Efficient event-driven simulation of large networks of spiking neurons and dynamical synapses

Maurizio Mattia, Paolo Del Giudice
Physics Laboratory, Istituto Superiore di Sanità
V.le Regina Elena 299, 00161 Roma, Italy
mattia@ibmter.iss.infn.it, paolo@ibmteo.iss.infn.it
http://jupiter.roma1.infn.it

November 2, 1999

Abstract

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 post-synaptic neurons. In fact, the computational load per neuron in presence of the synaptic dynamics grows linearly with the number of neurons and is only about 6% more than the load with fixed synapses. And 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 (IF) 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 the theory, when it cannot fully describe the richness of the phenomenology embedded in the mathematical model, they 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.

While there is a wide agreement on the above statements, there are several reasons why network simulations still wait for the inclusion of a crucial ingredient: the synaptic dynamics. Learning has been so far incorporated in the modeling of (recurrent) feedback neural networks as a static prescription for the 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 and Amit & Brunel, 1997a, 1997b).

But learning is guided by neural activities, which in turn are governed by the acquired synapses. Hence, if these models have to make the leap towards 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 the synaptic variables. Such evolution will be driven by the neural activities, driven in turn by the afferent stimuli. With dynamical synapses, an incoming stimulus will not only make a learned information pop up from the memory store and be put in an active state, but will also actively contribute to expand the store.

Moreover, if synaptic dynamics is a function of neural activities, then the question of the stability of any acquired synaptic stock and any activity distribution rears its head. This is true both for underlying levels of spontaneous activity and for computational expressions by selective activity distributions. In other words, any neural activity distribution may produce synaptic changes and consequently start drifting away, either from what was considered as the spontaneous rates or from 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) is put into question by the synaptic dynamics. Thus simulating the joint
dynamics is doubly necessary.

There are serious problems to be faced in carrying out such a program. Semi-realistic networks must contain a large number of cells to reproduce realistic spike dynamics. A typical cortical module contains of 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 re-scaled network of $O(10^4)$ neurons would be feasible and still capture the most relevant aspects of the collective behaviour. 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 the 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 possibly bring back large scale simulations with double, coupled dynamics of neurons and synapses into the range of feasibility.

We have found some attempts in this direction:

An event-driven approach, similar in concept to the present one, has been 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 which affects cN neurons will entail the management of cN events, with the associated computational load of the corresponding time ordering. The above features of the simulation are consistent with its intended purpose of simulating small networks to be implemented in silicon, on one hand, and envisage the possibility to simulate complex neural elements, on the other hand.

GENESIS (Bower & Beeman, 1998) is a widely used, general purpose simulation software, and it can serve as a basis for a wide class of simulation strategies. It has been mostly used in simulations of relatively small networks of complex, multi-compartment 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 consid-
ered partially *asynchronous* as far as the management of the communication between modules is concerned.\footnote{See http://www.psc.edu/\textsc{Packages}/PGEnESIS/}

SpikeNET (Delorme, Gautrais, van Rullen, \& Thorpe, 1999) is a simulator with the stated purpose of simulating large networks of Integrate-and-Fire neurons. In it the single neuron dynamics is fully synchronous, with a surprisingly high value for the time step ($dt = 1\text{ms}$ for neurons spiking at $1\text{Hz}$); the network operates in a feed-forward fashion and is constrained to be very sparsely connected: each layer propagates to the next one a list of neurons which 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, non-iterative 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; then, 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 where 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 progress, while summary and outlook are provided in Section 8. A preliminary account of the present work is been 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, Runge-Kutta), a time step $\Delta t$ is chosen which serves as a *clock* providing the timing for the synchronous update of the dynamical variables. $\Delta t$ 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 $\Delta t$, 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 (i.e. a spike emitted by a neuron is communicated on average to $cN$ neurons), 2) neurons emit on average $\nu$ spikes per second. With a time step $\Delta t$ seconds, one has on average $cN/\Delta t$ synaptic updates per second per neuron. On the other hand, $\Delta t$ must be limited to prevent conflicts between arriving spikes: the average number of spikes per second received by a neuron is $\nu cN$, 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 $\Delta t$ 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'kova, & 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 that also 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 below. In that case the events (spikes) whose occurrence determines changes in the synaptic state will be much sparser in time, because a synapse only sees two neurons while a neuron contacts $cN$ synapses. The fine 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 inter-spike evolution of both be exactly interpolated at the same instances and become computationally negligible.

