Simulation scalability of large brain neuronal networks thanks to time asynchrony

Cyrille Mascart[a,1], **Gilles Scarella**[b,a], **Patricia Reynaud-Bouret**[b], **and Alexandre Muzy**[a]

[a]Université Côte d'Azur, CNRS, I3S, France.; [b]Université Côte d'Azur, CNRS, LJAD, France.

This manuscript was compiled on June 11, 2021

We present here a new algorithm based on a random model for simulating efficiently large brain neuronal networks. Model parameters (mean firing rate, number of neurons, synaptic connection probability and postsynaptic duration) are easy to calibrate further on real data experiments. Based on time asynchrony assumption, both computational and memory complexities are proved to be theoretically linear with the number of neurons. These results are experimentally validated by sequential simulations of millions of neurons and billions of synapses in few minutes on a single processor desktop computer.

Brain neuronal networks | Random models | Point processes | Stochastic simulation | Discrete event systems | Time asynchrony | Hawkes processes.

There are more and more vast research projects, whose aim is to simulate brain areas or even complete brains to better understand the way it works. Let us cite for instance: the Human Brain Project (1) in Europe, the Brain Mapping by Integrated Neurotechnologies for Disease Studies (Brain/MINDS) (2) in Japan or the Brain Initiative (3) in the United-States. Several approaches are feasible. There is the biochemical approach (4), which is doomed for systems as complex as the brain. A more biophysical approach has been investigated, see for instance (5), where cortical barrels have been successfully simulated, but are limited to about 10^5 neurons. However, the human brain contains about 10^{11} neurons whereas a small monkey, like marmosets (2), has already 6×10^8 neurons (6) and a bigger monkey, like a macaque, has 6×10^9 neurons (6).

To simulate such huge networks, models reduction have to be made. In particular, a neuron has no more physical shape and is just represented by a point in a network with a certain voltage. Hodgkin-Huxley equations (7) are able to reproduce the physical shape if it is combined to other differential equations, representing the dynamic of ion channels, but the complexity of these coupled equations that form a chaotic system (8), makes the system quite difficult to simulate for huge networks. If ion channels dynamic is neglected, the simplest model of voltage is the Integrate-and-Fire model (9). With such models, it has been possible on supercomputers to simulate a human-scale cerebellar network reaching about 68×10^9 neurons (10).

However there is another point of view, which might allow us to simulate such massive networks with simplified models. Indeed, one can use much more random models to reproduce the essential dynamics of the neurons: their firing pattern. The randomization of not only the connectivity graph but also the dynamics on the graph is making the model closer to the data at hand and explain to a certain extent their variability. The introduction of randomness is not new and has been done in many models including Hodgkin-Huxley (11) and Leaky Integrate-and-fire (LIF for short) (12).

Here we want to focus on particular random models - point processes (13) - which have a particular property: time asynchrony, that is the inability of the model to have two spikes that are produced exactly at the same time by two different neurons. This includes in particular Hawkes models and variants such as GLM, Wold processes, Galves-Löcherbach models, and even some random LIF models with random or soft threshold, all of them having been used to fit real data (13–19).

This property, which is well known in mathematics (14, 20), combined with graph sparsity lead us to propose a new algorithm in (21). The computational complexity of this new algorithm has been computed. Thanks to time asynchrony and to the computational activity tracking of firing neurons (22), we have shown in particular that if the graph is sparse, the complexity cost of the computation of a new point in the system is linear in the number of neurons. However the memory burden was too high to reach networks of 10^8 neurons.

In a preliminary work (23) focusing on the mathematical aspects of mean-field limits of LIFs, we formalized a way to deal with this memory aspect without putting it into practice: the main point is to not keep in memory the whole network, but to regenerate it when need be. Recently, the same idea, under the name of procedural connectivity, has been applied with success on LIF models in (24): using GPU-based parallel programming, and without time asynchrony, the authors have been able to simulate a network of 4×10^6 neurons and 24×10^9 synapses on a desktop GPU computer.

But, as we show in the present article, the gain of procedural connectivity is even huger when combined with time

Significance Statement

Time asynchrony refers to models that prevent any two neurons to fire at the exact same time (up to the numerical precision). Brain models that are time asynchronous have a huge numerical advantage on other ones: they can be easily simulated without huge parallelization or use of GPU. Indeed, this work proposes to exploit the time asynchrony property to derive new simulation algorithms, whose computational and memory complexity can be estimated beforehand. Application on realistic set of parameters show that they can indeed easily reach the size of a small monkey brain, on a usual single processor desktop computer.

Please provide details of author contributions here.

Please declare any competing interests here.

[1]A.O.(Author One) contributed equally to this work with A.T. (Author Two) (remove if not applicable).

[2]To whom correspondence should be addressed. E-mail: author.twoemail.com

asynchrony. Indeed, classical parallel programming usually uses a discrete simulation time and computes for all neurons (or synapses) what happens at each time step in a parallel fashion (even if spikes can be communicated in between two time steps (10)). At each time step, each process corresponding to a different neuron has to wait for the calculations of all the other processes to know what needs to be updated before computing the next step. With time asynchrony, we can leverage discrete-event programming (25–28) to track the whole system in time by jumps: from one spike in the network to another spike in the network. Since a very small percentage of a brain is firing during a given unit of time (29), the gain we have is tremendous in terms of computations. Thanks to the procedural connectivity, the memory needed to access for the computation of each new spike is also controlled. Hence, thanks to procedural connectivity combined with time asynchrony, we propose a new algorithm for time asynchronous models, running sequentially on a single processor, thus simulating a realistic network of 10^8 neurons for which computational complexity as well as memory costs can be controlled beforehand.

Results

Time synchrony for parallel simulation. Brain simulation of large networks is usually done in parallel based on simulation synchronization. Of course, this depends on both the mathematical model at hand and the simulation algorithm. However, for most models, differential equations are used to derive the time course of the membrane voltage for each individual neurons. These equations are (approximately) solved by usual discrete-time numerical schemes (*cf.* Figure 1a) (24).

In this kind of implementation, when one presynaptic neuron fires at a time t, *i.e.*, emits a spike (red dots in Figure 1a), the synaptic transmission to post-synaptic neurons is done at the next time step $t + \Delta t$ (orange dots in Figure 1a). Between two synaptic transmissions, the membrane potential of a neuron evolves independently of the other neurons and can be computed in parallel (green dots in Figure 1a). However, since one does not know when a spike will be emitted in the network in advance, the membrane potential of all the neurons are classically computed in a synchronous way to be able to eventually transmit spikes, at each time step Δt of the algorithm.

Time asynchrony for sequential simulation. As said in the introduction, point processes models of neuronal network may guarantee time asynchrony if they have a stochastic intensity. Such processes include Hawkes processes, GLM approaches, Wold processes or Galvès Löcherbach models in continuous time (13, 15–18). Most of these models have proved their efficiency in terms of goodness-of-fit with respect to real spike train data (13–17, 30). Often, the intensity in these models can be informally interpreted as a function of the membrane voltage and for more evidence, we refer the reader to (31), where the spike train of a motor neuron has been shown to be adequately modeled by a point process whose stochastic intensity is a function of the membrane voltage.

These point processes models differ from classical LIF, mainly because the higher the intensity (or the membrane potential) of a given neuron is, the more likely it is that the neuron fires, but this is never for sure. In classical LIF models, the neurons fire when their membrane potential reaches a fixed threshold. Therefore it may happen that if a presynaptic neuron fires, and if the corresponding postsynaptic neurons have a potential close to the threshold, then all the postsynaptic neurons fire at the exact same time. This phenomenon can be massive (32, 33): this corresponds to the mathematical phenomenon of blow-up, which happens for some mean-field limits of such models. In this case, no time asynchrony is possible but such phenomenon is completely unrealistic from a biological point of view. There are LIF models with random or soft threshold (19, 23) which might not have this problem and which may also satisfy time asynchrony.

In (21), we proposed a discrete-event algorithm to simulate point processes with stochastic intensities. This algorithm is based on the theory of local independence graph (34), which is the directed neuronal network in our present case.

The algorithm works as follows (see Figure 1b). The spike events happen in continuous time in the system (up to the numerical precision). Once a spike on a particular presynaptic neuron happens (red dots in Figure 1b), the postsynaptic neurons are updated (orange dots in Figure 1b). The presynaptic and the post synaptic neurons compute their intensities (assimilated to membrane potential) and forecast its evolution (green arrows in Figure 1b) if nothing in between occurs in the system. They are therefore able to forecast their potential next spike (gray dots in Figure 1b). The algorithm maintains a scheduler containing all potential next spikes on all neurons and decides that the next neuron to fire effectively is the one corresponding to the minimum of these potential next spikes. For more details, we refer to (21).

