DEVS Simulation of Spiking Neural Networks

Rene Mayrhofer, Michael Affenzeller,
Herbert Prähofer, Gerhard Höfer, Alexander Fried
Institute of Systems Science
Systems Theory and Information Technology
Johannes Kepler University
Altenbergerstrasse 69
A-4040 Linz - Austria
email: rene.mayrhofer@vianova.at

Abstract

This paper presents a new model for simulating Spiking Neural Networks using discrete event simulation which might possibly offer advantages concerning simulation speed and scalability. Spiking Neural Networks are considered as a new computation paradigm, representing an enhancement of Artificial Neural Networks by offering more flexibility and degree of freedom for modeling computational elements. Although this type of Neural Networks is rather new and there is not very much known about its features, it is clearly more powerful than its predecessor, being able to simulate Artificial Neural Networks in real time but also offering new computational elements that were not available previously. Unfortunately, the simulation of Spiking Neural Networks currently involves the use of continuous simulation techniques which do not scale easily to large networks with many neurons. Within the scope of the present paper, we discuss a new model for Spiking Neural Networks, which allows the use of discrete event simulation techniques, possibly offering enormous advantages in terms of simulation flexibility and scalability without restricting the qualitative computational power.

1 Introduction

Spiking Neural Networks (SNNs) are considered as the third generation of artificial neural networks and try to model the biological behaviour more closely than the last generation. Although the currently used Artificial Neural Networks (ANNs) which use the firing rate of neurons as their computational element have proven to be very powerful for specific kinds of problems, some major properties of biological neural networks are ignored. Through empirical evidence from biological experiments (see [Thorpe *et al.*, 1996]) it became clear that some effects and computations of the human brain can not be carried out by just using the firing rate of the respective neurons as information transmitter. Also the exact timing of single spikes has to carry at least some of the transmitted information.

Because of these limitations of the currently used model of neural networks, spiking neural networks have been developed. Instead of using the firing rate of neurons, where a lot of information from the spike train is eliminated by averaging over some time window, these new kinds of neural networks deliberately model the exact timing of the spikes emitted by the neurons. It has to be stated, that at the moment there is not very much known about the computational possibilities when working with the timing of single spikes - but an understanding of these computational mechanisms may also help in decoding the spike trains used by the brain or in explaining some of the effects encountered in biological neural systems, such as synchronization and oscillation.

Currently the ways of simulating such SNNs differ heavily from the way ANNs are used in todays applications. This is caused by the immanent complex behaviour of spiking neurons compared to the relatively simple behaviour of sigmoidal neurons used in ANNs. Sigmoidal neurons usually only sum up the analog input values they receive and utilize a sigmoid function to compute their output value from this sum, while spiking neurons try to model the behaviour and structure of biological neurons as closely as possible. Therefore, spiking neurons are often simulated using the biologically very realistic Hodgkin-Huxley model, which uses differential equations to describe the shape of neuron potentials and generated spikes. These differential equations impose the use of continuous simulation techniques, where the values of mathematical functions are computed at simulation times with fixed or variable time steps. However, with this simulation technique, the inner states of neurons which are inactive at some simulation time also need to be computed, resulting in bad scalability of the simulation in terms of large networks with many neurons. A better simulation technique – concerning scalability – is the so-called discrete event simulation where computation is only necessary for simulation elements that change their state at the current simulation time. In large networks where only a small percentage of neurons is active at any given time, event simulation is expected to be faster by a order of magnitude of at least 10.

In this paper we will first present a general introduction into Spiking Neural Networks in chapter 2. Then we give a short description of discrete event simulation in chapter 3, also explaining the difference to continuous simulation. These chapters contain the basic terminology used in the definition of our new model in chapter 4, which explains the formal basis the simulation is built on. In chapter 5 we introduce the concept of input and output coding converters, while chapter 6 gives an overview over the simulation framework that has been developed. Finally, chapter 7 gives an outlook on how this new technique could possibly be used for advanced studies and practical applications.

