LETTER **Communicated by Stefano Fusi**

Event-Driven Simulation Scheme for Spiking Neural Networks Using Lookup Tables to Characterize Neuronal Dynamics

Eduardo Ros
eduardo@atc.ugr.es
Richard Carrillo
rcarrillo@atc.ugr.es
Eva M. Ortigosa
eva@atc.ugr.es
Department of Computer Architecture and Technology, E.T.S.I. Informática, University of Granada, 18071, Spain

Boris Barbour
barbour@ens.fr
Laboratoire de Neurobiologie, Ecole Normale Supérieure, 75230 Paris Cedex 05, France

Rodrigo Agís
ragis@atc.ugr.es
Department of Computer Architecture and Technology, E.T.S.I. Informática, University of Granada, 18071, Spain

Nearly all neuronal information processing and interneuronal communication in the brain involves action potentials, or spikes, which drive the short-term synaptic dynamics of neurons, but also their long-term dynamics, via synaptic plasticity. In many brain structures, action potential activity is considered to be sparse. This sparseness of activity has been exploited to reduce the computational cost of large-scale network simulations, through the development of event-driven simulation schemes. However, existing event-driven simulations schemes use extremely simplified neuronal models. Here, we implement and evaluate critically an event-driven algorithm (ED-LUT) that uses precalculated look-up tables to characterize synaptic and neuronal dynamics. This approach enables the use of more complex (and realistic) neuronal models or data in representing the neurons, while retaining the advantage of high-speed simulation. We demonstrate the method's application for neurons containing exponential synaptic conductances, thereby implementing shunting inhibition, a phenomenon that is critical to cellular computation. We also introduce an improved two-stage event-queue algorithm, which allows

the simulations to scale efficiently to highly connected networks with arbitrary propagation delays. Finally, the scheme readily accommodates implementation of synaptic plasticity mechanisms that depend on spike timing, enabling future simulations to explore issues of long-term learning and adaptation in large-scale networks.

1 Introduction

Most natural neurons communicate by means of individual spikes. Information is encoded and transmitted in these spikes, and nearly all of the computation is driven by these events. This includes both short-term computation (synaptic integration) and long-term adaptation (synaptic plasticity). In many brain regions, spiking activity is considered to be sparse. This, coupled with the computational cost of large-scale network simulations, has given rise to the event-driven simulation schemes. In these approaches, instead of iteratively calculating all the neuron variables along the time dimension, the neuronal state is updated only when a new event is received.

Various procedures have been proposed to update the neuronal state in this discontinuous way (Watts, 1994; Delorme, Gautrais, van Rullen, & Thorpe, 1999; Delorme & Thorpe, 2003; Mattia & Del Giudice, 2000; Reutimann, Giugliano, & Fusi, 2003). In the most widespread family of methods, the neuron's state variable (membrane potential) is updated according to a simple recurrence relation that can be described in closed form. The relation is applied on reception of each spike and depends only on the membrane potential following the previous spike, the time elapsed, and the nature of the input (strength, sign):

$$V_{m,t} = f(V_{m,t-\Delta t}, \Delta t, J), \tag{1.1}$$

where V_m is the membrane potential, Δt is elapsed time (since the last spike), and J represents the effect of the input (excitatory or inhibitory weight).

This method can describe integrate-and-fire neurons and is used, for instance, in SpikeNET (Delorme et al., 1999; Delorme & Thorpe, 2003). Such algorithms can include both additive and multiplicative synapses (i.e., synaptic conductances), as well as short-term and long-term synaptic plasticity. However, the algorithms are restricted to synaptic mechanisms whose effects are instantaneous and to neuronal models, which can only spike immediately upon receiving input. These conditions obviously restrict the complexity (realism) of the neuronal and synaptic models that can be used.

Implementing more complex neuronal dynamics in event-driven schemes is not straightforward. As discussed by Mattia and Del Giudice (2000), incorporating more complex models requires extending the event-driven framework to handle predicted spikes that can be modified if

intervening inputs are received; the authors propose one approach to this issue. However, in order to preserve the benefits of computational speed, it must, in addition, be possible to update the neuron state variables discontinuously and also predict when future spikes would occur (in the absence of further input). Except for the simplest neuron models, these are nontrivial calculations, and only partial solutions to these problems exist. Makino (2003) proposed an efficient Newton-Raphson approach to predicting threshold crossings in spike-response model neurons. However, the method does not help in calculating the neuron's state variables discontinuously and has been applied only to spike-response models involving sums of exponentials or trigonometric functions. As we shall show below, it is sometimes difficult to represent neuronal models effectively in this form. A standard optimization in high-performance code is to replace costly function evaluations with lookup tables of precalculated function values. This is the approach that was adopted by Reutimann et al. (2003) in order to simulate the effect of large numbers of random synaptic inputs. They replaced the online solution of a partial differential equation with a simple consultation of a precalculated table.

Motivated by the need to simulate a large network of "realistic" neurons (explained below), we decided to carry the lookup table approach to its logical extreme: to characterize all neuron dynamics off-line, enabling the event-driven simulation to proceed using only table lookups, avoiding all function evaluations. We term this method ED-LUT (for event-driven-lookup table). As mentioned by Reutimann et al. (2003), the lookup tables required for this approach can become unmanageably large when the model complexity requires more than a handful of state variables. Although we have found no way to avoid this scaling issue, we have been able to optimize the calculation and storage of the table data such that quite rich and complex neuronal models can nevertheless be effectively simulated in this way.

The initial motivation for these simulations was a large-scale real-time model of the cerebellum. This structure contains very large numbers of granule cells, which are thought to be only sparsely active. An event-driven scheme would therefore offer a significant performance benefit. However, an important feature of the cellular computations of cerebellar granule cells is reported to be shunting inhibition (Mitchell & Silver, 2003), which requires noninstantaneous synaptic conductances. These cannot be readily represented in any of the event-driven schemes based on simple recurrence relations. For this reason, we chose to implement the ED-LUT method. Note that noninstantaneous conductances may be important generally, not just in the cerebellum (Eckhorn et al., 1988; Eckhorn, Reitböck, Arndt, & Dicke, 1990).

The axons of granule cells, the parallel fibers, traverse large numbers of Purkinje cells sequentially, giving rise to a continuum of propagation delays. This spread of propagation delays has long been hypothesized to

underlie the precise timing abilities attributed to the cerebellum (Braitenberg & Atwood, 1958). Large divergences and arbitrary delays are features of many other brain regions, and it has been shown that propagation and synaptic delays are critical parameters in network oscillations (Brunel & Hakim, 1999). Previous implementations of event queues were not optimized for handling large synaptic divergences with arbitrary delays. Mattia and Del Giudice (2000) implemented distinct fixed-time event queues (i.e., one per delay), which, though optimally quick, would become quite cumbersome to manage when large numbers of distinct delays are required by the network topology. Reutimann et al. (2003) and Makino (2003) used a single ordered event structure in which all spikes are considered independent. However, for neurons with large synaptic divergences, unnecessary operations are performed on this structure, since the arrival order of spikes emitted by a given neuron is known. We introduce a two-stage event queue that exploits this knowledge to handle efficiently large synaptic divergences with arbitrary delays.

We demonstrate our implementation of the ED-LUT method for a model of a single-compartment neuron receiving exponential synaptic conductances (with different time constants for excitation and inhibition). In particular, we describe how to calculate and optimize the lookup tables and the implementation of the two-stage event queue. We then evaluate the performance of the implementation in terms of accuracy and speed and compare it with other simulation methods.

2 Overview of the ED-LUT Computation Scheme

The ED-LUT simulation scheme is based on the structures shown in Figure 1. A simulation is initialized by defining the network and its interconnections (including latency information), giving rise to the neuron list and interconnection list structures. In addition, several lookup tables that completely characterize the neuronal and synaptic dynamics are calculated: the exponential decay of the synaptic conductances; a table that can be used to predict if and when the next spike of a cell would be emitted, in the absence of further input; and a table defining the membrane potential (V_m) as a function of the combination of state variables at a given point in the past (in our simulations, this table gives V_m as a function of the synaptic conductances and the membrane potential, all at the time of the last event, and the time elapsed since that last event). If different neuron types are included in the network, they will require their own characterization lookup tables with different parameters defining their specific dynamics. Each neuron in the network stores its state variables at the time of the last event, as well as the time of that event. If short- or long-term synaptic dynamics are to be modeled, additional state variables are stored per neuron or per synapse.

When the simulation runs, events (spikes) are ordered using the event heap (and the interconnection list—see below) in order to be processed in

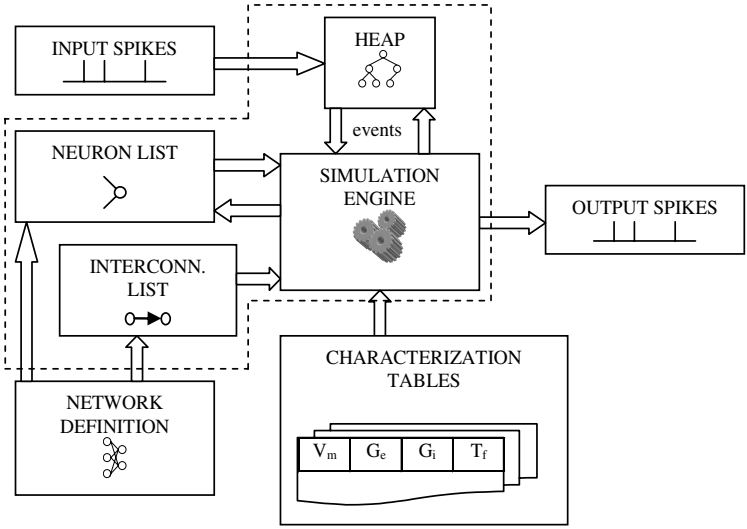


Figure 1: Main structures of the ED-LUT simulator. Input spikes are stored in an input queue and are sequentially inserted into the spike heap. The network definition process produces a neuron list and an interconnection list, which are consulted by the simulation engine. Event processing is done by accessing the neuron characterization tables to retrieve updated neuronal states and forecast spike firing times.

chronological order. The response of each cell to spikes it receives is determined with reference to the lookup tables, and any new spikes generated are inserted into the event heap. External input to the network can be fed directly into the event heap. Two types of events are distinguished: firing events, the times when a neuron emits a spike, and propagated events, the times when these spikes reach their target neurons. In general, each firing event leads to many propagated events through the synaptic connection tree. Because our synaptic and neuronal dynamics allow the neurons to fire after inputs have been received, the firing events are only predictions. The arrival of new events can modify these predictions. For this reason, the event handler must check the validity of each firing event in the heap before it is processed.

3 Two-Stage Event Handling

Events (spikes) must be treated in chronological order in order to preserve the causality of the simulation. The event-handling algorithm must therefore be capable of maintaining the temporal order of spikes. In addition, as our neuronal model allows delayed firing (after inputs), the algorithm

must cope with the fact that predicted firing times may be modified by intervening inputs.

Mattia and Del Guidice (2000) used a fixed structure (called a synaptic matrix) for storing synaptic delays. This is suited only for handling a fixed number of latencies. In contrast, our simulation needed to support arbitrary synaptic delays. This required that each spike transmitted between two cells is represented internally by two events. The first one (the firing event) is marked with the time instant when the source neuron fires the spike. The second one (the propagated event) is marked with the time instant when the spike reaches the target neuron. Most neurons have large synaptic divergences. In these cases, for each firing event, the simulation scheme produces one propagated event per output connection.