The gain comes from the fact that neurons that are not firing a lot, do not require a lot of computation either. In particular we do not have to update all neurons at each spike but only the pre and post synaptic neurons that are involved in the spiking event. This is the main difference with the parallel simulation framework detailed above. The other difference is that we can work with arbitrary precision, typically 10^{-15} if necessary, without impeding the time complexity of the algorithm.

Note that the whole algorithm is possible only because two neurons in the network will not spike at the same time: the whole concept is based on time asynchrony to be able to jump from one spike in the system to the next spike in the system. Of course, this is true only up to numerical precision: if two potential next spikes (gray dots on Figure 1b) happen at the exact same time with resolution 10^{-15}, by convention the neuron with the smaller index is said to fire. But the probability of such event is so small that this is not putting the simulation in jeopardy.

Also note that this does not prevent neurons to eventually synchronize frequently over a small time duration of a few milliseconds, as defined for instance in (35) and the references therein.

Procedural connectivity. One of the memory burden of both methods (parallel and time asynchrony) comes from the fact that a classic implementation stores the whole connectivity graph, which is huge for brain scale models.

If the connectivity is the result of a random graph and that each presynaptic neuron is randomly connected to its postsynaptic neurons, one can store the random seed instead of the result of the random attribution. Hence the whole

graph is never stored in full but only regenerated when need be (see Figure 1b). The random connectivity is regenerated at each spike taking advantage of the deterministic nature of the pseudo-random generator used in the simulation. Storing the generator initial seed, the seed of each neuron is computed based on initial seed value and neuron index (see Figure 1c). With this method, only the initial seed is stored in memory. Of course this dynamic regeneration at each spike has a cost in terms of time complexity, but this cost is negligible with respect to the other computations that need to be made and this saves memory.

This method has been evoked at first in (23) for time asynchronous algorithms, without being put into practice, whereas this method has been already implemented with success on parallel programming with GPU (24).

Computational and memory costs. In (21), we obtained an accurate estimate of the complexity of the algorithm without procedural connectivity, for the simulation of linear Hawkes processes (*cf.* Equation 7 of (21)). This can be reused to compute the computational complexity of the same model, when the procedural connectivity step is added. Thus the overall time complexity of our algorithm is of the order of

$$\mathcal{O}\left(T\left[Md^2\bar{m}^2 + \log(M)Md\bar{m} + Md\bar{m}\right]\right) \qquad [1]$$

where M is the number of neurons, p the connection probability, $d = \lceil pM \rceil$ the average degree of the network, $\bar{m}$ the average firing rate of the network and T the simulation duration. The last term in Eq. (1) corresponds to the computational cost of the procedural connectivity at each spike, which is indeed negligible with respect to other terms, as explained before.

Note that such computational costs depend in particular on the intensity shape of the underlying point process model. The linearity of the Hawkes processes makes it easy to derive, whereas this can be much more cumbersome with other models such as stochastic LIF, which needs to compute the distribution of the time at which the threshold is reached.

The maximal memory cost of the procedural connectivity, without the spike times storage, is of the order of $\mathcal{O}(d\omega) = \mathcal{O}(pM\omega)$, with ω the number of bits necessary for representing the index of a post-synaptic neuron.

The memory cost of a static storage of the whole graph is of the order of $\mathcal{O}(dM\omega) = \mathcal{O}(pM^2\omega)$. We do not include in this the memory costs for the storage of each spike of each neuron. However, this cost is the same whatever the method and it is of the order of $MT\bar{m}\epsilon$, where ϵ is the number of bits necessary to represent a spike, which depends on the numerical precision with which time is recorded. If d is thought to be a fixed parameter, the memory cost of our complete algorithm with procedural connectivity is thus

$$\mathcal{O}(d\omega + MT\bar{m}\epsilon) \qquad [2]$$

Conversely, the use of a static storage of the network is $\mathcal{O}(dM\omega + MT\bar{m}\epsilon)$. Note however that depending on what the program needs to return, we might not want to have the whole set of points but only summary statistics such as firing rates that will cost in memory much less than $\mathcal{O}(MT\bar{m}\epsilon)$.

Choice of brain scale parameters. Because of the precision of the actual measurements and the brain region and neuron

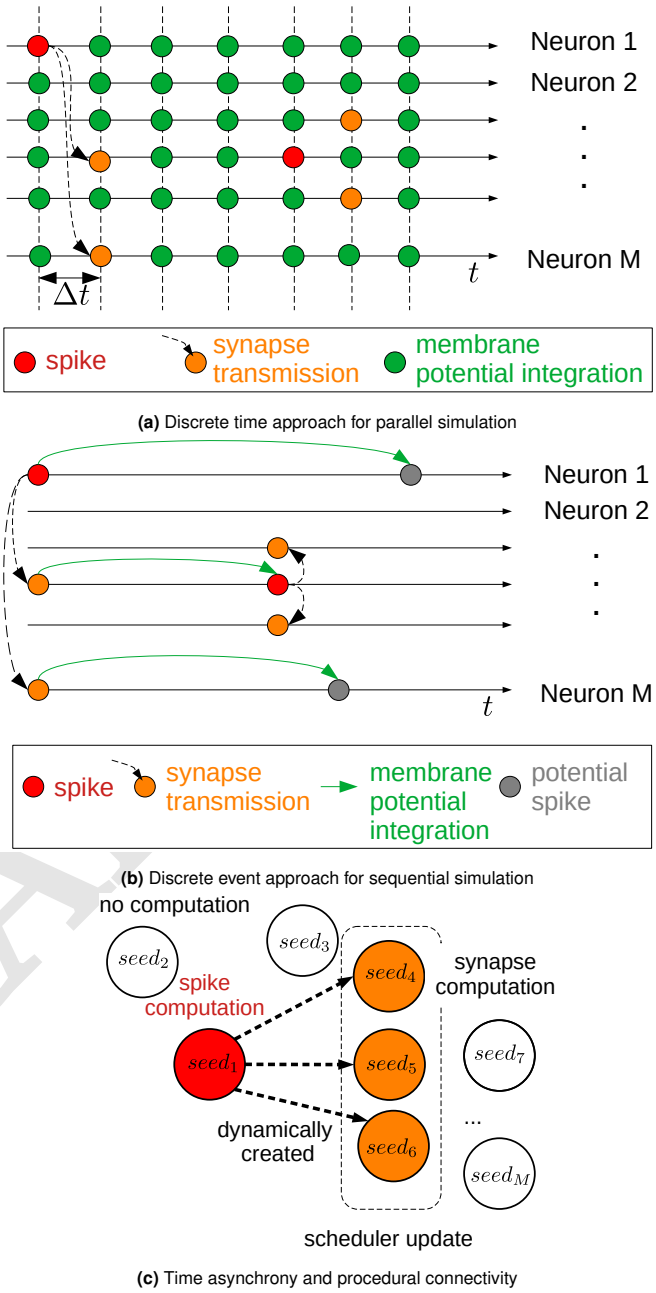


Fig. 1. Neuronal computations in time and in the network

variability, it is difficult to estimate quantitatively both physiological (number of synapses per neuron, etc.) and dynamic parameters (average firing rate, etc.) of neuronal networks in primates (6) and humans (36). Only rough estimates are available. The human brain being the more computationally intensive, we estimate here its main parameters for simulation. Our goal is indeed to show the algorithm scalability to simulate large networks with such parameters.

To our knowledge, the best documented region of the human brain is the (neo)cortex. Based on the structural statistics (number of neurons and synaptic connections) of neuronal networks in the (neo)cortex, we extrapolate here their representative parameter values at brain scale.

The *firing rate of a neuron* in the brain can be estimated

by the limited resources at its disposal, especially glucose. Measures of ATP consumption have shown (see (29)) that the firing rate of a neuron in human neocortex can be estimated around $0.16Hz$. Still based on ATP consumption, only 10% of the neurons in the neocortex can be active at the same time. So it seems coherent to choose an average of $0.16Hz$. These values can be extrapolated to the whole brain*, as follows.

The neocortex represents 80% of the volume of the brain (37) and consumes 44% of its energy (29). Considering that the energy consumed by the brain is proportional to the firing rate of the neurons, the power ratio then consists of

$$\frac{P_{cortex}}{P_{brain}} \sim \frac{V_{cortex}\bar{m}_{cortex}}{V_{brain}\bar{m}_{brain}},$$

with $\bar{m}_{cortex}$ the mean firing rate of individual neurons in the neocortex (resp. in the brain) and V_{cortex} the volume of the neocortex (resp. in the brain). The *average firing rate of the brain* then consists of $\bar{m}_{brain} = 0.8 \times \frac{0.16}{0.44} = 0.29\ Hz$ per neuron.