2 Spiking Neural Networks

As already stated in the introduction, Artificial Neural Networks typically encode the firing rate of biological neurons as analog values which are used as input and output values of the neurons. However, there is growing empirical evidence [Thorpe *et al.*, 1996] for the importance of the timing of single spikes. It was shown that humans can process visual patterns in 150 msec, where about 10 processing levels (neuron layers) are involved and neurons in these regions of the human brain usually have a firing rate of less then 100 Hz. Therefore, since this would involve averaging over the firing times of at least 2 received spikes, the processing time available to each layer is not enough for estimating the firing rate of neurons. As a result of these observations it can be argued that the computation has to be carried out using only the information transmitted with the first spike that is fired by each layer.

In principle, there are just three different possibilities currently known in neurophysiology for encoding information in such spike trains:

- rate coding: The essential information of the spike train is encoded in the firing rates, which are determined by averaging over some time window.
- temporal coding: The timing of single spikes, i.e. the exact firing time, carries the encoded information.
- population coding: The information is encoded by the activity of different populations of neurons, i.e. the percentage of neurons concurrently active in a population.

It seems that, from a theoretical point of view, temporal coding is the most simple coding scheme and that implementations for simulation purposes can be done easily. Therefore, this paper mainly focuses on this coding scheme; but all of the developed principles and techniques can be applied directly to other coding schemes - only the network inputs and outputs have to be adapted (see section "Network inputs and outputs"), the inner network structure can remain unchanged. The investigation of the computational power of different coding schemes will be a topic for future research. All of the developed principles and algorithms allow for arbitrarily complex coding schemes.

Currently there are many mathematical models which can be used for modeling and simulating Spiking Neural Networks. These models range from the biologically very realistic Hodgkin-Huxley model, which uses differential equations to model the electrical properties of neurons via the integrate-and-fire model to the rather simple spike-and-response model. However, all of these models are using spike timings as their primary code for computations and, furthermore, they are usually simulated continuously with fixed timesteps because of the representation of some mathematical functions in the model, especially the function modeling the post synaptic potential generated by synapses. The post synaptic potentials are the changes induced to the neuron potential of their subsequent neuron (the post synaptic neuron) generated by synapses whenever they receive a spike (from the pre synaptic neuron). Figure 1 shows the shape of a spike (action potential) and an excitatory and an inhibitory post synaptic potential (EPSP, IPSP).

In this paper we are using the following definition of spiking neural networks, directly cited from [Maass, 1996]:

An SNN 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*
- *for each neuron $v \in V - V_{in}$ a threshold function $\Theta_v : R^+ \to R \cup \{\infty\}$*
- *for each synapse $\langle u, v\rangle \in E$ a response function $\varepsilon_{u,v} : R^+ \to R$ and a weight $w_{u,v} \in R^+$*

We assume that the firing of input neurons $v \in V_{in}$ is determined from outside of the SNN, i.e. the sets $F_v \subseteq R^+$ of firing times ("spike trains") for the neurons $v \in V_{in}$ are given as the input of the SNN.

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

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

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 the SNN.

The complexity of a computation in a SNN is evaluated by counting each spike as a computation step.

This model represents a rather general definition of SNNs and is related to biological neurons as follows: the shape of a post synaptic potential (PSP) caused

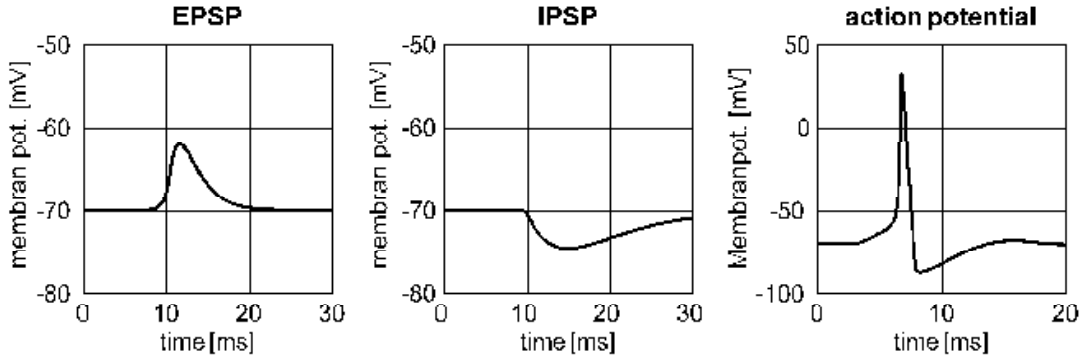

Figure 1: Post synaptic potentials and action potential, picture taken from http://www.cis.tugraz.at/igi/tnatschl/online/3rd_gen_ger/node1.html

