

***Reverse-engineering in spiking neural networks
parameters: exact deterministic parameters
estimation***

Horacio Rostro-González, Juan Carlos Vasquez-Betancur, Bruno Cessac, Thierry Viéville

N° 7199

September 2009

Thème BIO

 ***Rapport
de recherche***

Reverse-engineering in spiking neural networks parameters: exact deterministic parameters estimation

Horacio Rostro-González*, Juan Carlos Vasquez-Betancur*, Bruno
Cessac*, Thierry Viéville†

Thème BIO — Systèmes biologiques
Équipes-Projets NeuroMathComp & Cortex

Rapport de recherche n° 7199 — September 2009 — 41 pages

Abstract: We consider the deterministic evolution of a time-discretized network with spiking neurons, where synaptic transmission has delays, modeled as a neural network of the generalized integrate-and-fire (gIF) type. The purpose is to study a class of algorithmic methods allowing one to calculate the proper parameters to reproduce exactly a given spike train, generated by an hidden (unknown) neural network.

This standard problem is known as NP-hard when delays are to be calculated. We propose here a reformulation, now expressed as a Linear-Programming (LP) problem, thus allowing us to provide an efficient resolution. This allows us to “reverse-engineer” a neural network, i.e. to find out, given a set of initial conditions, which parameters (i.e., synaptic weights in this case), allow to simulate the network spike dynamics.

More precisely we make explicit the fact that the reverse-engineering of a spike train, is a Linear (L) problem if the membrane potentials are observed and a LP problem if only spike times are observed. Numerical robustness is discussed. We also explain how it is the use of a generalized IF neuron model instead of a leaky IF model that allows us to derive this algorithm.

Furthermore, we point out how the L or LP adjustment mechanism is local to each unit and has the same structure as an “Hebbian” rule. A step further, this paradigm is easily generalizable to the design of input-output spike train transformations. This means that we have a practical method to “program” a spiking network, i.e. find a set of parameters allowing us to exactly reproduce the network output, given an input.

Numerical verifications and illustrations are provided.

Key-words: Spiking neural networks, Discretized integrate and fire neuron models and computing with spikes.

* INRIA NeuroMathComp <http://www-sop.inria.fr/neuromathcomp>

† INRIA Cortex <http://cortex.loria.fr>

Reverse-engineering in spiking neural networks parameters: exact deterministic parameters estimation

Résumé : Nous considérons l'évolution déterministe d'un réseau de neurones à impulsion, du type intègre-et-tire généralisé (gIF) où des poids synaptiques incluent des délais. Le but est d'étudier un groupe de méthodes algorithmiques qui permettent de calculer de manière appropriée les paramètres afin de reproduire de manière exacte un train d'impulsions (*spike train*) donné, généré par un réseau inconnu.

Inclure les délais comme des paramètres à estimer conduit à un problème NP-complet. Nous proposons une re-formulation correspondant à un problème de programmation linéaire (LP) donnant ainsi une solution efficace au problème. De cette façon retrouver les paramètres (i.e. poids synaptiques) à partir d'un ensemble de données, ce qui nous permet de simuler la dynamique de ce réseau de neurones à impulsion.

De manière plus précise on explicite le fait que la *reverse-engineering* d'un train d'impulsions est un problème linéaire (L) si le potentiel de la membrane est observé et un problème LP dans le cas où seulement les impulsions sont connues. Le modèle gIF est utilisé ici. Les problèmes de robustesse numérique sont discutés.

De plus nous signalons comment le mécanisme d'ajustement dans le problème L ou LP se fait de manière locale en chaque unité avec une structure semblable à une règle du type Hebbien. En poussant le raisonnement, ce paradigme est facilement généralisable à la conception des transformations sur trains d'impulsions entrée/sortie. Nous obtenons une méthode pratique pour programmer un réseau de neurones, i.e. trouver un ensemble de paramètres qui nous permet de reproduire de manière exacte la sortie d'un réseau à partir d'une entrée donnée.

Mots-clés : Réseau de neurones à impulsion, modèle de neurones intègre et tire à temps discret, calcul avec spikes

1 Introduction

Neuronal networks have tremendous computational capacity, but their biological complexity make the exact reproduction of all the mechanisms involved in these networks dynamics essentially impossible, even at the numerical simulation level, as soon as the number of neurons becomes too large. One crucial issue is thus to be able to reproduce the “output” of a neuronal network using approximated models easy to implement numerically. The issue addressed here is “Can we program an integrate and fire network, i.e. tune the parameters, in order to exactly reproduce another network output, on a bounded time horizon, given the input”.

Calculability power of neural network models

The main aspect we are interesting here is the *calculability* of neural network models. It is known that recurrent neural networks with frequency rates are universal approximators [37], as multilayer feed-forward networks are [23]. This means that neural networks are able to simulate dynamical systems¹, not only to approximate measurable functions on a compact domain, as originally stated (see, e.g., [37] for a detailed introduction on these notions). Spiking neuron networks can be also universal approximators [30].

Theoretically, spiking neurons can perform very powerful computations with precise timed spikes. They are at least as computationally powerful as the sigmoidal neurons traditionally used in artificial neural networks [29, 31]. This results has been shown using a spike-response model (see [33] for a review) and considering piecewise linear approximations of the potential profiles. In this context, analog inputs and outputs are encoded by temporal delays of spikes. The authors show that any feed-forward or recurrent (multi-layer) analog neuronal network (à-la Hopfield, e.g., McCulloch-Pitts) can be simulated arbitrarily closely by an insignificantly larger network of spiking neurons. This holds even in the presence of noise [29, 31]. These results highly motivate the use of spiking neural networks, as studied here.

In a computational context, spiking neuron networks are mainly implemented through specific network architectures, such as Echo State Networks [25] and Liquid State Machines [32], that are called “reservoir computing” (see [47] for unification of reservoir computing methods at the experimental level). In this framework, the *reservoir* is a network model of neurons (it can be linear or sigmoid neurons, but more usually spiking neurons), with a random topology and a sparse connectivity. The *reservoir* is a recurrent network, with weights than can be either fixed or driven by an unsupervised learning mechanism. In the case of spiking neurons (e.g. in the model of [34]), the learning mechanism is a form of synaptic plasticity, usually STDP (Spike-Time-Dependent Plasticity), or a temporal Hebbian unsupervised learning rule, biologically inspired. The output layer of the network (the so-called “readout neurons”) is driven by a supervised learning rule, generated from any type of classifier or regressor, ranging from a least mean squares rule to sophisticated discriminant or regression algorithms. The ease of training and a guaranteed optimality guides the choice of the method. It appears that simple methods yield good results [47]. This distinction between a readout

¹As an example see the very interesting paper of Albers-Sprott using this property to investigate the dynamical stability conjecture of Pales and Smale in the field of dynamical systems theory [3] or route to chaos in high dimensional systems [2]

layer and an internal reservoir is indeed induced by the fact that only the output of the neuron network activity is constrained, whereas the internal state is not controlled.

Learning the parameters of a neural network model

In biological context, learning is mainly related to synaptic plasticity [21, 16] and STDP (see e.g., [42] for a recent formalization), as far as spiking neuron networks are concerned. This unsupervised learning mechanism is known to reduce the variability of neuron responses [7] and is related to the maximization of information transmission [43] and mutual information [15]. It has also other interesting computational properties such as tuning neurons to react as soon as possible to the earliest spikes, or segregate the network response in two classes depending on the input to be discriminated, and more general structuring such as emergence of orientation selectivity [22].

In the present study, the point of view is quite different: we consider supervised learning, since “each spike matter”, i.e., in the special case of a feed-forward sweep of visual activity in response to a brief visual presentation [22, 18]; we want, not only to statistically reproduce the spiking output, but also to reproduce it exactly.

The motivation to explore this track is twofold. On one hand we want to better understand what can be learned at a theoretical level by spiking neuron networks, tuning weights and delays. The key point is the non-learnability of spiking neurons [52], since it is proved that this problem is NP-complete, when considering the estimation of both weights and delays. Here we show that we can “elude” this caveat and propose an alternate efficient estimation, inspired by biological models.

We also have to notice, that the same restriction apply not only to simulation but, as far as this model is biologically plausible, also holds at the biological level. It is thus an issue to wonder if, in biological neuron networks, delays are really estimated during learning processes, or if a weaker form of weight adaptation, as developed now, is considered.

On the other hand, the computational use of spiking neuron networks in the framework of reservoir computing or beyond [38], at application levels, requires efficient tuning methods not only in “average”, but in the deterministic case. This is the reason why we must consider how to *exactly* generate a given spike train.

Approximating neural dynamics

In [35] the authors propose 2 variants of the IF model for the approximation of spike trains, the first variant correspond to a nonlinear IF model where the parameters depend on the instantaneous membrane potential, for the second variant it is based in the Spike Response Model and it depends on the time elapsed since the last spike time. The model is able to have a spike coincidence between 70 and 80% for random current input and 73% for random conductance injection. At the numerical level they present a technique for the parameters optimization based on spike trains in the full conductance-based model. This technique consist in minimizing the distance between the original spike train and the estimated spike train using the downhill simplex method. In a first instance the reader could be find that the work in [35] and the work presented in this paper is solved by the same way, but there is a very important difference; the work presented here use the simplex method in order to solve a Linear Programming problem and find the exact deterministic parameters estimation. The results showed below verify that our model have a 100% spike trains coincidence on artificial and biological data.

What is the paper about

In this paper we detail the proposed methods in three steps: first, discussing the neural model considered here; then, detailing the family of estimation problems corresponding to what is called reverse-engineering and discussing the related computational properties; finally, making explicit how a general input/output mapping can be “compiled” on a spiking neural network thanks to the previous developments.

In the subsequent section, numerical verifications and illustrations are provided, before the final discussion and conclusion.

2 Problem position: Discretized integrate and fire neuron models.

Let us consider a normalized and reduced “punctual conductance based generalized integrate and fire” (gIF) neural unit model [19] as reviewed in [36]. The model is reduced in the sense that both adaptive currents and non-linear ionic currents are no more explicitly depending on the potential membrane, but on time and previous spikes only (see [12] for a development).

Here we follow [8, 13, 12] after [39] and review how to properly discretize a gIF model. The precise derivation is not re-given here, except for one innovative aspect, whereas the method is only sketched out (see [13, 12] for details).

Time constrained continuous formulation. Let v be the normalized membrane potential, which triggers a spike for $v = 1$ and is reset at $v = 0$. The fire regime (spike emission) reads $v(t) = 1 \Rightarrow v(t^+) = 0$. Let us write $\tilde{\omega}_t = \{\dots t_i^n \dots\}$, the list of spike times $t_i^n < t$. Here t_i^n is the n -th spike-time of the neuron of index i . The dynamic of the integrate regime reads:

$$\frac{dv}{dt} + \frac{1}{\tau_L} [v - E_L] + \sum_j \sum_n \rho_j(t - t_j^n) [v - E_j] = i_m(\tilde{\omega}_t),$$

Here, τ_L and E_L are the membrane leak time-constant and reverse potential, while $\rho_j(s)$ and E_j are the spike responses and reversal potentials for excitatory/inhibitory synapses and gap-junctions. Furthermore, $\rho_j(s)$ is the synaptic or gap-junction response, accounting for the connection delay and time constant; shape showed in Fig. 1.

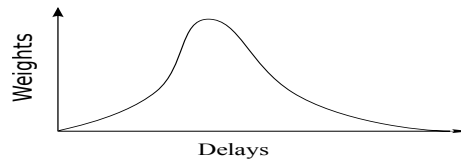


Figure 1: $\rho_j(s)$ profil

Finally, $i_m()$ is the reduced membrane current, including simplified adaptive and non-linear ionic current (see [12] for details).

The dynamic of the integrate regime thus writes:

$$\frac{dv}{dt} + g(t, \tilde{\omega}_t) v = i(t, \tilde{\omega}_t),$$

so that knowing the membrane potential at time t , the membrane potential at time $t + \delta$, writes:

$$\begin{aligned} v(t + \delta) &= \nu(t, t + \delta, \tilde{\omega}_t) v(t) + \int_t^{t+\delta} \nu(s, t + \delta, \tilde{\omega}_s) i(s, \tilde{\omega}_s) ds, \\ \log(\nu(t_0, t_1, \tilde{\omega}_{t_0})) &= - \int_{t_0}^{t_1} g(s, \tilde{\omega}_s) ds. \end{aligned}$$

The key point is that temporal constraints are to be taken into account [10]. Spike-times are bounded by a refractory period r , $r < d_i^{n+1}$, defined up to some absolute precision δt , while there is always a minimal delay dt for one spike to influence another spike, and there might be (depending on the model assumptions) a maximal interspike interval D such that either the neuron fires within a time delay $< D$ or remains quiescent forever). For biological neurons, orders of magnitude are typically, in milliseconds:

r	δt	dt	D
1	0.1	$10^{-[1,2]}$	$10^{[3,4]}$

Network dynamics discrete approximation. Combining these assumptions and the previous equations allows one (see [13] for technical details) to write the following discrete recurrence equation for a sampling period δ :

$$V_i[k] = \gamma_i V_i[k-1] (1 - Z_i[k-1]) + \sum_{j=1}^N \sum_{d=1}^D W_{ijd} Z_j[k-d] + I_{ik}, \quad (1)$$

where $V_i[k] = v_i(k\delta)$ and $Z_i[k] = \xi_{[1,+\infty[}(V_i[k])$, where ξ is the indicatrix function, $\xi_A(x) = 1$ if $x \in A$ and 0 otherwise.

Let us discuss in detail how (1) is derived from the previous equations.

The term $(1 - Z_i[k])$ implements the reset mechanism, since this term is equal to 0 when $V_i[k] \geq 1$. The interesting technical point is that this equation entirely specifies the integrate and fire mechanism.