This average firing rate should not be confused with the fact that particular neurons can have a much larger firing rate. Particularly, groups of neurons synchronize together for achieving a particular cognitive task: this is the concept of neuronal assemblies (38). In an assembly, neurons can usually increase their rates to tens Hz (possibly $50Hz$) over a short duration. Therefore, we choose a *firing rate in the brain* where most of the neurons have a firing rate of $0.3Hz$ but some have a much higher firing rate (up to $50Hz$) using an heavy tailed distribution, see Materials and Methods and Table 2.

The average number of synaptic connections in human brains is hard to estimate and depends heavily on the neuron types and brain regions. For example, in the brain, it is assumed that the majority of neurons are cerebellum granule cells (39). In (40), the number of synaptic connections to granule neurons is estimated to an average of only 4 connections, matching those observed anatomically. On the other hand, Purkinje neurons can have up to $200,000$ synapses on only one dendrite in the human brain (39). The approximate number of synapses in the cortex is 0.6×10^{14} (41). Assuming that the volume of the cortex represents around 80% of the volume of the brain, the number of synapses in the brain is of order 10^{14}. Considering that the number of neurons in the human brain is of order 10^{11} (36), we find that the average number of synapses is about $1,000$ synapses per neuron†. The *synaptic connection probability* thus depends on the number of neurons M: $p_M = \frac{1,000}{M}$.

Finally, an action potential arriving on one pre-synaptic neuron produces an Excitatory PostSynaptic Potential (EPSP), or an Inhibitory PostSynaptic Potential (IPSP), in the postsynaptic neuron. The duration of these postsynaptic potentials is about $\tau = 20ms$ (39).

Therefore the parameters that we used in the simulation are indicated in Table 1. Notice that these parameters are generic and intuitive and can be taken easily into account in further studies, either at a biological or at theoretical model level.

Simulation duration	$T = 5s$
Mean firing rate	$\bar{m} = 0.3Hz$
Number of neurons simulated	$M = \{10^5, 10^6, 10^7, 10^8\}$
Synaptic connection probability	$p_M = 1000/M$
Postsynaptic duration	$\tau = 20ms$

Table 1. Neuronal network parameters at human brain scale level.

Software and hardware configurations. The simulations have been run on a Symmetric shared Memory multiProcessor (SMP) computer equipped with Intel CascadeLake@2.6GHz processors‡. This kind of computer is used here to have access to larger memory capacities. At computational level, only one processor was used for the simulations. For small sizes of networks requiring small amounts of memory (*cf.* Figure 3b), *e.g.* a network of 10^6 neurons with a total of 10^9 synaptic connections, this computer is equivalent to a simple desktop computer. The simulation of such networks takes only 25 minutes for each biological second. This is of the order of the 4.13×10^6 neurons and 24.2×10^9 synapses simulated on GPUs (24), which takes about 15 minutes for each biological second. This GPU-based simulation was already running up to 35% faster than on 1024 supercomputer nodes (one rack of an IBM Blue Gene/Q) (42). Our simulation only requires a single usual processor and no GPU.

The implementation of the algorithm is written in C++ (2011) programming language and compiled using g++ 9.3.0.

Firing rate at network level. Table 2 presents classical elementary statistics on the simulated firing rates, whereas Figure 2 presents the corresponding densities. As one can see in Section Material and Methods, the system is initialized with a lot of neurons whose spontaneous spiking activity is null. The system needs to warm up to have almost all neurons spiking. This explains why the density at $T = 5$s is still rippled whereas, at $T = 50s$, it looks much smoother. This last case corresponds basically to the stationary version of the process. Indeed, as explained in Section Material and Methods, the parameters of the Hawkes model (in particular the spontaneous spiking activity) have been fixed to achieve a certain stationary distribution of the firing rates (with mean 0.3Hz), which is heavy tailed to achieve records as large as 50 Hz. As one can see (even if at $T = 5$s the system is not warmed up yet with a lot of non spiking neurons), one can still achieve the desired average firing rate and extremal values. These basic statistics are not varying a lot with T (see Table 2). Note that the density plots are roughly the same for all configurations: with ripples at $T = 5$s and smooth curve at $T = 50$s.

Our approach is particularly adapted to simulate precisely and efficiently a huge disparity in frequency distributions. Indeed, our simulation algorithm (22) allows focusing efficiently the computing resources on highly spiking neurons without computing anything for almost silent neurons (*cf.* Figures 1a and 1b). The last advantage of our approach is to be able to store time stamps with a precision of $10^{-15}s$.

Execution times and memory usage. The simulation execution times are presented in Figure 3a for different sizes of

*This calculus can be found on AI impact project webpage: https://aiimpacts.org/rate-of-neuron-firing/ (lastly verified: 02/09/2021)

†Calculus on AI impact project webpage: https://aiimpacts.org/scale-of-the-human-brain/ (lastly verified: 02/09/2021).

‡We used v100l and v100xl partitions on Joliot-Curie supercomputer at TGCC as a Fenix Infrastructure resource. Each node of v100l and v100xl has Intel CascadeLake@2.6GHz processors. A node on v100l is a dual-socket one with 2x18 cores, each core having a memory of 10 GBytes, so the total amount of available memory is 360 GBytes. A node on v100xl is a quad-socket one with 4x18 cores, each core having a memory of 41.5 GBytes, so the total amount of available memory is 3 TBytes.

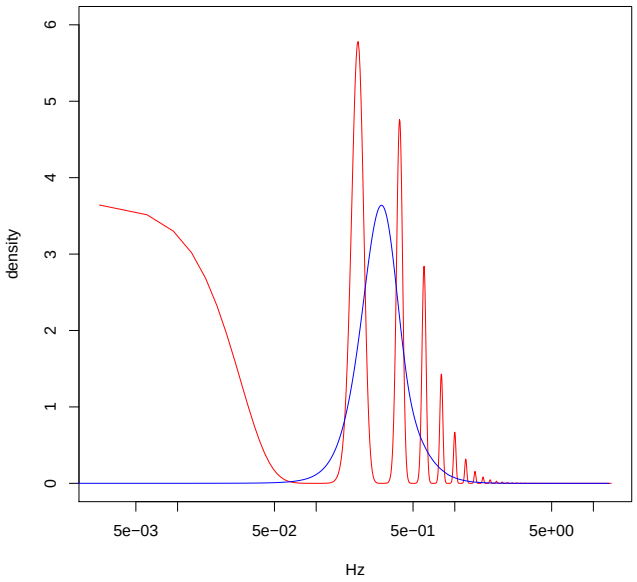


Fig. 2. Densities (on a logarithmic scale) of the simulated firing rates in the network with $M = 10^6$ neurons and $d = 1000$ post-synaptic connections in average. In red, for $T = 5$s and in blue for $T = 50$s. These densities are obtained with a Gaussian kernel estimator with bandwidth 0.02.

neural networks and different numbers of synaptic connections (called children). The experimental execution times obtained are in agreement with Eq. (1) which predicts, for instance, $O(10^{12})$ operations for $M = 10^7$ and $d = 10^3$. The curves are almost linear (with slopes around 1.1) with respect to the number of neurons, for different numbers of synaptic connections.

The total amount of memory used is displayed in Figure 3b. They are in agreement with the procedural memory complexity of Eq. (2) and also almost linear (with slopes slightly below 1). Note in particular that for $M = 10^7$, $d = 10^3$, $\omega = 32$ and $\epsilon = 64$ (leading to a 10^{-15} precision in time), the memory cost predicted by Eq. (2) is $O(10^{11})$ for the static implementation, whereas it is $O(10^9)$ for the procedural connectivity implementation. Besides notice that, as expected, increasing the average number of post-synaptic connections per neuron has few impact on the memory. Indeed, within the network, only the post-synaptic connections receiving spikes are dynamically generated.