The algorithm efficiency of event-driven schemes depends on the size of the event data structure, so performance will be optimal under conditions that limit load (low connectivity, low activity). However, large synaptic divergences (with many different propagation delays) are an important feature of most brain regions. Previous implementations of event-driven schemes have used a single event heap, into which all spikes are inserted and reordered (Reutimann et al., 2003; Makino, 2003). However, treating each spike as a fully arbitrary event leads to the event data structure's becoming larger than necessary, because the order of spike emission by a given neuron is always known (it is defined in the interconnection list).

We have designed an algorithm that exploits this knowledge by using a multistage event-handling process. Our approach is based on a spike data structure that functions as an interface between the source neuron events and target neurons. We use a heap data structure (priority queue) to store the spikes (see appendix A for a brief motivation). The output connection list of each neuron (which indicates its target cells) is sorted by propagation delay. When a source neuron fires, only the event corresponding to the lowest-latency connection is inserted into the spike heap. This event is linked to the other output spikes of this source neuron. When the first spike is processed and removed from the heap, the next event in the output connection list is inserted into the spike heap, taking into account the connection delay. Since the output connection list of each neuron is sorted by latency, the next connection carrying a spike can easily be found. This process is repeated until the last event in the list is processed. In this way, the system can handle large connection divergences efficiently. Further detail on the performance of this optimization is reported in appendix A.

Each neuron stores two time labels. One indicates the time the neuron was last updated. This happens on reception of each input. As described in Figure 2, when a neuron is affected by an event, the time label of this neuron is updated to t_{sim} if it is an input spike (propagated event) or to $t_{sim} + t_{refrac}$ if it is an output spike (firing event), to prevent it from firing again during the refractory period. This is important because when the characterization tables are consulted, the time label indicates the time that has elapsed since

```
While t_sim < t_end
{
Extract the event with a shortest latency in the spike heap

    If it is a firing event
        If it is still a valid event and the neuron is not under a refractory
        period
            Update the neuron state: V_m, g_exc, g_inh to the postfiring state.
            Prevent this neuron from firing during the refractory period.
            (Once this is done, update the neuron time label to t_sim + t_refrac).
            Predict if the source neuron will fire again with the current
            neuron state.
            If the neuron will fire:
                Insert a new firing event into the spike heap.
            Insert the propagated event with the shortest latency (looking
            at the output connection list).

    If it is a propagated event
        Update the target neuron state: V_m, g_exc, g_inh looking at the
        characterization tables, before the event is computed.
        Modify the conductances (g_exc, g_inh) using the connection weight
        (G_exc,i, G_inh,i) for the new spike.
        Update the neuron time label to t_sim.
        Predict if the target neuron will fire.
        If it fires:
            Insert the firing event into the spike heap with the predicted
            time.
        Insert only the next propagated event with the next shortest latency
        (looking at the output connection delay table).
}
```

Figure 2: Simulation algorithm. This pseudocode describes the simulation engine. It processes all the events of the spike heap in chronological order.

the last update. The other time label maintains the up-to-date firing time prediction. This is used to check the validity of events extracted from the central event heap.

The basic computation scheme consists of a processing loop, in each iteration of which the next event (i.e., with the shortest latency) is taken from the spike heap. This event is extracted from the spike heap structure, the target neuron variables are updated (in the neuron list structure), and if the affected neurons generate them, new events are inserted into the spike heap. Also, if the processed event is a propagated event, the next spike from the output connection list of the neuron is inserted into the heap. This computation scheme is summarized in Figure 2. It should be noted that

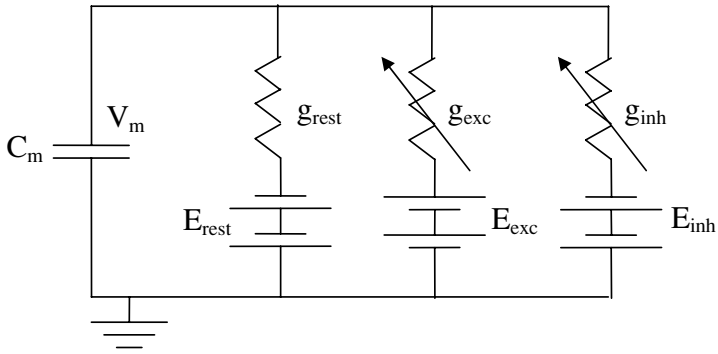


Figure 3: Equivalent electrical circuit of a model neuron. g_{exc} and g_{inh} are the excitatory and inhibitory synaptic conductances, while g_{rest} is the resting conductance, which returns the membrane potential to its resting state (E_{rest}) in the absence of input stimuli.

events are inserted into the heap in correct temporal sequence, but only the spike with the shortest latency is ever extracted. Events that are superseded by intervening inputs in the neuron concerned are left in the event heap. They are discarded upon extraction if invalid (this is checked against the up-to-date firing prediction stored in the neuron).

4 Neuronal and Synaptic Models

We model neurons as single compartments receiving exponential excitatory and inhibitory synaptic conductances with different time constants. The basic electrical components of the neuron model are shown in Figure 3. The neuron is described by the following parameters: (1) membrane capacitance, C_m, (2) the reversal potentials of the synaptic conductances, E_{exc} and E_{inh}, (3) the time constants of the synaptic conductances, τ_{exc} and τ_{inh}, and (4) the resting conductance and its reversal potential, g_{rest} and E_{rest}, respectively. The membrane time constant is defined as $\tau_m = C_m/g_{rest}$. The neuron state variables are the membrane potential (V_m), the excitatory conductance (g_{exc}), and the inhibitory conductance (g_{inh}). The synaptic conductances g_{exc} and g_{inh} depend on the inputs received from the excitatory and inhibitory synapses, respectively.

The decision was made to model synaptic conductances as exponential:

$$g_{exc}(t) = \begin{cases} 0 & , \ t < t_0 \\ G_{exc} \cdot e^{-(t-t_0)/\tau_{exc}} & , \ t \geq t_0 \end{cases}$$

$$g_{inh}(t) = \begin{cases} 0 & , \ t < t_0 \\ G_{inh} \cdot e^{-(t-t_0)/\tau_{inh}} & , \ t \geq t_0 \end{cases}, \tag{4.1}$$

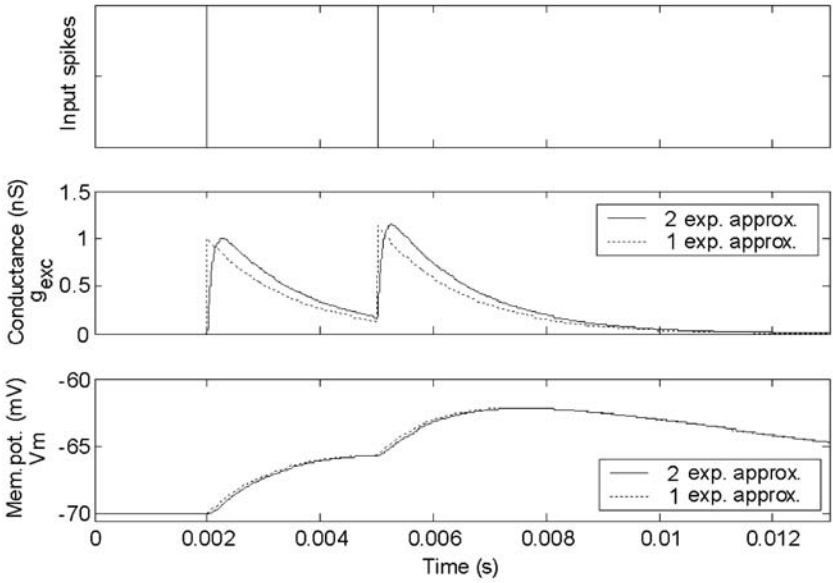


Figure 4: A postsynaptic neuron receives two consecutive input spikes (top). The evolution of the synaptic conductance is the middle plot. The two excitatory postsynaptic potentials (EPSPs) caused by the two input spikes are shown in the bottom plot. In the solid line plots, the synaptic conductance transient is represented by a double-exponential expression (one exponential for the rising phase, one for the decay phase). In the dashed line plot, the synaptic conductance is approximated by a single-exponential expression. The EPSPs produced with the different conductance waveforms are almost identical.

where G_{exc} and G_{inh} represent the peak individual synaptic conductances and g_{exc} and g_{inh} represent the total synaptic conductance of the neuron. This exponential representation has numerous advantages. First, it is an effective representation of realistic synaptic conductances. Thus, the improvement in accuracy from the next most complex representation, a double-exponential function, is hardly worthwhile when considering the membrane potential waveform (see Figure 4).

Second, the exponential conductance requires only a single state variable, because different synaptic inputs can simply be summed recursively when updating the total conductance:

$$g_{exc}(t) = G_{exc,j} + e^{-(t_{currentspike} - t_{previousspike})} g_{exc\_previous}(t). \tag{4.2}$$

($G_{exc,j}$ is the weight of synapse j; a similar relation holds for inhibitory synapses). Most other representations would require additional state

Table 1: Excitatory and Inhibitory Synaptic Characteristics, Based on the Cerebellar Granule Cell.

Excitatory synapse	Maximum conductance (G_{exc_max}) nS, 0–7.5	Time constant (τ_{exc}) ms, 0.5	Reversal potential (E_{exc}) mV, 0
Inhibitory synapse	Maximum conductance (G_{inh_max}) nS: 0–29.8	Time constant (τ_{inh}) ms: 10	Reversal potential (E_{inh}) mV: −80

Note: The first column is an estimation of the maximum cell conductance (summed over all synapses on the cell). The conductances of individual synapses (G_{exc} and G_{inh}) are not included in this table, as they depend on the connection strengths and are therefore provided through the network definition process and synaptic plasticity.

variables or storage of spike time lists, so the exponential representation is particularly efficient in terms of memory use.

In our simulations, the synaptic parameters have been chosen to represent excitatory AMPA-receptor-mediated conductances and inhibitory GABAergic conductances of cerebellar granule cells (Silver, Colquhoun, Cull-Candy, & Edmonds, 1996; Nusser, Cull-Candy, & Farrant, 1997; Tia, Wang, Kotchabhakdi, & Vicini, 1996; Rossi & Hamann, 1998). These are summarized in Table 1. Note that different synaptic connections in different cells might have quite distinct parameters; extreme examples in the cerebellum include the climbing fiber input to Purkinje cells and the mossy fiber input to unipolar brush cell synapses.

The differential equation 4.2 describes the membrane potential evolution (for $t \geq t_0$) in terms of the excitatory and inhibitory conductances at t_0, combined with the resting conductance,

$$C_m \frac{dV_m}{dt} = g_{exc}(t_0) e^{-(t-t_0)/\tau exc} (E_{exc} - V_m) + g_{inh}(t_0)\, e^{-(t-t_0)/\tau_{inh}} (E_{inh} - V_m) + G_{rest} (E_{rest} - V_m), \quad (4.3)$$

where the conductances $g_{exc}(t_0)$ and $g_{inh}(t_0)$ integrate all the contributions received through individual synapses. Each time a new spike is received, the total excitatory and inhibitory conductances are updated as per expression 4.2. Equation 4.3 is amenable to numerical integration. In this way, we can calculate V_m, g_{exc}, g_{inh}, and firing time t_f for given time intervals after the previous input spike. t_f is the time when the membrane potential would reach the firing threshold (V_{th}) in the absence of further stimuli (if indeed the neuron would fire).