The $\gamma_i \equiv \nu(t, t + \delta, \tilde{\omega}_t)|_{t=k\delta}$ term takes into account the multiplicative effects of conductances. The numerical analysis performed in [13] demonstrates that, for numerical values taken from bio-physical models, considering here $\delta \simeq 0.1ms$, this quantity related to the contraction rate, is remarkably constant, with small variations within the range:

$$\gamma_i \in [0.965, 0.995] \simeq 0.98,$$

considering random independent and identically distributed Gaussian weights. It has been numerically verified that taking this quantity constant over time and neurons does not significantly influence the dynamics. This the reason why we write γ_i as a constant here. This corresponds to a ‘‘current based’’ (instead of ‘‘conductance based’’) approximation of the connections dynamics.

The additive current

$$I_{ik} \equiv \int_t^{t+\delta} \nu(s, t + \delta, \tilde{\omega}_s) \left(i_m(\tilde{\omega}_s) + \frac{E_L}{\tau_L} \right) ds \Big|_{t=k\delta} \simeq \delta \gamma_i \left(i_m(\tilde{\omega}_t) + \frac{E_L}{\tau_L} \right) \Big|_{t=k\delta}$$

accounts for membrane currents, including leak. The right-hand side approximation assume γ_i is constant. Furthermore, we have to assume that the additive currents are independent from the spikes. This means that we neglect the membrane current non-linearity and adaptation.

On the contrary, the term related to the connection weights W_{ijd} is not straightforward to write and requires to use the previous numerical approximation. Let us write:

$$\begin{aligned} W_{ij}[k - k_j^n] &\equiv E_j \int_t^{t+\delta} \nu(s, t + \delta, \tilde{\omega}_t) \rho_j(t - t_j^n) ds \Big|_{t=k\delta, t_j^n=k_j^n\delta} \\ &\simeq E_j \delta \gamma_i \rho_j(t - t_j^n) \Big|_{t=k\delta, t_j^n=k_j^n\delta}, \end{aligned}$$

assuming $\nu(s, t + \delta, \tilde{\omega}_t) \simeq \gamma_i$ as discussed previously. This allows us to consider the spike response effect at time $t_j^n = k_j^n \delta$ as a function only of $k - k_j^n$. The response $W_{ij}[d]$ vanishes after a delay $D, \tau_r = D\delta$, as stated previously. We assume here that $\delta < \delta t$ i.e. that the spike-time precision allows one to define the spike time as $k_j^n, t_j^n = k_j^n \delta$ (see [13, 12] for an extensive discussion). We further assume that only zero or one spike is fired by the neuron of index j , during a period δ , which is obvious as soon as $\delta < r$.

This allows us to write $W_{ijd} = W_{ij}[d]$ so that:

$$\begin{aligned} \sum_{j=1}^n W_{ij}[k - k_j^n] &= \sum_{d=1}^D \sum_{j=1}^n W_{ij}[d] \xi_{\{k_j^n\}}(k - d) \\ &= \sum_{d=1}^D W_{ij}[d] \xi_{\{k_j^1, \dots, k_j^n, \dots\}}(k - d) \\ &= \sum_{d=1}^D W_{ijd} Z_j[k - d] \end{aligned}$$

since $Z_j[l] = \xi_{\{k_j^1, \dots, k_j^n, \dots\}}(l)$ is precisely equal to 1 on spike time and 0 otherwise, which completes the derivation of (1).

Counting the model's degrees of freedom. Let us consider a network of N units, whose dynamics is defined by (1), generating a raster of the form schematized in Fig. 2.

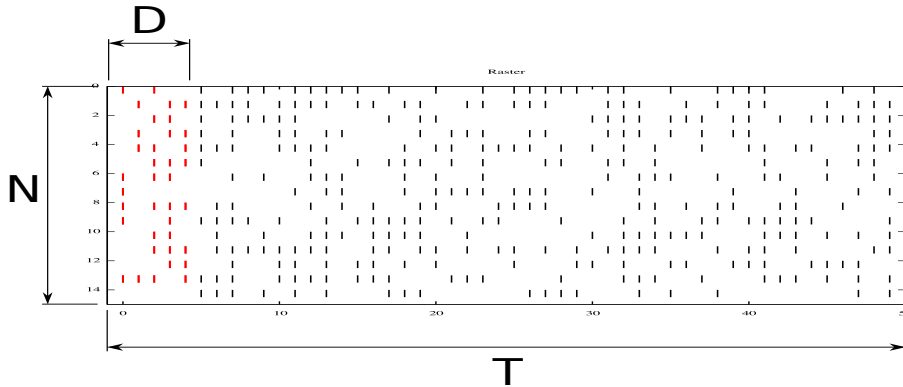


Figure 2: Schematic representation of a raster of N neurons observed during a time interval T after an initial conditions interval D (in red). This example corresponds to a periodic raster, but non-periodic raster are also considered. See text for further details.

In order to determine the dynamics of the neural network, we require the knowledge of the initial condition. Here, due to the particular structure of equation (1) with a

delay D , the initial condition is the piece of trajectory $V_i[k]$, $k \in \{0, D\}$. The notation $k \in \{0, D\}$ stands for $0 \leq k < D$.

If the neuron i has fired at least once, the dependence in the initial condition is removed thanks to the reset mechanism. This means that its state does not depend on $V_i[0]$ any more, as soon as spikes are known. We thus can further assume $V_i[0] = 0$, for the sake of simplicity.

The dynamics is parametrized by the weights W_{ijd} thus $N \times N \times D$ values. Here it is assumed that the γ_i are known and constant, while I_{ik} are also known, as discussed below.

When the potential and/or spikes are observed during a period of T samples, $N \times T$ numerical/binary values are measured.

3 Methods: Weights and delayed weights estimation

With the assumption that $V_i[0] = 0$ discussed previously, (1) writes:

$$V_i[k] = \sum_{j=1}^N \sum_{d=1}^D W_{ijd} \sum_{\tau=0}^{\tau_{ik}} \gamma^\tau Z_j[k - \tau - d] + I_{ik\tau} \quad (2)$$

writing $I_{ik\tau} = \sum_{\tau=0}^{\tau_{ik}} \gamma^\tau I_{i(k-\tau)}$ with:

$$\tau_{ik} = k - \arg \min_{l>0} \{Z_i[l-1] = 1\},$$

the derivation of this last form being easily obtained by induction from (1). Here τ_{ik} is the delay from the last spiking time, i.e., the last membrane potential reset. If no spike, we simply set $\tau_{ik} = k$.

This equation shows that there is a direct explicit linear relation between spikes and membrane potential. See [8] for more detailed about the one-to-one correspondence between the spike times and the membrane potential trajectory that defined by (1) and (2). Here, we use (2) in a different way.

Let us now discuss how to retrieve the model parameters from the observation of the network activity. We propose different solutions depending on the paradigm assumptions.

Retrieving weights and delayed weights from the observation of spikes and membrane potential

Let us assume that we can observe both the spiking activity $Z_i[k]$ and the membrane potential $V_i[k]$. Here, (2) reads in matrix form:

$$\mathbf{C}_i \mathbf{w}_i = \mathbf{d}_i \quad (3)$$

with:

$$\begin{aligned} \mathbf{C}_i &= \begin{pmatrix} \dots & \dots & \dots \\ \dots & \sum_{\tau=0}^{\tau_{ik}} \gamma^\tau Z_j[k - \tau - d] & \dots \\ \dots & \dots & \dots \end{pmatrix} \in \mathbb{R}^{T-D \times ND}, \\ \mathbf{d}_i &= (\dots \ V_i[k] - I_{ik\tau} \ \dots)^t \in \mathbb{R}^{T-D}, \\ \mathbf{w}_i &= (\dots \ W_{ijd} \ \dots)^t \in \mathbb{R}^{ND}. \end{aligned}$$

writing \mathbf{u}^t the transpose of \mathbf{u} .

Here, \mathbf{C}_i is defined by the neuron spike inputs, \mathbf{d}_i is defined by the neuron membrane potential outputs and membrane currents, and the network parameter by the weights vector \mathbf{w}_i .

More precisely, \mathbf{C}_i is a rectangular matrix with:

- $N D$ columns, corresponding to product with the $N D$ unknowns W_{ijd} , for $j \in \{1, N\}$ and $d \in \{1, D\}$,
- $T - D$ rows, corresponding to the $T - D$ measures $V_i[k]$, for $k \in \{D, T\}$, and,
- $T - D \times N D$ coefficients corresponding to the raster, i.e. the spikes $Z_j[k]$.

The weights are thus directly *defined by a set of linear equalities for each neuron*. Let us call this a Linear (L) problem.

The equation defined in (3), concerns only the weights of one neuron of index i . It is thus a weight estimation local to a neuron, and not global to the network. Furthermore, the weight estimation is given by the observation of the input $Z_i[k]$ and output $V_i[k]$. These two characteristics correspond to usual Hebbian-like learning rules architecture. See [21] for a discussion.

Given a general raster (i.e., assuming \mathbf{C}_i is of full rank $\min(T - D, N D)$):

- This linear system of equations has always solutions, in the general case, if:

$$N > \frac{T - D}{D} = O\left(\frac{T}{D}\right) \Leftrightarrow D > \frac{T}{N + 1} = O\left(\frac{T}{N}\right) \Leftrightarrow D(N + 1) > T. \quad (4)$$

This requires enough non-redundant neurons N or weight profile delays D , with respect to the observation time T . In this case, given any membrane potential and spikes values, *there are always weights able to map the spikes input onto the desired potential output*.

- On the other hand, if $N D \leq T - D$, then the system has no solution in the general case. This is due to the fact that we have a system with more equations than unknowns, thus with no solution in the general case. However, there is obviously a solution if the potentials and spikes have been generated by a neural network model of the form of (1).

If \mathbf{C}_i is not of full rank, this may correspond to several cases, e.g.:

- Redundant spike pattern: some neurons do not provide linearly independent spike trains.
- Redundant or trivial spike train: for instance with a lot of bursts (with many $Z_j[k] = 1$) or a very sparse train (with many $Z_j[k] = 0$). Or periodic spike trains.

Regarding the observation duration T , it has been demonstrated in [8, 13] that the dynamic of an integrate and fire neural network is generically² periodic. This however depends on parameters such as external current or synaptic weights, while periods can be larger than any accessible computational time.

²Considering a basic leaky integrate and fire neuron network the result is true except for a negligible set of parameters. Considering an integrate and fire neuron model with conductance synapses the result is true, providing synaptic responses have a finite memory.

In any case, several choices of weights \mathbf{w}_i (in the general case a $D(N+1) - T$ dimensional affine space) may lead to the same membrane potential and spikes. The problem of retrieving weights from the observation of spikes and membrane potential may thus have many solutions.

The particular case where $D = 1$ i.e. where there is no delayed weights but a simple weight scalar value to define a connection strengths is included in this framework.

Retrieving weights and delayed weights from the observation of spikes

Let us now assume that we can observe the spiking activity $Z_i[k]$ only (and not the membrane potentials) which corresponds to the usual assumption, when observing a spiking neural network.

In this case, the value of $V_i[k]$ is not known, whereas only its position with respect to the firing threshold is provided:

$$Z_i[k] = 0 \Leftrightarrow V_i[k] < 1 \text{ and } Z_i[k] = 1 \Leftrightarrow V_i[k] \geq 1,$$

which is equivalent to write the condition:

$$e_{ik} = (2Z_i[k] - 1)(V_i[k] - 1) \geq 0.$$

If the previous condition is verified for all time index k and all neuron index i , then the spiking activity of the network exactly corresponds to the desired firing pattern.

Expanding (2), with the previous condition allows us to write, in matrix form:

$$\mathbf{e}_i = \mathbf{A}_i \mathbf{w}_i + \mathbf{b}_i \geq 0 \quad (5)$$

writing:

$$\begin{aligned} \mathbf{A}_i &= \begin{pmatrix} \dots & \dots & \dots \\ \dots & (2Z_i[k] - 1) \sum_{\tau=0}^{T_{jk}} \gamma^\tau Z_j[k - \tau - d] & \dots \\ \dots & \dots & \dots \end{pmatrix} \in R^{T-D \times ND}, \\ \mathbf{b}_i &= (\dots \quad (2Z_i[k] - 1)(I_{ik\tau} - 1) \quad \dots)^t \in R^{T-D}, \\ \mathbf{w}_i &= (\dots \quad W_{ij d} \quad \dots)^t \in R^{ND}, \\ \mathbf{e}_i &= (\dots \quad (2Z_i[k] - 1)(V_i[k] - 1) \quad \dots)^t \in R^{T-D}, \end{aligned}$$

thus $\mathbf{A}_i = \mathbf{D}_i \mathbf{C}_i$ where \mathbf{D}_i is the non-singular $R^{T-D \times T-D}$ diagonal matrix with $\mathbf{D}_i^{kk} = 2Z_i[k] - 1 \in \{-1, 1\}$.

The weights are now thus directly *defined by a set of linear inequalities for each neuron*. This is therefore a Linear Programming (LP) problem. See [17] for an introduction and [6] for the detailed method used here to implement the LP problem.

Furthermore, the same discussion about the dimension of the set of solutions applies to this new paradigm except that we now have to consider a *simplex* of solution, instead of a simple *affine sub-space*.

A step further, $0 \leq e_{ik}$ is the “membrane potential distance to the threshold”. Constraining the e_{ik} is equivalent to constraining the membrane potential value $V_i[k]$.

It has been shown in [8] how:

$$|\mathbf{e}|_\infty = \min_i \inf_{k \geq 0} e_{ik} \quad (6)$$

can be interpreted as a “edge of chaos” distance, the smallest $|\mathbf{e}|$ the higher the dynamics complexity, and the orbits periods.

On the other hand, the higher e_{ik} , the more robust the estimation. If e_{ik} is high, sub-threshold and sup-threshold values are clearly distinct. This means that numerical errors are not going to generate spurious spikes or cancel expected spikes.

Furthermore, the higher $|e|_\infty$ the smaller the orbits period [8]. As a consequence, the generated network is expected to have rather minimal orbit periods.

In the next paper, in order to be able to use an efficient numerical implementation, we are going to consider a weaker but more robust norm, than $|e|_\infty$:

$$|e_i|_1 = \sum_k e_{ik} \quad (7)$$

We are thus going to maximize, for each neuron, the sum, thus, up to a scale factor, the average value of e_{ik} .

