-term value; a (b) is the up (down) jump of V_J, upon a presynaptic spike (they are dimensionless because V_J is). J_p (J_d) = long-term-memory potentiated (depressed) synaptic efficacies for the plastic synapses between excitatory neurons. θ_J = dimensionless threshold for V_J. θ_V = threshold for the postsynaptic depolarization. ν_{ext} = frequency of external neurons; ν_{ext}^{stim} = frequency of the external neurons when they code for stimuli. Of the $c_E N_E$ excitatory connections received by a neuron, $x c_E N_E$ are recurrent ($x = 0.5$ in our case) and $c_{ext} N_E = (1-x) c_E N_E$ are from external neurons. D = number of synaptic delays, ranging from d_0 to d_{D-1}. See also the text.

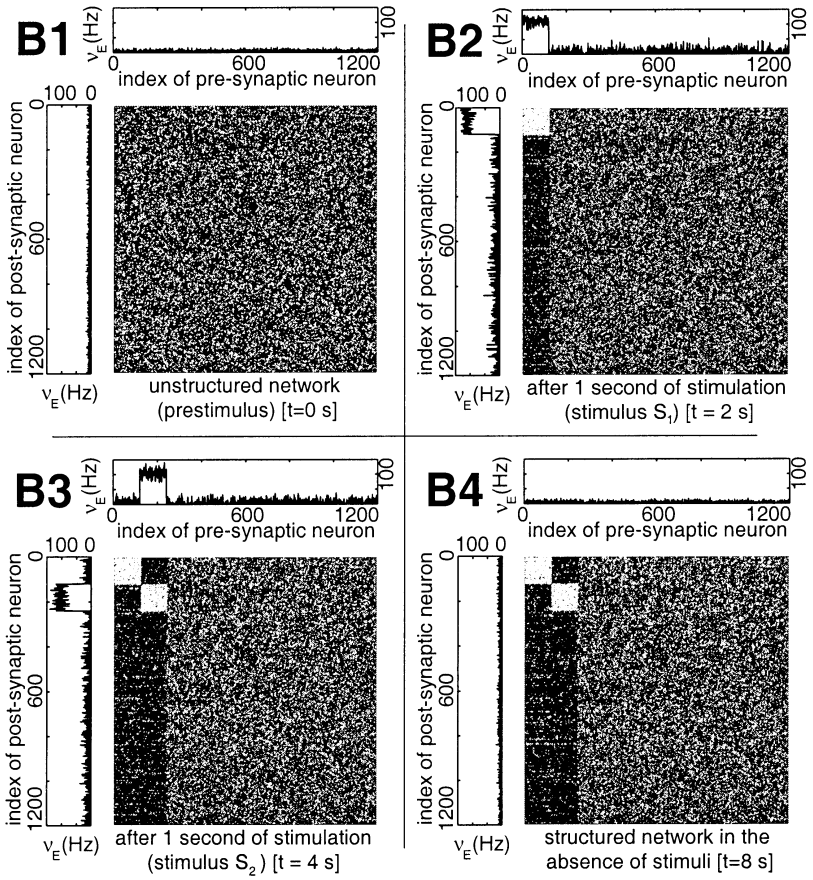

Figure 4: Illustration of the stochastic learning dynamics. See the text for details.

7 Performance

In section 2 we argued that a major advantage of an (asynchronous) event-driven approach is that the computational complexity of the simulation remains linear in N (of order νcN) when the synaptic dynamics is introduced. We now provide empirical evidence for this scaling, and also a quantitative measure of the memory occupation. Figure 5 (left) shows the CPU time per neuron, in milliseconds, needed to complete a simulation of 1 second of neural time, for both static and dynamic synapses versus network size N.