5 Table Calculation and Optimization Strategies

The expressions given in section 3 are used to generate the lookup tables that characterize each cell type, with each cell model requiring four tables:

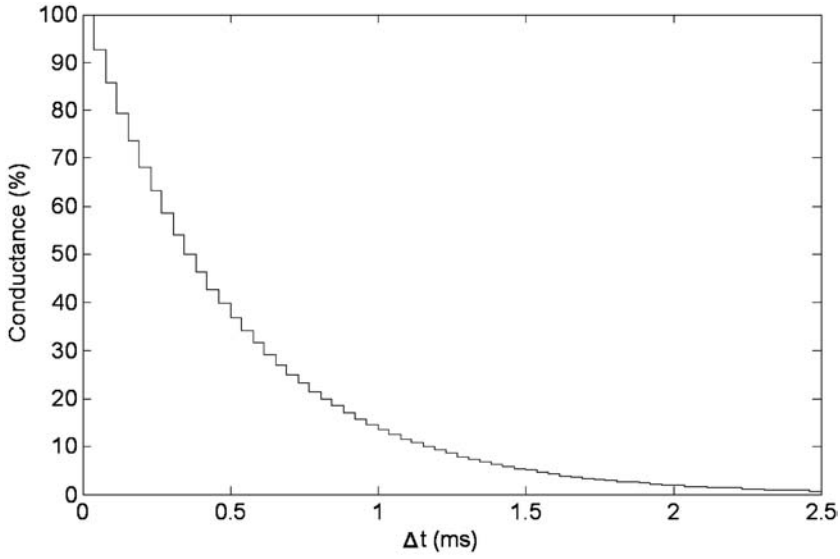


Figure 5: $f_g(\Delta t)$, the percentage conductance remaining after a time (Δt) has elapsed since the last spike was received. This is a lookup table for the normalized exponential function. The time constant of the excitatory synaptic conductance g_{exc} (shown here) was 0.5 ms and for $g_{inh}(t)$, 10 ms. Since the curve exhibits no abrupt changes in the time interval [0, 0.0375] seconds, only 64 values were used.

- **Conductances:** $g_{exc}(\Delta t)$ and $g_{inh}(\Delta t)$ are one-dimensional tables that contain the fractional conductance values as functions of the time Δt elapsed since the previous spike.
- **Firing time:** $t_f\,(V_{m,t_0},\ g_{exc,t_0},\ g_{inh,t_0})$ is a three-dimensional table representing the firing time prediction in the absence of further stimuli.
- **Membrane potential:** $V_m\,(V_{m,t_0},\ g_{exc,t_0},\ g_{inh,t_0},\ \Delta t)$ is a four-dimensional table that stores the membrane potential as a function of the variables at the last time that the neuron state was updated and the elapsed time Δt.

Figures 5, 6, and 7 show some examples of the contents of these tables for a model of the cerebellar granule cell with the following parameters: $C_m = 2pF$, $\tau_{exc} = 0.5$ ms, $\tau_{inh} = 10$ ms, $g_{rest} = 0.2$ nS, $E_{exc} = 0$ V, $E_{inh} = -80$ mV, $E_{rest} = -70$ mV, and $V_{th} = -70$ mV.

The sizes of the lookup tables do not significantly affect the processing speed, assuming they reside in main memory (i.e., they are too large for processor cache but small enough not to be swapped to disk). However, their size and structure obviously influence the accuracy with which the

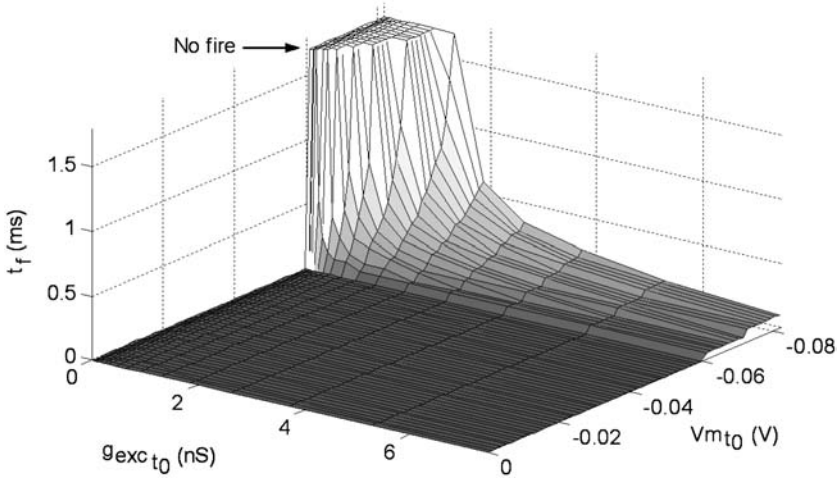


Figure 6: Firing time (t_f) plotted against g_{exc} and initial V_m. t_f decreases as the excitatory conductance increases and as V_{m,t_0} approaches threshold. $g_{inh} = 0$.

neural characteristic functions are represented. The achievable table sizes (in particular, the membrane potential table) are limited by memory resources. However, it is possible to optimize storage requirements by adapting the way in which their various dimensions are sampled. Such optimization can be quite effective, because some of the table functions change rapidly only over small domains. We evaluate two strategies: multiresolution sampling and logarithmic compression along certain axes. Different approaches for the membrane potential function V_m (V_{m,t_0}, g_{exc,t_0}, g_{inh,t_0}, Δt), the largest table, with respect to the inhibitory conductance (g_{inh,t_0}) are illustrated in Figure 8. It can be seen that a logarithmic sampling strategy in the conductance dimensions is an effective choice for improving the accuracy of the representation of neural dynamics. For the following simulation, we have used logarithmic sampling in the g_{inh} and g_{exc} dimensions of the V_m table (as illustrated in Figure 8C).

Storage requirements and calculation time are dominated by the largest table—that for V_m. We shall show in the next section that a table containing about 1 million data points (dimension sizes: $\Delta t = 64$, $g_{exc} = 16$, $g_{inh} = 16$, $V_{m,to} = 64$) gives reasonable accuracy. In order to populate this table, we solve numerically equation 4.3. This was done using a Runge-Kutta method with Richardson extrapolation and adaptive step size control. On a standard 1.8 GHz Pentium platform, calculation of this table takes about 12 s. The firing time table had the same dimensions for g_{exc}, g_{inh}, and $V_{m,to}$. As stated previously, the individual conductance lookup tables had 64 elements each.

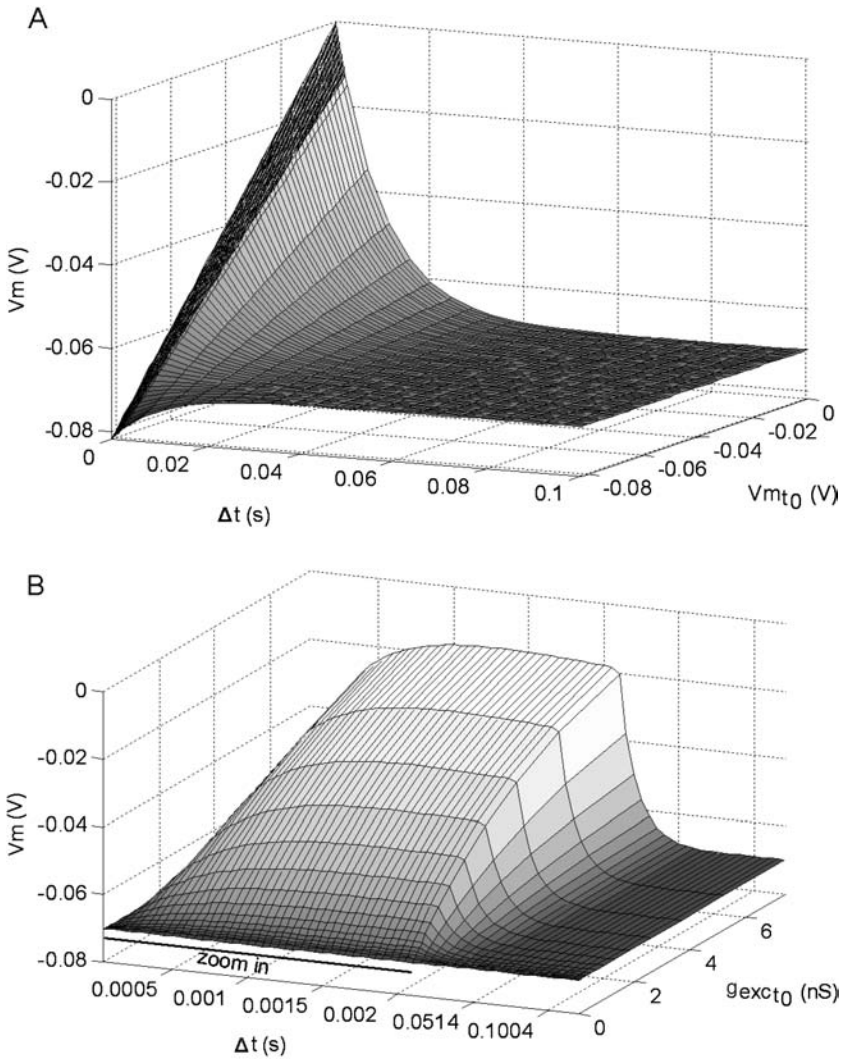
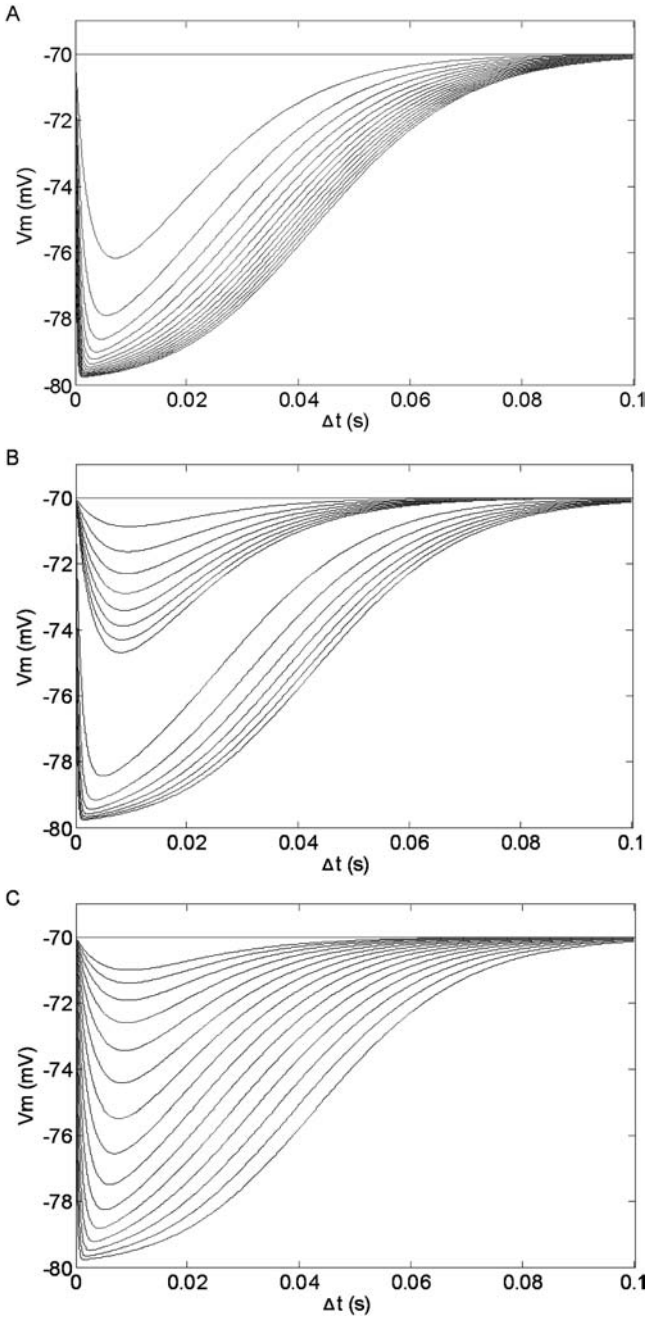


Figure 7: Membrane potential $V_m\left(V_{m,t_0},\ g_{exc,t_0},\ g_{inh,t_0},\ \Delta t\right)$ plotted as a function of (A) V_{m,t_0} and $\Delta t (g_{exc} = g_{inh} = 0)$; (B) g_{exc,t_0} and Δt $(g_{inh} = 0, V_{m,t_0} = E_{rest} = -70$ mV). The zoom in the Δt axis of plot B highlights the fact that the membrane potential change after receiving a spike is not instantaneous.

