On the Relevance of the Shape of Postsynaptic Potentials for the Computational Power of Spiking Neurons

Wolfgang Maass* **Berthold Ruf***

Abstract

The firing of a neuron in a biological neural system causes in certain other neurons excitatory postsynaptic potential changes (EPSP's) that are not "rectangular", but have the form of a smooth hill. We prove in this article for a formal model of a network of spiking neurons, that the rising respectively declining segments of these EPSP's are in fact essential for the computational power of the model.

1 Introduction

Apparently all computations in biological neural systems are realized through sequences of firings of neurons as a result of incoming postsynaptic potentials, see e.g. (Kandel et al., 1991). Each firing of a neuron in a biological neural system causes excitatory or inhibitory postsynaptic potentials (EPSP's respectively IPSP's) in those other neurons to which it is connected by synapses. A neuron fires if the sum of its incoming postsynaptic potentials becomes larger than its current threshold (which depends on the time of its last previous firing). Recently one has also started to explore special-purpose hardware that carries out computations in the form of operations on streams of pulses (Murray et al., 1994, Watts, 1994). Nevertheless the principles and limitations of computations with spiking neurons are so far only poorly understood. One important task for the theoretical investigation of computations in formal models of spiking neurons is to find out, which aspects of the assumed model are *accidental* for its computational power, and which ones are *essential*. As part of this program we investigate in this article the effect of the assumed shape of postsynaptic potentials on the computational power of an SNN (which is a formal model for a spiking neuron network).

It has been shown in (Maass, 1994a, 1994b, 1995) that if the EPSP's increase linearly in some segment, and decrease linearly in some other segment (which is biologically not implausible), then such SNN's can carry out with a bounded number of spikes on phase-differences of oscillators the operations ADD, SUBTRACT, MULTIPLY(β) (i.e. multiplication with a constant β), and

*Institute for Theoretical Computer Science, Technische Universitaet Graz, Klosterwiesgasse 32/2, A-8010 Graz, Austria. E-mail: {maass, bruf}@igi.tu-graz.ac.at

COMPARE. In this article we pursue the question whether the computational power of such models is significantly reduced, if they just exploit the "flat" (i.e. constant) parts of EPSP's and IPSP's, and make no use of their increasing or decreasing segments (or if they operate with rectangular pulses, as suggested by related hardware models, see (Murray et al., 1994)). This question is studied under the background assumption that one may still exploit the non-constant segments of the threshold-functions (which model the absolute and relative refractory periods of the neurons).

It turns out that under these assumptions an SNN can still carry out computations (and even simulate in principle arbitrary Turing machines). However it can no longer carry out with any bounded number of spikes the computational operations ADD or SUBTRACT on pairs of phase-differences, nor the operation MULTIPLY(β). Instead, it can only add or subtract constants from a phase-difference (i.e. carry out the operations ADD(β), SUBTRACT(β')), and decide which of two phase-differences is larger (i.e. carry out the operation COMPARE). Furthermore we show that for computations on bit strings such SNN can no longer carry out basic pattern matching operations in polynomial time.

2 Definitions

For the sake of completeness we recall here the precise definition of the SNN-model (see Maass 1994a, 1994b, for a detailed discussion). We set $\mathbf{R}^+ := \{x \in \mathbf{R} : x \geq 0\}$.

Definition of a Spiking Neuron Network (SNN): *An SNN $\mathcal{N}$ consists of a **finite** directed graph $\langle V, E \rangle$ (we refer to the elements of V as "neurons" and to the elements of E as "synapses"), a subset $V_{in} \subseteq V$ of input neurons, a subset $V_{out} \subseteq V$ of output neurons, a threshold-function $\Theta_v : \boldsymbol{R}^+ \rightarrow \boldsymbol{R} \cup \{\infty\}$ for each neuron $v \in V - V_{in}$, a response-function $\varepsilon_{u,v} : \boldsymbol{R}^+ \rightarrow \boldsymbol{R}$ and a weight $w_{u,v} \in \boldsymbol{R}^+$ for each synapse $\langle u, v \rangle \in E$. We assume that the firing of the input neurons $v \in V_{in}$ is determined from outside of $\mathcal{N}$, i.e. the sets $F_v \subseteq \boldsymbol{R}^+$ of firing times ("spike trains") for the neurons $v \in V_{in}$ are given as the input of $\mathcal{N}$.*