For different values of N, the synaptic efficacies are changed, keeping $c = 0.1$ and other parameters fixed, in order to maintain a fixed value for the (stable) spontaneous frequencies of the excitatory ($\nu = 2$ Hz) and inhibitory

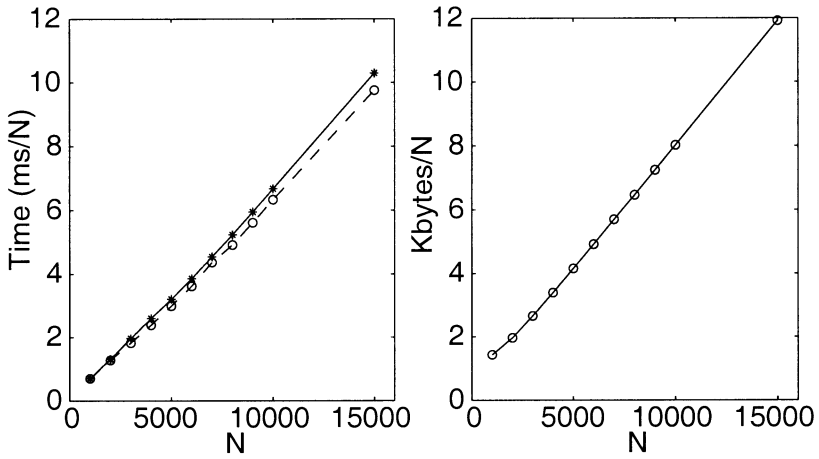

Figure 5: (Left) Execution time per neuron (in milliseconds) versus N. Solid line: synaptic dynamics on; dashed line: synaptic dynamics off. (Right) Memory allocation (in Kbytes) per neuron versus N. Each synapse is stored in 8 bytes (the analog, float value of V_J, the two-valued efficacy J, and the relative address of the next synapse on the same portion of the axon, same delay).

($\nu = 4$ Hz) populations. The test has been carried out on a Pentium II, 300 MHz PC with 256 Mb RAM. The expected linear scaling of the CPU time per neuron with N is remarkably well reproduced in the test and also for the case with synaptic dynamics.

It is also apparent from the figure that CPU time consumption is not much different going from static to dynamic synapses. The magnitude of the difference depends on the parameters of the network. Essentially it depends on the ratio between the number of dynamic synapses (those between excitatory neurons) and the number of intrinsically static synapses (those involving an inhibitory neuron, or an external neuron). With reference to Table 1, the following factors govern the above difference: N_E/N_I, x, and ν_E/ν_I. For the parameters chosen for the test, the introduction of synaptic dynamics accounts for about 6% of the CPU time, while about 30% goes for the generation of external events. The management of the queues associated with the synaptic delays accounts for another 6% in the case of $D = 4$ delays. Most of the remaining time is spent in updating the neural depolarizations and scanning the (compressed) axon in order to determine after each update the next target neuron for the current spike.

In Figure 5 (right) we plot the memory occupation per neuron in Kbytes, versus N. As expected, memory occupation is dominated by the storage of synapses, and the constant slope of the curve indicates that the storage per neuron scales linearly with N.

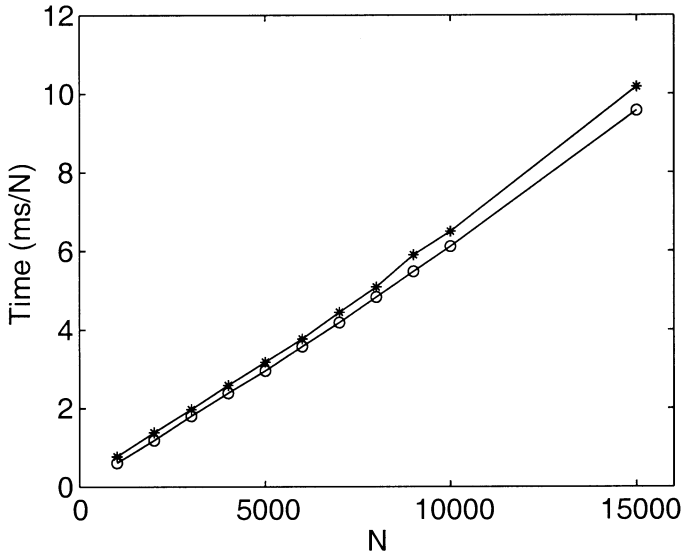

Figure 6: Execution time per neuron (in milliseconds) versus N, for two values of D (the number of delays). *: $D = 16$, o: $D = 1$. The relative difference stabilizes around 5% for large N.