M	d	Average freq.	Freq. min.	Freq. max.	Freq. std.	Percentage of non spiking neuron
1e5	250	0.279 (0.279)	0 (0)	14 (13.94)	0.315 (0.222)	31.2 (0.01)
1e5	500	0.334 (0.333)	0 (0.02)	6.6 (5.76)	0.328 (0.218)	23.5 (0)
1e5	1000	0.399 (0.398)	0 (0.04)	10.4 (11.64)	0.345 (0.220)	16.9 (0)
1e6	250	0.267 (0.267)	0 (0)	19.2 (20.84)	0.308 (0.217)	33 (0.01)
1e6	500	0.322 (0.324)	0 (0)	38.4 (39.38)	0.329 (0.225)	25.1 (0.00)
1e6	1000	0.383 (0.387)	0 (0.02)	13.4 (12.9)	0.344 (0.223)	18.5 (0)
5e6	250	0.26 (0.261)	0 (0)	27.4 (28.5)	0.307 (0.217)	34.1 (0.02)
5e6	500	0.315 (0.316)	0 (0)	34.2 (34.1)	0.324 (0.220)	26 (0.00)
5e6	1000	0.377	0	19.8	0.342	19
1e7	250	0.258 (0.259)	0 (0)	23.2 (21.62)	0.306 (0.217)	34.4 (0.02)
1e7	500	0.311	0	21.6	0.322	26.4
1e7	1000	0.374	0	21.8	0.342	19.3
5e7	250	0.253	0	38.2	0.304	35.4
5e7	500	0.305	0	50.6	0.321	27.2
1e8	250	0.251	0	46.2	0.303	35.7

Table 2. Firing rates elementary statistics obtained by simulation for different sizes of neural networks and different numbers of synaptic connections and $T = 5s$. The number between parentheses displays the results at $T = 50s$ for the less complex simulations

.

Material and Methods

Model. For a set of M neurons, we first design the graph of interaction by saying that neuron j influences neuron i if a Bernoulli variable $Z_{j\to i}$ of parameter p is non zero. The resulting network is an Erdös-Rényii graph.

Once the network is fixed, we design the spike apparition thanks to a Hawkes process, that is a point process whose intensity is given by

$$\lambda^i(t) = \nu_i + \sum_{j=1}^{M} \int_0^t h_{j\to i}(t-\tau) dN_\tau^j,$$

with dN^j the point measure associated to neuron j. In this formula, ν_i represents the spontaneous firing rate of the neuron i if the other neurons do not fire, whereas $h_{j\to i}$ is the interaction function, that is $h_{j\to i}(u)$ is the increase (if positive) or decrease (if negative) that the firing rate of neuron i suffers due to a spike on j, which happens u seconds before.

We are interested in a particular case of the Hawkes process where all the interaction functions are always the same when they are non null. More precisely, we set the interaction function

$$h_{j\to i} = Z_{j\to i}\theta h,$$

where h is a fixed positive interaction function of integral 1, θ is a tuning parameter that we need to calibrate to avoid explosion of the process. We also set $h_{i\to i} = 0$ (no self interaction). We take $h = 50\mathbb{1}_{[0,0.02]}$: the interaction function is a constant and non zero only on a small interval of length 20ms, which corresponds to typical Post Synaptic Potentials in the brain.

Let us denote $H_{j\to i} = \int_0^{+\infty} h_{j\to i}(t)dt)$ and $H = (H_{j\to i})_{i,j=1,\dots,M}$ is the corresponding matrix (line i corresponds to a triggered neuron, column j to a triggering neuron).

Note in particular that in this model, there are only excitatory neurons : if in the brain, there are inhibitory neurons, this will only reduce the number of points without changing the complexity. Moreover when all the interaction functions are non negative, we can easily understand the explosion condition.

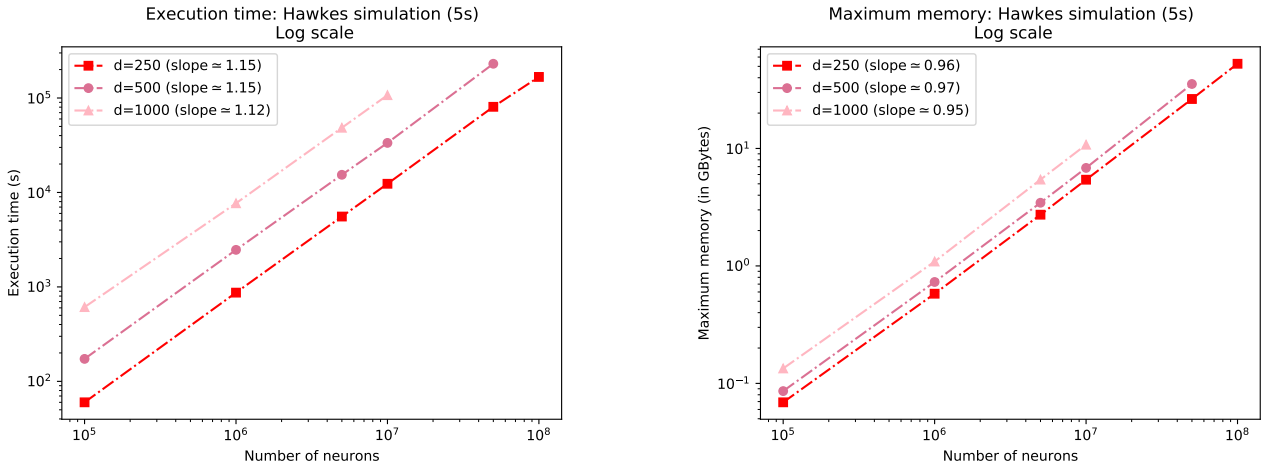
Indeed, this Hawkes process explodes, that is, it produces an exponentially increasing number of point per unit of time (see (43)) if the spectral radius of H is larger than 1.

If (*Condition Stat*) the spectral radius is strictly smaller than 1 (44), then a stationary version exists and the corresponding vector of mean firing rates $m = (m_i)_{i=1,\dots,M}$ is given by

$$m = (I - H)^{-1}\nu. \qquad [3]$$

Note also that if we start the simulation without points before 0 in (*Condition Stat*), the process is not stricto sensu stationary but it will converge to an equilibrium given by the stationary state (ergodic theorem) and that the number of points that will be produced is always smaller than the stationary version.

In the present case we want (i) to prevent explosion and (ii) to reach a certain vector m which is biologically realistic (average around 0.3 Hz, records around 50 Hz). Both of these calibrations can be done mathematically beforehand in the Hawkes model : we can guarantee the behavior of the whole



(a) Simulation execution times for different sizes of networks.

(b) Memory usages for different sizes of networks.

Fig. 3. Simulation execution times (3a) and memory usage (3b) for different sizes of networks.

system even before performing the simulation, whereas this might be much more intricate for other models such as LIF.

Choice of θ or how to avoid explosion. Note that $H = \theta\mathcal{Z}$, with $\mathcal{Z} = (Z_{j\to i})_{i,j=1,\dots,M}$. So if we can compute the largest eigenvalue of $\mathcal{Z}$ or an upper bound, we can decide how to choose θ.

We can use Gershgorin circles (45) to say that any complex eigenvalue λ of $\mathcal{Z}$ satisfies (because the diagonal is null),

$$|\lambda| \leq \max_{i=1,\dots,M} \sum_{j\neq i} Z_{j\to i}.$$

Therefore the spectral radius is upper bounded by $\max_{i=1,\dots,M} B_i$, where $B_i = \sum_{j\neq i} Z_{j\to i}$. This random quantity can be computed for small networks but it is clearly too intensive in our setting: indeed, with the procedural connectivity implementation, it is always easy to access the children ℓ of a given i, i.e. such that $i \to \ell$ is in the graph, but we need to look at all the neurons in the graphs to find out the set of parents j of i, i.e. such that $j \to i$ is in the graph. However, probabilistic estimates might be computed mathematically. Indeed B_i is just a sum of i.i.d. Bernoulli variables. So we can apply Bernstein's inequality (46). This leads to, for all positive x,

$$\forall i = 1, ..., M,$$
$$\mathbb{P}(B_i \geq (M-1)p + \sqrt{2(M-1)p(1-p)x} + x/3) \leq e^{-x},$$

and, by union bound, for the maximum

$$\mathbb{P}(\max_{i=1,\dots,M} B_i \geq (M-1)p + \sqrt{2(M-1)p(1-p)x} + x/3) \leq Me^{-x}.$$

Therefore let us fix a level α, say 1%, and take $x = \log(M) + \log(1/\alpha)$ in the previous equation. We obtain that with probability larger than $1-\alpha$, the spectral radius of $\mathcal{Z}$ is upper bounded

$$\rho_{\max} = (M-1)p + \xi_\alpha$$

with

$$\xi_\alpha = \sqrt{2(M-1)p(1-p)[\log(M) + \log(1/\alpha)]} + [\log(M) + \log(1/\alpha)]/3$$

Note that $\rho_{\max}$ is roughly $(M-1)p$, which is the largest eigenvalue of $\mathbb{E}(\mathcal{Z})$. Finally it means that if we take $\theta < 1/\rho_{\max}$, the process will not explode with probability larger than $1-\alpha$. In practice, to ensure a strong enough interaction, we take $\theta = 0.9\rho_{max}^{-1}$.