Such an event-driven approach is proposed in the present paper. 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 $\nu cN$ 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 time scales.

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

3 General features of the simulated network

The typical ingredients of the simulated network to be considered include:

- Neurons are of the Integrate-and-Fire type: the membrane depolar-
  ization 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 depolariza-
  tion surpasses a threshold. The interpolating dynamics can be generic.

- In addition to recurrent spikes, all neurons in the network also receive
  spikes from outside: low frequency, non-selective, 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: 1) external neurons emit consecutive spikes at random, independ-
ent instants with exponentially distributed inter-spike intervals (ISI)
and 2) spike emission by different external neurons is uncorrelated.
Current from outside is then a superposition of uncorrelated stochas-
tic 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 dy-
namics, such as the IF model for the neuron, the synaptic dynamics
will be assumed to be governed by the (instantaneous) pre-synaptic
spikes and the depolarization of the post-synaptic neuron, following a
recently proposed model (Annunziato, Badoni, Fusi, & Salamon, 1998;
Annunziato & Fusi, 1998). The pre-synaptic spikes serve as triggers for
synaptic changes, while the post-synaptic potential determines whether
the change is potentiating or depressing. Long-term memory is main-
tained 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, & Hop-
field, 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 a 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 in what follows 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 both for 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 which are generated in the network, and those which 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 the details of 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 \(d_0 < d_1 \ldots < d_{D-1}\) for spike transmission. Synapses are organized in matrix-structured ‘layers’, each layer corresponds 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 \(i\) which 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 which 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 pre-synaptic 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 yet sparser. The compressed version of the synaptic matrix in which only non-zero element 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 \(l\)-th queue is the pair \((i, t+d_l)\) \((d_l\) is the value of the \(l\)-th 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 of the synaptic values 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 retain
the same statistical distribution of the single train; this is the case for trains of spikes with Poissonian or Gaussian statistics.

These choices allow a significant simplification in the management of the external spikes: one can generate a single train of spikes, e.g. with a Poisson distribution, store successive external spikes in a single register 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 external Poissonian spike train of prescribed frequency (see the end of subsection 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 either by pre-filling the queues of events or by 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, the decay constant of the neuron’s depolarization.

Suppose a spike is emitted by neuron $k$ at time $T$ (A, in the figure). 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 post-synaptic 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 post-synaptic 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 post-synaptic 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 $Q_0$.
Figure 1: Schematic illustration of the algorithm. From 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 contains 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 text for details.
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. 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. This limitation can be relaxed. For the particular choice of synaptic dynamics, see below.

Once all the post-synaptic 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, it is necessary to sort the first events in all the queues, in order to choose the oldest among all events, to be processed. With a small number of allowed discrete values for the delays, this is a negligible additional computational load.

We have assumed (see e.g. Sec. 3) that the neuron's depolarization undergoes a monotonic decay between successive incoming spikes. This ensures that a spike can only be generated 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 mentioned at the end of the preceding subsection. 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 $\Delta t$ is randomly chosen from the exponential density function $p(\Delta t)$

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

where we assume that external neurons emit spikes at a mean rate $\nu_{ext}$ 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 $\nu_{\text{ext}}c_{\text{ext}}N$ external spikes per neuron, which is of $O(N)$. It should be pointed out that 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 next Section).

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 $O(\log_2(D + P))$ which is negligible in the used situation $D + P \ll N$, (see Section 7 and Fig. 6).

### 4.3 Remarks on specific features of the algorithm

- **Layered structure:** coding the synaptic matrix in a single 'layer' would imply: 1) 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 2) the spike just processed for one value of delay would have to be re-inserted 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 ($\nu d_{D-1}$) in the time window $(t - d_{D-1}, t)$ is very small. But when we consider the whole network, all neurons contribute a factor $\nu d_{D-1}$ to the average number of active events in the supposed single queue at time $t$, and so $m = N\nu d_{D-1}$.