In both plots, each point corresponds to 10 repetitions of the simulation for the corresponding value of N; the error bars are too small to be discernible. The runs contributing to each point differ in the initial neural and synaptic conditions, the realization of the random connectivity, and the trains of external spikes.

Figure 6 shows how the execution time per neuron depends on the number D of delays. For large N, the relative time difference between $D = 1$ and $D = 16$ is about 5%, thus proving that the algorithm is not very sensitive to the number of delays (layers).

8 Discussion

We have described a tool for fully event-driven simulations of large networks of spiking neurons and spike-driven, dynamical synapses. We pointed out reasons that it is efficient and flexible enough for pursuing in semirealistic conditions questions related to synaptic dynamics and its interaction with neural spike emission.

The algorithm relies on rather general assumptions on the neural and synaptic dynamics:

1. That neurons emit spikes as a result of the spikes they receive and a

deterministic evolution of the depolarization between afferent spikes; there are no constraints on the deterministic evolution of the depolarization following the arrival of an afferent spike.

2. That presynaptic spikes trigger changes in the synapses, with a deterministic evolution of the analog synaptic variable between two successive presynaptic spikes.[2]
3. That spikes emitted by different external neurons are statistically independent.
4. That synaptic delays can be assumed to form a discrete (not too large) set.

We have illustrated how the simulation affects synaptic structuring in a network of linear IF neurons according to a specific, spike-driven synaptic dynamics. This particular rule is new (Annunziato et al., 1998; Annunziato & Fusi, 1998), and the results shown are meant only as a preliminary insight into its working in the context of a network of IF neurons.

With the tool, proposed here, for the simulation of large networks with coupled neural and synaptic dynamics, ideas and hypotheses concerning the collective effects of synaptic dynamics can be developed and tested. Some were raised in section 1; others we mention next:

- The stability of neural activities can be probed when the synaptic dynamics is turned on.
- We have checked that a network with prelearned, static synapses, clamped to a configuration compatible with the expected structure due to LTP and LTD dynamics, is able to sustain stimulus-selective, stable collective states while maintaining a low-frequency, stable spontaneous activity. The question is whether, starting from an unstructured synaptic state, the specific synaptic dynamics used here is able to drive the noisy, fluctuating, finite-size network to the above expected synaptic structure. If not, what modification of the synaptic dynamics will do?
- If the answer to the previous question is positive, one must investigate whether the (stochastic) changes in the synaptic matrix have a negligible probability in the absence of stimuli, those induced by the stimuli are such that neural activities are kept stable, and that synaptic changes have a negligible probability also when the network is reverberating in an attractor, for these constraints ensure that the network does not forget recently learned, stimuli.

[2] If synaptic matrices are not compressed, it is straightforward to let the synaptic evolution be driven also by postsynaptic spikes, as in Markram, Lubke, Frotscher, and Sakmann (1997) and Häfliger et al. (1997).

- Simulating the coupled neural and synaptic dynamics makes it possible to study the conditions under which the system exhibits experimentally observed effects, such as the generation of context correlations (Miyashita, 1988; Yakovlev, Fusi, Berman, & Zohari, 1998), which reflect themselves in the structure of neural selective delay activities.
- The simulation tool is flexible enough to allow the study of modified or alternative synaptic dynamics, allowing, for example, the test of several recently proposed synaptic models (Häfliger et al., 1997; Senn et al., 1997) and their implications for the collective behavior of a large network.

These issues are under study, together with extensions and improvements of the algorithm, some of which have already been mentioned. A particularly interesting extension is related to the fact that the event-driven algorithm is well suited for distributed computing. Each computing node would contain a subpopulation of neurons and their complete dendritic tree (the corresponding set of synapses). This choice would eliminate the need to propagate synaptic information across different computing nodes, reducing the communication load. The messages to be passed among the processing machines would be essentially spikes, or what we have called events.