Choice of ν_i or how to constraint the distribution of the mean firing rates. The first step consists in deciding for a target distribution for the m_i. We have chosen to pick the m_i's independently as $0.1X$ where X is the absolute value of a student variable with mean 3 and 4 degrees of freedom. The choice of the student variable was driven by the wish of having a moderate heavy tail, which will ensure records around 50 Hz and a mean around 0.3Hz.

The problem is that the m_i's are not parameters of the model, so we need to understand how to tune ν_i to get such m_i's. Note that by inverting Eq. (3), we get that

$$(I - H)m = \nu$$

that is for all i

$$\nu_i = m_i - \theta \sum_{j\neq i} m_j Z_{j\to i},$$

which is very intuitive (47). Indeed the spontaneous rate that we need to put is the mean firing rate m_i minus what can be explained with the parents of i.

So in theory, the Hawkes model is very easy to tune for prescribed firing rates since there is a linear relationship between both. However, and for the same reasons as before, it might be too computationally intensive to compute this explicitly.

One possible way is to again use concentration inequalities, but this time on $\sum_{j\neq i} m_j Z_{j\to i}$ and not on B_i. However we decided to do something simpler, which works well (as seen in Figure 2).

Indeed $\sum_{j\neq i} m_j Z_{j\to i}$ is a sum of about $(M-1)p \simeq 1000$ i.i.d variables with mean $\bar{m} = 0.3$Hz. Hence it should be close to $\bar{m}B_i$. With the previous computations, we know already that ν_i should therefore be larger than $m_i - \theta\rho_{\max}\bar{m}$.

With the previous choice of $\theta = 0.9\rho_{\max}^{-1}$, we have chosen to take the positive part for the ν_i's in the simulation, that is :

$$\nu_i = \max(m_i - 0.9\bar{m}, 0).$$

Therefore ν_i remains positive or null, which guarantees that the Hawkes process stays linear. However, this also means that a non negligible portion of the neurons start with a null spontaneous firing rate, which explains the ripples of Figure 2.

With this choice, we cannot hope to have exactly the same distribution as the desired m_i's, but it conserves the same heavy tail and roughly the same mean firing rate as the one we wanted, as one can see on Table 2.

Discussion

Thanks to time asynchrony, we propose a new scalable algorithm to the simulate spiking activity of neuronal networks. We are able to generate roughly the same firing pattern as a real brain for a range between 10^5 and 10^8 neurons, in a few minutes on a single processor, most parameters being tuned thanks to general considerations inferred from the literature. Corresponding computational and memory complexities are shown to be both linear.

At simulation level, whereas usual simulations are based on the continuous variation of the electrical potentials of LIF neurons, point processes lead to much more efficient simulations. In particular, instead of computing the small continuous variations for all neurons, only discrete spikes and their interactions are simulated in the network. Between two spikes no computations are done. Point processes also lead to time asynchrony (two spikes cannot occur at the same time in the network), which is a fundamental hypothesis for the algorithm to work.

Combining the time asynchrony hypothesis with procedural connectivity drastically reduces the memory consumption and also, for the same network activity, reduces the computations per spikes (*cf.* Figures 1a and 1b). In particular, complexities (both theoretical and concrete) can be computed and proved to be almost linear in the number of neurons, when Hawkes processes are generated, leading to simulation scalability of the whole approach without precision loss.

Both modeling and simulation results open many research perspectives. We are currently developing new discrete event algorithms that are able to simulate the spiking activity of neurons embedded in potentially infinite neuronal networks (48). This paves the path for simulation of parts of the brain as an open physical system.

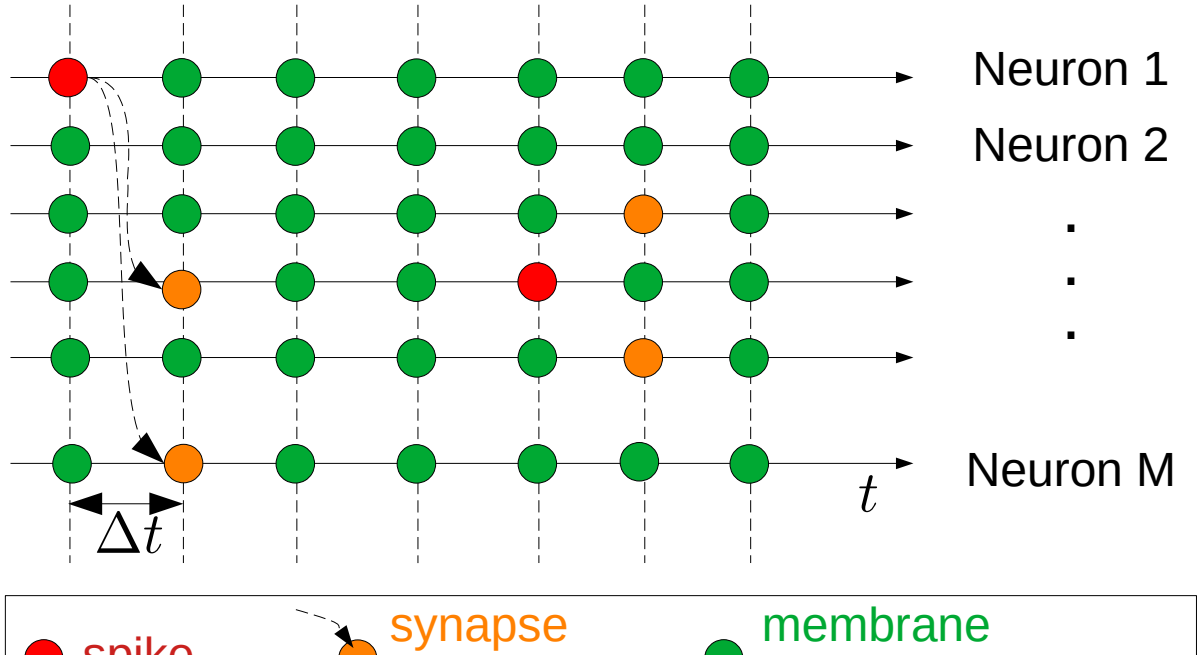
Furthermore, the minimal number of computations and memory storage obtained here open new exciting perspectives with respect to massive neuromorphic computers, by improving the energy saving consumption of neuromorphic components (49).

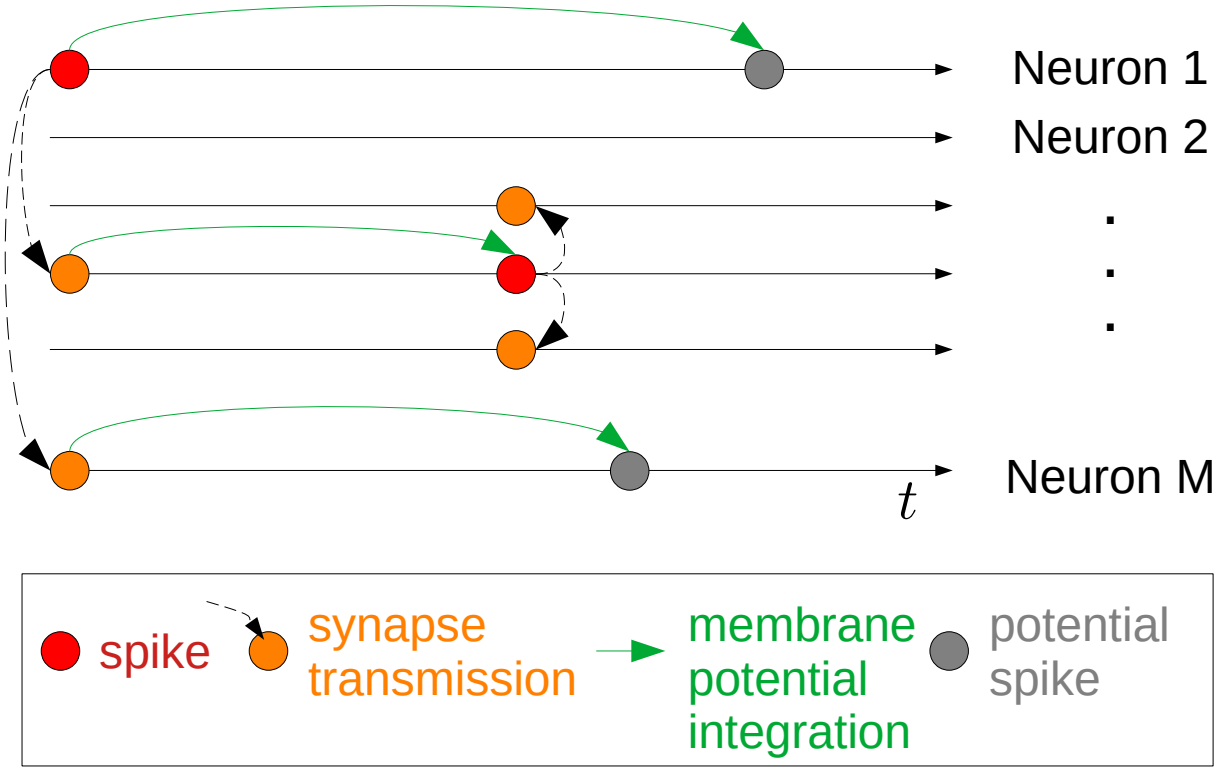
Finally, if we have proved that the simulation is doable, the point process model used here can be calibrated further on real data, by incorporating inhibition and more variability in the interaction functions. Also, this model can be used for reconstructing the functional connectivity of experimentally recorded neurons (15, 47) to have access to more realistic interaction functions. Besides, as only few computing resources (one single processor) is used with a minimal memory amount, this opens new possibilities to run in parallel many independent replications of stochastic simulations of large networks. This is particularly interesting for calibrating models based on real data collections.