**Example:** In a regime of spontaneous activity, in the absence of external
spikes, a large network of $N = 10^6$ neurons emitting $\nu = 2$ spikes per second, with maximal delay $d_{D-1} = 3ms$ 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 $O(D \log_2(600)) = O(D \log_2(d_{D-1}\nu N))$.

- Generation and handling of external spikes: having a separate generator of external spikes for each neuron in the network would imply 1) a relatively small increase in memory allocation, but 2) 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 one single element in the register for the external spike, $D + N$ elements would have to be sorted.

- Given the presence of a fast (neural) and a slow (synaptic) dynamics, one might think of a synchronous algorithm based on the simultaneous use of two very different $\Delta t$ for the neural and the synaptic evolution. But: 1) 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 $\Delta t$ would fail to reproduce correctly the most important regime. 2) A big $\Delta t$ for synapses would conflict with the resolution with which the time of emission of the spikes is determined, constrained in turn by the neural $\Delta t$; the input signal to the synapse would therefore be affected by a 'quantization' error, that 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, as an example, a linear IF neuron. Such 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 $\beta$ 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 $\theta$, a spike is emitted, $V_i$ is reset to $H \in (0, \theta)$, where it is forced to remain for a time $\tau_{\text{arp}}$, the absolute refractory period. In what follows $H = 0$. A ‘reflecting barrier’ at 0 forces $V_i$ to stay 0 whenever it is driven towards negative values, either by the constant decay, or by inhibitory spikes. This dynamics is illustrated in Figure 2.

![Figure 2: Illustration of the dynamics of the neural depolarization $V(t)$ (see 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 pre-synaptic neurons $j$ at times $t_j^{(k)}$ and transmitted with delays $d_{ij}$, whose post-synaptic contribution to $V_i(t)$ is $J_{ij}$.](image)

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 post-synaptic activities. An internal synaptic (dimensionless)
variable (‘synaptic potential’, $V_J$) describes the short-time analog response of the synapse to pre- and post-synaptic 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 pre-synaptic spike, $V_J$ undergoes a positive (negative) jump of size $a$ ($b$) if the post-synaptic depolarization is found to be above (below) a threshold $\theta_V$ (not the spike emission threshold, $\theta > \theta_V$). The accessible interval for $V_J$ is split in two parts by a synaptic threshold $\theta_J$. 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) $\theta_J$, $J$ undergoes a transition: $J_0 \rightarrow J_1$ (LTP) ($J_1 \rightarrow J_0$ (LTD)). In the time between two successive pre-synaptic spikes, $V_J$ is linearly driven towards 0 (1), if the last spike left it below (above) $\theta_J$. 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: 1) since any change in the synaptic state is triggered by a pre-synaptic spike, synaptic modifications will be rare for low frequency pre-synaptic neurons 2) the synaptic potential $V_J$ is attracted towards 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 pre-synaptic spikes must be shorter than the time needed for the linear decay to compensate for the jumps in $V_J$) 3) given the statistics of pre-synaptic spikes, the probability distribution of the post-synaptic depolarization determines the relative frequency of up and down jumps in $V_J$, and therefore of potentiations (LTP) and depressions (LTD): a post-synaptic neuron emitting at high rate will frequently have its depolarization near the threshold for spike emission $\theta$, hence above $\theta_V$, while a low rate neuron spends most of its time near the reflecting barrier at 0, and $V$ is likely to be below $\theta_V$. In the case of interest, of low transition probabilities for $J$, a synaptic transition can only occur when a fluctuation produces a burst of pre-synaptic 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 pre-synaptic activity, regardless the post-synaptic neural state, and high pre-synaptic activity will provoke potentiation (LTP) or depression (LTD) respectively for high or low spike rates in the post-synaptic neuron.
Figure 3: Illustration of the synaptic dynamics as driven by the activities of pre- and post-synaptic neurons. Top: depolarization and spikes of pre-synaptic neuron. Bottom: same for post-synaptic 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 absence of pre-synaptic spikes if $V_J(t) < \theta_J$ it drifts down; if $V_J(t) > \theta_J$ it drifts up. Pre-synaptic spikes trigger abrupt changes in $V_J$: upon arrival of a pre-synaptic spike, $V_J$ undergoes a positive jump of size $a$ if the post-synaptic depolarization is above the threshold $\theta_V$, 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 pre-synaptic spike makes $V_J$ cross the threshold $\theta_J$ (dashed line), and this provokes an instantaneous LTP transition of the synaptic efficacy $J = J_0 \rightarrow J = J_1$ (solid curve). The fifth pre-synaptic spike provokes a depression, LTD.
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 (∼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 post-synaptic neurons. To help inspection of the synaptic matrix, 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 (S1,S2), which are directed to neurons 1...120 and 121...240 respectively. After 1 s of spontaneous activity stimulus S1 is presented for 1 second (B2). The rates of neurons 1...120 can be seen to be much higher than spontaneous activity. Then, after one further second in absence of stimuli, S2 is presented for 1 second (B3). The rates of neurons 121...240 are high; the network is then left in absence of stimuli for 4 seconds (B4). The synaptic matrix is preserved.