Let us now derive a bound for e_{ik} . Since $0 \leq V_i[k] < 1$ for sub-threshold values and reset as soon as $V_i[k] > 1$, it is easily bounded by:

$$V_i^{min} = \sum_{jd, W_{ijd} < 0} W_{ijd} \leq V_i[k] \leq V_i^{max} = \sum_{jd, W_{ijd} > 0} W_{ijd}$$

and we must have at least $V_i^{max} > 1$ in order for a spike to be fired while $V_i^{min} \leq 0$ by construction. These bounds are attained in the high-activity mode when either all excitatory or all inhibitory neurons fire. From this derivation, $e^{max} > 0$ and we easily obtain:

$$e^{max} = \max_i(1 - V_i^{min}, V_i^{max} - 1)$$

$$0 < e_{ik} \leq e^{max}$$

thus an explicit bound for e_{ik} .

Collecting all elements of the previous discussion, the present estimation problem writes:

$$\max_{e_i, w_i} \sum_k e_k, \text{ with, } 0 < e_{ik} \leq e^{max}, \text{ and, } e_i = \mathbf{A}_i \mathbf{w}_i + \mathbf{b}_i \quad (8)$$

which is a standard bounded linear-programming problem.

The key point is that a LP problem can be solved in polynomial time, thus is not a NP-complete problem, subject to the curse of combinatorial complexity. In practice, this LP problem can be solved using one of the several LP solution methods proposed in the literature (i.e., Simplex Method, which is, in principle, NP-complete in the worst case, but in practice, as fast as, when not faster, than polynomial methods).

Retrieving signed and delayed weights from the observation of spikes

In order to illustrate how the present method is easy to adapt to miscellaneous paradigms, let us now consider the fact that the weight emitted by each neuron have a fixed sign, either positive for excitatory neurons, or negative for inhibitory neurons. This additional constraint, known as the ‘‘Dale principle’’ [40], is usually introduced to take into account the fact, that synaptic weights signs are fixed by the excitatory or inhibitory property of the presynaptic neuron.

Although we do not focus on the biology here, it is interesting to notice that this additional constraint is obvious to introduce in the present framework, writing:

$$W_{ijd} = S_{ijd} W_{ijd}^\bullet, \text{ with } S_{ijd} \in \{-1, 1\}, \text{ and } W_{ijd}^\bullet \geq 0$$

thus separating the weight sign $S_{ij d}$ which is a-priori given and the weight value $W_{ij d}^\bullet$ which now always positive.

Then, writing:

$$\mathbf{A}^\bullet ijkd = \mathbf{A}_{ijkd} S_{ij d}$$

the previous estimation problem becomes:

$$\max_{\mathbf{e}_i, \mathbf{w}_i^\bullet} \sum_k e_k, \text{ with, } 0 < e_{ik} \leq e^{max}, 0 \leq W_{ij d}^\bullet \leq 1, \text{ and, } \mathbf{e}_i = \mathbf{A}_i^\bullet \mathbf{w}_i^\bullet + \mathbf{b}_i \quad (9)$$

which is still a similar standard linear-programming problem.

Retrieving delayed weights and external currents from the observation of spikes

In the previous derivations, we have considered the membrane currents I_{ik} as inputs, i.e. they are known in the estimation. Let us briefly discuss the case where they are to be estimated too.

For adjustable non-stationary current I_{ik} , the estimation problem becomes trivial. An obvious solution is $W_{ij d} = 0, I_{ik} = 1 + a(Z_i[k] - 1/2)$ for any $a > 0$, since each current value can directly drive the occurrence or inhibition of a spike, without any need of the network dynamics.

Too many degrees of freedom make the problem uninteresting: adjusting the non-stationary currents leads to a trivial problem.

To a smaller extends, considering adjustable stationary currents I_i also ‘‘eases’’ the estimation problem, providing more adjustment variables. It is obvious to estimate not only weights, but also the external currents, since the reader can easily notice that yet another linear-programming problem can be derived.

This is the reason why we do not further address the problem here, and prefer to explore in details a more constrained estimation problem.

Considering non-constant leak when retrieving parametrized delayed weights

For the sake of simplicity and because this corresponds to numerical observations, we have assumed here that the neural leak γ is constant. The proposed method still works if the leak varies with the neuron and with time i.e. is of the form γ_{it} , since this is simply yet another input to the problem. The only difference is that, in (2) and the related equations, the term γ^τ is to be replaced by products of γ_{it} .

However, if γ is a function of the neural dynamics, $\gamma \equiv \gamma(\omega_{-\infty}^t)$, thus of $W_{ij d}$, the previous method must be embedded in a non linear estimation loop. Since we know from [13] that this dependency is numerically negligible in this context, we can propose the following loop:

1. Fix at step $t = 0$, $\gamma_{it}^0 \equiv \gamma(\omega_{-D}^0)$, to initial values.
2. k - Estimate the weights $W_{ij d}$, given leaks γ_{it}^k at $k = 0, 1, \dots$
3. k - Re-simulate the dynamics, given the weights and to obtain corrected values $\tilde{\gamma}_{it}^k$.
4. k - Smoothly modify $\gamma_{it}^{k+1} = (1 - v) \gamma_{it}^{k+1} + v \tilde{\gamma}_{it}^k$

5. $k + 1$ Repeat step 2, k - for $k + 1$, the convergence of this non-linear relaxation method being guaranteed for sufficiently small v . See [50] for an extended discussion about this class of methods.

This shows that considering models with leaks depending on the dynamics itself is no more a LP-problem, but an iterative solving of LP-problems.

Retrieving parametrized delayed weights from the observation of spikes

In order to further show the interest of the proposed method, let us now consider that the *profile* of the weights is fixed, i.e. that

$$W_{ijd} = W_{ij}^{\circ} \alpha_{\tau}(d) \text{ with, e.g., } \alpha(d) = \frac{d}{\tau} e^{-\frac{d}{\tau}}$$

thus the weights is now only parametrized by a magnitude W_{ij}° , while the temporal profile is known.

Here $\alpha_{\tau}(d)$ is a predefined synaptic profile, while τ is fixed by biology (e.g., $\tau = 2ms$ for excitatory connections and $\tau = 10ms$ for inhibitory ones). Let us note that the adjustment of τ would have been a much more complex problem, as discussed previously in the non-parametric case.

This new estimation is defined by:

$$\mathbf{e}_i = \mathbf{A}_i^{\circ} \mathbf{w}_i^{\circ} + \mathbf{b}_i > 0 \quad (10)$$

writing:

$$\begin{aligned} \mathbf{A}_i^{\circ} &= \begin{pmatrix} \cdots & & & & & \\ \cdots & (2Z_i[k] - 1) \sum_d \sum_{\tau=0}^{\tau_{jk}} \gamma^{\tau} Z_j[k - \tau - d] \alpha(d) & \cdots & & & \\ \cdots & & \cdots & & & \\ \cdots & & & & & \end{pmatrix} \in R^{T-D \times N} \\ \mathbf{w}_i^{\circ} &= (\dots W_{ij} \dots)^t \in R^N \end{aligned}$$

thus a variant of the previously discussed mechanisms.

This illustrates the nice versatility of the method. Several other variants or combinations could be discussed (e.g. parametrized delayed weights from the observation of spikes and potential, ..), but they finally leads to the same estimations.

About retrieving delays from the observation of spikes

Let us now discuss the key idea of the paper.

In the previous derivations, we have considered delayed weights, i.e. a quantitative weight value W_{ijd} at each delay $d \in \{1, D\}$.

Another point of view is to consider a network with adjustable synaptic delays. Such estimation problem may, e.g., correspond to the “simpler” model:

$$V_i[k] = \gamma_i V_i[k-1] (1 - Z_i[k-1]) + \sum_{j=1}^n W_{ij} Z_j[k - d_{ij}] + I_{ik},$$

where now the weights W_{ij} and delays d_{ij} are to estimated.

As pointed out previously, the non-learnability of spiking neurons is known [52], i.e. the previous estimation is proved to be NP-complete. We have carefully checked in [52] that the result still apply to the present setup. This means that in order to “learn” the proper parameters we have to “try all possible combinations of delays”. This is

intuitively due to the fact that each delay has no “smooth” effect on the dynamics but may change the whole dynamics in an unpredictable way.

We see here that the estimation problem of delays d_{ij} seems not compatible with usable algorithms, as reviewed in the introduction.

We propose to elude this NP-complete problem by considering *another* estimation problem. Here we do not estimate *one* delay (for each synapse) but consider connection weights at several delay and then estimate a weighted pondering of their relative contribution. This means that we consider a *weak* delay estimation problem.

Obviously, the case where there is a weight W_{ij} with a corresponding delay $d_{ij} \in \{0, D\}$ is a particular case of considering several delayed weights W_{ijd} (corresponding to have all equal weights to zero except at d_{ij} , i.e., $W_{ijd} = W_{ij}$ if $d = d_{ij}$ else 0).

We thus do not restrain the neural network model by changing the position of the problem, but enlarge it. In fact, the present estimation provides a smooth approximation of the previous NP-complete problem.

We can easily conjecture that the same restriction also apply of the case where the observation of spikes and membrane potential is considered.

We also have to notice, that the same restriction apply not only to simulation but, as far as this model is biologically plausible, also true at the biological level. It is thus an issue to wonder if, in biological neural network, delays are really estimated during learning processes, or if a weaker form of weight adaptation, as discussed in this paper, is considered.

4 Methods: Exact spike train simulation

4.1 Introducing hidden units to match any raster

Position of the problem

Up to now, we have assumed that a raster $\bar{Z}_i[k], i \in \{1, N\}, k \in \{1, T\}$ is to be generated by a network whose dynamics is defined by (1), with initial conditions $\bar{Z}_j[k], j \in \{1, N\}, k \in \{1, D\}$ and $V_j[0] = 0$. In the case where a solution exists, we have discussed how to compute it.

We have seen that a solution always exists, *in the general case*, if the observation period is small enough, i.e., $T < O(ND)$. Let us now consider the case where $T \gg O(ND)$.

In this case, there is, in general, no solution. This is especially the case when the raster has not been generated by a network given by (1), e.g., in the case when the raster is random.

What can we do then ? For instance, in the case when the raster is entirely random and is not generated by a network of type (1) ?

The key idea, borrowed from the reservoir computing paradigm reviewed in the introduction, is to add a reservoir of “hidden neurons”, i.e., to consider not N but $N + S$ neurons. The set of N “output” neurons is going to reproduce the expected raster $\bar{Z}_i[k]$ and the set of S “hidden” neurons to increase the number of degree of freedom in order to obtain $T < O((N + S)D)$, thus being able to apply the previous algorithms to estimate the optimal delayed weights. Clearly, in the worst case, it seems that we have to add about $S = O(T/D)$ hidden neurons. This is illustrated in Fig. 3.

In order to make this idea clear, let us consider a trivial example.

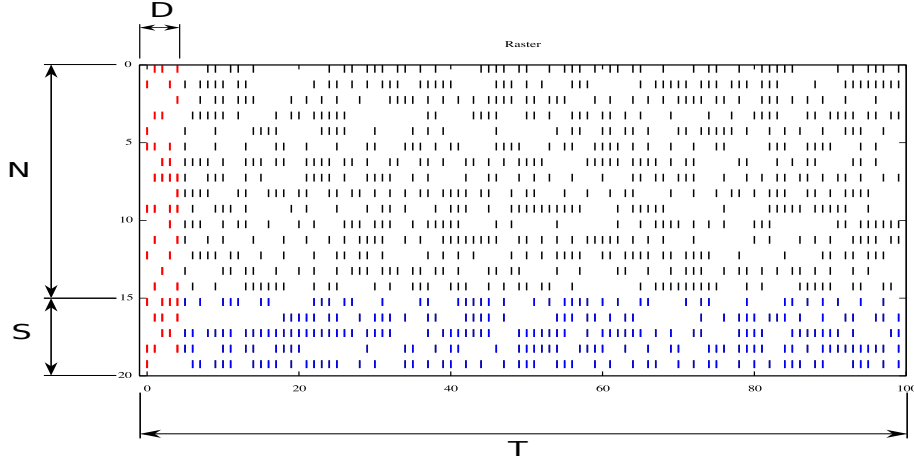


Figure 3: Schematic representation of a raster of N output neuron observed during a time interval T after an initial conditions interval D , with an add-on of S hidden neurons, in order increase the number of degree of freedom of the estimation problem. See text for further details.

Sparse trivial reservoir

Let us consider, as illustrated in Fig. 4, $S = T/D + 1$ hidden neurons of index $i' \in \{0, S\}$ each neuron firing once at $t_{i'} = i' D$, except the last one always firing (in order to maintain a spiking activity), thus:

$$Z_{i'}[k] = \delta(i' D - k), 0 \leq i' < S, Z_S[k] = 1$$

Let us choose:

$$\begin{aligned} W_{SS1} &> 1 \\ W_{i'S1} &= \frac{1-\gamma}{1-\gamma^{t_{i'}-1/2}} \\ W_{i'i'1} &< -\frac{\gamma^{2t_{i'}-\gamma^T}}{\gamma^T(1-\gamma^{t_{i'}})} < 0 \\ W_{i'j'd} &= 0 \text{ otherwise} \end{aligned}$$

with initial conditions $Z_{i'}[k] = 0, i' \in \{0, S\}$ and $Z_S[k] = 1, k \in \{1, D\}$, while $I_{i'k} = 0$.

A straight-forward derivation over equation (1) allows us to verify that this choice generates the specified $Z_{i'}[k]$. In words, as the reader can easily verify, it appears that:

- the neuron of index S is always firing since (through W_{SS1}) a positive internal loop maintains its activity;
- the neurons of index $i' \in \{0, S\}$, whose equation writes:

$$V_{i'}[k] = \gamma V_{i'}[k-1] (1 - Z_{i'}[k-1]) + W_{i'S1} + W_{i'i'1} Z_{i'}[k-1]$$

is firing at $t_{i'}$ integrating the constant input $W_{i'S1}$;

- the neurons of index $i' \in \{0, S\}$, after firing is inhibited (through $W_{i'i'1}$) by a negative internal loop, thus reset at value negative low enough not to fire anymore before T . We thus generate $Z_{i'}[k]$ as expected.