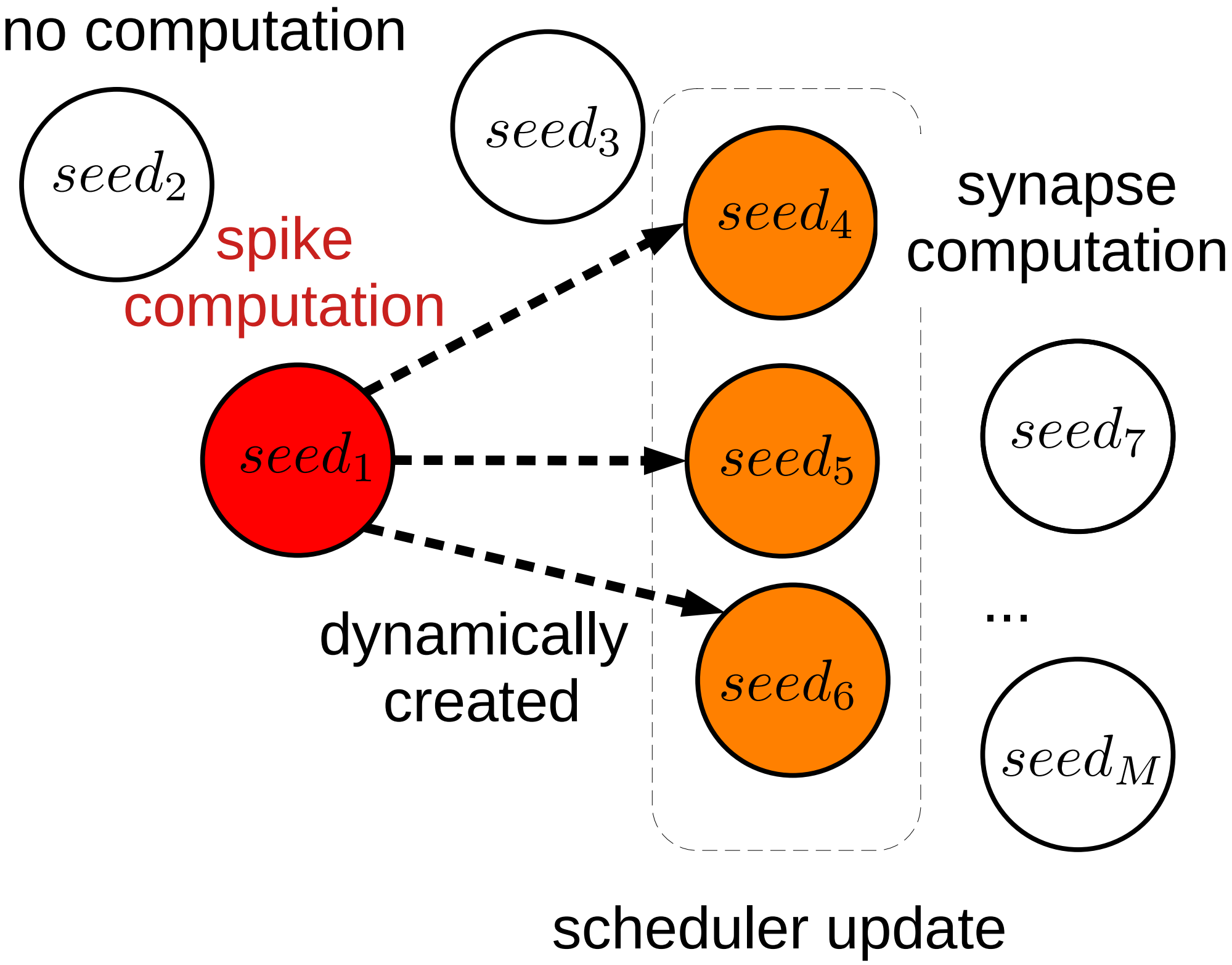
ACKNOWLEDGMENTS. This work is part of the project HyperBrain from Human Brain Project (HBP) EBRAINS EU initiative. The simulations were run on Fenix Infrastructure resources, which are partially funded from the European Union's Horizon 2020 research and innovation program through the ICEI project under the grant agreement No. 800858. Our research was supported by the French government, through CNRS, the UCA$^{\text{Jedi}}$ and 3IA Côte d'Azur Investissements d'Avenir managed by the National Research Agency (ANR-15- IDEX-01 and ANR-19-P3IA-0002), directly by the ANR project ChaMaNe (ANR-19-CE40-0024-02) and by the interdisciplinary Institute for Modeling in Neuroscience and Cognition (NeuroMod) of the Université Côte d'Azur.