The synaptic dynamics in the simulation is in qualitative agreement with expectations (see Amunziato et al., 1998). It is seen from the figures that: potentiation occurs for high pre- and post-synaptic frequencies (during stimulation); depression occurs for high pre- and low post-synaptic frequencies; for low pre-synaptic 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 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 the Introduction: 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 to affect the synaptic
\[ \begin{array}{|c|c|} \hline \text{Parameter} & \text{Value} \\ \hline N_E & 1200 \\ N_I & 300 \\ c_E & 0.2 \\ c_I & 0.1 \\ \hline \beta_E & 1 \ (\theta/s) \\ \beta_I & 4 \ (\theta/s) \\ J_{ext} & 0.025 \ (\theta) \\ J_{EI} & 0.090 \ (\theta) \\ J_{IE} & 0.027 \ (\theta) \\ J_{II} & 0.070 \ (\theta) \\ \Delta J & 0.25 \\ D & 4 \\ d_0 & 1 \ (ms) \\ d_{D-1} & 3 \ (ms) \\ \hline \end{array} \]

\[ \begin{array}{|c|c|} \hline \text{Parameter} & \text{Value} \\ \hline \alpha & 20 \ (s^{-1}) \\ \beta & 20 \ (s^{-1}) \\ a & 0.37 \\ b & 0.22 \\ J_p & 0.065 \ (\theta) \\ J_d & 0.02 \ (\theta) \\ \theta_J & 0.5 \\ \theta_V & 0.8 \ (\theta) \\ \nu_{ext} & 8 \ (Hz) \\ \nu_{stim} & 48 \ (Hz) \\ x & 0.5 \\ c_{ext} = (1-x)c_E & 0.1 \\ \hline \end{array} \]

Table 1: Parameters of the simulated network: \( N_E \), \( N_I \) are the number of excitatory and inhibitory neurons; \( c_E \), \( c_I \) are the excitatory and inhibitory connectivities; the neural threshold \( \theta \) for emission of a spike is set to one, and serves as the unit for the depolarization. \( \beta_E, \beta_I \) are the decay parameters for excitatory and inhibitory linear neurons; \( J_{ext} \) is the efficacy of synapses from external neurons; \( J_{EI}, J_{IE}, J_{II} \), \( \delta, \gamma \in (E,I) \) are the non-plastic synaptic efficacies involving inhibitory neurons (the \( E-E \) synapses, which are plastic, are treated separately, see below); \( \Delta J \) is the relative variance of the efficacies of the fixed synapses, all in units of \( \theta \); \( \alpha (\beta) \) is the the refresh term driving the synaptic dynamic variable \( V_J \) towards the lower (upper) long-term value; \( a \ (b) \) is the up (down) jump of \( V_J \), upon a pre-synaptic spike (they are dimensionless because \( V_J \) is); \( J_p \) \( J_d \) are the long-term memory potentiated (depressed) synaptic efficacies for the plastic synapses between excitatory neurons; \( \theta_J \) is the dimensionless threshold for \( V_J \); \( \theta_V \) is the threshold for the post-synaptic depolarization; \( \nu_{ext} \) is the frequency of external neurons; \( \nu_{stim} \) is the frequency of the external neurons when they code for stimuli; of the \( c_EN_E \) excitatory connections received by a neuron, \( xc_{ext}N_E \) are recurrent \((x = 0.5 \ \text{in our case})\) and \( c_{ext}N_E = (1-x)c_EN_E \) are from external neurons. \( D \) is the number of synaptic delays, ranging from \( d_0 \) to \( d_{D-1} \). See also text.
matrix.