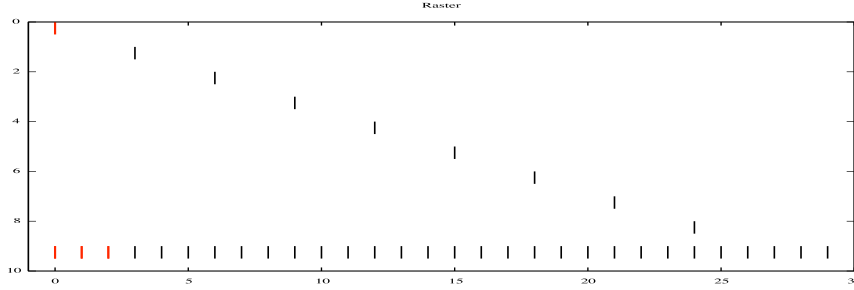


Figure 4: Schematic representation of a sparse trivial set of hidden neurons, allowing to generate any raster of length T .

Alternatively, the use of the firing neuron of index S can be avoided by introducing a constant current $I_{i'k} = W_{i'S1}$.

However, without the firing neuron of index S or some input current, the sparse trivial raster *can not* be generated, although $T < O(ND)$. This comes from the fact that the activity is too sparse to be self-maintained. This illustrates that when stating that “a solution exists, *in the general case*, if the observation period is small enough, i.e., $T < O(ND)$ ”, a set of singular cases, such as this one, was to be excluded.

The hidden neurons reservoir raster being generated, it is straight-forward to generate the output neuron raster, considering:

- no recurrent connection between the N output neurons, i.e., $W_{ijd} = 0, i \in \{1, N\}, j \in \{1, N\}, d \in \{1, D\}$,
- no backward connection from the N output neurons to the S hidden neurons i.e., $W_{i'jd} = 0, i' \in \{0, N\}, j \in \{1, N\}, d \in \{1, D\}$,
- but forward excitatory connections between hidden and output neurons:

$$W_{ij'd} = (1 + \epsilon) \bar{Z}_i[j'D + d] \quad \text{for some small } \epsilon > 0$$

yielding, from (1) :

$$\begin{aligned} V_i[k] &= \sum_{j'=1}^n \sum_{d=1}^D W_{ij'd} Z_{j'}[k-d] \\ &= \sum_{j'=1}^n \sum_{d=1}^D (1 + \epsilon) \bar{Z}_i[j'D - d] \delta(j'D - (k-d)) \\ &= (1 + \epsilon) \bar{Z}_i[k] \end{aligned}$$

setting $\gamma = 0$ for the output neuron and $I_{i'k} = 0$, so that $Z_i[k] = \bar{Z}_i[k]$, i.e., the generated spikes $Z_i[k]$ correspond to the desired $\bar{Z}_i[k]$, as expected.

The linear structure of a network raster

The previous construction allows us to state: *given any raster of N neurons and observation time T , there is always a network of size $N + T/D + 1$ with weights delayed up to D , which exactly simulates this raster.* What do we learn from this fact ?

This helps to better understand how the reservoir computing paradigm works: *Although it is not always possible to simulate any raster plot using a “simple” integrate and fire model such as the one defined in (1), adding hidden neurons allows to embed the problem in a higher-dimensional space where a solution can be found.*

This result is induced by the fact that we have made explicit, in the previous section, that learning the network weights is essentially a linear (L or LP) problem. With this interpretation, a neuron spiking sequence is a vector in this linear space, while a network raster is a vector set. Designing a “reservoir” simply means choosing a set of neurons which spiking activity *spans the space of expected raster*. We are going to see in the next section that this point of view still holds in our framework when considering network inputs.

This linear-algebra interpretation further explains our “trivial sparse” choice: We have simply chosen a somehow canonical orthonormal basis of the raster linear space. One consequence of this view is that, *given a raster, any other raster which is a linear combination of this raster* can be generated by the same network, by a little variation in the weights. This is due to the fact that a set of neurons defining a given raster corresponds to the set of vectors spanning the linear space of all possible rasters generated by this network. Generating another raster corresponds to a simple change of generating vectors in the spanning set. This also allows us to define, for a given raster linear space, a minimal set of generating neurons, i.e. a vector basis. The “redundant” neurons are those which spiking sequence is obtained by feed-forward connections from other neurons.

We must however take care of the fact the numerical values of the vector are binary values, not real numbers. This is a linear space over a finite field, whereas its scalar product is over the real numbers.

On optimal hidden neuron layer design

In the previous paragraph, we have fixed the hidden neuron spiking activity, choosing a sparse ad-hoc activity. It is clearly not the only one solution, very likely not the best one.

Given N output neurons and S hidden neurons, we may consider the following question: which are the “best” weights \mathbf{W} and the hidden neuron activity $Z_{j'}[k]$ allowing to reproduce the output raster.

By “best”, we mean optimal weights estimation with the smaller number of hidden neurons in order to estimate a given spike dynamics. In that case, instead of having to solve a LP-problem, as specified in (8), we have to consider a much more complicated problem now:

- not a linear but bi-linear problem (since we have to consider the products of weights to estimate with spike activity to estimate, as readable in (1));
- not a standard linear programming problem with real values to estimate, but a mixed integer programming problem with both integer values to estimate.

This has a dramatic consequence, since such problem is known as being NP-hard, thus not solvable in practice, as discussed previously for the estimating of delays.

This means that we can not consider this very general question, but must propose heuristics in order to choose or constraint the hidden neuron activity, and then estimate the weights, given the output and hidden neuron’s spikes, in order to still consider a LP-problem.

Let us consider one of such heuristic.

A maximal entropy heuristic

Since we now understand that hidden neuron activity must be chosen in order to span as much as possible the expected raster space, and since we have no a-priori information about the kind of raster we want to reproduce, the natural idea is to randomly choose the neuron activity with a maximal randomness.

Although it is used here at a very simple level, this idea is profound and is related to random sampling and sparse approximation of complex signal in noise (see [45, 46] for a didactic introduction), leading to greedy algorithms and convex relaxation [44, 24]. Since inspired by these elaborated ideas, the proposed method is simple enough to be described without any reference to such formalisms.

In this context, maximizing the chance to consider a hidden neuron with a spiking activity independent from the others, and which adds new independent potential information, simply corresponds to choose the activity “as random as possible”. This corresponds to a so called Bernouilli process, i.e., simply to randomly choose each spike state independently with equi-probability.

Since we want to simulate the expected raster with a minimal number of hidden neuron, we may consider the following algorithmic scheme:

1. Starts with no hidden but only output neurons.
2. Attempts to solve (8) on hidden (if any) and output neurons, in order to obtain weights which allows the reproduction of the expected raster on the output neurons.
3. If the estimation fails, add a new hidden neuron and randomly draw its spiking activity
4. Repeat step 2 and 3 until an exact reproduction of the expected raster is obtained

Clearly, adding more and more random points to the family of generating elements must generate a spanning family after a finite time, since randomly choosing point in an affine space, there is no chance to always stay in a given affine sub-space. This means that we generate a spanning family of neuron after a finite time, with a probability of one. So that the algorithm converges.

What is to be numerically experimented is the fact we likely obtain a somehow minimal set of hidden neurons or not. This is going to be experimented in section 6.

5 Application: Input/Output transfer identification

Let us now describe the main practical application of the previous algorithmic development, which is to “program” a spiking network in order to generate a given spike train or realize a given input/output spike train function.

In the present context, this means finding the “right” or the “best” spiking network parameters in order to map an input’s set onto an output’s set.

From (2) we can write the following equation in order to have a system input/output:

$$V_i[k] = \underbrace{\sum_{j=1}^{No+S} \sum_{d=1}^D W_{ijd} \sum_{\tau=0}^{\tau_{ik}} \gamma^\tau Z_j[k - \tau - d]}_{\text{output + hidden}} + \underbrace{\sum_{l=1}^{Ni} \sum_{d=1}^D W'_{ild} \sum_{\tau=0}^{\tau_{ik}} \gamma^\tau Z'_l[k - \tau - d]}_{\text{input}} + \sum_{\tau=0}^{\tau_{ik}} \gamma^\tau I_i[k - \tau] \quad (11)$$

All variables have been previously defined, observing the equation we can see that it is divided in 3 parts, the first part correspond to output and hidden activity in the system, in the output we will find the activity estimated by the function established from the inputs given, the hidden layer is described below; the second part correspond to the input dynamics, it is basically a spike trains containing all data that will be processed in the function, and finally the third part in the equation correspond to the external current.

From (11) we can deduce a LP problem, where W and W' are the parameters (delayed weights) to estimate.

$$\mathbf{A}_i \mathbf{W}_i + \mathbf{B}_i \mathbf{W}'_i + \mathbf{c}_i > 0$$

thus:

$$\begin{aligned} \mathbf{A}_i &= \begin{pmatrix} \dots & \dots & \dots \\ \dots & (2Z_i[k] - 1) \sum_{\tau=0}^{\tau_{ik}} \gamma^\tau Z_j[k - \tau - d] & \dots \\ \dots & \dots & \dots \\ \dots & \dots & \dots \end{pmatrix} \in R^{N_S(T-D) \times (N+N_h)D} \\ \mathbf{B}_i &= \begin{pmatrix} \dots & \dots & \dots \\ \dots & (2Z_i[k] - 1) \sum_{\tau=0}^{\tau_{ik}} \gamma^\tau Z'_i[k - \tau - d] & \dots \\ \dots & \dots & \dots \\ \dots & \dots & \dots \end{pmatrix} \in R^{N_S(T-D) \times N_i D} \\ \mathbf{c}_i &= (\dots \quad (2Z_i[k] - 1) (\sum_{\tau=0}^{\tau_{ik}} \gamma^\tau i[k - \tau] - 1) \quad \dots)^t \in R^{N_S(T-D)} \end{aligned}$$

On the number of hidden neurons

The number of hidden neurons necessary to do the matching between any input and any output could be calculated knowing the number of unknowns in the Linear Programming system (weights), the available number of equations (spike constraints) and the number of constraints (since we constraint all block of $N \times D$ initial conditions to be the same).

To summarize the system has solutions in the general case with $N = N_i + N_o + S$. In order to estimate the number of hidden neurons S we solve the the last LP problem in terms of N_S, T, D and N and we have:

$$S \geq \frac{T \times N_S}{D} + (N_S - 1) - N_i - N_o$$

Here N_S correspond to the number of samples to teach the system to learn the function.

What is pointed out here, is the fact that the previous formalism does not only apply to the simulation of a unique, input less, fully connected network, but is applicable to a much wider set of problems.

In order to make this explicit, let us consider the following specification of spiking neural networks with units defined by the recurrent equation (1).

connectivity We now assume a partially connected network with a connection graph \mathcal{K} , i.e., some connections weights can be zero.

input current We consider that any neurons can be driven by an input current I_{ik} , defining an “analog” input.

input spikes We have also to consider that the network can also be driven by external incoming spikes.

output neurons We consider that a subset of neurons are output neurons, with *read-out* state that must be constrained, as defined in (5). Other neurons are hidden neurons.

As discussed previously, the best heuristics is to randomly generate the hidden neuron required activity.

weighted estimation We further consider that depending on neuron and time the estimation requirement is not homogeneous, whereas there are times and neurons for which the importance of potential to threshold distance estimation differs from others. This generalized estimation is obvious to introduce in our formalism, defining:

$$|\mathbf{e}_i|_{1,\Lambda} = \sum_k \Lambda_{ik} e_{ik}, \Lambda_{ik} \geq 0 \quad (12)$$

for some metric Λ .

We further consider a “supervised learning paradigm” is the following sense. We now consider a family of L input current or spikes vectors:

$$\mathbf{I}^l = (\dots I_{ik}^l \dots)^t \in R^{N \times T-D},$$

to be mapped on family of output spike trains:

$$\mathbf{Z}^l = (\dots Z_i[k]^l \dots)^t \in R^{N \times T-D},$$

given initial states:

$$\mathbf{Z}_0^l = (\dots Z_i[k]^l \dots)^t \in R^{N \times D}, k \in \{0, D\},$$

for $l \in \{0, L\}$. We would like to find the correct weights \mathbf{w} allowing to perform this input/output mapping. The estimation problem is in fact strictly equivalent to (5), by concatenating the input information (except the initial states). This reads:

$$\begin{aligned} \mathbf{A}_i &= \begin{pmatrix} \dots & \dots & \dots \\ \dots & (2 Z_i[k]^l - 1) \sum_{\tau=\tau_{jk}}^0 \gamma^\tau Z_j[k - \tau - d]^l & \dots \\ \dots & \dots & \dots \end{pmatrix} \in R^{L(T-D) \times ND}, \\ \mathbf{b}_i &= (\dots (2 Z_i[k]^l - 1) (I_{ik\tau}^l - 1) \dots)^t \in R^{L(T-D)}. \end{aligned}$$

This formalism, thus allows us to find an exact input/output mapping, adding hidden neurons in order to have enough degree of freedom to obtain a solution.

Example: IO transfer identification of the “OR” Function

In order to illustrate the parameters estimation in a input-output system we will consider a simple “OR” function. Therefore we have only one spike as output if at least one neuron fire a spike in the precedent time. We have chosen the “OR” function because it has a trivial solution and we can observe the evolution on the weights. The system is training with N_S inputs and theirs respective outputs, where each output is the “OR” function calculated on each input raster; finally is tested with a different input (not used in the learning phase), the output of this input is calculated with the weights estimated using the Eq. (11), here the distance between the estimated output with the weights obtained and the expected output is calculated.