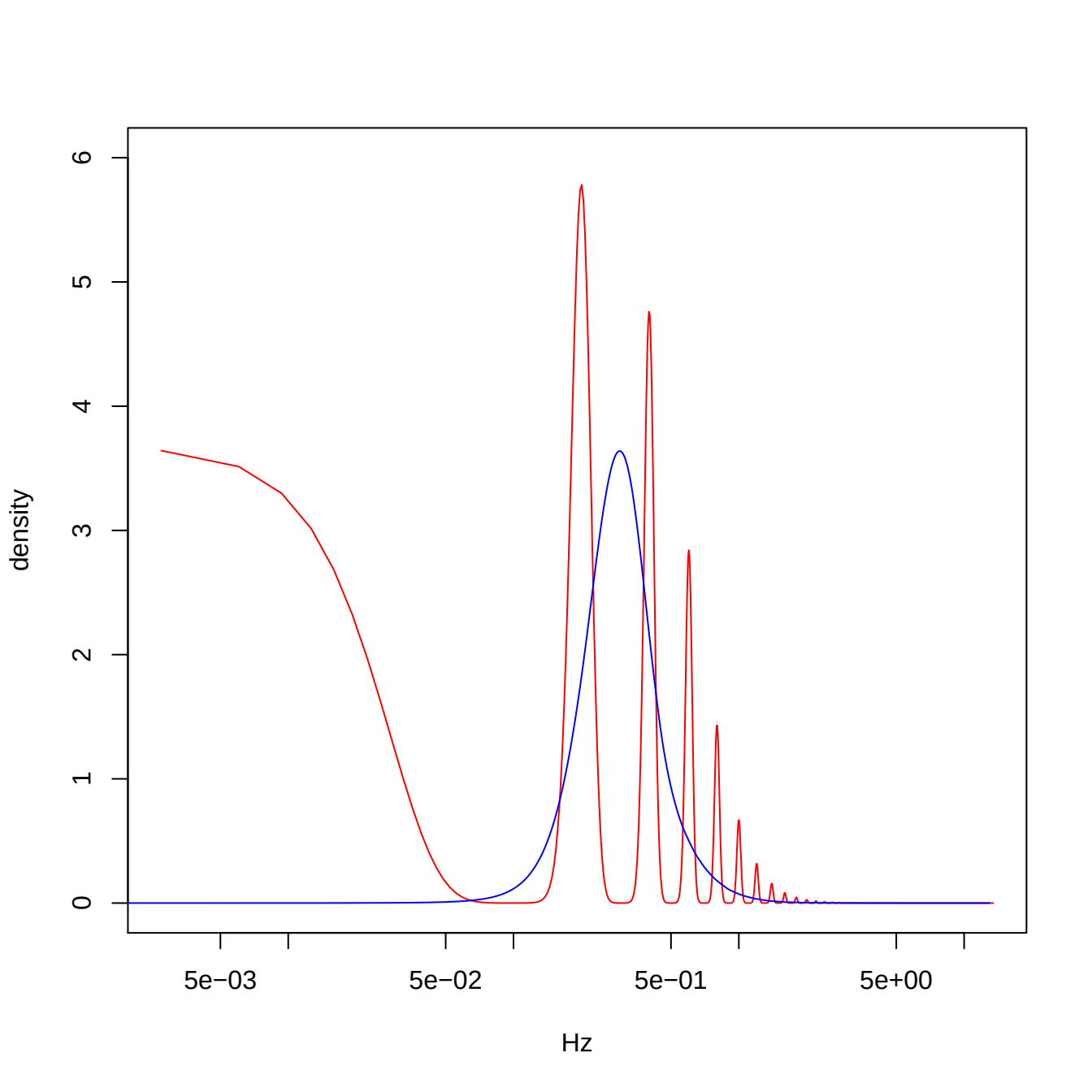
1. K Amunts, et al., The human brain project—synergy between neuroscience, computing, informatics, and brain-inspired technologies. PLoS biology **17**, e3000344 (2019).
2. MR Dando, Japan's brain/minds project (2020).
3. W Koroshetz, et al., The state of the nih brain initiative. J. Neurosci. **38**, 6427–6438 (2018).
4. RR Netz, WA Eaton, Estimating computational limits on theoretical descriptions of biological cells. PNAS **118** (2021).
5. E Gal, et al., Rich cell-type-specific network topology in neocortical microcircuitry. Nat. Neurosci. **20**, 1004–1013 (2017).
6. S Herculano-Houzel, CE Collins, P Wong, JH Kaas, Cellular scaling rules for primate brains. Proc. Natl. Acad. Sci. **104**, 3562–3567 (2007).
7. DA McCormick, Y Shu, Y Yu, Hodgkin and huxley model—still standing? Nature **445**, E1–E2 (2007).
8. J Guckenheimer, RA Oliva, Chaos in the hodgkin–huxley model. SIAM J. on Appl. Dyn. Syst. **1**, 105–114 (2002).
9. L Lapicque, Recherches quantitatives sur l'excitation electrique des nerfs traitee comme une polarization. J. Physiol. Pathology **9**, 620–635 (1907).
10. H Yamaura, J Igarashi, T Yamazaki, Simulation of a human-scale cerebellar network model on the k computer. Front. neuroinformatics **14**, 16 (2020).
11. RF Fox, Stochastic versions of the hodgkin-huxley equations. Biophys. journal **72**, 2068–2074 (1997).
12. AA Lazar, EA Pnevmatikakis, Reconstruction of sensory stimuli encoded with integrate-and-fire neurons with random thresholds. EURASIP J. on Adv. Signal Process. **2009**, 1–14 (2009).
13. W Truccolo, UT Eden, MR Fellows, JP Donoghue, EN Brown, A point process framework for relating neural spiking activity to spiking history, neural ensemble, and extrinsic covariate effects. J. Neurophysiol. **93**, 1074–1089 (2005).
14. EN Brown, R Barbieri, V Ventura, RE Kass, L Frank, The time-rescaling theorem and its application to neural spike train data analysis. Neural Comput. **14**, 325–346 (2002).
15. P Reynaud-Bouret, V Rivoirard, F Grammont, C Tuleau-Malot, Goodness-of-fit tests and nonparametric adaptive estimation for spike train analysis. The J. Math. Neurosci. **4**, 3 (2014).
16. J Pillow, et al., Spatio-temporal correlations and visual signalling in a complete neuronal population. Nature **454**, 995–999 (2008).
17. C Pouzat, A Chaffiol, Automatic spike train analysis and report generation. an implementation with r, r2html and star. J. Neurosci. Methods **181**, 119–144 (2009).
18. A Galves, E Löcherbach, Infinite systems of interacting chains with memory of variable length—a stochastic model for biological neural nets. J. Stat. Phys. **151**, 896–921 (2013).
19. L Sacerdote, MT Giraudo, Stochastic Biomathematical Models. (Lecture Notes in Mathematics, Springer) Vol. 2058, pp. 99–148 (2013).
20. P Brémaud, Point processes and queues. (Springer-Verlag, New York-Berlin), (1981) Martingale dynamics, Springer Series in Statistics.
21. C Mascart, A Muzy, P Reynaud-bouret, Efficient simulation of sparse graphs of point processes (2020).
22. A Muzy, Exploiting activity for the modeling and simulation of dynamics and learning processes in hierarchical (neurocognitive) systems. Comput. Sci. Eng. **21**, 84–93 (2019).
23. Grazieschi, Paolo, et al., Network of interacting neurons with random synaptic weights. ESAIM: ProcS **65**, 445–475 (2019).
24. JC Knight, T Nowotny, Larger gpu-accelerated brain simulations with procedural connectivity. Nat. Comput. Sci. **1**, 136—142 (2021).
25. A Muzy, BP Zeigler, F Grammont, Iterative specification as a modeling and simulation formalism for i/o general systems. IEEE Syst. J. **12**, 2982–2993 (2017).
26. R Brette, et al., Simulation of networks of spiking neurons: a review of tools and strategies. J. computational neuroscience **23**, 349–398 (2007).
27. M Rudolph, A Destexhe, Analytical integrate-and-fire neuron models with conductance-based dynamics for event-driven simulation strategies. Neural computation **18**, 2146–2210 (2006).
28. JD Touboul, OD Faugeras, A markovian event-based framework for stochastic spiking neural networks. J. computational neuroscience **31**, 485–507 (2011).
29. P Lennie, The cost of cortical computation. Curr. Biol. **13**, 493 – 497 (2003).
30. A Duarte, A Galves, E Löcherbach, G Ost, Estimating the interaction graph of stochastic neural dynamics. Bernoulli **25**, 771–792 (2019).
31. P Jahn, RW Berg, Jr Hounsgaard, S Ditlevsen, Motoneuron membrane potentials follow a time inhomogeneous jump diffusion process. J. Comput. Neurosci. **31**, 563–579 (2011).