by an incoming spike from neuron u is described by the response function $\varepsilon_{u,v}$, where $\varepsilon_{u,v}(t) = 0$ for $t \in [0, d_{u,v}]$ with $d_{u,v}$ modeling the delay between the generation of the action potential and the time when the resulting PSP starts to influence the potential of neuron v. A more detailed description of the features of this model can be found in [Ruf, 1998].

3 Discrete event simulation

Event simulation is a technique for simulating systems which are characterized by well-defined events. In contrast to continuous simulation, where the system states are calculated for specific simulation time points (either with fixed or with variable time steps), event simulation calculates the simulation time points when the system states reach specific values. Therefore, an event simulation advances from one simulation time point to the next whenever there are no more events to be handled at the current simulation time. The next simulation time is the time of the first event that has been scheduled for execution. For systems with a large quantity of different state variables and a low average change rate of these states, event simulation can offer obvious advantages in simulation speed and scalability. This is mainly because the exact trajectories between two relevant state values do not have to be calculated.

4 Model

During the development of a simulation framework, a new model of spiking neural networks was developed. This new model has been designed especially for discrete event simulation. It is based on the integrate-and-fire model and the mathematical definition of spiking neural networks introduced in [Maass, 1996] and cited in the section "Spiking Neural Networks", but does not use continuous functions for modeling post synaptic potentials and neuron potentials. Instead of this, we use piecewise linear functions, which offer the possibility to exactly and easily calculate the firing times of neurons using their potential and threshold functions. The neuron will fire whenever the value of its potential is equal to or higher than the value of its threshold, so calculating the next firing time is equal to calculating an intersection between two piecewise linear functions, which can be done efficiently. It has been shown in [Ruf, 1998] that spiking neural networks using piecewise linear functions are real-time equivalent to a Turing machine, i.e. the simulation of one computation step of a Turing machine by a SNN constructed especially for this purpose only takes a fixed amount of computation steps in the SNN. Therefore, the use of piecewise linear functions as synaptic response functions and potential and threshold functions describing the inner neuron state should not effect the qualitative computational power of SNNs negatively. Nevertheless, simulations will make it possible to study the quantitative influence of the shape of post synaptic potentials on the computational efficiency of spiking neural networks. Through the use of piecewise linear functions, our model makes it possible to approximate the shape of different functions as closely as needed or wanted in different applications. Additionally there is a gain in flexibility because there is no restriction to mathematical functions with a closed representation. Currently, studying the effects of different shapes of synaptic response functions is difficult because closed representations of these functions have to be found for the numeric integration techniques used in continuous simulations to work. Within our new model, this restriction no longer exists, offering neurobiologists more degrees of freedom in their study of neural models.

Furthermore, this novel model treats neurons and synapses as active elements and uses single spike events for modeling the spikes a neuron emits, only indicating the exact firing times but ignoring the shape of spikes (which are also not modeled in the widely used integrate-and-fire model). Because neurons are not coupled directly but only via synapses to each other, it are the synapses that receive the spike events sent by the neurons. Each synapse will receive spike events from only one neuron (the pre synaptic neuron) and will only have one neuron to send events to (the post synaptic neuron). Therefore, the network will form a bipartite graph of neurons and synapses with synapses being positioned between neurons as shown

in Fig. 2. In this paper we will use the convention that a neuron v receives its input from neurons $u = 1, ..., n$, which send their spike events to the respective input synapses of neuron v.

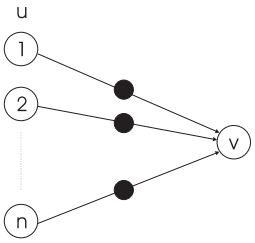

Figure 2: The input synapses of a neuron v receive spike events from neurons $u = 1, ..., n$, therefore generating the input values of neuron v.