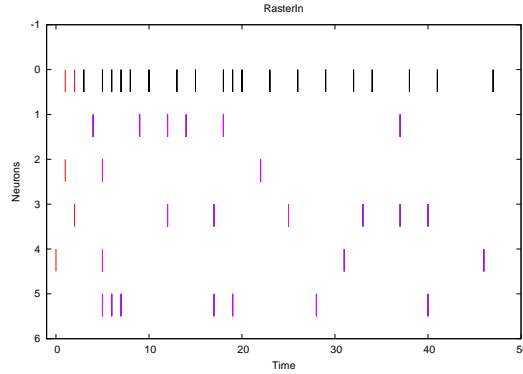


Figure 5: This figure show an Input-Output spike train, the purple lines are the spike times for the input neurons, the black ones are the spike times for the output neuron, which is determined by the or function from the inputs and the red ones are the initial conditions used to estimate the parameters.

5.1 Application: approximate Input/Output transfer identification

Let us finally discuss how to apply the previous formalism to approximate transfer identification. We consider, in our context, deterministic alignment metrics defined on spike times [49, 48].

Using alignment metric. In this context, the distance between two finite spike trains \mathcal{F} , \mathcal{F}' is defined in terms of the minimum cost of transforming one spike train into another. Two kinds of operations are defined:

- spike insertion or spike deletion, the cost of each operation being set to 1
- spike shift, the cost to shift from $t_i^n \in \mathcal{F}$ to $t_i^m \in \mathcal{F}'$ being set to $|t_i^n - t_i^m|/\tau$ for a time constant $\tau \geq 1$.

Although computing such a distance seems subject to a combinatorial complexity, it appears that quadratic algorithms are available, with the same complexity in the discrete and continuous case.

For small τ , the distance approaches the number of non-coincident spikes, since instead of shifting spikes has a lower cost than insert/delete non-coincident spikes, the distance being always bounded by the number of spikes in both trains.

For high τ , the distance basically equals the difference in spike number (rate distance), while for two spike trains with the same number of spikes, there is always a time-constant τ large enough for the distance to be equal to $\sum_n |t_i^n - t_i^n|/\tau$.

It appears that the LP algorithms in initial phase [17, 6] which attempt to find an initial solution before optimizing it, generates a divergence between the obtained and expected raster, this divergence being zero if a solution exists. Furthermore, this divergence can be related to the present alignment metric, for high τ , and on-going work out of the scope of this subject develops this complementary aspect.

When considering spike trains, an approach consists summing the distances for each alignment spike-to-spike. Another point of view is to consider that a spike can “jump”, with some cost, from one spike in \mathcal{F} to another spike in \mathcal{F}' . The related algorithmic complexity is no more quadratic but to the power of the number of spikes [4].

This family of metrics include alignments not only on spike times, but also on inter-spike intervals, or metrics which are sensitive to patterns of spikes, etc... They have been fruitfully applied to a variety of neural systems, in order to characterize neuronal variability and coding [48]. For instance, in a set of neurons that act as coincidence detectors with integration time (or temporal resolution) τ , spike trains have other post-synaptic effects if they are similar w.r.t. this metric. Furthermore, this metric generalizes to metric with causality (at a given time, the cost of previous spikes alignment decreases with the obsolescence of the spike) and non-linear shift's costs [10].

Application to approximate identification. Our proposal is to re-visit the maximal entropy heuristic algorithm defined previously and consider having a correct identification, if the distance between the expected and obtained raster is not zero but *lower than some positive threshold*.

This allows us to not only address:

The *exact* estimation problem, i.e. to find an exact input/output mapping if and only if there is one, but also the *approximate* estimation problem, i.e. to find an approximate input/output mapping that minimizes a given distance.

This however, is a trivial strategy because the alignment distance is not used to find the optimal solution, but only to check whether this solution is acceptable. The reason of such a choice is easy to explain: alignment metrics, as it, are highly non-differentiable with respect to the network parameters. Therefore variational methods, considering. e.g., the criterion gradient in order to drive the parameters value towards a local optimum do not apply here.

Several alternatives exist. One considers the spike time defined by the equation $V_{\mathbf{W}}(t_i^n) = \theta$, where V is the membrane potential, θ is the firing threshold, and \mathbf{W} are the network parameters to adjust [38]. From the implicit function theorem it is obvious to relate locally $d\mathbf{W}$ and dt_i^n , thus derive a parameter gradient with respect to the spike times. However, such method is numerically ill-defined, since the threshold value θ is not significant, but only a modeling trick.

Another alternative is to consider convolution metrics [11], in order to relate the spike train to a differentiable signal, thus in a context where variational methods can be applied. One instantiation of this idea considers an abstract specification of the input/output methods, using piece-wise linear SRM models [21]. This algebraic simplification allows us to implement variational methods [51, 27] in order to specify the network input/output relations. Such methods, however, are restrained to a given neuronal model and to a given coding (temporal coding in this case) of a continuous information using spike times. Such methods must thus be generalized in order to be applied in the present context.

When the present method fails, it still provides an approximate solution with a “maximal” number of correct spikes, by the virtue of the (8) minimization mechanism. Each “false” state corresponds to $e_{ik} < 0$ (i.e. either a spurious spike or a missing spike), and it is an easy exercise to relate this to a particular alignment metric. We conjecture that this is an interesting track in order to generalize the present work, from exact estimation, to exact or approximate estimation.

6 Results

6.1 Retrieving weights from the observation of spikes and membrane potential

In a first experiment, we consider the linear problem defined in (3) and use the singular value decomposition (SVD) mechanism [20] in order to obtain a solution in the least-square sense. Here the well-established `GSL`³ library SVD implementation is used.

This allows us to find:

- if one or more solution, the weights of minimal magnitude $|\mathbf{w}_i|^2 = \sum_{jd} W_{ijd}^2$;
- if no exact solution, the solution which minimizes $\sum_k (V_i[k] - \tilde{V}_i[k])^2$ where $\tilde{V}_i[k]$ is the membrane potential predicted by the estimation.

The master and servant paradigm.

We have seen that, if $D(N+1) > T$, i.e., if the observation time is small enough for any raster there is a solution. Otherwise, there is a solution if the raster is generated by a model of the form of (1). We follow this track here and consider a master/servant paradigm, as follows:

1. In a first step we randomly choose weights and generate a “master” raster.
2. The corresponding output raster is submitted to our estimation method (the “servant”), while the master weights are hidden. The weights are taken from a normal distribution $\mathcal{N}(0, \frac{\sigma^2}{N})$ with 70% excitatory connections and 30% for inhibitory one. The standard deviation $\sigma \in [1, 10]$ has been chosen in order to obtain an interesting dynamics, as discussed in [8].

The algorithm defined in (4) or in (8) has a set of spikes as input for which we are sure that a solution exists. Therefore it can be used and leads to a solution with a raster which must exactly correspond to the master input raster.

Note that this does not mean that the servant is going to generate the raster with the same set of weights, since several solutions likely exist in the general case. Moreover, except for the paradigm (4), the master and servant potential $V_i[k]$ are expected to be different, since we attempt to find potentials which distance to the threshold is maximal, in order to increase the numerical robustness of the method.

This is going to be the validation test of our method. As an illustration we show two results in Fig. 6 and Fig. 7 for two different dynamics. The former is “chaotic” in the sense that the period is higher than the observation time.

In the non trivial case in Fig. 6, it is expected that only one weight’s set can generate such a non-trivial raster, since, as discussed before, we are in the “full rank” case, thus with a unique solution. We observe the same weights for both master and servant in this case, as expected. This would not be the case for simpler periodic raster, e.g. in Fig. 7, where the weight’s estimation by the servant differs from the master’s weights, since several solutions are possible.

Retrieving weights from the observation of spikes *and* membrane potential has been introduced here in order to explain and validate the general method in a easy to explain case. Let us now turn to the more interesting cases where only the observation of spikes are available.

³<http://www.gnu.org/software/gsl>

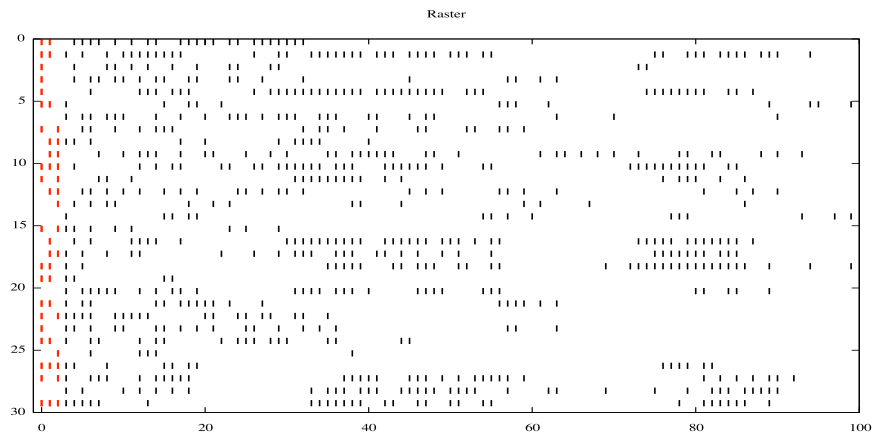


Figure 6: A “chaotic” dynamics with 30 neurons fully connected within network, initial conditions $D = 3$ and observation time $T = 100$, using both excitatory (70%) and inhibitory (30%) connections, with $\sigma = 5$ (weight standard-deviation). After estimation, we have checked that master and servant generate **exactly the same raster plot, thus only show the servant raster, here and in the next figures.**

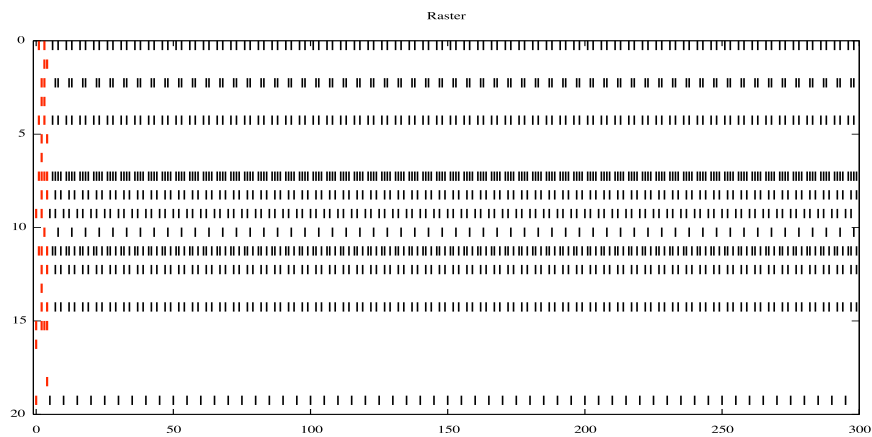


Figure 7: A “periodic” (58.2 periods of period 5) dynamics with 20 neurons fully connected within network and observation time $T = 300$, using both excitatory (70%) and inhibitory (30%) connections, with $\sigma = 1$. Again the master and servant rasters are the same.

6.2 Retrieving weights from the observation of spikes

In this setup we still use the master / servant paradigm, but now consider the LP problem defined previously. The numerical solutions are derived thanks to the well-established improved simplex method as implemented in GLPK⁴.

As an illustration we show two results in Fig. 8 and Fig. 9 for two different dynamics.

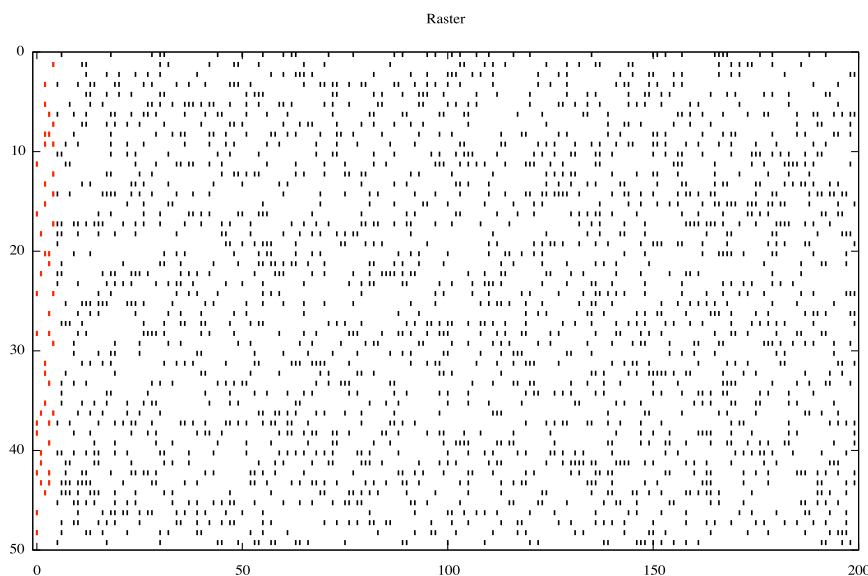


Figure 8: Example of rather complex “chaotic” dynamics retrieved by the LP estimation defined in (8) using the master / servant paradigm with 50 neurons fully connected, initial conditions $D = 3$ and observation time $T = 200$, used here to validate the method.

Interesting is the fact that, numerically, the estimated weights correspond to a parsimonious dynamics in the sense that the servant raster period tends to be minimal:

- if the master raster appears periodic, the servant raster is, with same period;
- if the master raster appears aperiodic (i.e., “chaotic”) during the observation interval, the servant raster is periodic with a period close (but not always identical) to the observation time T . This is coherent with the theoretical analysis of such networks [8, 13], and is further investigated in a next paper.

6.3 Retrieving delayed weights from the observation of spikes

In this next setup we still consider the same master/servant paradigm, for $N = 50$ units, with a leak $\gamma = 0.95$ and an external current $I = 0.3$, but in this case the master delayed weight profile has the standard form shown in Fig. 10.