The extension to multiple, concurrent computing nodes requires some care. One must solve problems of the causal propagation of events: the "real" CPU time each node spends per unit "biological" time may not be equal in all nodes because of different emission frequencies of neurons simulated in each node. Because of this, a "fast" node might receive old events from a "slow" one, with time labels that are "obsolete." This is a typical synchronization problem in concurrent computing. Its solution goes beyond the scope of this article.

Appendix: Extension to Arbitrary Interspike Neural Dynamics

The simulation method described in this article deals with cases in which in the absence of spikes, the neuron's depolarization is monotonically decreasing, and hence a neuron could emit a spike only upon the arrival of an excitatory spike. The technique can be extended to cases in which the neural state variables have an arbitrary deterministic evolution between consecutive afferent spikes.

In such cases, when a neuron receives a spike, the time t_s can be determined, when that neuron will in turn emit a spike (if any), on the basis of the known evolution of the depolarization. But before t_s, this same neuron might receive other spikes, and each one of those will in general redefine t_s. In other words, if there are events to be processed in the queues at times earlier than t_s, the emission of the spike, predicted to occur at t_s, is "uncertain," and t_s must be updated each time an older spike is processed. This alters

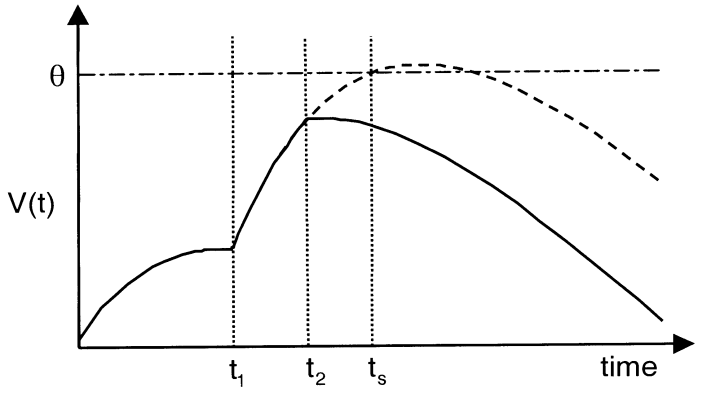

Figure 7: Time evolution of the depolarization $V(t)$ according to $\dot{V}(t) = -\beta + I(t)$, when the afferent current $I(t)$ is subjected to the dynamic law $\tau\dot{I}(t) = -I(t) + \sum_j J_j \sum_k \delta(t - t_j^{(k)} - d_j)$ as in Amit and Tsodyks (1992). The solid curve shows the time evolution of $V(t)$ when an excitatory spike is received in t_1, followed by an inhibitory one in $t_2 > t_1$. In this condition, the neuron does not emit in the time interval shown. The dashed line shows the time course of $V(t)$ in the absence of the second inhibitory spike in t_2. In this case, the neuron emits at $t = t_s > t_1$. The value of t_s can be deterministically computed from the above equations, possibly also using efficient methods such as those described in Srinivasan and Chiel (1993).

the natural order in time of the spikes emitted in the network needed to operate our algorithm. One must therefore devise a way of handling these "uncertain" events (to reproduce the correct causal dependencies) without increasing significantly the computational complexity of the simulation.

To show how this can be achieved, it is useful to consider the situation depicted in Figure 7 (it corresponds to a simplified description of the effects of dynamical conductances on $V(t)$). Two successive events are scheduled to reach the same neuron i at times t_1 and $t_2(> t_1)$. If the evolution of the neural state following t_1 is not a monotonic decay, it is possible for the target neuron i to reach the threshold and emit a spike at time $t_s > t_1$. If $t_s < t_2$, the spike at t_s becomes a "real" event and undergoes the usual processing. If $t_s > t_2$, as in Figure 7, when the spike received at $t = t_2$ is excitatory, it will shift the next emission to $t_s' < t_s$, while an inhibitory spike will delay it to $t_s' > t_s$ or suppress it (which is the case shown in the figure). The foreseen event (i, t_s) is therefore termed "uncertain."