For a neuron $v \in V - V_{in}$ one defines its set F_v of firing times recursively. The first element of F_v is $\inf\{t \in \boldsymbol{R}^+ : P_v(t) \geq \Theta_v(0)\}$, and for any $s \in F_v$ the next larger element of F_v is $\inf\{t \in \boldsymbol{R}^+ : t > s \text{ and } P_v(t) \geq \Theta_v(t-s)\}$, where the potential function $P_v : \boldsymbol{R}^+ \rightarrow \boldsymbol{R}$ is defined by

$$P_v(t) := 0 + \sum_{u \,:\, \langle u,v \rangle \in E} \; \sum_{s \in F_u \,:\, s < t} w_{u,v} \cdot \varepsilon_{u,v}(t-s)$$

(the trivial summand 0 makes sure that $P_v(t)$ is welldefined even if $F_u = \phi$ for all $\langle u, v \rangle \in E$). The firing times ("spike trains") F_v of the output neurons $v \in V_{out}$ that result in this way are interpreted as the output of $\mathcal{N}$.

This formal model is essentially a noise-free version of the *spike response model* of (Gerstner, 1991, Gerstner, Ritz, van Hemmen, 1993). A response-function $\varepsilon_{u,v}(t-s)$ describes the potential change at the trigger-zone of neuron

v at time t, as a result of a firing of neuron u at time s. We assume in the following that each response-function $\varepsilon_{u,v}$ either satisfies $\varepsilon_{u,v}(x) \geq 0$ for all $x \in \mathbf{R}^+$ (EPSP) or $\varepsilon_{u,v}(x) \leq 0$ for all $x \in \mathbf{R}^+$ (IPSP). Furthermore we assume that $\varepsilon_{u,v}(0) = 0, \varepsilon_{u,v}(x) \neq 0$ for some x, and $\varepsilon_{u,v}(x) = 0$ for all sufficiently large x. With regard to the threshold-functions we assume for each $v \in V - V_{in}$ that there exists some $\tau_v > 0$ such that $\Theta_v(x) = \infty$ for all $x \in (0, \tau_v)$ ("absolute refractory period"), $0 < \Theta_v(0) < \infty$, $\Theta_v(x) \geq \Theta_v(0)$ for all $x \in \mathbf{R}^+$, and $\Theta_v(x) = \Theta_v(0)$ for all sufficiently large x. These assumptions are biologically realistic.

In biological models one usually assumes in addition that Θ_v decreases continuously in the interval $[\tau_v, \infty)$, and that the sign of the derivative of each response-function $\varepsilon_{u,v}$ changes only once. However these assumptions are not required in our formal models.

Definition of an N$^-$-RAM: *An N$^-$-RAM is a random access machine (RAM) with a constant number of registers, which receives as its input, stores in its registers, operates on, and outputs real numbers from some bounded interval $[-B, B]$. It can employ arbitrary finite programs that consist of instructions of the form ADD(β), SUBTRACT(β'), COMPARE, LOAD, STORE, HALT for arbitrary parameters $\beta, \beta' \ldots \in \boldsymbol{R}$. These instructions may involve direct and indirect addressing, as well as conditional jumps. The input (as well as the value of parameters $\beta, \beta', \ldots$) is given as the initial content of certain registers, and the output is given as the difference between the content of certain other registers when the machine halts. The complexity of a computation is evaluated according to the uniform-cost criterion, i.e. one unit is charged for each execution of an instruction.*

In (Maass, 1994b) we had considered the somewhat stronger model of an N-RAM, that can in addition execute the instructions ADD, SUBTRACT, and MULTIPLY(β). This machine model was shown in (Maass, 1994b, 1995) to be real-time equivalent to SNN's with piecewise linear response- and threshold functions, and to recurrent analog neural nets with piecewise linear activation functions.

One says that a computational model M' *simulates* another computational model M *in real-time*, if there exists a bound b such that every computation step of M can be simulated by at most b computation steps of M' (see (Maass, 1994b) for a more precise definition). In the case of an SNN M we count every spike that occurs in M as one computation step of M. One says that two classes S and S' of computational models are *real-time equivalent* if every $M \in S$ can be simulated by some $M' \in S'$ in real-time and if every $M' \in S'$ can be simulated by some $M \in S$ in real-time.

We call a response-function $\varepsilon_{u,v} : \mathbf{R}^+ \to \mathbf{R}$, or a threshold-function $\Theta_v : [\tau_v, \infty) \to \mathbf{R}$, piecewise constant (respectively piecewise monotone and continuous) if one can partition its domain into finitely many intervals I that are left-closed and right-open, so that this function is constant (respectively continuous, and non-decreasing or non-increasing) on each of these intervals I.