In principle, these tables could also be based on electrophysiological recordings. Since one of the dimensions of the tables is the time, the experimenter would need to set up only the initial values of g_{exc}, g_{inh}, and V_m and then record the membrane potential evolution following this


A
-70
-72
-74
-76
-78
-80
Vm (mV)
0
0.02
0.04
0.06
0.08
0.1
Δt (s)
B
-70
-72
-74
-76
-78
-80
Vm (mV)
0
0.02
0.04
0.06
0.08
0.1
Δt (s)
C
-70
-72
-74
-76
-78
-80
Vm (mV)
0
0.02
0.04
0.06
0.08
0.1
Δt (s)

initial condition. With our standard table size, the experimenter would need to measure neuronal behavior for $64 \times 16 \times 16$ (G_{exc}, G_{inh}, V_m) triplets. If neural behavior is recorded in sweeps of 0.5 second (at least 10 membrane time constants), only 136 minutes of recording would be required, which is feasible (see below for ways to optimize these recordings). Characterization tables of higher resolution would require longer recording times, but such tables could be built up by pooling or averaging recordings from several cells. Moreover, since the membrane potential functions are quite smooth, interpolation techniques would allow the use of smaller, easier-to-compile tables.

In order to control the synaptic conductances (g_{exc} and g_{inh}), it would be necessary to use the dynamic clamp method (Prinz, Abbott, & Marder, 2004). With this technique, it is possible to replay accurately the required excitatory and inhibitory conductances. It would not be feasible to control real synaptic conductances, though prior determination of their properties would be used to design the dynamic clamp protocols. Dynamic clamp would most accurately represent synaptic conductances in small, electrically compact neurons (such as the cerebellar granule cells modeled here). Synaptic noise might distort the recordings, in which case it could be blocked pharmacologically. Any deleterious effects of dialyazing the cell via the patch pipette could be prevented by using the perforated patch technique (Horn & Marty, 1988), which increases the lifetime of the recording and ensures that the neuron maintains its physiological characteristics.

6 Simulation Accuracy

An illustrative simulation is shown in Figure 9. A single cell with the characteristics of a cerebellar granule cell receives excitatory and inhibitory spikes (upper plots). We can see how the membrane conductances change abruptly due to the presynaptic spikes. The conductance tables emulate the excitatory AMPA-receptor-mediated and the inhibitory GABAergic synaptic inputs (the inhibitory inputs have a longer time constant). The conductance transients (excitatory and inhibitory) are also shown. The bottom plot shows a comparison between the event-driven simulation scheme, which updates the membrane potential at each input spike (these updates are

Figure 8: Each panel shows 16 V_m relaxations with different values of g_{inh,t_0}. The sampled conductance interval is $g_{inh,t_0} \in$ [0,20]nS. (A) Linear approach: [0,20]nS was sampled with a constant intersample distance. (B) Multiresolution approach: two intervals [0,0.35]nS and [0.4,20]nS with eight traces each were used. (C) Logarithmic approach: g_{inh,t_0} was sampled logarithmically.

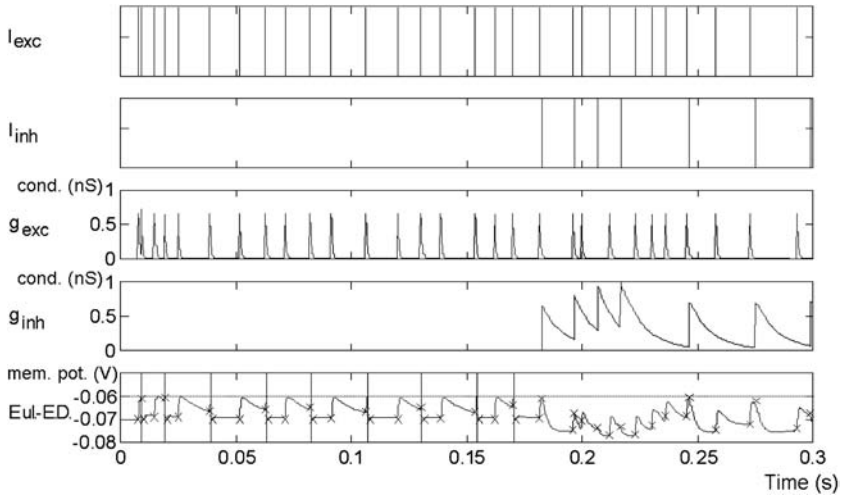


Figure 9: Single neuron simulation. Excitatory and inhibitory spikes are indicated on the upper plots. Excitatory and inhibitory conductance transients are plotted in the middle plots. The bottom plot is a comparison between the neural model simulated with iterative numerical calculation (continuous trace) and the event-driven scheme, in which the membrane potential is updated only when an input spike is received (marked with an x).

marked with an x) and the results of an iterative numerical calculation (Euler method with a time step of 0.5 μs). This plot also includes the output spikes produced when the membrane potential reaches the firing threshold. The output spikes are not coincident with input events, although this is obscured by the timescale of the figure. It is important to note that the output spikes produced by the event-driven scheme are coincident with those of the Euler simulation (they superimpose in the bottom plot). Each time a neuron receives an input spike, both its membrane potential and the predicted firing time of the cell are updated. This occurs only rarely, as the spacing of the events in the event-driven simulation illustrates.

It is difficult to estimate the appropriate size of the tables for a given accuracy. One of the goals of this simulation scheme is to be able to simulate accurately large populations of neurons, faithfully reproducing phenomena such as temporal coding and synchronization processes. Therefore, we are interested in reproducing the exact timing of the spikes emitted. In order to evaluate this, we need a way to quantify the difference between two spike trains. We used the van Rossum (2001) distance between two spike trains. This is related to the distance introduced by Victor and Purpura (1996, 1997), but is easier to calculate, with expression 6.1 and has a more

natural physiological interpretation (van Rossum, 2001):

$$D^2(f,g)_{t_c} = \frac{1}{t_c}\int_0^{\infty} [f(t) - g(t)]^2\, dt \tag{6.1}$$

$$f(t) = \sum_i^M H(t - t_i) e^{-(t-t_i)/t_c}. \tag{6.2}$$

In expression 6.2, H is the Heaviside step function ($H(x) = 0$ if $x < 0$ and $H(x) = 1$ if $x \geq 0$) and M is the number of events in the spike train. In expression 6.1, the distance D is calculated as the integration of the difference between f and g, which are spike-driven functions with exponential terms, as indicated in expression 6.2. Note that the resulting distance and, indeed, its interpretation, depends on the exponential decay constant, t_c in expression 6.2, whose choice is arbitrary (van Rossum, 2001). We used $t_c = 10$ ms. The distance also depends on the number of spikes in the trains. For this reason, we have chosen to report a crudely normalized version $D^2(f,g)_{tc}/M$. Two trains differing only by the addition or removal of a single spike have a normalized distance of ($1/2M$). Two trains differing only by the relative displacement of one spike by δt have a normalized distance of $(1 - \exp(-|\delta t|/t_c))/M$.

In order to evaluate the accuracy of the ED-LUT method and evaluate the influence of table size, we computed the neural model using iterative calculations and the ED-LUT method and then calculated the distance between the output spike trains produced by the two methods.

Figure 10 illustrates how the accuracy of the event-driven approach depends on the synaptic weights of each spike in an example using a Poisson input spike train. We plot as a function of synaptic weight the normalized van Rossum distance between the output spike trains calculated with the Euler method and obtained with ED-LUT. Spikes with very low weights do not generate output events (in either the event-driven scheme or the numerical computation one). Conversely, spikes with very large weights will always generate output events. Therefore, the deviation between the event-driven and the numerical approach will be low in both cases. However, there is an interval of weights in which the errors are appreciable, because the membrane potential spends more time near threshold and small errors can cause the neuron to fire or not to fire erroneously. In general, however, a neuron will have a spread of synaptic weights and is unlikely to show such a pronounced error peak. Action potential variability in subthreshold states is also seen in biological recordings (Stern, Kincaid, & Wilson, 1997); therefore, a certain level of error may be affordable at a network scale.

The accuracy of the event-driven scheme depends on the sampling resolution of the different axes in the tables. We varied the resolution of various

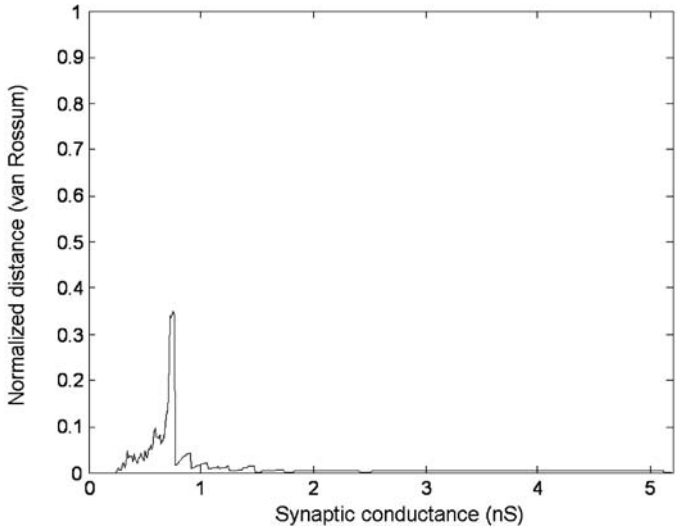


Figure 10: The accuracy of the event-driven simulation depends on the weights of the synapses, with maximal error (normalized van Rossum distance) occurring over a small interval of critical conductances. All synaptic weights were equal.

parameters and quantified the normalized van Rossum distance of the spike trains produced, with respect to the "correct" output train obtained from an iterative solution. The axes of the V_m and t_f table were varied together, but the conductance lookup tables were not modified. Effective synaptic weights were drawn at random from an interval of [0.5, 2] nS, thus covering the critical interval illustrated in Figure 10. From Figure 11 we see that the accuracy of Δt and g_{exc} is critical, but the accuracy of the event-driven scheme becomes more stable when table dimensions are above 1000 K samples. Therefore, we consider appropriate resolution values are the following: 16 values for g_{exc,t_0} and g_{inh,t_0}, 64 values for Δt, and 64 values for V_{m,t_0}. These dimensions will be used henceforth.