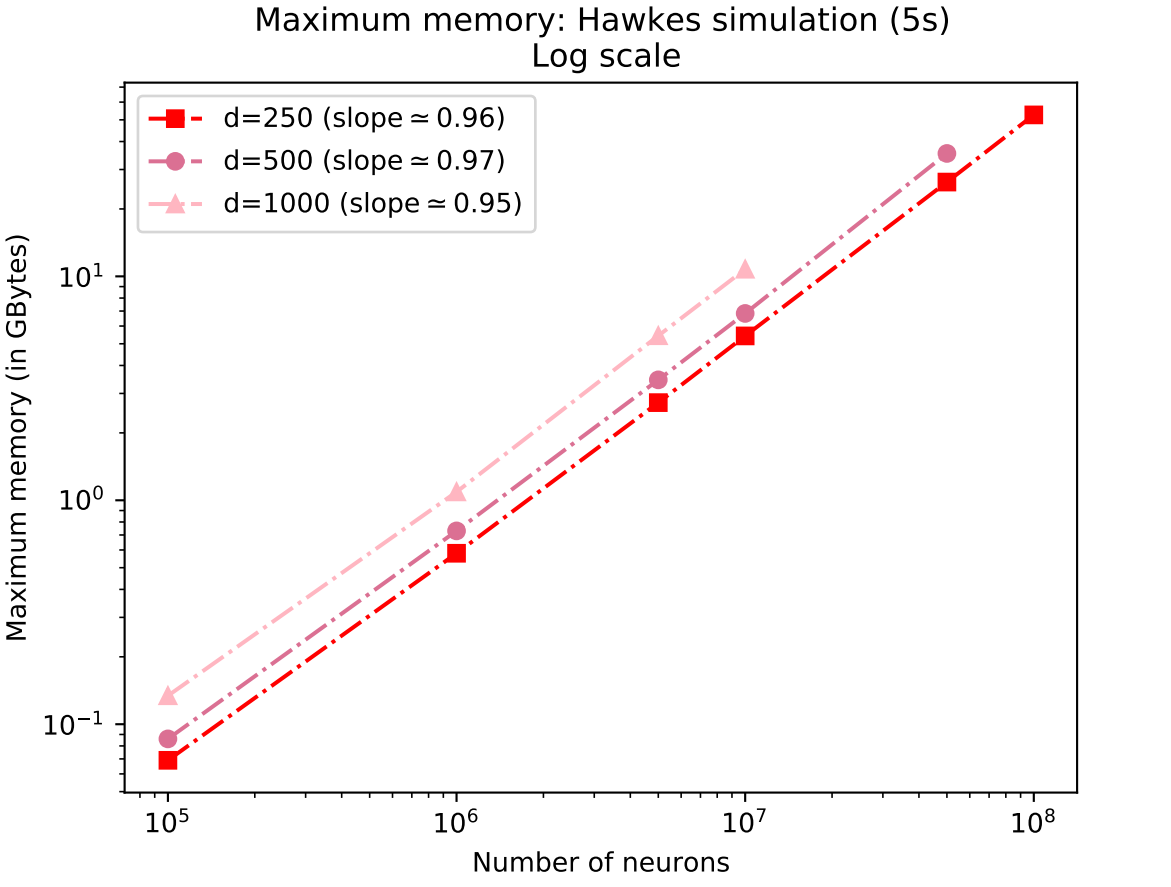
32. MJ Cáceres, JA Carrillo, B Perthame, Analysis of nonlinear noisy integrate & fire neuron models: blow-up and steady states. J. Math. Neurosci. **1**, Art. 7, 33 (2011).
33. F Delarue, J Inglis, S Rubenthaler, E Tanré, Global solvability of a networked integrate-and-fire model of McKean-Vlasov type. Ann. Appl. Probab. **25**, 2096–2133 (2015).
34. V Didelez, Graphical models of markes point processes based on local independence. J.R. Stat. Soc. B **70**, 245–264 (2008).
35. C Tuleau-Malot, A Rouis, F Grammont, P Reynaud-Bouret, Multiple Tests Based on a Gaussian Approximation of the Unitary Events Method with Delayed Coincidence Count. Neural Comput. **26**, 1408–1454 (2014).
36. S Herculano-Houzel, R Lent, Isotropic fractionator: A simple, rapid method for the quantification of total cell and neuron numbers in the brain. J. Neurosci. **25**, 2518–2521 (2005).
37. H Stephan, H Frahm, G Baron, New and revised data on volumes of brain structures in insectivores and primates. Folia primatologica **35**, 1–29 (1981).
38. F Grammont, A Riehle, Spike synchronization and firing rate in a population of motor cortical neurons in relation to movement direction and reaction time. Biol. cybernetics **88**, 360–373 (2003).
39. GM Shepherd, The synaptic organization of the brain. (Oxford university press), (2004).
40. A Litwin-Kumar, KD Harris, R Axel, H Sompolinsky, L Abbott, Optimal degrees of synaptic connectivity. Neuron **93**, 1153–1164.e7 (2017).
41. DA Drachman, Do we have brain to spare? Neurology **64**, 2004–2005 (2005).
42. MO Gewaltig, M Diesmann, Nest (neural simulation tool). Scholarpedia **2**, 1430 (2007).
43. S Delattre, N Fournier, M Hoffmann, Hawkes processes on large networks. Ann. App. Probab. **26**, 216 – 261 (2016).
44. AG Hawkes, D Oakes, A cluster process representation of a self-exciting process. J. Appl. Probab. **11**, 493–503 (1974).
45. R Varga, Gershgorin and his circles. (Springer-Verlag), (2004) Springer Series in Computational Mathematics.
46. S Boucheron, G Lugosi, P Massart, Concentration inequalities. (Oxford University Press, Oxford), (2013) A nonasymptotic theory of independence, With a foreword by Michel Ledoux.
47. P Reynaud-Bouret, A Muzy, I Bethus, Towards a mathematical definition of functional connectivity. (2021).
48. TC Phi, A Muzy, P Reynaud-Bouret, Event-Scheduling Algorithms with Kalikow Decomposition for Simulating Potentially Infinite Neuronal Networks. SN Computer Science **1** (2020).
49. LA Plana, et al., spinnlink: Fpga-based interconnect for the million-core spinnaker system. IEEE Access **8**, 84918–84928 (2020).

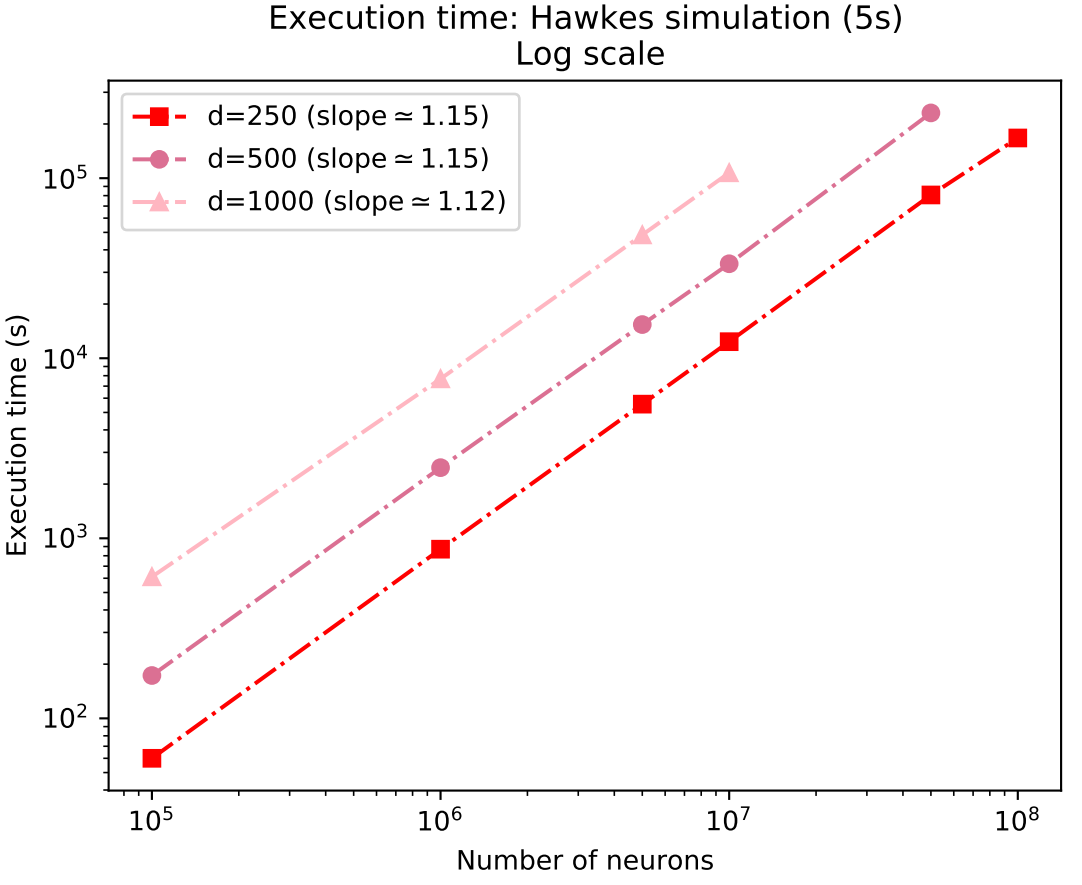

Neuron 1
Neuron 2
Neuron M
t
Δt
spike
synapse transmission
membrane potential integration


Neuron 1
Neuron 2
Neuron M
t
spike
synapse transmission
membrane potential integration
potential spike


no computation
$seed_2$
$seed_3$
spike
computation
$seed_1$
$seed_4$
$seed_5$
$seed_6$
synapse
computation
$seed_7$
...
$seed_M$
dynamically
created
scheduler update


density
0
1
2
3
4
5
6
5e−03
5e−02
5e−01
5e+00
Hz


Maximum memory: Hawkes simulation (5s)
Log scale
d=250 (slope ≃ 0.96)
d=500 (slope ≃ 0.97)
d=1000 (slope ≃ 0.95)
Maximum memory (in GBytes)
10^{1}
10^{0}
10^{-1}
10^{5}
10^{6}
10^{7}
10^{8}
Number of neurons


Execution time: Hawkes simulation (5s)
Log scale
d=250 (slope ≃ 1.15)
d=500 (slope ≃ 1.15)
d=1000 (slope ≃ 1.12)
Execution time (s)
10^2
10^3
10^4
10^5
10^5
10^6
10^7
10^8
Number of neurons