⁴<http://www.gnu.org/software/glpk>

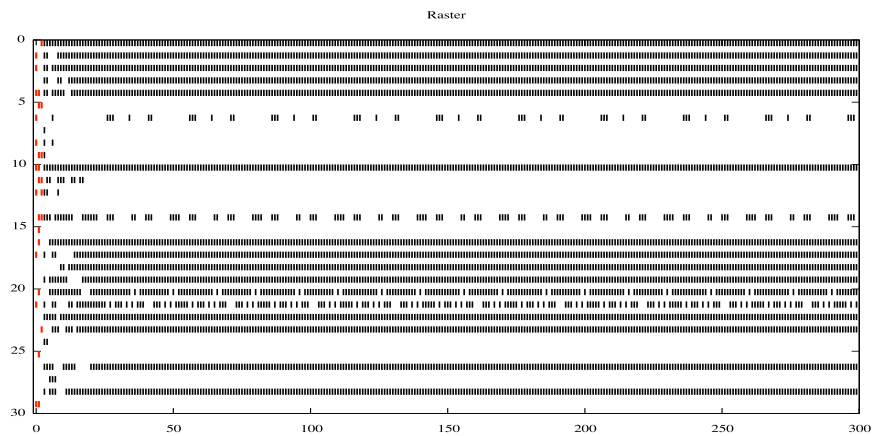


Figure 9: Example of periodic dynamics retrieved by the LP estimation defined in (8) using the master / servant paradigm, here a periodic raster of period 30 is observed during 8.3 periods. ($N = 30$, $T = 300$ and $D = 3$) As expected from by the theory, the estimated dynamics remains periodic after the estimation time, thus corresponding to a parsimonious estimation.

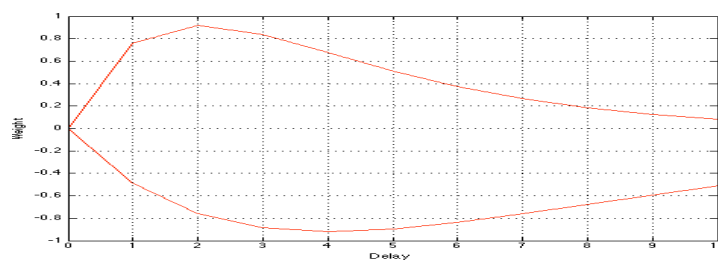


Figure 10: Weights distribution (positive and negative) used to generate delayed weights, with $D = 10$.

In the case of trivial dynamics, it is observed that the estimated servant weights distribution is sparse as illustrated in Fig. 11. However, as soon as the dynamics is non trivial, the proposed algorithm uses all delayed weight parameters in order to find an optimal solution, without any correspondence between the master initial weight distribution and the servant estimated weight distribution. This is illustrated in Fig. 13, where instead of the standard profiles shown in Fig. 10, a “Dirac” profile has been used in the master, while the estimated weights are distributed at all possible delays. In order to complete this illustration a non trivial dynamics is shown in Fig. 12.

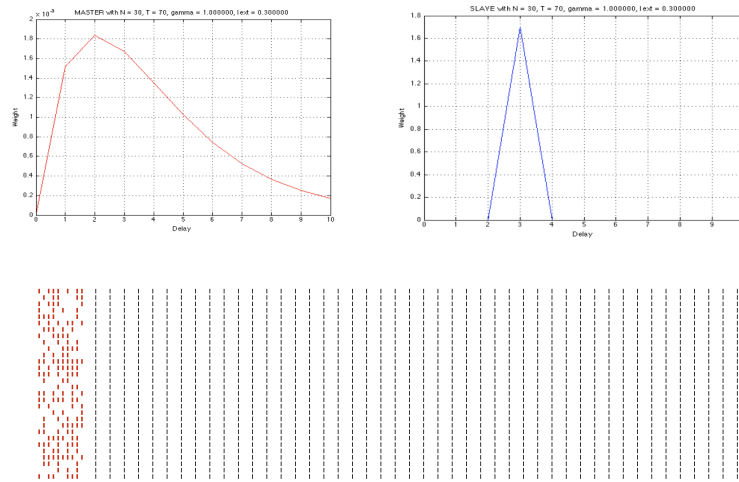


Figure 11: An example of trivial dynamics obtained with excitatory weights profiles shown in the top-left view (master weight’s profile), with $N = 30$, $\gamma = 0.98$, $D = 10$, $T = 70$. The estimated weights profile (servant weight’s profile) is shown in the top-right view. To generate such trivial periodic raster, shown in the bottom view, only weights with a delay equal to the period have non zero values. This corresponds to a minimal edge of the estimation simplex, this parsimonious estimation being a consequence of the chosen algorithm.

On the complexity of the generated network

Here we maximize (7) which is in relation with (6). The latter was shown to be a relevant numerical measure of the network complexity [8, 13]. We thus obtain, among networks which generate exactly the required master, the “less complex” network, in the sense of (7). A very simple way to figure out how complex is the servant network is to observe its generated raster after T , i.e., after the period of time where it matches exactly the required master’s raster. They are indeed the same before T .

After T , in the case of delayed weights, we still observe that if the original raster is periodic, the generated raster is still periodic with the same period.

If the original raster is aperiodic, for small N and T , we have observed that the generated master is still periodic, as illustrated in Fig. 14. We however, have not observed any further regularity, for instance changes of regime can occur after the T delay, huge period can be observed for relatively small numbers of N and T , etc.. Further investigating this aspect is a perspective of the present work.

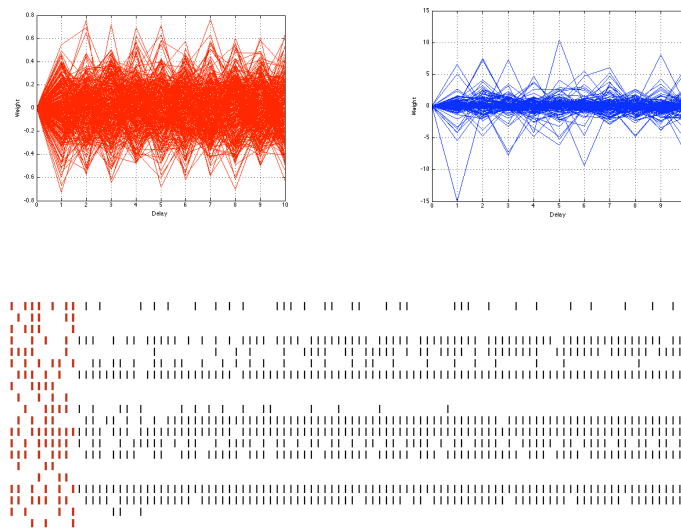


Figure 12: An example of non-trivial dynamics, with $N = 30$, $\gamma = 0.98$, $D = 10$, $T = 100$. Profiles corresponding to the master's excitatory profiles are superimposed in the top-left figure, those corresponding to the master's inhibitory profiles are superimposed in the top-right figure. The estimated raster is shown in the bottom view. This clearly shows that, in the absence of additional constraint, the optimal solution corresponds to wide distribution of weight's profiles.

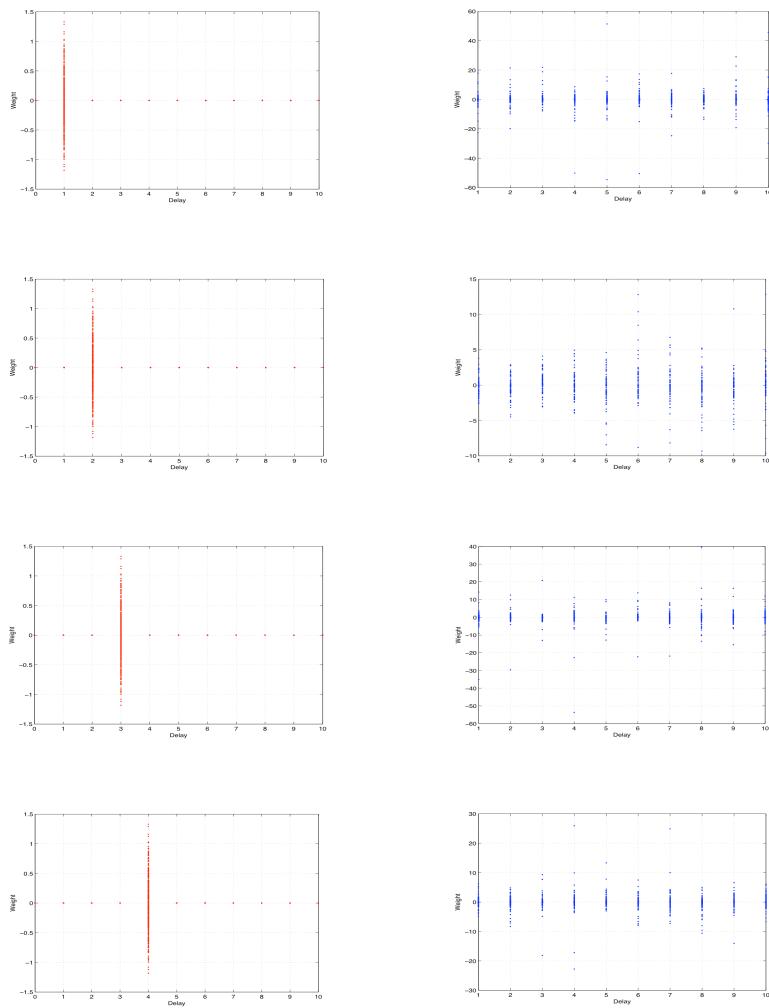
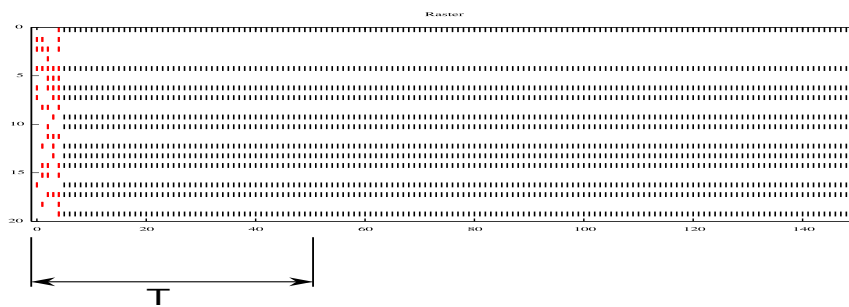
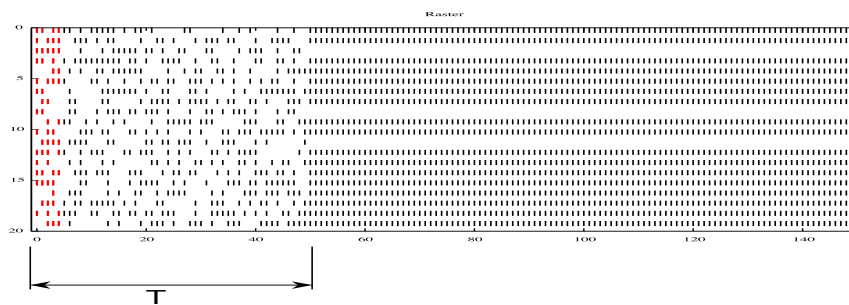


Figure 13: In this figure we show that whatever be the weights and delays in the master (left), with $N = 20$, $\gamma = 0.98$, $D = 10$ $T = 100$, the estimator use all the weights and delays for calculate the raster, in order to obtain an optimal solution.



(a)



(b)

Figure 14: Two examples of observation of the raster period, on the slave network, observing the raster after the time T where it matches the master raster (shown by an arrow in the figure). (a) With $N = 20$, $\gamma = 0.98$, $\sigma = 5$, $D = 5$, $T = 50$, a periodic regime of period $P = 1$ is installed after a change in the dynamics. (b) With $N = 20$, $\gamma = 0.98$, $\sigma = 5$, $D = 5$, $T = 50$, a periodic regime of period $P = 1$ corresponds to the master periodic regime.

6.4 Retrieving delayed weights from the observation of spikes, using hidden units

In this last set of numerical experiments we want to verify that the proposed method in section 4 is “universal” and allows one to evaluate the number of hidden neurons to be recruited in order to exactly simulate the required raster. If it is true, this means that we have here available a new “reservoir computing” mechanism.

Considering Bernoulli distribution

We start with a completely random input, drawn from a uniform Bernoulli distribution. This corresponds to an input with maximal entropy. Here the master/servant trick is no more used. Thus, the raster to reproduce has no chance to verify the neural network dynamics constraints induced by (1), unless we add hidden neurons as proposed in section 4.

As observed in Fig. 15, we obtain as expected an exact reconstruction of the raster, while as reported in Fig. 16, we need an almost maximal number of hidden neurons for this reconstruction, as expected since we are in a situation of maximal randomness,

Considering correlated distributions

We now consider a correlated random input, drawn from a Gibbs distribution [14, 9]. To make it simple, the raster input is drawn from a Gibbs distribution, i.e. a parametrized rank R Markov conditional probability of the form:

$$P(\{Z_i[k], 1 \leq i \leq N\} | \{Z_i[k-l], 1 \leq i \leq N, 1 \leq l < R\}) = \frac{1}{Z} \exp(\Phi_\lambda(\{Z_i[k-l], 1 \leq i \leq N, 0 \geq l > -R\}))$$

where $\Phi_\lambda()$ is the related Gibbs potential parametrized by λ and Z a normalization constant.

This allows to test our method on highly-correlated rasters. We have chosen a potential of the form:

$$\Phi_\lambda(Z|_{k=0}) = r \sum_{i=1}^N Z_i[0] + C_t \sum_{i=1}^N \prod_{l=0}^R Z_i[l] + C_i \prod_{l=0}^R Z_i[0]$$

thus with a term related to the firing rate r , a term related to temporal correlations C_t , and a term related to inter-unit correlation C_i .

We obtain a less intuitive result in this case, as illustrated in Fig. 17: event strongly correlated (but aperiodic) rasters are reproduced only if using as many hidden neurons as in the non-correlated case. In fact we have drawn the number S of hidden neurons against the observation period T randomly selecting 45000 inputs and have obtained the same curve as in Fig. 16.

This result is due to the fact that since the raster is aperiodic, non predictable changes occur in the general case, at any time. The raster must thus be generated by a maximal number of degrees of freedom, as discussed in the previous sections.

In order to further illustrate this aspect, we also show in Fig. 18 how a very sparse raster is simulated. We again obtain a solution with the same ratio of hidden neurons. Clearly the number of hidden neurons could have been less, as discussed in the previous sections. This shows that the algorithm is very general, but not optimal in terms of number of hidden neurons.