To take into account the uncertain events, the algorithm is modified as follows:

- One new queue is created, in which the uncertain events are placed; all events generated in the network are now born "uncertain."

- Each new uncertain event is placed at the correct position in the new queue, according to its temporal label. If the length of the queue at that moment is n, this implies an additional computational load $\mathcal{O}(\log_2(n))$ (insertion of an element in an ordered list; see below for an estimate of n).
- If an event (i, t_s) is already present in the queue, each spike received by neuron i will redefine t_s. As a result, the event (i, t_s) could change its position in the queue, implying yet another $\mathcal{O}(\log_2(n))$ load (in the worst case of an arbitrary shift of the position in the queue)—or the event could be suppressed and be deleted from the queue.
- The previous operations preserve the order of the queue such that the initially ordered queue of uncertain events is kept ordered throughout the simulation without any need of global sorting.
- An event (i, t_s) becomes "real," and enters the first queue of events to be processed when there are no more spikes to be processed in the network at times earlier than t_s: there are no more events that could affect t_s.

The additional computational complexity is then mainly due to the insertion and repositioning of an uncertain event in a sorted array.

The average length n of the queue of uncertain events can be estimated by the product of the mean time T the spike will stay there (waiting to become real or to disappear) and the number of events $N\nu$ generated per unit time: $n \approx N\nu T$.

To get an estimate of T, we note that on average, the interval (t_1, t_2) is shorter than the interval between two successive spikes emitted by neuron i. From the definition of the uncertain event (generated at t_1, and scheduled to happen at t_s), we cannot have a "real" spike emitted by the same neuron i between t_1 and t_s, so if we call t_{s-1} the emission time of the previous spike, $T = \langle t_s - t_1 \rangle < \langle t_s - t_{s-1} \rangle = 1/\nu$, since $t_{s-1} < t_1$. Therefore, $T < 1/\nu$.

In the worst case, a neuron generates an uncertain event each time it receives a spike, so in the unit time, the average number of insertion per neuron is $c\nu N$.

Rearrangements, which involve only neurons with a predicted uncertain event, are n in the worst case, and each spike in the network affects cn of the neurons associated with uncertain events. On average, ν spikes are emitted per unit time by each neuron, so the number of rearrangements is $cn\nu = c\nu^2 NT$.

Allowing an arbitrary time evolution for the postspike depolarization will therefore cost an additional complexity per neuron of order $\nu cN(1 + \nu T)\log_2(N\nu T)$ ($c\nu N \log_2(n) = c\nu N \log_2(N\nu T)$ due to the insertion of new events in the ordered queue, plus $c\nu^2 NT \log_2(N\nu T)$ due to rearrangements in the queue—still a modest price to pay when compared to order N^2 complexity of the synchronous simulation.

Acknowledgments

Most of this work has been discussed in depth with D. J. Amit. We heartily thank him for his many stimulating contributions. We acknowledge very useful comments from S. Fusi, D. Del Pinto, and F. Carusi. This study has been supported in part by the RM12 initiative of INFN.

References

Amit, D. J. (1995). The Hebbian paradigm reintegrated: Local reverberations as internal representations. *Behavioural and Brain Sciences*, *18*, 617–657.

Amit, D. J., & Brunel, N. (1997a). Model of global spontaneous activity and local structured (learned) delay activity during delay periods in cerebral cortex. *Cerebral Cortex*, *7*, 237–252.

Amit, D. J., & Brunel, N. (1997b). Dynamics of a recurrent network of spiking neurons before and following learning. *Network*, *8*, 373–404.

Amit, D. J., & Fusi, S. (1994). Learning in neural networks with material synapses. *Neural Computation*, *6*(5), 957–982.

Amit, D. J., & Tsodyks, M. (1992). Effective and attractor neural networks in cortical environment. *Network*, *3*, 121–137.