Those synapses $\langle u, v\rangle$ are, upon the receipt of a spike event from their pre synaptic neuron u, also responsible for generating the post synaptic potentials in the form of piecewise linear functions $\varepsilon_{u,v}(t)$ and forwarding them to their post synaptic neuron v, scaled by their specific synaptic weight $w_{u,v}$ and deferred by their synaptic delay $d_{u,v}$.

Fig. 3 shows the input and output of a synapse which is generating an excitatory post synaptic potential due to the receipt of a spike. After a neuron v receives a new post synaptic potential from one of its input synapses $\langle u, v\rangle$, it merges it with its currently valid potential $P_v(t)$ and recalculates if and when the next firing occurs by calculating the time when the potential function $P_v(t)$ intersects with the threshold function $\Theta_v(t)$. The neuron threshold is also modeled as a time dependent function instead of a constant value to prevent a neuron from firing again instantly afterwards because the potential $P_v(t)$ might still be higher than the threshold. Therefore, the threshold function $\Theta_v(t)$ is defined to be infinite for some time $\tau_{v,ref}$ (the so-called absolute refraction period of the neuron) after the neuron v has fired, effectively preventing it from firing until the threshold again has a finite value. After this absolute refraction period ends at time $\tau_{v,ref}$, the neuron enters the relative refraction period, during which the threshold function rapidly approaches its constant value $\Theta_v(0)$, which by definition is reached at time $\tau_{v,end}$. In Fig. 4 the firing of a neuron v due to the intersection of the potential and the threshold function is shown.

For efficient handling and computations we use a special optimization by not representing the piecewise linear functions by a list of time/value tuples for the function points but by a list of time/gradient tuples for the function segments. Furthermore, we defined the functions to start and end at a value of zero. Therefore, a piecewise linear function $f(t)$ is defined as follows: Between the time points s_i and s_{i+1} the function has a constant gradient λ_i, whereas $s_0 := 0$, $f(0) := 0$, $f(s_{m+1}) := 0$, $\lambda_0 := 0$ and $\lambda_{m+1} := 0$ for m being the number of gradient changes in $f(t)$. Only the segments $\langle s_i, \lambda_i\rangle$ (for $i = 1, ..., m$) need to be specified to completely define $f(t)$, because λ_{m+1} is defined to be zero and s_{m+1} (the time when the function again reaches a value of 0) can be computed from the other values. Fig. 3 shows the shape of such a piecewise linear function with 5 segments.

This makes it possible to represent a piecewise linear function consisting of N segments with N time/gradient tuples and also offers a wide range of possible optimizations in the simulation, like using an extended but very fast merge-sort algorithm for combining a newly received post synaptic potential with the current neuron potential which completely eliminates the need to sum up over the neuron inputs as it has to be done in the standard integrate-and-fire model.

We can now define the synapse response function $\varepsilon_{u,v}(t)$ as

$$\varepsilon_{u,v}(t) := \sum_{i=1}^{k_{u,v}-1} (s_{u,v,i+1} - s_{u,v,i}) \cdot \lambda_{u,v,i} + (t - s_{u,v,k_{u,v}}) \cdot \lambda_{u,v,k_{u,v}}$$

and $\varepsilon_{u,v}(0) := 0$, whereas $k_{u,v} := \max_i(s_{u,v,i} < t)$ (the index of the last gradient change before the current time t). Therefore the current gradient of $\varepsilon_{u,v}(t)$ is denoted by $\lambda_{u,v,k_{u,v}}$. If, after each gradient change, we calculate the function value $\varepsilon_{u,v}(s_{u,v,k_{u,v}})$ immediately before this change, then we can use the recursive form

$$\varepsilon_{u,v}(t) = \varepsilon_{u,v}(s_{u,v,k_{u,v}}) + (t - s_{u,v,k_{u,v}}) \cdot \lambda_{u,v,k_{u,v}}.$$

The assumption that $\varepsilon_{u,v}(t) = 0$ for $t \in [0, d_{u,v}]$, i.e. the initial synaptic delay between the receipt of a spike from neuron u until the post synaptic potential starts changing the potential of neuron v, can be modeled easily by choosing $s_{u,v,1} = d_{u,v}$ ($\lambda_{u,v,0}$, the gradient within the interval $[0, s_{u,v,1}]$ is 0 by definition).