Considering biological data

As a final numerical experiment, we consider two examples of biological data set borrowed from [1] by the courtesy of the authors. Data are related to spike synchronization

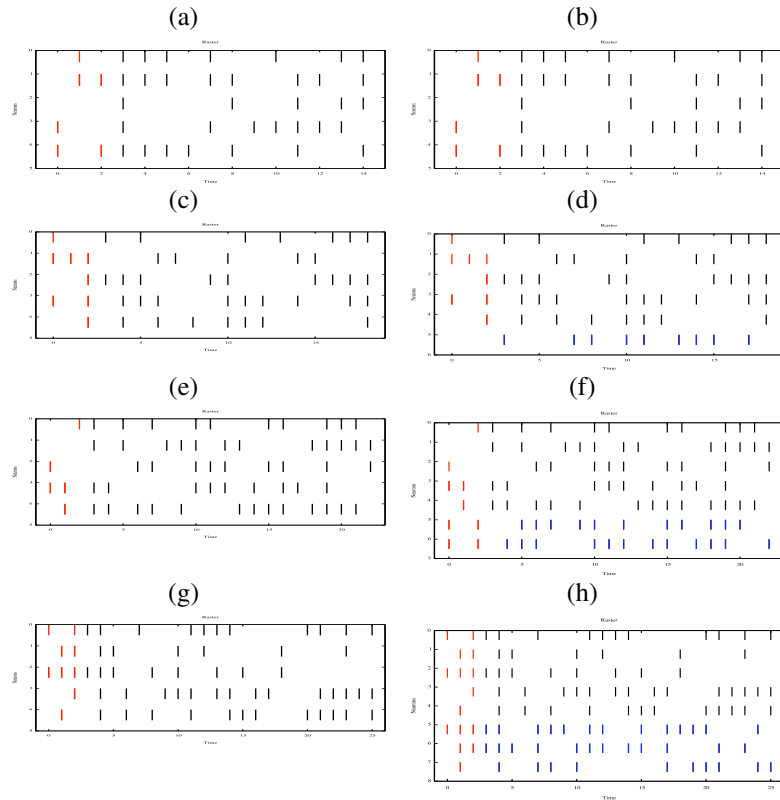


Figure 15: Finding the expected dynamics from a raster with uniform distribution. (a), (c), (e) and (g) correspond to different raster with Bernoulli Distribution in addition (b), (d), (f) and (h) show the raster calculated by the methodology proposed. The red lines correspond to initial conditions (initial raster), the black ones are the spikes calculated by the method and the blues ones are the spikes in the hidden layer obtained with a Bernoulli Distribution. We can also observe that the number of neurons in the hidden layer increases, 1 by 1, between (b), (d), (f) and (h), this is because the observation time T is augmented by 4, as predicted. Here $N = 5$, $\gamma = 0.95$, $D = 3$; in (a)(b) $T = 15$ with $S = 0$, in (c)(d) $T = 19$ with $S = 1$, in (e)(f) $T = 23$ with $S = 2$, in (g)(h) $T = 27$ with $S = 3$, while S correspond to the number of neurons in the hidden layer, detailed in the text.

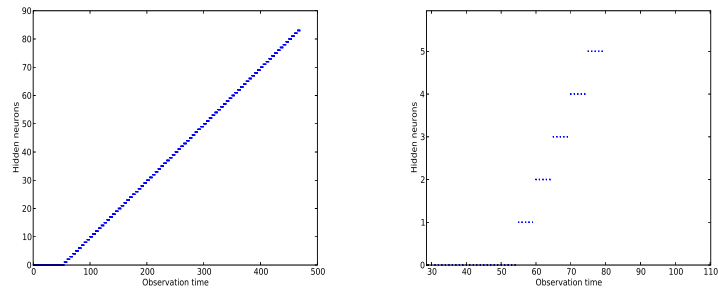


Figure 16: Relationship between the number of hidden neurons S and the observation time T , here $N = 10$, $T = 470$, $D = 5$, $\gamma = 0.95$ for this simulation. The right-view is a zoom of the left view. This curves shows the required number of hidden neurons, using the proposed algorithm, in order to obtain an exact simulation. We observe that $S = \frac{T}{D} - N$, thus that an almost maximal number of hidden neuron is required. This curve has been drawn from 45000 independent randomly selected inputs.

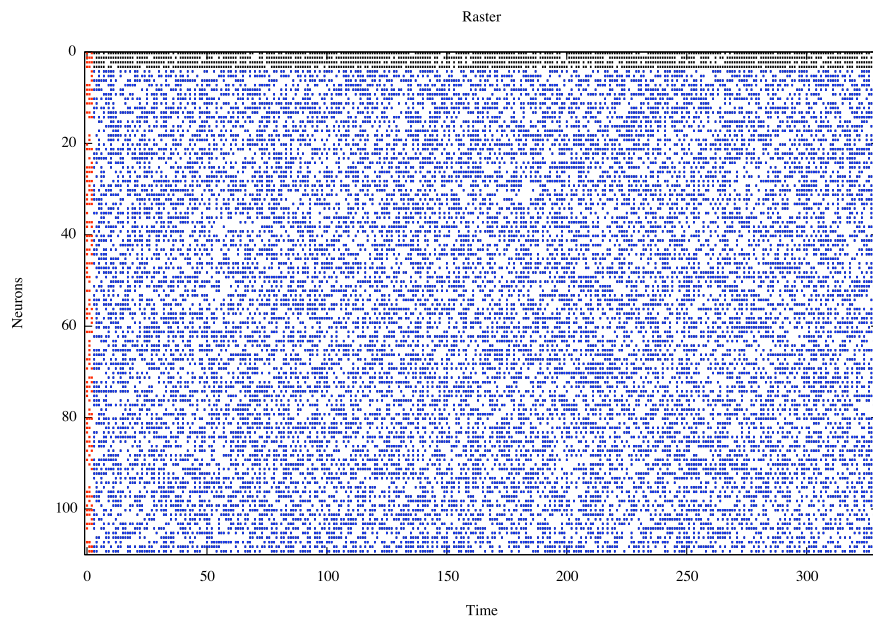


Figure 17: Raster calculated, by the proposed method, from a highly correlated Gibbs distribution. Here $r = -1$, $C_t = -0.5$ and $C_i = -1$. The other parameters are $N = 4$, $\gamma = 0.95$, $D = 3$, $T = 330$ with $S = 106$. The red lines correspond to initial conditions (initial raster), the black ones are the input/output spikes and the blues ones are the spikes in the hidden layer.

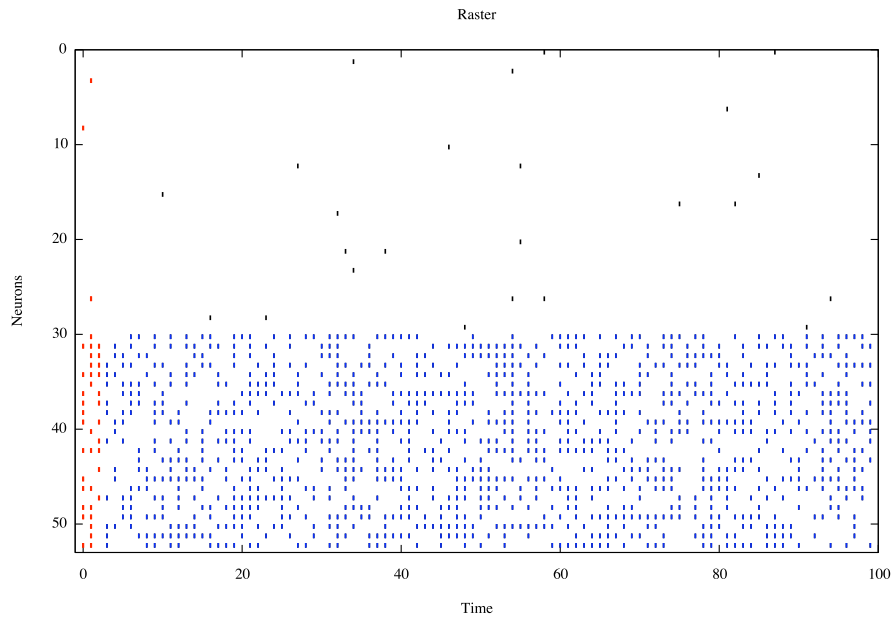


Figure 18: Raster calculated, by the proposed method, from a very sparse raster, with $N = 30$, $\gamma = 0.95$, $D = 3$, $T = 100$ and $S = 23$. The hidden neurons derived by the present algorithm simply allow to maintain the network activity in order to fire the sparse spikes at the right time. Color codes are the same as previously.

in a population of motor cortical neurons in the monkey, during preparation for movement, in a movement direction and reaction time paradigm. Raw data are presented trial by trial (without synchronization on the input), for different motion directions and the input signal is not represented, since meaningless for the purpose. Original data resolution was $0.1ms$ while we have considered a $1ms$ scale here.

What is interesting here is that we can apply the proposed method on non-stationary rasters, qualitatively very different, such as a very sparse raster, similar to the one shown in Fig. 18, a raster with two activity phases (presently movement preparation and execution) in Fig. 19 and a raster with a rich non-stationary activity in Fig. 20. In fact a dozen of such data sets have been tested, with the same result: exact raster reconstruction, with the same hidden unit ratio.

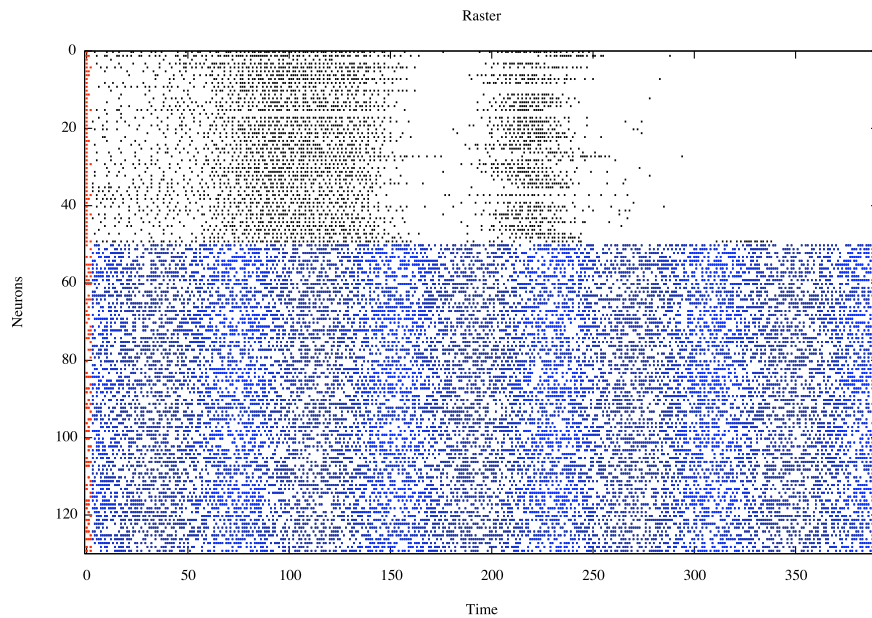


Figure 19: Raster calculated, by the proposed method, from a biological data set, with $N = 50$, $\gamma = 0.95$, $D = 3$ $T = 391$ and $S = 80$. From [1] by the courtesy of the authors.



Figure 20: Raster calculated, by the proposed method, from a biological data set, with $N = 50$, $\gamma = 0.95$, $D = 5$ $T = 291$ and $S = 8$. From [1] by the courtesy of the authors.

On the computation time

Since the computation time is exclusively the LP problem resolution computation time we have simply verify that we roughly obtain what is generally observed with this algorithm, i.e. that the computation time order of magnitude is:

$$O(ST)$$

when $N \ll T$, which is the case in our experimentation. On a standard laptop computer, this means a few seconds.

6.5 Input/Output estimation

In this section we present results on the Input/Output matching, the objective is to find the parameters (delayed weights) for a transfer function and demonstrate that the methodology proposed in this work is also capable to learn certain functions in order to approximate input-output functions.

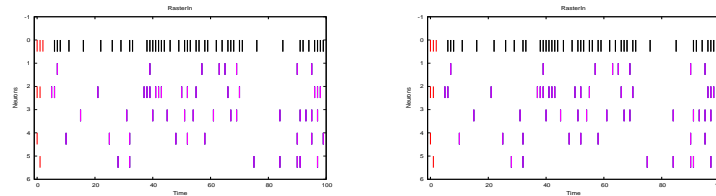


Figure 21: Input-Output dynamics matching, the purple lines represent the input dynamics, the black ones are the or function of the inputs, it means that if at least one of the input neurons fire a spike in t the output fire a spike in $t + 1$, finally the red ones represent the initial conditions. $N_i = 5$, $N_o = 1$, $D = 0$, $T = 100$ and $S = 6$. Exact matching (diff = 0).

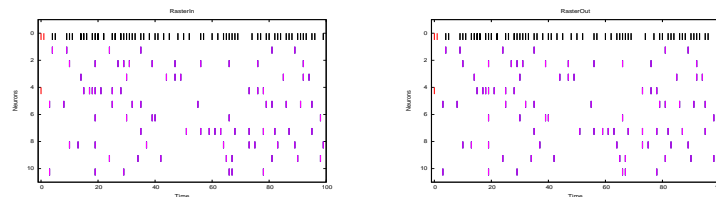


Figure 22: Input-Output dynamics matching, the purple lines represent the input dynamics, the black ones are the or function of the inputs, it means that if at least one of the input neurons fire a spike in t the output fire a spike in $t + 1$, finally the red ones represent the initial conditions. $N_i = 10$, $N_o = 1$, $D = 0$, $T = 100$ and $S = 10$. Approximate matching (diff = 2).

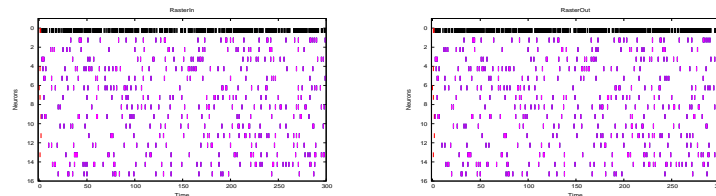


Figure 23: Input-Output dynamics matching, the purple lines represent the input dynamics, the black ones are the or function of the inputs, it means that if at least one of the input neurons fire a spike in t the output fire a spike in $t + 1$, finally the red ones represent the initial conditions. $N_i = 15$, $N_o = 1$, $D = 0$, $T = 300$ and $S = 15$. Approximate matching (diff = 21).