Annunziato, M., Badoni, D., Fusi, S., & Salamon, A. (1998). Analog VLSI implementation of a spike driven stochastic dynamical synapse. In L. Niklasson, M. Boden, & T. Ziemke (Eds.), *Proceedings of the 8th International Conference on Artificial Neural Networks, Skovde, Sweden* (Vol. 1, pp. 475–480). Berlin: Springer-Verlag.

Annunziato, M., & Fusi, S. (1998). Queuing theory for spike-driven synaptic dynamics. In L. Niklasson, M. Boden, & T. Ziemke (Eds.), *Proceedings of the 8th International Conference on Artificial Neural Networks, Skovde, Sweden* (Vol. 1, pp. 117–122). Berlin: Springer-Verlag.

Bower, J. M., & Beeman, D. (1998). *The book of GENESIS.* New York: Springer-Verlag.

Delorme, A., Gautrais, J., van Rullen, R., & Thorpe, S. (1999). SpikeNET: A simulator for modeling large networks of integrate and fire neurons. In J. M. Bower (Ed.), *Computational neuroscience: Trends in research 1999, neurocomputing* (Vols. 26–27, pp. 989–996). Amsterdam: Elsevier Science.

Fusi, S., Annunziato, M., Badoni, D., & Amit, D. J. (2000). Spike-driven synaptic plasticity: Theory, simulation, VLSI implementation. *Neural Computation, 12*(10), 2227–2258.

Fusi, S., & Mattia, M. (1999). Collective behavior of networks with linear (VLSI) integrate-and-fire neurons. *Neural Computation*, *11*(3), 633–653.

Häfliger, P., Mahowald, M., & Watts, L. (1997). A spike based learning neuron in analog VLSI. In M. C. Mozer, M. I. Jordan, & T. Petsche (Eds.), *Advances in neural information processing systems, 9*, (p. 692). Cambridge, MA: MIT Press.

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

Markram, H., Lubke, J., Frotscher, M., & Sakmann, B. (1997). Regulation of synaptic efficacy by coindence of postsynaptic APs and EPSPs. *Science*, *275*, 213–215.

Mattia, M., Del Giudice, P., & Amit, D. J. (1998). Asynchronous simulation of large networks of spiking neurons and dynamical synapses. In L. Niklasson, M. Boden, & T. Ziemke (Eds.), *Proceedings of the 8th International Conference on Artificial Neural Networks, Skovde, Sweden* (Vol. 2, pp. 1045–1050). Berlin: Springer-Verlag.

Miyashita, Y. (1988). Neuronal correlate of visual associative long-term memory in the primate temporal cortex. *Nature*, *335*, 817.

Petersen, C. H., Malenka, R. C., Nicoll, R. A., & Hopfield, J. J. (1998). All-or-none potentiation at CA3–CA1 synapses. *Proc. Natl. Acad. Sci. USA*, *95*, 4732–4737.

Senn, W., Tsodyks, M., & Markram, H. (1997). An algorithm for synaptic modification based on exact timing of pre- and post-synaptic action potentials. In W. Gerstner, A. Germond, M. Asler, & J.-D. Nicoud (Eds.), *Artificial neural networks–ICANN '97* (Vol. 1327, pp. 121–126). Berlin: Springer-Verlag.

Srinivasan, R., & Chiel, H. J. (1993). Fast calculation of synaptic conductances. *Neural Computation*, *5*, 200–204.

Tsodyks, M., Mit'kov, I., & Sompolinsky, H. (1993). Pattern of synchrony in inhomogeneous networks of oscillators with pulse interactions. *Phys. Rev. Lett.*, *71*, 1280.

Watts, L. (1994). Event-driven simulation of networks of spiking neurons. In J. D. Cowan, G. Tesauro, & J. Alspector (Eds.), *Advances in neural information processing systems*, *6*, (pp. 927–934). San Mateo, CA: Morgan Kaufmann Publishers.

Yakovlev, V., Fusi, S., Berman, E., & Zohari, E. (1998). Inter-trial neuronal activity in inferior temporal cortex: A putative vehicle to generate long-term visual associations. *Nature Neuroscience*, *1*(4), 310–317.

Received February 22, 1999; accepted September 28, 1999.