The potential of a neuron can now be defined as

$$P_v(t) := \sum_{\substack{u \in V \\ \langle u,v\rangle \in E}} \sum_{\substack{s_u \in F_u \\ s_u < t}} w_{u,v} \cdot \varepsilon_{u,v}(t - s_u)$$

with $P_v(0) := 0$, where V specifies the set of neurons, E the set of synapses and F_u the set of firing times of the neuron u. Therefore the potential $P_v(t)$ is defined as the linear sum of all incoming PSPs caused by spikes from the neurons u, whereas multiple PSPs from a single synapse $\langle u, v\rangle$, started at times $s_u \in F_u$, can be active (the response function $\varepsilon_{u,v}(t - s_u)$ has not reached 0 again) concurrently.

The threshold function $\Theta_v(t)$ is also defined as a piecewise linear function as

$$\Theta_v(t) := \begin{cases} \infty & if \quad \max_i(s_{v,i}) \le t < \max_i(s_{v,i}) + \tau_{v,ref} \\ \Theta_{v,ref} & \\ -\sum_{i=1}^{l_v-1}(s_{\Theta_v,i+1} - s_{\Theta_v,i}) \cdot \lambda_{\Theta_v,i} & if \quad \max_i(s_{v,i}) + \tau_{v,ref} \le t \\ +(t - s_{\Theta_v,l}) \cdot \lambda_{\Theta_v,l} & < \max_i(s_{v,i}) + \tau_{v,ref} + \tau_{v,end} \\ \Theta_v(0) & otherwise \end{cases}$$

where $\Theta_v(0)$ is some constant value, $s_{v,i}$ specifies the firing times of neuron v, $\Theta_{v,ref}$ the initial value of $\Theta_v(\max_i(s_{v,i}) + \tau_{v,ref})$ after the absolute refraction

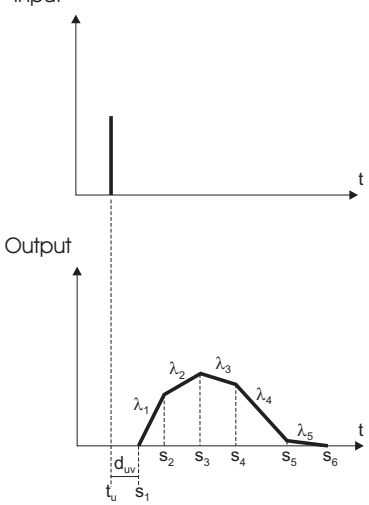

Figure 3: Synapse functioning: The synapse generates an excitatory post synaptic potential (EPSP) after the receipt of a spike event.

period with $\Theta_v(t) = \infty$, l_v the number of gradient changes and $\lambda_{\Theta_v,i}$ the gradients within the intervals $[s_{\Theta_v,i}, s_{\Theta_v,i+1}]$ (see Fig.4).

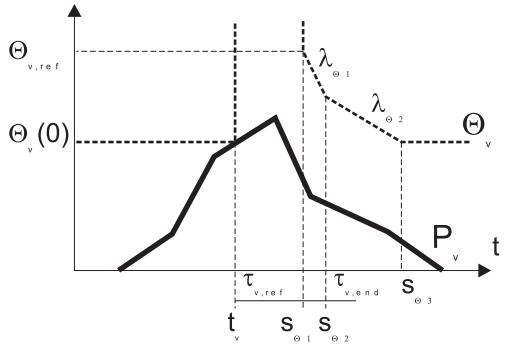

Figure 4: Neuron potential intersecting with the threshold function

The goal of the event simulation is now to calculate the simulation times s_v (in Fig. 4 specified as t_v for better readability) at which the neuron v will fire.

5 Network inputs and outputs

For a SNN to prove useful in practical applications, there is a need for well-known coding schemes at the inputs and outputs of the network. The spike coding scheme, transporting information within the neural network on the firing times of spikes, seems to offer many benefits in terms of computation power and compact coding of information. Nevertheless, at the moment there exist practically no sensors or actors utilizing this coding scheme so it can not be used for interacting with the real world. Therefore, input and output converters which transform the external information coding to spike trains at the SNN input and the spike trains to the external coding at the SNN output are needed. This effectively makes the SNN a black box of which the inner functioning does not have to be known.