7 Conclusion

Considering a deterministic time-discretized spiking network of neurons with connection weights having delays, we have been able to investigate in details to which extend it is possible to back-engineer the networks parameters, i.e., the connection weights. Contrary to the known NP-hard problem which occurs when weights and delays are to be calculated, the present reformulation, now expressed as a Linear-Programming (LP) problem, provides an efficient resolution and we have discussed extensively all the potential applications of such a mechanism, including regarding what is known as reservoir computing mechanisms, with or without a full connectivity, etc..

At the simulation level, this is a concrete instantiation of the fact rasters produced by the simple model proposed here, can produce any rasters produced by more realistic models such as Hodgkin-Huxley, for a finite horizon. At a theoretical level, this property is reminiscent of the shadowing lemma of dynamical systems theory [26], stating that chaotic orbits produced by a uniformly hyperbolic system can be approached arbitrary close by periodic orbits.

At the computational level, we are here in front of a method which allows to “program” a spiking network, i.e. find a set of parameters allowing us to exactly reproduce the network output, given an input. Obviously, many computational modules where information is related to “times” and “events” are now easily programmable using the present method. A step further, if the computational requirement is to both consider “analog” and “event” computations, the fact that we have studied both the unit analog state and the unit event firing reverse-engineering problems (corresponding to the L and LP problems), tends to show that we could generalize this method to networks where both “times” and “values” have to be taken into account. The present equations are to be slightly adapted, yielding to a LP problem with both equality and inequality constraints, but the method is there.

At the modeling level, the fact that we do not only statistically reproduce the spiking output, but reproduce it *exactly*, corresponds to the computational neuroscience paradigm where “each spike matters” [22, 18]. The debate is far beyond the scope of the present work, but interesting enough is the fact that, when considering natural images, the primary visual cortex activity seems to be very sparse and deterministic, contrary to what happens with unrealistic stimuli [5]. This means that it is not a nonsense to address the problem of estimating a raster exactly.

As far as modeling is concerned, the most important message is in the “delayed weights design: the key point, in the present context is not to have one weight or one weight and delay but several weights at different delays”. We have seen that this increase the computational capability of the network. In other words, more than the connection’s weight, the connection’s profile matters.

Furthermore, we point out how the related LP adjustment mechanism is distributed and has the same structure as an “Hebbian” rule. This means that the present learning mechanism corresponds to a local plasticity rule, adapting the unit weights, from only the unit and output spike-times. It has the same architecture as another spike-time dependent plasticity mechanism. However, this is supervised learning mechanisms, whereas usual STDP rules are unsupervised ones, while the rule implementations is entirely different.

To which extends this LP algorithm can teach us something about how other plasticity mechanisms is an interesting perspective of the present work. Similarly, better understanding the dynamics of the generated networks is another issue to investigate, as pointed out previously.

We consider the present approach as very preliminary and point out that it must be further investigated at three levels:

- i *optimal number of hidden units*: we have now a clear view of the role of these hidden units, used to span the linear space corresponding to the expected raster, as detailed in section 4. This opens a way, not only to find a correct set of hidden units, but a minimal set of hidden units. This problem is in general NP-hard, but efficient heuristics may be found considering greedy algorithms. We have not further discussed this aspect in this paper, because quite different non trivial algorithms have to be considered, with the open question of their practical algorithmic complexity. But this is an ongoing work.
- ii *approximate raster matching*: we have seen that, in the deterministic case using, e.g., alignment metric, approximate matching is a much more challenging problem, since the distance to minimize are not differentiable, thus not usable without a combinatorial explosion of the search space. However, if we consider other metric (see [38, 10] for a review), the situation may be more easy to manage, and this is to be further investigated.
- iii *application to unsupervised or reinforcement learning*: though we deliberately have considered, here, the simplest paradigm of supervised learning in order to separate the different issues, it is clear that the present mechanism must be studied in a more general setting of, e.g., reinforcement learning [41], for both computational and modeling issues. Since the specification is based on a variational formulation, such a generalization considering criterion related to other learning paradigms, seems possible to develop.

Though we are still far from solving the three issues, the present study is completed in the sense that we not only propose theory and experimentation, but a true usable piece of software⁵.

⁵ Source code available at <http://enas.gforge.inria.fr>.

References

- [1] A. Riehle A, F. Grammont, M. Diesmann, and S. Grn. Dynamical changes and temporal precision of synchronized spiking activity in monkey motor cortex during movement preparation. *J. Physiol (Paris)*, 94:569–582, 2000.
- [2] D. J. Albers and J. C. Sprott. Routes to chaos in high-dimensional dynamical systems : A qualitative numerical study. *Physica D*, 223:194–207, 2006.
- [3] D. J. Albers and J. C. Sprott. Structural stability and hyperbolicity violation in high-dimensional dynamical systems. *Non linearity*, 19:1801–1847, 2006.
- [4] Dmitriy Aronov. Fast algorithm for the metric-space analysis of simultaneous responses of multiple single neurons. *Journal of Neuroscience Methods*, 124(2), 2003.
- [5] P. Baudot. *Nature is the code: high temporal precision and low noise in VI*. PhD thesis, Univ. Paris 6, 2007.
- [6] Robert E. Bixby. Implementing the simplex method: The initial basis. *J. on Computing*, 4(3), 1992.
- [7] S. M. Bohte and M. C. Mozer. Reducing the variability of neural responses: A computational theory of spike-timing-dependent plasticity. *Neural Computation*, 19(2):371–403, 2007.
- [8] B. Cessac. A discrete time neural network model with spiking neurons. rigorous results on the spontaneous dynamics. *J. Math. Biol.*, 56(3):311–345, 2008.
- [9] B. Cessac, H.Rostro-Gonzalez, J.C. Vasquez, and T. Viéville. Statistics of spikes trains, synaptic plasticity and gibbs distributions. In *Neurocomp 2008*, 2008.
- [10] B. Cessac, H.Rostro-Gonzalez, J.C. Vasquez, and T. Viéville. To which extend is the "neural code" a metric ? In *Neurocomp 2008*, 2008.
- [11] B. Cessac, H. Rostro, J.-C. Vasquez, and T. Viéville. To which extend is the "neural code" a metric ? In *Deuxième conférence française de Neurosciences Computationnelles*, 2008.
- [12] B. Cessac and T. Viéville. Introducing numerical bounds to improve event-based neural network simulation. *Frontiers in neuroscience*, 2008. submitted.
- [13] B. Cessac and T. Viéville. On dynamics of integrate-and-fire neural networks with adaptive conductances. *Frontiers in neuroscience*, 2(2), jul 2008.
- [14] J.R. Chazottes, E. Floriani, and R. Lima. Relative entropy and identification of gibbs measures in dynamical systems. *J. Statist. Phys.*, 90(3-4):697–725, 1998.
- [15] G. Chechik. Spike-timing-dependent plasticity and relevant mutual information maximization. *Neural Computation*, 15(7):1481–1510, 2003.
- [16] L.N. Cooper, N. Intrator, B.S. Blais, and H.Z. Shouval. *Theory of Cortical Plasticity*. World Scientific Publishing, 2004.
- [17] Richard B. Darst. *Introduction to Linear Programming Applications and Extensions*. Marcel Dekker Ltd, New-York, 1990.
- [18] A. Delorme, L. Perrinet, and S. Thorpe. Network of integrate-and-fire neurons using rank order coding b: spike timing dependant plasticity and emergence of orientation selectivity. *Neurocomputing*, 38:539–545, 2001.
- [19] Alain Destexhe. Conductance-based integrate and fire models. *Neural Computation*, 9:503–514, 1997.
- [20] F.R. Gantmacher. *Matrix Theory*. Chelsea, New-York, 1977.
- [21] W. Gerstner and W. M. Kistler. Mathematical formulations of hebbian learning. *Biological Cybernetics*, 87:404–415, 2002.
- [22] R. Guyonneau, R. vanRullen, and S.J. Thorpe. Neurons tune to the earliest spikes through stdp. *Neural Computation*, 2004. In review.
- [23] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2:359–366, 1989.
- [24] M. J. Strauss J. A. Tropp, A. C. Gilbert. Algorithms for simultaneous sparse approximation. part i: Greedy pursuit. *Signal Processing*, 86:572–588, 2006.
- [25] H. Jaeger. Adaptive nonlinear system identification with Echo State Networks. In S. Becker, S. Thrun, and K. Obermayer, editors, *NIPS*2002, Advances in Neural Information Processing Systems*, volume 15, pages 593–600. MIT Press, 2003.
- [26] A. Katok and B. Hasselblatt. *Introduction to the modern theory of dynamical systems*. Kluwer, 1998.

- [27] P. Kornprobst, T. Vieville, S. Chemla, and O. Rochel. Modeling cortical maps with feed-backs. In *29th European Conference on Visual Perception*, page 53, aug 2006.
- [28] J.W. Pillow L. Paninski and E.P. Simoncelli. Maximum likelihood estimation of a stochastic integrate-and-fire neural encoding model. *J. Neurophysiol*, 92:959–976, 2004.
- [29] W. Maass. Fast sigmoidal networks via spiking neurons. *Neural Computation*, 9:279–304, 1997.
- [30] W. Maass. On the relevance of time in neural computation and learning. *Theoretical Computer Science*, 261:157–178, 2001. (extended version of ALT’97, in LNAI 1316:364-384).
- [31] W. Maass and T. Natschläger. Networks of spiking neurons can emulate arbitrary hopfield nets in temporal coding. *Neural Systems*, 8(4):355–372, 1997.
- [32] W. Maass, T. Natschläger, and H. Markram. Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural Computation*, 14(11):2531–2560, 2002.
- [33] Wolfgang Maass and Christopher M. Bishop, editors. *Pulsed Neural Networks*. MIT Press, 2003.
- [34] H el ene Paugam-Moisy, R egis Martinez, and Samy Bengio. Delay learning and polychronization for reservoir computing. *Neurocomputing*, 71:1143–1158, 2008.
- [35] T. J. Lewis R. Jolivet and W. Gerstner. Generalized integrate-and-fire models of neuronal activity approximate spike trains of a detailed model to a high degree of accuracy. *J. Neurophysiol*, 92:959–976, 2004.
- [36] M. Rudolph and A. Destexhe. Analytical integrate and fire neuron models with conductance-based dynamics for event driven simulation strategies. *Neural Computation*, 18:2146–2210, 2006.
- [37] Anton Maximilian Sch afer and Hans Georg Zimmermann. Recurrent neural networks are universal approximators. *Lecture Notes in Computer Science*, 4131:632–640, 2006.
- [38] Benjamin Schrauwen. *Towards Applicable Spiking Neural Networks*. PhD thesis, Universiteit Gent, Belgium, 2007.
- [39] H. Soula and C. C. Chow. Stochastic dynamics of a finite-size spiking neural networks. *Neural Computation*, 19:3262–3292, 2007.
- [40] P. Strata and R. Harvey. Dale’s principle. *Brain Res. Bull.*, 50:34950, 1999.
- [41] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.
- [42] T. Toyozumi, J.-P. Pfister, K. Aihara, and W. Gerstner. Optimality model of unsupervised spike-timing dependent plasticity: Synaptic memory and weight distribution. *Neural Computation*, 19:639–671, 2007.
- [43] Taro Toyozumi, Jean-Pascal Pfister, Kazuyuki Aihara, and Wulfram Gerstner. Generalized bienenstock-cooper-munro rule for spiking neurons that maximizes information transmission. *Proceedings of the National Academy of Science*, 102:5239–5244, 2005.
- [44] J. A. Tropp. Algorithms for simultaneous sparse approximation. part ii: Convex relaxation. *Sparse approximations in signal processing*, 86:589–602, 2006.
- [45] Joel A. Tropp. Greed is good: Algorithmic results for sparse approximation. *IEEE Trans. Inform. Theory*, 50:2231–2242, 2004.
- [46] Joel A. Tropp. Just relax: Convex programming methods for subset selection and sparse approximation. Technical report, Texas Institute for Computational Engineering and Sciences, 2004.
- [47] D. Verstraeten, B. Schrauwen, M. DHaene, and D. Stroobandt. An experimental unification of reservoir computing methods. *Neural Networks*, 20(3):391–403, 2007.
- [48] J.D. Victor. Spike train metrics. *Current Opinion in Neurobiology*, 15(5):585–592, 2005.
- [49] J.D. Victor and K.P. Purpura. Nature and precision of temporal coding in visual cortex: a metric-space analysis. *J. Neurophysiol*, 76:1310–1326, 1996.
- [50] T. Vi eville, D. Lingrand, and F. Gaspard. Implementing a multi-model estimation method. *The International Journal of Computer Vision*, 44(1), 2001.
- [51] T. Vi eville and O. Rochel. One step towards an abstract view of computation in spiking neural-networks. In *International Conf. on Cognitive and Neural Systems*, 2006.
- [52] Jiří Š ima and Jiří Sgall. On the nonlearnability of a single spiking neuron. *Neural Computation*, 17(12):2635–2647, 2005.



Centre de recherche INRIA Sophia Antipolis – Méditerranée
2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Centre de recherche INRIA Bordeaux – Sud Ouest : Domaine Universitaire - 351, cours de la Libération - 33405 Talence Cedex
Centre de recherche INRIA Grenoble – Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier
Centre de recherche INRIA Lille – Nord Europe : Parc Scientifique de la Haute Borne - 40, avenue Halley - 59650 Villeneuve d'Ascq
Centre de recherche INRIA Nancy – Grand Est : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex
Centre de recherche INRIA Paris – Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex
Centre de recherche INRIA Rennes – Bretagne Atlantique : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex
Centre de recherche INRIA Saclay – Île-de-France : Parc Orsay Université - ZAC des Vignes : 4, rue Jacques Monod - 91893 Orsay Cedex

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
<http://www.inria.fr>
ISSN 0249-6399