3 Computations with Numerical Inputs

Theorem 1: *SNN's with piecewise constant response-functions and piecewise monotone and continuous threshold-functions are for computations on real-valued input variables real-time equivalent to N$^-$-RAM's. Furthermore these SNN's are real-time equivalent to SNN's where both the response-function and the threshold function are piecewise constant.*

Idea of the ***proof*** *of Theorem 1:* For the simulation of such SNN by an N$^-$-RAM M one reserves for each neuron v a constant number of registers of M, that record the most recent firing times of v (as many as may be relevant for future firings of any neuron). M computes recursively all firing times t that occur for neurons in the SNN. In order to determine for some firing time t the *next* firing time of a neuron, M first computes for each neuron v the first time $t_v > t$ when v would fire *provided* that no other neuron fires within the time-interval (t, t_v). M then determines for which neuron (or neurons) v this time $t_v > t$ is minimal. Then by definition of t_v, it represents an actual firing time of neuron v, and this will be the next firing time after t. In order to keep all register contents that record a firing time within a bounded interval, M subtracts from these register contents a suitable constant C (and erases those among these registers whose content is $< C$) once one of them has reached a certain threshold. This is justified because only a bounded "time-window" of previous firings is relevant for determining future firings. ∎

Theorem 2: *No SNN (not even with arbitrary real-valued parameters) with piecewise constant response-functions and piecewise monotone and continuous threshold-functions is able to double an arbitrary phase-difference between two oscillators through computations that involve at most a bounded number of spikes. Furthermore such SNN cannot carry out with a bounded number of spikes any of the operations ADD, SUBTRACT, MULTIPLY(β) (for any $\beta > 0$ with $\beta \neq 1$) on phase-differences between oscillators.*

Idea of the ***proof*** *of Theorem 2:* We prove in fact a slightly stronger result: no such SNN can *decide* with a bounded number of spikes for given (arbitrarily small) phase differences $a, b, c \geq 0$ (between three oscillators and a "pacemaker") whether $a + b = c$, $a - b = c$, or $a \cdot \beta = c$ (for any fixed $\beta > 0$ with $\beta \neq 1$). If there would exist an SNN of the considered type that solves any of these decision-problems with a bounded number of spikes, then (by Theorem 1) there would also exist an N$^-$-RAM M that could solve such decision problems for arbitrarily small inputs $a, b, c \geq 0$ with a bounded number of (say: at most ℓ) computation steps. One can represent all possible computations of length $\leq \ell$ of this N$^-$-RAM M by a linear decision tree T of depth ℓ with some rather special form of linear decision at its branching nodes. For all inputs $\langle a, b, c\rangle \in \mathbf{R}^+$ for which the computation of M arrives at the same leaf of T, M will give the same answer regarding the question whether $a + b = c$ (respectively $a - b = c$, $a \cdot \beta = c$). If one takes a closer look at the special form of linear decisions which may arise at the branching nodes of T, one sees that these are all of the form "$r + \gamma \geq s + \delta$", where $r, s \in \{a, b, c\}$ and $\gamma, \delta \in \mathbf{R}^+$ and certain constants for this branching node. By choosing a, b, c sufficiently small and such that $0 \leq a \leq b \leq c$, one can make sure that each such comparison "$r + \gamma \geq s + \delta$" holds for *all* such a, b, c if $\gamma > \delta$, and holds for *no* such

a, b, c if $\gamma < \delta$ (if $\gamma = \delta$, its validity is predetermined by the pre-arranged order $a \leq b \leq c$ of a, b, c). Thus all these triples $\langle a, b, c \rangle$ will arrive at the same leaf of T, in spite of the fact that $a + b = c$ holds for some of these triples, and does not hold for others.

Since $a - b = c$ holds if and only if $a = b + c$, the preceding argument automatically also covers the case of decisions "$a - b = c$". The argument for "$a \cdot \beta = c$" is analogous. ∎

Remark: It was shown in (Maass, 1994b) that an SNN with linearly increasing and decreasing segments in the EPSP's *can* carry out all operations that were considered in Theorem 2 with a bounded number of spikes.

4 Computations with Boolean Inputs

We write $\{0,1\}^*$ for the set of all binary strings of finite length. In this section we consider the case where the SNN receives an input $\underline{w} \in \{0,1\}^*$ in an online fashion, i.e. bit by bit. We allow that the SNN signals through the firing of a designated neuron that it wants to receive the next input bit.