Illustrative output spike trains for different table sizes, as well as the reference train, are shown in Figure 12. The spike trains obtained with the iterative method and the event-driven scheme are very similar for the large table with increased resolution in Δt. A spurious spike difference is observed in the other simulations. Doubling the resolution in dimensions other than Δt does not increase the accuracy in this particular simulation. We can also see how the spike train obtained with the small tables is significantly different. This is consistent with the accuracy study results shown in Figure 11.

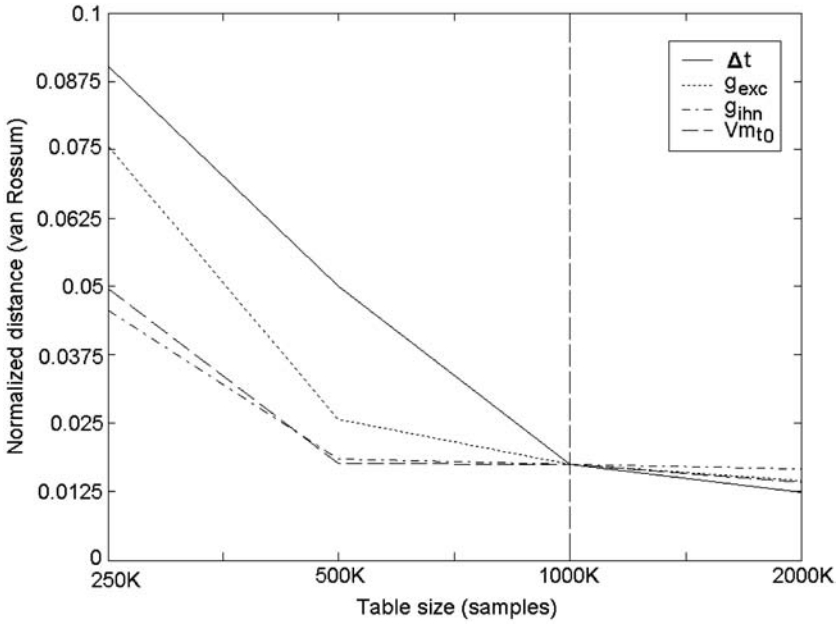


Figure 11: The accuracy of the event-driven approach depends on the resolution of the different dimensions and therefore on the table sizes. To evaluate the influence of table size on accuracy, we ran the simulations with different table sizes. For this purpose, we chose an initial V_m table of 1000 K samples (64 values for Δt, 16 values for g_{exc,t_0}, 16 values for g_{inh,t_0}, and 64 values for V_{m,t_0}). We then halved the size of individual dimensions, obtaining tables of size 500 K samples and 250 K samples from the original table of 1000 K samples. Finally, we doubled the sampling density of individual dimensions to obtain the largest tables of 2000 K samples. For each accuracy estimation, we used an input train of 100 excitatory and 33 inhibitory spikes generating 26 output spikes (when simulated with iterative methods and very high temporal resolution).

7 Simulation Performance and Comparisons with Other Methods

With ED-LUT as described, the simulation time is essentially independent of the network size, depending principally on the rate of events that need to be processed. In other words, the simulation time depends on the network activity, as illustrated in Figure 13.

This implementation allows, for instance, the simulation of $8 \cdot 10^4$ neurons in real time with an average firing rate of 10 Hz on a 1.8 GHz Pentium IV platform. This implies the computation at a rate of $8 \cdot 10^5$ spikes per second as illustrated in Figure 13. Large numbers of synaptic connections of single neurons are efficiently managed by the two-stage strategy described

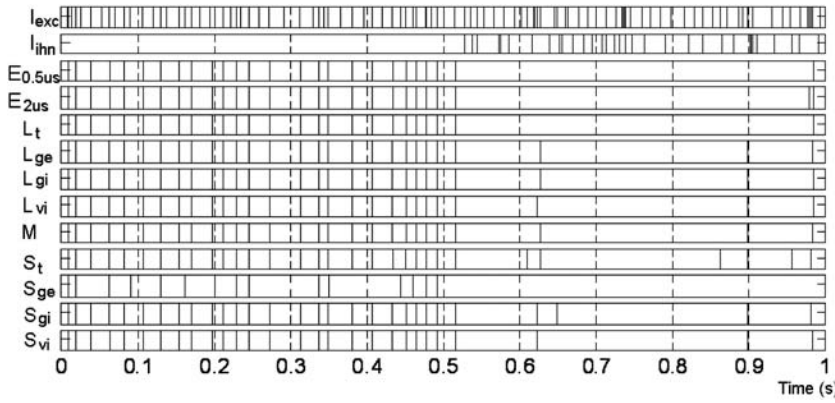


Figure 12: Output spike trains for different table sizes. The first two plots represent the excitatory and inhibitory spikes. The E plots are the output events obtained with numerical iterative methods with different time step resolutions (Euler method with 0.5 μs and with 2 μs). The other plots represent the outputs generated with the event-driven scheme using different table sizes: small (S) of 500 K elements, medium (M) of 1000 K elements, and large (L) of 2000 K elements. The subscripts indicate which dimension resolution has been doubled (or halved) from the medium (M) size table.

in Figure 2. The size of the event queue is affordable, even in simulations with neurons with several thousands of synapses each.

The number of synapses that the simulation engine is able to handle is limited by memory resources. Each neuron requires 60 bytes and each synapse 52 bytes. Therefore, a simulation of $8 \cdot 10^5$ neurons consumes about 46 MB and a total of $62 \cdot 10^6$ connections consumes about 3 GB.

In order to illustrate the potential of the ED-LUT method, we have compared the performance of this computation scheme with other methods (see Table 2). We have implemented three alternative strategies:

- Time-driven iterative algorithm with a fixed time step (TD-FTS). We have used the Runge-Kutta method with a fixed time step.
- Time-driven iterative algorithm with variable time step (TD-VTS). We use the Runge-Kutta method with step doubling and the Richardson extrapolation technique (Cartwright & Piro, 1992). In this case, the computational accuracy is controlled by defining the error tolerance. In this scheme, the iterative computations are done with time step sizes that depend on the smoothness of the function. If a calculation leads to an error estimation above the error tolerance, the time step is reduced. If the error estimation is below this threshold, the time step is doubled. This scheme is expected to be fast when only smooth changes occur in the neuronal states (between input spikes). Although

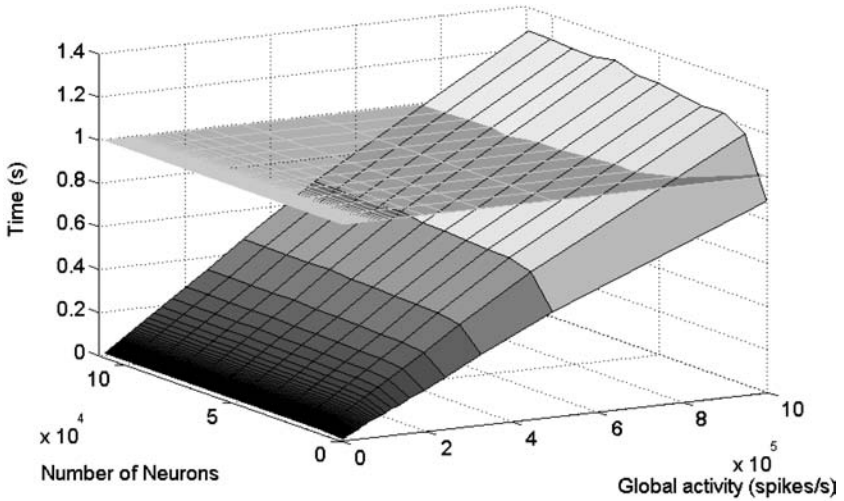


Figure 13: The time taken to simulate 1 second of network activity on a Pentium IV (1.8 GHz) platform. Global activity represents the total number of spikes per second in the network. The network size did not have a significant impact on the time required. The time was almost linear with respect to network activity. The horizontal grid represents the real-time simulation limit—1 second of simulation requiring 1 second of computation.

this method is time driven, its computation speed depends on the cell input in the sense that the simulation passes quickly through time intervals without input activity, and when an input spike is received, the computation approach reduces the time step to simulate accurately the transient behavior of the cell. A similar simulation scheme with either global or independent variable time step integration has been adopted in NEURON (Hines & Carnevale, 2001; Lytton & Hines, 2005).

- Pseudoanalytical approximation (PAA) method. In this case we have approximated the solution of the differential equations that govern the cell. In this way we can adopt an event-driven scheme similar to that proposed in Makino (2003) and Mattia and Del Giudice (2000), in which the neuron behavior is described with analytical expressions. As in Makino (2003), the membrane potential is calculated with the analytical expression, and the firing time is calculated using an iterative method based on Newton-Raphson. Since the differential equations defining the cell behavior of our model have no analytical solution, we need to approximate a four-dimensional function. Even using advanced mathematical tools, this represents a hard task. The accuracy of this approach depends significantly on how good this

Table 2: Performance Evaluation of Different Methods: Accuracy versus Computing Time Trade-Off.

		Normalized van Rossum Distance	Computing Time (s)
	Time step (s)		
Time driven with	$56 \cdot 10^{-5}$	0.061	0.286
fixed time step	$43 \cdot 10^{-5}$	0.033	0.363
(TD-FTS)	$34 \cdot 10^{-5}$	0.017	0.462
	Error tolerance		
Time driven with	$68 \cdot 10^{-5}$	0.061	0.209
variable time step	$18 \cdot 10^{-5}$	0.032	0.275
(TD-VTS)	$2 \cdot 10^{-5}$	0.017	0.440
Pseudoanalytical approximation method (PAA)		0.131	0.142
	Table size (10^6 samples)		
Lookup-table-based	1.05	0.061	0.0066
event-driven	6.29	0.032	0.0074
scheme (ED-LUT)	39.32	0.017	0.0085

Note: We have focused on the computation of a single neuron with an input spike train composed of 100 seconds of excitatory and inhibitory input spikes (average input rate 200 Hz) and 100 seconds of only excitatory input spikes (average input rate 10 Hz). Both spike trains had a standard deviation of 0.2 in the input rate and random weights (uniform distribution) in the interval [0,0.8] nS for the excitatory inputs and [0,1] nS for the inhibitory inputs.

approximation is. In order to illustrate the complexity of the complete cell behavior, it is worth mentioning that the expression used was composed of 15 exponential functions. As shown in Table 2, even this complex approximation does not provide sufficient accuracy, but we have nevertheless used it in order to estimate the computation time of this event-driven scheme.

- Event driven based on lookup tables (ED-LUT). This is our approach, in which the transient response of the cell and the firing time of the predicted events are computed off-line and stored in lookup tables. During the simulations, each neuronal state update is performed by taking the appropriate value from these supporting tables.

In order to determine the accuracy of the results, we obtained the "correct" output spike train using a time-driven scheme with a very short time step. The accuracy of each method was then estimated by calculating the van Rossum distance (van Rossum, 2001) between the obtained result and "correct" spike train.

In all methods except the pseudoanalytical approach, the accuracy versus computation time trade-off is managed with a single parameter (time step

in TD-FTS, error tolerance in TD-VTS, and table size in ED-LUT). We have chosen three values for these parameters that facilitate the comparison between different methods (i.e., values that lead to similar accuracy values). It is worth mentioning that all methods except the time-driven with fixed time step require a computation time that depends on the activity of the network.