Additionally, these converters offer more flexibility because the external and internal information coding schemes are decoupled. Ongoing research suggests the use of different spike coding schemes, such as temporal coding, rate coding, population coding or much more complex coding schemes (like fractal coding). With our simulation framework, this is possible without having to change the external representation of information. It also eases the use of different external information codings, e.g. when adding additional sensors or actors for interaction with the environment.

Currently input and output converters between analog vectors outside the SNN and temporal coding as inner coding scheme have been specified and implemented. Temporal coding is defined as follows: The input neurons u fire at simulation times $s_u = T_{input} - d \cdot x_u$ where x_u specifies the analog value that should be coded, T_{input} a reference time marking the input value zero, and d a common scaling factor. It has been shown that this coding scheme can be used easily to detect equality or near-equality of values (see [Maass, 1997b]), simulate McCulloch-Pitts neurons (see [Maass, 1999]), compute radial basis functions (RBFs, see [Hopfield, 1995]) and weighted sums (see [Ruf, 1998]) as well as simulating sigmoidal neurons (also [Ruf, 1998]) and approximating any continuous function (see [Ruf, 1998], [Maass, 1999] and [Maass, 1997a]).

6 Implementation

At the moment, a Java framework for DEVS simulation of spiking neural networks is developed and the core parts concerning the structure of the neural networks and handling of events including calculating firing times of neurons are nearly completed. Using event simulation, we expect an enormous increase in simulation speed and flexibility compared to the currently used continuous simulation techniques, making it possible to simulate large networks of spiking neurons and thus allowing studies of advanced networks holding a lot more information than networks with only a small number of neurons. Additionally, we expect the simulation speedup gained by event simulation to increase with the size of a network, because typically the percentage of concurrently active neurons will decrease as the number of neurons in the network increases. With this framework it will be possible to simulate and visualize Spiking Neural Networks with piecewise linear functions and to experiment effectively with new signal forms and learning methods in the context of computational theory as well as from the biological viewpoint. Although the use of other function types is possible in the core parts of the framework, the current implementation only covers piecewise linear ones, which seem to be sufficient for reproducing most computational effects of biological neural networks. Nevertheless, piecewise linear

functions also allow an approximation of an function shapes as closely as needed or wanted, even if there are more efficient approximations for some function types.

7 Outlook

Due to the use of discrete event simulation techniques it might be possible to simulate large-scale neural networks with thousands of neurons on standard workstations, making it possible to investigate the influence of different function types on the computational power of spiking neurons. However, future research has to show that the assumptions we made hold true and that effects of biological neural networks (such as synchronization between quasi-chaotically firing neurons) are reproduceable with our model.

Another direction for future research will be the reuse of components within SNNs. With this new event based model, some computational elements built of spiking neurons (e.g. a visual 2D filter modeled after the cuneate nucleus in the human brain) could be seen as black boxes with defined input and output behaviour and used within a larger neural network. By this way, powerful neural networks might be composed of rather simple components, on one hand enabling reuse of elements and on the other hand making it possible to construct spiking neural networks with well-known system theoretical methods.

References

[Hopfield, 1995] J.J. Hopfield. Pattern recognition computation using action potential timing for stimulus representation. *Nature*, 3676:33–36, 1995.

[Maass, 1996] Wolfgang Maass. Lower bounds for the computational power of networks of spiking neurons. *Neural Computation*, 8:1–40, 1996.

[Maass, 1997a] Wolfgang Maass. Fast sigmoidal networks via spiking neurons, neural computation. *Neural Computation*, 3:105–119, 1997.

[Maass, 1997b] Wolfgang Maass. Networks of spiking neurons: the third generation of neural network models. *Neural Networks*, 10(9):1659–1671, 1997.

[Maass, 1999] Wolfgang Maass. *Paradigms for computing with Spiking Neurons, Models of Neural Networks*. Springer, 1999.

[Ruf, 1998] Berthold Ruf. *Computing and Learning with Spiking Neurons - Theory and Simulations*. PhD thesis, Graz University of Technology, 1998.

[Thorpe *et al.*, 1996] S.J. Thorpe, D. Fize, and C. Marlot. Speed of precessing in the human visual system. *Nature*, 381:520–522, 1996.