It was shown in (Maass, 1994a, 1994b) that one can construct a fixed SNN with rational parameters of finite size that can simulate *any* Turing machine (with arbitrarily large tape inscriptions) in real-time, by employing linearly increasing and decreasing segments of its response-functions. In contrast to that we show in Theorem 3 that the power of an SNN with rational parameters drops down to that of a DFA if it employs in its computations only "flat" pieces of its response-functions.

Theorem 3: *Any SNN with piecewise constant response-functions and piecewise linear threshold-functions with rational parameters can be simulated for boolean input that it receives in an online fashion by a deterministic finite automaton (DFA).*

Conversely, any DFA can be simulated in real-time by an SNN of this type (even with piecewise constant threshold functions). ∎

An SNN of the type considered in Theorem 3, but with *real-valued* parameters, is computationally more powerful than a DFA, since it can simulate any Turing-machine (see section 4 of (Maass, 1994b)). The following results imply however that *any* such simulation requires exponential time (see the comment after Corollary 5).

Theorem 4: *Assume that a language $L \subseteq \{0,1\}^*$ is accepted in polynomial time by an online SNN with arbitrary piecewise constant response-functions and arbitrary piecewise monotone and continuous threshold functions, whose definition may involve arbitrary real-valued parameters. Then for every $n \in \boldsymbol{N}$ the initial segment $L \cap \{0,1\}^{\leq n}$ of L can be accepted by some DFA with at most polynomially in n many states.*

Idea of the ***proof*** *of Theorem 4:* One can prove a corresponding version of Theorem 1 also for online boolean input. Hence it suffices to prove the claimed property for an arbitrary N^--RAM. ∎

Corollary 5: *No SNN of the type considered in Theorem 4 can decide in polynomial time whether $\underline{w} = \underline{\tilde{w}}$ for two sequentially presented bit strings*

$\underline{w}, \underline{\tilde{w}} \in \{0,1\}^n$ *(i.e.* $\underline{w}\,\underline{\tilde{w}}$*, or* $\underline{w}\#\underline{\tilde{w}}$ *with a separation marker* #*, is given as the input in an online fashion).*

Idea of the ***proof*** *of Corollary 5:* One just has to note that any DFA which carries out such a decision for any fixed n has to employ at least 2^n states to record the first half $\underline{w}$ of the input. ∎

The pattern matching task from Corollary 5 can obviously be carried out by a Turing machine in linear time. Hence no SNN of the type considered in Theorem 4 and Corollary 5 can simulate an arbitrary Turing machine in polynomial time (i.e. in such a way that the simulation of t Turing machine steps requires at most polynomially in t many spikes). This provides a strong contrast to the situation for SNN's with linearly increasing and decreasing segments in their response-functions, that can simulate any Turing machine in real-time (hence in linear time) even if all their parameters are rationals.

5 Conclusion

We have shown that both for numerical and for boolean inputs an SNN has much less computational power if it cannot exploit the increasing and decreasing segments of the excitatory postsynaptic potentials. In addition, Theorem 1 provides a complete characterization of the computational power of such SNN's in terms of a familiar (and easy to program) type of computational model.

Acknowledgement: We would like to thank Eric Allender and Pekka Orponen for helpful comments.

References

W. Gerstner. (1991) Associative memory in a network of "biological" neurons. *Advances in Neural Information Processing Systems, vol. 3, Morgan Kaufmann*: 84-90.

W. Gerstner, R. Ritz, J. L. van Hemmen. (1992) A biologically motivated and analytically soluble model of collective oscillations in the cortex. *Biol. Cybern.* 68: 363-374.

E. R. Kandel, J. H. Schwartz, T. M. Jessel. (1991) Principles of Neural Science. *Prentice-Hall.*

W. Maass. (1994a) On the computational complexity of networks of spiking neurons (extended abstract). *Advances in Neural Information Processing Systems, vol. 7, MIT-Press*, to appear.

W. Maass. (1994b) Lower bounds for the computational power of networks of spiking neurons. *Neural Computation*, to appear.

W. Maass. (1995) Analog computations on networks of spiking neurons (extended abstract). *Proceedings of the 7th Italian Workshop on Neural Nets*, to appear.

A. Murray, L. Tarassenko. (1994) Analogue Neural VLSI: A Pulse Stream Approach. *Chapman & Hall.*

L. Watts. (1994) Event-driven simulation of networks of spiking neurons. *Advances in Neural Information Processing Systems, vol. 6, Morgan Kaufmann*: 927-934.