Table 2 illustrates several points:

- The computing time using tables (ED-LUT) of very different sizes is only slightly affected by the memory resource management units.
- The event-driven method based on analytical expressions is more than an order of magnitude slower than ED-LUT (and has greater error). This is caused by the complexity of the analytical expression and the calculation of the firing time using the membrane potential expression and applying the Newton-Raphson method.
- The ED-LUT method is about 50 times faster than the time-driven schemes (with an input average activity of 105 Hz).

8 Discussion

We have implemented an event-driven network simulation scheme based on precalculated neural characterization tables. The use of such tables offers flexibility in the design of cell models while enabling rapid simulations of large-scale networks. The main limitation of the technique arises from the size of the tables for more complex neuronal models.

The aim of our method is to enable simulation of neural structures of reasonable size, based on cells whose characteristics cannot be described by simple analytical expressions. This is achieved by defining the neural dynamics using precalculated traces of their internal variables. The proposed scheme efficiently splits the computational load into two different stages:

- Off-line neuronal model characterization. This preliminary stage requires a systematic numerical calculation of the cell model in different conditions to scan its dynamics. The goal of this stage is to build up the neural characterization tables. This can be done by means of a large numerical calculation and the use of detailed neural simulators such as NEURON (Hines & Carnevale, 1997) or GENESIS (Bower & Beeman, 1998). In principle, this could even be done by compiling electrophysiological recordings (as described in section 6).
- Online event-driven simulation. The computation of the simulation process jumps from one event to the next, updating the neuron states according to precalculated neuron characterization tables and efficiently managing newly produced events.

The proposed scheme represents a simulation tool that is intermediate between the very detailed simulators, such as NEURON (Hines & Carnevale, 1997) or GENESIS (Bower & Beeman, 1998), and the event-driven simulation schemes based on simple analytically described cell dynamics (Delorme et al., 1999; Delorme & Thorpe, 2003). The proposed scheme is able to capture cell dynamics from detailed simulators and accelerate the simulation of large-scale neural structures. The approach as implemented here allows the simulation of $8 \cdot 10^4$ neurons with up to $6 \cdot 10^7$ connections in real time with an average firing rate of 10 Hz on a 1.8 GHz Pentium IV platform.

It is difficult to make a precise performance comparison between our method and previous event-driven methods, since they are based on different neuron models. Nevertheless, we have evaluated different computational strategies to illustrate the potential of our approach (see section 7). Mattia and Del Giudice (2000) used a cell model whose dynamics are defined by simple analytical expressions, and Reutimann et al. (2003) extended this approach by including stochastic dynamic. They avoided numerical methods by using precalculated lookup tables. In this case, provided that the reordering event structure is kept of reasonable size (in those approaches, large, divergent connection trees may overload the spike reordering structure), the computation speed of these schemes is likely to be comparable to our approach, since the evaluation of a simple analytical expression and a lookup table consultation consume approximately the same time.

The method has been applied to simulations containing one-compartment cell models with exponential synaptic conductances (with different time constants) approximating excitatory AMPA receptor-mediated and GABAergic inhibitory synaptic inputs. The inclusion of new mechanisms, such as voltage-dependent channels, is possible. However it would require the inclusion of new neural variables and thus new table dimensions. Although very complex models may eventually require lookup tables that exceed current memory capacities, we have shown how even a modest number of table dimensions can suffice to represent quite realistic neuronal models. We have also evaluated several strategies for compressing the tables in order to accommodate more complex models. Furthermore, in appendix C, the proposed table-based methodology is used to simulate the Hodgkin and Huxley model (1952).

The event-driven scheme could be used for multicompartment neuron models, although each compartment imposes a requirement for additional (one to three) dimensions in the largest lookup table. There are two ways in which multicompartment neurons may be partially or approximately represented in this scheme. After preliminary studies, using suitable sampling schemes in order to achieve reasonable accuracy with a restricted table size, we can manage lookup tables of reasonable accuracy with more than seven dimensions. Therefore, we can add two extra dimensions to enable two-compartment simulations. Quite rich cellular behavior could

be supplied by this extension. More concretely, we plan the addition of a second electrical compartment containing an inhibitory conductance. This new compartment will represent the soma of a neuron, while the original compartment (containing both excitatory and inhibitory conductances) will represent the dendrites. The somatic voltage and inhibitory conductance require two additional dimensions in the lookup table. With this model, it would be possible to separate somatic and dendritic processing, as occurs in hippocampal and cortical pyramidal cells, and implement the differential functions of somatic and dendritic inhibition (Pouille & Scanziani, 2001, 2004) (note that most neurons do not receive excitation to the soma).

If individual dendrites can be active and have independent computational functions (this is currently an open question), it may be possible to approximate the dendrites and soma of a neuron as a kind of two-layer network (Poirazi, Brannon, & Mel, 2003), in which dendrites are actually represented in a manner similar to individual cells, with spikes that are routed to the soma (another cell) in the standard manner.

We have embedded spike-driven synaptic plasticity mechanisms (see appendix B) in the event-driven simulation scheme. For this purpose, we have implemented learning rules approximated by exponential terms that can be computed recursively using an intermediate variable. Short-term dynamics (Mattia & Del Guidice, 2000) are also easy to include in the simulations. They are considered important in the support of internal stimulus representation (Amit, 1995; Amit & Brunel, 1997a, 1997b) and learning.

In summary, we have implemented, optimized, and evaluated an event-driven network simulation scheme based on prior characterization of all neuronal dynamics, allowing simulation of large networks to proceed extremely rapidly by replacing all function evaluations with table lookups. Although very complex neuronal models would require unreasonably large lookup tables, we have shown that careful optimization nevertheless permits quite rich cellular models to be used. We believe ED-LUT will provide a useful addition to available simulation tools.

This software package is currently being evaluated in the context of real-time simulations in four labs in different institutions. We plan to extend its use to other labs in the near future. The software is available on request from the authors. Using this method, neural systems of reasonable complexity are already being simulated in real time, in experiments related to robot control by bio-inspired processing schemes (Boucheny, Carrillo, Ros, & Coenen, 2005).

Appendix A: Event Data Structure

Complex data structures, such as balanced trees, can be used for this purpose, offering good performance for both sorted and random-order input streams. To prevent performance degradation, they optimize their structure after each insertion or deletion. However, this rebalancing process adds

more complexity and additional computational overhead (Karlton, Fuller, Scroggs, & Kaehler, 1976). Insertion and deletion of elements in these structures have a computational cost of $O(\log(N))$, where N is the number of events in the structure.

Another candidate data structure is the "skip list" (Pugh, 1990), but in this instance, the cost of the worst case may not be $O(\log(N))$ because the insertion of an input stream can produce an unbalanced structure. Consequently, the search time for a new insertion may be longer than in the balanced trees. This structure offers optimal performance in searching specific elements. However, this is not needed in our computation scheme as we need to extract only the first element (i.e., the next spike).

Finally, the heap data structure (priority queue) (Aho, Hopcroft, & Ullman, 1974; Chowdhury & Kaykobad, 2001; Cormen, Lierson, & Rivest, 1990) offers a stable computational cost of $O(\log(N))$ in inserting and deleting elements. This is the best option as it does not require more memory resources than the stored data. This is because it can be implemented as an array, while the "balanced trees" and "skip lists" need further pointers or additional memory resources.

For all of these methods, the basic operation of inserting an event costs roughly $O(\log(N))$, where N is the number of events in the event data structure. Clearly, the smaller the data structure, the less time such insertions will take. We explain in section 3 the two-stage event handling process we have implemented in order to minimze event heap size while allowing arbitrary divergences and latencies. Compared to a method using a single-event data structure, we would expect the event insertions to be $O(\log(c))$ quicker, where c is the average divergence (connectivity). In Figure 14, we compare the use of one- and two-stage event handling within our simulation scheme. Although event heap operations represent only part of the total computation time, there is a clear benefit to using the two-stage process. For divergences of up to 10,000, typical for recurrent cortical networks, a better than twofold improvement of total computation time is observed.

Appendix B: Spike-Timing-Dependent Synaptic Plasticity

We have implemented Hebbian-like (Hebb, 1949) spike-driven learning mechanisms (spike-timing-dependent plasticity, STDP). The implementation of such leaning rules is suitable because the simulation scheme is based on the time labels of the different events. Spike-time-dependent learning rules require comparison of the times of presynaptic spikes (propagated events) with postsynaptic spikes (firing event). In principle, this requires the trace of the processed presynaptic spikes during a time interval to be kept in order for them to be accessible if postsynaptic spikes occur. Different definite expressions can be used for the learning rule (Gerstner & Kistler, 2002). The weight change function has been approximated with exponential expressions (see equation B.1) to accommodate the experimental results of

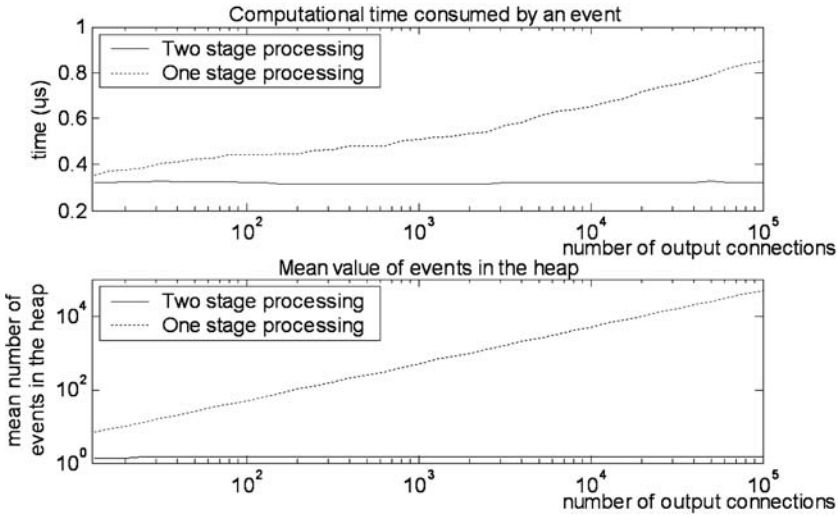


Figure 14: Total computation time for processing an event (top) and size of the event heap (bottom) for one-stage (dashed plot) and two-stage (continuous plot) as functions of synaptic divergence.

Bi and Poo (1998). The computation of this learning rule, by means of exponential terms, facilitates its implementation in a recursive way, avoiding the need to keep track of previous spikes:

$$f(s) = \begin{cases} a_{pre} e^{-b_{pre} s} & if \quad s < 0 \\ a_{post} e^{b_{post} s} & if \quad s > 0 \end{cases}, \tag{B.1}$$

where $s = t^{post} - t^{pre}$ represents the temporal delay between the postsynaptic spike and the presynaptic one. The aim function (Bi & Poo, 1998) can be calculated with expression B.1 using the following parameters ($a_{pre} = 0.935, b_{pre} = -0.075, a_{post} = -0.326, b_{post} = -0.036$). They have been approximated using the Trust-region method (Conn, Gould, & Toint, 2000).