Figure 4: Illustration of the stochastic learning dynamics. See 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 $\nu c N$) 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 sec. of neural time, both for static and for dynamic synapses vs 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 ($\nu = 2Hz$) and inhibitory ($\nu = 4Hz$) 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, also for the case with synaptic dynamics.

It is also apparent from the figure that the 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: $N_E/N_I$, $x$, $\nu_E/\nu_I$. 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 the updating of the neural depolarizations and in 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, vs $N$. 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 $N$.

In both plots, each point corresponds to ten 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, in the realizations of the random connectivity and in the trains of external spikes.

Figure 6 shows how the execution time per neuron depends on the number $D$ of delays. It is seen that for large $N$ the relative time difference between $D = 1$ and $D = 16$ is about 5%, thus proving that the algorithm is not much 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
Figure 5: Left: Execution time per neuron (in milliseconds) vs $N$. Solid line: synaptic dynamics on; dashed line: synaptic dynamics off. Right: memory allocation (in Kbytes) per neuron, vs $N$; 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).

reasons why it is efficient and flexible enough for pursuing in semi-realistic 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; no constraints on the deterministic evolution of the depolarization following the arrival of an afferent spike, 2) that pre-synaptic spikes trigger changes in the synapses, with a deterministic evolution of the analog synaptic variable between two successive pre-synaptic 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 effects synaptic structuring in a network of linear IF neurons, according to a specific, spike-driven synaptic dynamics. This particular rule is new (Anunziato et al., 1998; Anunziato & Fusi, 1998), and the results shown are only meant as a preliminary insight into its working in the context of a network of IF neurons.

$^2$If synaptic matrices are not compressed, it is straightforward to let the synaptic evolution be driven also by post-synaptic spikes, as in Markram, Lubke, Frotscher, and Sakmann (1997) and Häfliger et al. (1997).
Figure 6: Execution time per neuron (in milliseconds) vs $N$, for two values of $D$ (the number of delays). $*$: $D = 16$, $o$: $D = 1$. The relative difference stabilizes around 5% for large $N$.

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 the Introduction, others we mention below:

1) The stability of neural activities when the synaptic dynamics is turned on.

2) We have checked that a network with pre-learned, 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 the trick?

3) If the answer to (2) is positive, one must investigate whether a) the (stochastic) changes in the synaptic matrix have a negligible probability in the absence of stimuli, b) those induced by the stimuli be such that neural activities are kept stable, c) synaptic changes have a negligible probability also when the network is reverberating in an attractor, for these constraints ensure that the network does not loose memory of recently learned, stimuli.
4) 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 structure of neural selective delay activities.

5) 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 (Häufliger et al., 1997; Markram et al., 1997) and their implications for the collective behaviour 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 sub-population 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 called above “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 which are ‘obsolete’ for its present time. This is a typical ‘synchronization’ problem in concurrent computing. Its solution goes beyond the scope of this report.

Acknowledgments

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

Appendix. Extension to arbitrary inter-spike neural dynamics

The simulation method described above 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 re-define \( t_s \). 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 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 \( t = t_s \) 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 \( O(\log_2(n)) \) (insertion of an element in an ordered list, see later 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 \( O(\log_2(n)) \) load (in the
Figure 7: The 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 using also efficient methods such as those described in Srinivasan and Chiel (1993).

- 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 above operations preserve the order of the queue such that the initially ordered queue of uncertain events is kept ordered throughout the simulation without 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 the re-positioning 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: $n \approx N\nu T$. 

25
To get an estimate of $T$, we note that on the 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, $T = (t_s - t_1) < (t_s - t_{s-1}) = 1/\nu$, 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 $\alpha \nu N$.

Re-arrangements, 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 $\nu$ spikes are emitted per unit time by each neuron, so the number of re-arrangements is $cn\nu = \alpha \nu^2 N T$.

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

**References**