The learning rules are applied each time a cell both receives and fires a spike. Each time a spike from cell i reaches a neuron j, the connection weight (w_{ij}) is changed according to expression B.2, taking into account the time since the last action potential (AP) in the postsynaptic neuron. This time is represented by s in expression B.1:

$$\begin{aligned} &w_{ij} \leftarrow w_{ij} + \Delta w_{ij} \\ &where \\ &\Delta w_{ij} = w_{ij} f(s). \end{aligned} \tag{B.2}$$

Other postsynaptic spikes are not taken into account for the sake of simplicity, but they can be included if necessary.

Each time cell j fires a spike, the learning rule of expression B.3 is applied, taking into account all the presynaptic spikes received in a certain interval:

$$\begin{aligned} &w_{ij} \leftarrow w_{ij} + \Delta w_{ij} \\ &where \\ &\Delta w_{ij} = \sum_k w_{ij} f(s_k). \end{aligned} \tag{B.3}$$

In order to avoid keeping track of all the presynaptic spikes during the learning window, we can rearrange the sum of expression B.3, since the learning rule can be expressed in terms of exponentials B.1.

Each time the neuron fires a spike, the learning rule is applied in each input connection, taking into account the previous spikes received through these inputs. Therefore, each weight changes according to expression B.4:

$$w_{ij} \leftarrow w_{ij} + \sum_{k=1}^{N} w_{ij} f\left(s_k\right) = w_{ij} \left(1 + \sum_{k=1}^{N} a_{pre} e^{b_{pre} s_k}\right), \tag{B.4}$$

where k is iterated over all N presynaptic spikes from cell i received by the neuron j in a time window. This expression can be rearranged as follows:

$$\begin{aligned} w_{ij} &\leftarrow w_{ij} + w_{ij} \left(1 + a_{pre} \left(e^{b_{pre} s_1} \left(1 + e^{b_{pre} s_2} \left(\ldots \left(1 + e^{b_{pre} s_N}\right)\right)\right)\right)\right) \\ w_{ij} &\leftarrow w_{ij} + w_{ij} \left(1 + a_{pre} \left(e^{b_{pre} s_1} + e^{b_{pre} s_1 + b pre s_2} + \ldots + e^{b_{pre} s_1 + \ldots + b_{pre} s_N}\right)\right) \end{aligned} \tag{B.5}$$

This expression can be calculated recursively, accumulating all the multiplicative terms in an intermediate variable $A_{ij,}$ as indicated in expression B.6. s is the time difference between the action potential of cell j and the last presynaptic spike received from cell i:

$$A_{ij} \leftarrow 1 + A_{ij} e^{b_{pre} s}. \tag{B.6}$$

The learning rule is applied recursively as indicated in expression B.7, incorporating the last presynaptic spike. Note that the term A_{ij} accumulates the effect of all previous presynaptic spikes:

$$\begin{aligned} &w_{ij} \leftarrow w_{ij} + \Delta w_{ij} \\ &where \\ &\Delta w_{ij} = w_{ij} a_{pre} \left(e^{b_{pre} s} A_{ij}\right). \end{aligned} \tag{B.7}$$

Table 3: Hodgkin and Huxley Model (1952).

$$\frac{dV_m}{dt} = \left(I - g_K \cdot n^4 \cdot (V_m - V_K) - g_{Ns} \cdot m^3 \cdot h \cdot (V_m - V_{Na}) - g_l\,(V_m - V_l)\right)/C_m$$

$$\frac{dn}{dt} = \phi \cdot (\alpha_n \cdot (1-n) - \beta_n \cdot n)\,;\ \frac{dm}{dt} = \phi \cdot (\alpha_m \cdot (1-m) - \beta_m \cdot m)\,;$$
$$\frac{dh}{dt} = \phi \cdot (\alpha_h \cdot (1-h) - \beta_h \cdot h)$$

$$\alpha_n = \frac{0.01 \cdot V_m + 0.1}{\exp(0.1 \cdot V_m + 1) - 1};\ \alpha_m = \frac{0.1 \cdot V_m + 2.5}{\exp(0.1 \cdot V_m + 2.5) - 1};\ \alpha_h = 0.07 \cdot \exp(0.05 \cdot V_m)$$

$$\beta_n = 0.125 \cdot \exp(V_m/80)\,;\ \beta_m = 4\exp(V_m/18)\,;\ \beta_h = \frac{1}{\exp(0.1 \cdot V_m + 3) + 1}$$

$$\phi = 3^{(T-6.3)/10}$$

$$I = -g_{exc} \cdot (V_m - E_{exc}) - g_{inh} \cdot \left(V_m - E_{inh}\right)$$

$$\frac{dg_{exc}}{dt} = -\frac{g_{exc}}{\tau_{exc}};\ \frac{dg_{inh}}{dt} = -\frac{g_{inh}}{\tau_{inh}}$$

Note: The first expression describes the membrane potential evolution. The differential equations of n, m, and h govern the ionic currents. The last two expressions of the table describe the input-driven currents and synaptic conductances. The parameters are the following: $C_m = 1\mu$ F/cm^2, $g_K = 1$ mS/cm^2, $g_{Na} = 120$ mS/cm^2, $g_l = 0.3$ mS/cm^2, $V_{Na} = -115$ mV, $V_K = 12$ mV, $V_l = -10.613$ mV, and $T = 6.3^oC$. The parameters of the synaptic conductances are the following: $E_{exc} = -65$ mV, $E_{inh} = 15$ mV, $\tau_{exc} = 0.5$ ms, and $\tau_{inh} = 10$ ms.

Appendix C: Hodgkin and Huxley Model

In order to further validate the simulation scheme, we have also compiled into tables the Hodgkin and Huxley model (1952) and evaluated the accuracy obtained with the proposed table-based methodology. Table 3 shows the differential expressions that define the neural model. We have also included expressions for synaptic conductances.

Interfacing the explicit representation of the action potential to the event-handling architecture, which is based on idealized instantaneous action potentials, raises a couple of technical issues. The first is the choice of the precise time point during the action potential that should correspond to the idealized (propagated) event. This choice is arbitrary; we chose the peak of the action potential. The second issue arises from the interaction of this precise time point with discretization errors during updates close to the peak of the action potential. As illustrated in Figure 15, a simple-minded implementation can cause the duplication (or by an analogous mechanism, omission) of action potentials, a significant error. This can happen when an update is triggered by an input arriving just after the peak of the action potential (and thus after the propagated event). Discretization errors can cause the prediction of the peak in the immediate future, equivalent to a very slight shift to the right of the action potential waveform. Since we have identified the propagated event with the peak, a duplicate action potential would be emitted. The frequency of such

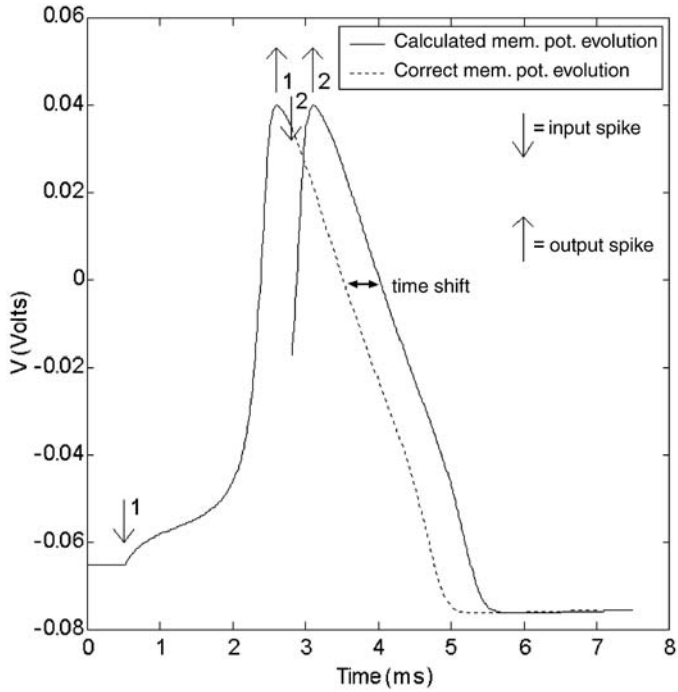


Figure 15: Discretization errors could allow an update shortly following an action potential peak to predict the peak of the action potential in the immediate future, leading to the emission of an erroneous duplicate spike. (The errors have been magnified for illustrative purposes.)

errors depends on the discretization errors and thus the accuracy (size) of the lookup tables and on the frequency of inputs near the action potential peaks. These errors are likely to be quite rare, but as we now explain, they can be prevented.

We now describe one possible solution (which we have implemented) to this problem (see Figure 16). We define a "firing threshold" (θ_f; in practice, -10 mV). This is quite distinct from the physiological threshold, which is much more negative. If the membrane potential exceeds θ_f, we consider that an action potential will be propagated under all conditions. We exploit this assumption by always predicting a propagated event if the membrane potential is greater than θ_f after the update, even if the "present" is after the action potential peak (in this case, emission is immediate). This procedure ensures that no action potentials are omitted, leaving the problem of duplicates.

We also define a postemission time window. This extends from the time of emission (usually the action potential peak) to the time the membrane

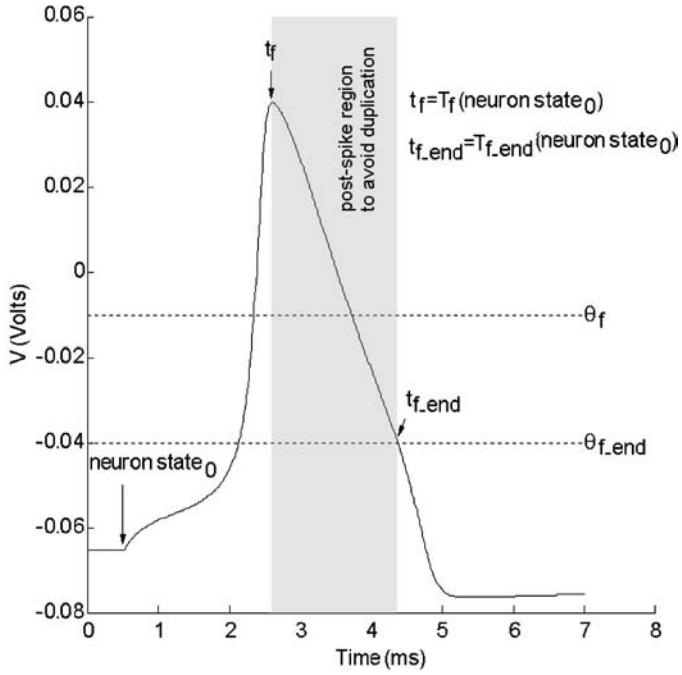


Figure 16: Prevention of erroneous spike omission and duplication. Once the neuron exceeds θ_f, a propagated event is ensured. In this range, updates that cause the action potential peak to be skipped cause immediate emission. This prevents action potential omission. Once the action potential is emitted (usually at t_f), the time t_{f_end} is stored, and no predicted action potential emissions before this time are accepted. This ensures that no spikes are propagated more than once.

potential crosses another threshold voltage, θ_{f_end}. This time, t_{f_end}, is stored in the source neuron when the action potential is emitted. Whenever new inputs are processed, any predicted output event times are compared with t_{f_end} and only those predicted after t_{f_end} are accepted. This procedure eliminates the problem of duplicate action potentials.

In order to preserve the generality of this implementation, we chose to define these windows around the action potential peak by voltage level crossings. In this way, the implementation will adapt automatically to changes of action potential waveform (possibly resulting from parameter changes). This choice entailed the construction of an additional large lookup table. Simpler implementations based on fixed time windows could avoid this requirement. However, the cost of the extra table was easily borne.

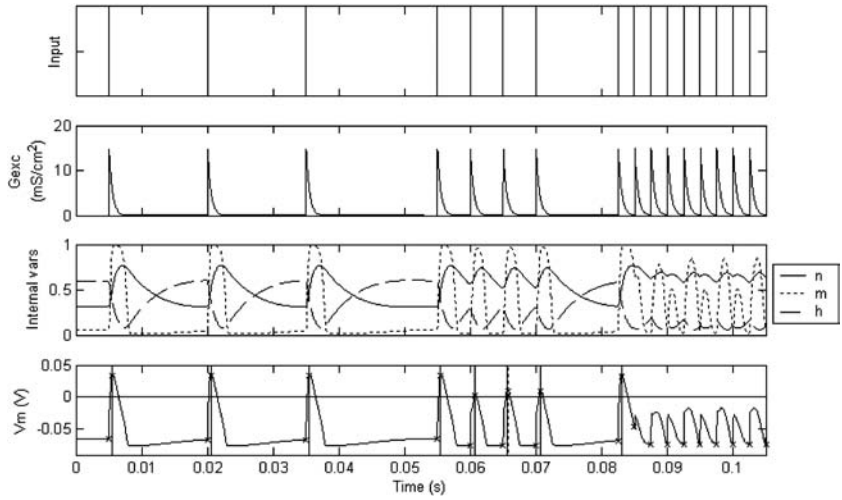


Figure 17: Single neuron event-driven simulation of the Hodgkin and Huxley model. Note that in order to facilitate the comparison of the plots with the ones of the integrate-and-fire model (see Figure 9) the variable (V) has been calculated using the following expression $V = (-V_m - V_{rest})/1000$ with $V_{rest} = 65$ mV.

We have compiled the model into the following tables:

- One table of seven dimensions for the membrane potential, $V_m = f(\Delta t, g_{exc\_0}, g_{inh\_0}, n_0, m_0, h_0, V_0)$.
- Three tables of seven dimensions for the variables driving ionic currents, $n = f(\Delta t, g_{exc\_0}, g_{inh\_0}, n_0, m_0, h_0, V_0)$, $m = f(\Delta t, g_{exc\_0}, g_{inh\_0}, n_0, m_0, h_0, V_0)$, $h = f(\Delta t, g_{exc\_0}, g_{inh\_0}, n_0, m_0, h_0, V_0)$.
- Two tables of two dimensions for the conductances, $g_{exc} = f(\Delta t, g_{exc\_0})$, $g_{inh} = f(\Delta t, g_{inh\_0})$.
- Two tables of six dimensions for the firing prediction, $t_f = f(g_{exc}, g_{inh}, n_0, m_0, h_0, V_0)$ and $t_{f\_end} = f(g_{exc}, g_{inh}, n_0, m_0, h_0, V_0)$. With $\theta_f = -0.01$ V and $\theta_{f\_end} = -0.04$ V.

An accurate simulation of this model (as shown in Figure 17) requires approximately 6.15 Msamples (24.6 MB using 4-byte floating point data representation) for each seven-dimension table. We use a different number of samples for each dimension: $\Delta t(25)$, $g_{exc\_0}(6)$, $g_{inh\_0}(6)$, $n_0(8)$, $m_0(8)$, $h_0(8)$, and $V_0(14)$. The table calculation and compilation stage of this model requires approximately 4 minutes on a Pentium IV 1.8 Ghz.

Figure 17 shows an illustrative simulation of the Hodgkin and Huxley model using the table-based event-driven scheme. Note that the simulation engine is able to accurately jump from one marked instant (bottom plot) to

the next one (according to either input or generated events). The membrane potential evolution shown in the bottom plot has been calculated using a numerical method (continuous plot) and the marks (placed onto the continuous trace) have been calculated using the event-driven approach. We have also included the generated events using numerical calculation (vertical continuous lines) and those generated by the table-based event-driven approach (vertical dashed lines).

In order to evaluate the model accuracy, we have adopted the same methodology described in section 5. We have simulated a single cell receiving an input spike train using numerical calculation to obtain a reference output spike train. Then we have used the proposed table-based event-driven approach to generate another output spike train. As in section 7, the accuracy measurement is obtained calculating the van Rossum (2001) distance between the reference and the event-driven output spike trains. We have used a randomly generated test input spike train of average rate 300 Hz with a standard deviation of 0.7 and a uniform synaptic weight distribution in the interval [0.1,1] mS/cm^2. Using the table sizes mentioned above, the van Rossum distance (with a time constant of 10 ms and the normalization mentioned in section 6) between the reference spike train and that obtained with the proposed method is 0.057 (in the same range as the Rossum distances obtained when comparing other simpler neural models in Table 2). In fact, in order to obtain a similar accuracy using Euler numerical calculation, a time step shorter than 65 μs is required.

Acknowledgments

This work has been supported by the EU projects SpikeFORCE (IST-2001-35271), SENSOPAC (IST-028056) and the Spanish National Grant (DPI-2004-07032). We thank Olivier Coenen, Mike Arnold, Egidio D'Angelo, and Christian Boucheny for their interesting suggestions during this work.

References

Aho, A. V., Hopcroft, J. E., & Ullman, J. D. (1974). *The design and analysis of computer algorithms*. Reading, MA: Addison-Wesley.

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.

Bi, G., & Poo, M. (1998). Synaptic modifications in cultured hippocampal neurons: Dependence on spike timing, synaptic strength, and postsynaptic cell type. *J. Neurosci., 18*, 10464–10472.

Boucheny, C., Carrillo, R., Ros, E., & Coenen, O. J. M. D. (2005). Real-time spiking neural network: An adaptive cerebellar model. *Lecture Notes in Computer Science, 3512*, 136–144.

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

Braitenberg, V., & Atwood, R. P. (1958). Morphological observations on the cerebellar cortex. *J. Comp. Neurol., 109*, 1–33.

Brunel, N., & Hakim, V. (1999). Fast global oscillations in networks of integrate-and-fire neurons with low firing rates. *Neural Computation, 11*, 1621–1671.

Cartwright, J. H. E., & Piro, O. (1992). The dynamics of Runge-Kutta methods. *Int. J. Bifurcation and Chaos, 2*, 427–449.

Chowdhury R. A., & Kaykobad, M. (2001). Sorting using heap structure. *International Journal of Computer Mathematics, 77*, 347–354.

Conn, A. R., Gould, N. I. M., & Toint, P. L. (2000). *Trust-region methods*. Philadelphia: SIAM.

Cormen, T. H., Lierson, C. E., & Rivest, R. L. (1990). *Introduction to algorithms*. Cambridge, MA: MIT Press.

Delorme, A., Gautrais, J., van Rullen, R., & Thorpe, S. (1999). SpikeNET: A simulator for modelling large networks of integrate and fire neurons. *Neurocomputing, 26–27*, 989–996.

Delorme, A., & Thorpe, S. (2003). SpikeNET: An event-driven simulation package for modelling large networks of spiking neurons. *Network: Computation in Neural Systems, 14*, 613–627.

Eckhorn, R., Bauer, R., Jordan, W., Brosh, M., Kruse, W., Munk, M., & Reitböck, H. J. (1988). Coherent oscillations: A mechanism of feature linking in the visual cortex? *Biol. Cyber., 60*, 121–130.

Eckhorn, R., Reitböck, H. J., Arndt, M., & Dicke, D. (1990). Feature linking via synchronization among distributed assemblies: Simulations of results from cat visual cortex. *Neural Computation, 2*, 293–307.

Gerstner, W., & Kistler, W. (2002). *Spiking neuron models: Single neurons, populations, plasticity*. Cambridge: Cambridge University Press.

Hebb, D. O. (1949). *The organization of behavior*. New York: Wiley.

Hines, M. L., & Carnevale, N. T. (1997). The NEURON simulation environment. *Neural Computation, 9*, 1179–1209.

Hodgkin, A. L., & Huxley, A. F. (1952). A quantitative description of membrane current and its application to conduction and excitation in nerve. *Journal of Physiology, 117*, 500–544.

Horn, R., & Marty, A. (1988). Muscarinic activation of ionic currents measured by a new whole-cell recording method. *Journal of General Physiology, 92*, 145–159.

Karlton, P. L., Fuller, S. H., Scroggs, R. E., & Kaehler, E. B. (1976). Performance of height-balanced trees. *Communications of ACM, 19*(1), 23–28.

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

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

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

Mitchell, S. J., & Silver, R. A. (2003). Shunting inhibition modulates neuronal gain during synaptic excitation. *Neuron, 38*, 433–445.

Nusser, Z., Cull-Candy, S., & Farrant, M. (1997). Differences in synaptic GABA(A) receptor number underlie variation in GABA mini amplitude. *Neuron, 19*(3), 697–709.

Poirazi, P., Brannon, T., & Mel, B. W. (2003). Pyramidal neuron as two-layer neural network. *Neuron, 37*(6), 989–999.

Pouille, F., & Scanziani, M. (2001). Enforcement of temporal fidelity in pyramidal cells by somatic feed-forward inhibition. *Science, 293*(5532), 1159–1163.

Pouille, F., & Scanziani, M. (2004). Routing of spike series by dynamic circuits in the hippocampus. *Nature, 429*(6993), 717–723.

Prinz, A. A., Abbott, L. F., & Marder, E. (2004). The dynamic clamp comes of age. *Trends Neurosci., 27*, 218–224.

Pugh, W. (1990). Skip lists: A probabilistic alternative to balanced trees. *Communications of the ACM, 33*(6), 668–676.

Reutimann, J., Giugliano, M., & Fusi, S. (2003). Event-driven simulation of spiking neurons with stochastic dynamics. *Neural Computation, 15*, 811–830.

Rossi, D. J., & Hamann, M. (1998). Spillover-mediated transmission at inhibitory synapses promoted by high affinity alpha6 subunit GABA(A) receptors and glomerular geometry. *Neuron, 20*(4), 783–795.

Silver, R. A., Colquhoun, D., Cull-Candy, S. G., & Edmonds, B. (1996). Deactivation and desensitization of non-NMDA receptors in patches and the time course of EPSCs in rat cerebellar granule cells. *J. Physiol., 493*(1), 167–173.

Stern, E. A., Kincaid, A. E., & Wilson, C. J. (1997). Spontaneous subthreshold membrane potential fluctuations and action potential variability of rat corticostriatal and striatal neurons in vivo. *J. Neurophysiol., 77*, 1697–1715.

Tia, S., Wang, J. F., Kotchabhakdi, N., & Vicini, S. (1996). Developmental changes of inhibitory synaptic currents in cerebellar granule neurons: Role of GABA(A) receptor alpha 6 subunit. *J. Neurosci., 16*(11), 3630–3640.

van Rossum, M. C. W. (2001). A novel spike distance. *Neural Computation, 13*, 751–763.

Victor, J. D., & Purpura, K. P. (1996). Nature and precision of temporal coding in visual cortex: A metric-space analysis. *J. Neurophysiol., 76*, 1310–1326.

Victor, J. D., & Purpura, K. P. (1997). Metric-space analysis of spike trains: Theory, algorithms and application. *Network: Computation in Neural Systems, 8*, 127–164.

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. 967–934). San Mateo, CA: Morgan Kaufmann.

Received November 30, 2004; accepted April 28, 2006.